הרצאה 1: הקדמה

הקורס שלפניכם עוסק בתורת האלגוריתמים המתמטית. במהלך הקורס:

1. נגדיר את המושגים הבסיסיים הנדרשים לעיסוק בתחום.
2. נלמד לבטא אלגוריתמים בעזרת קוד דמה (Pseudo Code).
3. נלמד איך להוכיח נכונות של אלגוריתמים ואיך לנתח את הסיבוכיות שלהם.
4. נלמד מספר פרדיגמות (שיטות) לפיתוח אלגוריתמים.
5. נלמד מספר רב של אלגוריתמים בסיסיים.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

המושגים הבסיסיים

כל תורה מתמטית מתחילה בהצגת המושגים הבסיסיים בהם היא משתמשת וזוהי מטרת ההרצאה שלנו. המושגים הבסיסיים שעלינו להכיר ולהבין הם:

1. בעיה חישובית.
2. אלגוריתם
3. נכונות של אלגוריתם.
4. סיבוכיות של אלגוריתם.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

איך מגדירים בעיה חישובית

בעיה חישובית היא בעיה שניתנת לפתרון על ידי אלגוריתם, או תכנית מחשב.

אם ננסה להשתמש בהגדרה זו, אנו עלולים להגיע להגדרה מעגלית בה נגדיר את המושג "בעיה חישובית" באמצעות המושג "אלגוריתם", ואחר כך נגדיר את המושג "אלגוריתם" באמצעות המושג "בעיה חישובית". להלן אנו מגדירים את המושג "בעיה חישובית" ללא שימוש במושג "אלגוריתם", אך אנו זוכרים כל העת שזוהי המוטיבציה להגדרה זו.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

דוגמה להגדרת בעיה חישובית

בטרם נגדיר "בעיה חישובית" באופן כללי, נביא כדוגמה את הגדרת הבעיה החישובית "חישוב סכום של מערך".

דוגמה 1.1: הגדרת סכום של מערך

קבוצת הקלטים

קבוצת המערכים שאיבריהם מספרים.

קבוצת הפלטים

קבוצת המספרים.

תיאור הפלט

לכל מערך קלט A, ובו n איברים, הפלט המתאים הוא $\sum_{i=1}^{n} A[i]$.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

<u>הערות</u>

1. בקורס זה המושגים **<u>קלט</u>** **<u>ופלט</u>** מתייחסים למידע שהאלגוריתם מקבל ומחזיר בהתאמה, ולא לדו שיח עם המשתמש בעזרת מקלדת וטרמינל.
2. שם המערך (ובאופן כללי, כל שם של משתנה או מבנה נתונים) אינו משמעותי, אפשר לבחור כל שם אחר.
3. הבעיה היא פשוטה במיוחד כדי שההדגמה תהיה פשוטה. בהמשך נציג אלגוריתם לפתרון הבעיה ונוכיח את נכונותו.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

<u>הגדרת המושג "בעיה חישובית"</u>

בעיה חישובית P מסומנת על ידי השלשה $P = \langle In, Out, f \rangle$, כאשר:

<u>קבוצת הקלט – In</u>

קבוצה שאיבריה הם כל הקלטים,

<u>קבוצת הפלט - Out</u>

קבוצה המכילה את כל הפלטים.

<u>פונקצית ההתאמה - f</u>

פונקציה $f : In \rightarrow Out$. זוהי פונקציה המתאימה לכל איבר בקבוצת הקלט, את הפלט המתאים..

© כל הזכויות שמורות לפרופסור עמוס ישראלי

<u>דוגמה 1.2: הגדרת בעית מיון מערך</u>

<u>קבוצת הקלט</u>

מערכים שאיבריהם מספרים.

<u>קבוצת הפלט</u>

מערכים שאיבריהם מספרים.

<u>התאמת הפלט לקלט</u>

לכל מערך קלט A הפלט המתאים הוא מערך ממוין $SortA$ שאיבריו הם תמורה של איברי המערך A, כלומר אותם האיברים, אך אולי בסדר שונה.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

<u>המושג "אלגוריתם"</u>

אלגוריתם הוא תיאור של שיטת עבודה הניתנת לביצוע על ידי תכנה, לפתרון בעיה חישובית (שכבר הגדרנו).

אנו מעוניינים באלגוריתמים שיפתרו בעיות חישוביות כגון:

1. הכפלת זוג מספרים.
2. מיון מערך מספרים.
3. כפל מטריצות.

כל אלגוריתם, מתייחס לבעיה חישובית מסוימת.

זוהי שיטת עבודה המקבלת איבר מקבוצת הקלטים, ומחשבת את הפלט המתאים.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

הגדרת המושג "אלגוריתם לפתרון בעיה חישובית P"

נתונה בעיה חישובית $P=\langle In,Out,f\rangle$.
"אלגוריתם לפתרון הבעיה P" הוא שיטה חישובית Al בעלת התכונות הבאות:

1. Al מקבל כקלט כל איבר $i\in In$.
2. Al מסיים כל חישוב המקבל כקלט כל איבר i, $i\in In$.
3. לכל איבר בקלט $i\in In$, חישוב Al על קלט כלשהו $i\in In$, מחזיר כפלט את $f(i)$.

תיאור אלגוריתמים

כדי להתייחס לאלגוריתם, נחוצה דרך אחידה לתיאורם. מאחר שכל אלגוריתם מהווה בסיס לכתיבת אפליקציה, הדרך ה"טבעית" לבטא אלגוריתמים היא באמצעות תכנה.
מצד שני, כל תכנה מכילה פרטים טכניים רבים, אשר מקשים על הבנת עקרונות האלגוריתם.
בבואנו לתאר אלגוריתם, עלינו להעדיף את הבהירות הנדרשת להבנה אנושית, על פני הדיוק הטכני הנדרש מתכנה ניתנת לביצוע.
הדרך שנבחרה לתיאור אלגוריתמים היא שימוש בקוד דמה (Pseudo Code).

יסודות קוד דמה

קוד דמה הוא שפה המשמשת לתיאור לא פורמלי של אלגוריתמים.
אלגוריתם המתואר באמצעות קוד דמה הוא סדרה של הוראות, כאשר כל הוראה מבוטאה באופן חפשי, על ידי ערוב של כתיב דמוי שפת תכנות, כתיב מתמטי וכתיב חפשי.
מטרת השימוש בקוד דמה היא ליצור תיאור חד משמעי של האלגוריתם אשר מאפשר יישום האלגוריתם באמצעות תוכנה.
אנו מניחים כי קורא מיומן, יכול לעקוב אחר ביצוע האלגוריתם באופן חד משמעי.

כותרת האלגוריתם

השורה הראשונה בקוד מכונה **כותרת האלגוריתם** והיא מכילה את שם האלגוריתם ואת הקלט הארגומנטים) עליו האלגוריתם מתבצע.

הוראות בקרה

הוראות בקרה מבוטאות בעזרת מבנים מקובלים כגון:

1. לתיאור טססים נשתמש במבנה
if then else.
2. לתיאור לולאות נשתמש במבנה
for …endfor
while … endwhile
או
Do … while
כמקובל בשפות תכנות כגון Java.
3. לציון הפלט נשתמש בהוראת
$Out(...)$
כאשר הפלט מצויין בין הסוגריים.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

דוגמה: אלגוריתם לחישוב סכום של מערך

נציג כעת את האלגוריתם Sum אשר פותר את הבעיה "חישוב סכום של מערך":

```
Sum(A,n)
  sum ← 0
  for i ← 1 to n
    sum ← sum + A[i]
  end(for)
  out(sum)
```

אלגוריתם 1.3: אלגוריתם לחישוב סכום של מערך

© כל הזכויות שמורות לפרופסור עמוס ישראלי

הערות לאלגוריתם

1. כבר צינו כי הבעיה המוצגת היא פשוטה והאלגוריתם הפותר אותה הם פשוטים במיוחד ומטרתם להדגים את עקרונות הטיפול הפורמלי באלגוריתם.
2. הארגומנט n מייצג את מספר האיברים במערך A. בדרך כלל, לא נציין את המספר הזה.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

הוכחת נכונות של אלגוריתם

נניח שקיבלנו קטע של קוד דמה ונטען שזהו אלגוריתם לפתרון בעיה P. איך נוכל לוודא שהטענה הזו נכונה?
הרעיון הראשון שעולה הוא לממש את האלגוריתם באמצעות שפת התכנה החביבה עלינו ולהריץ אותו עם קלטים שונים מן הקבוצה. לצערנו, השיטה הזו לא יכולה להניב תוצאה טובה.
הסיבה היא שקבוצת הקלטים היא אין סופית, ואילו השיטה שהצענו, מתאימה רק לבדיקת מספר סופי של קלטים.
מסיבה זו, נצטרך להוכיח נכונות של אלגוריתמים אך ורק בדרכים מתמטיות, כפי שמפורט להלן.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

הגדרת נכונות של אלגוריתם

הגדרה 1.4

אלגוריתם Al הוא **נכון** אם לכל **קלט חוקי** האלגוריתם מחשב את **הפלט המתאים**.

נדגיש כי אם קיים אפילו פלט יחיד עבורו האלגוריתם נכשל בחישוב הפלט, האלגוריתם הוא **שגוי**.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

הוכחת נכונות של אלגוריתמים

נניח כי P בעיה חישובית: ו - Al הוא אלגוריתם לפתרון P.

כדי להוכיח את נכונותו של Al יש להוכיח:

1. **סיום -** לכל קלט $i \in In$, חישוב Al על i **מסתיים** תוך פרק זמן סופי.
2. **נכונות פלט -** לכל קלט $i \in In$, חישוב Al על In מסתיים עם הפלט $f(i)$.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

הוכחת סיום

כדי להוכיח סיום של תוכנית איטרטיבית, **מספיק** לבדוק כי **מספר המעברים בכל אחת מן הלולאות הוא חסום**. בדרך כלל, הוכחת סיום היא פשוטה ולא נתעכב עליה.

במקרים נדירים הוכחת הסיום היא קשה מאוד. לדוגמה, השיטה phi (אלגוריתם 1.5) היא שיטה פשוטה ביותר. לכל ערך נתון של הארגומנט n, אפשר להריץ את השיטה והביצוע תמיד מסתיים, אך עד כה לא הוכח כי השיטה עוצרת לכל המספרים הטבעיים.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

הוכחת סיום (המשך)

```
phy(n)
  Do forever
    if (n = 1) return
    if (even n) n ← n/2
        else n ← 3n+1
     endif
  end do
```

אלגוריתם 1.5: הפונקציה *phi*

© כל הזכויות שמורות לפרופסור עמוס ישראלי

<u>הוכחת נכונות פלט</u>

הוכחת **נכונות הפלט** היא לרוב מסובכת ודורשת מיומנות מתמטית. לעתים קרובות, הנכונות **ברורה אינטואיטיבית**, אולם ההוכחה המתמטית הפורמלית היא קשה וטכנית.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

<u>הכנות להוכחת הנכונות</u>

בטרם ניגש להוכחת נכונות האלגוריתם עלינו להבין באופן אינטואיטיבי את דרך פעולתו ולנסות לזהות שגיאות אפשריות. הדרך להשיג משימה זו היא **<u>לבצע הרצות נסיון.</u>**

<u>הרצות נסיון</u>

1. בוחרים מספר קלטים לדוגמה. כל קלט כזה, צריך לייצג בעיה פוטנציאלית באלגוריתם. לדוגמה:
   - מערך עם איבר יחיד
   - מערך עם איברים שליליים.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

<u>הרצות נסיון (המשך)</u>

2. מריצים את האלגוריתם על כל אחד מן הקלטים לדוגמה ומוודאים באופן בלתי תלוי שהאלגוריתם מניב פלט נכון.
3. במקרה שהאלגוריתם המוצע אינו נכון, יש סיכוי טוב שהרצת הנסיון תניב תוצאה שגויה ובכך תחסוך את המאמץ בהוכחת אלגוריתם שגוי.
4. זיהוי השגיאה בדרך כלל מסייע בתיקון הפגם באלגוריתם.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

<u>הבנה אינטואיטיבית של האלגוריתם</u>

נניח כי ביצענו מספר הרצות ניסיון ולא זיהינו שגיאה באלגוריתם.
במצב זה, עלינו לנסות ולהבין באופן אינטואיטיבי את דרך פעולת האלגוריתם.
הכוונה היא לנסות ולהבין את תפקידם של המשתנים המופיעים בקוד ואת הדרך בה מחושב הפלט.
בדוגמה (הטריויאלית) שלנו, אנו רואים כי הפלט הוא המשתנה *sum*.
במקרה זה, ה"אינטואיציה" אומרת לנו כי לאחר המעבר בלולאת האלגוריתם בפעם ה-i, **ערכו של המשתנה** *sum* **שווה לסכום** i **האיברים הראשונים במערך הקלט**.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

הבנה אינטואיטיבית של האלגוריתם (המשך)

מכאן אפשר להסיק באופן מיידי כי: לאחר המעבר בלולאת האלגוריתם בפעם ה-n, **ערכו של המשתנה** sum **שווה לסכום כל אברי המערך**.

כעת נדגים איך הופכים את ההבנה האינטואיטיבית להוכחה פורמלית של נכונות האלגוריתם.

בניית משפט נכונות – תבנית הטענה

השלב הראשון בהוכחת הנכונות הוא בניית **משפט נכונות**.

כל משפט נכונות נבנית לפי התבנית הבאה:

תבנית משפט נכונות

לכל קלט $i \in In$, **חישוב** Al **על** i **מסתיים עם הפלט** $f(i)$.

כדי להפוך את התבנית למשפט נכונות מלא, יש **להחליף** את הביטוי i בתאור מדוייק של הקלט ואת הביטוי $f(i)$ בתאור מדוייק של הפלט.

דוגמא - בניית משפט נכונות

משפט 1.6: נכונות אלגוריתם *Sum*

לכל **מערך מספרים** A (תיאור הקלט), האלגוריתם Sum מחשב את **סכום איברי המערך** A (תיאור הפלט).

שימו לב

משפט הנכונות תלוי אך ורק בבעיה החישובית שהאלגוריתם פותר. ההתיחסות לאלגוריתם היא הזכרת שם האלגוריתם.

הוכחת נכונות של אלגוריתמים איטרטיביים (עם לולאה)

משפט הנכונות הוא **טענה מתמטית**. כמו כל טענה מתמטית אחרת, הטענה דורשת **הוכחה מתמטית**.

באלגוריתם איטרטיבי (אלגוריתם המכיל לולאה עלינו להשתמש באינדוקציה. אנו נוכיח את משפט הנכונות **באינדוקציה על מספר המעברים בלולאת האלגוריתם.**

דוגמא - הוכחת נכונות אלגוריתם *Sum*

לדוגמה נציג שוב את אלגוריתם *Sum* ונוכיח את נכונותו.

```
Sum(A,n)
  sum ← 0
  for i ← 1 to n
    sum ← sum + A[i]
  end(for)
out(sum)
```

אלגוריתם 1.3: האלגוריתם *Sum*

הוכחת סיום

מן הקוד נובע באופן מיידי כי האלגוריתם מסתיים לאחר n מעברים בלולאה.

משפט 1.7: נכונות אלגוריתם *Sum*

לכל **מערך מספרים** A האלגוריתם *Sum* מחשב את **סכום איברי המערך** A.

נוכיח את הטענה הבאה:

טענה

לכל i, $i=1,2,...$, כאשר בקרת האלגוריתם מגיעה לסוף הלולאה בפעם ה-i, מתקיים:

$$sum=\sum_{j=1}^{i}A[j]$$

שימו לב: לביטוי במסגרת קוראים **"שמורת הלולאה".**

הוכחת הטענה (באינדוקציה על המעברים בלולאה)

בסיס

עלינו להוכיח כי כאשר בקרת האלגוריתם מגיעה לסוף הלולאה, מתקיים $sum=A[1]$.

הוכחה

מיידית על ידי מעקב על ביצוע האלגוריתם.

שלב האינדוקציה

נניח כי לאחר $i-1$ מעברים בלולאה מתקיים

$$sum=\sum_{j=1}^{i-1}A[j]$$

הוכחת הטענה

במהלך המעבר ה-i בלולאה, מחברים למשתנה *sum* את $A[i]$. מכאן ומהנחת האינדוקציה נובע כי בתם המעבר ה-i בלולאה מתקיים:

$$sum=\sum_{j=1}^{i-1}A[j]+A[i]=\sum_{j=1}^{i}A[j]$$

מש"ל

<u>תזכורת:</u> הדוגמה היא **<u>פשוטה במיוחד</u>** ומשמשת להדגמת הטכניקה של ההוכחה באינדוקציה.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

<u>שיטת השמורה בשלושה שלבים</u>

נתאר כעת בפרוט את שלבי שיטת ההוכחה באינדוקציה על המעברים בלולאה.

<u>שלב 1: ניסוח השמורה</u>

מנסחים טענה אודות הקשר בין מספר המעברים בלולאה לבין ערך המשתנים. דורשים שהטענה הזו תתקיים (תהיה נכונה) בכל פעם שהאלגוריתם מגיע לנקודה מסוימת בלולאה.
לטענה קוראים **<u>שמורת הלולאה</u>**, ולנקודה שבה השמורה מתקיימת קוראים **<u>נקודת השמורה.</u>**

© כל הזכויות שמורות לפרופסור עמוס ישראלי

<u>שלב 2: הוכחת בסיס האינדוקציה</u>

מוכיחים על ידי מעקב אחרי ביצוע האלגוריתם כי כאשר בקרת האלגוריתם מגיעה לנקודת השמורה בפעם הראשונה, השמורה מתקיימת.

<u>שלב 3: שלב האינדוקציה</u>

מניחים כי השמורה נכונה כאשר בקרת האלגוריתם מגיעה לנקודת השמורה במעבר ה - i, ומוכיחים, **שוב בעזרת מעקב,** כי השמורה נכונה כאשר בקרת האלגוריתם מגיעה לנקודת השמורה במעבר ה - $i+1$.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

<u>שמורות חליפיות</u>

בדרך כלל, אפשר לבחור כמה שמורות חלופיות. לכל שמורה חלופית תתאים נקודת שמורה משלה:
לדוגמא: אפשר גם לבחור כשמורה את הטענה בתחילת המעבר ה - i בלולאת האלגוריתם, ($1 \le i \le n$), **לפני ביצוע הוראת ההשמה** מתקיים:

$$sum = \sum_{j=1}^{i-1} A[j]$$

© כל הזכויות שמורות לפרופסור עמוס ישראלי

שמורות חליפיות (המשך)

בתרגיל 1.1 אתם מתבקשים להוכיח את נכונות האלגוריתם *Sum* בעזרת השמורה החליפית.

באופן כללי, כדאי לבחור את השמורה בהוכחה, לפי פשטות ההוכחה המתקבלת.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

שמורות גרועות

יש אינסוף ביטויים אשר הם נכונים בכל פעם שבקרת התכנה מגיעה לנקודת השמורה, אך הם חסרי תועלת להוכחת הנכונות.

דוגמה לשמורה כזו היא

$1+2=3$.

זה ביטוי נכון, אך הוא אינו קושר בין הקלט לבין משתני התכנית או בין משתני התכנית לבין הפלט.

עוד שמורה גרועה היא

$1 \le i \le n$

בחירת השמורה הנכונה היא האתגר שבהוכחת הנכונות.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

סיבוכיות האלגוריתם

להשלמת הטיפול באלגוריתם *Sum* נציג את ניתוח הסיבוכיות שלו:

1. הלולאה מתבצעת בדיוק n פעמים.
2. בכל מעבר מתבצע מספר פעולות קבוע (Constant), כלומר מספר פעולות שאינו תלוי בגודל הקלט.

משתי נקודות אלה נובע כי הסיבוכיות היא לינארית, כלומר $\Theta(n)$.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

האלגוריתם *Max*: חישוב איבר מרבי במערך חד ממדי

נתון האלגוריתם *Max*

```
Max(A)
   M ← A[1]
   for i = 2 to n
      M ← max(M, A[i])
   end for
out(M)
```

אלגוריתם 1.8: האלגוריתם *Max*

בתרגיל 1.2 תתבקשו לנסח את הבעיה החישובית שהאלגוריתם פותר, להוכיח את נכונותו, ולנתח את הסיבוכיות שלו.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

האלגוריתם *Copy*: העתקת מערך חד ממדי

נתון האלגוריתם *Copy*

```
Copy(A)
   for i ← 1 to n
      C[i] ← A[i]
   end for
out(C)
```

אלגוריתם 1.9: האלגוריתם *Copy*

שימו לב: אלגוריתם 1.9 משתמש במערך C ללא הגדרה. נזכור שזהו קוד דמה. כל זמן שהסימונים מובנים, אין צורך בפורמליסטיקה מיותרת.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

בתרגיל 1.3 תתבקשו לנסח את הבעיה החישובית שהאלגוריתם *Copy* פותר, להוכיח את נכונותו, ולנתח את הסיבוכיות שלו.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

טיפול בלולאות מקוננות

עד כה התווינו את דרך הטיפול באלגוריתמים שיש בהם לולאה אחת. לסיום הרצאה זו, נדגים טיפול באלגוריתם *Max2d*.

אלגוריתם זה פותר את הבעיה החישובית הבאה:

בעיה 1.7:חישוב איבר מרבי במערך

קלט

מערך דו ממדי A שאיבריו מספרים.

פלט

איבר מרבי במערך A.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

אלגוריתם לחישוב איבר מכסימלי

```
Max2d(A)
   for i = 1 to n
      for j = 1 to n
         Line[j] ← A[i, j]
      end for
      B[i] ← Max(Line)
   end for
out(Max(B))
```

אלגוריתם 1.10: האלגוריתם *Max2d*

© כל הזכויות שמורות לפרופסור עמוס ישראלי

אלגוריתם *Max2d* - תיאור

נסמן את השורה ה-i של המערך A ב-$A[i]$. האלגוריתם מבצע n איטרציות (מעברים בלולאה): באיטרציה ה-i האלגוריתם מחשב את האיבר המרבי בשורה $A[i]$ ומכניס איבר זה אל $B[i]$ לאחר מכן האלגוריתם מפעיל את השיטה $Max(B)$ כדי לחשב את האיבר המרבי במערך B.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

הוכחת נכונות

אחת הדרכים להוכחת נכונות האלגוריתם היא להוכיח נכונות של אלגוריתם אחר השקול לאלגוריתם המקורי.

נתבונן באלגוריתם 1.11:

© כל הזכויות שמורות לפרופסור עמוס ישראלי

אלגוריתם שקול לחישוב איבר מכסימלי

$Max2d2(A)$
  for $i=1$ **to** n
    $Line \leftarrow Copy(A[i])$
    $B[i] \leftarrow Max(Line)$
  end for
out$(Max(B))$

אלגוריתם 1.11: האלגוריתם *Max2d2*

שימו לב: באלגוריתם זה, הסימון $A[i]$ מייצג את השורה ה-i במערך הקלט A.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

אלגוריתם שקול לחישוב איבר מכסימלי

בתרגיל 1.4 תתבקשו לנסח את הבעיה החישובית שהאלגוריתם $Max2d2$ פותר, להוכיח את נכונותו, ולנתח את הסיבוכיות שלו.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

פיתוח אלגוריתם כתהליך

פיתוח אלגוריתם הוא תהליך **מחזורי** (איטרטיבי):

- מתחילים בהצעה ראשונית.
- בכל מחזור (איטרציה) **בודקים את ההצעה הנוכחית** בעזרת הרצת דוגמאות.
- אם מזהים שגיאה באלגוריתם.
- מתקנים את השגיאה.

לאחר שלא מוצאים יותר שגיאות בדרך הזו, ניגשים להוכחת האלגוריתם.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

ניתוח זמן ריצה (חזרה)

זמן ריצה הוא הגורם החשוב ביותר שיש לבחון כאשר בוחרים אלגוריתם לפתרון בעייה חישובית כלשהי.
לאחר שהשתכנענו שאלגוריתם הוא נכון, יש לנתח את **סיבוכיות הזמן** שלו.
סיבוכיות הזמן של אלגוריתם, היא פונקציה מתמטית המאפשרת לנו לחזות את זמן הריצה של תוכנית המבוססת על האלגוריתם ולהשוות בין אלגוריתמים שונים.
בנספח נביא חזרה קצרה בנושא ניתוח סיבוכיות הזמן של אלגוריתמים.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

סיכום

בהצאה זו הגדרנו והדגמנו את המושגים הבסיסיים שישמשו אותנו במהלך הקורס והם:

1. בעיה חישובית.
2. אלגוריתם.
3. נכונות אלגוריתם.
4. סיבוכיות אלגוריתם.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

תרגילים

תרגיל 1.1: נתונה טענה:

$$sum = \sum_{j=1}^{i-1} A[j]$$

זהו נקודה בלולאה שבאלגוריתם *Sum* (אלגוריתם 1.7) שבה הטענה מהווה שמורה. השתמשו בשמורה הנתונה, כדי להוכיח את נכונות האלגוריתם.

תרגיל 1.2: נסחו את הבעיה החישובית שהאלגוריתם *Max* פותר.
הוכיחו את נכונות האלגוריתם *Max*, באינדוקציה על מספר המעברים בלולאה ונתחו את סיבוכיות האלגוריתם.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

<u>תרגיל 1.3:</u> נסחו את הבעיה החישובית שהאלגוריתם *Copy* פותר.
הוכיחו את נכונות האלגוריתם *Copy*, באינדוקציה על מספר המעברים בלולאה ונתחו את סיבוכיות האלגוריתם.

<u>תרגיל 1.4:</u> נסחו את הבעיה החישובית שהאלגוריתם *Max2d2* פותר.
הוכיחו את נכונות האלגוריתם *Copy*, באינדוקציה על מספר המעברים בלולאה ונתחו את סיבוכיות האלגוריתם.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

<u>נספח: הגדרת סיבוכיות הזמן של אלגוריתמים</u>

בתחילת הדיון למדוד מהם הגורמים המשפיעים על זמן הביצוע של תכנית לפתרון בעיה כלשהי. הגורמים החשובים הם:

1. יעילות הקידוד.
   (תרגום מאלגוריתם לתכנית)
2. יעילות המהדר.
   (תרגום משפת על לשפת מכונה)
3. יעילות המעבד.
4. יעילות האלגוריתם.

האם אתם מכירים גורם נוסף?

© כל הזכויות שמורות לפרופסור עמוס ישראלי

<u>מה מידת ההשפעה של כל גורם?</u>

1. יעילות הקידוד - כל שורה משורות האלגוריתם תקודד למספר **קבוע** של הוראות בשפה עילית.

יעילות המהדר - כל הוראה בשפה עילית תתורגם למספר **קבוע** של הוראות בשפת מכונה. **ללא תלות באורך הקלט**

מהדר יעיל יותר יכול להקטין מספר זה **בגורם קבוע** כלשהו.

3. הגדלת מהירות המעבד -

(לדוגמא מ 1 MHz ל1 GHz , תקטין את זמן הריצה בגורם קבוע (בדוגמא 1000).

© כל הזכויות שמורות לפרופסור עמוס ישראלי

<u>שיפורים בגורם קבוע</u>

כל שיפור באיכות ה**קידוד המהדר** או **המעבד**, משפיע על זמן הריצה **אך ורק** על ידי הכפלה בקבוע, שאינו תלוי באורך הקלט.
אין לזלזל בשיפור שיקטין את זמן הריצה למחצית, לשליש, לעשירית, או אף לאלפית (כאשר משפרים מחשב מערכו הקודם. כאשר מתמודדים עם בעייה קשה, יש למצות את כל אפשרויות השיפור של הפתרון.
לעתים כל האפשרויות גם יחד אינן מספיקות בהתמודדות עם בעיות בגודל (אורך קלט) הולך וגדל.

האם יש אפשרויות שיפור נוספות?

© כל הזכויות שמורות לפרופסור עמוס ישראלי

<u>שיפור האלגוריתם</u>

מספר הצעדים שאלגוריתם מסויים מבצע הוא **פונקציה של אורך הקלט לאלגוריתם.** במהלך הקורס אנו נוכח כי אפשר לשפר אלגוריתמים מסויימים בסדר גודל, למשל מסיבוכיות n^2 לסיבוכיות j=]. ברור אם כן כי בעזרת שיפור האלגוריתם אפשר לשפר את זמן הריצה בגורם **העולה על קבוע. בשום דרך אחרת אי אפשר להשיג שיפור כזה.**

© כל הזכויות שמורות לפרופסור עמוס ישראלי

<u>סיבוכיות זמן ריצה</u>

נתון אלגוריתם A הפותר בעיה P.

נתבונן בסדרת קלטים מ-P

$$I_1, I_2, \ldots, I_n, \ldots$$

שאורכם הולך וגדל.

בדרך הטבע, ככל שאורך הקלט גדול יותר, האלגוריתם יעבוד זמן רב יותר.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

<u>הגדרת סיבוכיות הזמן</u>

סיבוכיות הזמן של האלגוריתם או בפשטות **זמן הריצה** של האלגוריתם תוגדר כפונקציה המתאימה את מספר צעדי האלגוריתם לאורך הקלט.

$t(n) =$ ***מספר צעדים מירבי בריצה של האלגוריתם על קלט באורך*** n

© כל הזכויות שמורות לפרופסור עמוס ישראלי

<u>סיבוכיות המקרה הגרוע ביותר</u>

יש אלגוריתמים שזמן הריצה שלהם על קלטים באורך שווה משתנה מקלט לקלט.

דוגמה: במיון מהיר (Quick-Sort), יש קלטים רבים באורך n, שזמן הריצה שלהם הוא $\Theta(n \log n)$ אך עבור קלט **ממוין,** זמן הריצה הוא $\Theta(n^2)$.

במקרה זה, עלינו לבחור מהו זמן הריצה המייצג את הקלטים באורך n.

בדרך כלל, מקובל לבחור עבור כל n את הקלט עבורו האלגוריתם רץ במהירות הנמוכה ביותר (Worst Case).

© כל הזכויות שמורות לפרופסור עמוס ישראלי

נימוקים לשימוש בWorst Case

הנימוק המרכזי הוא **מניעת הפתעות.**

כשמשתמשים ב Worst Case,

סיבוכיות האלגוריתם היא **חסם עליון** על זמן הריצה עבור כל המקרים.

נימוק חשוב נוסף הוא שיחסית קל לחשב את המקרה הגרוע ביותר.

במקרים מסוימים (כגון במיון מהיר) נתיחס גם **לזמן הריצה הממוצע** (יוגדר מאוחר יותר) .

שימו לב: לעיתים משתמשים במושג Best Case לצרכי הסבר.

אין משמעות לסיבוכיות המקרה הטוב ביותר ואין להשתמש במושג זה לצרכי ניתוח זמן ריצה.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

הגדרה פורמלית של סיבוכיות זמן

יהי Al אלגוריתם כלשהו. לכל קלט חוקי I נסמן ב $t_{Al}(I)$ את זמן הריצה של Al על הקלט I.

סיבוכיות הזמן של Al מוגדרת על ידי:

$$t_{Al}(n) = \max_{|I|=n}\{t_{Al}(I)\}$$

לכל $n > 0$. כאשר המכסימום נלקח על קבוצת כל הקלטים באורך n.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

ניתוח זמן ריצה

מטרתו של ניתוח זמן ריצה של אלגוריתם היא:

1. לסייע לנו **לצפות** את זמן הריצה של תכנית המבוססת על האלגוריתם ללא הרצה של התוכנית.
2. להשוות יעילות של כמה אלגוריתמים.
3. להחליט אם לחפש אלגוריתם נוסף.

שימו לב: מפני שאלגוריתם הוא ברמת הפשטה גבוהה, לעתים קשה לעמוד על משמעות שינוי בגורם קבוע.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

מסקנה

בדיון בסיבוכיות זמן של אלגוריתמים נבדוק רק שיפורים שיקטינו את **סדר הגודל** של זמן הריצה **ונתעלם מגורמים קבועים.**

דוגמא

נניח כי אלגוריתם A פותר בעיה מסוימת בזמן n^2+7 ואלגוריתם B פותר אותה הבעיה בסיבוכיות n^2+12 אנו נאמר כי לשני האלגוריתמים סיבוכיות זמן $\Theta(n^2)$.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

השוואת קצבי גידול

הסימונים O, Θ, וΩ מאפשרים לנו להשוות בין קצבי הגידול של פונקציות שונות כאשר n שואף לאינסוף.

דוגמא

נניח כי זמן הריצה של תוכנית כלשהי מבוטא על ידי הפונקציה $T(n)$ המוגדרת על ידי:

$$T(n)=31n^3+17.5n^2-3.2n+1077$$

קל לראות (איך?) כי כאשר $n\to\infty$ ערך $T(n)$ נשלט על ידי n^3. במקרה כזה נאמר כי $T(n)=\Theta(n^3)$.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

הגדרות

נאמר כי $f(n)=O(g(n))$ (הפונקציה $g(n)$ **חוסמת מלמעלה באופן אסימפטוטי** את הפונקציה $f(n)$) אם קיימים קבועים c_0 ו n_0 המקיימים: לכל $n>n_0$, $f(n)\le c_0 g(n)$.

נאמר כי $f(n)=\Omega(g(n))$ (הפונקציה $g(n)$ **חוסמת מלמטה באופן אסימפטוטי** את הפונקציה $f(n)$) אם קיימים קבועים c_1 ו n_1 המקיימים: לכל $n>n_1$, $f(n)\ge c_1 g(n)$

שימו לב: אם $f(n)=O(g(n))$ אז $g(n)=\Omega(f(n))$.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

המשך ההגדרות

הפונקציה $f(n)$ היא $\Theta(g(n))$ (הפונקציה $f(n)$ **שוה אסימפטוטית** לפונקציה $g(n)$) אם

1. $f(n)=O(g(n))$

וגם

2. $g(n)=O(f(n))$

שימו לב: אם $f(n)=\Theta(g(n))$ אז גם $g(n)=\Theta(f(n))$

© כל הזכויות שמורות לפרופסור עמוס ישראלי

דוגמא

תהי

$$T(n)=31n^3+17.5n^2-3.2n+1077$$

נראה כי $T(n)$ היא $O(n^3)$.

$$T(n)\le 31n^3+17.5n^2+1077$$

עבור $n>1077$ נקבל כי

$$T(n)\le 31n^3+n^3+n^3=33n^3$$

מש"ל.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

סקלה למדידת קצב הגידול של פונקציות

אנו מחפשים דרך לתאר התנהגות של פונקציות. הדרך שנבחרה היא להשוות כל פונקציה לקבוצת פונקציות בסיסיות המוכרות לכל.

הפונקציות הבסיסיות הן:

1.פונקציות מעריכיות:

$f(n)=a^{n^k}$, $k=1,2,$.

2.מונומים $f(n)=n^k$,

$k=1,2,$

3.פונקציות לוגריתמיות

$f(n)=(\log n)^k$ $k=1,2,$.

שימו לב: כל הפונקציות הללו הן **חיוביות**.

הסדר האסימפטוטי

להלן הסדר האסימפטוטי של כמה פונקציות מוכרות:

$$1<\log n<(\log n)^k<\log n^{k+1}$$
$$<n^{\varepsilon}(\forall\varepsilon>0)<n^{1/2}<$$
$$n<n\log n<n^k<n^{k+1}<$$
$$<2^n<n!$$

השוואת סדרי גודל

בהינתן פונקציה כלשהי נרצה להשוות אותה לאחת הפונקציות התקניות. ברובם המכריע של המקרים איננו צריכים לחשב גבולות כלל אלא להשתמש בהגיון ובכמה חוקים בסיסיים.

כדוגמא נביא שאלה מתוך בוחן:

קבע את יחסי סדר הגודל בין

$$f(n)=n(\log n)^2$$

לבין

$$g(n)=\frac{n^2}{\log n}$$

בדוגמא זו, קל מאוד לראות, למשל על ידי הכפלת שתי הפונקציות בפונקציה $\log n$ כי $g(n)$ גדלה מהר יותר מ $f(n)$ כלומר $f(n)=O(g(n))$.

שימו לב:

לעתים אנו אומרים כי סיבוכיות האלגוריתם היא $O(f(n))$ כאשר בעצם אנו מתכוונים לומר $\Theta(f(n))$.

<u>דוגמא נוספת</u>

נתון כי הפונקציה $f(n)$ מתנהגת באופן שונה עבור ערכי *n* שונים כמתואר בטבלה הבאה:

>250	26-250	0-25	*n*
n^2	n^3	n^4	$f(n)$

מהי סיבוכיות הפונקציה *f*?

<u>תשובה:</u> מן ההגדרות נובע מייד כי מה שקובע את סיבוכיות הפונקציה היא התנהגותה כאשר ערכי *n* שואפים לאינסוף. להתנהגות *f* עבור מספר סופי של ערכי *n* אין שום חשיבות ולכן

$$f(n)=\Theta(n^2)$$

© כל הזכויות שמורות לפרופסור עמוס ישראלי

<u>הרצאה 2: אלגוריתמי מיון פשוטים</u>

בשליש הראשון של הקורס נעסוק בבעית המיון. בעיה זו היא אחת הבעיות החשובות והנחקרות ביותר בתורת האלגוריתמים.

בהרצאה זו נציג שני אלגוריתמי מיון פשוטים בסיבוכיות $\Theta\left(n^2\right)$:

1. מיון בחירה (Selection Sort).
2. מיון בועות (Bubble Sort) .

לכל אחד מן האלגוריתמים האלה, נוכיח את נכונותו וננתח את הסיבוכיות שלו.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

<u>הגדרה אינטואיטיבית</u>

אלגוריתם מיון מקבל כקלט מערך מספרים, ומחזיר מערך המכיל את אותם המספרים, מסודרים בסדר עולה או יורד. להלן נגדיר את בעית המיון באופן מתמטי.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

<u>הגדרה 2.1: תמורה</u>

מערך B הוא **<u>תמורה של מערך A</u>** אם המערכים A ו-B מכילים **<u>בדיוק אותם איברים</u>** וכל איבר מופיע בדיוק אותו מספר פעמים ב-A וב-B.

<u>לדוגמא</u>

1,2,3 היא תמורה של 3,1,2.

5,1,1 היא תמורה של 1,5,1

4,7,7 **<u>אינה תמורה</u>** של 4,4,7

3,2,7 **<u>אינה תמורה</u>** של 3,2,7,7

© כל הזכויות שמורות לפרופסור עמוס ישראלי

<u>בעיה חישובית 2.2: מיון</u>

<u>תיאור הקלט</u>

מערך A בן n מספרים המופיעים בסדר שרירותי. למען הפשטות נניח כי כל איברי המערך A **<u>נבדלים</u>**, כלומר: שונים זה מזה (distinct).

<u>תיאור הפלט</u>

מערך B ובו איברי A מסודרים לפי גודלם.

