<u>הרצאה 1: מושג הגרף - אלגוריתמים על גרפים - מסלולים ומרחקים בגרפים</u>

בהרצאה זו, נציג את מושג הגרף, נדון באלגוריתמים המקבלים גרפים כקלט ונלמד אלגוריתם אחד מסוג זה.
ההרצאה תכלול את הסעיפים הבאים :

1. הקדמה והגדרות פורמליות.
2. ייצוג גרפים במחשב.
3. הגדרת מסלולים ומרחקים.
4. הצגת בעיית המרחקים.
5. הדגמת אלגוריתם BFS.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

<u>הקדמה לנושא הגרפים</u>

הגרף הוא אחד המבנים הכי שימושיים במדעי המחשב. שימושיו הם רבים מספור.
באופן בלתי פורמלי: גרף הוא אוסף נקודות הקרויות **צמתים** (או **קודקודים**) ובאנגלית **nodes** או **vertices**. בין כל שני צמתים תיתכן **קשת** (או **צלע**) ובאנגלית **edge** או **arc**.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

<u>דוגמאות</u>

1. **רשת תקשורת -** כל מחשב הוא **צומת** כל קו תקשורת הוא **קשת**.
2. **מעגל אלקטרוני -** כל רכיב הוא **צומת**, כל קו (Bus) הוא **קשת**.
3. **מפה גיאוגרפית -** כל עיר היא **צומת** כל כביש הוא **קשת**.
4. **טבלה אירגונית -** כל תפקיד הוא **צומת** כל כפיפות היא **קשת**.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

<u>הגדרות פורמליות</u>

<u>הגדרה 1.1</u>

1. **גרף** הוא **זוג** $G=(V,E)$.
2. <u>**הצומת**</u> הוא יחידת הבסיס של כל גרף.
3. קבוצת כל הצמתים מסומנת ב- V. מספר הצמתים סופי ואפשר לסמנם ב-$\{1,2,...,n\}$ או ב-$\{v_1,...,v_n\}$.
4. <u>**קשת**</u> היא זוג צמתים. אם $v_1,v_2\in V$ נסמן $e=(v_1,v_2)$ ונאמר שהקשת e **מחברת את** v_1 **עם** v_2. שני צמתים שביניהם יש קשת נקראים **סמוכים** **(Adjacent)**.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

<u>הגדרה 1.1 (המשך)</u>

5. קשת (v_1,v_2) יכולה להיות **מכוונת** מv_1 אל v_2 או **לא מכוונת**.
6. קבוצת כל **הקשתות** מסומנת ב- E.
7. גרף שכל קשתותיו **מכוונות** (**לא מכוונות**) נקרא **גרף מכוון** (**לא מכוון**).

© כל הזכויות שמורות לפרופסור עמוס ישראלי

<u>הגדרה 1.2</u>

להלן נניח כי G הוא גרף **לא מכוון**.

1. מספר הקשתות הנוגע בצומת v נקרא ה**דרגה** של v ומסומן ב $\deg(v)$.
2. **לולאה עצמית** היא קשת מצומת אל עצמו $e=(v,v)$.
3. שתי קשתות e_1 ו e_2 קרויות **מקבילות** אם $e_1=(v_1,v_2)$ וגם $e_2=(v_1,v_2)$.

שימו לב: e_1 **ו** e_2 הן קשתות **שונות**.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

<u>הגדרה 1.3</u>

1. גרף לא מכוון שאין בו קשתות מקבילות ולולאות עצמיות נקרא **גרף פשוט**.

<u>שימו לב:</u> כל הגרפים בקורס הם **גרפים פשוטים** (אלא אם נאמר אחרת).

2. גרף פשוט $G=(V,E)$ הוא **מלא** **(Complete)** אם לכל שני צמתים $a,b\in V$ הקשת $(a,b)\in E$.

<u>אבחנה 1.4</u>

מספר הקשתות - מספר הקשתות המכסימלי בגרף **לא מכוון ופשוט** הוא

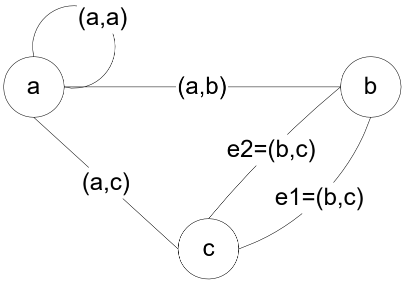
$$\frac{|V|\cdot(|V|-1)}{2}$$

© כל הזכויות שמורות לפרופסור עמוס ישראלי

<u>דוגמא לגרף</u>

$$V=\{a,b,c\}$$

$$E=\left\{\begin{matrix}(a,b),(a,c),(a,b),\\ e_1=(b,c),e_2=(b,c)\end{matrix}\right\}$$



© כל הזכויות שמורות לפרופסור עמוס ישראלי

ייצוג גרפים במחשב

יהי $G(V,E)$ גרף. כדי לייצג את G במחשב עלינו לאחסן מיידע אודות קבוצת הצמתים V וקבוצת הקשתות E. את קבוצת הצמתים נייצג בעזרת מערך בגודל $\Theta(|V|)$ בו נשמור ערכים שונים המתייחסים לכל צומת. דרך אחרת היא לייצג כל צומת כעצם (אובייקט) ולשמור מערך של עצמים אלה.

ישנן כמה שיטות לייצוג קשתות הגרף. אנו נתאר כעת את **שיטת רשימות השכנים.** בשלב מאוחר יותר נציג גם את **שיטת מטריצת הסמיכויות**.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

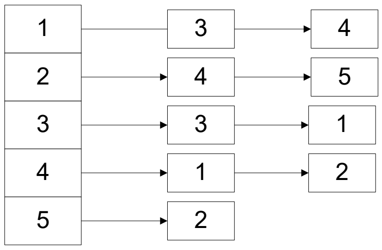
ייצוג גרף על ידי רשימת שכנויות

רשימת שכנויות (Adjacency Lists) היא דרך לייצוג גרפים בזכרון המחשב. בשיטה זו, צמתי הגרף מיוצגים על ידי מערך עצמים בגודל $|V|=n$. לכל (עצם המיצג) צומת יש **רשימת שכנים** שהיא מקושרת של עצמים שכל אחד מהם מייצג את אחד משכני הצומת.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

דוגמא

בתרשים המופיע להלן, מיוצג הגרף הלא מכוון אשר מופיע בדוגמא.



אנו רואים כי צומת 1 מחובר לצמתים 3 ו4, צומת מס׳ 2 מחובר לצמתים 4 ו5, צומת מס 3 מחובר לצומת מס׳ 1 ולעצמו וכן הלאה.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

דוגמא

יש להציג אלגוריתם המקבל גרף כקלט ומדפיס את כל צמתי הגרף ואת כל הקשתות שלו, כאשר כל קשת מודפסת פעם אחת בלבד.

גרף הקלט מיוצג על ידי רשימת סמיכויות.

תאור האלגוריתם

עבור על מערך הצמתים, לכל צומת במערך, $v \in V$, בצע:

1. הדפס את שם הצומת, v.
2. לכל קשת הנוגעת בv, (u,v)

אם $u \ge v$ הדפס את הקשת (u,v)

.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

איך האלגוריתם מתבצע?

אנו מניחים כי גרף הקלט G מיוצג על ידי רשימת סמיכויות.

האלגוריתם מקבל מצביע אל **מערך הצמתים** ועובר על צמתי הגרף, על ידי מעבר על מערך הצמתים.

לכל צומת v, עוברים על רשימת הצמתים השכנים של v ולכל צומת u, שכן של v, בודקים האם הקשת (u,v) עוד לא הודפסה, על ידי בדיקה האם $u \geq v$. אם כן, מדפיסים אותה

נכונות האלגוריתם: ברור כי האלגוריתם מסתיים וכי כל צומת וכל קשת מודפסים בדיוק פעם אחת.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

סיבוכיות האלגוריתם

תזכורת

סיבוכיות של אלגוריתם Al, היא פונקציה מתמטית המתאימה לכל מספר טבעי n, את זמן הריצה המירבי של Al על קלט בגודל n.

כדי לדון בסיבוכיות האלגוריתם יש לדון בגודל הקלט. כאשר הקלט הוא **גרף** גודל הקלט הוא גודל קבוצת הצמתים $|V| = n$ ועוד גודל קבוצת הקשתות $|E| = m$.

אנו נבטא את סיבוכיות האלגוריתמים שנפתח כפונקציות של $|V|$ ושל $|E|$.

נשים לב כי בדרך כלל מתקיים $|E| > |V|$.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

הערכה פשוטה

בדוגמא שלנו, האלגוריתם עובר על כל אחד מן הצמתים בדיוק פעם אחת. לכל צומת v, האלגוריתם עובר על כל השכנים של v.

יש $|V|$ צמתים ולכל צומת יש לכל היותר $|V|-1 = O(|V|)$ שכנים. מיד נובע כי סיבוכיות האלגוריתם היא $O(|V^2|)$.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

הערכה משופרת

עלינו לנסות למצוא את הסיבוכיות כפונקציה של $|V|$ ושל $|E|$.

במקום לחסום את מספר הפעמים בהם עוברים בלולאה הפנימית, נחשב מספר זה במדויק:

עבור כל מעבר בלולאה החיצונית נרשום את מספר המעברים בלולאה הפנימית:

במעבר הראשון $\deg(v_1)$ פעמים.

במעבר השני $\deg(v_2)$ פעמים.

...

במעבר הi $\deg(v_i)$ פעמים.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

בסך הכל, בכל המעברים בלולאה החיצונית עוברים בלולאה הפנימית:

$$\sum_{i=1}^{|V|} \deg(v_i)$$

נשים לב, כי סכום זה, שוה בדיוק לסכום מספר האיברים בכל רשימות השכנים.

מאחר שכל קשת מופיעה **בדיוק** בשתי רשימות, פעם אחת עבור כל צומת קצה, נקבל כי מספר המעברים הכולל בלולאה הפנימית הוא $2|E|$.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

לסיכום

- על כל צומת עוברים פעם אחת באתחול ופעם אחת בסיור (בלולאה החיצונית).
- על כל קשת עוברים פעמיים (בסיור בלולאה הפנימית).

מכאן: סיבוכיות האלגוריתם היא $\Theta(|E|+|V|)$.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

הגדרה 1.4: תת גרף

יהי $G=(V,E)$ גרף. גרף $H=(V',E')$ הוא **תת גרף** של G אם מתקיים:

1. $V'\subseteq V$.
2. $E'\subseteq V'\times V'$ וגם $E'\subseteq E$

סימון

אם H הוא תת גרף של G מסמנים $H\subseteq G$.