<u>במלים אחרות:</u> הפלט הוא תמורה ממוינת של איברי הקלט.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

מיון בחירה (Selection Sort)

```
SelectionSort(A)
for i = 1 to n − 1
    minind ← index of minimal
          element of A[i...n]
   Swap(A[i], A[minind])
  end for
Out(A)
```

אלגוריתם 2.3: מיון בחירה

```
Swap(i, j)
  temp ← i
  i ← j; j ← temp
end
```

אלגוריתם 2.4: השיטה *Swap*

© כל הזכויות שמורות לפרופסור עמוס ישראלי

תיאור האלגוריתם

הקלט מאוחסן במערך A שאיבריו הם מספרים.
המיון נערך **במקום** (in Place), כלומר בסיום ביצוע האלגוריתם המערך A מכיל את איברי **מערך הקלט** ממוינים.
לולאת האלגוריתם מבוצעת $n-1$ פעמים. במעבר ה-i בלולאה, האלגוריתם מחליף (*Swap*) את האיבר המינימלי בתת המערך $A[i...n]$ עם האיבר $A[i]$.
השורה השלישית באלגוריתם (ההוראה הראשונה בלולאת ה-**for**, מחשבת את האינדקס של האיבר המינימלי בתת המערך $A[i..n]$. אפשר לבצע הוראה זו בעזרת שיטה פשוטה שהסיבוכיות שלה היא $\Theta(n-i)$.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

תזכורת: הוכחת נכונות של אלגוריתם

כדי להוכיח נכונות של אלגוריתם עלינו להוכיח:

1. האלגוריתם מסתיים בכל ריצה עם קלט חוקי.
2. בכל ריצה עם קלט חוקי, הפלט נכון.

כדי לענות על שתי הדרישות הללו, עלינו להתיחס במדויק אל פעולת האלגוריתם כולל:

1. מספר המעברים בכל לולאה.
2. ערכי המשתנים בכל שלב של החישוב.
3. הצלחה או כישלון בביצוע טסטים כגון **if** ו-**while**.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

מיון בחירה - הוכחת סיום

כמו במקרים רבים, הוכחת הסיום של האלגוריתם היא מיידית ונובעת מכך שהאלגוריתם מכיל לולאה מקוננת אחת ושמספר המעברים בכל לולאה חסום על ידי גודל הקלט n.
לאחר הוכחת נכונות הפלט, נראה כי סיבוכיות האלגוריתם היא $\Theta(n^2)$.
מכאן נובע כי לכל קלט האלגוריתם מסתיים.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

נכונות פלט אלגוריתמי מיון

כדי להוכיח נכונות פלט של אלגוריתם מיון עלינו:

1. להוכיח כי מערך הפלט הוא תמורה של מערך הקלט.
2. להוכיח כי מערך הפלט ממוין.

משפט 2.5: נכונות מיון בחירה

אלגוריתם *SelectionSort* מקבל כקלט מערך של מספרים ומחזיר תמורה ממוינת של מערך הקלט.

שימו לב

טענת הנכונות לכל אלגוריתם מיון היא זהה, למעט שם האלגוריתם.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

הוכחה

נתבונן שוב בקוד

```
SelectionSort(A)
for i = 1 to n-1
    minind ← index of minimal
            element of A[i...n]
    Swap(A[i], A[minind])
end for
Out(A)
```

בטרם נציע שמורה נגדיר סימונים:

m_1 - האיבר המינימלי במערך A.

m_2 - האיבר השני בקטנו במערך A.

...

m_i - האיבר ה-i בקטנו במערך A.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

השמורה

בתום המעבר ה-i, $0 \le i \le n$, מתקיים

$$A[1...i] = [m_1, m_2, ...m_i]$$

הוכחת נכונות השמורה נותרת כתרגיל לקורא.

הוכחת משפט 2.5

כדי להוכיח את נכונות האלגוריתם, נשים לב כי לאחר $n-1$ מעברים בלולאה השמורה נראית כך:

$$A[1...n-1] = [m_1, m_2, ...m_{n-1}]$$

האיבר היחיד במערך הקלט שלא מופיע בשמורה הוא m_n. מאחר שהוכחנו שמערך הפלט הוא תמורה של מערך הקלט, m_n חייב להופיע במקום ה-n

© כל הזכויות שמורות לפרופסור עמוס ישראלי

במערך הפלט, כלומר מערך הפלט ממוין. **מש"ל**

סיבוכיות מיון בחירה

האלגוריתם מבצע $n-1$ מעברים (איטרציות) בלולאה הראשית.

במעבר ה-i, $1 \le i \le n-1$, האלגוריתם מחשב מינימום של $n-i+1$ איברים.

מכאן, סיבוכיות המעבר ה-i, $1 \le i \le n-1$, היא $\Theta(n-i+1) = \Theta(n-i)$.

מכאן נובע כי סיבוכיות האלגוריתם שווה לסכום הטור החשבוני

$$T(n) = \sum_{i=1}^{n-1}(n-i) = \Theta(n^2)$$

© כל הזכויות שמורות לפרופסור עמוס ישראלי

מיון בועות (Bubble Sort)

מיון בועות הוא אלגוריתם קלאסי ששנים ארוכות נחשב כאלגוריתם המיון היעיל ביותר הקיים.
יתרונו הגדול של האלגוריתם הוא קלות הקידוד.
בהרצאה הבאה, נציג את אלגוריתם **מיון מיזוג** שהסיבוכיות שלו היא $\Theta(n\log n)$. מכאן אפשר להסיק כי מיון בועות (וגם מיון בחירה) אינם אלגוריתמים יעילים.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

מיון בועות: הקוד

```
BubbleSort(A)
for Lim = n downto 2
     for j = 1 to Lim-1
          if A[j] > A[j+1]
               Swap(A[j], A[j+1])
          end if
     end for
end for
out(A)
```

אלגוריתם 2.6: מיון בועות

© כל הזכויות שמורות לפרופסור עמוס ישראלי

תיאור מיון בועות

המיון נערך במקום, מערך הקלט הוא גם מערך הפלט. האלגוריתם מבצע $n-1$ מעברים בלולאה הראשית. משתנה הלולאה הוא Lim, ערכו התחילי שווה למספר איברי הקלט, n. בכל מעבר בלולאה הראשית, ערכו של Lim קטן ב-1.
במלים אחרות: במעבר ה-i בלולאה, ($Lim=n-i+1$,$i=1,2,...,n-1$), בכל מעבר ההאלגוריתם בוחן את זוגות האיברים הסמוכים הבאים, בסדר הנתון:

$$(1,2),(2,3),...,(Lim-1,Lim)$$

בכל פעם שזוג איברים כזה מסודר **בסדר יורד**, איברים אלה **מוחלפים** (*Swap*).

© כל הזכויות שמורות לפרופסור עמוס ישראלי

תיאור מיון בועות (המשך)

להלן נשתמש בסימונים הבאים:

M_1 - האיבר המרבי במערך A.

M_2 - האיבר השני בגדלו במערך A.

...

...

M_i - האיבר ה-i בגדלו במערך A.

שימו לב: בהוכחה הקודמת האיבר M_i סומן על ידי m_{n-i+1}.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

תיאור מיון בועות (המשך)

במבט מיידי אפשר לקבוע כי

- בתם המעבר הראשון בלולאה החיצונית, $(Lim = n)$, $A[n] = M_1$.
- בתם המעבר השני בלולאה החיצונית, $(Lim = n-1)$, $A[n-1] = M_2$.
- בתם המעבר ה-i $(Lim = n-i+1)$, $A[n-i+1] = M_i$.

תיאור מיון בועות (המשך)

האלגוריתם נקרא **מיון בועות** שכן במעבר ה-i בלולאה הראשית, האיבר M_i עולה לראש המערך כבועה הצפה על פני המים.

אנו נשתמש באבחנות אלה במהלך הוכחת נכונות האלגוריתם.

משפט 2.7 נכונות *Bubblesort*

אלגוריתם *BubbleSort* המקבל כקלט מערך A שאיבריו מספרים, ומחזיר תמורה ממוינת של איברי מערך הקלט.

שימו לב לדמיון בין משפט 2.5 למשפט 2.7

הוכחת המשפט

כזכור עלינו להוכיח:

1. האלגוריתם מסתיים.
2. הפלט הוא תמורה של הקלט.
3. הפלט ממוין.

הוכחת 1

האלגוריתם מכיל שתי לולאות מקוננות, מספר המעברים בכל לולאה חסום על ידי n ולכן ברור כי האלגוריתם מסתיים.

הוכחת 2

בכל מהלך ביצוע האלגוריתם, מיקום איברי המערך משתנה אך ורק על ידי ביצוע החלפה (*Swap*) בין שניים מהם ועל כן ברור כי מערך הפלט הוא תמורה של מערך הקלט.

נותר לנו להוכיח אך ורק כי מערך הפלט ממוין.

הוכחת 3: הגדרת השמורה

בתום המעבר ה-i בלולאה הראשית, נחלק את המערך A ל:

1. חלק **עליון** ובו i האיברים הגדולים במערך בסדר עולה: $M_i,...,M_1$.
2. לחלק **תחתון** ובו $n-i$ האיברים הקטנים במערך הקלט, אשר מופיעים בסדר **שרירותי**.

שימו לב: במהלך המעבר ה-i, $Lim = n-i+1$, כלומר המשתנה Lim מצביע על האיבר העליון בחלק התחתון.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

איור ??.2 תיאור גרפי של מצב המערך

© כל הזכויות שמורות לפרופסור עמוס ישראלי

הוכחת הנכונות בסיס האינדוקציה

ההוכחה באינדוקציה על המעברים בלולאה.

בסיס $(i=1)$

עלינו להוכיח כי בתום המעבר הראשון, $A[n]=M_1$ והאיבר $A[n]$ לא ישתנה יותר.

נניח כי בתחילת המעבר הראשון $A[j]=M_1$

במהלך האיטרציה הראשונה, M_1 לא זז ממקומו עד אשר האיבר $A[j]$ מושווה עם $A[j+1]$ ואז הם מוחלפים.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

הוכחת בסיס האינדוקציה (המשך)

מרגע זה ועד תום האיטרציה, האיבר M_1 מושווה עם כל איבר הנמצא מעליו. בכל השוואה כזו, M_1 מחליף את מקומו (כי הוא האיבר המכסימלי במערך הקלט). בסוף האיטרציה מתקיים $A[n]=M_1$.

מן הקוד נובע כי באיטרציות הבאות, האיבר $A[n]$ לא יושווה יותר עם אף איבר נוסף ומקומו לא ישתנה עד תום ביצוע האלגוריתם.

מש״ל

© כל הזכויות שמורות לפרופסור עמוס ישראלי

צעד האינדוקציה

הנחה

נניח כי השמורה מתקיימת בכל מעבר בלולאה עד המעבר ה-i, ונוכיח כי היא מתקיימת גם בתום המעבר ה-$i+1$.

לפי ההנחה, בתחילת המעבר ה-$i+1$, בחלק העליון מאוחסנים i האיברים הגדולים במערך הקלט, ממוינים בסדר עולה.

במצב זה, האיבר המרבי בחלק התחתון הוא M_{i+1}.

בדומה להוכחת הבסיס, במעבר הנוכחי M_{i+1} יושווה רק עם איברים הקטנים ממנו. בסוף המעבר $A[n-i]=M_{i+1}$.

מש"ל

© כל הזכויות שמורות לפרופסור עמוס ישראלי

אלגוריתמים א' הרצאה מס' 2 - פרופ' עמוס ישראלי

סיום הוכחת המשפט

הוכחת הנכונות מושלמת על ידי:

טענה 2.8

בתום פעולת מיון בועות הפלט ממוין.

הוכחה

נציב בשמורה $i=n-1$, ונקבל כי בתום המעבר ה-$n-1$ מתקיים:

$$A[2,...,n]=M_{n-1},...,M_1$$

כבר הוכחנו כי בכל שלב לביצוע האלגוריתם המערך A מאחסן תמורה של מערך הקלט ומכאן נובע כי בתום המעבר ה-$n-1$, $A[1]=M_n$.

מש"ל

© כל הזכויות שמורות לפרופסור עמוס ישראלי

אלגוריתמים א' הרצאה מס' 2 - פרופ' עמוס ישראלי

ניתוח סיבוכיות הזמן

בקרת התכנית, למעט הקריאות לשגרה $Swap$, אינה תלויה באברי הקלט אלא באורכו בלבד. בריצה עם מערך באורך n, מתבצעים $n-1$ מעברים, עבור $LIM=n,,...,2$.

בכל מעבר כזה מתבצעות $LIM-1=n-1,...,1$ השוואות וכל השוואה גוררת לכל היותר חילוף אחד.

מספר ההשוואות הכולל הוא

$$\sum_{1}^{n-1} i=\frac{n(n-1)}{2}=\frac{n^2-n}{2}=\Theta(n^2)$$

מכאן, סיבוכיות הזמן של האלגוריתם היא $\Theta(n^2)$.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

אלגוריתמים א' הרצאה מס' 2 - פרופ' עמוס ישראלי

מוטיבציה לאלגוריתם המשופר

באלגורים $BubbleSort$, מספר האיברים בחלק העליון של המערך A, לאחר המעבר ה-i בלולאה הראשית הוא i.

יש מערכי קלט בהם בשלב מסוים של ביצוע האלגוריתם, מתקבל חלק עליון ממוין גדול יותר.

האלגוריתם המשופר מזהה את גודל החלק העליון המתקבל בפועל ומשתמש במידע הזה כדי להקטין את מספר צעדי האלגוריתם עבור קלטים כאלה.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

האלגוריתם המשופר

```
ImpBubble(A)
Lim ← n
while Lim > 1 do
    newLim ← 1
  for j = 1 to Lim − 1 do
    if A[j] > A[j+1]
        Swap(A[j], A[j+1])
          newLim ← j
     end if
   end for
  Lim ← newLim
End while
```

איור 2.3: מיון בועות משופר

© כל הזכויות שמורות לפרופסור עמוס ישראלי

הערות לאלגוריתם המשופר

באלגוריתם המשופר, ערך המשתנה Lim נקבע עם תום כל איטרציה, כך שיצביע על האיבר התחתון של החילוף האחרון שהתבצע במהלך המעבר.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

דוגמא

נניח שהקלט הוא המערך הבא:

1	5	3	2	4	7	6	8

להלן נציג באופן גרפי את מהלך ביצוע האלגוריתם על הקלט הנתון.
בתיאור שנציג המשתנה Lim מצביע על התא המוצלל בהיר והמשתנה $newLim$ מצביע על התא המוצלל כהה:

© כל הזכויות שמורות לפרופסור עמוס ישראלי

תיאור המעבר הראשון בלולאה

1	5	3	2	4	7	6	8

במהלך המעבר הראשון $Lim = 8$.
בתחילת המעבר $NewLim = 1$.

1	3	5	2	4	7	6	8

לאחר ביצוע $Swap[2,3]$,
$NewLim = 2$.

1	3	2	5	4	7	6	8

לאחר ביצוע $Swap[3,4]$,
$NewLim = 3$.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

1	3	2	4	5	7	6	8

לאחר ביצוע $Swap[4,5]$,

$.NewLim = 4$

1	3	2	4	5	6	7	8

© כל הזכויות שמורות לפרופסור עמוס ישראלי

תיאור המעבר השני בלולאה

במהלך המעבר השני $.Lim = 6$

בתחילת המעבר $.NewLim = 1$

1	3	2	4	5	6	7	8

לאחר ביצוע $Swap[2,3]$,

$.NewLim = 2$

1	2	3	4	5	6	7	8

© כל הזכויות שמורות לפרופסור עמוס ישראלי

תיאור המעבר השלישי בלולאה

1	2	3	4	5	6	7	8

במהלך המעבר השלישי $.Lim = 2$

בתחילת המעבר $.NewLim = 1$

במהלך המעבר השלישי

ברור מספר הצעדים שהאלגוריתם

מבצע על הקלט שבדוגמה קטן מאוד.
להלן נוכיחכונות האלגוריתם המשופר

וננתח את הסיבוכיות שלו.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

הוכחת סיום האלגוריתם המשופר

טענה

בכל מעבר בלולאה הראשית הערך של

המשתנה Lim קטן לפחות ב-1.

הוכחה

נכונות הטענה נובעת מן הקוד (הוכיחו

זאת על ידי מעקב).

מן הטענה נובע כי האלגוריתם

מסתיים לאחר לכל היותר $n-1$

מעברים בלולאה הראשית.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

הוכחת נכונות האלגוריתם המשופר

בטענת הנכונות שהוכחנו קודם השתמשנו בעובדה שבסיום האיטרציה ה-i ערך המשתנה Lim הוא $n-i$.

הטענה הבאה מתקבלת מן הטענה הקודמת על ידי שימוש בערך המשתנה Lim עצמו.

טענה

לכל $1 \le i \le n$, לאחר האיטרציה ה- i מתקיים:

$$A[n]=M_1,\dots,A[n-Lim+1]=M_{Lim}$$

שיוויונים אלה נשמרים עד סוף ביצוע האלגוריתם.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

להלן יקראו אברים אלה (כולל M_{Lim}) בשם: החלק הממוין של המערך A.

החלק השני של המערך A:

$$A[1],A[2],\dots,A[n-Lim]$$

יקרא: אזור אי הודאות

הוכחת הטענה מתקבלת על ידי התאמת ההוכחה הקודמת והיא נשארת כתרגיל לקורא.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

נכונות מיון בועות (המשך)

הוכחת הנכונות מושלמת על ידי המשפט הבא:

משפט

בתום פעולת מיון בועות הפלט ממוין.

הוכחה

ההוכחה נובעת מיידית מנכונות טענת האינדוקציה בסיום המעבר האחרון בלולאת האלגוריתם.

מש"ל

© כל הזכויות שמורות לפרופסור עמוס ישראלי

סיבוכיות הזמן באלגוריתם המשופר

באלגוריתם המשופר, זמן הריצה תלוי במספר ההיפוכים בקלט ולא רק באורכו:

1. עבור קלט ממוין בסדר עולה, סיבוכיות הזמן היא לינארית. זהו המקרה הטוב ביותר.
2. סיבוכיות הWorst Case מתקבלת על ידי קלט בממוין בסדר יורד. במקרה זה, באיטרציה ה i מתבצעים בדיוק $n-i$ חילופים ובסך הכל, מספר החילופים הכולל הוא , כמו באלגוריתם הקודם, $O(n^2)$. אפשר להראות כי זהו גם מספר הצעדים המכסימלי שהאלגוריתם מבצע עבור קלט כלשהו.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

<u>סיכום</u>

בהרצאה זו הצגנו שני אלגוריתמי מיון קלאסיים (ולא יעילים):

1. מיון בחירה (Selection Sort).
2. מיון בועות (Bubble Sort) .

לכל אלגוריתם, הוכחנו את נכונותו ונתחנו את הסיבוכיות שלו במקרה הגרוע ביותר. לאכזבתנו, סיבוכיות שלושת האלגוריתמים הללו היא $\Theta(n^2)$.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

<u>תרגילים</u>

<u>תרגיל 2.1</u>

נתון הקוד של מיון בחירה:

$SelectionSort(A)$
for $i = 1$ **to** $n-1$
    $minind \leftarrow$ index of minimal element of $A[i...n]$
    $Swap(A[i], A[ind])$
end for
$Out(A)$

© כל הזכויות שמורות לפרופסור עמוס ישראלי

הוכיחו, באינדוקציה על המעברים בלולאה, כי בתום המעבר ה-i, $0 \le i \le n$, מתקיים

$$A[1...i] = [m_1, m_2, ...m_i]$$

וכי ערכים אלה לא משתנים עד סיום ביצוע האלגוריתם.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

<u>הרצאה 3: בנייה של אלגוריתמי מיון יעילים בשיטת הפרד ופתור</u>

הרצאה זו מוקדשת להכרות עם **הפרדיגמה** לבניית אלגוריתמים רקורסיביים הקרויה **הפרד ופתור (Divide and Conquer)**.

אנו נציג את השיטה באופן כללי ולאחר מכן נדגים אותה על ידי בנייה של אלגוריתם מיון יעיל הקרוי **מיון באמצעות מיזוג (Merge-sort)**.

סיבוכיות הזמן של האלגוריתם היא $\Theta(n\log n)$.

פרדיגמה (paradigm) היא שיטה כללית לפתרון סוג מסוים של בעיות. במדעי המחשב אנו מדברים על פרדיגמות של תכנון אלגוריתמים כגון:
הפרד ופתור, תכנון דינמי ועוד, ועל פרדיגמות של תכנות כגון Top Down, תכנות מונחה עצמים וכ"ו.

<u>הערת צד 3.1</u>

<u>דוגמא: חישוב מספר הצמתים בעץ</u>

<u>בעיה חישובית 3.2</u>

<u>קלט</u>

עצים בינאריים.

<u>פלט</u>

מספר הצמתים (צמתים פנימיים ועלים) בעץ הקלט.

<u>רעיון האלגוריתם</u>

1. אם העץ **ריק** יש בו 0 צמתים.
2. בעץ ש**אינו ריק** מספר הצמתים שוה לסכום מספר הצמתים בתת העץ השמאלי (קריאה רקורסיבית מס' 1), מספר הצמתים בתת העץ הימני (קריאה רקורסיבית מס' 2) ועוד אחד (השורש).

האלגוריתם *Count*

$Count(T)$

$\textbf{if } T=\phi \ out(0)$

$c_l \leftarrow Count(T_l)$

$c_r \leftarrow count(T_r)$

$out(c_l + c_r + 1)$

אלגוריתם 3.3: *Count*

מתכון לבניית אלגוריתמים בשיטת הפרד ופתור

נניח כי נתונה בעיה חישובית $P=<In,Out,f>$. להלן מוצג "מתכון" לאלגוריתם רקורסיבי *GenRec* לפתרון הבעיה *P*.

1. נזהה תת קבוצה *Base* של קבוצת הקלט *In*, ובה קלטים הניתנים לפתרון ישיר על ידי שיטה שנציג ונקרא לה *Direct*.
2. נציג שיטה *Break* השוברת כל קלט $i_{original} \in In - Base$, למספר קבוע, *l*, של תת קלטים $i_1,...,i_l$.

3. האלגוריתם שנציג יפעיל את השיטה *GenRec* על כל אחד מתת הקלטים $i_1,...,i_l$, כדי לקבל את הפלטים $o_1,...,o_l$.
4. נציג שיטה *Combine* המקבלת כקלט את $o_1,...,o_l$ ובונה פלט $o_{original}$ המתאים לקלט המקורי.

הסכמה המתקבלת מוצגת באיור 3.3 .

סכמה של הפרד ופתור

$GenRec(i_{original})$

$\quad \textbf{if } I_{original} \in Base$

$\qquad out(Direct(I_{original}))$

$\qquad [i_1,...,i_l] \leftarrow Split(i_{original})$

$\quad \textbf{for} \leftarrow 1 \textbf{ to } l$

$\qquad o_i \leftarrow GenRec(i_i)$

$\quad \textbf{end for}$

$\qquad o_{original} \leftarrow$

$\qquad\quad Combine(o_1,...,o_l)$

$\quad out(o_{original})$

איור 3.2: סכמה של אלגוריתם בשיטת הפרד ופתור.

כאשר מפעילים אלגוריתם רקורסיבי עם קלט שאינו מקרה קצה, הקלט מתפרק למספר תת קלטים **ראשיים**. תת הקלטים הראשיים, הם מקרי קצה או שהם ניתנים לפרוק נוסף. אם נעקוב אחרי שרשרת הקריאות הרקורסיביות, נוכח כי למעשה, כל בעיה מתפרקת למספר מקרי קצה אשר כל אחד מהם נפתר בעזרת קריאה ל *Direct*.
הפלטים של הקריאות האלה משולבים לפלט לבעיה המקורית בעזרת קריאות ל *Combine*.

הערת צד 3.3

© כל הזכויות שמורות לפרופסור עמוס ישראלי

נכונות אלגוריתימים רקורסיביים

באלגוריתם רקורסיבי אין לולאות ולכן אין אפשרות להוכיח את נכונותו באינדוקציה על המעברים בלולאה.
במקום זאת, נשתמש **באינדוקציה על אורך הקלט**.
ה"מתכון" להוכחות כאלה, מתואר בשקף הבא.

ה"מתכון" אינו מבטיח נכונות. תמיד יש לבדוק היטב את ההוכחה. הוכחה, צריך לוודא כי היא אכן אינה שגויה.

הערת צד 3.4

© כל הזכויות שמורות לפרופסור עמוס ישראלי

"מתכון" להוכחת נכונות של אלגוריתמים רקורסיביים

הוכחת הנכונות של אלגוריתמים רקורסיביים עוקבת במדויק אחרי מבנה האלגוריתמים הללו, כפי שמופיע **בסכמה 3.2:**

- **בסיס האינדוקציה**: יש להוכיח ישירות, על ידי מעקב, כי השיטה *Base* מחזירה פלט נכון בכל קריאה על אחד ממקרי הקצה.
- **הנחת האינדוקציה:** יש להניח כי בכל קריאה רקורסיבית לשיטה *GenRec* המופעלת עם כל אחד מקלטי המשנה $o_1,\ldots,o_l$ מוחזר פלט נכון.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

- **ההוכחה:** יש להוכיח, שוב על ידי מעקב, כי $o_{original}$ הוא הפלט המתאים לקלט המקורי $i_{original}$.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

באופן פורמלי, עלינו להוכיח גם כי לא תיתכן שרשרת קריאות רקורסיבית אין סופית.
באופן לא פורמלי, אנו מסתפקים בהוכחה כי בכל קריאה רקורסיבית אורך הקלט יורד. תחת ההנחה כי מקרה הקצה מטפל במכל הקלטים באורך מינימלי (שהיא נכונה בכל האלגוריתמים שאנו מציגים) ההוכחה נכונה.

הערת צד 3.5

© כל הזכויות שמורות לפרופסור עמוס ישראלי

נכונות האלגוריתם *Count*

בטרם ניגש להוכחה נתבונן שוב בקוד האלגוריתם

האלגוריתם Count

$Count(T)$

  if $T=\phi$ $out(0)$

  $c_l \leftarrow Count(T_l)$

  $c_r \leftarrow count(T_r)$

  $out(c_l + c_r + 1)$

נשים לב כי באלגוריתם זה, השיטות *Direct*, *Split*, ו-*Combine*, אינן מופיעות בקוד באופן מוחצן.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

נכונות האלגוריתם Count

משפט 3.6

אלגוריתם *Count* מקבל כקלט עץ בינארי T ומחשב את מספר הצמתים בעץ T.

הוכחה

נוכיח את נכונות הטענה באינדוקציה על מספר הצמתים בעץ T.

בסיס: $(T=\phi)$

זהו מקרה הקצה. מן הקוד נובע כי האלגוריתם מחזיר 0.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

צעד האינדוקציה

הנחת האינדוקציה

נניח כי האלגוריתם מחשב באופן נכון את מספר הצמתים בכל עץ שמספר הצמתים בו קטן מ-n (נשתמש באינדוקציה שלמה).

הוכחה

נניח כי T הוא עץ בינארי עם n צמתים. במהלך ביצוע $Count(T)$ מתבצעות הקריאות הרקורסיביות $Count(T_l)$ ו- $count(T_r)$.

מספר הצמתים ב-T_l (בהתאמה T_r) קטן ממש מ-n, כי שורש העץ לא שייך ל-T_l (T_r).

© כל הזכויות שמורות לפרופסור עמוס ישראלי

לפי הנחת האינדוקציה הקריאה $Count(T_l)$ (בהתאמה $Count(T_r)$), מחזירה את מספר הצמתים ב-T_l (T_r). נכונות האלגוריתם נובעת מיידית מן הקוד.

מש"ל

© כל הזכויות שמורות לפרופסור עמוס ישראלי

"מתכון" לניתוח סיבוכיות אלגוריתמים רקורסיביים

בניתוח הסיבוכיות, עלינו להמנע מספירה כפולה של צעדי חישוב המבוצעים בקריאות הרקורסיביות. אנו משיגים זאת על ידי:

1. הערכת מספר הצעדים שכל קריאה לאלגוריתם מבצעת. בהערכה זו סיבוכיות כל קריאה רקורסיבית נחשבת כקבוע (Constant).
2. הערכת מספר הקריאות הרקוסיביות המתבצעות.
3. הערכת מספר הצעדים הכולל המתבצע על ידי כל הקריאות.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

ניתוח סיבוכיות *Count*

1. נניח כי האלגוריתם *Count* נקרא עם קלט עץ בינארי T. מספר הקריאות לשיטה *Count* הוא $2|T|+1$. כדי לראות זאת, נשים לב כי עבור כל צומת של העץ T, האלגוריתם מבצע בדיוק שתי קריאות רקורסיביות, ולכן מספר הקריאות הרקורסיביות הוא $2|T|$. ביחד עם הקריאה המקורית, נקבל כי מספר הקריאות הכולל הוא $2|T|+1$.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

2. מספר הצעדים שכל קריאה מבצעת הוא קבוע (Const).
3. נניח מספר צעדי החישוב שכל קריאה רקורסיבית מבצעת, חסום על ידי קבוע c. מכאן נובע מיידית כי סיבוכיות האלגוריתם חסומה על ידי

$$2(|T|+1)c=\Theta(|T|)$$

כעת נפנה להצגת אלגוריתם **מיון באמצעות מיזוג (Merge Sort)** . האלגוריתם מבוסס על תבנית הפרד ופתור.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

מיון באמצעות מיזוג (Merge Sort)

להלן תיאור האלמנטים השונים באלגוריתם:

1. *Direct* - מקרי הקצה של האלגוריתם הם כל המערכים שלהם איבר יחיד. כל מערך כזה הוא כמובן ממוין ומוחזר כפלט ללא שינוי.
2. *Split* - מחלקת את מערך הקלט A לשני תת מערכים באורך $n/2$ כל אחד.
3. **קריאה רקורסיבית** של אלגוריתם המיון עם כל אחד מתת המערכים.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

4. *Combine* - מיזוג (**merge**) של שני תת המערכים הממוינים למערך ממוין אחד.

נתחיל את פיתוח האלגוריתם בהצגת אלגוריתם מיזוג לשני מערכים ממוינים.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

אלגוריתם מיזוג (Merge)

אלגוריתם המיזוג הוא אלגוריתם **איטרטיבי**, אשר משמש כשיטת איחוד תת הפתרונות (*Combine*) באלגוריתם *MergeSort*.

הנחות: A ו-B הם מערכים ממוינים בסדר עולה, שגודלם m ו-n בהתאמה.

האלגוריתם *Merge* ממזג את A למערך C שארכו $n+m$ ואיבריו ממוינים בסדר עולה.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

תיאור האלגוריתם

בתחילת ביצוע האלגוריתם מאתחלים שלושה מצביעים (Pointers) כך שיצביעו אל תחילת שלושת המערכים. בכל שלב של האלגוריתם משווים בין שני האיברים אליהם מכוונים המצביעים ל-A ול-B ומעבירים את הקטן מבין השנים לאיבר עליו מכוון המצביע ל - C. לאחר מכן, מקדמים את המצביע שהצביע על האיבר שהועבר ואת המצביע של מערך C.

כדי למנוע גלישה מעבר לגבולות A ו-B, מוסיפים לכל אחד מהם את **איבר חוסם** שערכו ∞.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

אלגוריתם Merge

$$
\begin{array}{l}
Merge(A,B) \\
\quad A[m+1] \leftarrow \infty, B[n+1] \leftarrow \infty \\
\quad ap \leftarrow 1;\ bp \leftarrow 1;\ cp \leftarrow 1; \\
\quad \textbf{while } (cp < m+n+1) \\
\qquad \textbf{if } A[ap] \le B[bp] \\
\qquad\quad C[cp] \leftarrow A[ap];\ ap++ \\
\qquad\quad \textbf{else } C[cp] \leftarrow B[bp];\ bp++ \\
\qquad \textbf{end if} \\
\qquad cp++ \\
\quad \textbf{end while} \\
\textbf{out}(C)
\end{array}
$$

אלגוריתם 3.6: אלגוריתם *merge*

© כל הזכויות שמורות לפרופסור עמוס ישראלי

הוספת האיברים החוסמים מפשטת את הצגת האלגוריתם. כאשר מממשים את האלגוריתם בתכנה, יש להשתמש במנגנון תכנתי מתאים כדי למנוע גלישה.
לדוגמה: לאחר כל ביצוע של הוראת ה-**if**, אפשר לבדוק האם המצביע המתאים עבר את גודל המערך.

הערת צד 3.7

© כל הזכויות שמורות לפרופסור עמוס ישראלי

נכונות אלגוריתם *merge*

משפט 3.8

לכל $1 \le i \le m+n$, לאחר i מעברים בלולאה הראשית, המערך $C[1..i]$ מכיל תמורה ממוינת של i האיברים הקטנים ביותר מבין איברי המערכים A ו-B. המצביע pa (בהתאמה pb) מצביע על האיבר הראשון במערך A (בהתאמה B) שטרם העבר למערך C. ערכו של המצביע pc הוא $i+1$.

הוכחה

תרגיל בית

משפט 3.9

אלגורתם המיזוג הוא נכון.

הוכחה

תרגיל בית.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

סיבוכיות אלגוריתם *merge*

משפט 3.10

סיבוכיות אלגוריתם המיזוג היא לינארית בסכום אורכי המערכים $t_{merge}(n,m) = \Theta(n+m)$.

הוכחה

תרגיל בית.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

אלגוריתם המיון - תאור

מקרי קצה

מקרי הקצה של אלגוריתם *MergeSort* הם **מערכים עם איבר יחיד**.ברור כי כל מערך כזה הוא ממוין.

חלוקה לתת-קלטים ראשיים

נחלק את מערך הקלט לחלק תחתון ובו $\lfloor n/2 \rfloor$ איברים, ולחלק עליון ובו $n-\lfloor n/2 \rfloor$ איברים. בהנתן פתרונים, לשני תת-הקלטים הישירים, כלומר שני תת-מערכים ממוינים, נשתמש באלגוריתם המיזוג, שהצגנו קודם, כדי לקבל מהם מערך ממוין המכיל את איברי הקלט המקורי.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

אלגוריתם המיון - הקוד

$$
\begin{aligned}
& MergeSort(A) \\
& \quad \textbf{if } |A|=1 \ out(A) \\
& \quad A_l = A[1 \ldots \lfloor n/2 \rfloor] \\
& \quad A_r = A[\lfloor n/2 \rfloor + 1 \ldots n] \\
& \quad B_l = MergeSort(A_l) \\
& \quad B_r = MergeSort(A_r) \\
& \quad C \leftarrow merge(B_l, B_r) \\
& \quad out(C)
\end{aligned}
$$

איור 3.4: מיון מיזוג (Merge Sort)

© כל הזכויות שמורות לפרופסור עמוס ישראלי

דוגמא

בשקפים הבאים מוצגת דוגמא מפורטת לשימוש במיון מיזוג. בדוגמא זו, קריאות רקורסיביות מוצגות כאילו נעשו במקביל. למעשה, קריאות אלה מתבצעות באופן סדרתי.

	Sort(1)							
In (sort1)	2	7	5	3	6	8	4	1
	Sort(2)				**Sort(2)**			
In (sor2)	2	7	5	3	6	8	4	1
	Sort(3)		**Sort(3)**		**Sort(3)**		**Sort(3)**	
In (sort3)	2	7	5	3	6	8	4	1
	Sort	**Sort**	**Sort**	**Sort**	**Sort**	**Sort**	**Sort**	**Sort**
In(sort4)	2	7	5	3	6	8	4	1
Out(sort4)	2	7	5	3	6	8	4	1

© כל הזכויות שמורות לפרופסור עמוס ישראלי

	Mer(3)		**Mer(3)**		**Mer(3)**		**Mer(3)**	
In(merge3)	2	7	5	3	6	8	4	1
Out(merge3)	2	7	3	5	6	8	1	4
Out(sort3)	2	7	3	5	6	8	1	4
	Merge(2)				**Merge(2)**			
In(merge2)	2	7	3	5	6	8	1	4
Out(merge2)	2	3	5	7	1	4	6	8
Out(sort2)	2	3	5	7	1	4	6	8
	Merge(1)							
In(merge1)	2	3	5	7	1	4	6	8
Out(merge1)	1	2	3	4	5	6	7	8
Out(sort1)	1	2	3	4	5	6	7	8

איור 3.5: סדר הקריאות בהפעלת מיון מיזוג

© כל הזכויות שמורות לפרופסור עמוס ישראלי

<u>נכונות מיון מיזוג</u>

<u>משפט 3.5:</u>

לכל $n > 0$, אלגוריתם מיון-מיזוג ממיין כל קלט חוקי באורך n .

<u>הוכחה</u>

נוכיח את המשפט באינדוקציה על **אורך הקלט** n.

<u>בסיס</u> $(n = 1)$

במקרה זה, הקלט ממוין באופן טריויאלי ולכן הקלט שווה לפלט. מן הקוד נובע כי הבקרה תוחזר מייד עם פלט שווה לקלט.

מש"ל

© כל הזכויות שמורות לפרופסור עמוס ישראלי

<u>צעד האינדוקציה</u> $(n > 1)$

נניח כי מיון מיזוג מחזיר פלט ממוין בכל קריאה עם קלט שארכו **קטן** מ-n ונניח כי האלגוריתם נקרא עם קלט שארכו n.

מן הקוד נובע כי, אם $n > 1$, מערך הקלט מחולק ל**שני** תת-קלטים ראשיים: A_l, שארכו $\lfloor n/2 \rfloor$ ו-A_r שארכו $n - \lfloor n/2 \rfloor$. ארכו של כל תת-קלט ראשי **קטן** מ-n ולכן, לפי הנחת האינדוקציה, מערכי הפלט B_l ו-B_r ממוינים בסדר עולה.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

<u>צעד האינדוקציה (המשך)</u>

לאחר מכן, *mergesort* מפעיל את השיטה *merge* על המערכים B_l ו-B_r ומחזיר את המערך הממוזג.

מאחר שהמערכים B_l ו-B_r ממוינים, נובע מנכונות האלגוריתם *merge* כי מערך הפלט ממוין בסדר עולה.

מש"ל

© כל הזכויות שמורות לפרופסור עמוס ישראלי

<u>ניתוח סיבוכיות מיון-מיזוג</u>

כדי לחשב את סיבוכיות האלגוריתם, נציג **<u>תחילה את הסיבוכיות מערכת משוואות רקורסיביות</u>**

נסמן ב $t(n)$ את הזמן הדרוש למיון-מיזוג מערך בן n איברים.

הזמן הנדרש לכל קריאה רקורסיבית הוא כמובן $t\left(\frac{n}{2}\right)$.

כבר הוכחנו כי מיזוג שני מערכים ממוינים באורך $n/2$ כל אחד אורך זמן לינארי dn כאשר d הוא קבוע הפרופורציה.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

הצגת סיבוכיות מיון-מיזוג על ידי מערכת משוואות רקורסיביות

נניח כי c הוא הזמן הקבוע הנדרש למיון מערך באורך 1 ונקבל:

$$t(n) = 2t\left(\frac{n}{2}\right) + dn$$

$$t(1) = c$$

כאשר d הוא קבוע הפרופורציה באלגוריתם המיזוג ו c הוא הזמן הקבוע הנדרש למיון מערך באורך 1.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

סיבוכיות מיון מיזוג

משפט 3.6

סיבוכיות מיון מיזוג היא $\Theta(n\log n)$.

הוכחה

מטעמים טכניים, נניח כי $n = 2^m$ עבור m שלם כלשהו. (שימו לב: $m = \log n$)

נוכיח באינדוקציה על m כי

$$t(2^m) = a2^m + dm2^m$$

עבור a שלם כלשהו ו- d הוא הקבוע המופיע במשוואות הרקורסיה.