שימו לב

הקשתות של H נלקחות מבין הקשתות של G אשר מחברות **צמתים** של H **בלבד**.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

הגדרה 1.5: מסלול בגרף

1. יהיו $c,d\in V$ שני צמתים בגרף G. מסלול מv_0 אל v_l הוא סדרת קשתות:

$$(v_0,v_1),(v_1,v_2),\ldots,(v_{l-1},v_l)$$

כאשר הקשת הראשונה נוגעת בצומת v_0, הקשת האחרונה נוגעת בצומת v_l, וכל קשת במסלול **סמוכה** אל הקשת הקודמת לה.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

<u>שימו לב</u>

1. בגרף מכוון אנו נדרוש כי כל הקשתות במסלול תהינה מכוונות באותו הכיוון : מן **הצומת הראשון** במסלול, אל ה**צומת האחרון.**
2. **אורך המסלול** מוגדר כמספר הקשתות במסלול.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

<u>הגדרה 1.5 (המשך)</u>

2. **אורך המסלול** מוגדר כמספר הקשתות במסלול.
3. אם $v_0 = v_l$, אזי α הוא **מעגל (Cycle)**.
4. מסלול (מעגל) α יקרא **פשוט (Simple)** אם אף צומת לא מופיע ב- α יותר מפעם אחת.
5. יהיו $\alpha = (v_1, v_2, ..., v_l)$ ו- $\beta = (v_l, v_{l+1}, ..., v_{l+m})$ שני מסלולים בגרף $G = (V, E)$ נגדיר את <u>**השרשור**</u> של המסלולים α ו- β כמסלול

$$\alpha \circ \beta = (v_0, v_1, ..., v_l, v_{l+1}, ..., v_{l+m})$$
.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

<u>הגדרה 1.6 : מרחקים בין צמתים</u>

יהי $G = (V, E)$ גרף קשיר ולא ממושקל ויהי $\alpha = v_0, v_{1,}, ..., \ v_l$ מסלול ב- G.

אורך המסלול α יוגדר כמספר הקשתות במסלול, l, ונסמן $|\alpha| = l$.

המרחק בין הצומת u לבין הצומת v מוגדר כ**אורך המסלול המינימלי** בין u לבין v.

$$\delta(u, v) = \min_\alpha (|\alpha|)$$

α מסלול מ u אל v

© כל הזכויות שמורות לפרופסור עמוס ישראלי

<u>למה 3.1</u>

יהי $G = (V, E)$ גרף כלשהו ויהיו $v_0, v_l \in V$ שני צמתים כלשהם. יהי $\alpha = v_0, v_2, ..., v_l$ מסלול **קצר ביותר** בין v_0 לבין v_l, כלומר :

$$|\alpha| = \sum_{i=0}^{l-1} w(v_i, v_{i+1}) = \delta(v_0, v_l).$$

יהי $\alpha_{ij} = v_i, ..., v_j$, $0 \le i < j \le l$, תת מסלול כלשהו של α, אזי α_{ij} הוא מסלול קצר ביותר מ- v_i אל v_j.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

<u>הוכחה</u>

נניח בשלילה כי קיימים שני אינדקסים, i ו-j, $0 \le i < j \le l$ כך שקיימים מסלול β_{ij}, מ-v_i אל v_j המקיים: $|\beta_{ij}| < |\alpha_{ij}|$.

במקרה זה, נתבונן במסלול α' מ-v_0 אל v_l המוגדר כך:

$$\alpha' = (v_0,...,v_i)\circ\beta_{ij}\circ(v_j,...,v_l)$$

המקיים $|\alpha'| < |\alpha|$, **סתירה** כי לפי הנחתנו הראשונית, α הוא מסלול קצר ביותר מ-v_0 אל v_l.

מש"ל

© כל הזכויות שמורות לפרופסור עמוס ישראלי

<u>סיכום</u>

בהרצאה זו, הגדרנו גרפים לסוגיהם השונים. לאחר מכן הגדרנו מסלולים ומרחקים בגרפים.
מטרתנו בקורס היא לסקור אלגוריתמים שונים המקבלים גרפים כקלט. כדי לבצע זאת קבענו למצוא דרך תקנית לייצוג גרפים במחשב.
לאחר מכן הצגנו את בעיית חישוב המרחקים בגרף לא ממושקל.
סיימנו את ההרצאה בהדגמת אלגוריתם BFS אותו נציג באופן פורמלי בהרצאה הבאה.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

הרצאה 2: חישוב מרחקים בגרפים לא ממושקלים

בהרצאה זו, נדון באלגוריתם **סיור לרוחב (Breadth First Search)** המכונה **BFS**.
לאחר חזרה קצרה, נציג את האלגוריתם, נוכיח את נכונות ונתח את הסיבוכיות שלו.

הגדרה 2.1

1. גרף $G(V,E)$ הוא **קשיר** אם בין כל שני צמתים $u,v\in V$ יש מסלול.
2. גרף $T(V,E)$ הוא **עץ** אם הוא **קשיר וחסר מעגלים.**

© כל הזכויות שמורות לפרופסור עמוס ישראלי

הגדרה 2.2: עץ פורש

יהי $G=(V,E)$ גרף קשיר.
יהי $T(V,E')$ תת גרף של G המקיים:
הגרף T הוא עץ וגם קבוצות הצמתים של G ושל T **שוות.**
בתנאים אלה הגרף T הוא **עץ פורש (Spanning Tree)** של הגרף G.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

הגדרה 2.2: עץ מרחקים

יהי $G=(V,E)$ גרף קשיר, יהי $T=(V,E')$ עץ פורש של G ויהי $s\in V$ צומת כלשהו בגרף G.
העץ T הוא **עץ מרחקים ביחס לגרף** G **ולצומת** $s\in V$ אם מתקיים:
לכל צומת $v\in V$

$$\delta_G(s,v)=\delta_T(s,v)$$

כלומר המרחק ב-G בין s לבין v **שווה** למרחק ב-T בין s לבין v.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

הגדרת הבעיה החישובית

בהרצאה זו נדון בבעיה החישובית הבאה:

קלט

1. גרף לא מכוון $G=(V,E)$.
2. צומת $s\in V$.

פלט

1. לכל צומת $v\in V$ משתנה $d[v]$ המקיים:
$$d[v]=\delta_G(s,v)$$
2. עץ מרחקים של G ביחס לצומת s, $T=(V,E')$.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

אפיון האלגוריתם

אלגוריתם BFS מקבל כקלט גרף (מכוון או לא) $G=(V,E)$ וצומת $s\in V$, ומחשב לכל צומת $v\in V$:

1. משתנה $d[v]$ המקיים:
$d[v]=\delta_G(s,v)$
2. משתנה $\pi(v)$ המקיים:
לכל צומת $v\in V$, הקשת $(\pi[v],v)$ היא קשת אחרונה במסלול מן הצומת s אל הצומת v שארכו $d[v]$.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

שימו לב

1. המסלול הזה הוא מסלול מינימלי בין הצומת s לבין הצומת v.
2. תהי $E'=\{(v,\Pi[v])v\in V-s\}$.
הגרף $T=(V,E')$, מהווה עץ מרחקים של G ביחס לצומת s.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

מבני נתונים - BFS

לכל צומת v נשמור:

1. משתנה $d[v]$ המיועד לחישוב המרחק של v מ-s. בסוף הביצוע מתקיים $d[v]=\delta(s,v)$.
2. משתנה $\pi[v]$ המיועד לאחסון (חלק מ) מסלול קצר ביותר מ-s אל v. בסוף הביצוע, $\pi[v]$ מאחסן את **הצומת הקודם של** v (האחרון לפני v) במסלול קצר ביותר מs אל v.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

מבני נתונים - BFS (המשך)

3. משתנה $color[v]$ המוגדר להלן:
בכל שלב בביצוע האלגוריתם, צמתי הגרף מחולקים לשלוש קבוצות:
   - **לבנים** צמתים שעוד לא התגלו.
   - **שחורים** - צמתים שהם וכל שכניהם כבר התגלו.
   - **אפורים** – צמתים שהתגלו אך חלק משכניהם טרם התגלו.

התור Q

בנוסף, האלגוריתם משתמש בתור (FIFO) Q, כדי לנהל את קבוצת הצמתים האפורים.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

תיאור האלגוריתם

אנו מניחים שהגרף נתון ושהשדות d, π, ו-$Color$, כבר הוגדרו. כמו כן, אנו מניחים שהתור Q מוגדר גם הוא.

בתחילת האלגוריתם מופעלת השיטה $BFS-Initialize(V,E,s)$ המאתחלת את כל המשתנים שהוספו לגרף.

הסיור עצמו מבוצע על ידי השיטה $BFS-Op(V,E,s)$. להלן מובא קוד האלגוריתם:

```
BFS(V,E,s)
  BFS-Initialize(V,E,s)
  BFS-Op(V,E,s)
```

אלגוריתם 2.1: BFS

תיאור האלגוריתם -אתחול

בתחילת הביצוע, לכל צומת $v \neq s$, $\delta(s,v)$ אינו ידוע ולכן ערך $d[v]$ הוא ∞, ערך $\pi[v]$ הוא $Null$ והצבע של v הוא לבן כל הערכים האלה ישתנו במהלך הסיור. כמו כן, הצומת v נמצא מחוץ לתור Q.

עבור הצומת s ידוע כי $\delta(s,s)=0$ וכי ל-s אין צומת קודם ולכן נקבעים הערכים 0 ו-$Null$ ל-$d[s]$ ול-$\pi[v]$ בהתאמה. ערכים אלה **לא ישתנו יותר.** כמו כן, צבעו של s נקבע **לאפור** והוא מוכנס אל התור Q.

בשקף הבא מופיע קוד האלגוריתם.

הקוד - האיתחול

$BFS-Initialize(V,E,s)$

$Q \leftarrow \phi$

for each vertex $v \in V[G]-\{s\}$

  $color[v] \leftarrow white$

  $\pi[v] \leftarrow Null$

  $d[v] \leftarrow \infty$

End_for

$color[s] \leftarrow gray$

$\pi[s] \leftarrow Null$

$d[s] \leftarrow 0$

$Q.Enque(s)$

אלגוריתם 2.2: אתחול BFS

תיאור האלגוריתם - הסיור

האלגוריתם קרוי **סיור לרוחב** מפני שהוא מתקדם **לרוחב הגרף.**

האלגוריתם פועל ב"פאזות" כאשר בתחילת פאזה i, כל הצמתים שמרחקם מן הצומת s הוא i, נמצאים בתור. קבוצה זו מכונה בשם ה"שכבה ה-i" . במהלך הפאזה ה-i האלגוריתם "מגלה" את כל הצמתים שמרחקם מ-s הוא $i+1$, מכניס אותם לתור, ובמקביל מוציא מן התור את הצמתים שמרחקם מ-s הוא i. בסוף הפאזה, כל הצמתים שמרחקם מ-s הוא $i+1$ נמצאים בתור Q ואלה הם הצמתים היחידים ב-Q.

תיאור מפורט של פאזה

במהלך כל פאזה, הסיור מתבצע באיטרציות: בכל איטרציה, מטפלים בצומת הנמצא בראש התור. נניח כי זהו u. במהלך האיטרציה, רשימת השכנים של הצומת u נסרקת, וכל שכן v שצבעו לבן נצבע באפור, ומתבצע

$$d[v]\leftarrow d[u]+1$$
$$\pi[v]\leftarrow u$$

והצומת v מוכנס אל התור Q.

לאחר שכל הרשימה נסרקה, הצומת u נצבע לשחור ומוצא מן התור.

ביצוע האלגוריתם מסתיים כאשר התור מתרוקן.

הקוד - הסיור

```
BFS-Op(V,E,s)
 phase ← 0
while Q ≠ φ
   while d[head(Q)] = phase
      u ← Q.deque( )
      for each v ∈ adj[u]
         if color[v] = white
              color[v] ← gray
              d[v] ← d[u]+1
              π[v] ← u
              Q.enque(v)
          end_if
      end_for
         color[u] ← black
    end_while
    phase ← phase+1
end while
end
```

אלגוריתם 2.3: *BFS-Op*

הוכחת נכונות האלגוריתם

ההוכחה תערך בשלושה שלבים:

1. הוכחת למה (משפט עזר) בנוגע לתכונות כלליות של גרפים.
2. הוכחת למה בנוגע לאלגוריתם.
3. הוכחת משפט הנכונות.

שימו לב

האלגוריתם פועל הן על גרפים לא מכוונים והן על גרפים מכוונים.

אנו נוכיח את נכונות האלגוריתם עבור גרפים פשוטים, לא מכוונים, וקשירים. הכללת ההוכחה לגרפים מכוונים אינה קשה.

הגדרה 2.1

יהי $s\in V$ צומת כלשהו בגרף פשוט, לא מכוון, וקשיר $G=(V,E)$.

נגדיר את ***השכבה ה-i, ביחס לצומת s***, כקבוצת כל הצמתים שמרחקם מן הצומת s הוא i. נסמן קבוצה זו ב-L_i.

שימו לב

$$L_0=\{v:\delta(s,v)=0\}=\{s\}$$
$$L_i=\{v:\delta(s,v)=i\}$$

למה 2.1

יהי $v \in V - s$, צומת שרירותי. שימו לב כי $\delta(s,v) = l > 0$. אזי,

1. לצומת v , יש לפחות שכן אחד u המקיים $\delta(s,u) = l - 1$.
2. לצומת v, אין אף שכן u המקיים $\delta(s,u) < l - 1$.
3. לצומת v, אין אף שכן u המקיים $\delta(s,u) > l + 1$.

שימו לב

השכנים של v נמצאים כולם בשכבות $l+1, l, l-1$.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

הוכחה

1. נתון כי $\delta(s,v) = l > 0$. מן ההגדרה נובע כי קיים מסלול α, מs אל v ,ובו לפחות קשת אחת. יהי u הצומת האחרון ב-α לפני v . אפשר להוכיח (תרגיל בית) כי $\delta(s,u) = l - 1$
2. אם היה לv שכן u המקיים $\delta(s,u) = f < l - 1$, היה מסלול מs אל v שארכו $f + 1 < l$ בסתירה להנחה כי $\delta(s,v) = l$.
3. לכל u, שכן של v, מתקיים: $\delta(s,u) \le \delta(s,v) + 1 = l + 1$.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

למה 2.2

לכל צומת $v \in V$, ערכי המשתנים $d[v]$, ו-$\Pi[v]$ משתנים לכל היותר פעם אחת במהלך הסיור.

הוכחה

הטענה נובעת מיידית מן הקוד.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

משפט

לכל $l \ge 0$, במהלך ביצוע BFS, יש רגע יחיד, עם תום הפאזה ה-$l-1$, בו מתקיימות הטענות הבאות:

1. כל צומת v, המקיים $\delta(s,v) = l$, נמצא בתור Q, צבעו אפור, ומתקיים:
   - $d[v] = l = \delta(s,v)$
   - אם $\delta(s,v) > 0$ אזי קיים u המקיים $\pi[v] = u$ כאשר $\delta(s,u) = l - 1 = \delta(s,v) - 1$.
   - יש מסלול באורך l בין s אל v וכל הקשתות במסלול הן מהצורה $(\pi[x], x)$.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

2. כל צומת v, המקיים $\delta(s,v) < l$, נמצא מחוץ לתור Q, צבעו שחור,ומתקיים:
   - $d[v] = \delta(s,v)$
   - אם $\delta(s,v) > 0$ אז $\pi[v] = u$ כאשר $\delta(s,u) \le \delta(s,v) - 1$.
3. כל צומת v, המקיים $\delta(s,v) > l$, נמצא מחוץ לתור Q, צבעו לבן, ומתקיים:
   - $d[v] = \infty$
   - $\pi[v] = nil$

© כל הזכויות שמורות לפרופסור עמוס ישראלי

<u>הוכחה (באינדוקציה על l)</u>

<u>בסיס</u> ($l = 0$)

הצומת היחיד שמרחקו מs הוא 0 הוא s עצמו. ברגע שהאתחול מסתיים, s נמצא בתור ומתקיים: $d[s] = 0$, $\pi[s] = NIL$ ו$color[s] = grey$.

המסלול הריק הוא מסלול מs אל עצמו וכל (0) הקשתות שלו הן מן הצורה $(\pi[v], v)$.

סעיף 2 נכון באופן ריק, כי אין אף צומת v המקיים $\delta(s,v) < l$.

לכל צומת v, $v \neq s$, v נמצא מחוץ לתור ומתקיים: $color[v] = white$ $d[v] = \infty$ כנדרש על ידי סעיף 3.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

<u>צעד האינדוקציה</u>

נניח כי סעיפים 1,2,3 מתקיימים עם תום הפאזה ה- l בביצוע BFS עבור l , כלשהו כאשר $l > 0$.

באותו רגע, כל השכבה ה -l של הBFS , נמצאת בתוך התור Q, כל הצמתים בשכבות הקודמות, צבועים בשחור, וכל הצמתים בשכבות הבאות צבועים בלבן, עדיין לא נכנסו אל התור Q וסימונם הוא ∞.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

<u>צעד האינדוקציה (המשך)</u>

ברצוננו להוכיח כי עם תום הפאזה ה - $l + 1$ של האלגוריתם, טענות 1,2,3 מתקימות עבור $l + 1$. נתבונן בפעולות המתבצעות במהלך הפאזה ה - $l + 1$, בה עוברת לולאת האלגוריתם על כל הצמתים הנמצאים בתור בתחילת הפאזה:

עבור כל צומת כזה, u, הצומת u מוצא מן התור, שכני u נסרקים וכל אחד מהם, v, שצבעו לבן, מוכנס אל התור, נצבע באפור, ומקבל ערכים $d[v] = l + 1$, ו - $\pi[v] = u$.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

<u>הוכחה (באינדוקציה על l)</u>

<u>צעד האינדוקציה (המשך)</u>

לאחר מכן, הצומת u, מוצא מן התור ונצבע בשחור.
עלינו להוכיח כי עם תום הפאזה, לאחר שהאלגוריתם עבר על כל אחד מן הצמתים המסומנים ב l שנמצאו בתור בתחילת הפאזה, כל צומת $v \in V$, מקיים את התנאי המתאים למרחקו של v מן הצומת s.

<u>הוכחת 1</u>

אם v מקיים $\delta(s,v) = l+1$, אזי:
לפי הנחת האינדוקציה, בתחילת הפאזה צבעו של v היה לבן. לפי **למה 1** לv יש לפחות שכן אחד w, המקיים $\delta(s,w) = l$. לפי הנחת האינדוקציה, בתחילת הפאזה, כל הצמתים אשר מרחקם מs הוא l, נמצאים בתור Q.
יהי u, הצומת הראשון בתור המקיים $\delta(s,u) = l$ ו u שכן של v .
(**שימו לב:** יתכן כי $u = w$).

כאשר u מוצא מן התור, שכניו נסרקים והצומת v , אשר לפי הנחת האינדוקציה, צבעו עד אותו רגע הוא לבן, נצבע לאפור, המשתנה $d[v]$ מקבל את הערך $l+1$, וערך $\pi[v]$ נקבע ל u .
לפי הנחת האינדוקציה יש מסלול α מ s אל u המקיים: $|\alpha| = l$, וכל הקשתות במסלול α הן מן הצורה $(\pi[x], x)$. נתבונן במסלול $\alpha \circ (u,v)$:
ארכו הוא $l+1$ וכל הקשתות בו הן מן הצורה $(\pi[x], x)$ כנדרש בסעיף 1.

<u>הוכחת 2</u>

אם v מקיים $\delta(s,v) < l+1$, אזי לפי הנחת האינדוקציה, מתקיים $d[v] < \infty$ ו$\pi[v] \neq nil$ וערכים אלה מקימים את הדרישות בתחילת הפאזה ה$l+1$. לפי **למה 2**, ערכים אלה לא ישתננו יותר ויקיימו אותן הדרישות בסוף פאזה ה $l+1$. נותר לנו להוכיח כי בסוף הפאזה ה - $l+1$, צבעו של v הוא שחור. אם $\delta(s,v) < l$, הטענה נובעת מיידית מהנחת האינדוקציה. אם $\delta(s,v) = l$, הטענה נובעת מפעולת האלגוריתם במהלך הפאזה ה - $l+1$, כפי שתוארה בפתיחת ההוכחה.

הוכחת 3

אם $\delta(s,v) > l+1$, אזי לפי הנחת האינדוקציה, בתחילת הפאזה ה - $l+1$ כל הטענות מתקיימות.

לפי **למה 1**, לצומת v אין אף שכן שמרחקו מ s קטן או שווה ל l ומכאן, לפי הנחת האינדוקציה, בתחילת הפאזה ה - $l+1$, לצומת v אין אף שכן בתור Q. מכיוון שהתור הוא $FIFO$, ברור כי אף שכן של v, לא ישמש בתפקיד u במהלך הפאזה ה - $l+1$.

ומכאן: ערכי v לא ישתנו במהלך הפאזה וכל דרישות המשפט יתקיימו גם בסוף הפאזה ה - $l+1$.

מש"ל

© כל הזכויות שמורות לפרופסור עמוס ישראלי

הגדרת עץ פורש

גרף לא מכוון $G(V,E)$ הוא **קשיר (Connected)** אם בין כל שני צמתים ב - G יש מסלול.

גרף G הוא **חסר מעגלים (Acyclic)** אם ב G אין אף מעגל.