בסיס ($n = 2^m = 1$,$m = 0$)

במקרה זה, נתון כי $t(1) = c$ והטענה נובעת מייד.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

צעד האינדוקציה

נניח כי עבור $n = 2^m$ מתקיים

$$t(2^m) = a2^m + dm2^m$$

נתבונן ב $t(2^{m+1})$. לפי המשוואה הרקורסיבית מתקיים:

$$t(2^{m+1}) = 2t(2^m) + d\,2^{m+1}$$

נציב במשוואה את הביטוי עבור $t(2^m)$ מטענת האינדוקציה ונקבל:

$$t(2^{m+1}) = 2\left[a2^m + dm2^m\right] + d\,2^{m+1}$$
$$= a2^{m+1} + dm2^{m+1} + d\,2^{m+1}$$
$$= a2^{m+1} + d(m+1)2^{m+1}$$
$$= an + dn\log n$$

מש"ל

© כל הזכויות שמורות לפרופסור עמוס ישראלי

מיון מיזוג - סיכום

מיון מיזוג משיג סיבוכיות $\Theta(n\log n)$. בהרצאה הבאה, אנו נוכיח כי זוהי הסיבוכיות האופטימלית עבור כל אלגוריתם מיון **מבוסס השוואות**. אלגוריתם מיון נוסף המשיג סיבוכיות זו הוא **מיון ערמה (Heap Sort)** שאותו לא נלמד בקורס.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

<u>סיכום ההרצאה</u>

פתחנו את ההרצאה בהצגת שיטת **הפרד ופתור** המהווה בסיס לבניית אלגוריתמים רקורסיביים.

בהמשך ההרצאה, הדגמנו את השימוש בשיטה על ידי בניית אלגוריתם **מיון באמצעות מיזוג** .

המשכנו את ההרצאה בהוכחת נכונות האלגוריתם תוך שימוש באינדוקציה על **אורך הקלט**.

בסיום ההרצאה, הוכחנו כי סיבוכיות האלגוריתם היא $\Theta(n\log n)$. זהו אלגוריתם המיון שהסיבוכיות שלו היא הטובה ביותר מבין כל האלגוריתמים שנכיר.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

<u>תרגילים</u>

<u>תרגיל3.1</u>

נתונים הקוד לאלגוריתם *merge* ומשפט הנכונות של האלגוריתם.

$$
\begin{array}{l}
merge(A,B)\\
\quad A[m+1]\leftarrow\infty, B[n+1]\leftarrow\infty\\
\quad ap\leftarrow 1;\ bp\leftarrow 1;\ cp\leftarrow 1;\\
\quad \textbf{while }(cp<m+n+1)\\
\qquad \textbf{if } A[ap]\le B[bp]\\
\qquad\quad C[cp]\leftarrow A[ap];\ ap++\\
\qquad\ \textbf{else } C[cp]\leftarrow B[bp];\ bp++\\
\qquad \textbf{end if}\\
\qquad cp++\\
\quad \textbf{end while}\\
\textbf{out}(C)
\end{array}
$$

© כל הזכויות שמורות לפרופסור עמוס ישראלי

הוכיחו את משפט הנכונות ונתחו את סיבוכיות האלגוריתם.

<u>משפט 3.8</u>

לכל $1\le i\le m+n$, לאחר i מעברים בלולאה הראשית, המערך $C[1..i]$ מכיל תמורה ממוינת של i האיברים הקטנים ביותר מבין איברי המערכים A ו-B. המצביע pa (בהתאמה pb) מצביע על האיבר הראשון במערך A (בהתאמה B) שטרם הועבר למערך C. ערכו של המצביע pc הוא $i+1$.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

הרצאה 4: מיון מהיר (Quicksort)

הרצאה זו מוקדשת לאלגוריתם **מיון מהיר (Quick Sort)**.

מיון מהיר מקובל היום כאלגוריתם המיון ש**מהירותו הממוצעת** היא הגבוהה ביותר מבין כל אלגוריתמי המיון המוכרים.

בנוסף, אלגוריתם זה, קל ביותר לתכנות. כתוצאה, רוב המערכות המציעות אלגוריתם מיון מערכתי משתמשות במיון מהיר.

חסרוננו העיקרי של המיון המהיר: סיבוכיות הזמן במקרה הגרוע ביותר היא $\Theta\left(n^2\right)$.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

תכונות וסימונים

1. לאורך כל ההרצאה אנו מניחים כי הקלט למיון הוא המערך A ובו n איברים **נבדלים** (שונים זה מזה). התאמת האלגוריתם למקרה שבמערך הקלט יש כמה איברים שווים זה לזה אינה קשה.
2. הרעיון במיון מהיר הוא לחסוך ככל האפשר בצעדי חישוב. כדי לעשות זאת, נבצע את כל החישובים על תוך שימוש במערך A.
3. לכל $1 \le i \le j \le n$ נסמן את תת המערך A, בין המשתנה i למשתנה j על ידי $A\left[i...j\right]$

© כל הזכויות שמורות לפרופסור עמוס ישראלי

תיאור האלגוריתם

1. בחר איבר כלשהו מן המערך כ**ציר (Pivot)**.
2. השתמש בציר כדי לחלק את אברי המערך לפי גודלם: כל האיברים הקטנים מן הציר בתת המערך התחתון. כל האיברים הגדולים מן הציר או שווים לו (כולל הציר עצמו) בתת המערך העליון.
3. הפעל מיון מהיר באופן רקורסיבי על כל אחד מתת המערכים.
4. האלגוריתם מתבצע **במקום**, כלומר בתוך מערך הקלט A.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

מיון מהיר – קוד האלגוריתם

להלן קוד האלגוריתם.

```
quicksort(A,i,j)
  If (i = j) return
   p ← choosepivot(A,i,j)
   boundary ←
              partition(A,i,j,p)
  quicksort(A,i,boundary − 1)
  quicksort(A,boundary,j)
  out(A)
```

אלגוריתם 1.4: מיון מהיר (Qick Sort)

© כל הזכויות שמורות לפרופסור עמוס ישראלי

הערות

1. השיטה הרקורסיבית *QuickSort* מקבלת כקלט תת מערך של מערך הקלט A .
2. גבולות המערך מצויינים על ידי אינדקס האיבר הנמוך ביותר, i, ואינדקס האיבר הגבוה ביותר j .
3. הקריאה הראשונה לשיטה היא $QuickSort\left(A,1,n\right)$.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

בחירת הציר

השיטה לבחירת הציר היא מאוד פשוטה: נבחר כציר את האיבר הגדול מבין שני איברי המערך הראשונים. בדרך זו מבטיחים כי בכל אחד מתת המערכים יהיה לפחות איבר אחד:

- האיבר **הקטן** מבין השניים, בחלק התחתון.
- האיבר **הגדול** מביניהם (הציר עצמו), בחלק העליון.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

השיטה *Choosepivot*

להלן הקוד:

$$
\begin{array}{l}
Choosepivot\left(A,i\right)\\
\quad out\left(\max\left(A\left[i\right],A\left[i+1\right]\right)\right)
\end{array}
$$

אלגוריתם 4.3: השיטה *choosepivot*

© כל הזכויות שמורות לפרופסור עמוס ישראלי

חלוקת ציר.

חלוקת ציר היא שיטה המקבלת שני ארגומנטים:

4. מערך הקלט A ובו n איברים **נבדלים**.
5. איבר ממערך הקלט הנקרא **ציר** (באנגלית **pivot**).

השיטה מחלקת את איברי מערך הקלט כמפורט להלן:

© כל הזכויות שמורות לפרופסור עמוס ישראלי

תכונות חלוקת הציר

חלוקת הציר מחלקת את איברי מערך הקלט לחלק תחתון (שמאלי) ולחלק עליון (ימני) כך שמתקיים: החלק התחתון מכיל את כל איברי המערך הקטנים ממש מן הציר. החלק העליון מכיל את כל הציר ואת כל איברי המערך הגדולים ממנו.

<u>התוצאה:</u> כל איברי החלק התחתון קטנים ממש מכל אחד מאיברי החלק העליון.

<u>אבחנה 4.4</u>

אם נבצע חלוקת ציר ואחר כך נמיין כל אחת מתת המערכים שהתקבלו, מערך הפלט יהיה ממוין.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

אלגוריתם חלוקה פשוט

האלגוריתם הפשוט דורש שני מעברים על מערך הקלט ושימוש במערך עזר.

במעבר הראשון על מערך הקלט, משוים כל אחד מאיבריו לציר וכל איבר הקטן מן הציר, או שוה לו מעבירים אל מערך העזר. את האיברים הגדולים מן הציר, או שווים לו, לא מזיזים. האיברים שהועברו מהווים את תת המערך התחתון.

במעבר השני, מעבירים את איברי הקלט הנותרים להמשך מערך העזר. איברים אלה מהווים את תת המערך העליון.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

אלגוריתם חלוקה פשוט (המשך)

במהלך ביצוע האלגוריתם מבצעים **שני מעברים** על הקלט. סיבוכיות הזמן היא $\Theta(n)$.

שאלות

האם אפשר לשפר את הסיבוכיות?

האם אפשר לשפר את זמן הביצוע?

תרגיל: הראו כיצד אפשר לוותר על המעבר השני.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

אלגוריתם חלוקה יעיל – תיאור

אפשר לבצע את אלגוריתם החלוקה ללא שימוש במערך עזר (in place) תוך חסכון בהזזת איברים ממקומם. בדרך זו חוסכים הן בזמן והן במקום, אך החסכון הוא בגורם קבוע בלבד.

החלוקה נערכת בשלבים: בכל שלב, מזהים שני איברים אשר אינם נמצאים בחלק המתאים. האיבר הראשון, נמצא בחלק התחתון, אך הוא גדול או שווה מן הציר. האיבר השני, נמצא בחלק העליון אך הוא קטן מן הציר. לאחר מציאת זוג האיברים מחליפים ביניהם, על ידי השיטה *Swap*, והשלב נגמר.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

באלגוריתם זה, משתמשים בשני מחוונים המכונים l ו-r.

לפני תחילת החלוקה, l מצביע אל האיבר הראשון (השמאלי ביותר) של המערך ו-r מצביע אל האיבר האחרון (הימני ביותר) של המערך.

בכל שלב, l מקודם ימינה עד שנמצא איבר הגדול מן הציר או שוה לו. לאחר מכן, r מקודם שמאלה עד שנמצא איבר קטן ממש מן הציר. כאשר נמצאו שני איברים כאלה, בודקים האם l ו-r חלפו זה על פני זה, כלומר האם $l \geq r$, אם כן, החלוקה הסתיימה. אם לא, מבצעים החלפה (*Swap*) בין $A[l]$ לבין $A[r]$ וממשיכים לשלב הבא.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

<u>חלוקה במקום - הדגמה</u>

קלט:

שימו לב: הציר הוא 4

1	4	9	5	3	1	2	4	8	2	7	3	2	1
l													r

שלב 1.1: זיהוי האיברים להחלפה

1	4	9	5	3	1	2	4	8	2	7	3	2	1
	l												r

שלב 1.2: ביצוע ההחלפה

1	1	9	5	3	1	2	4	8	2	7	3	2	4
	l												r

שלב 2.1: זיהוי האיברים להחלפה

1	1	9	5	3	1	2	4	8	2	7	3	2	4
		l										r	

שלב 2.2: ביצוע ההחלפה

1	1	2	5	3	1	2	4	8	2	7	3	9	4
		l										r	

© כל הזכויות שמורות לפרופסור עמוס ישראלי

שלב 3.1: זיהוי האיברים להחלפה

1	1	2	5	3	1	2	4	8	2	7	3	9	4
			l								r		

שלב 3.2: ביצוע ההחלפה

1	1	2	3	3	1	2	4	8	2	7	5	9	4
			l								r		

שלב 4.1: זיהוי האיברים להחלפה

1	1	2	3	3	1	2	4	8	2	7	5	9	4
								l	r				

שלב 4.2: ביצוע ההחלפה

1	1	2	3	3	1	2	2	8	4	7	5	9	4
							l	r					

שלב 5: סיום

החלק העליון מסומן, $boundary = l$.

1	1	2	3	3	1	2	2	8	4	7	5	9	4
							r	l					

© כל הזכויות שמורות לפרופסור עמוס ישראלי

<u>אלגוריתם החלוקה - הפרמטרים</u>

A - מערך הקלט.

i ו j - מציינים את גבולות מערך הקלט.

pivot - הציר, נמסר לשגרה על ידי השגרה הקוראת.

הערך המחזר: מציין את האינדקס בו מתחיל תת המערך העליון.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

אלגוריתם החלוקה - הקוד

$Partition(A,i,j,pivot)$
$l \leftarrow i; r \leftarrow j$
do forever
  while $(A[l] < pivot)\ l++$
  while $(A[r] \geq pivot)\ r--$
  if $(l \geq r)\ \ out(l)$
    else $swap(A[l], A[r])$
  end if
end do
end

אלגוריתם 4.5: השיטה *Partition*

© כל הזכויות שמורות לפרופסור עמוס ישראלי

הוכחת סיום אלגוריתם החלוקה

טענה 4.6

אם קוראים לאלגוריתם $partiton$ כאשר $l < r$ האלגוריתם מסתיים.

הוכחה

בכל מעבר בלולאה הראשית, למעט אולי במעבר הראשון מתקיים:

- המשתנה l גדל לפחות ב-1
- המשתנה r קטן לפחות ב-1

מכאן, בכל מעבר בלולאה הראשית, המרחק בין l ל-r קטן. מאחר שתנאי הסיום הוא $l \geq r$, סיום האלגוריתם מובטח.

מש"ל

© כל הזכויות שמורות לפרופסור עמוס ישראלי

הוכחת נכונות אלגוריתם החלוקה

טענה 4.7

בכל מעבר בלולאה הראשית, מיד לאחר ביצוע שתי הלולאות הפנימיות מתקיימים שני הביטויים הבאים:

1. כל האיברים בתחום $A[i...l-1]$ קטנים מן הציר. $A[l] \geq pivot$.
2. כל האיברים בתחום $A[r+1...j]$ גדולים מן הציר או שווים לו. $A[r] < pivot$.
3. $l \neq r$

שימו לב: במהלך המעבר הראשון יתכן כי התחומים $A[i...l-1]$, ו-$A[r+1...j]$ ריקים.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

טענה 4.7 (המשך)

בכל מעבר בלולאה הראשית, לאחר ביצוע החילוף, מתקיימים שני הביטויים הבאים:

4. כל האיברים בתחום $A[i...l]$ קטנים מן הציר.
5. כל האיברים בתחום $A[r...j]$ גדולים מן הציר או שווים לו.

תרגיל: הוכיחו את נכונות חמשת השמורות בעזרת מעקב.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

<u>טענה 4.8</u>

כאשר ביצוע האלגוריתם מסתיים, $l = r + 1$.

<u>הוכחת הטענה</u>

לפי תנאי הסיום מתקיים $l \geq r$. לפי שמורה 3, לא ייתכן כי $l = r$.

אפשר להוכיח, באמצעות מעקב, כי מייד לאחר שאחד המצביעים חולף על פני השני, האלגוריתם עוצר.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

<u>משפט 4.9</u>

בסיום אלגוריתם החלוקה:

- כל איברי המערך עד $boundary - 1$ קטנים מן הציר.
- כל איברי המערך מ$boundary$ והלאה גדולים מן הציר או שווים לו.

<u>הוכחה</u>

השיטה $Partition$ מחזירה את הערך של המצביע l. נכונות האלגוריתם נובעת מנכונות שמורות 1 ו-2.

סיבוכיות $Partition$ היא **<u>לינארית</u>**.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

<u>מיון מהיר - הוכחת נכונות</u>

<u>משפט</u>

בסיום ביצוע אלגוריתם המיון המהיר על מערך A , מערך הפלט ממוין.

<u>הוכחה</u>

ההוכחה באינדוקציה על אורך מערך הקלט A.

<u>בסיס</u> $|A| = 1$

במקרה זה, קל לראות כי שגרת המיון אינה מבצעת שום פעולה וכמובן מערך הפלט ממוין.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

<u>צעד האינדוקציה</u>

נניח כי אלגוריתם $QuickSort$ ממין כל מערך באורך קטן או שווה n ונוכיח לגבי מערך A באורך $n+1$.

נניח כי האלגוריתם מופעל על A.

בתחילת הביצוע נבחר ציר ומתבצעת חלוקה.

לפי הטענה בדבר נכונות החלוקה, לאחר ביצוע החלוקה, מערך הפלט מחולק לשני חלקים לא ריקים כאשר איברי תת המערך העליון גדולים כולם מאיברי תת המערך התחתון.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

צעד האינדוקציה (המשך)

לאחר ביצוע החלוקה, מופעל מיון מהיר על כל אחד מתת המערכים. מאחר ששני תת המערכים אינם ריקים, אורך כל תת מערך קטן או שווה n. אנו נמצאים בתנאי טענת האינדוקציה, ומן הטענה נובע כי בסיום ההפעלה הרקורסיבית, שני תת המערכים ממוינים. מאחר שכל איברי תת המערך התחתון קטנים מכל איברי תת המערך העליון אפשר להסיק כי לאחר סיום הקריאות הרקורסיביות, המערך כולו ממוין.

מש"ל.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

מיון מהיר - סיבוכיות זמן

כל ביצוע של חלוקה הוא ליניארי באורך המערך המחולק וסיבוכיות הזמן תלויה ישירות בטיב הצירים שנבחרו.

במקרה הגרוע ביותר, הציר הנבחר הוא האיבר הגדול ביותר (או הקטן ביותר) במערך הממוין. במקרה זה המערך מתחלק לחלק עליון בגודל 1 ולחלק תחתון בגודל $n-1$.

אם מקרה זה חוזר על עצמו, זמני ריצת שלבי המיון הם:

$$n, n-1, \ldots, 2, 1$$

בסך הכל, במקרה הגרוע ביותר, סיבוכיות הזמן היא $\Theta(n^2)$.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

הערות

1. בנספח מס' 1 להרצאה, אנו מוכיחים כי הסיבוכיות הממוצעת של מיון מהיר היא $T(n) = O(n \log n)$.
2. אפשר להראות כי המקרה הטוב ביותר מתקבל כאשר בכל שלב, כל קובץ מתחלק בדיוק ל2. במקרה זה מתקיים:

$$T(n) = 2T\left(\frac{n}{2}\right) + cn$$

פתרון משוואה זו הוא $\Theta(n \log n)$ ולכן $T(n) = \Theta(n \log n)$.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

3. קריאות רקורסיביות גורמות לפעולות מחסנית רבות שהן אינן יעילות עבור ערכי n קטנים. לכן, עבור $n<10$ (למשל) מוצע להשתמש במיון ריבועי כגון מיון בועות.
4. בספר מוצעת גירסה שונה למיון מהיר. בגירסה זו, הציר נבחר באופן אקראי ומוכחת תוצאה דומה לסיבוכיות הממוצעת.
5. בחיים האמיתיים מיון מהיר הוא **יעיל יותר** ממיון מיזוג. הסיבה לתופעה זו היא כי במיון מיזוג, בכל איטרציה מוזזים כל האיברים, בעוד שבמיון מהירת

© כל הזכויות שמורות לפרופסור עמוס ישראלי

בכל איטרציה מוזזים בממוצע, מחצית האיברים. סימולציות מראות כי מיון מהיר יעיל ממיון מיזוג **פי שתים**.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

סיכום

בהרצאה זו, הצגנו את אלגוריתם **מיון מהיר**, הוכחנו את נכונותו והראנו כי סיבוכיות האלגוריתם היא $\Theta\left(n^2\right)$. כאמור, **הסיבוכיות הממוצעת** של מיון מהיר היא $\Theta(n\log n)$. זהו אלגוריתם יעיל מאוד וגם פשוט מאוד והוא משמש פעמים רבות כאלגוריתם מערכת למיון.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

נספח מס' 1: ניתוח הסיבוכיות הממוצעת של מיון מהיר

נסמן ב- $T(n)$ את זמן הריצה הממוצע של מיון מהיר על קלט באורך n.
נניח כי קבוצת הקלטים עליהם מחושב זמן הריצה הממוצע היא קבוצת התמורות מעל $1,2,\ldots,n$ וכי כל התמורות הן שוות הסתברות.

תזכורת:

קבוצת התמורות S_n היא קבוצת הוקטורים באורך n שאיבריהם הם $1,2,\ldots,n$. ידוע כי $|S_n|=n!$.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

הסיבוכיות הממוצעת מתקבלת מן הנוסחה:

$$T(n)=\sum_{v\in S_n} p(v)t(v)$$

כאשר $t(v)$ מסמן את זמן הריצה של מיון מהיר על התמורה v. לכל תמורה $v\in S_n$ אנו מניחים כי $p(v)=1/n!$

ואז מקבלים כי $T(n)=\frac{1}{n!}\sum_{v\in S_n} t(v)$

כדי לחשב ביטוי זה, נחלק את הקבוצה S_n ל- $n-1$ תת קבוצות כדלהלן:

© כל הזכויות שמורות לפרופסור עמוס ישראלי

חלוקת המשנה

לכל i, $1 \le i \le n$, תהי $S_n^i \subset S_n$ קבוצת כל התמורות המקיימות: אורך תת המערך התחתון שאיבריו קטנים מן הציר, $A_<$, הוא i, ואורך תת המערך העליון, $A_\ge$, הוא $n-i$. כעת אפשר לרשום כי

$$T(n)=\sum_{v\in S_n} p(v)t(v)=\sum_{i=1}^{n-1}\frac{|S_n^i|}{n!}T\left(S_n^i\right)$$

כאשר $T\left(S_i^n\right)$ מסמן את זמן הריצה הממוצע בביצוע quicksort על כל המערכים בקבוצה S_n^i.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

טענה

לכל i, $1 \le i \le n$, זמן הריצה הממוצע עבור כל התמורות ב - S_n^i הוא:

$$T\left(S_n^i\right)=\underbrace{T(i)}_{\text{זמן דרוש למיון חלק תחתון}}+\underbrace{T(n-i)}_{\text{זמן רושמ למיון מע עליון}}+\underbrace{cn}_{\text{זמן דרוש לחלוקה}}$$

הסבר

$T(i)$ הוא הזמן הממוצע הנדרש למיון תמורה בת i איברים. נכונות הטענה נובעת מן העובדה שאם מתבצעת חלוקה, ההסתברות (הסיכוי) לכל אחת מן התמורות ב S_i, זהה.

ההסבר עבור $T(n-i)$ - זהה.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

הוכחה

נתבונן בקבוצת החלקים התחתונים המתקבלים מכל הוקטורים ב - S_n^i, שאורכם הוא תמיד i. אפשר להוכיח כי בקבוצה זאת, כל התמורות ב - S_i מתקבלות באותה השכיחות. מטענה זו נובע כי הזמן הממוצע הנדרש למיון החלק התחתון בקריאה הרקורסיבית הוא $T(i)$ (המוגדר כזמן הממוצע הנדרש למיון כל התמורות ב-S_i). באופן דומה, אפשר להוכיח כי $T(n-i)$ הוא הזמן הממוצע הנדרש בקריאה הרקורסיבית למיון החלק העליון.

ברור כי זמן החלוקה הוא $\Theta(n)$.

מש"ל.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

הסיבוכיות הממוצעת (המשך)

כדי לחשב את ערך $T(n)$ בדרך נכונה, עלינו לשים לב כי לכל $1 \le i < j \le n-1$ מתקיים $|S_n^i| \ne |S_n^j|$ מן הטענה הקודמת ומהגדרת הסיבוכיות הממוצעת נובע כי:

$$T(n)=\sum_{i=1}^{n-1}\left[\frac{|S_n^i|}{n!}\big(T(i)+T(n-i)\big)+cn\right]$$

$$=\sum_{i=1}^{n-1}\frac{|S_n^i|}{n!}\big(T(i)+T(n-i)\big)+$$

$$+\sum_{i=1}^{n-1}\frac{|S_n^i|}{n!}cn$$

$$=\sum_{i=1}^{n-1}\left[\frac{|S_n^i|}{n!}\big(T(i)+T(n-i)\big)\right]+cn$$

© כל הזכויות שמורות לפרופסור עמוס ישראלי

טענה

$$\frac{|S_n^i|}{n!}=\frac{2i}{n(n-1)}$$

הוכחה

נשים לב לעובדה ש כדי שמערך $A\in S_n$
יקיים $A\in S_n^i$, הציר חייב להיות $i+1$.
לכל i, $1\le i<n$, כדי ש-$i+1$ יבחר
כציר צריך להתקיים:

1. $A[1]=i+1$ ו-$A[2]<i+1$ **או**
2. $A[1]<i+1$ ו-$A[2]=i+1$.

נתבונן במקרה 1:
קיימות בדיוק i אפשרויות לבחירת
$A[2]$ ועבור כל בחירה כזו, יש

© כל הזכויות שמורות לפרופסור עמוס ישראלי

$(n-2)!$ אפשרויות עבור
$A[3],A[4],\ldots,A[n]$. מכאן נקבל כי
קיימים בדיוק $i(n-2)!$ המקיימים
את מקרה 1.
באופן סימטרי, קיימים בדיוק
$i(n-2)!$ המקיימים את מקרה 2.
ומשתי מסקנות אלה נקבל:

$$\frac{|S_n^i|}{n!}=\frac{2i(n-2)!}{n!}=\frac{2i}{n(n-1)}$$

הסבר

קל לראות כי $(n-2)!/n!=1/n(n-1)$

© כל הזכויות שמורות לפרופסור עמוס ישראלי

הסיבוכיות הממוצעת (המשך)

כעת נציב את הערכים שקיבלנו
בנוסחה לחישוב $T(n)$ ונקבל:

$$T(n)=\sum_{i=1}^{n-1}\left\{\frac{2i}{n(n-1)}[T(i)+T(n-i)+cn]\right\}$$
$$=\frac{1}{2}\sum_{i=1}^{n-1}\frac{2i}{n(n-1)}[T(i)+T(n-i)+cn]$$
$$+\frac{1}{2}\sum_{i=1}^{n-1}\frac{2(n-i)}{n(n-1)}[T(n-i)+T(i)+cn]$$

נכונות המעבר האחרון נובעת מן
העובדה שבשתי השורות האחרונות
מסכמים אותם האיברים **בסדר הפוך**.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

הסיבוכיות הממוצעת (המשך)

נמשיך בפיתוח:

$$\begin{aligned}T(n)=\sum_{i=1}^{n-1}\Bigg(&\frac{2iT(i)+2iT(n-i)}{n(n-1)}\\&+\frac{2nT(n-i)+2nT(i)}{n(n-1)}+\\&-\frac{2iT(n-i)+2iT(i)}{n(n-1)}\Bigg)+\\&\frac{1}{2n(n-1)}\sum_{i=1}^{n-1}2i\end{aligned}$$

כעת נשים לב שהביטוי העליון
בסוגריים בשורה הראשונה והשלישית
מבטלים זה את זה, וערכו של הביטוי
בשורה הרביעית הוא cn ונקבל:

© כל הזכויות שמורות לפרופסור עמוס ישראלי

$$T(n)=\frac{1}{2}\left(\sum_{i=1}^{n-1}\frac{2n\left[T(n-i)+T(i)\right]}{n(n-1)}\right)+cn$$

$$=\frac{1}{(n-1)}\left(\sum_{i=1}^{n-1}T(i)+T(n-i)\right)+cn$$

מאחר ששני הביטויים בתוך הסכום מסתכמים לאותו מספר נקבל: את נוסחת הנסיגה הבאה:

$$T(n)=\frac{2}{n-1}\left(\sum_{i=1}^{n-1}T(i)\right)+cn$$

© כל הזכויות שמורות לפרופסור עמוס ישראלי

<u>נוסחת הנסיגה</u>

הגענו לנוסחת נסיגה עבור הסיבוכיות הממוצעת של מיון מהיר:

$$T(n)=\frac{2}{n-1}\left(\sum_{i=1}^{n-1}T(i)\right)+cn$$

בשקפים הבאים נוכיח כי $T(n)=\Theta(n\log n)$.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

<u>למה</u>

סיבוכיות נוסחת הנסיגה

$$T(n)=\frac{2}{n-1}\sum_{i=1}^{n-1}T(i)+cn$$

עבור $c\geq 1$ היא $\Theta(n\log n)$.

<u>הוכחה</u>

נגדיר $d=4c$ ונוכיח באינדוקציה על n כי לכל n מתקיים, $T(n)\leq dn\log n$

בסיס: עבור $n=2$ ובהנחה ש $T(1)=1$,

$$T(2)=2T(1)+2c=2\left(\underbrace{1}_{=T(1)}+c\right)\leq$$

$$\leq 8c=2d\log 2$$

© כל הזכויות שמורות לפרופסור עמוס ישראלי

צעד: נניח כי לכל $i<n$ מתקיים $T(i)\leq di\log i$ ונתבונן ב $T(n)$:

$$T(n)=\frac{2}{n-1}\sum_{i=1}^{n-1}T(i)+cn$$

כעת נשתמש בהנחה ונקבל:

$$T(n)\leq\frac{2d}{n-1}\sum_{i=1}^{n-1}i\log i+cn$$

כדי להעריך את הסכום, נפרק אותו לשני חלקים ונקבל:

$$T(n)\leq\frac{2d}{n-1}\left(\sum_{i=1}^{n/2}i\log\frac{n}{2}+\sum_{i=n/2+1}^{n-1}i\log n\right)+cn=$$

ומאחר ש $\log(n/2)=\log n-1$ נקבל:

© כל הזכויות שמורות לפרופסור עמוס ישראלי

$$= \frac{2d}{n-1}\left(\sum_{i=1}^{n/2} i(\log n - 1) + \sum_{i=n/2+1}^{n-1} i\log n\right) + cn =$$

$$= \frac{2d}{n-1}\left(\sum_{i=1}^{n-1} i\log n - \sum_{i=1}^{n/2} i\right) + cn$$

$$= \left(\frac{2d\log n}{n-1}\right)\left(\frac{n(n-1)}{2}\right) -$$

$$\left(\frac{2d}{n-1}\right)\left(\frac{\frac{n}{2}\left(\frac{n}{2}-1\right)}{2}\right) + cn$$

$$= dn\log n - \frac{dn^2}{4(n-1)} - \frac{dn}{2(n-1)} + cn$$

© כל הזכויות שמורות לפרופסור עמוס ישראלי

$$\le dn\log n - \frac{dn^2}{4n} - \frac{dn}{2n} + cn$$

$$= dn\log n - \Bigg(\underbrace{\frac{d}{4} - c}_{=0}\Bigg)n - \frac{d}{2} \le dn\log n$$

מש״ל

© כל הזכויות שמורות לפרופסור עמוס ישראלי

הרצאה 5: סיבוכיות של בעיות - סיבוכיות בעית המיון כדוגמא

בהרצאה זו, נעסוק ב**סיבוכיות של בעיות**. בתחילת ההרצאה, נגדיר סיבוכיות של בעיות באופן כללי וכדוגמא נחשב את סיבוכיות בעית המיזוג.

עיקר ההרצאה, יוקדש לניתוח סיבוכיות בעית המיון. במהלכה נוכיח חסם תחתון $\Omega(n\log n)$ לבעית המיון, עבור אלגוריתמים מבוססי השוואות. בסיכום ההרצאה ונסיק כי סיבוכיות בעיית המיון, עבור אלגוריתמי מיון **מבוססי השוואות** היא $\Theta(n\log n)$.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

סיבוכיות של בעיות

תהי P בעיה חישובית כלשהי.

הסיבוכיות של P היא הסיבוכיות של האלגוריתם היעיל ביותר (מהיר ביותר) לפתרון P.

שימו לב: הכוונה אינה לסיבוכיות האלגוריתם יעיל ביותר **המוכר לנו** אלא לסיבוכיות האלגוריתם היעיל ביותר האפשרי.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

איך מחשבים סיבוכיות של בעיה?

תהי P בעיה חישובית כלשהי. כדי לקבוע את הסיבוכיות של P, אנו צריכים להוכיח שני חסמים **הדוקים** על הסיבוכיות: חסם עליון וחסם תחתון. במלים אחרות: עלינו להוכיח כי קיימת פונקציה $f(n)$ המקיימת:

חסם תחתון

סיבוכיות **כל אלגוריתם** לפתרון P היא $\Omega(f(n))$, כלומר הסיבוכיות **חסומה מלמטה** על ידי $f(n)$.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

חסם עליון

קיים אלגוריתם Al לפתרון P המקיים: סיבוכיות Alהיא $O(f(n))$, כלומר הסיבוכיות **חסומה מלמעלה** על ידי $f(n)$.

שימו לב

החסם התחתון מתייחס לסיבוכיות **כל אלגוריתם** לחישוב P. החסם העליון מתייחס ל**קיום אלגוריתם יחיד** לחישוב P.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

בעיות בסיבוכיות לינארית

נציג כעת שתי מחלקות גדולות של בעיות חישוביות שהסיבוכיות שלהן ניתנת לחישוב.

כפי שכבר אמרנו: כדי לקבוע סיבוכיות של בעיה, P עלינו להוכיח **חסם תחתון** על סיבוכיות כל האלגוריתמים לפתרון P.

כאשר קיים אלגוריתם לינארי, הוכחה כזאת אינה קשה.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

דוגמא 5.1 - סיבוכיות בעית המיזוג

סיבוכיות בעיית המיזוג של שני מערכים ממוינים בגודל n ו-m בהתאמה, היא $\Theta(n+m)$.

הוכחה

כבר ראינו כי קיים אלגוריתם מיזוג שהסיבוכיות שלו היא $\Theta(n+m)$.

כדי להוכיח את המשפט, עלינו להראות כי **לא קיים אלגוריתם מיזוג שהסיבוכיות שלו נמוכה מ-**$n+m$.

מאחר ש**כל אלגוריתם מיזוג יוצר את וקטור הפלט** גדלו $n+m$ המשפט נובע מיידית.

מש"ל

© כל הזכויות שמורות לפרופסור עמוס ישראלי

דוגמא 4.2 - סיבוכיות בעית הסכום

סיבוכיות בעיית סכום האיברים במערך בגודל n, היא $\Theta(n)$.

הוכחה

ברור שקיים אלגוריתם לחישוב סכום מיזוג שהסיבוכיות שלו היא $\Theta(n)$.

כדי להוכיח את המשפט, עלינו להראות כי **לא קיים אלגוריתם מיזוג שהסיבוכיות שלו נמוכה מ-**n.

מאחר ש**כל אלגוריתם סכום חייב לבחון את כל אברי מערך הקלט** שגדלו n המשפט נובע מיידית.

מש"ל

המשפט הבא מכליל את שתי הדוגמאות שהבאנו:

© כל הזכויות שמורות לפרופסור עמוס ישראלי

משפט 4.3

תהי P בעיה חישובית.

1. אם לכל n קיים קלט בגודל n, כך שהפלט המתאים הוא בגודל n, או בגודל $\Omega(n)$, הסיבוכיות של כל אלגוריתם לפתרון P, היא $\Omega(n)$.
2. אם אם לכל n טבעי קיים קלט I_n, $|I_n|=n$, כך שכל אלגוריתם לפתרון P חייב לבחון כל איבר ב-I_n, הסיבוכיות של כל אלגוריתם לפתרון P, היא $\Omega(n)$.

הוכחה

ההוכחה מיידית ונשארת כתרגיל לקורא.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

מחלקות של בעיות בסיבוכיות לינארית

קימות אם כן, שתי מחלקות גדולות של בעיות בסיבוכיות לינארית:

1. מחלקת הבעיות שקיים אלגוריתם לינארי לפתרונן, ואשר יוצרות פלט שגדלו הוא $\Theta(n)$. לדוגמא: בעיית המיזוג.
2. מחלקת הבעיות שקיים אלגוריתם לינארי לפתרונן, וכל אלגוריתם נדרש לבחון את כל הקלט. לדוגמא: בעיית הסכום.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

חסם תחתון לסיבוכיות בעיית המיון

מה המעמד של בעיות שהאלגוריתם הטוב ביותר המוכר לנו הוא סופר לינארי (כלומר יותר מלינארי)?
בדרך כלל, הוכחת חסמים תחתונים לבעיות כאלה היא קשה ביותר.
די אם נציין כי עד כה, לא הצלחנו לקבוע את הסיבוכיות של רוב הבעיות החישוביות, שהאלגוריתם הטוב ביותר המוכר לפתרונן הוא סופר-לינארי.ה
בהרצאה זו, נכיר את מודל עץ ההחלטה, והשימוש בו, כדי לקבוע את סיבוכיות בעיית המיון מבוסס ההשוואות.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

סיבוכיות בעיית המיון

נתבונן בבעיית המיון. לכל מופע בגודל n, גודל הפלט הוא n ולכן, סיבוכיות הזמן של בעיית המיון היא $\Omega(n)$.
בעבר, האלגוריתם הטוב ביותר הידוע למיון היה מיון בועות (Bubble Sort) שזמן הריצה שלו הוא $\Theta(n^2)$.
בזמן זה, הידע האנושי בדבר סיבוכיות בעיית המיון היה:

$\Omega(n) \le$ סיבוכיות מיון $\le \Theta(n^2)$

במלים אחרות: פער אי הוודאות בנושא סיבוכיות מיון היה בין n לבין n^2.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

סיבוכיות בעיית המיון

פער אי הוודאות המוזכר לעיל נסגר על ידי שתי ההתפתחויות הבאות:

1. פותחו אלגוריתמי מיון יעילים יותר, כגון מיון בעזרת מיזוג (Merge-Sort) שהסיבוכיות שלהם היא $\Theta(n \log n)$.
2. בהרצאה זו נוכיח חסם תחתון הקרוי "**חסם תורת האינפורמציה**" של $\Omega(n \log n)$ על סיבוכיות הזמן של אלגוריתמי מיון מבוססי השוואות (זהו סוג מסוים של אלגוריתמי מיון).

אלו חסמים **הדוקים** (tight). כתוצאה נקבל: **סיבוכיות בעיית המיון מבוסס השוואות היא** $\Theta(n \log n)$.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

<u>הערות</u>

1. לאורך כל ההרצאה נניח כי כל איברי מערך הקלט הם נבדלים.
2. החסם **תופס** רק עבור אלגוריתמים מבוססי השוואות.
3. החסם **אינו** מוכיח כי כל הריצות הנן באורך $n\log n$. לדוגמא: מיון בועות עבור קובץ ממוין יתבצע בזמן לינארי.

נפתח את ההרצאה במספר הגדרות:

הגדרה 4.4: <u>תמורה מעל n איברים</u>

וקטור ובו n איברים שהם המספרים $1,2,\ldots,n$ מסודרים בסדר שרירותי.

הגדרה 4.5: <u>חבורת התמורות מעל n איברים</u>

קבוצת כל התמורות מעל n איברים, כלומר קבוצת כל הוקטורים שאיבריהם הם $1,2,\ldots,n$ בסדר כלשהו. קבוצה זו מכונה S_n. ידוע כי $|S_n|=n!$.

<u>הגדרה 4.6:</u>

1. יהי Al אלגוריתם כלשהו המקבל כקלט מערך A. **<u>השוואה</u>** ב-Al היא הביטוי $A[i]<A[j]$, כי משווים שניים מאיברי מערך הקלט. תוצאת ההשוואה יכולה להיות T או F.

<u>הערה 1:</u> אפשר כמובן להפוך את סדר הביטוי כלומר לכתוב $A[i]>A[j]$.

<u>הערה 2:</u> הביטוי $A[i]<0$ אינו מוגדר כהשוואה, כי משווים איבר ממערך הקלט עם קבוע מספרי.

<u>הגדרה 4.7:</u>

מערך A בגודל n שאיבריו מספרים שלמים הוא **<u>שקול</u>** לתמורה α אם A ו-α שומרים על אותם יחסי גודל.
התמורה α נקראת **<u>התמורה המייצגת</u>** את המערך A.