עץ $T(V,E)$ הוא גרף **לא מכוון קשיר וחסר מעגלים.**

יהי $G(V,E)$ גרף לא מכוון וקשיר.
העץ $T(V,E')$ (**שימו לב:** קבוצת הצמתים של T **שווה** לקבוצת הצמתים של G) **פורש את הגרף** G אם מתקיים $E' \subseteq E$.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

עצי מרחקים

יהי $G(V,E)$ גרף קשיר ונניח כי $T(V,E')$ הוא **עץ** פורש של G.

העץ $T(V,E')$ הוא **עץ מרחקים ביחס ל** G **ולצומת** $s \in V$ אם מתקיים:
לכל צומת $v \in V$

$$\delta_G(s,v) = \delta_T(s,v)$$

© כל הזכויות שמורות לפרופסור עמוס ישראלי

תוצאה

1. אלגוריתם BFS מחשב נכונה את המרחקים מצומת המקור s אל כל שאר הצמתים בגרף.
2. קבוצת הקשתות $E' = \{(\pi[v],v), v \in V - s\}$ מהווה עץ מרחקים, כלומר: לכל צומת $v \in V$, קבוצת הקשתות מכילה מסלול היחיד מ s אל v ומסלול זה הוא מסלול קצר ביותר מ s אל v.

הוכחה

יהי $v \in V$ צומת שרירותי ונניח כי $\delta(s,v) = l$.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

1. לפי המשפט, במהלך הפאזה הl נקבע $d[s]=l$. לפי **למה 2** הערך של $d[v]$ לא ישתנה שוב במהלך האלגוריתם.
2. לפי המשפט, יש מסלול מs אל v שארכו l וכל הקשתות בו הן מן הצורה $(\pi[v],v)$. מאחר ולכל צומת, v $v\in V$, $v\neq s$ מתאימה בדיוק קשת אחת, בקבוצה $E'=\{(\pi[v],v),v\in V-s\}$ יש $|V|-1$ קשתות. באחת ההרצאות הבאות, נוכיח כי קבוצת קשתות כזו מגדירה עץ.

מש"ל

© כל הזכויות שמורות לפרופסור עמוס ישראלי

סיור לרוחב - הסיבוכיות

תזכורת: הסיבוכיות של אלגוריתם היא פונקציה המתאימה לכל גודל קלט את זמן הריצה המתאים.
באלגוריתם, אשר הקלט שלו הוא גרף, גודל הקלט מתבטא על ידי גודל קבוצת קשתות $|E|$ וגודל קבוצת הצמתים $|V|$.
אנו נבטא את סיבוכיות האלגוריתמים שנפתח כפונקציות של $|V|$ ושל $|E|$.
נשים לב כי בדרך כלל מתקיים $|E|>|V|$.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

סיור לרוחב - הסיבוכיות

טענה 1: עלות האתחול היא $\Theta(|V|)$.
הוכחה: מיידית.
טענה 2: כל צומת נצבע פעם יחידה לאפור ופעם יחידה לשחור.
הוכחה: מיידית.
טענה 3: האלגוריתם עובר בדיוק פעם אחת על כל רשימת שכנויות.
הוכחה: מיידית.
טענה 4: סיבוכיות BFS היא $\Theta(|V|+|E|)$.
הוכחה: מיידית.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

סיכום

בהרצאה זו, חזרנו אל בעיית חישוב המרחקים בגרף לא ממושקל ופתרנו אותה בעזרת אלגוריתם **סיור לרוחב.**
ההרצאה הסתיימה בהוכחת נכונות האלגוריתם ובחישוב הסיבוכיות שלו.
לאחר מכן, הגדרנו **עץ מרחקים** והוכחנו כי אלגוריתם BFS מחשב עץ מרחקים של גרף הקלט.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

הרצאה 3: גרפים ממושקלים

בהרצאה זו, נדון **בגרפים ממושקלים** בהם לכל קשת צמוד מספר הקרוי **משקל** או **אורך**. במהלך ההרצאה נדון בסעיפים הבאים:

1. הגדרת גרפים ממושקלים ומרחקים בגרפים ממושקלים.
2. אלגוריתם Dijkstra המקבל כקלט גרף ממושקל עם פונקציית משקל חיובית, וצומת מקור ומחשב לכל צומת בגרף את המרחק מצומת המקור.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

גרפים ממושקלים

גרף ממושקל הוא שלשה, $G=(V,E,w)$ כאשר $G=(V,E)$ הוא גרף ו- $w:E\rightarrow R$ היא **פונקציית משקל** אשר מתאימה לכל $e\in E$, את הערך $w(e)$. ערך זה נקרא **המשקל (או האורך) של** w.

גרף ממושקל הוא מודל מצוין למקרים בהם קשרים בין איברים שונים אינם זהים כמו במקרה שקשת מבטאת **מרחק גיאוגרפי**, **מחיר למעבר מנקודה לנקודה**, **עלות של שינוי במצב**, וכדומה.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

מסלולים ומרחקים בגרפים ממושקלים

הגדרה 4.1

1. יהי $G=(V,E,w)$ גרף ממושקל לא מכוון. יהי $\alpha=u_0,u_1,\ldots,u_l$ מסלול בגרף G. **אורך המסלול** α יוגדר כסכום משקלי קשתות המסלול.

$$|\alpha|=\sum_{i=0}^{l-1}w(u_i,u_{i+1})$$

2. לכל שני צמתים $u,v\in V$ המרחק בין u לבין v מוגדר על ידי

$$\delta(u,v)=\min_{\alpha}(|\alpha|)$$
α מסלול מ u אל v

שימו לב: הגדרת המרחק זהה להגדרה עבור גרפים לא ממושקלים. ההבדל נעוץ בהגדרת אורך הקשת.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

תזכורת - עצי מרחקים

יהי $G=(V,E,w)$ גרף קשיר וממושקל ויהי $T=(V,E')$ עץ פורש של G.

העץ $T=(V,E')$ הוא **עץ מרחקים ביחס ל-** G **ולצומת** $s\in V$ אם מתקיים:

לכל צומת $v\in V$

$$\delta_G(s,v)=\delta_T(s,v)$$

כאשר המרחקים ב- G וב- T נמדדים בהתאם להגדרה עבור גרפים ממושקלים.

שימו לב: ההבדל היחיד בהגדרות שהגדרנו עד כה הוא בהגדרת ה**מרחק**. שאר ההגדרות נותרו זהות.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

הבעיה החישובית

קלט:

1. גרף ממושקל לא מכוון $G=(V,E,w)$.
   לכל $e\in E$ $w(e)\geq 0$.
2. צומת $s\in V$ (המקור).

פלט:

1. לכל צומת $v\in V$ ערך $d[v]$ המקיים:
   $$d[v]=\delta(s,v)$$
2. לכל צומת $v\in V$, צומת קודם $\pi[v]$ המקיים: הקשת $(\pi[v],v)$ היא הקשת אחרונה במסלול קצר ביותר בין s לבין v.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

אלגוריתם Dijkstra

אלגוריתם Dijkstra מקבל כקלט גרף $G=(V,E,w)$ עם פונקציית משקל **אי-שלילית**, כלומר, לכל $e\in E$ $w(e)\geq 0$, וצומת מקור s, ומחשב את המרחקים בין s לבין כל צמתי הגרף.

אלגוריתם Dijkstra פועל הן עבור גרפים מכוונים והן עבור גרפים בלתי מכוונים. להלן נציג את האלגוריתם עבור **גרף לא מכוון**.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

אלגוריתם Dijkstra - מבני נתונים

כמו באלגוריתם BFS, לכל צומת v, יהיו שני משתנים: $d[v]$ ו-$\pi[v]$. סיום האלגוריתם, מתקיים $d[v]=\delta(s,v)$ כלומר ערכו של $d[v]$ שווה למרחק בין הצומת s לבין הצומת v. המשתנה $\pi[v]$ מאחסן את הצומת **לפני האחרון** במסלול קצר ביותר מ s אל v. קבוצת הקשתות
$$E'=\{(\pi[v],v):v\in V-s\}$$
משרה עץ מרחקים. כלומר הגרף $T=(V,E')$ הוא עץ מרחקים של G ביחס לצומת s.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

תיאור האלגוריתם

במבט על, אלגוריתם דאקסטרה, דומה מאוד ל-BFS:
בתחילת האלגוריתם מופעלת השיטה $Dijkstra-Init(V,E,s)$ המאתחלת את כל המשתנים שהוספו לגרף.
הסיור עצמו מבוצע על ידי השיטה $Dijkstra-Op(V,E,s)$. להלן מובא קוד האלגוריתם:

$$\begin{array}{l} Dijkstra(V,E,w,s) \\ \quad Dijkstra-Init(V,E,w,s) \\ \quad Dijkstra-Op(V,E,w,s) \end{array}$$

אלגוריתם 3.1: אלגוריתם Dijkstra

© כל הזכויות שמורות לפרופסור עמוס ישראלי

תיאור האלגוריתם -אתחול

בתחילת הביצוע ידוע כי יש מסלול באורך 0 מ-s אל עצמו ולכן ערכו התחילי של המשתנה $d[s]$ הוא 0. לכל אורך האלגוריתם, לצומת s אין צומת קודם ולכן מאתחלים:

$$d[s] \leftarrow 0$$
$$\pi[s] \leftarrow Null$$

לכל צומת v, $v \neq s$, לא ידוע דבר על מרחקו של v מ s ולכן מאתחלים:

$$d[v] \leftarrow \infty$$
$$\pi[v] \leftarrow Null$$

בשקף הבא מופיע קוד האלגוריתם.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

הקוד - האיתחול

$Dijkstra-Init(V,E,w,s)$
foreach vertex $v \in V-\{s\}$
  $d[v] \leftarrow \infty$
  $\pi[v] \leftarrow Null$
end foreach
$d[s] \leftarrow 0$
$\pi[s] \leftarrow Null$
end

אלגוריתם 3.2: *Dijkstra-Init*

© כל הזכויות שמורות לפרופסור עמוס ישראלי

אלגוריתם Dijkstra - הנחת היסוד

בכל שלב של אלגוריתם Dijkstra ולכל צומת $v \in V$, אם $d[v]=l<\infty$, אזי בגרף G קיים מסלול, α, בין הצומת s לבין הצומת v.
ארכו של המסלול α, הוא l והקשת האחרונה במסלול זה היא $(\pi[v],v)$.
α הוא המסלול קצר ביותר בין s לבין v, **המוכר לנו בשלב זה.**

שימו לב: הנחת היסוד מובאת כאן ברמה אינטואיטיבית בלבד. ההנחה תוכח בלמה 3.2.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

צעד שיפור

נניח כי בגרף $G=(V,E)$ מבוצע אלגוריתם Dijkstra ובשלב כלשהו מתקיים:

- $u \in V$ הוא צומת ב-G ו-$d[u]=l$.
- $(u,v) \in E$ קשת ב G.