<u>לדוגמא:</u>

המערך 7,8,9 שקול לתמורה 1,2,3 והמערך 30,20,40 שקול למערך 2,1,3.

אלגוריתמי מיון מבוססי השוואות

הגדרה 4.8:

אלגוריתם מיון הוא **מבוסס השוואות** אם כל אחת מהחלטות הבקרה שלו מתקבלת אך ורק על ידי ביצוע השוואה.

הערה

כל אלגוריתמי המיון שלמדנו עד כה הם מבוססי השוואות.

שימו לב

כל חישוב של אלגוריתם מבוסס השוואות נקבע **אך ורק על פי תוצאות של השוואה**, לכן כל אלגוריתם כזה, מתנהג באופן זהה על כל המערכים **השקולים לתמורה מסוימת.**

© כל הזכויות שמורות לפרופסור עמוס ישראלי

תכונות אלגוריתמי מיון מבוססי השוואות

משפט 4.9

תהיינה $\sigma,\pi \in S_n$ שתי תמורות שרירותיות ויהי Al אלגוריתם מיון מבוסס השוואות כלשהו. תהי $c_1,c_2,...$ ($c'_1,c'_2,...$ בהתאמה). סדרת השוואות המתבצעות בחישוב Al עם σ (עם π בהתאמה). אזי

1. לא ייתכן כי $c_1,...,c_n = c'_1,...,c'_n$, וגם כל תוצאות ההשוואות בשתי הסדרות זהות.
2. קיים אינדקס מינימלי i_0, המקיים ש $c_{i_0} = c'_{i_0}$ אך תוצאות ההשוואות הללו מנוגדות.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

הוכחת 1

מאחר ש- Al הוא אלגוריתם מבוסס השוואות, כל החלטות הבקרה של האלגוריתם Al תלויות אך ורק בהשוואות הנערכות על ידי Al.

אם שתי סדרות ההשוואות זהות ואם כל תוצאות ההשוואות זהות, אזי כל פעולות האלגוריתם בשני החישובים זהות. במקרה זה מתקיים: לכל i, $1 \le i \le n$, האיברים $\sigma[i]$ ו-$\pi[i]$ מגיעים לאותו מקום בפלט.

ואפשרות זאת לא תיתכן מפני ש-Al הוא **אלגוריתם מיון.**

כדי להבין את המסקנה האחרונה טוב יותר נתבונן בדוגמא הבאה:

© כל הזכויות שמורות לפרופסור עמוס ישראלי

דוגמא

אם לדוגמא נתון

1	5	4	6	2	8	3	...	σ

1	5	7	6	2	8	3	...	π

אז לאחר המיון נקבל

1	2	3	4			...	$out(\sigma)$

1	2	3	7			...	$out(\pi)$

וזאת סתירה כי הפלט $out(\pi)$ **אינו ממוין.**

© כל הזכויות שמורות לפרופסור עמוס ישראלי

<u>הוכחת 2</u>

מאחר ש-Al הוא אלגוריתם מבוסס השוואות, סדרת צעדי האלגוריתם, עד לביצוע ההשוואה הראשונה זהה עבור כל תמורה עליה האלגוריתם מופעל.

לכל i, $1 \le i < n$ אם מתקיים $c_1,...,c_i = c'_1,...,c'_i$ אז כל החלטות הבקרה שיתקבלו לפני ההשוואה הבאה, זהות בשני הביצועים ומכאן נובע כי גם $c_{i+1} = c'_{i+1}$.

לפי סעיף 1, לא ייתכן כי $c_1,...,c_n = c'_1,...,c'_n$ וגם כל התשובות לשתי הסדרות זהות. מכאן נובע כי קיימת השוואה, נניח באינקס i_0,

© כל הזכויות שמורות לפרופסור עמוס ישראלי

המקיימת $c_{i_0} = c'_{i_0}$, אך התשובה להשוואה זו שונה בשתי הסדרות.

מש"ל

.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

<u>עץ החלטה</u>

בטרם ניגש להוכחת החסם התחתון עלינו להכיר מבנה נתונים תיאורטי בשם **עץ החלטה למיון** n **איברים** או בקיצור **עץ מיון ל-** n **איברים**.

להלן נגדיר עץ מיון בשני שלבים.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

<u>הגדרה 4.10: עץ מסומן</u>

<u>עץ מסומן</u> הוא עץ בינרי אשר כל אחד מן הצמתים שלו מסומן כדלהלן:

- כל צומת פנימי בעץ מסומן בהשוואה בין שנים מאיברי תמורת הקלט.
- שתי הקשתות היוצאות מצומת פנימי כזה מסומנות האחת ב-T והשניה ב-F.
- כל עלה בעץ מסומן על ידי תמורה $\pi \in S_n$. להלן נסמן את המסלול משורש העץ ועד לעלה המסומן בתמורה π בשם α_π.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

הגדרה 4.11: מסלול תקין

מסלול α_π בעץ מסומן T נקרא **תקין** אם התמורה π **מספקת** (מאמתת) כל אחת מן ההשוואות ב-α_π.

הגדרה 4.12: עץ החלטה

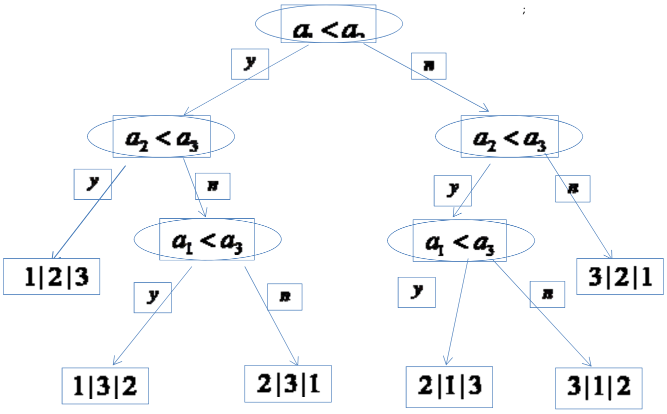
עץ החלטה למיון n איברים, הוא עץ מסומן T המקיים את הדרישות הבאות:

1. בעץ T יש בדיוק $n!$ עלים.
2. כל תמורה $\pi \in S_n$ מסמנת **עלה** **יחיד** בעץ T.
3. כל אחד מ - $n!$ המסלולים בעץ T הוא **תקין**.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

דוגמא: עץ החלטה עבור S_3

בעץ שלפנינו יש להתייחס לסימון a_i כאילו כתוב $A[i]$.



איור 4.1: עץ החלטה למיון 3 איברים

בטרם תמשיכו בקריאה, ודאו שכל המסלולים בעץ T הם **תקינים**.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

הערות

1. עץ החלטה אינו אלגוריתם מיון. זהו מודל תיאורטי בלבד.
2. אפשר להתייחס לעץ החלטה כאל **אלגוריתם לזיהוי תמורת הקלט**.
3. כל עץ החלטה ל - n איברים אמור לזהות כל אחת מתמורות הקלט. זיהוי תמורה π, מתבצע על ידי מעבר על המסלול α_π.
4. מאחר שמספר תמורות הקלט האפשריות הוא $n!$, לכל עץ החלטה למיון n איברים, יש $n!$ עלים.
5. בהנתן וקטור V, שכל איבריו **נבדלים** עץ המיון מזהה את התמורה המייצגת את V.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

חסם תחתון לסיבוכיות אלגוריתמים מבוססי השוואות למיון

להלן נוכיח כי הסיבוכיות של כל אלגוריתם מיון מבוסס השוואות היא $\Omega(n \log n)$, כלומר הסיבוכיות חסומה מלמטה על ידי $n \log n$.

במלים פחות מדויקות אפשר לומר כי סיבוכיות בעית המיון מבוסס ההשוואות היא $\Omega(n \log n)$.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

סקירת הוכחת החסם

הוכחת החסם תתקבל על ידי הוכחת הטענות הבאות:

טענה 1: כל אלגוריתם מבוסס השוואות למיון n איברים, Al_n, **משרה** עץ החלטה ל- n איברים, T_{Al_n} ובו $n!$ עלים.

טענה 2: סיבוכיות האלגוריתם Al_n, גדולה או שווה מעומק העץ המושרה T_{Al_n}.

טענה 3: הגובה של כל עץ החלטה למיון n איברים הוא $\Omega(n\log n)$.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

בניית עץ המיון המושרה

ההוכחה הפורמלית לטענה 1, מתווה אלגוריתם לבנית עץ החלטה.

אלגוריתם הבנייה, שיכונה להלן Bal

Bal מקבל כקלט, אלגוריתם מיון מבוסס השוואות Al_n ובונה את עץ ההחלטה T_{Al_n}.

שימו לב:

1. אלגוריתם הבניה, מקבל כקלט **אלגוריתם מיון.**
2. **מבנה** עץ ההחלטה המושרה, תלוי באלגוריתם המיון המשרה אותו.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

תאור האלגוריתם Bal

קלט: אלגוריתם מבוסס השוואות למיון n איברים - Al_n.

פלט: עץ החלטה למיון n איברים - T_{Al_n}.

אלגוריתם Bal בונה את T_{Al_n}, תוך כדי הרצת Al_n על כל $n!$ התמורות של n איברים.

לכל תמורה $\pi \in S_n$, האלגוריתם Bal הבנייה בונה את α_π על ידי **מעקב** אחר ביצוע האלגוריתם Al_n על הקלט π.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

דוגמא: עץ מיון המושרה על ידי מיון בועות לשלושה איברים

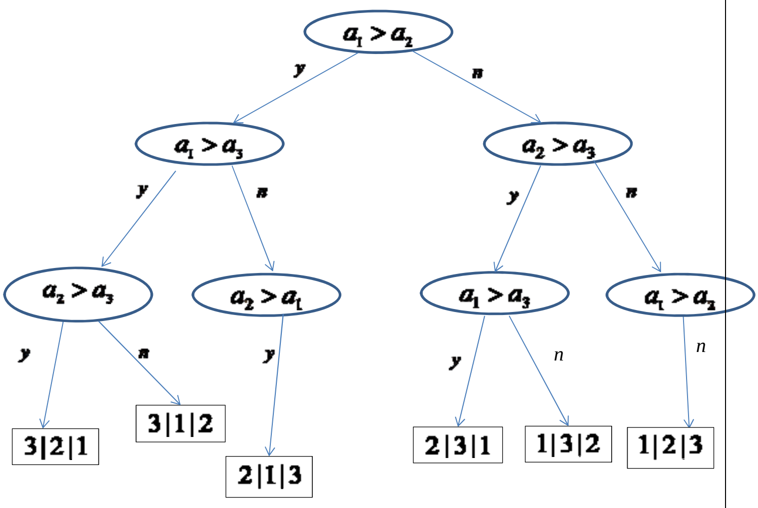
בטרם נציג את Bal, נדגים את פעולתו על ידי בניית עץ המיון T_{BS_3}, המושרה על ידי מיון בועות, בגרסה הראשונה, ל-3 איברים:

```
BubbleSort(A,3)
  for Lim ← 3 downto 2
    for j = 1 to Lim − 1
      if (A[j] > A[j+1])
           Swap(A[j], A[j+1])
    end for
  end for
end
```

איור 4.2: מיון בועות ל-3 איברים

© כל הזכויות שמורות לפרופסור עמוס ישראלי

העץ המושרה



איור 4.3: עץ ההחלטה המושרה על ידי אלגוריתם מיון בועות ל-3 איברים

נא בדקו שכל המסלולים הם תקינים.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

תיאור אלגוריתם *Bal* לבניית $\alpha(\Pi)$

אלגוריתם *Bal* עוקב אחר חישוב A_n על הקלט Π, לפי הנקודות המפורטות להלן:

1. כל הרצה, ללא תלות בתמורת הקלט, Π, נפתחת בהשוואת זוג מאיברי Π. השוואה זו מסמנת את שורש העץ T_{A_n}.
2. לאחר ביצוע ההשוואה הראשונה, אלגוריתם *Bal* עוקב אחרי α_Π על ידי בחירת הקשת המתאימה לתשובה שהתקבלה בחישוב A_n על Π (התלויה בΠ), כדלהלן:

© כל הזכויות שמורות לפרופסור עמוס ישראלי

אם ההשוואה מתקיימת, נבחרת הקשת המסומנת ב-y; אם לא- נבחרת הקשת המסומנת ב-n.

3. אם קשת שנבחרה, אינה מופיעה ב T_{A_n}, אלגוריתם *Bal* מוסיף אותה ל T_{A_n} וממשיך בבניית α_Π כאשר כל השוואה נוספת ש-A_n מבצע, תורמת צומת נוסף ל α_Π.
4. בסיום הרצת A_n על Π, אלגוריתם *Bal* מוסיף עלה ומסמנו על ידי Π.

שימו לב (הערה חשובה)

כל ההשוואות נעשות בין איברי תמורת הקלט ולא מתחשבים בחילופי האיברים הנעשים במהלך ביצוע האלגוריתם.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

משפט 4.4

אלגוריתם *Bal* בונה עץ השוואה למיון n איברים.

הוכחה

אלגוריתם *Bal* מפעיל את האלגוריתם שבקלט A_n על כל התמורות ב-S_n אחת אחרי השניה. לפי סעיף 1 במשפט 4.3, כל הפעלה כזו נפתחת בביצוע השוואה מסוימת, ללא תלות בתמורת הקלט. ההשוואה הזאת היא כמובן ההשוואה המופיעה בשורש העץ הנבנה. מהגדרת *Bal* נובע כי כל התמורה הראשונה שעליה מופעל A_n, בונה מסלול תקין בעץ. נניח כי עד לביצוע A_n על תמורה

© כל הזכויות שמורות לפרופסור עמוס ישראלי

כלשהי Π, אלגוריתם Bal בנה עץ שכל מסלוליו תקינים. לפי סעיף 2 במשפט 4.3 כאשר A_n מופעל על Π, לכל תמורה Σ, עליה הפעל A_n עד כה, המסלול הנבנה בעץ עבור Π, מבצע השוואה אחת הכלולה בביצוע עם Σ, אך תוצאת ההשוואה שונה. מכאן נובע כי המסלול הנבנה עבור Π **שונה מכל אחד מן המסלולים שנבנו בעץ עד כה.** כתוצאה מכך, המסלול הנבנה עבור Π הוא תקין וסך הכל אלגוריתם הבניה בונה בדיוק $n!$ מסלולים כלומר: **העץ הנבנה הוא עץ השוואה.** להלן נכנה את עץ ההשוואה הזה ב- T_{A_n}.

מש"ל

© כל הזכויות שמורות לפרופסור עמוס ישראלי

הוכחת טענה 2

טענה 2: סיבוכיות האלגוריתם A_n, גדולה או שווה מגובה העץ המושרה T_{A_n}.

ההוכחה מסתמכת על האלגוריתם לבנית T_{A_n}: לכל תמורה Π, המסלול $\alpha(\Pi)$ שומר את סדרת השוואות הנערכות במהלך החישוב על Π.

תהי Σ התמורה המסמנת את המסלול הארוך ביותר ב T_{A_n}. מאחר שחישוב A_n על Σ, כולל את כל ההשוואות המאוחסנות לאורך α_Σ ביחד עם צעדי חישוב נוספים, מספר הצעדים בביצוע A_n, בריצה עם Σ, גדול או שווה מאורכו של α_Σ. מכאן: סיבוכיות המקרה הגרוע ביותר של A_n גדולה או שווה מ $|\alpha_\Sigma|$. **מש"ל**

© כל הזכויות שמורות לפרופסור עמוס ישראלי

חסימת גובה עץ ההשוואה

הוכחת החסם לסיבוכיות בעיית המיון תושלם על ידי חסימת גובה עץ המיון.

משפט

יהי T עץ בינארי כלשהו עם n עלים. גבהו של T הוא $\Omega(\log n)$.

הוכחה

הוכחת המשפט מופיעה בנספח להרצאה.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

חסימת גובה העץ (המשך)

תוצאה

נזכר כי לכל עץ מיון יש בדיוק $n!$ עלים. מן המשפט שהוכחנו נובע כי גבהו של כל עץ מיון הוא:

$$\Omega(\log n!)$$

מה ערכו של $\log n!$?

אי השוויונות הבאים מעידים כי

$$\log n! = \Theta(n\log n)$$

$$n\log n \ge \log(n^n) \ge \log(n!) \ge$$

$$\ge \log\left(\left(\frac{n}{2}\right)^{\frac{n}{2}}\right) = \frac{n}{2}(\log n - \log 2)$$

© כל הזכויות שמורות לפרופסור עמוס ישראלי

חסימת גובה העץ (המשך)

מנוסחת Stirling לחישוב $n!$ נובע:

$$n! \approx \left(\frac{n}{e}\right)^n$$

$$\log n! \approx n(\log n - \log e) \approx$$

$$\approx n\log n - 1.44n$$

© כל הזכויות שמורות לפרופסור עמוס ישראלי

סיכום ההוכחה

לפי **טענה 1**, כל אלגוריתם מיון **מבוסס השוואות** למיון n איברים A_n משרה עץ מיון כלשהו, T_{A_n}.

מ**טענה 2** נובע כי גובה העץ המושרה T_{A_n}, הוא חסם תחתון על סיבוכיות המקרה הגרוע ביותר של האלגוריתם המשרה, A_n.

מן המשפט האחרון שהוכחנו, נובע כי גבהו המינימלי של עץ מיון ל-n איברים חסום על ידי

$$\Omega(\log n!) = \Omega(n\log n)$$

© כל הזכויות שמורות לפרופסור עמוס ישראלי

הוכחת חסמים תחתונים בעזרת רדוקציה

סיימנו את הוכחת הטענה כי סיבוכיות כל אלגוריתם השוואות למיון היא $\Omega(n\log n)$.

משפט זה מאפשר לנו הוכחה של חסמים תחתונים רבים בשיטת הרדוקציה. בשיטה זו, אפשר להוכיח כי עבור בעיה חישובית P, לא קיים אלגוריתם **מבוסס השוואות** לפתרון P בסיבוכיות קטנה ממש מ $n\log n$.

בשלב זה, נדגים את שיטת הרדוקציה על ידי הוכחת חסם תחתון על בעיה אופינית:

© כל הזכויות שמורות לפרופסור עמוס ישראלי

דוגמה: בעיית המערך הקמור

מערך קמור הוא מערך A שמספר איבריו זוגי והם מקיימים:

$$A[i] < A[i+1] \quad \text{if} \quad 1 \le i \le n/2$$

$$A[i] > A[i+1] \quad \text{if} \quad n/2+1 \le i \le n$$

תהי P הבעיה החישובית הבאה:

קלט: מערך A שאיבריו ניתנים להשוואה.

פלט: תמורה של איברי המערך A המהווה מערך קמור.

משפט

לא קיים אלגוריתם **מבוסס השוואות** לפתרון הבעיה P שהסיבוכיות שלו נמוכה מ-$\Theta(n\log n)$.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

הוכחת החסם התחתון בשיטת הרדוקציה

נניח בשלילה כי קיים אלגוריתם Al_n המקבל כקלט מערך A ובו n איברים בני השוואה ומארגנם למערך קמור, וכי סיבוכיות האלגוריתם Al_n, $f(n)$ מקיימת $f(n) < n\log n$.

להלן נציג את אלגוריתם *Stira* אשר פותר את בעיית המערך הקמור תוך שימוש באלגוריתם Al_n כשגרת עזר.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

אלגוריתם *Stira* - הקוד

$Stira(A)$

1. $A \leftarrow Al_n(A)$
2. for $i \leftarrow n/2+1$ to n

   $B[i] \leftarrow A[n-i+1]$
3. $merge(B[1..n/2], B[n/2+1.$
4. $out(B)$

איור 4.4: אלגוריתם *Stira*

© כל הזכויות שמורות לפרופסור עמוס ישראלי

אלגוריתם *Stira* - תיאור

מהלך האלגוריתם הוא כדלהלן:

בתחילה, האלגוריתם מפעיל את Al_n והופך את מערך הקלט A למערך קמור.

לאחר מכן, המערך הקמור מוסב למערך ממוין על ידי **מיזוג** שני החצאים של המערך A למערך ממוין.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

המשך הוכחת החסם התחתון

נכונות *Stira* נובעת מיידית מתכונות Al_n ומנכונות *Merge*.

קל לראות כי *Stira* הוא אלגוריתם מונחה השוואות.

סיבוכיות שלב 1 היא $f(n)$ וסיבוכיות שאר השלבים היא לינארית.

מכאן נובע כי סיבוכיות האלגוריתם $Stira(A)$ היא $f(n) < n\log n$.

כעת נשים לה לכך ש *Stira* הוא אלגוריתם מיון מבוסס השוואות שהסיבוכיות שלו קטנה ממש

© כל הזכויות שמורות לפרופסור עמוס ישראלי

מ-$\Theta(n\log n)$, וזאת כמובן **<u>סתירה</u>** **<u>לחסם התחתון שהוכחנו</u>** בחלקה הראשון של ההרצאה.

מש״ל

© כל הזכויות שמורות לפרופסור עמוס ישראלי

<u>סיבוכיות בעיית המערך הקמור</u>

בטרם נציג את שיטת הרדוקציה באופן כללי, נחסום את סיבוכיות בעיית המערך הקמור, על ידי הצגת אלגוריתם מבוסס השוואות לפתרון הבעיה בסיבוכיות $n\log n$.

<u>משפט</u>

סיבוכיות הבעיה P, עבור אלגוריתמים מבוססי השוואות,היא $\Theta(n\log n)$.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

<u>הוכחה</u>

כדי להוכיח את המשפט עלינו להציג אלגוריתם מבוסס השוואות הפותר את הבעיה בסיבוכיות $\Theta(n\log n)$.

בהקשר זה, נכנה לעתים אלגוריתם כזה בכינוי **<u>חסם עליון</u>**, וזאת בניגוד להוכחת **<u>החסם התחתון</u>**.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

<u>הוכחת החסם העליון: האלגוריתם</u>

1. $A \leftarrow MergeSort(A)$
2. for $i \leftarrow 1$ to $n/2$
   $B[i] \leftarrow A[i]$
3. for $i \leftarrow 1$ to $n/2$
   $B[n/2+i] \leftarrow A[n-i+1]$
4. $out(B)$

קל לראות כי מערך הפלט B הוא קמור וכי סיבוכיות האלגוריתם המוצג היא $\Theta(n\log n)$.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

שיטת הרדוקציה

כעת נציג את שיטת הרדוקציה באופן כללי. להלן פירוט שלבי השיטה:

1. נניח כי ברצוננו להוכיח כי הסיבוכיות של כל אלגוריתם מבוסס השוואות לפתרון בעיה חישובית P היא $\Omega(n\log n)$.
2. נניח בשלילה כי קיים אלגוריתם מבוסס השוואות Al_p, לפתרון הבעיה P בסיבוכיות $f(n)$ המקיימת, $f(n) < n\log n$.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

שיטת הרדוקציה (המשך)

3. נראה כי אפשר להשתמש באלגוריתם Al_p, כדי לקבל אלגוריתם מיון מבוסס השוואות שהסיבוכיות שלו היא $\Omega(n\log n)$ סתירה לחסם התחתון שהוכחנו.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

הדרך לקבלת אלגוריתם המיון

אלגוריתם המיון שבו משתמשים בשיטת הרדוקציה מתקבל בדרך הבאה:

1. הסב את הקלט I_s לקלט לבעיה P, I_p.
   (הערה: במקרים מסוימים, שלב זה מיותר)
2. הפעל את אלגוריתם Al_p על הקלט I_p וקבל את הפלט O_p.
3. הסב את הפלט O_p לפלט $O_s = sort(I_s)$.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

מסקנה

אם שלושת השלבים הללו מתבצעים באלגוריתמים מבוססי השוואות שהסיבוכיות שלהם קטנה ממש מ $n\log n$ - קיבלנו אלגוריתם מיון מבוסס השוואות בסיבוכיות $f(n) < n\log n$ בסתירה למשפט שהוכחנו.
מכאן אפשר להסיק שהנחתנו הראשונה בדבר קיומו של Al_n אינה נכונה.

מש"ל

© כל הזכויות שמורות לפרופסור עמוס ישראלי

שיטת הרדוקציה (המשך)

לאור העובדה שאנו משתמשים באלגוריתם לפתרון P כדי להציג פתרון לבעיית המיון, אנו אומרים כי הבעיה P **ניתנת לרדוקציה** לבעיית המיון.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

סיכום

בהרצאה זו, עסקנו בסיבוכיות של בעיות חישוביות. פתחנו בהגדרת מושג **סיבוכיות של בעיה חישובית**. הסברנו כי, בניגוד להוכחת סיבוכיות של אלגוריתם, הוכחת סיבוכיות של בעיה כרוכה בהוכחת **חסם תחתון** לסיבוכיות **כל אלגוריתם אפשרי לפתרון הבעיה.**

לאחר מכן, הוכחנו חסמים פשוטים לסיבוכיות של בעיות **לינאריות.**

עיקר ההרצאה הוקדש לניתוח של סיבוכיות בעית **מיון מבוסס השוואות**. במסגרת זו הוכחנו כי סיבוכיות בעיה זו היא $\Theta(n\log n)$. לסיום ההרצאה

© כל הזכויות שמורות לפרופסור עמוס ישראלי

פיתחנו את שיטת הרדוקציה להוכחת חסמים תחתונים לבעיות אחרות. הדגמנו את שיטת הרדוקציה בעזרת הוכחה כי סיבוכיות בעית חישוב וקטור קמור היא $\Theta(n\log n)$.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

נספח: עצים בינאריים

נפתח בחזרה על כמה מושגים הקשורים לעצים בינאריים כלליים:

צומת - רשומה המכילה נתונים ומחוונים אל צמתים אחרים. כל מחוון לצומת אחרת נקרא גם **קשת.**

שימו לב: יתכנו צמתים ללא מידע או ללא מחוונים.

עץ בינארי - מבנה נתונים המכיל צמתים וקשתות. **שורש העץ** r הוא צומת אליו לא נכנסת אף קשת. לכל צומת בעץ הבינארי, למעט השורש יש **אב** יחיד. לכל צומת יש 0-2 **בנים.** אחד הבנים של הצומת יקרא בן **שמאלי** והשני בן **ימני.** אם לצומת יש רק בן אחד, הוא יכול להיות שמאלי או ימני.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

מושגים בעצים בינאריים (המשך)

רמה של צומת היא **מספר הקשתות** במסלול (היחיד) מן השורש אל הצומת.

עלה - צומת שאין לו בנים.

גובה של עץ - רמה מכסימלית של עלה.

עץ בינארי מלא - עץ בינארי בו לכל צומת שאינו עלה יש **שני בנים.**

עץ בינארי מאוזן - עץ בינארי מלא בו כל העלים נמצאים באותה רמה.

עץ בינארי כמעט מאוזן - עץ בינארי בו כל הרמות מלאות למעט הרמה האחרונה. הרמה האחרונה מלאה משמאל עד לנקודה כלשהי.

כל עץ בינארי מתחלק ל:

1. תת עץ שמאלי (יתכן שהוא ריק).
2. תת עץ ימני (יתכן שהוא ריק).

תכונות עצים בינאריים

טענה 1

יהי T_1 עץ בינארי מאוזן בגובה h. אזי מתקיים:

1. מספר הצמתים ברמה l, $l = 0,1,...,h$ הוא 2^l.
2. מספר הצמתים ב T_1 הוא $2^{h+1} - 1$.

הוכחת טענה 1

תחילה נוכיח את 1 באינדוקציה על l.

בסיס: $l = 0$ - ברמה 0 יש צומת יחיד ואכן מתקיים.

צעד: נניח כי ברמה l יש 2^l צמתים. העץ T_1 מאוזן, ולכן, לכל צומת ברמה l

יש בדיוק שני בנים, ברמה $l+1$. מכאן נובע כי מספר הצמתים ברמה $l+1$ של T_1 הוא $2 \cdot 2^l = 2^{l+1}$.

כדי להוכיח את סעיף 2 נשים לב כי אם n הוא מספר צמתי T_1 אזי מתקיים:

$$n = \sum_{i=0}^{h} 2^i = 2^{h+1} - 1$$

טענה 2

יהי T_2 עץ בינארי כמעט מאוזן שגבהו h. נסמן את מספר איברי T_2 ב - n. אזי n מקיים:

$$2^h \le n \le 2^{h+1} - 1$$

מכאן נובע כי:

$$h \le \log n < h + 1$$

או

$$h = \lfloor \log n \rfloor$$

הוכחה

מיידית.

<u>טענה 3</u>

יהי T עץ בינארי כלשהו עם n עלים. גבהו של T הוא $\Omega(\log n)$.

<u>הוכחה</u>

אם T הוא כמעט מאוזן, נכונות הטענה נובעת באופן מיידי מטענה 2.
נניח אם כן כי T אינו כמעט מאוזן ונראה כי קיים עץ בינארי כמעט מאוזן עם n עלים, T_{ab} אשר גבהו קטן או שווה מגבהו של העץ T.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

<u>הוכחה (המשך)</u>

העץ T_{ab} יתקבל בשני שלבים כדלהלן:
שלב 1: מציאת עץ בינארי מלא, T_c, עם n עלים, אשר גבהו $h(T_c)$ מקיים:

$$h(T) \geq h(T_c)$$

שלב 2: מציאת עץ בינארי כמעט מאוזן, עם n עלים, T_{ab}, אשר גבהו, $h(T_{ab})$ מקיים:

$$h(T) \geq h(T_c) \geq h(T_{ab})$$

מאחר ש T_{ab} הוא עץ בינארי כמעט מאוזן עם n עלים, גבהו הוא $\Theta(\log n)$ והמשפט נובע מאי השוויונים שצוינו.
נותר לנו להראות כיצד נבצע את שלבים 1 ו 2.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

<u>ביצוע שלב 1</u>

העץ T_c ימצא כך: אם T מלא, אז $T_c = T$. אם T אינו מלא, נתבונן בסדרת העצים הבינאריים

$$T = T_0, T_1, \ldots, T_k = T_c$$

כאשר T_0 הוא העץ הנתון T, T_k הוא העץ המלא T_c, ולכל i, $1 \leq i \leq k$, T_i מתקבל מ T_{i-1} על ידי מחיקת צומת v שדרגתו 1, ביחד עם הקשת (v, u) היוצאת ממנו. הצומת u , בנו של v, "יתלה" על אביו של v.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

<u>ביצוע שלב 1 (המשך)</u>

בדרך זו, מספר העלים של T_i שווה למספר העלים של T_{i-1} וגבהו של T_i קטן או שווה מגבהו של T_{i-1}. כל עוד T_i אינו מלא, קיים בו צומת שדרגת היציאה שלו שווה ל 1 והוא ניתן להסרה ולכן התהליך יעצר רק כאשר נגיע לעץ בינארי מלא.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

ביצוע שלב 2

העץ T_{ab} ימצא כך: אם T_c הוא כמעט מאוזן, הטענה נובעת מייד. אם T_c אינו כמעט מאוזן, נתבונן בסדרת העצים הבינאריים

$$T_c = T_0, T_1, \ldots, T_m = T_c$$

כאשר T_0 הוא העץ T_c ו T_m הוא עץ בינארי כמעט מאוזן T_{ab}.

לכל i, $1 \le i \le m$, T_i מתקבל מ T_{i-1} על ידי אחד משני השינויים הבאים:

1. העברת זוג עלים מצומת שמרחקו מן השורש הוא l (לכל $l > 1$), להיות בניו של עלה אשר מרחקו מן השורש קטן מ $l - 1$.

ביצוע שלב 2(המשך)

2. העברת עלה מצומת שמרחקו מן השורש l אל צומת שמאלי יותר שמרחקו מן השורש הוא l.

כל אחד משני שינויים אלה משמר את מספר העלים בעץ ואינו מגדיל את גובה העץ. כל עוד T_i אינו כמעט מאוזן, אפשר לבצע לפחות אחד משני השינויים המתוארים ולכן, התהליך המתואר יעצור רק כאשר נגיע לעץ כמעט מאוזן.

הוכחה (המשך)

מאחר ש T_{ab} הוא כמעט מאוזן ויש לו n עלים נובע מטענה 2 בנספח להרצאה זו, כי גבהו של T הוא $\Theta(\log n)$. מאחר שגבהו של T גדול או שווה מגבהו של T_{ab}, נובע כי גבהו של T הוא $\Omega(\log n)$.

מש"ל

הרצאה 6: מיון שאינו מבוסס השוואות

בהרצאה זו, נציג אלגוריתמי מיון שאינם מבוססי השוואות. האלגוריתמים שנציג מתאימים למערכי קלט שאיבריהם שייכים לקבוצה בגודל m (לדוגמא הטבעיים בין 1 לבין m). סיבוכיות האלגוריתמים שנציג היא $\Theta(n+m)$.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

דוגמא 6.1

מיון של **מערך בינארי** (מערך שאיבריו הם אך ורק 0 ו-1).

1	0	0	1	1	0	1	1	0	0	0	1

איור 6.2: מערך בינארי

תיאור האלגוריתם

1. ספור כמה 0-ים מופיעים במערך.
2. ספור כמה 1-ים מופיעים במערך.
3. בנה את הפלט.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

מיון מניה - תאור האלגוריתם

נניח כי m הוא וכי מספר טבעי (חיובי ושלם) וכי A הוא מערך שאיבריו הם מספרים טבעיים המקיימים:

$$1 \le A[i] \le m$$

מיון מניה ממיין את המערך A בעזרת **מערך מונים** שיקרא $Counter$ ובו m איברים.

האלגוריתם מתבצע ב 3 שלבים:

1. איתחול מונים.
2. ביצוע סטטיסטיקה מלאה של הקלט לפי ערכי תחום ההגדרה.
3. בניית הפלט.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

מיון מניה- הקוד

```
CountingSort(A)
/* איתחול מונים */
for i ← 1 to m  Counter[i] ← 0
/* ביצוע סטטיסטיקה על הקלט */
for j ← 1 to n
   elt ← A[j]
   Counter[elt]++
End for
/* בניית הפלט */
j ← 1
for i ← 1 to m
   while Counter[i] ≠ 0
      outV[j++] ← i;  Counter[i]--
   end while
end for
out(outV)
```

אלגוריתם 6.3: מיון מניה

© כל הזכויות שמורות לפרופסור עמוס ישראלי

שימו לב

מיון מניה כולל את ההשוואה $Counter[i] \neq 0$. נזכור שבמיון מבוסס השוואות מותרות רק השוואות מן הצורה $A[i] < A[j]$. מכאן: מין מניה אינו **מיון מבוסס השוואות.**

להלן נוכח את נכונות האלגוריתם.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

טענה 6.4

לאחר ביצוע הלולאה הראשונה $Counter[i] = 0$ לכל i, $1 \leq i \leq m$.

הוכחה

מיידית.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

טענה 6.5

נסמן ב-n_i את מספר המופעים של הערך i במערך הקלט A. לאחר ביצוע שתי הלולאות הראשונות מתקיים:

1. לכל i, $1 \leq i \leq m$, $Counetr[i] = n_i$.
2. $\sum_{i=1}^{m} counter[i] = \sum_{i=1}^{m} n_i = n$

© כל הזכויות שמורות לפרופסור עמוס ישראלי

הוכחה

נסמן ב n_i^j את מספר המופעים של i בתת המערך $A[1...j]$. אפשר להשתמש באינדוקציה על המעברים בלולאה השניה ולהוכיח כי בסיום המעבר ה-j בלולאה מתקיים:

1. לכל i, $1 \leq i \leq m$, $counetr[i] = n_i^j$.
2. $\sum_{i=1}^{m} counter[i] = \sum_{i=1}^{m} n_i^j = j$.

מכיון ש $n_i^n = n_i$ נכונות טענה 6.6 נובעת.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

טענה 6.6

לכל i, $1 \le i \le m$, נסמן ב-j_i את ערך המשתנה j ברגע בו המעבר ה-i בלולאת ה-**while** (הלולאה השלישית) מסתיים. לצורך זה נניח כי $j_0 = 1$ וגם ש-$n_0 = 1$. אזי באותו רגע מתקיים:

1. $j_i = 1 + \sum_{k=1}^{i} n_k$
2. $outV\left[j_{i-1} \ldots j_{i-1} + n_i\right] = \left[i, i, \ldots, i\right]$
3. j_i האיברים הראשונים במערך $outV$ מכילים תמורה ממוינת של כל איברי מערך הקלט בתחום $1, 2, \ldots, i$.

הוכחה

אפשר להוכיח באינדוקציה על המעברים בלולאה הפנימית כי במהלך המעבר ה-i בלולאה החיצונית, הלולאה הפנימית מבצעת בדיוק n_i פעמים ולאחר סיומה מתקיים:

1. הערך i מופיע בדיוק n_i פעמים במערך $outV$.
2. ערך המשתנה j גדל בדיוק ב-n_i.

בהנחת שני הטיעונים הללו, אפשר להווכיח את נכונות סעיפים 1-3 בטענה 6.6 באינדוקציה על המעברים בלולאה החיצונית.

משל

משפט 6.7

אלגוריתם מיון מניה (Counting Sort) המופעל על מערך קלט A, שערכיו הם מספרים טבעיים בין 1 ל-m, מחשב תמורה ממויינת של המערך A.

הוכחה

נכונות האלגורתם נובעת מיידיית מסעיף 3 בטענה 6.6

ניתוח סיבוכיות האלגוריתם

להלן נוכיח כי סיבוכיות האלגוריתם היא $\Theta(n+m)$. סיבוכיות החלק הראשון והחלק השני הן $\Theta(m)$ ו-$\Theta(n)$ בהתאמה. ניתוח סיבוכיות החלק השלישי מסובך קצת יותר.

הלולאה החיצונית מתבצעת m פעמים, ולכל i, $1 \le i \le n$, $0 \le counter[i] \le n$.

לכן, בכל מעבר בלולאה החיצונית, הלולאה הפנימית תתבצע **לכל היותר** n פעמים, והסיבוכיות חסומה מלמעלה על ידי $O(n \cdot m)$.

סיבוכיות הלולאה הכפולה

להלן ננתח סיבוכיות לולאה זו באופן מדויק:

```
for i ← 1 to m
  while Counter[i] ≠ 0
     outV[j] ← i; j++
                 Counter[i]--
  end while
end for
out(outV)
```

איור 6.8: הלולאה הכפולה

אלגוריתמים א' הרצאה 6 – פרופ' עמוס ישראלי

סיבוכיות הלולאה הכפולה (המשך)

במעבר ה-i בלולאה החיצונית הוראת ה-**while** מתבצעת **בדיוק** $Counter[i]+1$ ולכן, מספר הצעדים הכולל שהוראת ה-**while** מתבצעת הוא

$$\sum_{i=1}^{m}(counter[i]+1)=$$

$$=\left(\sum_{i=1}^{m}counter[i]\right)+m=n+m$$

מכאן נובע כי סיבוכיות הלולאה השלישית היא בדיוק $\Theta(n+m)$.

אלגוריתמים א' הרצאה 6 – פרופ' עמוס ישראלי

מתי כדאי להשתמש במיון מנייה?

כל עוד $m<\Theta(n\log n)$ עדיף להשתמש במיון מניה.

שימו-לב:

אפשר לענות על שאלה זו אך ורק על ידי "השאפת" גודל הקלט לאינסוף.

אם $m=\Theta(n\log n)$ עלינו לשקול גם את גודל הזכרון הנדרש.

אלגוריתמים א' הרצאה 6 – פרופ' עמוס ישראלי

מוטיבציה למיון דליים (Bucketsort)

אלגוריתם מיון ספירה *CountingSort* יכול לשמש כאשר ממיינים מערך ובו מספרים טבעיים בעלי ערך חסום. נניח כעת כי נתון מערך קלט שאיבריו הם **רשומות עם כמה שדות** (לדוגמה רשומות המתארות סטודנטים).

ברצוננו למיין את הרשומות הללו לפי **מפתח** שהוא אחד השדות. אם המפתח הוא **מספר טבעי** שערכיו **חסומים** (למשל גיל בשנים, או ממוצע ציונים מעוגל לערך שלם), עולה הרעיון להשתמש במיון לינארי.

מוטיבציה למיון דליים (המשך)

לרוע המזל, אם רוצים לשמור על מלוא המידע הכלול בכל רשומה אין אפשרות להשתמש במיון מניה. אלגוריתם **מיון דליים** (**Bucket Sort**) המתואר להלן משתמש ברעיון זהה, בדרך שונה במקצת.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

מיון דליים (Bucketsort)

הרעיון: נחליף את המערך *Counter* במערך ובו m רשימות מקושרות. כל מבני נתונים המכונים **דליים** (**buckets**). נעבור על הקלט ו"נשליך" כל רשומה עם מפתח שערכו i אל הדלי ה-i. לאחר מעבר זה, נקבל את הפלט על ידי שירשור את האיברים בכל הדליים שאינם ריקים.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

מיון דליים בעזרת רשימות מקושרות

באלגוריתם זה, כל "דלי" ממומש בעזרת **רשימה מקושרת**.

כל הרשומות בעלות מפתחות שווים, מאוחסנות ברשימה מקושרת אחת.

מתקבל אלגוריתם מיון אשר משתמש אך ורק בהזזת מצביעים (Pointers).

סיבוכיות הזמן של האלגוריתם היא $\Theta(n+m)$ וסיבוכיות המקום גם היא $\Theta(n+m)$.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

מיון דליים - מבנה הנתונים

נניח כי מערך הקלט, A מכיל רשומות (או מצביעים אל רשומות) וכי אנו רוצים למיין רשומות אלו לפי השדה *key*. נסמן את ערך השדה *key* ברשומה $A[i]$ על ידי $key[i]$. (אפשר גם לסמנו על ידי $A[i].key$)

האלגוריתם משתמש בשני מערכים של מצביעים, H (Heads) ו T (Tails) שבכל אחד מהם יש m איברים.

המערכים H ו-T מאחסנים m ראשי רשימות, כלומר מצביעים אל האיבר הראשון ברשימה, ו-m סופי רשימות, בהתאמה.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

מיון דליים - מהלך האלגוריתם

שלב 1: איתחול מונים

לכל i, $1\le i\le m$, האלגוריתם מאתחל את המשתנים $H[i]$ ו-$T[i]$ ל-$null$.

שלב 2: ביצוע סטטיסטיקה על הקלט

האלגוריתם עובר על הקלט ומכניס כל איבר שערך השדה key שלו הוא j, $1\le j\le m$, אל קצה הרשימה שראשה מאוחסן ב-$H[j]$, ו"זנבה" ב-$T[j]$.

שימו לב: כדי לבצע הכנסה זו בזמן שאינו תלוי באורך הרשימה, משתמשים במצביע $T[j]$ המצביע אל קצה הרשימה הזו.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

שלב 3: בניית הפלט

לאחר ביצוע שלב הסטטיסטיקה, **וקטור הפלט** $outV$ מתקבל על ידי העברת האיברים המאוחסנים בכל אחת מן הרשימות אל מערך הפלט, החל ב-$H[1]$ וכלה ב-$H[m]$.

שימו לב

אפשר לבצע את כל פעולות האלגוריתם, למעט בניית הפלט, על ידי **פעולות על מצביעים בלבד.**

© כל הזכויות שמורות לפרופסור עמוס ישראלי

מיון דליים - הקוד

```
BucketSort(A)
/* אתחול רשימות */
for j ← 1 to m
    init H[j] and T[j] to empty
/* ביצוע סטטיסטיקה על הקלט */
for i ← 1 to n
    insert A[i] to list H[key[i]]
            using T[key[i]]
/* בניית הפלט */
j ← 1
/* בניית הפלט */
for i ← 1 to m
   while list H[i] not empty
      out[j] ← remove(H[i])
      j++
   end while
```

אלגוריתם 6.9: מיון דליים

© כל הזכויות שמורות לפרופסור עמוס ישראלי

הערות

1. אלגוריתם מיון דליים משתמש ברעיון זהה לרעיון המונח ביסוד מיון מניה.
2. בקוד מיון דליים, שמרנו את ההערות שהשתמשנו בהן בקוד של מיון מניה, וזאת לצורך הבלטת הדמיון בין שני האלגוריתמים.
3. ההבדל בקוד נובע מן ההבדל באיברי מערך הקלט ומן הרצון לשמור על כל המידע הכלול באיברים אלה.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

תכונות מיון דליים

טענה 6.9

לאחר ביצוע שתי הלולאות הראשונות, הרשימה $H[i]$ מכילה את כל הרשומות המקיימות $key[j]=i$.

הוכחה

מיידית.

טענה 6.10

המערך $outV$ מכיל תמורה ממוינת של מערך הפלט. המיון מתבצע רשימה מקושרת של כל רשומות הקלט.

המערך $outV$ ממוין לפי סדר עולה של השדות key.

הוכחה

מיידית.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

סיבוכיות

$$\Theta(n+m)$$

© כל הזכויות שמורות לפרופסור עמוס ישראלי

יציבות של אלגוריתם מיון

כאשר ממיינים רשומות, לפי מפתח, יש משמעות לשאלה מהו הסדר הפנימי בין כל הרשומות בעלות מפתח שווה.

לדוגמא: מה יהיה הסדר בפלט של כל הסטודנטים שלהם ממוצע ציונים 75.

בהקשר זה נגדיר: אלגוריתם מיון הוא **אלגוריתם יציב** אם הסדר **בפלט** של איברים בעלי מפתחות שווים, **זהה** לסדרם של אותם איברים **בקלט**.

שימו לב

כאשר האיברים הממוינים הם **סקלרים** אין משמעות ליציבות המיון. היציבות מתבטאה כאשר האיברים הממוינים הם מבנים מורכבים יותר.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

יציבות אלגוריתמי מיון (המשך)

כדי לבין זאת טוב יותר נתבונן בבעיה הבאה:

נניח כי בידינו קובץ רשומות המיצגות אנשים. בכל קובץ יש שדה המציין שם ושדה נוסף המציין גיל והקובץ ממוין לפי השם.

דוגמא

אברהם	20
דוד	20
חיים	30
יצחק	20
משה	30
פנחס	18

© כל הזכויות שמורות לפרופסור עמוס ישראלי

דוגמא (המשך)

נניח וכי אנו רוצים למיין את הקובץ לפי גיל כאשר עבור כל גיל, הקובץ יהיה ממוין לפי השם.

הפלט יהיה:

פנחס	18
אברהם	20
דוד	20
יצחק	20
חיים	30
משה	30

© כל הזכויות שמורות לפרופסור עמוס ישראלי

הסבר

אנו מחפשים אלגוריתם אשר ימיין את הקובץ לפי הגיל, כאשר עבור כל קבוצת גיל הסדר המקורי בקובץ הקלט ישמר.
אם נמיין את הקובץ, לפי הגיל, במיון לא יציב, יתכן שבפלט אברהם יופיע אחרי יצחק ודוד.
שימוש במיון יציב, ישמור את הסדר האלפביתי של השמות הפרטיים של אנשים אם אותו שם משפחה ויבטיח את קבלת הפלט הרצוי ללא מאמץ נוסף.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

יציבות מיון דליים

יציבות מיון דליים תלויה במימוש הדליים. במימוש שתיארנו, כל דלי ממומש על ידי **רשימה מקושרת** בה מאוחסנים כל איברי הקלט שלהם מפתחות זהים. כדי להשיג יציבות, סדר האיברים המאוחסנים בכל צריך להיות זהה לסדר בו האיברים הללו מופיעים במערך הקלט. בדרך זו, הסדר יישמר גם בעת בניית מערך הפלט. כדי להבטיח את שמירת סדר ההופעה, יש להכניס כל איבר לקצה הרשימה המקושרת תוך שימוש במצביעים $T[i]$.
לאחר שהגדרנו את מושג היציבות נקבל:

© כל הזכויות שמורות לפרופסור עמוס ישראלי

משפט 6.11

אלגוריתם מיון דליים ממיין רשומות באופן **יציב** בסיבוכיות $\Theta(m+n)$.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

מיון שארית (Radix Sort)

האלגוריתם האחרון בו נדון בהרצאה זו הוא **מיון שארית (Radix Sort)**.
אלגוריתם זה משמש כאשר נתון מערך (גדול מאוד) שאיבריו הם רשומות בנות בני k תווים (למשל ספרות או אותיות).
האלגוריתם כולל לולאה המתבצעת בדיוק k פעמים. בסיום המעבר ה-i, איברי המערך ממוינים לפי i הספרות **הפחות משמעותיות** שלהם.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

אלגוריתם אינטואיטי שאינו מומלץ

נניח שהקלט הוא מערך מספרים בני k ספרות, כגון מערך של מספרי זהות.
האלגוריתם האינטואיטיבי

1. מיין קובץ מספרים לפי הספרה המשמעותית ביותר (הראשונה משמאל).
2. בשלב השני יש למיין (כל אחד מ10 קבצי הביניים שהתקבלו כפלט מן השלב הראשון) לפי הספרה השניה.
3. בשלב ה i, $1 \le i \le k$, יש למיין כל אחד מ 10^{i-1} קבצי הביניים לפי הספרה ה-i.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

הבעיתיות באלגוריתם שהוצע

כל שלב במיון זה, אפשר לבצע על ידי מיון דליים.
הבעיה המתגלה במיון זה היא:
במהלך המיון, עלינו לזהות את גבולות הקבצים, באופן דינמי, או לשמור על כמות של $m \cdot k$ מצביעים שיאורגנו בהירכיה מסובכת למדי.
מצב זה, מסבך את התיכנות ומגביר מאוד את כמות הבגים הצפויה.
אלגוריתם **מיון שארית** מתגבר על הבעיה הזו באופן אלגנטי.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

מיון שארית - הרעיון

במיון לפי הסדר המילוני (לכסיקוגרפי), המיקום היחסי של כל שני איברים a ו b (כלומר מי מהם מופיע ראשון בקובץ הפלט) נקבע לפי האות (ספרה) המשמעותית ביותר שבה האיברים שונים זה מזה.
תכונה זו מנוצלת כך:
נשתמש בגרסה **יציבה** של מיון דליים, נמיין את הקובץ כולו k פעמים, בכל פעם אך ורק לפי הספרות בעמודה מסוימת ובסדר **הפוך** לסדר המשמעותיות, כלומר החל בספרת היחידות (הראשונה מימין) וכלה בספרה המשמעותית ביותר, האחרונה מימין (הראשונה משמאל).

© כל הזכויות שמורות לפרופסור עמוס ישראלי

מיון שארית - הרעיון (המשך)

האלגוריתם יערך ב-k איטרציות.
במעבר הראשון ימוינו כל הרשומות לפי הספרה הכי פחות משמעותית, כלומר הראשונה מימין. במעבר השני ימוינו **שוב** כל הרשומות לפי הספרה השניה בסדר המשמעותיות (ספרת העשרות), וכך לכל i, $3 \le i \le k$, במעבר ה-i ימוינו **שוב** כל הרשומות לפי הספרה ה-i בסדר המשמעותיות.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

מיון שארית - הרעיון (המשך)

בדרך זו, היחס בין כל שני איברים a ו-b נקבע אך ורק לפי הספרה המשמעותית ביותר עבורה a ו-b שונים זה מזה, ללא תלות במיקומם לפני איטרציה זו.
כל הפעולות האחרות, לא משפיעות כלל על המצב היחסי a ו-b.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

תאור האלגוריתם

עבור מפתחות בני k תווים (k קבוע), $b_k, b_{k-1}, ..., b_1$ האלגוריתם יעבוד ב - k איטרציות. באיטרציה ה i, $1 \le i \le k$, הקובץ כולו ימוין במיון דליים, לפי הספרה ה - i.
נשתמש בגרסה יציבה של מיון דליים ובכך נבטיח כי הסדר היחסי הנכון בין המפתחות הממוינים ישמר.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

דוגמא - הקלט

2534				
1769				
4932				
5136				
6492				
1186				
7293				
8116				
3229				
2478				

© כל הזכויות שמורות לפרופסור עמוס ישראלי

דוגמא - לאחר האיטרציה הראשונה

2534	4932			
1769	6492			
4932	7293			
5136	2534			
6492	5136			
1186	1186			
7293	8116			
8116	2478			
3229	1769			
2478	3229			

© כל הזכויות שמורות לפרופסור עמוס ישראלי

דוגמא - לאחר האיטרציה השניה

2534	4932	8116		
1769	6492	3229		
4932	7293	4932		
5136	2534	2534		
6492	5136	5136		
1186	1186	1769		
7293	8116	2478		
8116	2478	1186		
3229	1769	6492		
2478	3229	7293		

© כל הזכויות שמורות לפרופסור עמוס ישראלי

דוגמא - לאחר האיטרציה השלישית

2534	4932	8116	8116	
1769	6492	3229	5136	
4932	7293	4932	1186	
5136	2534	2534	3229	
6492	5136	5136	7293	
1186	1186	1769	2478	
7293	8116	2478	6492	
8116	2478	1186	2534	
3229	1769	6492	1769	
2478	3229	7293	4932	

© כל הזכויות שמורות לפרופסור עמוס ישראלי

דוגמא - לאחר האיטרציה הרביעית

2534	4932	8116	8116	1186
1769	6492	3229	5136	1769
4932	7293	4932	1186	2478
5136	2534	5136	3229	2534
6492	5136	2534	7293	3229
1186	1186	1769	2478	4932
7293	8116	2478	6492	5136
8116	2478	1186	2534	6492
3229	1769	6492	1769	7293
2478	3229	7293	4932	8116

© כל הזכויות שמורות לפרופסור עמוס ישראלי

<u>הוכחת נכונות</u>

נכונות האלגוריתם נובעת מן הטענה הבאה:

<u>טענה 6.12:</u>

לאחר i איטרציות $0 \le i \le k$ איברי המערך ממוינים לפי i הספרות **הפחות משמעותיות** שלהם.

<u>הוכחה:</u>

נכונות הטענה נובעת מיידית מיציבות המיון.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

<u>סיכום</u>

בהרצאה זו, דנו במיונים אשר אינם מבוססי השוואות.

בתחילת ההרצאה הצגנו שתי גרסאות של מיון דליים והוכחנו את נכונותן.

לאחר מכן, הגדרנו מהו **מיון יציב** והסברנו מה התועלת ביציבות.

סיימנו את ההרצאה בהצגת **מיון שארית** (Radix Sort) המשתמש **במיון דליים יציב** כשגרה.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

<u>סיכום נושא המיון</u>

בארבע ההרצאות האחרונות עסקנו בבעית המיון. הנושאים שלמדנו הם:

1. אלגוריתמים פשוטים למיון בסיבוכיות ריבועית: מיון בחירה (Selection Sort) ומיון בועות (Bubble Sort).
2. מיון מיזוג (Merge Sort) בסיבוכיות $\Theta(n\log n)$.
3. מיון מהיר (Quick Sort) בסיבוכיות $\Theta\left(n^2\right)$ ובסיבוכיות ממוצעת $\Theta(n\log n)$.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

<u>סיכום נושא המיון (המשך)</u>

4. חסם תחתון (חסם תורת האינפורמציה) בגובה $\Omega(n\log n)$ על סיבוכיות אלגוריתמי מיון מבוססי השוואות.
5. אלגורית מיון לינאריים (שאינם מבוססי השוואות).

© כל הזכויות שמורות לפרופסור עמוס ישראלי

הרצאה 7: תכנות דינמי

(Dynamic Programming)

קיימת בעיות חישוביות רבות שיש להן אלגוריתם הפרד ופתור בסיבוכיות מעריכית. **תכנות דינמי** היא פרדיגמה המאפשרת פתרון יעיל לחלק מן הבעיות האלה.

בתחילת ההרצאה נציג בעיה פשוטה ונפתור אותה תוך שימוש בתכנות דינמי. לאחר מכן נציג את עיקרי השיטה באופן כללי.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

בעיה 7.1: $P1$

נתון מערך A ובו n איברים חיוביים. מחפשים תת מערך (לא רציף) שסכום איבריו מרבי כאשר מכל שני איברים סמוכים – מותר לקחת לכל היותר אחד. להלן נכנה בעיה זו בשם $P1$.

דוגמאות

נתבונן בדוגמאות הבאות המבהירות את הבעיה:

5	4	**9**

הפתרון: 14

3	**12**	6

הפתרון: 12

12	5	6	**17**	9

הפתרון: 29

13	3	5	**9**

הפתרון: 22

© כל הזכויות שמורות לפרופסור עמוס ישראלי

מסקנות

מהדוגמאות אפשר להסיק מספר האיברים הרצופים שאינם כלולים בפתרון הוא 2. המסקנה הזו מתבטאת בלמה הבאה:

למה 7.2

לכל קלט חוקי לבעיה $P1$ מתקיים:

1. הפתרון האופטימלי מכיל לפחות איבר אחד מכל שלישית איברים רצופים.
2. הפתרון האופטימלי מכיל תמיד את $A[n-1]$ או את $A[n]$, אך כמובן לא את שניהם.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

הוכחה

1. נניח בשלילה כי קיים אינדקס i המקיים: הפתרון האופטימלי לא מכיל את $A[i]$, $A[i+1]$, וגם את $A[i+2]$. מאחר שנתון כי $A[i+1] > 0$, אפשר לראות כי צרוף $A[i+1]$ לפתרון הנתון, מניב פתרון חוקי שסכום איבריו גדול יותר מסכום איברי הפתרון הקודם - סתירה.
2. אם $A[n-2]$ לא נכלל בפתרון כלשהו, אז $\max(A[n-1], A[n])$ נכלל בפתרון הזה ואם $A[n-1]$ לא נכלל בפתרון, אז $A[n]$ נכלל בו.

מש"ל

© כל הזכויות שמורות לפרופסור עמוס ישראלי

אלגוריתם $P1S$

האלגוריתם משתמש בשיטה רקורסיבית $P1R$. הקריאה $P1R(A,i)$ מחשבת את פתרון $P1$ עבור תת המערך $A[1...i]$ תחת האילוץ שהאיבר $A[i]$ נכלל בפתרון.

לדוגמה: עבור המערך $A=[3,12,4,6]$

$P1R(A,3)=A[1]+A[3]=7$.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

אלגוריתם $P1S(A)$

```
P1S(A)
Out(Max(P1R(A,n), P1R(A,n-1)))
------------------------------------------
P1R(A,i)
  if ((i=1) || (i=2)) return A[i]
  if (i=3) return A[3]+A[1]
  return
  A[i] + max(P1R(A,i-2), P1R(A,i-3))
```

אלגוריתם 7.3: $P1S(A)$

© כל הזכויות שמורות לפרופסור עמוס ישראלי

משפט 7.4

1. השיטה הרקורסיבית $P1R(A,i)$ מחשבת את פתרון $P1$ על תת המערך $A[1...i]$ תחת האילוץ שהאיבר $A[i]$ שייך לסכום.
2. אלגוריתם $P1S$ פותר נכון את בעייה $P1$.

הוכחת סעיף 1

נוכיח את הטענה באינדוקציה על i.

בסיס (i=1,2,3)

נכונות תנאי הקצה נובעת ישירות מהגדרת השיטה $P1R$

© כל הזכויות שמורות לפרופסור עמוס ישראלי

הנחה

נניח כי השיטה $P1R(A,j)$ פותרת את בעיה $P1$ כמתואר בסעיף 1 של המשפט לכל $1 \le j < i$, ונוכיח זאת לגבי i.

הוכחה

נתון ש-$A[i]$ נכלל בסכום. לפי הגדרת הבעיה, $A[i-1]$ לא נכלל בסכום. מסעיף 1 בלמה 7.2 נובע כי או האיבר $A[i-2]$ או $A[i-3]$ ייכלל בסכום. פונקצית ה-max בשורה האחרונה של הקוד בוחרת את האפשרות שתרומתה לסכום מרבית.

מש"ל

© כל הזכויות שמורות לפרופסור עמוס ישראלי

הוכחת סעיף 2

מאחר שהפתרון האופטימלי חייב להכיל את $A[n-1]$ או את $A[n]$, האלגוריתם מחזיר את $\max\{P1R(A,n-1),P1R(A,n)\}$.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

סיבוכיות $P1R$

עץ הקריאות של $P1R$ מתואר באיור 7.5 . אפשר לראות כי עץ הקריאות מכיל תת-עץ בינארי מאוזן בגובה $n/3$

n
$n-2$ $n-3$
$n-4$ $n-5$ $n-5$ $n-6$
....
................................5
$n/3$...........................3 2
....
....
....
4...................
3.2

איור 7.6: עץ הקריאות של PIR

© כל הזכויות שמורות לפרופסור עמוס ישראלי

סיבוכיות $P1S$

בתת העץ הזה יש (בערך) $2^{n/3}$ צמתים, וכל צומת הוא קריאה רקורסיבית, כלומר : מספר הקריאות הרקורסיביות הוא **אקספוננציאלי**, מכאן נובע כי סיבוכיות $P1S$ היא מעריכית.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

אלגוריתם בתכנות דינמי לפתרון $P1$

הארגומנטים השונים עליהם מופעלת השיטה $P1R$ הם המספרים $1,2,...,n$, ומספרם הוא בדיוק n.

מכאן נובע כי קיים לפחות ארגומנט אחד אשר השיטה $P1R$ מבוצעת עליו מספר מעריכי של פעמים.

כדי להשיג אלגוריתם **לינארי** נשתמש **טבלת מעקב (LookUp Table)**.

נשתמש **במערך עזר** B באורך n. נגדיר את $B[i]$ כשווה לערך המחזר על ידי $P1R(A,i)$, אך נחשב את הערך הזה **ללא קריאה** ל$P1R(A,i)$.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

© כל הזכויות שמורות לפרופסור עמוס ישראלי

חישוב איברי מערך העזר

נשתמש בתנאי הקצה ונקבל:

$B[1]=5$, $B[2]=2$, $B[3]=3+5=8$.

כדי לחשב את $B[i]$, $4\le i\le n$, נשים לב כי לפי ההגדרה, $A[i]$ חייב להיכלל בסכום. מכאן, לפי הגדרת $P1$, $A[i-1]$ לא נכלל בסכום.

לפי סעיף 1 בלמה 7.2 אחד מבין האיברים $A[i-1]$, $A[i-2]$, או $A[i-3]$, חייב להיכלל בסכום. מאחר ש-$A[i-1]$ לא יכול להיכלל, הפתרון מכיל או את $A[i-2]$ או את $A[i-3]$.

והסכום המרבי יתקבל על ידי:

$B[i]=A[i]+\max(B([i-3],B[i-2])$.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

המשך תיכון אלגוריתם בתכנות דינמי

מסעיף 2 בלמה 7.1 נובע כי הערך המוחזר הוא $\max\{B(n-1),B(n)\}$.

תובנות אלה מניבות את האלגוריתם הבא:

```
P1D(A)
  B[1] ← A[1]; B[2] ← A[2];
  B[3] ← B[1] + A[3];
  for i ← 4 to n
     B[i] ← A[i] +
            + max(B[i-3], B[i-2])
  endfor
  return(max(B[n], B[n-1]))
```

אלגוריתם 7.7: *P1D*

© כל הזכויות שמורות לפרופסור עמוס ישראלי

דוגמה 7.8

נתבונן בקלט לדוגמה:

5	2	3	8	4	6	9	1	8	4

ערכי האיברים במערך העזר יהיו

הקלט:	5	2	3	8	4	6	9	1	8	4
מערך עזר:	5	2	8	13	12	19	22	20	30	26

© כל הזכויות שמורות לפרופסור עמוס ישראלי

משפט 7.9

האלגוריתם $P1D$ מחשב את סכום תת המערך המרבי בהתאם לדרישות.

הוכחה

ההוכחה זהה להוכחת הנכונות של אלגוריתם $P1S$.

מש"ל

> במקרה זה, כמו במקרים רבים של אלגוריתמים בתכנות דינמי, השמורה שווה להגדרת מערך העזר.

הערת צד 7.9

הסיבוכיות

לינארית.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

חישוב תת המערך המרבי

כעת נחשב את תת המערך המרבי עצמו ולא רק את סכום האיברים שלו.
נרשום מחדש את מערך הקלט ואת מערך העזר ונוסיף להם מערך עזר נוסף. באיבר ה- i נרשום את האינדקס שהניב את המכסימום בעזרתו חשבנו את $B[i]$.

האינדקס :	1	2	3	4	5	6	7	8	9	10
הקלט :	5	2	3	8	4	6	9	1	8	4
מערך עזר :	5	2	8	13	12	19	22	20	30	26
המקור :	-	-	1	1	3	4	4	6	7	7

© כל הזכויות שמורות לפרופסור עמוס ישראלי

חישוב תת המערך המרבי(המשך)

אפשר לראות כי הסכום $B[9]=30$
התקבל על ידי $B[9]=B[7]+A[9]$
אם נמשיך בדרך זו נקבל :

$$B[9]=B[7]+A[9]=B[4]+A[7]+A[9]$$

ומאחר ש

$$B[4]=A[1]+A[4]$$

נקבל

$$B[9]=A[1]+A[4]+A[7]+A[9]$$

© כל הזכויות שמורות לפרופסור עמוס ישראלי

סיכום הצעדים שביצענו בבנית $P1D$

1. פתרנו את $P1$ בעזרת השיטה הרקורסיבית $P1R$ אשר מחשבת את סכום תת המערך המרבי עבור תת המערך $A[1:i]$, כאשר נתון ש-$A[i]$ שייך לפתרון.
2. הגדרנו מערך עזר B כאשר $B[i]=P1R(A,i)$.
3. חישבנו את איברי מערך העזר.
4. חישבנו את סכום הוקטור המרבי עבור המערך A.
5. חישבנו את האינדקסים של האיברים של תת המערך המרבי.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

מתכון לבניית אלגוריתם בתכנות דינמי

1. הצג שיטה רקורסיבית R לפתרון הבעיה.
2. אם סיבוכיות R גבוהה מאוד, אך מספר הארגומנטים השונים ש-R מופעלת עליהם, נמוך יחסית, אז :
3. השתמש בטבלת מעקב, חשב את ערכי R על הארגומנטים השונים בעזרת טבלת המעקב, **וללא קריאה לשיטה** R עצמה.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

בעיה 7.10: $P2$

נתון מערך A ובו n איברים חיוביים. מחפשים את הסכום המרבי אשר אפשר לקבל כאשר - מכל שני איברים סמוכים מותר לקחת לכל היותר אחד וחצי מן השני. להלן נכנה בעיה זו בשם $P2$.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

תיכון האלגוריתם

נשתמש בתכנות דינמי. נחשב שני וקטורי עזר :

$B[i]$ - הסכום המרבי המקבלים מן המערך $A[1...i]$ כאשר $A[i]$ כלול בסכום.

$C[i]$ - הסכום המרבי המקבלים מן המערך $A[1:i]$ כאשר $1/2\cdot A[i]$ כלול בסכום.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

אלגוריתם $P2D$

$P2D(A)$

$B[1]\leftarrow A[1];\ C[1]\leftarrow 1/2\cdot A[1]$

$B[2]\leftarrow A[2]+1/2\cdot A[1];\ C[2]\leftarrow 1/2\cdot A[2]+A[1]$

for $i\leftarrow 4$ to n

$B[i]\leftarrow A[i]+C[i-1]$

$C[i]\leftarrow 1/2\cdot A[i]+\max\{B[i-1],C[i-1]\}$

end for

return $\max\{B[n],C[n]\}$

אלגוריתם 7.11: $P2D$

© כל הזכויות שמורות לפרופסור עמוס ישראלי

דוגמא 7.12

A	5	2	3	8	4	6	9	8
B	5	4.5	9	15.5	17	23.5	29	36
C	2.5	6	7.5	13	17.5	20.5	28	33

והפלט הוא $\max(B[n],C[n])=36$

© כל הזכויות שמורות לפרופסור עמוס ישראלי

חישוב תרומת כל איבר

לכל i ברצוננו לדעת האם בסכום המירבי מופיע $A[i]$ או $1/2\cdot A[i]$.

נוסיף עוד מערך עזר ונסמן בו לכל i, $3\le i\le n$, האם $C[i]$ התקבל מ $B[i-1]$ או מ $C[i-1]$.

	1	2	3	4	5	6	7	8
A	5	2	3	8	4	6	9	8
B	5	4.5	9	15.5	17	23.5	29	36
C	2.5	6	7.5	13	17.5	20.5	28	33
	-	-	*C*	*B*	*B*	*C*	*B*	*B*

© כל הזכויות שמורות לפרופסור עמוס ישראלי

חישוב תרומת כל איבר

נשתמש במערך הנוסף כדי לקבל את התוצאה:

$$out = A[8]+1/2\cdot A[7]+A[6]+1/2\cdot A[5]$$
$$A[4]+1/2\cdot A[3]+1/2\cdot A[2]+A[1]$$

© כל הזכויות שמורות לפרופסור עמוס ישראלי

בעיה 7.12:

נתון מערך A ובו n איברים לוא דווקא חיוביים. מחפשים את הסכום המרבי אשר אפשר לקבל כאשר - מכל שני איברים סמוכים מותר לקחת לכל היותר אחד, כמו כן נתון שחייבים לבחור לפחות באיבר אחד מן המערך.

להלן תכונה בעיה זו בשם $P3$.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

תיכון אלגוריתם בתכנות דינמי

נגדיר מערך עזר B כמו ב - $P1D$:

$B[i]$ - פתרון $P3$ עבור תת המערך
$A[1:i]$ עם האילוץ $A[i]$ כלול בסכום.

נקבל: $B[1]=A[1]$ ו- $B[2]=A[2]$.

ההבדל בין $P1$ (המערך A מכיל רק איברים חיוביים) לבין $P3$ (יתכן שהמערך A מכיל איברים שליליים) מתבטא בחישוב $A[3]$: אם $A[1]<0$ וגם $A[2]<0$, מתקיים לפי ההגדרה כי $B[3]=A[3]$. לכן ערכם של $B[3]$ ושל $B[4]$ מתקבל כך:

$$B[3]=A[3]+\max\{A[1],0\}$$
$$B[4]=A[4]+\max\{B[1],B[2],0\}$$.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

בעיה 7.3 (המשך)

למעשה אפשר לראות כי לכל $2<i\le n$

$$B[i]=A[i]+\max\{B[1],\ldots,B[i-2],0\}$$

לאחר חישוב המערך B כולו, נקבל את הפלט על ידי החזרת האיבר המרבי במערך B. (נמקו זאת!).

לרוע המזל, סיבוכיות האלגוריתם שהתקבל היא **ריבועית**.

כדי להתגבר על כך נחשב מערך עזר נוסף C אשר יוגדר כך:

$C[i]$- פתרון הבעיה עבור המערך
$A[1:i]$ כאשר $A[i]$ לא כלול בסכום.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

בעיה 7.4 - בעית התשלום המינימלי

נתון כביש מהיר ובו n תחנות תשלום.
בכל תחנה התשלום שונה.
מכונית העוברת בכביש צריכה לשלם לפחות בכל שער שלישי. נניח כי יש 10 תחנות, המכונית יכולה לשלם לפי הסדרות הבאות:

3,6,9

1,4,7,8

2,3,5,6,7,10

2,3,4,7,9,10

סדרת התשלום הבאה אינה מותרת:

1,4,8,10

כי לא משלמים בשערים מס' 5,6,7

© כל הזכויות שמורות לפרופסור עמוס ישראלי

בעית התשלום המינימלי (המשך)

יש להציע אלגוריתם לפתרון הבעיה הבאה:

קלט - מערך pay עם n איברים,
$pay[i]$ הוא התשלום בתחנה i .

פלט - התשלום המינימלי הנדרש למעבר הכביש.

בניית אלגוריתם רקורסיבי

נסה לפתור את הבעיה בעזרת פרוצדורה רקורסיבית $Mpay$.

לכל $1\le i\le n$, נגדיר את $Mpay(i)$ כתשלום המינימלי שעלינו לשלם מתחנה מס' 1 ועד תחנה i כאשר ידוע לנו כי עלינו לשלם בתחנה i.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

חישוב רקורסיבי של $Mpay(i)$

תנאי הקצה

מן ההגדרה נובע באופן מיידי כי:

$$Mpay(1)=pay[1]$$

$$Mpay(2)=pay[2]$$

$$Mpay(2)=pay[3]$$

ואלו הם **תנאי הקצה**.

הרקורסיה

לכל $4\le i\le n$, כדי לחשב את $Mpay(i)$ נזכור כי נתון **שאנו חייבים לשלם בתחנה** i והסכום הוא $pay[i]$.
כעת עלינו לברר מהי התחנה האחרונה בה נשלם לפני שנשלם בתחנה i.
האפשרויות הן: תחנה $i-1$, בתחנה $i-2$ או בתחנה $i-3$.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

אם בחרנו לשלם בתחנה $i-1$ יהיה עלינו לשלם עוד $Mpay(i-1)$.
אם בחרנו לשלם בתחנה $i-2$ יהיה עלינו לשלם עוד $Mpay(i-2)$.
אם בחרנו לשלם בתחנה $i-3$ יהיה עלינו לשלם עוד $Mpay(i-3)$.
כדי להחליט מהי התחנה הבאה בה נשלם נבצע 3 קריאות רקורסיביות ונחשב את המינימום בין הערכים המוחזרים:

$$Mpay(i)\leftarrow pay[i]+\min\left\{\begin{matrix}Mpay(i-1)\\ Mpay(i-2)\\ Mpay(i-3)\end{matrix}\right\}$$

© כל הזכויות שמורות לפרופסור עמוס ישראלי

החישוב הסופי

כל מכונית שעוברת בכביש חייבת לשלם בתחנה n או בתחנה $n-1$ או בתחנה $n-2$.
מכונית שעוברת בתחנה n תשלם לפחות $Mpay(n)$.
מכונית שעוברת בתחנה $n-1$ תשלם לפחות $Mpay(n-1)$.
מכונית שעוברת בתחנה $n-2$ תשלם לפחות $Mpay(n-2)$.

התשלום המינימלי יתקבל על ידי:

$$fpay\leftarrow\min\left\{\begin{matrix}Mpay(n)\\ Mpay(n-1)\\ Mpay(n-2)\end{matrix}\right\}$$

© כל הזכויות שמורות לפרופסור עמוס ישראלי

האלגוריתם הרקורסיבי

```
Mpay(i)
  if i = 1 or i = 2 or i = 3
    return(pay[i])
  fpay ← pay[i] + min{ Mpay(i+1), Mpay(i+2), Mpay(i+3) }
return(fpay)
```

איור 7.4: האלגוריתם הרקורסיבי

© כל הזכויות שמורות לפרופסור עמוס ישראלי

האלגוריתם הסופי

כאמור, כדי לקבוע את התשלום הסופי עלינו לבחור לשלם באחת מבין 3 התחנות האחרונות, כפי שמתקבל מן הקוד הבא:

$$\mathit{compute}(\mathit{pay})$$

$$\mathit{fpay} \leftarrow \min\begin{Bmatrix} \mathit{Mpay}(n) \\ \mathit{Mpay}(n-1) \\ \mathit{Mpay}(n-2) \end{Bmatrix}$$

$$\mathrm{return}(\mathit{fpay})$$

איור 7.5: האלגוריתם הסופי

הוכחת נכונות

מיידית.

הסיבוכיות

נתבונן בעץ הקריאות הרקורסיביות. כל קריאה עם $0 \le i \le n-3$ מניבה עץ **טרנרי** מלא (לכל צומת פנימי יש 3 בנים).

בעץ זה, אורך המסלולים מן השורש אל העלים הוא בין $n/3$ לבין n.

אם **נגזום** את העץ בעומק $n/3$, נקבל עץ טרנרי **מאוזן** ובו $3^{n/3}$ עלים.

מספר הקריאות הרקורסיביות גדול מ $3^{n/3}$ כלומר **אקספוננציאלי**. כמו בדוגמאות הקודמות, הסיבה סיבוכיות זו היא קריאה מרובה לשיטה הרקורסיבית Mpay.

אלגוריתם בתכנות דינמי

כדי להגיע לאלגוריתם לינארי, נשתמש בתכנות דינמי: למעשה השיטה Mpay נקראת עם **בדיוק** n פרמטרים שונים.

נקצה מערך עזר tMpay ובו n איברים.

נשתמש במערך tMpay כדי לחשב את ערכי השיטה Mpay כדלהלן:

$$\mathit{tMpay}[1] \leftarrow \mathit{pay}[1]$$

$$\mathit{tMpay}[2] \leftarrow \mathit{pay}[2]$$

$$\mathit{tMpay}[3] \leftarrow \mathit{pay}[3]$$

כעת נחשב את $\mathit{tMpay}[4]$ על ידי

$$\mathit{tMpay}[4] \leftarrow \mathit{pay}[4] + \min\begin{Bmatrix} \mathit{tMpay}[1] \\ \mathit{tMpay}[2] \\ \mathit{tMpay}[3] \end{Bmatrix}$$

אלגוריתם בתכנות דינמי (המשך)

באותו אופן לכל $5 \le i \le n$ בכל פעם שיהיו בידינו האיברים $\mathit{tMpay}[i-1]$, $\mathit{tMpay}[i-2]$ ו- $\mathit{tMpay}[i-3]$ נוכל לחשב את $\mathit{tMpay}[i]$ על ידי

$$\mathit{pay}[i] + \min\begin{Bmatrix} \mathit{tMpay}[i-1] \\ \mathit{tMpay}[i-2] \\ \mathit{tMpay}[i-3] \end{Bmatrix}$$

לאחר מילוי המערך, התוצאה הסופית תוחזר על ידי ביצוע

$$\min\begin{Bmatrix} \mathit{tMpay}[n-2] \\ \mathit{tMpay}[n-1] \\ \mathit{tMpay}[n] \end{Bmatrix}$$

אלגוריתם בתכנות דינמי

$$
\begin{array}{l}
DPcompute(pay) \\
tMpay[1] \leftarrow pay[1] \\
tMpay[2] \leftarrow pay[2] \\
tMpay[3] \leftarrow pay[3] \\
\textbf{for } i \leftarrow 4 \textbf{ to } n \textbf{ do} \\
tMpay[i] \leftarrow pay[i] + \min \begin{Bmatrix} tMpay[i-1] \\ tMpay[i-2] \\ tMpay[i-3] \end{Bmatrix} \\
\textbf{end for} \\
fpay \leftarrow \min \begin{Bmatrix} tMpay[n] \\ tMpay[n-1] \\ tMpay[n-2] \end{Bmatrix} \\
\textbf{return}(fpay)
\end{array}
$$

איור 7.6: אלגוריתם בתכנות דינמי

© כל הזכויות שמורות לפרופסור עמוס ישראלי

סיכום ההרצאה

שיטת ה**תכנות דינמי** משמשת להורדת סיבוכיות של אלגוריתמים רקורסיביים בהם מתבצע מספר רב של קריאות רקורסיביות עם ארגומנטים זהים.