בשלב זה, לפי הנחת היסוד, ב-G יש מסלול α מ-s אל u ו-$d[u]=|\alpha|$.
כתוצאה, ב G יש מסלול המסלול מן הצומת s אל הצומת v המתקבל על ידי המסלול α שבקצהו משורשרת הקשת (u,v). נסמן את המסלול הזה ב $\alpha \circ (u,v)$ וארכו הוא $|\alpha|+w(u,v)=d[u]+w(u,v)$.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

<u>צעד שיפור - המשך</u>

השיטה $Relax(u,v,w)$ בודקת אם

$$d[v] > d[u] + w(u,v)$$

אם התנאי מתקיים, ב-G יש מסלול מ-s אל v, שארכו הוא $d[u]+w(u,v)$ ולכן אפשר **להקטין** את $d[v]$ על ידי ביצוע

$$d[v] \leftarrow d[u] + w(u,v)$$

לאחר השינוי, הקשת האחרונה במסלול מ-s אל v, שארכו הוא $d[v]$ היא הקשת (u,v) ולכן מבצעים:

$$\pi[v] \leftarrow u$$

© כל הזכויות שמורות לפרופסור עמוס ישראלי

<u>צעדי שיפור – השיטה *Relax*</u>

להלן הקוד של השיטה *Relax*:

$Relax(u,v,w)$

if $d[v] > d[u] + w(u,v)$ **then**

  $d[v] \leftarrow d[u] + w(u,v)$

  $\pi[v] \leftarrow u$

endif end

<u>אלגוריתם 3.3: השגרה *Relax*</u>

© כל הזכויות שמורות לפרופסור עמוס ישראלי

<u>למה 3.1</u>

נניח כי בשלב כלשהו התבצעה השגרה $Relax(u,v,w)$. לאחר ביצוע השגרה מתקיים $d[v] \le d[u] + w(u,v)$.

<u>הוכחה</u>

ההוכחה נשארת כתרגיל לקורא.

אם לפני ביצוע $Relax(u,v,w)$ מתקיים $d[v] \le d[u] + w(u,v)$, השיטה לא משנה שום ערך. במקרה זה נאמר כי התבצע **צעד שיפור ריק.**

© כל הזכויות שמורות לפרופסור עמוס ישראלי

<u>תיאור השיטה *Dijkstra-op*</u>

לאחר האתחול, מתבצעת השיטה *Dijkstra-op*. בתחילת הביצוע, כל צמתי גרף הקלט מוכנסים ל**<u>תור עדיפויות</u>** PQ. לאחר מכן מתבצעת לולאה:

בכל מעבר בלולאה, נשלף מ-PQ הצומת u, עבורו הערך $d[u]$ הוא מינימלי. לאחר מכן מתבצע צעד שיפור מן הצומת u אל כל אחד מן השכנים שלו.

. לאחר מכן מבצעים צעד שיפור מ-u אל כל אחד מן השכנים שלו.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

השיטה Dijkstra-op - הקוד

```
Dijkstra-op(V, E, w)
    PQ ⇐ V(G)
    while PQ ≠ φ do
        u ← ExtractMin(PQ)
        for each v ∈ adj(u) do
            Relax(u, v, w)
        end for
    end while
```

אלגוריתם 3.4: *Dijkstra-op*

© כל הזכויות שמורות לפרופסור עמוס ישראלי

נכונות אלגוריתם Dijkstra

הטענה היסודית

בלמה הבאה אנו מוכיחים את הטענה היסודית שהובאה קודם.

למה 3.2

יהי $G=(V,E)$ גרף מכוון ויהי $s \in V$ צומת כלשהו ב G. נניח כי מבצעים ב-G את אלגוריתם Dijkstra החל מן הצומת s. לכל $v \in V$, בכל שלב של האלגוריתם, אם מתקיים $d[v] < \infty$, אזי בגרף G יש מסלול α מ-s אל v ומתקיים:

1. $|\alpha| = d[v]$.
2. אם $s \neq v$ אז $(\pi[v], v)$ היא הקשת אחרונה במסלול α.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

אינטואיציה

עבור הצומת s, הטענה מידית.

עבור $v \neq s$: אם $d[v] = l$, אזי $d[v]$ הוא תוצאה של צעד שיפור מצומת כלשהו u.

אם הטענה נכונה עבור u, אזי היא נכונה עבור v באופן מידי.

בדרך זו, אפשר לסגת מן הצומת v אל u ומ-u אל $\pi[u]$ וכן הלאה.

בסוף התהליך חייבים להגיע אל הצומת s, ממנו מתחילים כל צעדי השיפור.

הדרך הפורמלית להוכיח טענה כזו היא באינדוקציה על סדר צעדי השיפור.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

הוכחה (באינדוקציה)

תהי $R = r_1, r_2, ...$ סדרת צעדי השיפור שבוצעו במהלך האלגוריתם. הוכחת המשפט היא באינדוקציה על סדר צעדי השיפור:

בסיס $(i=0)$

לפני ביצוע r_1 מתקיים:

1. $d[s] = 0$.
2. לכל צומת $v \neq s$ $d[v] = \infty$.

בשלב זה, הטענה מתקיימת לכל הצמתים.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

<u>שלב האינדוקציה</u>

נניח כי כל הטענה נכונה עבור $r_1, r_2, \ldots, r_i$, כלומר לכל צומת $u \in V$, אם $d[u] < \infty$ אזי קיים מסלול α מ-s אל u, $|\alpha| = d[u]$ והקשת $(\pi[u], u)$ היא קשת אחרונה במסלול α.

נניח כי $r_{i+1} = Relax(u,v,w)$. נתבונן במקרים הבאים:

<u>מקרה 1: השיפור הוא ריק</u>

במקרה זה, לכל $v \in V$, ערך $d[v]$ לא השתנה בשלב r_{i+1} ולכן הטענה נובעת באופן מידי מהנחת האינדוקציה.

<u>שלב האינדוקציה (המשך)</u>

<u>מקרה 2: השיפור שבוצע אינו ריק</u>

במקרה זה, הצומת היחיד שערך d שלו משתנה במהלך r_{i+1} הוא v. לכל צומת $u \neq v$, נכונות הטענה לגבי u נובעת מידית מהנחת האינדוקציה.

נתבונן כעת בצומת v:

לאחר סיום צעד השיפור מתקיים:

$d[v] = d[u] + w(u,v)$ ו- $\pi[v] = u$.

לפי הנחת האינדוקציה, יש מסלול מ-s אל u שארכו הוא $d[u]$. מכאן נובע מידית כי יש מסלול מ-s אל v שארכו הוא $d[v]$ והקשת האחרונה במסלול זה היא (u,v).

מש״ל

<u>תוצאה 3.3</u>

בכל שלב בביצוע אלגוריתם Dijkstra ולכל צומת $v \in V$, מתקיים

$$d[v] \geq \delta(s,v)$$

<u>הוכחה</u>

לפי למה 3.3, לכל צומת v ובכל שלב של האלגוריתם, אם $d[v] \neq \infty$ אז $d[v]$ הוא אורך של מסלול כלשהו מ-s אל v.

הטענה נובעת מידית מהגדרת $\delta(s,v)$ כאורך המסלול **המינימלי** מ s אל v.

<u>למה 3.5</u>

יהי $G = (V,E)$ גרף, מכוון או לא מכוון, עם פונקציית משקל אי שלילית w. אם מפעילים על G אלגוריתם Dijkstra, החל מצומת $s \in V$ אזי, לכל צומת $v \in V$, כאשר v עוזב את Q, מתקיים:

$$d[v] = \delta(s,v)$$

<u>אינטואיציה להוכחה</u>

מניחים כי הטענה נכונה עד להוצאת הצומת ה-t מתור העדיפויות PQ ומוכיחים זאת עבור פעולת ההוצאה שמספרה $t+1$. באופן פורמלי זוהי הוכחה באינדוקציה על סדר הוצאת הצמתים מ-PQ.

<u>הוכחה</u>

נוכיח את הלמה באינדוקציה על סדר פעולות הוצאת צמתים מתור העדיפויות PQ.

<u>בסיס</u> ($v = s$)

מיד לאחר האתחול $d[s]=0$ ולכל צומת $v \neq s$ מתקיים $d[v]=\infty$. במצב זה, הצומת הראשון המוצא מ-Q הוא s וברור כי המסלול הריק מהווה מסלול מs אל עצמו וארכו הוא

$$d[s]=0=\delta(s,s)$$

© כל הזכויות שמורות לפרופסור עמוס ישראלי

<u>שלב האינדוקציה</u>

נניח כי הטענה נכונה לכל אחד מן הצמתים שהוצאו מ-PQ בזמנים $1,2,...,t$ ויהי v הצומת המוצא מ PQ בזמן $t+1$.

עלינו להוכיח כי בזמן $t+1$ מתקיים

$$d[v]=\delta(s,v)$$

יהי $\alpha = s, v_1, v_2, ..., v_l = v$ מסלול מינימלי מ s אל v ויהי v_k, $k < l$, הצומת האחרון על α אשר הוצא מ-PQ לפני זמן $t+1$. תמיד יש צומת כזה כי α מתחיל ב-s שכבר הוצא מ-PQ.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

לפי הנחת האינדוקציה, בזמן הוצאת v_k מ-Q מתקיים, $d[v_k]=\delta(s,v_k)$.

מיד לאחר הוצאת v_k מPQ מתבצעים צעדי שיפור מv_k אל כל שכניו ובפרט מתבצע

$$Relax(v_k, v_{k+1}, w)$$

מאחר ש α הוא מסלול מינימלי, נובע **מלמה 4.3** כי לאחר ביצוע צעד השיפור הזה מתקיים:

$$d[v_{k+1}]=\delta(s,v_{k+1})$$

© כל הזכויות שמורות לפרופסור עמוס ישראלי

<u>המשך הוכחת הלמה</u>

אם $v_{k+1}=v_l$, המשפט נובע מידית מן הטענה.

אם לא אזי מתקיים:

$$\delta(s,v_l)\le d[v_l]\le d[v_{k+1}]=$$
$$=d[v_{k+1}]=\delta(s,v_{k+1})\le\delta(s,v_l)$$

תרגיל: ציין את הסיבה לנכונות כל אחד מאי השוויונים בשורה זו.

מאחר שמשני צדי שורת אי השוויונים נמצא ביטוי זהה, נובע כי בשורת אי השוויונים, אפשר להחליף כל אחד מן הסימנים $\le$ בסימן $=$.