אם נבצע כל קריאה רקורסיבית, תתקבל לסיבוכיות גבוהה ביותר. במקום זאת, נבצע את הקריאות הקרורסיביות אחת אחר השניה, באופן שיטתי.

בכל פעם שנסיים פתרון תת בעיה כלשהי, נאחסן את הפתרון כך שיהיה זמין לשימוש חוזר ככל שיידרש.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

סיכום ההרצאה - תיכון אלגוריתמים בשיטת התכנות הדינמי

1. מציגים אלגוריתם רקורסיבי לפתרון הבעיה.
2. מגלים כי סיבוכיות האלגוריתם הרקורסיבי היא גבוהה, על אף שמספר צרופי הפרמטרים האפשריים הוא נמוך יחסית.
3. מגלים כי הסיבה לסיבוכיות הגבוהה היא ביצוע קריאות חוזרות עם וקטור פרמטרים זהה.
4. מגדירים מערך עזר המיועד לאחסון הערכים המחושבים על ידי השיטה הרקורסיבית.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

תיכון אלגוריתמים בשיטת התכנות הדינמי (המשך)

5. מחשבים את הערכים המוחזרים על ידי השיטה הרקורסיבית החל בתנאי הקצה. מאחסנים במערך העזר כל אחד מן הערכים המחושבים, ומשתמשים בערכים אלה לחישוב Bottom Up של שאר הערכים.
6. כתוצאה מקבלים אלגוריתם איטרטיבי. מוכיחים את נכונות האלגוריתם כאשר השמורה תעיד על שויון הערכים במערך העזר לערכי השיטה הרקורסיבית המקורית.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

תרגילים

1. השלימו את הוכחת הנכונות.
2. השלימו את קוד האלגוריתם כך שיחשב את האינדקסים של האיברים מהם מתקבל תת הוקטור המירבי.
3. הציעו דרך נוספת לחישוב האיברים מהם מתקבל תת המערך המרבי, ללא שימוש במערך עזר.
4. הציעו אלגוריתם לפתרון הבעיה הבאה: נתון מערך A ובו n איברים חיוביים. מחפשים תת מערך (לא רציף) שסכום איבריו **מינימלי** כאשר מכל שני איברים סמוכים - חייבים לקחת לפחות אחד.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

P2

1. הוכיחו את נכונות האלגוריתם.
2. השלימו את האלגוריתם כך שיחשב את תרומתו של כל איבר של מערך הקלט.
3. הציעו אלגוריתם לחישוב תרומת כל איבר ללא שימוש במערך עזר נוסף.
4. הציעו אלגוריתם לפתרון בעיה 7.2 תוך שימוש במערך עזר יחיד.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

P3

תרגילים

1. כתבו קוד דמה לאלגוריתם בהתאם לקווים המוצעים.
2. הציעו אלגוריתם לינארי לפתרון בעיה 7.3.
3. הציעו דרך לחישוב האיברים המופיעים בפתרון.
4. פתרו את בעיה 2 ללא האילוץ שכל האיברים חיוביים.
5.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

הרצאה 8: חישוב תת הסדרה המשותפת הארוכה ביותר

בהרצאה זו נפתור את בעיית **תת הסדרה המשותפת המכסימלית (Longest Common Subsequence)** או **LCS**. להלן תיאור הבעיה:

בעיה 8.1

קלט

שתי סדרות תוים X ו- Y

פלט

תת סדרה משותפת ל-X ול-Y שארכה מכסימלי.

שימו לב

יתכנו מספר תת סדרות מכסימליות .

© כל הזכויות שמורות לפרופסור עמוס ישראלי

סימון

נסמן את תת הסדרה המשותפת ב $LCS(X,Y)$.

דוגמא 8.2

$$X=(a,c,d,b,a,e,d,b,c,\tilde{a},e,b,h)$$

$$Y=(c,a,b,c,\tilde{e},\tilde{a},b,h,d,a,c,d,h)$$

תת הסדרה המרבית הראשונה היא $LCS1$ והיא מסומנת על ידי ... מתחת לכל תו בתת הסדרה.

אם נחליף את התו $\tilde{a}$ ב-$\tilde{e}$ נקבל את $LCS2$.

$$LCS1(X,Y)=(c,a,b,c,a,b,h)$$

$$LCS2(X,Y)=(c,a,b,c,e,b,h)$$

© כל הזכויות שמורות לפרופסור עמוס ישראלי

האם אפשר לצרף את שני התווים האלה לאחת מתת הסדרות?

שאלה למחשבה 8.3

© כל הזכויות שמורות לפרופסור עמוס ישראלי

הגדרה 8.4:

1. תהי $X=(x_1,...,x_n)$ סדרה. **הרישא** ה-i של X, $0\le i\le n$, היא תת הסדרה $(x_1,...,x_i)$. נסמן $X_i=(x_1,...,x_i)$.
2. **הסדרה הריקה** היא הסדרה שאינה מכילה אפילו תו אחד. נסמן את הסדרה הריקה ב Λ.

כמה סדרות ריקות קיימות?

שאלה למחשבה 8.5

© כל הזכויות שמורות לפרופסור עמוס ישראלי

הכנות לאלגוריתם רקורסיבי

שתי הלמות הבאות ישמשו באלגוריתם הרקורסיבי לחישוב LCS שנפתח בהמשך.

למה 8.6 - תנאי קצה

אם X היא הסדרה הריקה, ו-Y סדרה כלשהי אז

$$LCS(Y,\Lambda)=LCS(\Lambda,Y)=\Lambda$$

הוכחה

האורך של תת הסדרה המכסימלית **קטן או שווה** לאורך כל אחת מן הסדרות.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

למה 8.7: קיצור הקלט

תהיינה $X=(x_1,...,x_n)$ ו-$Y=(y_1,...,y_m)$ סדרות.

אם $x_n=y_m$, אזי

$$LCS(X,Y)=LCS(X_{n-1},Y_{m-1})\circ x_n$$

כלומר: אם התו האחרון של X, שווה לתו האחרון של Y, אזי כל LCS של X ו-Y מתקבלת על ידי שרשור של אחת מתת הסדרות המשותפות הארוכות ביותר של X_{n-1} ו-Y_{m-1} עם התו המשותף x_n.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

דוגמא

$$X=(a,c,b,d,f)$$
$$Y=(b,c,d,a,f)$$
$$LCS(X,Y)=$$
$$=LCS(X_4,Y_4)\circ f=(c,d,f)$$

הוכחת למה 8.7

תהי $Z=(z_1,...,z_k)$ תת סדרה מרבית, כלומר תת סדרה משותפת שאינה ניתנת להארכה, של X ושל Y. נניח בשלילה כי $z_k\neq x_n$. במצב זה, ברור כי $Z\circ x_n$ היא תת סדרה משותפת של X ושל Y - סתירה להנחתנו כי Z אינה ניתנת להארכה.

מש"ל

© כל הזכויות שמורות לפרופסור עמוס ישראלי

למה 8.8: קיצור הקלט (2)

יהיו Xו-Y סדרות. אם $x_n\neq y_m$, אזי

$$LCS(X,Y)=$$
$$LCS(X,Y_{m-1})\,|\,LCS(X_n,Y)$$

כלומר: אפשר להוריד תו אחד מן הקצה הימני של אחת הסדרות, בלי לפגוע ב-LCS.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

דוגמא

$X = (c, b, d, g, f)$

$Y = (b, c, d, g)$

אפשר להוריד את התו האחרון מן הסדרה X ולקבל

$$LCS(X,Y) = LCS(X_4, Y) = (c, d, g)$$

© כל הזכויות שמורות לפרופסור עמוס ישראלי

הוכחה

אם התו האחרון באחת הסדרות משתתף ב-LCS, הוא חייב להיות במקום האחרון של ה-LCS. במקום זה יכול להיות תו אחד בלבד. לכן, אפשר להוריד את התו האחרון בסדרה השניה.

אם שני התווים האחרונים אינם משתתפים ב-LCS, בוודאי אפשר להוריד כל אחד מהם.

מש"ל

© כל הזכויות שמורות לפרופסור עמוס ישראלי

אלגוריתם רקורסיבי לחישוב אורך *LCS*

בהנתן שתי סדרות X ו-Y, אפשר לחשב ישירות את תת הסדרה המכסימלית אולם, מסתבר כי קל יותר לחשב תחילה את האורך של תת הסדרה הזו ורק לאחר מכן לחשב את תת הסדרה המכסימלית עצמה.

להלן נציג את $L(X_i, Y_j)$: אלגוריתם רקורסיבי לחישוב אורך ה-LCS של הרישות X_i ו-Y_j. האלגוריתם משתמש בלמות 8.6-8.8.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

קוד האלגוריתם

$L(X_i, Y_j)$
    if $(i = 0)$ **return** 0
    if $(j = 0)$ **return** 0
    if $(x_i = y_j)$
        return $L(X_{i-1}, Y_{j-1}) + 1$
  else return
    $\max\{L(X_{i-1}, Y_j), L(X_i, Y_{j-1})\}$
end

אלגוריתם 8.9: *LCS*

נכונות האלגוריתם נובעת מיידית מלמות 8.6-8.8.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

סיבוכיות האלגוריתם

סיבוכיות האלגוריתם תלויה במספר הקריאות הרקורסיביות. במקרה הגרוע ביותר, בו כל אותיות הקלט שונות זו מזו, כל קריאה לאלגוריתם, עם קלטים שאינם ריקים ואשר סכום אורכיהם n , מבצעת שתי קריאות רקורסיביות עם קלטים שסכום אורכיהם הוא $n-1$ ולכן:

$$T(n)=2T(n-1)+c$$

במקרה זה, הסיבוכיות של T היא אקספוננציאלית.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

האם יתכן שכל אותיות הקלט יהיו שונות זו מזו?

שאלה למחשבה 8.10

© כל הזכויות שמורות לפרופסור עמוס ישראלי

אלגוריתם בתכנות דינמי

נניח כי X ו-Y, הן שתי סדרות שארכן n ו-m. מספר צרופי הפרמטרים השונים לפרוצדורה L, כולל הסדרה הריקה, הוא בדיוק $(m+1)\cdot(n+1)$
האלגוריתם שנציג משתמש במערך דו ממדי TL מסדר $(m+1)\times(n+1)$ אשר האינדקסים שלו הם $n-1$, ו-$m-1$ בהתאמה. בסיום האלגוריתם נקבל $TL[i,j]=L(X_i,Y_j)$, כלומר האיבר $TL[i,j]$ יכיל את **האורך** של תת הסדרה המכסימלית של X_i ושל Y_j.
ערכי המערך יחושבו ללא קריאה לאלגוריתם L.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

תאור האלגוריתם בתכנות דינמי

תחילה, האלגוריתם מאפס את השורה ה-0 ואת העמודה ה-0 של המערך TL מטריצה המתאימות לאורך LCS של שתי סדרות שאחת מהן ריקה.
לאחר מכן, האלגוריתם מחשב את איברי המערך, שורה אחר שורה, תוך שימוש בנוסחת הנסיגה מן האלגוריתם הרקורסיבי, ובערכי המטריצה שחושבו כבר.
עם סיום האלגוריתם מתקיים:

$$|LCS(X,Y)|=L(X_n,Y_m)=TL[n,m]$$

© כל הזכויות שמורות לפרופסור עמוס ישראלי

קוד האלגוריתם

$DPL(X,Y)$
for $i \leftarrow 0$ **to** m **do** $TL[i,0] \leftarrow 0$
for $j \leftarrow 1$ **to** n **do** $TL[0,j] \leftarrow 0$
for $i \leftarrow 1$ **to** m
  for $j \leftarrow 1$ **to** n
    if $(x_i = y_j)$
      $(TL[i,j] \leftarrow TL[i-1,j-1]+1)$
    else
      $TL[i,j] \leftarrow$
        $\max\{T[i-1,j], T[i,j-1]\}$
    endif
  end
end
return $TL(n,m)$

אלגוריתם 8.11: *DPL*

© כל הזכויות שמורות לפרופסור עמוס ישראלי

דוגמא

נתבונן בדוגמא שהצגנו בתחילת ההרצאה:

$$X = (a, \underset{\dots}{c}, d, b, \underset{\dots}{a}, e, d, \underset{\dots}{b}, \underset{\dots}{c}, \tilde{a}, \underset{\dots}{e}, \underset{\dots}{b}, \underset{\dots}{h})$$

$$Y = (\underset{\dots}{c}, \underset{\dots}{a}, \underset{\dots}{b}, \underset{\dots}{c}, \underset{\dots}{\tilde{e}}, \tilde{a}, \underset{\dots}{b}, \underset{\dots}{h}, d, a, c, d, h)$$

להלן, נבנה את המערך TL עבור הדוגמא הזו.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

דוגמא 8.12

	Λ	c	a	b	c	e	a	b	f	d	a	c	d	f
Λ														
a														
c														
d														
b														
b														
e														
d														
b														
c														
a														
e														
b														
f														

© כל הזכויות שמורות לפרופסור עמוס ישראלי

דוגמא 8.12 (המשך)

	Λ	c	a	b	c	e	a	b	f	d	a	c	d	f
Λ	0	0	0	0	0	0	0	0	0	0	0	0	0	0
a	0	0	1	1	1	1	1	1	1	1	1	1	1	1
c	0	1	1	1	2	2	2	2	2	2	2	2	2	2
d	0	1	1	1	2	2	2	2	2	3	3	3	3	3
b	0	1	1	2	2	2	2	3	3	3	3	3	3	3
b	0	1	2	2	2	2	3	3	3	3	4	4	4	4
e	0	1	2	2	2	3	3	3	3	3	4	4	4	4
d	0	1	2	2	2	3	3	3	3	4	4	4	5	5
b	0	1	2	3	3	3	3	4	4	4	4	4	5	5
c	0	1	2	3	4	4	4	4	4	4	4	5	5	5
a	0	1	2	3	4	4	5	5	5	5	5	5	5	5
e	0	1	2	3	4	5	5	5	5	5	5	5	5	5
b	0	1	2	3	4	5	5	6	6	6	6	6	6	6
f	0	1	2	3	4	5	5	6	7	7	7	7	7	7

© כל הזכויות שמורות לפרופסור עמוס ישראלי

חישוב תת הסדרה המכסימלית -

תכונות המערך TL

לאחר שסיימנו את ביצוע האלגוריתם, אורך תת הסדרה המכסימלית נמצא באיבר $TL[n,m]$.

כעת נפתח אלגוריתם לחישוב תת הסדרה המכסימלית עצמה.

נשים לב כי:

1. הפרש הערכים בין כל שני תאים סמוכים הוא 0 או 1.
2. הערך באיבר ה$TL[i,j]$ מחושב כפונקציה של האיברים $TL[i,j-1]$, $TL[i-1,j]$, ו-$TL[i-1,j-1]$.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

תאור האלגוריתם

להלן מוצגות את חמש התצורות האפשריות לקבוצות בנות ארבעה תאים סמוכים במערך TL.

אנו מניחים כי האיבר הימני בשורה התחתונה הוא $TL[i,j]$ ו-$TL[i,j]=k$, כמתואר בטבלה הבאה:

$TL[i-1,j-1]$	$TL[i-1,j]$
$TL[i,j-1]$	$TL[i,j]=k$

איור 8.13: התצורה של $TL[i,j]$

© כל הזכויות שמורות לפרופסור עמוס ישראלי

תצורות 1-4

בכל אחת מתצורות 1-4, הערך $TL[i,j]=k$, יכול היה להתקבל מאחד האיברים הסמוכים, ללא תוספת תו ל LCS.

תצורה 1

k	k
k	k

תצורה 2

$k-1$	k
k	k

תצורה 3

$k-1$	k
$k-1$	k

תצורה 4

$k-1$	$k-1$
k	k

איור 8.13: תצורות 1-4

© כל הזכויות שמורות לפרופסור עמוס ישראלי

תצורה 5

בתצורה 5 הערך $TL[i,j]=k$ מתקבל **אך ורק** על ידי זיהוי תו משותף ותוספת 1 לערך $TL[i-1,j-1]$.

$k-1$	$k-1$
$k-1$	k

איור 8.14: תצורה 5

© כל הזכויות שמורות לפרופסור עמוס ישראלי

חילוץ תת הסדרה המכסימלית – תיאור האלגוריתם

כדי לבנות את תת הסדרה המשותפת המכסימלית, מבצעים סיור מ $TL[n,m]$ אל $TL[0,0]$. במהלך הסיור בונים את תת הסדרה המשותפת המכסימלית, מן הסוף אל ההתחלה.

לפני תחילת הסיור מאתחלים:

$$(j,i) \leftarrow (m,n)$$

$$LCS \leftarrow \Lambda$$

הזוג (i,j) נקרא **מיקום האלגוריתם.**

בכל צעד בסיור, עוברים מן המיקום (i,j) אל אחד האיברים הסמוכים.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

תיאור האלגוריתם (המשך)

המעבר מ-(i,j) אל המיקום הבא, תלוי בתצורה של איבר זה: אם זוהי אחת מן התצורות 1-3, האלגוריתם עובר אל $(i-1,j)$.

שימו לב: בתצורות 1-2, אפשר לעבור הן אל $(i-1,j)$ והן אל $(i,j-1)$. מצב זה קורה, כאשר ל-X_i ול-Y_j יש יותר מ-LCS יחידה. המעבר אל $(i-1,j)$, קובע את תת הסדרה שהאלגוריתם בונה. מעבר אל $(i,j-1)$ יניב תת סדרה משותפת מכסימלית **אחרת**.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

אלגוריתם לחישוב תת הסדרה המכסימלית - צעד (המשך)

אם (i,j) הוא בתצורה 4, המיקום הבא הוא $(i,j-1)$.

בכל המקרים שתוארו עד כה, הערך של $TL[i,j]$, התקבל מאיבר שכן, ללא תוספת תו לתת הסדרה המכסימלית.

אם לעומת זאת (i,j) הוא בתצורה 5, המיקום הבא הוא $(i-1,j-1)$.

במקרה זה, מתקיים $x_i = y_j$ ותוך כדי המעבר אל $(i-1,j-1)$ האלגוריתם מוסיף לתת הסדרה המשותפת את התו המשותף x_i על ידי ביצוע

$$LCS \leftarrow x_i \circ LCS .$$

© כל הזכויות שמורות לפרופסור עמוס ישראלי

הקוד

```
Extract_LCS(TL)
  LCS ← Λ; (i, j) ← (n,m)
  while (i, j) ≠ (1,1) do
    if  TL[i, j] = TL[i−1, j]
                    (i, j) ← (i−1, j)
    elseif  (TL[i, j] = TL[i, j−1])
                    (i, j) ← (i, j−1)
    else
      (TL[i, j] ≠ TL[i−1, j]  and
           TL[i, j] ≠ TL[i, j−1])
           LCS ← x_i ∘ LCS
      (i, j) ← (i−1, j−1)
    endif
  endwhile
end
```

אלגוריתם 8.15: חילוץ LCS

© כל הזכויות שמורות לפרופסור עמוס ישראלי

סיכום

בהרצאה זו, הוצג אלגוריתם יעיל
לפתרון בעית **תת הסדרה המכסימלית**
תוך שימוש בתכנות דינמי.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

<u>הרצאה 9: תכנות דינמי - מכפלת מטריצות בשרשרת</u>

בהרצאה זו, נציג את בעיית **מכפלת מטריצות בשרשרת** ונציג אלגוריתם לפתרונה.
האלגוריתם מהווה הדגמה נוספת של **תכנות דינמי**.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

<u>מכפלת מטריצות בשרשרת</u>

תהינה M_1 ו-M_2 מטריצות. נניח כי ממדי M_1 (מספר השורות ומספר העמודות) הם x_1 ו-y_1 (נסמן זאת ב-(x_1, y_1)), וממדי M_2 (x_2, y_2).
אם $y_1 = x_2$ אז M_1 ו-M_2 ניתנות להכפלה. נסמן $M_{12} = M_1 \times M_2$.
מימדי M_{12} הם (x_1, y_2). נסמן את האיבר ה-(i, j) של M_{12} ב-m_{ij}.

$$m_{ij} = \sum_{k=1}^{x_2} a_{ik} b_{kj}$$

כאשר a_{ik} ו-b_{kj} , $1 \le k \le x_2$ הם איברי השורה ה-i והעמודה ה-j ב-M_1 וב-M_2.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

<u>מספר פעולות הכפל הסקאלריות הנדרשות לחישוב המכפלה</u>

לחישוב m_{ij} נדרשות x_2 פעולות כפל סקאלריות. מאחר שמספר אברי המטריצה M_{12} הוא $x_1 y_2$ נקבל כי מספר פעולות הכפל הנדרשות לחישוב M הוא $x_1 x_2 y_2$.
מספר פעולות החיבור הסקלארי הנדרשות דומה. לכן, מעתה נזניח את פעולות החיבור ונדון אך ורק במספר פעולות הכפל הסקלארי הנדרשות לביצוע המכפלה.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

<u>מספר פעולות הכפל הסקאלריות הנדרשות לחישוב המכפלה (המשך)</u>

נניח כעת כי ברצוננו לחשב מכפלה של שלוש מטריצות $M_1 \times M_2 \times M_3$ שמימדיהן הן (x_1, y_1), (x_2, y_2), ו-(x_3, y_3) בהתאמה.
נמספר את פעולות כפל המטריצות משמאל לימין, מספר הפעולה הראשונה הוא 1 ומספר השניה הוא 2.
כפל מטריצות הוא **אסוציאטיבי** ולכן באפשרותנו לבצע את הכפל בשני **זימונים** (סדרי פעולות) **שונים**, כמפורט בשקף הבא:

© כל הזכויות שמורות לפרופסור עמוס ישראלי

מספר פעולות הכפל הסקאלריות הנדרשות לחישוב המכפלה (המשך)

1. הזימון $(1,2)$ מסמן את סדר המכפלות $(M_1 \times M_2) \times M_3$ -
במקרה זה מספר הפעולות יהיה:
$$x_1 x_2 y_2 + x_1 x_3 y_3 = x_1 x_3 (x_2 + y_3)$$
2. הזימון $(2,1)$ מסמן את סדר המכפלות $M_1 \times (M_2 \times M_3)$ -
במקרה זה מספר הפעולות יהיה:
$$x_2 x_3 y_3 + x_1 x_2 y_3 = x_2 y_3 (x_3 + x_1)$$

© כל הזכויות שמורות לפרופסור עמוס ישראלי

הדגמת זימון

בטרם נגדיר את הבעיה חישובית באופן פורמלי, נדגים מהו זימון:
נניח כי $n = 5$ כלומר
$$M_{15} = M_1 \times M_2 \times M_3 \times M_4 \times M_5$$
נתבונן בזימון $(4,1,3,2)$.
זימון זה מכתיב כי פעולת הכפל הראשונה שתתבצע היא $M_4 \times M_5$.
לאחר ביצוע פעולה זו נישאר עם:
$$M_{15} = M_1 \times M_2 \times M_3 \times M_{45}$$
אפשר לציין זאת על ידי:
$$M_{15} = M_1 \times M_2 \times M_3 \times (M_4 \times M_5) .$$
כעת נבצע את פעולה מס' 1 ונישאר עם
$$M_{15} = M_{12} \times M_3 \times M_{45}$$

© כל הזכויות שמורות לפרופסור עמוס ישראלי

הדגמת זימון (המשך)

אפשר לציין זאת על ידי:
$$M_{15} = M_1 \times M_2 \times M_3 \times (M_4 \times M_5)$$
כעת נבצע את פעולה מס' 3 $M_3 \times M_{45}$
ונשאר עם $M_{15} = M_{12} \times M_{35}$.
לבסוף נבצע את פעולה מס' 2 ונקבל את התוצאה הסופית, M_{15}.
סדר ביצוע הפעולות המוכתב על ידי הזימון הוא:
$$M_{15} = (M_1 \times M_2) \times [M_3 \times (M_4 \times M_5)]$$
בזימון זה, פעולה מס' 2 נקראת **הפעולה הראשית** או הפעולה האחרונה.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

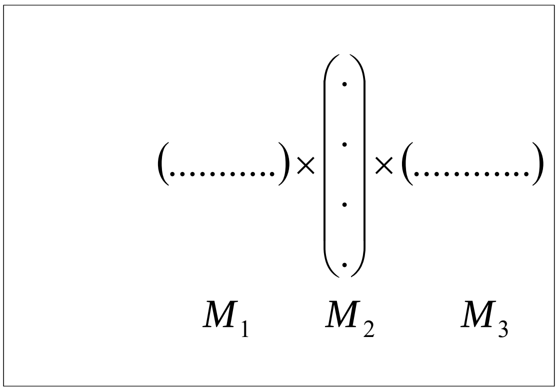
זימון מכפלות אופטימלי

בהנתן שלישית מטריצות ניתנות להכפלה, קימים שני זימונים אפשריים לביצוע המכפלה: $(1,2)$, ו- $(2,1)$.
אפשר לחשב את עלות שני הזימונים ולבחור את הזימון הדורש ביצוע פחות פעולות כפל סקאלריות, כמתואר בדוגמה 9.1.

דוגמה 9.1

נניח כי רוצים להכפיל 3 מטריצות שמימדיהן הם: $(x_1, y_1) = (1, n)$,
$(x_2, y_2) = (n, 1)$, ו-$(x_3, y_3) = (1, n)$
כמתואר באיור 9.2

© כל הזכויות שמורות לפרופסור עמוס ישראלי



איור 9.2: מכפלת מטריצות

אם נכפיל את המטריצות בדוגמה לפי זימון $(1,2)$, מספר פעולות הכפל הסקלריות הנדרש הוא $2n$. לעומת זאת, אם נבצע את כפל המטריצות לפי הזימון $(2,1)$, מספר פעולות הכפל הסקלריות הנדרשות הוא $2n^2$.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

אם נתבקש להכפיל שלוש מטריצות נתונות, אפשר לחשב את עלות שני הזימונים ולהחליט ביניהם.
נעבור כעת לדון בבעיית בחישוב זימון אופטימלי עבור מספר מטריצות גדול.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

הגדרה פורמלית של הבעיה

הקלט

n זוגות של מספרים טבעיים $(x_1, y_1),\dots,(x_n, y_n)$. לכל $1 \le i \le n-1$, (x_i, y_i) הוא זוג מימדי המטריצה M_i ולכל $1 \le i \le n-1$ מתקיים $y_i = x_{i+1}$.

שימו לב: המטריצות עצמן אינן חלק מן הקלט.

הפלט

1. מספר פעולות הכפל הסקאלריות המינימלי הנדרש לחישוב המכפלה
$$M = M_1 \times \ldots \times M_n$$
2. זימון פעולות כפל המטריצות המניב את מספר הפעולות המינימלי

© כל הזכויות שמורות לפרופסור עמוס ישראלי

ניסיון 1: האלגוריתם הישיר (בדיקת כל האפשרויות)

סימונים

1. לכל $1 \le i < j \le n$ נסמן
$$M_{ij} = M_i \times \ldots \times M_j$$
2. לכל $1 \le i \le n-1$ מספרה של הפעולה בין M_i לבין M_{i+1} יהיה i.

מאחר שיש n-1 פעולות כפל מטריצות, הזימון (פלט מס' 2) מהווה **תמורה** מתוך S_{n-1}.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

האלגוריתם הישיר

האלגוריתם הישיר עובר על כל אחד מן הזימונים, מחשב את עלותו (מספר המכפלות הסקאלריות) של כל זימון ובוחר את הזימון בעל העלות המינימלית.

סיבוכיות האלגוריתם הישיר

מאחר שכל זימון הוא תמורה מתוך S_{n-1}, ברור כי מס' האפשרויות הוא $(n-1)!=\Omega(2^n)$ ואפשרות זו נפסלת.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

ניסיון 2: אלגוריתם הפרד ופתור

לכל זוג מספרים (i, j), $i \le j$, נסמן ב-$mc(i, j)$ את מספר פעולות הכפל המינימלי הנדרש כדי לחשב את M_{ij} .

כעת נפתח אלגוריתם רקורסיבי לחישוב $mc(i, j)$. האלגוריתם מבוסס על שתי הלמות הבאות:

למה 9.3

לכל $1 \le i \le j \le n$, מתקיים:

1. $mc(i,i)=0$.
2. $mc(i,i+1)=x_i x_{i+1} y_{i+1}$.

הוכחה

נכונות סעיף 1 נובע מיידית מהגדרת $mc(i, j)$. סעיף 2 הוכח בתחילת ההרצאה. **מש"ל**

© כל הזכויות שמורות לפרופסור עמוס ישראלי

אלגוריתם הפרד ופתור (המשך)

הלמה הבאה מציבה משוואה רקורסיבית כדי לחשב את $mc(i, j)$:

למה 9.4

לכל $1 \le i \le j \le n$, מתקיים:

אם $j > i+1$ אז

$$mc(i, j)=\min_{k=i}^{j-1}\left(mc(i,k)+mc(k+1, j)+x_i x_{k+1} y_j\right)$$

להלן נסמן את המשוואה הזו ב-(*).

© כל הזכויות שמורות לפרופסור עמוס ישראלי

הוכחה

יהי k מס' טבעי, $1 \le k \le j-1$. נניח כי הפעולה הראשית בחישוב M_{ij} היא פעולה מס' k, כלומר החישוב נראה כך:

$$M_{ij}=M_{ik}\times M_{k+1,j}$$

בתנאים אלה, מס' פעולות הכפל המינימלי הנדרש לחישוב הוא:

$$mc(i,k)+mc(k+1, j)+x_i x_{k+1} y_j$$

מאחר שאיננו יכולים לנחש מהי הפעולה הראשית עלינו לבדוק את כל האפשרויות.

מש"ל

© כל הזכויות שמורות לפרופסור עמוס ישראלי

תכונות הנוסחה (*)

נשים לב כי :

1. אם ערך המינימום התקבל עבור k אזי נובע כי פעולה k היא **הפעולה הראשית (האחרונה)** בזימון אופטימלי.
2. המספר המינימלי של פעולות הכפל הסקאלריות הנדרש להכפלת כל n המטריצות הוא כמובן $mc(1,n)$.

האלגוריתם הרקורסיבי – הקוד

$mc(i,j)$
  if $(j=i)$ **return**(0)
  if $(j=1+1)$ **return**$(x_i \cdot x_j \cdot y_j)$
  $MC = \min_{k=i}^{j-1}\{mc(i,k)+$
    $+mc(k+1,j)+x_i x_{k+1} y_j\}$
return(MC)
end

אלגוריתם 9.5 - חישוב $mc(i,j)$

נכונות האלגוריתם

האלגוריתם הוא פשוט ביותר. תנאי הקצה נובעים מלמה 9.3 והרקורסיה מבוססת ישירות על למה 9.4.

סיבוכיות האלגוריתם

כדי לחסום את מלמטה את סיבוכיות האלגוריתם, נתבונן בעץ הקריאות הרקורסיביות : בעץ זה, דרגת היציאה (מספר הקריאות הרקורסיביות) של כל צומת, נעה בין $2(n-1)$ (עבור השורש) לבין 2 (ברמה התחתונה) ואורך כל ענף גדול מ-$n-1$.

קל לראות כי מספר הצמתים בעץ גדול (בהרבה מאוד) מ-2^n ומכאן נובע כי סיבוכיות האלגוריתם היא מעריכית (אקספוננציאלית)

בנספח 1 אנו מוכיחים כי סיבוכיות האלגוריתם חסומה מלמטה על ידי 2^{n-1}.

מסקנה: גם שיטה זו בזבזנית מדי.

ניסיון 3 : תכנות דינמי

כדי לפתח אלגוריתם יעיל עלינו למצוא את הגורם לבזבוז בשיטה הקודמת :
אם נפתח את המשוואות הרקורסיביות שלבים נוספים, נראה כי ישנם זוגות מספרים (k,l) כך ש $mc(k,l)$ מחושב פעמים רבות כנראה בדוגמה 9.6.

דוגמה 9.6

כדי לחשב את $mc(1,5)$ יש למצוא את המינימום בין:

$$mc(1,1)+mc(2,5)+x_1x_2y_5$$
$$mc(1,2)+mc(3,5)+x_1x_3y_5$$
$$mc(1,3)+mc(4,5)+x_1x_4y_5$$
$$mc(1,4)+mc(5,5)+x_1x_5y_5$$

כדי לחשב את $mc(2,5)$ יש למצוא את המינימום בין:

$$mc(2,2)+mc(3,5)+x_2x_3y_5$$
$$mc(2,3)+mc(4,5)+x_2x_4y_5$$
$$mc(2,4)+mc(5,5)+x_2x_5y_5$$

© כל הזכויות שמורות לפרופסור עמוס ישראלי

דוגמה 9.6 (המשך)

כבר כאן אנו רואים כי בדרך זו אנו מחשבים את $mc(3,5)$ ואת $mc(4,5)$ פעמיים.

אם נמשיך בדוגמא, הכפילות תלך ותגדל וזמן הריצה יגיע לפונקציה מעריכית.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

תכנות דינמי

נשים לה כי לכל אחד משני הארגומנטים באלגוריתם הרקורסיבי $mc(i,j)$ יש בדיוק n ערכים מותרים. מכאן נובע כי מספר צרופי הארגומנטים האפשריים הוא n^2. כדי להמנע מחזרה מיותרת על חישובי ביניים, נשמור את התוצאה של כל חישוב **בטבלת מעקב** (Lookup Table) בגודל n^2.

נסמן את טבלת המעקב באות T. בכל פעם שנחשב את $mc(k,l)$ עבור k ו-l כלשהם, נאחסן את הערך שחישבנו ב-$T[k,l]$.

כאשר נדרש לערך זה בפעם הבאה, נשלוף אותו מן הטבלה במקום לחשבו מחדש.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

תכנות דינמי (המשך)

למעשה האלגוריתם יתקדם על ידי חישוב $mc(i,j)$ עבור זוגות (i,j) שמרחקם ההדדי, $j-i$ גדל והולך, כדלקמן:

שלב 0: חשב באופן ישיר, על ידי שימוש בתנאי קצה מס' 1 את אלכסון 0 (האלכסון הראשי):

$$T[1,1],...,T[n,n]$$

שלב 1: חשב באופן ישיר, על ידי שימוש בתנאי קצה מס' 2 את אלכסון 1 (האלכסון מעל האלכסון הראשי):

$$T[1,2],...,T[n-1,n]$$

© כל הזכויות שמורות לפרופסור עמוס ישראלי

תכנות דינמי (המשך)

שלב 2: השתמש בנוסחה (*) ובתוצאות שלבים 0 ו-1 כדי לחשב את אלכסון 2:

$$T[1,3],...,T[n-2,n]$$

שלב i: השתמש בנוסחה (*) ובתוצאות שלבים 0 עד $i-1$ כדי לחשב את אלכסון i:

$$T[1,i+1],...,T[n-i,n]$$

בשלב $n-2$ מחשבים את $T[1,n-1]$ ואת $T[2,n]$.

בשלב $n-1$ מחשבים את $T[1,n]$.

שימו לב: בשלב i, $0 \le i < n$, מחשבים $n-i$ איברים.

תכנות דינמי (המשך)

כפי שמודגם באיור 9.6, מתקבל מערך ובו n שורות ו-n עמודות. חישוב הערכים במערך מתבצע כך: האלכסון הראשי מתמלא ראשון ואחריו האלכסונים מעל הראשי בזה אחר זה.

$$\begin{pmatrix} (1,1)(1,2).......................................(1,n-1)(1,n) \\ (2,2)(2,3)..............................(2,n-1)(2,n) \\ (3,3)...............................(3,n-1)(3,n) \\ (4,4)......................................(4,n) \\ .... \\ .... \\ ....(n-1,n-1)(n-1,n) \\ ....(n,n) \end{pmatrix}$$

איור 9.7: הטבלה T

אלגוריתם *Dmc* לחישוב הטבלה *T*

```
Dmc(x_1, y_1, ....x_n, y_n)
for i = 1 to n
                    \\ חישוב אלכסון 0
      T[i,i] ← 0
for i = 1 to n−1
                    \\ חישוב אלכסון 1
        T[i,i+1] ← x_i x_{i+1} y_{i+1}
for l = 2 to n−1
                 \\ 2 ≤ l ≤ n−1 ,l חישוב אלכסון
    for i = 1 to n−l
      j ← i + l
      T[i, j] ← min_{k=i}^{j−1} {T[i+k] +
                T[k+1, j] + x_i x_{k+1} y_j
   end for
end for
Out(T)
```

אלגוריתם 9.8 – חישוב טבלת המעקב

סיבוכיות האלגוריתם

טענה

סיבוכיות האלגוריתם Dmc היא $\Theta(n^3)$.

הוכחה

כדי להוכיח זאת נחסום את זמן החישוב של המערך T מלמעלה ומלמטה.

שימו-לב: זמן החישוב של הזימון האופטימלי הוא לינארי ואינו משנה את סיבוכיות האלגוריתם.

חסימה הסיבוכיות מלמעלה

למעשה אנו מחשבים כאן איברי מערך ריבועי מסדר $n \times n$. חישוב איברי המערך, השונים מ-0, מתבצע בעזרת הנוסחה (*).

בנוסחה זו, אנו מוצאים את המינימום בין לכל היותר n איברים ולכן מספר הצעדים לחישוב כל איבר הוא $O(n)$.

מכאן, מספר הצעדים הכללי הוא לכל היותר:

$$O\left(n^2 \cdot n\right) = O\left(n^3\right)$$

חסימה הסיבוכיות מלמטה

מספר האיברים השונים מ-0 במחצית העליונה של המערך T הוא:

$$\Omega\left(\underbrace{\frac{n}{2}}_{\text{גובה המשולש}} \cdot \underbrace{\frac{n}{2}}_{\text{אורך הבסיס}} \cdot \frac{1}{2}\right) = \Omega\left(n^2\right)$$

אפשר גם לחשב מספר זה במדויק כדלהלן:

בשורה $n/2-1$ יש $n/2$ איברים $\neq 0$

בשורה $n/2-2$ יש $n/2-1$ איברים $\neq 0$

…

בשורה 1 יש איבר יחיד $\neq 0$

חסימה הסיבוכיות מלמטה (המשך)

לפי נוסחת הסכום של טור חשבוני, מס' האיברים הכולל השונה מ-0 במחצית העליונה של המערך T הוא:

$$\left(\frac{n}{2}+1\right)\cdot\frac{n}{2}\cdot\frac{1}{2} = \frac{n^2}{8}+\frac{n}{4}$$

חישוב כל איבר כזה דורש לפחות $n/2$ פעולות כלומר $\Omega(n)$ פעולות (חישוב מינימום של מספר איברים $\leq n/2$) ומכאן מספר הפעולות הכולל הוא $\Omega\left(n^3\right)$.