מש״ל

© כל הזכויות שמורות לפרופסור עמוס ישראלי

משפט

נניח כי אלגוריתם Dijkstra מקבל כקלט גרף $G(V,E,w)$ עם פונקציית משקל **אי-שלילית** w (מכוון או שאינו מכוון) וצומת $s\in V$.

אזי :

1. האלגוריתם מחשב נכונה את מרחקי כל הצמתים בG מן הצומת s.
2. קבוצת הקשתות
$$E'=\{(v,\pi[v]),v\in V-s\}$$
מהווה **עץ מרחקים ב** G.

הוכחה

סעיף 1 נובע מידית מלמה 5.

סעיף 2 נובע מידית מלמה 3.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

סיבוכיות אלגוריתם Dijkstra

עלות האתחול היא $\Theta(|V|)$. הלולאה העיקרית מתבצעת $\Theta(|V|)$ פעמים. בכל שלב מתבצעים שני תהליכים :

1. בחירת צומת מינימלי.
2. שיפור כל שכניו.

במהלך כל צעדי השיפור אנו בוחנים כל קשת פעמים, פעם אחת בכל כיוון.

מכאן, עלות כל צעדי השיפור ביחד היא $\Theta(|E|)$. בחירת צומת מינימלי בצורה ישירה דורשת $\Theta(|V|)$ צעדים. בחירה זו נערכת $\Theta(|V|)$ פעמים ולכן הסיבוכיות המתקבלת היא

$$\Theta(|V|^2+|E|)=\Theta(|V|^2)$$

© כל הזכויות שמורות לפרופסור עמוס ישראלי

שיפור הסיבוכיות עבור גרפים לא צפופים

גרף המקיים $|E|<\frac{|V|^2}{\log|V|}$ נקרא **גרף לא צפוף**.

אם G אינו צפוף אפשר לשפר את סיבוכיות האלגוריתם כדלהלן :

1. הקבוצה Q תוחזק כ**ערימת מינימום**.
2. בחירת איבר מינימלי מQ תבוצע בזמן $O(\log|V|)$.
3. כל צעד שיפור, הגורם לשינוי $d[v]$ ידרוש **תיקון** של Q. עלות התיקון היא $O(\log|V|)$.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

הוספת מצביעים

כדי לבצע את השיפור המוצע, כל צומת ייוצג כעת על ידי :

1. איבר ברשימת הצמתים.
2. צומת מייצג בערימה Q.

במצב זה, צריך לאפשר גישה ישירה, בזמן קבוע, בין שני המייצגים של כל צומת v. כדי לעשות זאת, מוסיפים לכל איבר ברשימת הצמתים מצביע אל הצומת המייצג בערימה. באופן דומה, כל צומת בערימה, יחוזק על ידי מצביע אל האיבר המייצג ברשימת הצמתים לכל איבר ברשימת הצמתים.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

סיבוכיות האלגוריתם עבור גרפים לא- צפופים

סיבוכיות האלגוריתם תחושב כך:

1. בחירת צומת מינימלי תארך $O(\log|V|)$. עלות כל הבחירות יחד היא $O(|V|\log|V|)$.
2. כל אחד מ$|E|$ צעדי השיפור, יש לבצע תיקון בערימה. עלות כל התיקונים ביחד היא:
$O(|E|\log|V|)$
3. הסיבוכיות הכוללת היא
$O((|E|+|V|)\log|V|)=$
$O(|E|\log|V|)$

© כל הזכויות שמורות לפרופסור עמוס ישראלי

סיכום

בהרצאה זו, הצגנו את מודל הגרף הממושקל והמרחק בין צמתים בגרפים ממושקלים.
לאחר מכן, הצגנו את אלגוריתם Dijkstra לחישוב מרחקים בגרפים ממושקלים **אי-שליליים** הוכחנו את נכונותו וניתחנו את הסיבוכיות שלו.
סיימנו את ההרצאה בהצגת מימוש נוסף לאלגוריתם תוך שימוש בערמה כדי להשיג סיבוכיות טובה יותר על יותר עבור גרפים לא-צפופים.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

© כל הזכויות שמורות לפרופסור עמוס ישראלי

הרצאה 4: מרחקים בגרפים עם פונקצית משקל כללית

בהרצאה זו, נדון בגרפים ממושקלים עם פונקציית משקל המקבלת ערכים ממשיים כלשהם, וזאת בניגוד לגרפים עם פונקציית משקל שערכיה הם אי-שליליים. פונקציית משקל כזו מאפשרת קיום מעגל בגרף שסכום משקלי הקשתות שלו שלילי.

במצב זה, המרחק בין כל זוג צמתים הנמצאים על **מעגל שלילי**, אינו מוגדר.

בהרצאה זו, נבין את המשמעות של ההגדרות הללו ונציג את אלגוריתם Bellman-Ford, המחשב את המרחקים מצומת כלשהו אל שאר הצמתים בגרף תוך בדיקת קיום מעגלים שליליים בגרף.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

קשתות שליליות ומעגלים שליליים

יהי $G=(V,E,w)$ גרף ממושקל ונניח כי הפונקציה w יכולה לקבל ערכים שליליים.

הגדרה 4.1

1. קשת $e\in E$ היא **שלילית** אם $w(e)<0$.
2. מעגל הוא **שלילי** אם סכום משקלי הקשתות שלו הוא שלילי.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

מרחקים בגרפים ממושקלים כללית

יהי $G=(V,E,w)$ גרף עם פונקצית משקל כללית, ויהי σ מעגל שלילי בגרף G.

לכל שני צמתים $u,v\in\sigma$, המרחק בין u לבין v אינו מוגדר, כי אפשר להגיע מ-u אל v, תוך ביצוע מספר בלתי חסום של **מעברים במעגל** σ, וכל מעבר במעגל, מוסיף מספר שלילי לאורך המסלול.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

מרחקים בגרפים ממושקלים כללית (המשך)

יהי $s\in V$ צומת מקור בגרף G. אם ב-G יש מסלול מן הצומת s אל אחד הצמתים ב-σ, אז המרחק מן הצומת s אל כל צומת ב-σ, גם הוא $-\infty$.

למעשה, המרחק מן הצומת s, אל כל צומת ב-G, אשר נגיש מ-σ, גם הוא $-\infty$. לכן, במקרה שקיים ב-G מעגל שלילי, לא נדון במרחקים בין צמתים ב-G.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

קשתות שליליות בגרף לא מכוון

יהי $G=(V,E,w)$ גרף לא מכוון עם פונקצית משקל כללית. אם ב-G יש קשת שלילית, $e=(u,v)$ אזי הקשת e מהווה מעגל שלילי, כי אפשר לעבור על פני הקשת e הלוך וחזור. לכן, מעתה ועד לסיום פרק זה, נדון אך ורק בגרפים מכוונים.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

זיהוי מעגלים שליליים

בגרף מכוון וממושקל שיש בו קשתות שליליות, אפשר לחשב מרחקים, רק אם **אין בו מעגלים שליליים**.

הגדרה 4.2: מעגל פשוט

מעגל $\sigma=(v_0,v_1,\ldots,v_l=v_0)$ הוא **פשוט** אם אין בו צומת החוזר על עצמו (מלבד v_0).

משפט 4.1:

אפ בגרף $G=(V,E,w)$ קיים מעגל שלילי אז ב-G קיים מעגל שלילי פשוט.

הוכחה

תרגיל לקורא.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

חסם תחתון על מספר המעגלים בגרף מלא

כמה מעגלים מכוונים פשוטים יש בגרף?

התשובה המלאה תלויה כמובן בגרף. כאן אנו נסתפק במשפט פשוט המראה כי מספר המעגלים בגרף מלא הוא גבוה מאוד.

למה 4.2

יהי $G=(V,E)$ גרף מכוון ומלא עם n צמתים, כאשר בין כל שני צמתים ב-G יש שתי קשתות אנטי-מקבילות. בגרף G יש $(n-1)!$ מעגלים פשוטים באורך n.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

הוכחה

תהי $V=\{v_1,v_2,\ldots,v_n\}$ ותהי π תמורה שרירותית. נתבונן בסדרת הצמתים $\sigma=\left(v_{\pi(1)},v_{\pi(2)},\ldots,v_{\pi(n)}\right)$. מאחר שהגרף מלא, הסדרה σ מהווה מעגל פשוט בגרף. למעשה המעגל הזה מוגדר על ידי n תמורות שונות המתקבלות על ידי הפעלת תמורה ציקלית על התמורה π.

קל לראות כי כל תמורה המתחילה במספר מסוים, למשל 1, מגדירה מעגל פשוט באורך n בגרף G (הסבירו למה). ולכן מספר המעגלים הפשוטים באורך n הוא $n!/n=(n-1)!$.

מש״ל

© כל הזכויות שמורות לפרופסור עמוס ישראלי

<u>מסקנה</u>

אם ננסה לבדוק האם קיים מעגל שלילי, בגרף על ידי בדיקה פרטנית של כל אחד מן המעגלים, נקבל סיבוכיות **מעריכית (אקספוננציאלית).**

לכן עלינו לחפש אלגוריתם מתוחכם יותר לפתרון הבעיה, וזהו אלגוריתם Bellman-Ford.

להלן הצגה פורמלית של הבעיה ואחריה הצגת האלגוריתם.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

<u>הבעייה החישובית</u>

<u>קלט</u>

1. גרף מכוון וממושקל $G=(V,E,w)$ עם פונקציית משקל **כללית.**

2. צומת $s \in V$.

<u>פלט</u>

1. חיווי האם ב-G יש מעגל שלילי.
2. אם ב-G לא קיים מעגל שלילי, אז לכל $v \in V$ יש לחשב:

   2.1 $d[v]=\delta(s,v)$.

   2.2 $\pi[v]$ הוא צומת אחרון במסלול מ-s אל v שארכו $\delta(s,v)$.

<u>שימו לב</u>

דרישות 2.1 ו-2.2 בהגדרת הפלט זהות להגדרת הפלט באלגוריתם Dijkstra.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

<u>האלגוריתם של Bellman-Ford</u>

האלגוריתם של Bellman-Ford מבצע פעולה כפולה:

1. **האלגוריתם "מכריע" האם קיים מעגל שלילי, נגיש מ - s .**
2. במקביל, האלגוריתם מחשב ערכים $d[v]$ ו $\pi[v]$ לכל צומת $v \in V$.

בסיום האלגוריתם מתקיים:

במקרה שלא קיים מעגל שלילי, $d[v]=\delta(s,v)$ ו-$\pi[v]$ הוא צומת לפני אחרון במסלול מs אל v שארכו $\delta(s,v)$, כמו בסיום אלגוריתם Dijkstra.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

<u>תיאור אלגוריתם Bellman-Ford</u>

האלגוריתם דומה מאוד לאלגוריתם Dijkstra:

1. מבני הנתונים זהים, למעט תור העדיפויות.
2. האתחול זהה.
3. כמו Dijkstra, האלגוריתם מבצע סדרה של צעדי שיפור על גרף הקלט. להלן הקוד:

$$
\begin{aligned}
&BelmanFord(V,E,w,s)\\
&\quad Dijkstra-Init(V,E,w,s)\\
&\quad BelmanFord-Op(V,E,w,s)
\end{aligned}
$$

<u>אלגוריתם 4.1: אלגוריתם Bellman-Ford</u>

© כל הזכויות שמורות לפרופסור עמוס ישראלי

<u>הנחת היסוד</u>

בכל שלב של אלגוריתם Bellman-Ford ולכל צומת $v \in V$, אם $d[v]=l<\infty$, אזי בגרף G קיים מסלול, α, בין הצומת s לבין הצומת v.

ארכו של המסלול α, הוא l וזהו המסלול קצר ביותר בין s לבין v, <u>**המוכר לנו בשלב זה.**</u>

נכונות הטענה כבר הוכחה ב**למה 3.2** בהרצאה הקודמת.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

<u>תאור גוף האלגוריתם</u>

מסדרים את הקשתות בסדר שרירותי ומבצעים $|V|$ מעברים על קשתות הגרף. בכל מעבר, מבצעים צעד שיפור על כל אחת מקשתות הגרף, לפי הסדר שנקבע.

אם במעבר האחרון מתבצע שיפור שאינו ריק, מודיעים על קיום מעגל שלילי. במקרה זה, יש להתעלם מן הערכים $d[v]$ ו-$\pi[v]$ אשר כבר חושבו.

אם במעבר האחרון לא מתבצע שיפור באף קשת - הערכים $d[v]$ ו-$\pi[v]$ נכונים.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

<u>האלגוריתם</u>

```
BellmanFord-Op(V,E,w,s)
  for  i ← 1 to |V|-1
     foreach (u,v) ∈ E
               Relax(u,v,w)
     end foreach
  end for
 foreach edge(u,v) ∈ E  (* test *)
    if d[v] > d[u] + w(u,v)
       return(True) (* יש מעגל *)
 end foreach
 return(False) (* אין מעגל *)
```

<u>אלגוריתם 4.2: BellmanFord-Op</u>

© כל הזכויות שמורות לפרופסור עמוס ישראלי

<u>סיבוכיות אלגוריתם Bellman-Ford</u>

נחרוג ממנהגנו ונדון בסיבוכיות האלגוריתם לפני שנוכיח את נכונותו.

סיבוכיות האלגוריתם היא כמובן $\Theta(|V|\cdot|E|)$

במקרה הגרוע ביותר, עבור גרפים צפופים, הסיבוכיות מגיעה ל- $\Theta(|V|^3)$.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

נכונות אלגוריתם Bellman-Ford

לצורך נוחיות ההוכחה, נניח כי כל צמתי גרף הקלט **נגישים** מצומת המקור s.

למה 4.3

אם בגרף הקלט G אין מעגל שלילי, אז לכל צומת $v \in V$, יש מסלול מינימלי **פשוט** מצומת המקור s אל v. בכל מסלול פשוט יש לכל היותר n צמתים ו-$n-1$ קשתות.

הוכחה

תרגיל לקורא.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

למה 4.4

אם בגרף הקלט אין מעגל שלילי, האלגוריתם מחשב מרחקים נכונים לאחר $|V-1|$ מעברים בלולאה החיצונית. לאחר מכן, לא יהיו יותר צעדי שיפור. כמו כן, קבוצת הקשתות

$$\{(v,\pi[v]) : v \in V-s\}$$

משרה עץ מרחקים בגרף G ביחס לצומת s.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

הוכחה

לפי **למה 4.3**, לכל צומת $v \in V$, קיים מסלול מינימלי מצומת המקור s אל הצומת v ובו l קשתות, $0 \le l < n$.

שימו לב

בגרף ממושקל, מספר הקשתות במסלול שונה מ**אורך המסלול**.

לדוגמא

בגרף המשורטט כאן ...

© כל הזכויות שמורות לפרופסור עמוס ישראלי

הוכחה (המשך)

להלן נוכיח, באינדוקציה על המעברים בלולאה החיצונית (לולאת ה-**for**), כי לאחר i מעברים בלולאה מתקיים:
לכל צומת $v \in V$, אם קיים מסלול מינימלי מצומת המקור s אל הצומת v ובו i קשתות, $0 \le i < n$, אז לאחר i מעברים בלולאת ה-**for** מתקיים $d[v] = \delta(s,v)$.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

בסיס: $(l=i=0)$:

הצומת היחיד שיש מסלול פשוט מ-s אליו, הוא צומת המקור s עצמו.

נכונות הטענה נובעת מן האתחול.

מינימלי אליו.

הנחת האינדוקציה $(l=i)$:

נניח כי לכל צומת $u\in V$, אם קיים מסלול מינימלי מ-s אל u, ובו i קשתות, אז אחרי i מעברים בלולאה מתקיים $d[u]=\delta(s,u)$.

הוכחה $(l=i+1)$:

יהי $v\in V$ צומת כלשהו ב-

ויהי $G=(V,E)$

$\alpha=(s=v_0,v_1,\ldots,v_{i+1}=v)$ מסלול פשוט מינימלי בין s ל- v.

יהי $\alpha_i=(s=v_0,v_1,\ldots,v_i)$ תת המסלול המתקבל על ידי קיצוץ הקשת האחרונה ב-α. המסלול α_i הוא תת מסלול של מסלול פשוט ומינימלי ולכן גם הוא פשוט ומינימלי.

לפי הנחת האינדוקציה, לאחר i מעברים בלולאה מתקיים $d[v_i]=\delta(s,v_i)$. במעבר ה- $i+1$ מתבצעת הקריאה $Relax(v_i,v_{i+1},w)$ וברור כי לאחר ביצוע הקריאה הזאת מתקיים $d[v_{i+1}]=\delta(s,v_{i+1})$.

קל לראות כי לכל צומת $v\in V$, לאחר שמתקיים $d[v]=\delta(s,v)$, הערך של $d[v]$ לא ישתנה יותר.

נכונות עץ המרחקים נובעת ישירות מהוכחת אלגוריתם דאקסטרה.

מש״ל

משפט 4.2: נכונות האלגוריתם

אלגוריתם Bellman-Ford מזהה קיום מעגל שלילי בגרף ובמקרה שלא קיים מעגל כזה, האלגוריתם מחשב נכון את מרחקי הצמתים מצומת המקור ואת עץ המרחקים.

הוכחה

נכונות האלגוריתם במקרה שבגרף הקלט אין מעגל שלילי הוכחה **בלמה 4.4**. נותר לנו להוכיח כי אם ב-G קיים מעגל שלילי, אז במעבר האחרון בלולאה (ה-test) יהיה צעד שיפור שאינו ריק. באופן אינטואיטיבי ברור כי תמיד יהיה צעד שיפור לאורך אחת מקשתות המעגל. להלן נוכיח טענה זו באופן פורמלי.

<u>המשך ההוכחה</u>

נניח בשלילה, כי בגרף G יש מעגל שלילי, נגיש מן הצומת s, אך ברגע מסוים, אין יותר צעדי שיפור לאורך המעגל.

יהי המעגל השלילי

$\sigma = v_0 v_1, v_2, ..., v_l = v_0$.

מאחר שהמעגל נגיש מן הצומת s, לאחר לכל היותר $n-1$ מעברים בלולאה החיצונית, לכל צומת v_i על המעגל מתקיים

$$d[v_i] < \infty$$

© כל הזכויות שמורות לפרופסור עמוס ישראלי

מאחר שלאורך המעגל אין צעדי שיפור, לכל קשת (v_i, v_{i+1}) לאורך המעגל מתקיים:

$$d[v_{i+1}] \le d[v_i] + w(v_i, v_{i+1})$$

נרשום את אי השוויונים האלה האחד לאחר השני ונקבל:

$$d[v_1] \le d[v_0] + w(v_0, v_1)$$
$$d[v_2] \le d[v_1] + w(v_1, v_2)$$
$$\cdots$$
$$\cdots$$
$$d[v_0] = d[v_l] \le d[v_{l-1}] + w(v_{l-1}, v_l)$$

© כל הזכויות שמורות לפרופסור עמוס ישראלי

אם נסכם את כל אי השוויונים האלה לאורך המעגל השלילי, ונזכור כי $v_0 = v_l$ נקבל:

$$\sum_{i=0}^{l} d[v_i] \le \sum_{i=0}^{l} d[v_i] + \sum_{i=1}^{l} w(v_i, v_{i+1})$$

שני הסכומים הראשונים הם שווים שכן אלו בדיוק אותם הצמתים ולכן נקבל:

$$\sum_{i=1}^{l} w(v_i, v_{i+1}) \ge 0$$

בסתירה להנחה כי המעגל הוא שלילי.

מש״ל

© כל הזכויות שמורות לפרופסור עמוס ישראלי

<u>סיכום</u>

בהרצאה זו, פתחנו את אלגוריתם Bellman-Ford אשר מקבל כקלט גרף מכוון ממושקל כללית $G = (V, E, w)$ וצומת $s \in V$, בודק האם ב-G קיים מעגל שלילי.

לאחר סיום הבדיקה, האלגוריתם מודיע האם קיים מעגל שלילי.

אם לא קיים מעגל שלילי אז:

1. לכל $v \in V$ מתקיים $d[v] = \delta(s, v)$.
2. קבוצת הקשתות $\{(\pi[v], v) : v \in V - s\}$ משרה עץ מרחקים בגרף G ביחס לצומת s.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

הרצאה 5: מציאת כל המרחקים בגרף

עד כה, טיפלנו בקביעת מרחקים של כל הצמתים בגרף מצומת מקור.
לעתים אנו מעונינים למצוא את המרחקים ההדדיים בין כל זוגות הצמתים בגרף בבת אחת.
בהרצאה זו, נדון בבעיה הזאת.
במהלך ההרצאה, נציג את אלגוריתם Floyd-Warshall (להלן אלגוריתם Floyd) הפותר בעיה זו בעזרת תכנות דינמי בסיבוכיות $\Theta\left(|V|^3\right)$.

מציאת כל המרחקים בגרף

הדרך הפשוטה ביותר לבצע מטלה זו היא להפעיל $|V|$ פעמים את אלגוריתם Dijkstra או Bellman-Ford (בהתאם לפונקצית המשקל), פעם אחת מכל צומת.
אם פונקצית המשקל אינה שלילית, נשתמש באלגוריתם Dijkstra והסיבוכיות תהיה $\Theta\left(|V|^3\right)$ (או $O\left(|V|\,|E|\log|V|\right)$ עבור גרפים דלילים).
במקרה של פונקצית משקל כללית נשתמש באלגוריתם Bellman - Ford ונקבל סיבוכיות $O\left(|V|^2\cdot|E|\right)$.