חישוב הזימון האופטימלי

לאחר חישוב הטבלה T, ידוע לנו כי המספר המינימלי של מכפלות סקלריות הנדרש לחישוב המטריצה $M = M_1 \times \ldots \times M_n$ מאוחסן ב-$T[1,n]$. כעת נסביר מהי התוספת הנדרשת לחישוב הזימון האופטימלי עצמו: לכל $1 \leq i \leq j \leq n$ נסמן את הפעולה האחרונה בזימון אופטימלי לחישוב M_{ij} ב-$LastOp(i,j)$. להלן נציג בטבלה T את הערכים $mc(i,j)$ ביחד עם $LastOp(i,j)$

זימון מכפלת מטריצות – דוגמא

להלן נדגים את חישוב הטבלה T, בצירוף חישוב הפעולה האחרונה, על הקלט הבא:

קלט: $(7,3),(3,8),(8,4),(4,7),(7,5)$

חישוב אלכסון 0 (האלכסון הראשי)

ערכי האיברים באלכסון הראשי הם 0. ערכי האיברים באלכסון הראשון מעל האלכסון הראשי, מחושבים ישירות מתנאי הקצה השני.

$$T[1,1],...,T[5,5]=0$$

© כל הזכויות שמורות לפרופסור עמוס ישראלי

חישוב אלכסון 1

נחשב את הערכים באלכסון 1 בחישוב ישיר, בעזרת תנאי הקצה המופיע בסעיף 1 בלמה 9.2, וכך נקבל:

$$T[1,2]=7\cdot 5\cdot 8=280$$

$$T[2,3]=3\cdot 8\cdot 4=96$$

$$T[3,4]=8\cdot 4\cdot 7=224$$

$$T[4,5]=4\cdot 7\cdot 5=140$$

מן ההגדרה ברור כי לכל i, $1\le i\le n-1$, מתקיים:

$$LastOp(i,i+1)=i.$$

© כל הזכויות שמורות לפרופסור עמוס ישראלי

חישוב אלכסון 1 (המשך)

איור 9.3 מציג את הטבלה T לאחר חישוב אלכסונים 0 ו-1, כאשר האות Λ מסמנת איבר שטרם הגדר.

$$\begin{pmatrix} 0 & |168(1)| & \Lambda & | & \Lambda & | & \Lambda \\ \Lambda | & 0 & |96(2)| & & \Lambda & | & \Lambda \\ \Lambda | & \Lambda & | & 0 & |224(3)| & & \Lambda \\ \Lambda | & \Lambda & |\Lambda & | & 0 & & |140(4) \end{pmatrix}$$

איור 9.3: הטבלה T לאחר חישוב אלכסונים 0 ו-1

האות Λ מסמנת איבר שטרם הוגדר.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

חישוב אלכסון 2

נעבור כעת לחישוב אלכסון מס' 2. (האלכסון השני מעל האלכסון הראשי):

$$mc(1,3)=\min\{$$
$$mc(1,1)+mc(2,3)+x_1\cdot x_2\cdot y_3,$$
$$mc(1,2)+mc(3,3)+x_1\cdot x_3\cdot y_3\}=$$
$$=\min\{0+96+84,168+0+224\}=180$$

המינימום מתקבל עבור האיבר הראשון והפעולה הראשית היא פעולה מס' 1. הזימון האופטימלי הוא

$$M_{13}=M_1\times(M_2\times M_3)$$

© כל הזכויות שמורות לפרופסור עמוס ישראלי

חישוב אלכסון 2 (המשך)

באופן דומה:

$$mc(2,4) = \min\{$$
$$mc(2,2) + mc(3,4) + x_2 \cdot x_3 \cdot y_4,$$
$$mc(2,3) + mc(4,4) + x_2 \cdot x_4 \cdot y_4\} =$$
$$= \min\{0 + 224 + 168, 96 + 0 + 84\} = 180$$

הפעולה הראשית היא פעולה מס' 3

והזימון האופטימלי הוא

$$M_{24} = (M_2 \times M_3) \times M_4$$

© כל הזכויות שמורות לפרופסור עמוס ישראלי

חישוב אלכסון 2 $mc(3,5)$

$$mc(3,5) = \min\{$$
$$mc(3,3) + mc(4,5) + x_3 \cdot x_4 \cdot y_5,$$
$$mc(3,4) + mc(5,5) + x_3 \cdot x_5 \cdot y_5\} =$$
$$\min\{0 + 140 + 160, 224 + 0 + 280\} = 300$$

הפעולה הראשית היא פעולה מס' 3,

והזימון האופטימלי הוא

$$M_{35} = M_3 \times (M_4 \times M_5)$$

$$\begin{pmatrix} 0 & 168(1) & 180(1) & \Lambda & \Lambda \\ \Lambda & 0 & 96(2) & 180(3) & \Lambda \\ \Lambda & \Lambda & 0 & 224(3) & 300(3) \\ \Lambda & \Lambda & \Lambda & 0 & 140(4) \end{pmatrix}$$

איור 9.4: הטבלה T לאחר חישוב אלכסון 2

© כל הזכויות שמורות לפרופסור עמוס ישראלי

חישוב אלכסון 3

$$mc(1,4) = \min\{$$
$$mc(1,1) + mc(2,4) + x_1 \cdot x_2 \cdot y_4,$$
$$mc(1,2) + mc(3,4) + x_1 \cdot x_3 \cdot y_4,$$
$$mc(1,3) + mc(4,4) + x_1 \cdot x_4 \cdot y_4\} =$$
$$= \min\{0 + 180 + 147, 168 + 224 + 392,$$
$$180 + 0 + 196\} = 327$$

הפעולה הראשית היא פעולה מס' 1,

הזימון האופטימלי לחישוב M_{14} הוא

$$M_{14} = M_1 \times M_{24}$$

© כל הזכויות שמורות לפרופסור עמוס ישראלי

חישוב אלכסון 3 (המשך)

נמשיך בחישוב $mc(2,5)$

$$mc(2,5) = \min\{$$
$$mc(2,2) + mc(3,5) + x_2 \cdot x_3 \cdot y_5,$$
$$mc(2,3) + mc(4,5) + x_2 \cdot x_4 \cdot y_5,$$
$$mc(2,4) + mc(5,5) + x_2 \cdot x_5 \cdot y_5\} =$$
$$= \min\{0 + 300 + 120, 96 + 140 + 60,$$
$$180 + 0 + 105\} = 285$$

הפעולה הראשית היא פעולה מס' 4.

הזימון האופטימלי הוא:

$$M_{25} = M_{24} \times M_5$$

© כל הזכויות שמורות לפרופסור עמוס ישראלי

חישוב אלכסון 4

נסיים את חישוב המערך בחישוב האלכסון הרביעי ובו האיבר היחיד $mc(1,5)$:

$$\begin{aligned} mc(1,5) = \min\{ & \\ & mc(1,1)+mc(2,5)+x_1\cdot x_2\cdot y_5, \\ & mc(1,2)+mc(3,5)+x_1\cdot x_3\cdot y_5, \\ & mc(1,3)+mc(4,5)+x_1\cdot x_4\cdot y_5, \\ & mc(1,4)+mc(5,5)+x_1\cdot x_5\cdot y_5\} = \\ = \min\{ & 0+285+105, 168+295+280, \\ & 180+135+140, 327+0+245\} = 390 \end{aligned}$$

הפעולה הראשית היא פעולה מספר 1 והזימון האופטימלי לחישוב M הוא

$$M = M_1 \times M_{25}$$

© כל הזכויות שמורות לפרופסור עמוס ישראלי

סיכום הדוגמא

המערך הסופי נראה כך :

$$\begin{pmatrix} 0 & 168(1) & 180(1) & 327(1) & 390(1) \\ \Lambda & 0 & 96(2) & 180(3) & 285(4) \\ \Lambda & \Lambda & 0 & 224(3) & 300(3) \\ \Lambda & \Lambda & \Lambda & 0 & 140(4) \end{pmatrix}$$

איור 9.5: הטבלה T בסיום החישוב

עלות הזימון האופטימלי מופיעה ב-$T[1,5]$

כעת נעבור לחישוב הזימון האופטימלי.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

חישוב הזימון האופטימלי

נסמן את הזימון האופטימלי ב-$OptSched$. בתחילת החישוב מתקיים $OptSched = \Lambda$ כעת נתבונן ב-$T[1,5]$ ונקבל:

$$LastOp(1,5) = 1$$

כלומר הזימון האופטימלי לחישוב M_{15} נראה כך : $M_{15} = M_1 \times M_{25}$.
ומכאן $OptSched = (.,.,.,.,1)$.
כדי להמשיך בחישוב הזימון האופטימלי נתבונן כעת ב-$T[2,5]$ ונמצא כי $mc(2,5) = 285$ וכי $LastOp(2,5) = 4$

© כל הזכויות שמורות לפרופסור עמוס ישראלי

חישוב הזימון האופטימלי (המשך)

מכאן נובע כי $OptSched = (.,.,4,1)$ והזימון האופטימלי יהיה:

$$M_{15} = M_1 \times (M_{24} \times M_5)$$

נותר לנו לתזמן את פעולות 2 ו 3.
כעת נתבונן ב-$T[2,4]$ ונגלה כי $mc(2,4) = 180$ וכי $LastOp(2,4) = 3$. מכאן נובע כי $OptSched = (.,3,4,1)$. מכאן נובע כי הזימון האופטימלי יהיה:

$$M_{15} = M_1 \times ((M_{23} \times M_4) \times M_5)$$

ברור כי הפעולה הראשונה תהיה 2 ולכן הזימון האופטימלי הסופי יהיה: $OptSched = (2,3,4,1)$.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

<u>חישוב הזימון האופטימלי (המשך)</u>

נסה לוודא זאת על ידי חישוב עלות זימון זה באופן ישיר. הקלט הוא:

$$(7,3),(3,8),(8,4),(4,7),(7,5)$$

נחשב ונקבל:

$$x_2\cdot x_3\cdot y_3 + x_2\cdot x_4\cdot y_4 +$$
$$x_2\cdot x_5\cdot y_5 + x_1\cdot x_2\cdot y_5 =$$
$$3\cdot 8\cdot 4+3\cdot 4\cdot 7+$$
$$3\cdot 7\cdot 5+7\cdot 3\cdot 5=390$$

כפי שציפינו.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

<u>זימונים שקולים</u>

נניח כי עבור ערכי קלט מסוימים מתקבל זימון הפעולות הבא:

$$OptSched=(4,1,2,3)$$

במקרה זה, הפעולה הראשית המתבצעת היא פעולה מס׳ 3 אשר **מפרידה** בין פעולות 1,2 לפני פעולה 4. אם נבצע את פעולות 1,2 **לפני** פעולה 4, מספר פעולות הכפל הסקאלריות **לא ישתנה**. הזימונים $Sched=(4,1,2,3)$ ו-$Sched'=(1,2,4,3)$ נקראים **שקולים.**

© כל הזכויות שמורות לפרופסור עמוס ישראלי

<u>זימונים שקולים (המשך)</u>

כדי לבחור בין כמה זימונים שקולים, נסכים כי תמיד נבחר בפעולות שמספרן נמוך יותר. בדוגמא הנוכחית הזימון שייבחר הוא $Sched'$.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

<u>השגרה *Extract*</u>

השגרה *Extract* מחשבת ומחזירה זימון אופטימלי תוך שימוש בנתונים שנאגרו במערך T.

$Extract(i,j,T)$
if $i=j$ **return** Λ
$last \leftarrow lastOp(i,j)$
$LSched = Extract(i,last)$
$RSched = Extract(last+1,j)$
Return $LSched \circ RSched \circ Last$

<u>אלגוריתם 9.3 - חישוב זימון אופטימלי</u>

שימו לב: הסימון $\circ$ מסמל פעולת שרשור (Concatenation).

© כל הזכויות שמורות לפרופסור עמוס ישראלי

<u>האלגוריתם הסופי</u>

האלגוריתם הסופי משתמש בשגרה Dmc כדי לחשב את הטבלה T. לאחר מכן, האלגוריתם מפעיל את השגרה $Extract$ אשר מחזירה את הזימון האופטימלי. להלן הקוד עבור האלגוריתם הסופי:

$OptSched\left(x_1, y_1, ...x_n, y_n\right)$

$T = Dmc\left(x_1, y_1, ...x_n, y_n\right)$

return $Extract\left(1, n, T\right)$

<u>אלגוריתם 9.4 – האלגוריתם הסופי לחישוב זימון אופטימלי</u>

<u>סיכום</u>

בהרצאה זו, השתמשנו פעם נוספת ב**תכנות דינמי** כדי להציג אלגוריתם פולינומי לחישוב **זימון אופטימלי להכפלת מטריצות בשרשרת**.

<u>נספח 1: סיבוכיות האלגוריתם הרקורסיבי</u>

נסמן ב $T(n)$ את הזמן הנדרש כדי לחשב את $mc(1,n)$, כלומר כדי לחשב זימון אופטימלי למכפלת n מטריצות. אזי מתקיים:

$$T(2) = 2$$

$$T(n) = \sum_{k=1}^{n-1}\left(T(k) + T(n-k) + 2\right)$$

שימו לב: המספר 2 המופיע בתוך הסכימה שווה לזמן הנדרש לביצוע המכפלה $x_i y_k y_b$.

<u>סיבוכיות האלגוריתם (המשך)</u>

נראה כעת כי הסיבוכיות היא מעריכית.

<u>טענה</u>

$T(n) \ge 2^{n-1}$.

<u>הוכחה</u>

באינדוקציה על מספר המטריצות n.

<u>בסיס</u>

$$T(2) = 2 = 2^1$$

<u>צעד האינדוקציה</u>

$$T(n)=\sum_{k=1}^{n-1}\left(T(k)+T(n-k)+2\right)$$

$$\geq\sum_{k=1}^{n-1}T(k)+T(n-k)+1$$

אם נוציא את 1 מחוץ לסכימה נקבל:

$$T(n)\geq n-1+\sum_{k=1}^{n-1}T(k)+T(n-k)$$

$$\geq n-1+2\sum_{k=1}^{n-1}T(k)$$

© כל הזכויות שמורות לפרופסור עמוס ישראלי

<u>צעד האינדוקציה (המשך)</u>

לפי הנחת האינדוקציה, לכל $1\leq k<n$, מתקיים $T(k)\geq 2^{k-1}$. כעת, נציב את אי השוויונים הללו בנוסחה הקודמת ונקבל:

$$T(n)\geq n-1+2\sum_{k=1}^{n-1}2^{k-1}$$

$$\geq n-1+2\left(2^{n-1}-1\right)$$

$$=n+2^{n}-3\geq 2^{n-1}$$

מש"ל

© כל הזכויות שמורות לפרופסור עמוס ישראלי

הרצאה 10: משפט האב (Master Theorem) אלגוריתם Strassen למכפלת מטריצות ריבועיות

להרצאה זו, שני חלקים:

1. נציג ונוכיח את משפט האב המאפשר חישוב סיבוכיות של אלגוריתמים רקורסיביים רבים.
2. נציג את אלגוריתם Strassen להכפלת מטריצות ריבועיות ונחשב את הסיבוכיות שלו, בעזרת משפט האב.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

משפט האב

קיימים אלגוריתמים רקורסיביים רבים שאנו מתקשים בחישוב הסיבוכיות שלהם. משפט האב מספק לנו מכשיר רב עוצמה לטיפול ברוב האלגוריתמים הרקורסיביים. אנו נלמד גרסה מסויימת של משפט האב. גרסה אחרת נמצאת בספר הקורס.

שימו לב: אתם כבר למדתם את משפט האב בקורס מבנה נתונים.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

התנאים להפעלת משפט האב

משפט האב מאפשר לנו לדון באלגוריתמים רקורסיביים המקיימים את התנאים הבאים:

לכל בעיה בגודל n, אשר אינה מקרה קצה, האלגוריתם מבצע a קריאות רקורסיביות לפתרון a תת בעיות בגודל n/b.

שימו לב

1. a ו b הם קבועים התלויים בבעיה אך לא ב n.
2. a תת הבעיות אינן בהכרח חלוקה של הקלט ל a חלקים זרים.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

דוגמא

מיון ע"י מיזוג:

1. **חלק** את מערך הקלט ל 2 מערכים בגודל $n/2$. ($a=b=2$).
2. **מיין** כל תת-מערך באופן על ידי קריאה רקורסיבית למיון-מיזוג.
3. **מזג** את שתי המחציות הממוינות.

לעומת דוגמא זו, **מיון מהיר** אינו עונה לדרישות אלה משום שבמיון מהיר מחלקים את מערך הקלט לשני קבצים שהיחס בין גדליהם אינו קבוע.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

הצגת סיבוכיות מיון-מיזוג על ידי מערכת משוואות רקורסיביות

נסמן ב $T(n)$ את הזמן הדרוש למיון-מיזוג מערך בן n איברים.
הזמן הנדרש לכל קריאה רקורסיבית הוא כמובן $T\left(\frac{n}{2}\right)$ ומאחר שמיזוג של שני מערכים באורך $n/2$ כל אחד אורך זמן ליניארי ומיון קובץ באורך 1 אורך זמן קבוע כלשהו נקבל כי:

$$T(n) = 2T\left(\frac{n}{2}\right) + dn$$

$$T(1) = c$$

עבור c ו d קבועים כלשהם.

הכללת מערכת המשוואות

נתבונן באלגוריתם רקורסיבי A לפתרון בעיה כלשהי P. נניח כי אם מריצים את A על קלט בגודל n אזי:

1. נדרשות a קריאות רקורסיביות עם קלט בגודל n/b.
2. מספר הפעולות בשגרה הקוראת (top level), למעט הקריאות הרקורסיביות, הוא $df(n)$ כאשר $f(n)$ היא פונקציה כלשהי התלויה בגודל הקלט.
3. פתרון מקרה הקצה של הרקורסיה, עבור קלט בגודל 1, דורש c פעולות.
4. הקבועים a, b, c, d והפונקציה f תלויים בבאלגוריתם A.

הכללת מערכת המשוואות (המשך)

נסמן ב$T(n)$ את הזמן הנדרש להרצת האלגוריתם A על קלט בגודל n.
אזי מתקיים:

$$T(n) = aT\left(\frac{n}{b}\right) + df(n)$$

$$T(1) = c$$

להלן נציג את משפט האב (Master Theorem) הפותר את מערכת המשוואות הזו באופן כללי.

משפט הMaster

תהי $T(n)$ פונקציה המוגדרת על ידי:

$$T(n) = aT\left(\frac{n}{b}\right) + df(n)$$

$$T(1) = c$$

אזי אם f היא פונקציה כפלית (תוגדר להלן) התנהגות הפונקציה $T(n)$ תלויה ביחס בין a לבין $f(b)$ כמתואר להלן:

1. אם $a > f(b)$, $T(n) = \Theta\left(n^{\log_b a}\right)$.
2. אם $f(b) = a$, $T(n) = \Theta\left(n^{\log_b a} \log n\right)$.
3. אם $a < f(b)$, $T(n) = \Theta(f(n))$.

הערות למשפט הMaster

1. **במקרה 1 -** רוב צעדי A מתבצעים על ידי הקריאות הרקורסיביות.
   במקרה 2 - רוב צעדי A מתבצעים מחוץ לקריאות הרקורסיביות.
   במקרה 3 - שני הגורמים הקודמים תורמים לסיבוכיות.
2. לערכי c וd אין שום השפעה על $T(n)$.
3. בספר מוצגת גרסה קצת שונה של המשפט, אשר אינה מניחה כי הפונקציה f היא כפלית. במקרה זה, הטיפול המתמטי בבעיה שונה, אולם ההשלכות המעשיות דומות.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

הוכחת משפט הMaster

הוכחת משפט ה-Master מתקבלת על ידי פתרון המשוואות הרקורסיביות.
אנו נטפל במשוואה פשוטה יותר:

$$T(n) = aT\left(\frac{n}{b}\right) + f(n)$$

נציב בנוסחה את $T\left(\frac{n}{b}\right)$ ונקבל:

$$T(n) = a\left[aT\left(\frac{n}{b^2}\right) + f\left(\frac{n}{b}\right)\right] + f(n)$$

נפשט ונקבל:

$$T(n) = a^2T\left(\frac{n}{b^2}\right) + af\left(\frac{n}{b}\right) + f(n)$$

© כל הזכויות שמורות לפרופסור עמוס ישראלי

פתרון המשוואות הרקורסיביות

נמשיך להציב את $T\left(\frac{n}{b^2}\right)$ ונקבל:

$$T(n) = a^2\left[aT\left(\frac{n}{b^3}\right) + f\left(\frac{n}{b^2}\right)\right] + af\left(\frac{n}{b}\right) + f(n))$$

נפשט ונקבל:

$$T(n) = a^3T\left(\frac{n}{b^3}\right) + a^2 f\left(\frac{n}{b^2}\right) + af\left(\frac{n}{b}\right) + f(n))$$

$$= a^3T\left(\frac{n}{b^3}\right) + \sum_{i=0}^{2} a^i f\left(\frac{n}{b^i}\right)$$

© כל הזכויות שמורות לפרופסור עמוס ישראלי

פתרון המשוואות הרקורסיביות

כדי שהרקורסיה תיגמר יש להגיע עד למקרה הקצה כלומר $T(1)$. כדי להשיג זאת, על הקריאה הרקורסיבית להתבצע לעומק i, המקיים $n = b^i$ כלומר $i = \log_b n$.
המשך ההצבה $\log_b n$ פעמים מניב:

$$T(n) = a^{\log_b n}T\left(\frac{n}{b^{\log_b n}}\right) + \sum_{i=0}^{(\log_b n)-1} a^i f\left(\frac{n}{b^i}\right)$$

אם נציב $n = b^{\log_b n}$ ונזכור כי $T(1) = c$ נקבל:

$$T(n) = a^{\log_b n}\, T(1) + \sum_{i=0}^{(\log_b n)-1} a^i f\left(\frac{b^{\log_b n}}{b^i}\right)$$

© כל הזכויות שמורות לפרופסור עמוס ישראלי

<u>פתרון המשוואות הרקורסיביות</u>

כעת נשים לב כי:

$$a^{\log_b n} = \left(b^{\log_b a}\right)^{\log_b n} =$$

$$= \left(b^{\log_b n}\right)^{\log_b a} = n^{\log_b a}$$

אם נציב שני ביטויים אלה ונקבל כי

$$T(n) = \underbrace{cn^{\log_b a}}_{*} + \underbrace{\sum_{i=0}^{(\log_b n)-1} a^i f\left(b^{\log_b n-i}\right)}_{**}$$

המחובר הראשון, $(*)$, מתנהג כמו $\Theta\left(n^{\log_b a}\right)$ וזהו פתרון הבעיה כאשר $f(n)=0$, אולם אין זה ברור כלל וכלל איך אפשר לטפל במחובר השני $(**)$.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

<u>פונקציות כפליות</u>

כדי לטפל במחובר השני נניח כי f היא **פונקציה כפלית**.

פונקציה $f(x)$ היא **פונקציה כפלית** אם לכל x ו y f מקיימת:

$$f(xy) = f(x)f(y).$$

דוגמא: הפונקציה הפולינומית $f(x)=x^d$ (d קבוע) היא כפלית שכן

$$f(xy) = (xy)^d = x^d y^d = f(x)f(y)$$

שימו לב: באופן מעשי, רוב הפונקציות המתקבלות בשימוש במשפט האב הן כפליות.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

<u>תכונות פונקציות כפליות</u>

תהי f פונקציה כפלית כלשהי. אזי $f(1)$ מקיים:

$$f(1) = f(1\cdot 1) = f(1)\cdot f(1) = 1$$

נחשב כעת את ערך $f\left(\frac{a}{b}\right)$ עבור מנה כלשהי $\frac{a}{b}$:

$$f(1) = 1 = f\left(b\cdot\frac{1}{b}\right) = f(b)\cdot f\left(\frac{1}{b}\right) \Rightarrow$$

נחלק את שני האגפים ב $f(b)$ ונקבל:

$$f\left(\frac{1}{b}\right) = \frac{f(1)}{f(b)}$$

$$f\left(\frac{a}{b}\right) = f(a)\cdot f\left(\frac{1}{b}\right) = \frac{f(a)}{f(b)}$$

© כל הזכויות שמורות לפרופסור עמוס ישראלי

<u>פתרון המשוואות הרקורסיביות</u>

<u>עבור פונקציה כפלית f</u>

נניח כעת כי הפונקציה f היא **כפלית**.

במקרה כזה נפתח את $(**)$ ונקבל:

$$(**) = \sum_{i=0}^{(\log_b n)-1} a^i f\left(\frac{b^{\log_b n}}{b^i}\right) =$$

$$= \sum_{i=0}^{(\log_b n)-1} a^i \frac{f\left(b^{\log_b n}\right)}{f\left(b^i\right)} =$$

$$= f(b^{\log_b n}) \sum_{i=0}^{(\log_b n)-1} \frac{a^i}{f(b)^i} =$$

$$= (f(b))^{\log_b n} \sum_{i=0}^{(\log_b n)-1} \left(\frac{a}{f(b)}\right)^i =$$

וזהו טור גיאומטרי עם מנה $q = \frac{a}{f(b)}$

© כל הזכויות שמורות לפרופסור עמוס ישראלי

פתרון המשוואות הרקורסיביות

עבור פונקציה כפלית *f* (המשך)

אם נניח כי $a \neq f(b)$ ונשתמש בנוסחת הטור הגיאומטרי עבור $q = \frac{a}{f(b)}$ נקבל:

$$(**) = (f(b))^{\log_b n} \frac{\left(\frac{a}{f(b)}\right)^{\log_b n} - 1}{\frac{a}{f(b)} - 1} =$$

ולאחר פישוט נוסף נגיע ל

$$(**) = \frac{a^{\log_b n} - (f(b))^{\log_b n}}{\frac{a}{f(b)} - 1}$$

© כל הזכויות שמורות לפרופסור עמוס ישראלי

פתרון המשוואות הרקורסיביות

עבור פונקציה כפלית *f* (המשך):

נתבונן ב $(**)$

$$(**) = \frac{a^{\log_b n} - (f(b))^{\log_b n}}{\frac{a}{f(b)} - 1}$$

בביטוי זה, המכנה הוא קבוע ולכן ערכו של הביטוי תלוי ביחס בין a לבין $f(b)$ כמתואר בניתוח הבא:

© כל הזכויות שמורות לפרופסור עמוס ישראלי

מקרה 1

$a > f(b)$ - במקרה זה, ערך $(**)$ המונה מתנהג כמו $\Theta(a^{\log_b n})$ והמכנה הוא קבוע ולכן במקרה זה נקבל:

$$(**) = \Theta(a^{\log_b n}) = \Theta(n^{\log_b a})$$

מאחר שערך $(*)$ גם הוא $\Theta(n^{\log_b a})$ נקבל כי במקרה זה מתקיים:

$$T(n) = \Theta(n^{\log_b a}) .$$

© כל הזכויות שמורות לפרופסור עמוס ישראלי

מקרה 2

$a < f(b)$ - במקרה זה, ערך המונה הוא $\Theta(f(b^{\log_b n})) = \Theta(f(n))$ וסימנו שלילי. מאחר שגם סימן המכנה הוא שלילי נקבל:

$$(**) = \Theta(f(b^{\log_b n})) = \Theta(f(n))$$

מאחר שערך $(*)$ הוא $\Theta(a^{\log_b n})$ ו $a < f(b)$, במקרה זה, ערך $(*)$ קטן מערך $(**)$ ומתקיים כי

$$T(n) = \Theta(f(n)) .$$

© כל הזכויות שמורות לפרופסור עמוס ישראלי

מקרה 3

$$a = f(b)$$

במקרה זה, נוסחת הטור הגיאומטרי אינה שמישה, שכן ערך המכנה בנוסחה הוא 0. אולם, במקרה זה מתקיים:

$$(**) = \sum_{i=0}^{(\log_b n)-1} a^i \frac{f\left(b^{\log_b n}\right)}{f\left(b^i\right)} =$$

$$= \sum_{i=0}^{\log_b n-1} f(n) = \log_b n \cdot f(n)$$

ומאחר שמתקיים:

$$f(n) = f\left(b^{\log_b n}\right) = (f(b))^{\log_b n} =$$

$$= a^{\log_b n} = n^{\log_b a}$$

מקבלים כי במקרה זה:

$$T(n) = \Theta\left(n^{\log_b a} \log_b n\right)$$

© כל הזכויות שמורות לפרופסור עמוס ישראלי

פתרון המשוואות הרקורסיביות עבור פונקציה כפלית f (סיכום):

לכל פונקציה כפלית f, ערך הפונקציה T המוגדרת על ידי נוסחת הנסיגה

$$T(n) = aT\left(\frac{n}{b}\right) + df(n)$$

$$T(1) = c$$

הוא

1. עבור $a > f(b)$ $\Theta\left(n^{\log_b a}\right)$.
2. עבור $a < f(b)$ $\Theta(f(n))$.
3. עבור $a = f(b)$ $\Theta\left(n^{\log_b a}\log n\right)$.

תרגיל: אפשר (וצריך) לבדוק כי הוספת הקבוע d אינה משנה את הפיתוח.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

דרכים לשיפור אלגוריתם הפרד ופתור

יהי A אלגוריתם הפרד ופתור שסיבוכיותו מקיימת את מערכת המשוואות המופיעה במשפט האב. מהן הדרכים האפשריות לשיפור הסיבוכיות של A?

1. אם ש$a > f(b)$, נסה להקטין את $\log_b a$ על ידי הקטנת a או הגדלת b.
2. אם $a \leq f(b)$ נסה להקטין את סיבוכיות הפעולות שאינן כלולות בקריאות הרקורסיביות.
3. הקבועים c וd אינם מתבטאים בפתרון ולכן הקטנתם לא תשנה את הסיבוכיות.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

דוגמא - סיבוכיות מיון-מיזוג

כזכור, מיון מיזוג מקיים:

$$T(n) = 2T\left(\frac{n}{2}\right) + dn$$

$$T(1) = c$$

מאחר ש $a = b = 2$ ו $f(n) = n$, אנו מקבלים כי

$$a = 2 = f(2) = f(b)$$

ומכאן, סיבוכיות מיון-מיזוג היא

$$T(n) = \Theta\left(n^{\log_2 2}\log_2 n\right) = \Theta(n\log n)$$

כעת, נציג את אלגוריתם Strassen למכפלת מטריצות מרובעות. סיבוכיות האלגוריתם תקבע תוך שימוש במשפט האב.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

אלגוריתם Strassen

מכפלת מטריצות ריבועיות:

קלט: שתי מטריצות **ריבועיות** A ו B מסדר $n\times n$.

פלט: מכפלת הטריצות $C = A\times B$.

שימו לב: גודל הקלט הוא $O\left(n^2\right)$.

נפתח את ההרצאה בחזרה על האלגוריתם התקני ולאחר מכן נפתח את אלגוריתם Strassen . נסיים את ההרצאה בחישוב הסיבוכיות של האלגוריתם, שהיא $\Theta\left(n^{2.81...}\right)$, תוך שימוש במשפט האב.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

מכפלת מטריצות ריבועיות

מכפלת המטריצות A ו B היא מטריצה C, המקיימת

$$c_{ij} = \sum_{k=1}^{n} a_{ik} b_{kj}$$

ומסמנים $C = A\times B$.

מהי סיבוכיות בעיית חישוב מכפלת שתי מטריצות?

ברור כי $\Omega\left(n^2\right)$ היא חסם תחתון לסיבוכיות הבעיה (למה?) .

© כל הזכויות שמורות לפרופסור עמוס ישראלי

אלגוריתם ישיר

האלגוריתם הישיר לחישוב C הוא:

for $i = 1$ **to** n **do**
  for $j = 1$ **to** n **do**
    $c_{ij} \leftarrow 0$
    for $k = 1$ **to** n **do**
      $c_{ij} \leftarrow c_{ij} + a_{ik} \cdot b_{kj}$
    end
  end
end

זמן הריצה: $\Theta\left(n^3\right)$.

בהמשך ההרצאה נפתח אלגוריתם יעיל יותר המשתמש בשיטת הפרד ופתור ומשיג זמן ריצה $\Theta\left(n^{2.81...}\right)$.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

תצוגה מטריצית של כפל מטריצות

נפתח כעת אלגוריתם אחר, רקורסיבי, למכפלת מטריצות. נפתח על ידי הצגת מטריצה בעזרת 4 תת-מטריצות:

$$A = \begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix}$$

המטריצה A_{11} היא תת המטריצה ברביע השמאלי והעליון במטריצה A, כמתואר בציור. המטריצות A_{12}, A_{21}, ו-A_{22} ממוקמות באופן אנלוגי.

תחת תצוגה זו, המכפלה $C = A\times B$ מתקבלת כמכפלת שתי מטריצות מסדר 2×2 שאבריהן הן תת מטריצות של מטריצות הקלט.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

תצוגה מטריצית של כפל מטריצות

$$C=\begin{pmatrix}A_{11} & A_{12}\\ A_{21} & A_{22}\end{pmatrix}\times\begin{pmatrix}B_{11} & B_{12}\\ B_{21} & B_{22}\end{pmatrix}=$$

$$\begin{pmatrix}A_{11}B_{11}+A_{12}B_{21} & A_{11}B_{12}+A_{12}B_{22}\\ A_{21}B_{11}+A_{22}B_{21} & A_{21}B_{12}+A_{22}B_{22}\end{pmatrix}=$$

$$=\begin{pmatrix}C_{11} & C_{12}\\ C_{21} & C_{22}\end{pmatrix}$$

נכונות האלגוריתם הזה מוכחת באופן ישיר. לדוגמא: נניח כי A_{ij} וגם B_{ij} הן מטריצות מסדר 2×2: במקרה זה מתקיים:

© כל הזכויות שמורות לפרופסור עמוס ישראלי

$$A_{11}=\begin{pmatrix}a_{11} & a_{12}\\ a_{21} & a_{22}\end{pmatrix}\qquad A_{12}=\begin{pmatrix}a_{13} & a_{14}\\ a_{23} & a_{24}\end{pmatrix}$$

$$B_{11}=\begin{pmatrix}b_{11} & b_{12}\\ b_{21} & b_{22}\end{pmatrix}\qquad B_{21}=\begin{pmatrix}b_{31} & b_{32}\\ b_{41} & b_{42}\end{pmatrix}$$

אם ננסה כעת לחשב את האיבר הראשון בשורה הראשונה של $A_{11}B_{11}+A_{12}B_{21}$ נקבל

$$\left(a_{11}b_{11}+a_{12}b_{21}\right)+\left(a_{13}b_{31}+a_{14}b_{41}\right)$$

השווה בדיוק לאיבר הראשון בשורה הראשונה במכפלה $A\times B$. אם נמשיך כך אפשר להוכיח את נכונות הטענה כולה.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

הערות

1. כל אחת מן המטריצות A_{ij}, B_{ij} ו-C_{ij}, $i,j=1,2$ היא מסדר $\frac{n}{2}\times\frac{n}{2}$ ובכל מטריצה כזו יש $\frac{n^2}{4}$ איברים.
2. במקרה זה, פעולות החיבור והכפל הן **פעולות של מטריצות**. פעולות הכפל מחושבות על ידי קריאות רקורסיביות לאלגוריתם הכפל. הסימון $\times$ הושמט כדי לחסוך במקום. פעולות חיבור המטריצות משתמשות באלגוריתם הבא:

© כל הזכויות שמורות לפרופסור עמוס ישראלי

סכום מטריצות

תהיינה A ו-B מטריצות ריבועיות מסדר n (בגודל $n\times n$).

סכום המטריצות $D=A+B$ הוא מטריצה D המקיימת

$$d_{ij}=a_{ij}+b_{ij}\quad i,j=1,2,\ldots,n$$

נתבונן בבעיית חישוב סכום מטריצות:

קלט: שתי מטריצות מסדר n.

פלט: סכום המטריצות.

טענה: סיבוכיות בעיית חישוב סכום מטריצות היא $\Theta\left(n^2\right)$.

הוכחה: כל אלגוריתם חייב לייצר את מטריצת הסכום שגודלה הוא n^2.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

<u>אלגוריתם הפרד ופתור (רקורסיבי)</u>

<u>לחישוב מכפלת מטריצות בגודל</u> $n \times n$

1. "רפד" את שתי המטריצות באפסים עד שמימדן יגיע לחזקה של 2 הגדולה מ-n והקרובה אליו ביותר.
2. **אם** $n=1$ הכפל A ב-B (כפל שלמים)
3. **אחרת**

3.1 חלק כל מטריצה לארבע.

3.2 חשב את שמונה המכפלות הנדרשות (בעזרת קריאות רקורסיביות).

3.3 בצע ארבע פעולות חיבור.

3.4 מזג התוצאות למטריצה הנדרשת.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

<u>אנליזה של זמן החישוב</u>

נסמן ב $T(n)$ את מספר הפעולות האריתמטיות, חיבור וכפל, הנדרשות להכפלת שתי מטריצות שגודלן $n \times n$.

מן האלגוריתם הרקורסיבי שהצגנו נובעות **משוואות הרקורסיה** הבאות:

$$T(n) = 8T\left(\frac{n}{2}\right) + 4\left(\frac{n}{2}\right)^2 =$$

$$8T\left(\frac{n}{2}\right) + n^2$$

ידוע גם כי $T(1)=1$ כי מטריצה מסדר 1×1 היא סקלר ומכפלת שתי מטריצות כאלה דורשת פעולת כפל יחידה.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

<u>הצבה במשוואות הרקורסיה</u>

נשתמש במשוואות הרקורסיה כדי לחשב:

$$T(2) = 8T(1) + 4 \cdot 1^2 = 12$$

$$T(4) = 8T(2) + 4^2 = 96 + 16 = 112$$

בדרך זו נוכל לחשב את מספר הצעדים הדרושים עבור זוג מטריצות מסדר n, לכל $n > 0$ אך לא נוכל לחשב את מספר הצעדים הזה **כפונקציה של** n.

כדי לחשב את סיבוכיות האלגוריתם, נשתמש במשפט האב ונקבל:

$$c=1,\ d=1,\ a=8,\ b=2,\ f(x) = x^2$$

במקרה זה, $f(2) = 4 < a$ ומקבלים

$$T(n) = \Theta\left(n^{\log_2 8}\right) = \Theta(n^3)$$

© כל הזכויות שמורות לפרופסור עמוס ישראלי

<u>דיון</u>

ברור כי $2n^3 + n^2 = \Theta(n^3)$, כלומר סיבוכיות האלגוריתם החדש שווה לסיבוכיות האלגוריתם הישיר. מה הרווחנו?

<u>תשובה:</u>

הרווחנו דרך חדשה להסתכלות על הבעיה. כעת אנו יכולים לחפש אלגוריתם רקורסיבי **יעיל יותר** כדי לצמצם את זמן החישוב הנדרש כדי לחשב את תת המטריצות C_{ij}, $i, j = 1,2$.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

האלגוריתם של Strassen

מדען המחשבים השויצרי, Strassen, גילה שיטה חלופית להכפלת שתי מטריצות מגודל 2×2 .

בשיטת Strassen מתבצעות שבע פעולות כפל מטריצות במקום שמונה פעולות כפל מטריצות המתבצעות בשיטה המקובלת. בתמורה עולה מספר פעולות החיבור.

הפעלת שיטת Strassen מניבה אלגוריתם להכפלת מטריצות שהסיבוכיות שלו היא:

$$\Theta\left(n^{\log_2 7}\right) = \Theta\left(n^{2.81...}\right).$$

האלגוריתם של Strassen (המשך)

האינטואיציה: המקור לאיבר n^3 הן המכפלות הרקורסיביות. סיבוכיות החיבור היא n^2 בלבד. כדי להקטין את הסיבוכיות נוריד את מספר המכפלות ונעלה את מספר הסכומים.

הרעיון: כמו באלגוריתם הרקורסיבי המקורי נחשב את ארבע המטריצות C_{11}, C_{12}, C_{21} ו C_{22}.
כדי לחשב מטריצות אלה נחשב תחילה שבע מטריצות ביניים:

שבע מטריצות הביניים

כדי לחשב את המטריצות C_{11}, C_{12}, C_{21} ו C_{22}, נחשב תחילה, כשלב בינים, את שבע המטריצות הבאות:

$$M_1 = (A_{12} - A_{22}) \times (B_{21} + B_{22})$$
$$M_2 = (A_{11} + A_{22}) \times (B_{11} + B_{22})$$
$$M_3 = (A_{11} - A_{21}) \times (B_{11} + B_{22})$$
$$M_4 = (A_{11} + A_{12}) \times B_{22}$$
$$M_5 = A_{11} \times (B_{12} - B_{22})$$
$$M_6 = A_{22} \times (B_{21} - B_{11})$$
$$M_7 = (A_{21} + A_{22}) \times B_{11}$$

חוק הפילוג עבור מטריצות

לפני שנמשיך בהצגת האלגוריתם יש לציין כי צריך (וגם קל) לוודא כי **חוק הפילוג (דיסטריבוטיביות)** של הכפל מעל החיבור מתקיים גם עבור כפל וחיבור מטריצות כלומר: לכל שלוש מטריצות מסדר $n \times n$ A, B ו C מתקיים:

$$A \times (B + C) = A \times B + A \times C$$
$$(B + C) \times A = B \times A + C \times A$$

מדוע יש צורך בשני חוקי פילוג?

תרגיל:

שכנעו את עצמכם כי חוק הפילוג עבור כפל וחיבור מטריצות אכן מתקיים.

<u>האלגוריתם של Strassen (המשך):</u>

לאחר חישוב שבע מטריצות הביניים נשתמש בהן כדי לחשב את ארבעת המטריצות הסופיות כדלהלן:

$$C_{11} = M_1 + M_2 - M_4 + M_6$$

$$C_{12} = M_4 + M_5$$

$$C_{21} = M_6 + M_7$$

$$C_{22} = M_2 - M_3 + M_5 - M_7$$

נבדוק את הנוסחה הראשונה:

$$\begin{aligned}
&A_{12}B_{21} - A_{22}B_{21} + A_{12}B_{22} - A_{22}B_{22} && (= M_1)\\
&+ A_{11}B_{11} + A_{11}B_{22} + A_{22}B_{11} + A_{22}B_{22} && (= M_2)\\
&- A_{11}B_{22} - A_{12}B_{22} && (= M_4)\\
&+ A_{22}B_{21} - A_{22}B_{11} && (= M_6)\\
&= C_{11}
\end{aligned}$$

תרגיל: בדוק נכונות שאר הנוסחאות.

<u>סיבוכיות אלגוריתם Strassen:</u>

סיבוכיות הזמן נתונה על ידי המשוואה הרקורסיבית הבאה:

$$T(n) = 7T\left(\frac{n}{2}\right) + 18\left(\frac{n}{2}\right)^2$$

$$T(1) = c$$

נשתמש במשפט האב ונקבל:

$a = 7$, $b = 2$, $f(n) = n^2$.

פרמטרים אלה מקיימים:

$$f(2) = 4 < a$$

ולכן

$$T(n) = \Theta\left(n^{\log_2 7}\right) = O(n^{2,81...})$$

<u>סיכום</u>

בהרצאה זו למדנו את **משפט האב (Master Theorem)**, המאפשר לנו לקבוע סיבוכיות של מגוון רחב של אלגוריתמים רקורסיביים.

פתחנו את ההרצאה בהצגה פרמטרית של מערכת משוואות רקורסיבית כך שתשקף את סיבוכיות הזמן של אלגוריתמים רקורסיביים המקיימים דרישה מסוימת.

לאחר מכן הצגנו את משפט האב המאפשר חישוב הסיבוכיות של אלגוריתמים כאלה.

<u>סיכום (המשך)</u>

כדי להדגים את השיטה, טיפלנו בבעית כפל המטריצות הריבועיות מסדר n.

בתחילה הצגנו את האלגוריתם הישיר שסיבוכיותו היא $\Theta\left(n^3\right)$ (יש לזכור כי גודל הקלט הוא: $\Theta\left(n^2\right)$).

המשכנו בהצגת אלגוריתם הפרד ופתור פשוט פותר את הבעיה על ידי **חלוקת מטריצות הקלט ל4**. כתוצאה, הבעיה המקורית מחולקת ל**8 תת בעיות הנפתרות באופן רקורסיבי**.

סיכום **(המשך)**

האלגוריתם הזה, לא שיפר את סיבוכיות האלגוריתם המקורי אך פתח את הדלת ל**אלגוריתם** Strassen אשר שיפר את הסיבוכיות על ידי מציאת דרך מתוחכמת ל**חישוב 8 תוצאות המכפלות הנדרשות** על ידי ביצוע בפועל של **7 מכפלות בלבד.**

מספר הפעמים בהן חישבנו **סכום של מטריצות עלה מ 4 ל 18**. ניתוח סיבוכיות הזמן של אלגוריתם Strassen, התקבל כתוצאה מיידית של משפט האב (Master Theorem).

סיבוכיות אלגוריתם Strassen היא $\Theta\left(n^{2.81...}\right)$.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

סיכום (המשך)

אלגוריתם Strassen מראה את הדרך לשיפורים נוספים על ידי הורדת מספר המכפלות הנדרש והעלאת מספר הסכומים.

היום משתמשים בשיטות דומות אך מסובכות יותר, מחלקים כל בעיה למספר רב יותר של חלקים, במקום 4 בלבד, ומגיעים לאלגוריתמים למכפלת מטריצות בסיבוכיות זמן $O\left(n^{2.3...}\right)$.

שימו לב: אלה אלגוריתמים בעלי **ערך תיאורטי בלבד**, שכן **קבוע הפרופורציה** עולה לערכים בלתי אפשריים.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

סיכום (המשך)

מה צריך לזכור מהרצאה זו?

1. פיתוח מערכת המשוואות הרקורסיביות.
2. משפט האב ושימושיו.
3. להבין את הוכחת המשפט.
4. הגדרת כפל מטריצות והאלגוריתם הישיר.
5. האלגוריתם הרקורסיבי הפשוט.
6. רעיון אלגוריתם Strassen (אין צורך לזכור את מטריצות הביניים ואת הפיתוח בעל פה.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

נספח: חישוב מדויק של סיבוכיות האלגוריתם הרקורסיבי הראשון מציאת נוסחה סגורה

כדי לחשב את סיבוכיות האלגוריתם, עלינו למצוא **נוסחה סגורה** לחישוב $T(n)$.

טענה: נתון

$$T(n) = 8T\left(\frac{n}{2}\right) + n^2 \;,\; T(1) = 1$$

אזי (ניחוש): לכל $n = 2^i$

$$T(n) = 2n^3 - n^2$$

© כל הזכויות שמורות לפרופסור עמוס ישראלי

<u>הוכחה</u>

עבור $n=2^i$ ההוכחה באינדוקציה על i. כלומר, יש להניח נכונות לגבי n ולהוכיח לגבי $2n$. נשתמש בנוסחה הרקורסיבית לחישוב $T(2n)$ בעזרת $T(n)$:

$$T(2n)=8T(n)+(2n)^2 =$$

בתוך נוסחה זו נציב את פתרוננו עבור $T(n)$:

$$T(2n)=8\cdot(2n^3-n^2)+4n^2$$

ונפשט כדי לקבל:

$$=16n^3-4n^2 =$$

$$=2(2n)^3-(2n)^2$$

מש"ל

© כל הזכויות שמורות לפרופסור עמוס ישראלי

הרצאה 11: אלגוריתם למציאת החציון - Order Statistic

Definition

In statistics, the k-th **order statistic** of a statistical sample is equal to its kth-smallest value.

או בעברית:

בסטטיסטיקה, האיבר ה-k בגדלו מלמטה במדגם נקרא
The k-th order statistics
בהרצאה זו נציג אלגוריתם יעיל לחישוב האיבר הזה, כפי שמוגדר בשקף הבא:

© כל הזכויות שמורות לפרופסור עמוס ישראלי

הבעיה

קלט: 1. מערך A בן n איברים **נבדלים** השייכים לתחום סדור.
2. מס' שלם k, $1 \le k \le n$.

פלט: האיבר ה - k בגדלו (מלמטה) ב- A.

דוגמא

קלט $k = 3$

A

1	7	8	2	4	6	5	3

פלט 3

© כל הזכויות שמורות לפרופסור עמוס ישראלי

אלגוריתם נאיבי

מיין את A והחזר את האיבר ה - k בוקטור הממוין.

נכונות

מיידית מהגדרת הבעיה.

סיבוכיות

$\Theta(n \log n)$

שאלה: האם קיים אלגוריתם יעיל יותר?

תשובה: להלן נציג אלגוריתם בזמן ריצה ליניארי למציאת האיבר ה - k בגודלו ב- A.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

סקירת האלגוריתם

להלן סקירת שלבי האלגוריתם:

1. בחר ציר "חכם".
2. בצע חלוקת ציר לתת מערך תחתון (שנסמנו ב- $A_<$) ולתת מערך עליון (שנסמנו ב- $A_\geq$).
תזכורת: כל איבר ב- $A_\geq$ גדול ממש מכל איבר ב- $A_<$.
3. זהה את תת המערך המכיל את האיבר המבוקש.
4. המשך את החיפוש בתת המערך המכיל את האיבר המבוקש. (קריאה רקורסיבית).

© כל הזכויות שמורות לפרופסור עמוס ישראלי

תכונות חלוקת ציר

נניח שמתקיים $|A_<| = m$ ונניח גם שאנו מחפשים את האיבר ה-k בגדלו שנסמנו ב-m_k.

לאחר ביצוע חלוקת הציר עלינו לענות על השאלה הבאה:

באיזה תת מערך נמצא m_k ?

התשובה תלויה ביחס בין m ל-k כמוכח בטענה הבאה:

© כל הזכויות שמורות לפרופסור עמוס ישראלי

למה 11.1

יהי A מערך מעל תחום סדור. נניח כי ב -A בוצעה חלוקת ציר שהניבה את החלקים $A_<$ בגודל m ו - $A_\geq$ בגודל $n-m$. לאחר חלוקת הציר מתקיים:

1. אם $k \leq m$ אזי m_k הוא האיבר ה-k בגודלו ב-$A_<$.
2. אם $k > m$ אזי m_k הוא האיבר ה- $(k-m)$ בגודלו ב - $A_\geq$.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

דוגמא

A

3	8	-2	13	11	6	9	2	14	7

$pivot = 8$, $k = 6$

$A_<$

3	-2	6	2	7

$A_\geq$

8	13	11	9	14

$m = 5$

ב $A_<$ נמצאים 5 האיברים הקטנים ביותר ב A.

האיבר ה 6 בגודלו בA, 8, הוא האיבר המינימלי ב$A_\geq$

© כל הזכויות שמורות לפרופסור עמוס ישראלי

הוכחת הלמה

ב - A יש k-1 איברים, הקטנים מ - m_k. כל שאר איברי A גדולים או שווים מ - m_k. לאחר שמתבצעת חלוקת הציר, ב - $A_<$ נמצאים m האיברים הקטנים ביותר ב-A, מסודרים בסדר שרירותי.

הוכחת סעיף 1

אם $k \leq m$ אזי ברור ש-m_k הוא בין הוא בין m האיברים הקטנים ביותר במערך A, ולכן הוא מאוחסן בתת המארך התחתון.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

<u>הוכחת סעיף 2:</u>

אם, לעומת זאת, $k > m$ אזי איברי $A_<$ הם m האיברים הקטנים ביותר במערך A, ו-m_k הוא האיבר ה $k - m$ בגודלו ב - $A_\geq$.

מש"ל

© כל הזכויות שמורות לפרופסור עמוס ישראלי

<u>תנאי הקצה</u>

אנו מפתחים אלגוריתם רקורסיבי. ידוע כי שיטות רקורסיביות אינן יעילות עבור קלט בגדלים קטנים. כדי להגדיל את היעילות, בגורם קבוע בלבד, נקבע כי אם מספר איברי מערך הקלט קטן או שווה מ-10 , נקרא לשיטה *Find* להלן:

$$Find(A,k)$$
$$A \leftarrow Bubblesort(A)$$
$$\mathbf{return}(A[k])$$

<u>אלגוריתם 11.1: השיטה *Find*</u>

© כל הזכויות שמורות לפרופסור עמוס ישראלי

<u>האלגוריתם</u>

להלן קוד האלגוריתם *Select* המחשב את האיבר ה-k בגדלו במערך הקלט A:

$Select(A,k)$
if $|A| \leq 10$ **then**
  return $Find(A,k)$
$p \leftarrow WiseChoosePivot(A)$
Partition A into $A_<$ and $A_\geq$, using p
if $k \leq m$ **return** $Select(A_<, k)$
if $k > m$ **return** $Select(A_\geq, k-m)$

<u>אלגוריתם 11.2: השיטה *Select*</u>

© כל הזכויות שמורות לפרופסור עמוס ישראלי

<u>נכונות</u>

נכונות האלגוריתם נובעת מיידית מלמה 11.1 הטענה שהוכחנו.

<u>סיבוכיות האלגוריתם – איכות איבר הציר</u>

סיבוכיות השיטה *Find* אינה תלויה בגודל הקלט n, ולכן היא לא משפיעה על סיבוכיות האלגוריתם כולו.
סיבוכיות החלוקה היא כמובן לינארית.
הגורם היחיד המשפיע על סיבוכיות אלגוריתם *Select* הוא הגודל היחסי של תת המערכים $A_<$ ו-$A_\geq$. גודל זה נקבע ישירות על ידי **<u>איכות איבר הציר</u>**.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

איכות איבר הציר (המשך)

אם לדוגמא נשתמש בשיטה $Choosepivot$ שהשתמשנו בה במיון מהיר, אזי במקרה הגרוע ביותר, אחד מתת המערכים יכיל איבר יחיד.
אם תופעה זו תחזור על עצמה (כפי שקורה בדוגמאות מסויימות), סיבוכיות האלגוריתם תגיע ל $\Theta(n^2)$.
כדי להתגבר על בעיה זו, נשקיע עבודה **לינארית** בבחירת איבר הציר וכתוצאה נקבל שסיבוכיות המקרה הגרוע ביותר באלגוריתם כולו תהיה גם היא **לינארית**.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

בחירת איבר ציר איכותי

אנו מחפשים איבר ציר שיקיים:
מספר האיברים בכל אחד מתת המערכים $A_<$ ו-$A_\geq$ יהיה לפחות αn ולכל היותר ה $(1-\alpha)n$, כאשר α הוא קבוע שאינו תלוי ב-n ואשר מקיים $0<\alpha<1$.
האלגוריתם שנציג מבטיח בחירה של ציר שמרחקו מכל אחד משני קצות מערך הקלט הוא לפחות $3/10n$.
מכאן נקבל כי גדלו של כל אחד מתת המערכים יהיה לכל היותר $7/10n$, כלומר:

$$3/10n<|A_<|<7/10n$$

$$3/10n<|A_\geq|<7/10n$$

© כל הזכויות שמורות לפרופסור עמוס ישראלי

תאור האלגוריתם לבחירת הציר

יהי A מערך עם n איברים. **החציון** של A הוא האיבר ה$\lfloor n/2\rfloor+1$ בגודלו ב-A.
האלגוריתם פועל כך:

1. נחלק את המערך A ל-$\lceil n/5\rceil$ קבוצות בנות 5 איברים כל אחת. בכל חמישיה, נמצא את האיבר השלישי בגודלו, כלומר את החציון של כל חמישייה כזו. (זמן החישוב הכולל בסעיף זה הוא $\Theta(n)$).

© כל הזכויות שמורות לפרופסור עמוס ישראלי

תאור האלגוריתם לבחירת הציר (המשך)

2. תהי M קבוצת החציונים שמצאנו ויהי m החציון של M. m הוא הציר שנבחר.
3. כדי לחשב את m, נבצע קריאה רקורסיבית: $Select(M,\lfloor|M|/2\rfloor+1)$ (זמן החישוב $T(n/5)$).

הקוד מופיע בשקף הבא:

© כל הזכויות שמורות לפרופסור עמוס ישראלי

השיטה WiseChoosePivot

$WiseChoosePivot(A)$

for $i=1$ **to** $\lceil n/5 \rceil$

/* בחר את החמישיה ה-i */

  for $j=1$ **to** 5 **do**

    $B[j] \leftarrow A[(i-1)*5+j]$

  endfor

  $B \leftarrow sort(B)$

  /* בחר את חציון החמישיה ה-i */

  $M[i] \leftarrow B[3]$

endfor

Return $Select(M, \lfloor |M|/2 \rfloor + 1)$

אלגוריתם 11.2: *WiseChoosePivot*

© כל הזכויות שמורות לפרופסור עמוס ישראלי

דוגמא - הוקטור A ממוין בחמישיות

35	46	72	77	90
12	17	19	21	23
50	69	70	71	125
10	18	22	128	256
-4	25	40	41	42
-120	-60	-40	200	400
10	11	560	580	600
7	13	199	201	326
38	39	61	147	269
-100	-66	52	421	856
500	601	702	803	904
62	63	64	65	66

© כל הזכויות שמורות לפרופסור עמוס ישראלי

דוגמא (המשך)

הוקטור M

72	19	70	22	40	64	560	199	61	52	702

החציון של M הוא $m=64$.

שימו לב:

1. מספר האיברים ב M הוא כמספר החמישיות כלומר $\left\lceil \frac{n}{5} \right\rceil$.
2. מספר איברי M הקטנים או שווים ל m הוא $\left\lceil \frac{n}{10} \right\rceil$.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

דוגמא (המשך)

3. כל האיברים הנמצאים משמאל לאיבר הקטן מm גם הם קטנים מ-m ולכן, ב-A יש $\left\lceil \frac{3n}{10} \right\rceil$ איברים הקטנים מ m.
4. מספר איברי M הגדולים או שווים מ m הוא $\left\lceil \frac{n}{10} \right\rceil$.
5. כל האיברים הנמצאים מימין לאיבר הגדול מm גם הם גדולים מ m ולכן, בA יש $\left\lceil \frac{3n}{10} \right\rceil$ איברים הגדולים מ m.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

חישוב α

טענה: יהי m החציון של הוקטור M.

מספר איברי A הקטנים ממש מ-m הוא לפחות $n\left\lceil \frac{3}{10} \right\rceil$.

מספר איברי הוקטור A הגדולים ממש מ-m הוא לפחות $n\left\lceil \frac{3}{10} \right\rceil$.

שימו לב

מטענה זו נובע מיידית

$$\left|A_{<p}\right| \le \left\lfloor \frac{7}{10} \right\rfloor n \qquad \left|A_{\ge p}\right| \le \left\lfloor \frac{7}{10} \right\rfloor n$$

כלומר, גודל הקלט בקריאה הרקורסיבית, הוא 7/10 מגודל הקלט המקורי.

הוכחה

מספר האיברים ב M הוא $\lceil n/5 \rceil$. כל אחד מאיברי M, m_i, הוא **חציון** של 5 מאיברי A. מכאן, m_i גדול או שווה מ3 מאיברי A (שני האיברים הקטנים בחמישייה של m_i וm_i עצמו).

מאחר ש m הוא החציון של M, נובע כי m גדול מ $n/10$ מאיברי M.

כאמור, כל איבר מ M, גדול או שווה מ3 מאיברי A, מכאן נובע כי m גדול או שווה מ$3n/10$ מאיברי A.

באופן סימטרי אפשר להוכיח כי m קטן או שווה מ $3n/10$ מאיברי A.

מש"ל

דוגמא

נניח כי מחפשים את האיבר ה - 70 במערך A בן 200 איברים. כדי לבצע זאת מפעילים את הקריאה $Select(A,70)$. בשקף הבא, אנו מפורטות (חלק מן) הקריאות הרקורסיביות שתתבצענה.

אנו מניחים כי $Find$ היא פרוצדורה פשוטה למציאת האיבר הi בגדלו במערך שבו לכל היותר 10 איברים.

בסוגרים רשומות הנחות לגבי גודל המערכים המתקבלים בחלוקת ציר.

פרוט הקריאות

$Select(A,70)$

$ChoosePivot$

compute M_1, $(|M_1| = 40)$

$Select(M_1,21)$

* $Choosepivot$

* compute M_2, $(|M_2| = 8)$

* $m_2 \leftarrow Find$ median of M_2

* $M_{11} \leftarrow M_{1,<m_2}$, $(|M_{11}| = 12)$

* $M_{12} \leftarrow M_{1,\ge m_2}$, $(|M_{12}| = 28)$

* $Select(M_{12},9)$

$m_1 \leftarrow Find^*$ 9th elt of M_{12}

$A_1 \leftarrow A_{<m_1}$, $(|A_1| = 140)$

$A_2 \leftarrow A_{\ge m_1}$, $(|A_2| = 60)$

תיאור הקריאות בדוגמא

כדי לבצע זאת מפעילים את הקריאה

$Select(A,70)$

במהלך קריאה זו מתבצעות הפעולות הבאות:

1. בוחרים מערך M_1 בן 40 איברים.
2. מפעילים באופן רקורסיבי את הקריאה $m_1 \leftarrow Select(M_1,21)$

לאחר מציאת m_1 כמפורט בשקף הבא מתבצע:

3. מחלקים את A בחלוקת ציר לפי m_1.

תיאור הפעולות במהלך $Select(M_1,21)$

2.1 בוחרים מערך M_2 בן 8 איברים.

2.2 מפעילים שגרה פשוטה למציאת m_2, החציון (האיבר ה5) של M_2.

2.3 מחלקים את M_1 לפי m_2 כציר.

יהיו M_{11} ו M_{12} החלק התחתון והחלק העליון בחלוקת M_1 לפי m_2. לפי המשפט שהוכחנו, מובטח לנו כי $12 \le |M_{11}| \le 28$ ו $12 \le |M_{12}| \le 28$. נניח כי $|M_{11}| = 12$ ו $|M_{12}| = 28$.

2.4 מפעילים את $Select(M_{12},9)$.

לאחר קריאה רקורסיבית נוספת מוצאים את m_1, האיבר ה - 9 ב - M_{12} והחציון של M_1.

דוגמא - סיכום

לפי המשפט שהוכחנו מובטח לנו כי לאחר חלוקת מערך הקלט A לפי m_1, מובטח לנו כי מספר האיברים בכל אחד מחלקי A יהיה גדול מ - $\frac{3}{10}200 = 60$.

במקרה הגרוע ביותר, נקבל חלוקה לחלק תחתון עם 140 איברים ולחלק עליון עם 60 איברים. במקרה זה נקרא כעת ל $Select(A_{<p},70)$.

סיבוכיות האלגוריתם

משפט

סיבוכיות האלגוריתם למציאת האיבר הk היא לינארית.

הוכחה

נסמן ב $T(n)$ את זמן הריצה של האלגוריתם על מערך בגודל n.

לאלגוריתם שני קטעים בעלי סיבוכיות לינארית:

1. בחירת המערך M.
2. ביצוע חלוקת הציר.

נסמן ב cn את מספר הצעדים הכולל, הדרוש לביצוע שני חלקים אלה.

המשך ההוכחה

בנוסף לכך, במהלך החישוב מתבצעות שתי קריאות רקורסיביות לשגרה $Select$:

1. למציאת m החציון של M.

הסיבוכיות היא, $T(|M|)=T\left(\frac{n}{5}\right)$.

2. להמשך הפתרון בתת הבעיה שנמצאה: במקרה הגרוע ביותר, הסיבוכיות היא,

$T\left(\max|A_{<p}|,|A_{\ge p}|\right)=T\left(\frac{7n}{10}\right)$.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

המשוואות המתקבלות

המשוואות הרקורסיביות המתקבלות הן:

$$T(10)=const$$

$$T(n)=\underbrace{T\left(\frac{n}{5}\right)}_{\text{בחירת ציר}}+\underbrace{T\left(\frac{7n}{10}\right)}_{\text{קריאה רקורסיבית}}+\underbrace{cn}_{\text{חלוקה}}$$

עלינו להוכיח: $T(n)=\Theta(n)$.

ברור כי $T(n)>n$. לכן, מספיק להראות כי $T(n)<dn$ עבור קבוע כלשהו d, $d>0$.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

ניתוח המשוואות

משפט: מערכת המשוואות שהצבנו מקיימת $T(n)<10cn$.

הוכחה: באינדוקציה על n.

בסיס: $T(10)=const$ וכן המשפט מתקיים.

צעד: נניח כי לכל $m<n$ הטענה מתקיימת, כלומר $T(m)<10cm$.

בפרט מתקיים:

$$T(|M|)=T\left(\frac{n}{5}\right)\le\frac{10cn}{5}=2cn$$

$$t\left(\max|A_{<p}|,|A_{\ge p}|\right)\le T\left(\frac{7n}{10}\right)\le\frac{7\cdot 10cn}{10}$$

$$t\left(\max|A_{<p}|,|A_{\ge p}|\right)\le 7cn$$

© כל הזכויות שמורות לפרופסור עמוס ישראלי

ניתוח המשוואות

נציב תוצאות אלה במשוואה ונקבל:

$$T(n)=T\left(\frac{n}{5}\right)+T\left(\frac{7n}{10}\right)+cn\le$$

$$\le 2cn+7cn+cn=10cn$$

מש"ל

© כל הזכויות שמורות לפרופסור עמוס ישראלי

<u>חשוב לציין</u>

בחירת הציר בדרך המוצעת מבטיחה כי סיבוכיות האלגוריתם היא לינארית עבור כל קלט וקלט.

תוצאה זו לא תתקבל "חינם".

התשלום הוא הגדלת הסיבוכיות הממוצעת בגורם קבוע.

במלים אחרות: הסיבוכיות הממוצעת בשני האלגוריתמים היא **לינארית** אך באלגוריתם השני, המקדם גדול יותר.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

© כל הזכויות שמורות לפרופסור עמוס ישראלי

הרצאה 11: אלגוריתמים חמדניים (Greedy Algorithms)

החמדנות היא פרדיגמה חשובה בתורת האלגוריתמים. באופן כללי ביותר, השיטה החמדנית מציעה כי בכל שלב, כאשר יש לבחור בין כמה צעדים חלופיים, בדרך לפתרון בעיה כלשהי, נבחר בצעד המביא **רווח מיידי מכסימלי**.

בדרך זו נבנים אלגוריתמים פשוטים יחסית, אך לא תמיד אלגוריתמים אלה מניבים פתרונים אופטימליים. הקושי העיקרי בשיטה הוא ביצוע הבדיקה האם האלגוריתם שפיתחנו פותר את הבעיה.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

אלגוריתם חמדני לבחירת קידוד בינארי

בהרצאה זו נעסוק בדרך האופטימלית לאחסון מידע, בקידוד בינארי. לאחר הצגת הבעיה, נפתח את האלגוריתם (החמדני) של Huffman, לבחירת קידוד אופטימלי ונוכיח את נכונותו.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

קידוד בינארי של תווים

נניח כי אנו רוצים לאחסן סדרת תווים (להלן נקרא לסדרה כזו בשם **קובץ**) מקבוצת תוים נתונה C. בקידוד תווים בינארי, או בקיצור **קידוד בינארי**, כל תו מקודד (מוצפן) כסדרת סיביות:

נאמר ש D הוא **קידוד בינארי של C**, אם D היא התאמה של סדרת סיביות לכל תו $c \in C$.

סדרה הסיביות המקודדת את התו c נקראת **הקידוד של c** ומסומנת על ידי על ידי $D(c)$.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

קידוד בינארי של תווים

לכל מחרוזת תווים S, **הקידוד של S על ידי D**, $D(S)$, מתקבל על ידי שרשור הקידודים של התווים ב-S.

עלות הקידוד של S, שווה למספר הסיביות ב-$D(S)$ ומסומן על ידי $|D(S)|$.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

<u>סוגי קידוד</u>

<u>בקידוד בינארי שווה אורך</u>

(**Fixed-length Codeword**), מספר הסיביות בקידוד כל אחד מן התווים שווה.

<u>בקידוד באורך משתנה</u>

(**Variable-length Codeword**), אורך הקידוד של תווים שונים יכול להיות שונה.

<u>השכיחות של תו</u> u בקובץ S , היא היחס (באחוזים) בין מספר המופעים של u ב-S לבין מספר התווים הכולל ב-S.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

<u>דוגמה לדחיסת מידע (לקוחה מן הספר [CLR])</u>

נניח כי אנו רוצים לאחסן קובץ ובו 100 מלים מעל א״ב ובו 6 תווים. טבלה 11.1 מציגה את שכיחויות ששת התוים ושני קידודים בינאריים של התווים האלה.

תו	f	e	D	c	B	a
שכיחות	5	9	16	12	13	45
שוה אורך	101	100	011	010	001	000
אורך משתנה	1100	1101	111	100	101	0

<u>טבלה 11.1: טבלת שכיחויות וקידודים</u>

© כל הזכויות שמורות לפרופסור עמוס ישראלי

<u>דוגמה לדחיסת מידע (המשך)</u>

בשורת השלישית בטבלה מופיע **קידוד שווה אורך.**

בשורה שאחריה מופיע **קידוד באורך משתנה**

חישוב קצר מעלה כי אם בקובץ יש 100 מלים, אז הקידוד הראשון דורש 300 סיביות. הקידוד השני, לעומת זאת, דורש 224 סיביות.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

<u>דוגמה לדחיסת מידע (המשך)</u>

דוגמא זו מבהירה מייד את חשיבות הבחירה בקידוד יעיל לדחיסת מידע. כאשר כל תו מקודד במספר סיביות שווה, אין התחשבות בשכיחות המופעים של כל תו בטקסט נתון. אם נתחשב בשכיחות זו, בטקסט נתון כלשהו, נוכל לחסוך במספר הסיביות הכולל הנדרש לקידוד הטקסט הנתון.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

קידוד תחיליות

קידוד תחיליות (**Prefix Code**) הוא קידוד בינארי בו לכל שני תווים c_1 ו-c_2, **הקידוד של c_1 אינו תחילית של הקידוד של c_2**.

כלומר $D(c_1)$ אינו תחילית (prefix) של $D(c_2)$:

הערת צד

כל קידוד שווה אורך הוא קידוד תחיליות.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

קידוד תחיליות של מחרוזת

כדי לקודד מחרוזת, מאתחלים את המחרוזת המקודדת כמילה ריקה, ומשרשרים את קידודי התוים במחרוזת המקודדת בזה אחרי זה.

לדוגמה:

נסמן את פעולת שרשור המחרוזות על ידי •. כדי לקודד את המחרוזת *adf*, בקידוד שווה האורך המוצג בטבלה 11.1 משרשרים למילה הריקה את הקידוד של *a*, *000*, ומקבלים $D(a)=000$.

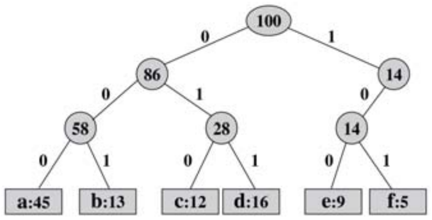
בהמשך נשרש את הקידוד של d ושל f ונקבל $D(adf)=000\bullet 011\bullet 101$.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

הצגת קידוד תחיליות כעץ בינארי

אפשר להציג כל קידוד תחיליות כעץ בינארי בו כל תו מקודד על ידי מסלול בעץ כמודגם באיור 11.2:

תו	*f*	*e*	*D*	*c*	*B*	*a*
שכיחות	5	9	16	12	13	45
שווה אורך	101	100	011	010	001	000

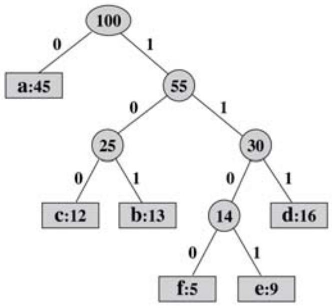


איור 11.2(a): עץ בינארי המייצג קידוד שווה אורך

© כל הזכויות שמורות לפרופסור עמוס ישראלי

הצגת קידוד תחיליות כעץ בינארי

תו	*f*	*e*	*D*	*c*	*B*	*a*
שכיחות	5	9	16	12	13	45
אורך משתנה	1100	1101	111	100	101	0



איור 11.2(b): עץ בינארי המייצג קידוד באורך משתנה

© כל הזכויות שמורות לפרופסור עמוס ישראלי

<u>פיענוח מחרוזת</u>

כדי לפענח מחרוזת סיביות, מתחילים בשורש עץ הקידוד, ומתקדמים שמאלה או ימינה, בהתאם לסיביות המחרוזת. כאשר מגיעים לעלה, רושמים את התו המופיע בו, מורידים ממחרוזת הסיביות את כל הסיביות שהשתמשנו בהן ושבים אל שורש העץ.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

<u>עלות של קידוד</u>

תהי S מחרוזת כלשהי מעל C עם מערך שכיחויות f. נניח ש- $D=D(C)$ הוא קידוד של C. נגדיר את **העלות** של S, ביחס לקידוד D, כמספר הסיביות ב-$D(S)$.

קל לראות כי

$$|D(S)|=\sum_{c\in C} f[c]D(c)$$

© כל הזכויות שמורות לפרופסור עמוס ישראלי

<u>עלות של קידוד (המשך)</u>

יהי T עץ בינארי המייצג את הקידוד התחילי D. מספר הסיביות בקידוד של תו $c\in C$, שווה לאורך המסלול מן השורש של T, אל העלה של c. אם נסמן את אורך המסלול משורש העץ T אל העלה c, ב-$l(c)$, ואת עלות קידוד S על ידי העץ T ב-$B(T)$, נקבל:

$$|B(T)|=\sum_{c\in C} f(c)l(c)=|D(S)|$$

© כל הזכויות שמורות לפרופסור עמוס ישראלי

<u>קידוד אופטימלי</u>

קידוד תחילי D הוא **אופטימלי ביחס ל-** S, אם לא קיים קידוד תחילי D', $D'\neq D$, כך ש $|D(S)|>|D'(S)|$.

בהמשך ההרצאה נציג את אלגוריתם האפמן (Huffman) לחישוב קידוד תחילי אופטימלי.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

למה 11.3

עץ המייצג קידוד תחיליות אופטימלי הוא עץ מלא.

תזכורת

עץ בינארי מלא, הוא עץ בו לכל צומת שאינו עלה יש **שני בנים**.

הוכחה

נניח ש-T הוא עץ הקידוד אינו מלא.

בעץ T קיים צומת פנימי u, שיש לו בן יחיד v (יתכן שיש כמה צמתים כאלה).

נתבונן בעץ T' המתקבל מ-T על ידי מחיקת הצומת v. אם v הוא עלה, נהפוך את u לעלה, והוא ייצג את התו

ש-v מיצג ב-T. אם לא, נחבר את כל בניו של v, לצומת u. מאחר של-v יש לכל היותר 2 בנים, T' הוא עץ קידוד שעלותו קטנה ממש מעלות T. בסתירה להנחתנו כי T הוא עץ קידוד אופטימלי.

אלגוריתם הפמן

אלגוריתם הפמן מקבל כקלט מערך זוגות D, כאשר כל זוג ב-D, מכיל תו ושכיחות התו במחרוזת נתונה S.

האלגוריתם מחשב עץ קידוד תחיליות אופטימלי עבור המחרוזת S.

נסמן ב-n את מספר הזוגות במערך הקלט D.

בתחילת הביצוע, האלגוריתם יוצר מכל זוג, צומת של עץ קידוד, ומכניס את העלים שנוצרו לתור עדיפויות Q.

צמתים אלה יהיו העלים של עץ הפלט.

אלגוריתם הפמן – תיאור השלבים

לאחר מכן האלגוריתם מבצע $n-1$ שלבים: בכל שלב האלגוריתם מוציא מ-Q את שני הצמתים ששכיחותם מינימלית, יוצר צומת חדש אשר שכיחותו שווה לסכום השכיחויות של שני הצמתים שהוצאו מ-Q, "תולה" את שני הצמתים שהוצאו מ-Q על הצומת החדש, ומכניס את הצומת החדש חזרה לתור Q. בסיום האלגוריתם נותר ב-Q צומת יחיד והוא שורש עץ הקידוד שהאלגוריתם יוצר.

אלגוריתם הפמן – הקוד

```
Huffman(D)
n ← |D|
Q ⇐ D
for i ← 1 to n − 1
   z ← new Node()
   x ← Q.extractMin()
   z.left ← x
   y ← Q.extractMin()
   z.right ← y
   f[z] ← f[x] + f[y]
   Q.insert(z)
end_for
return Q.extractMin()
```

אלגוריתם 11.9: *Huffman*

© כל הזכויות שמורות לפרופסור עמוס ישראלי

הוכחת נכונות

כעת עלינו להוכיח שהקידוד המיוצג על ידי העץ שהאלגוריתם מחשב הוא אופטימלי, או במלים אחרות: לא קיים קידוד תחיליות D' המקודד את התווים הנתונים בשכיחויות הנתונות, ואשר מקיים $|D'(S)|<|D(S)|$.

טענה זו נובעת משתי הלמות הבאות:

© כל הזכויות שמורות לפרופסור עמוס ישראלי

למה 11.10

נניח ש-S היא מחרוזת מעל קבוצת תווים C. לכל תו $c \in C$, נסמן ב-$f[c]$ את שכיחות התו c במחרוזת S.

נניח ש- $x, y \in C$ הם שני התווים בעלי שכיחות מינימלית ב-S. בתנאים אלה קיים ל-C קידוד תחיליות אופטימלי בהם התווים x ו-y מקודדים על ידי שתי מלים שאורכן שווה הנבדלות אך ורק בסיבית האחרונה.

שימו לב

בעץ הקידוד האופטימלי הזה, התווים x ו-y הם ״אחים״.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

הוכחה

יהי T עץ המייצג קידוד אופטימלי כלשהו עבור S. לפי למה 11.8, T הוא עץ מלא, לכן בעץ T יש שני עלים ״אחים״, a ו-b, שמרחקם מן השורש מרבי. העלים a ו-b מקיימים:

$$l_T(a) = l_T(b) \ge l_T(x) \ge l_T(y).$$

קל לראות כי אם נחליף את קידוד התווים a ו- b עם הקידוד של x ו-y, עלות הקידוד לא תגדל וייתכן אף שתקטן.

מש״ל

© כל הזכויות שמורות לפרופסור עמוס ישראלי

למה 11.11

יהי T עץ בינארי מלא המייצג קידוד תחיליות אופטימלי מעל אלף בית C, עם מערך שכיחויות $f[c]$. נניח ש-x ו-y הם שני תוים המופיעים כאחים בעץ T, ונניח כי האב המשותף שלהם הוא הצומת z.

נתבונן בעץ הקידוד T' המתקבל על ידי מחיקת העלים x ו-y, והפיכת z לתו ששכיחותו היא $f[z]=f[x]+f[y]$.

העץ T' הוא עץ קידוד אופטימלי לאלף בית $C'=C-\{x,y\}+\{z\}$.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

הוכחה

תחילה נראה כי אפשר לבטא את עלות העץ T, $B(T)$, באמצעות עלות T', $B(T')$. לכל $c\in C-\{x,y\}$,

$d_T(c)=d_{T'}(c)$.

$d_T(x)=d_T(y)=d_{T'}(z)+1$.

מכאן נובע:

$$f[x]d_T(x)+f[y]d_T(y)=$$
$$=(f[x]+f[y])(d_{T'}(z)+1)=$$
$$=f[z]d_{T'}(z)+(f[x]+f(y))$$

ומכאן

$$B(T)=B(T')+f[x]+f[y]$$

© כל הזכויות שמורות לפרופסור עמוס ישראלי