אלגוריתם Floyd - Warshall

ביכולתנו לחשב את כל המרחקים כולם בעזרת אלגוריתם Floyd - Warshall (להלן Floyd).
אלגוריתם זה, משתמש בתכנות דינמי ומחשב את כל המרחקים בגרף עם פונקצית משקל כלשהי בדרך פשוטה ויעילה.

מטריצת משקלים לגרף ממושקל

יהי $G=(V,E,w)$ גרף ממושקל פשוט. **מטריצת המשקלים** של G, $W(G)$ מייצגת את מבנה הגרף ואת משקלי הקשתות. מבנה המטריצה דומה למבנה מטריצת סמיכויות כאשר עבור כל קשת שומרים את המשקל שלה.
הגדרת מטריצת המשקלים היא:

$$w_{ij}=\begin{cases} w(i,j) & (i,j)\in E \\ 0 & i=j \\ \infty & (i,j)\notin E \wedge i\neq j \end{cases}$$

שימו לב

1. לכל $0 \le i \le n$,
$w_{ii} = \delta(i,i) = 0$.
2. אם G הוא גרף לא מכוון, $W(G)$ היא מטריצה סימטרית.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

דוגמה 5.1

נתבונן בגרף הבא:

© כל הזכויות שמורות לפרופסור עמוס ישראלי

המשך הדוגמה

מטריצת **המשקלים** של הגרף היא:

0	1	2	3	4	5
1	0	1	10	∞	1
2	1	0	1	10	7
3	10	1	0	1	∞
4	∞	10	1	0	20
5	1	7	∞	20	0

זוהי מטריצה סימטרית, כלומר מטריצת משקלים של גרף **לא מכוון**.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

צמתי בינים במסלולים וגרף מושרה

הגדרה 5.1 צומת בינים

יהי $G=(V,E)$ גרף. יהי $\alpha = u_0, u_1, ..., u_l$ מסלול ב G. **צומת בינים** ב-α הוא כל צומת u_i, $1 \le i < l$

תזכורת: תת גרף

יהי $G(V,E)$ גרף. גרף $H(V',E')$ הוא **תת גרף** של G אם מתקיים:

1. $V' \subseteq V$.
2. $E' \subseteq V' \times V'$ וגם $E' \subseteq E$

שימו לב: קשתותיו של H נלקחות מבין הקשתות של G אשר מחברות **צמתים** של H **בלבד**.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

<u>תת גרף מושרה</u>

יהי $G=(V,E)$ גרף ותהי $V'\subseteq V$, תת קבוצה כלשהי של קבוצת הצמתים של הגרף G. תת הגרף המושרה על ידי קבוצת הצמתים V' הוא הגרף $H=(V',E')$. הקבוצה E' מוגדרת כך: $E'=\{(u,v)\mid u,v\in V'\}$. כלומר הקבוצה E' מכילה את כל הקשתות ששני הצמתים שלהם מוכלים בקבוצה V'.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

<u>הסימונים</u> $\alpha_{ij}^{(k)}$ ו- $d_{ij}^{(k)}$

יהי $G=(V,E,w)$ גרף שאין בו מעגלים שליליים ונניח כי $V=\{1,2,...,n\}$

1. לכל זוג צמתים, i,j, $1\le i<j\le n$, נסמן ב $\alpha_{ij}^{(k)}$ את אחד המסלולים המינימליים בגרף G המשתמש בצמתי ביניים רק בצמתים מבין $1,2,...,k$
2. נסמן $d_{ij}^{(k)}=|\alpha_{ij}^{(k)}|$.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

<u>שימו לב לנקודות הבאות:</u>

1. $\alpha_{ij}^{(k)}$ הוא מסלול קצר ביותר מן הצומת i אל הצומת j, בתת הגרף המושרה על ידי קבוצת הצמתים $\{i,j,1,...,k\}$.
2. לעתים $\alpha_{ij}^{(k)}$ אינו קיים. במקרה זה נסמן $d_{ij}^{(k)}=\infty$.
3. בגרף לא מכוון תמיד מתקיים $\alpha_{ij}^{(k)}=\alpha_{ji}^{(k)}$.
4. הסימון $\alpha_{ij}^{(0)}$ מסמן את המסלול הקצר ביותר בין i לj ללא כל צמתי ביניים כלומר הקשת (i,j) אם היא קיימת.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

<u>הסימונים</u> $\alpha_{ij}^{(k)}$ ו $d_{ij}^{(k)}$ <u>(המשך)</u>

נתבונן ב-$\alpha_{ij}^{(n)}$. לפי ההגדרה זהו אחד המסלולים הקצרים ביותר בין הצומת i לבין הצומת j אשר צמתי הבינים שלו משתייכים לקבוצה $\{1,2,...,n\}$. מאחר שקבוצה זו מכילה את כל צמתי הגרף, $\alpha_{ij}^{(n)}$ הוא מסלול קצר ביותר בין הצומת i לצומת j בגרף G ומתקיים $|\alpha_{ij}^{n}|=d_{ij}^{(n)}=\delta(i,j)$.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

דוגמאות

בגרף הקודם, שמטריצת **אורכי הקשתות** שלו היא:

0	1	2	3	4	5
1	0	1	10	∞	1
2	1	0	1	10	7
3	10	1	0	1	∞
4	∞	10	1	0	20
5	1	7	∞	20	0

$d_{12}^{(0)} = 1 \quad d_{45}^{(0)} = 20 \quad d_{13}^{(0)} = 10$

$d_{12}^{1} = 1 \quad d_{25}^{(1)} = 2 \quad d_{35}^{(1)} = 11$

$d_{35}^{(2)} = 3$

© כל הזכויות שמורות לפרופסור עמוס ישראלי

תרגיל

נתון גרף ממושקל $G=(V,E,w)$ ושלישית מספרים $1 \le i, j, k \le n$.

הציעו אלגוריתם לחישוב $\alpha_{ij}^{(k)}$.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

המטריצות $D^{(k)}$

יהי $G=(V,E,w)$ גרף ממושקל. נסמן ב- $D^{(0)}$ את מטריצת המשקלים של G, $W(G)$. לכל k, $1 \le k \le n$, נסמן ב $D^{(k)}$ את המטריצה שהאיבר ה i, j שלה הוא $d_{ij}^{(k)}$.

אלגוריתם Floyd מחשב את המטריצה $D^{(n)}$, כלומר, לכל $1 \le i, j \le n$ האלגוריתם מחשב את

$$d_{ij}^{(n)} = |\alpha_{ij}^{(n)}| = \delta(i, j)$$

וזאת ללא חישוב אף מסלול אחד.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

אלגוריתם Floyd - תיאור כללי

אלגוריתם Floyd מקבל כקלט את המטריצה $D^{(0)} = W(G)$ של G.

האלגוריתם עובד ב-n איטרציות. באיטרציה ה-k, $1 \le k \le n$, משתמשים במטריצה $D^{(k-1)}$ ומחשבים את המטריצה $D^{(k)}$.

כלומר, באיטרציה ה-k, עבור כל זוג צמתים, i, j, $1 \le i, j \le n$ נחשב את האורך של $\alpha_{ij}^{(k)}$.

באיטרציה ה-n האלגוריתם מחשב את $D^{(n)}$, כלומר, לכל i ו-j, $1 \le i, j \le n$, האלגוריתם מחשב את $d_{ij}^{(n)} = \delta(i, j)$.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

אלגוריתם Floyd- Warshall

$Floyd(V,E,w)$
$D^{(0)} \leftarrow W$
for $k \leftarrow 1$ **to** n (1)
  for $i \leftarrow 1$ **to** n (2)
    for $j \leftarrow 1$ **to** n (3)
      $d_{ij}^{k} \leftarrow \min(d_{ij}^{k-1}, d_{ik}^{k-1} + d_{kj}^{k-1})$
    end for (3)
  end for (2)
  for $i \leftarrow 1$ **to** n (2)
    if $d_{ii}^{k} < 0$ **return**$(False)$
      (*יש מעגל שלילי*)
  end for (2)
end for (1)
return $D^{(n)}$

אלגוריתם 5.1: Floyd-Warshall

אלגוריתם Floyd - האתחול

כדי להתחיל בביצוע האלגוריתם עלינו לחשב את $D^{(0)} = W$.

תרגיל

נתון גרף $G=(V,E)$ המיוצג בעזרת רשימת שכנויות.
הציגו אלגוריתם לחישוב המטריצה $W(G)$.
מהי סיבוכיות האלגוריתם?
מהי סיבוכיות הבעיה?

אלגוריתם Floyd – האיטרציות

לאחר האתחול מבצעים n איטרציות.
בכל איטרציה, האלגוריתם מחשב את המטריצה $D^{(k)}$ בעזרת איברי המטריצה $D^{(k-1)}$,
לאחר חישוב $D^{(k)}$ מתבצעת לולאה נוספת אשר בודקת קיום מעגלים שליליים.

הוכחת נכונות אלגוריתם Floyd

נכונות האלגוריתם נובעת משני המשפטים הבאים:

משפט 5.1

לכל גרף ממושקל $G=(V,E,w)$, אם ב-G אין מעגלים שליליים אזי לכל $1 \le i < j \le |V|$ ולכל $1 \le k \le n$:

$$d_{ij}^{(k)} = \min\left\{d_{ij}^{(k-1)}, d_{ik}^{(k-1)} + d_{kj}^{(k-1)}\right\}$$

הוכחה

נשים לב לעובדה כי מאחר שידוע כי בגרף G אין מעגלים שליליים, לכל $1 \le i,j,k \le n$, קיים מסלול מינימלי **פשוט** מן הצומת i אל הצומת j.

<u>הוכחה (המשך)</u>

נסמן $G_{ij}^{(k)}$ את תת הגרף המושרה על ידי קבוצת הצמתים $\{i, j, 1, ..., k\}$.

נשים לב לעובדה כי מאחר שידוע כי בגרף G אין מעגלים שליליים, לכל $1 \le i, j, k \le n$, קיים מסלול מינימלי **<u>פשוט</u>** מן הצומת i אל הצומת j .

לכל $1 \le i, j, k \le n$ נגדיר את $\alpha_{ij}^{(k)}$ כאחד המסלולים המינימליים הפשוטים, מן הצומת i אל הצומת j .

<u>הוכחה (המשך 2)</u> לכל $1 \le i, j, k \le n$,

$\alpha_{ij}^{(k-1)}$

ו-$\alpha_{ij}^{(k-1)} \circ \alpha_{kj}^{(k-1)}$ הם מסלולים ב-$G_{ij}^{(k)}$,

ומתקיים:

1. $|\alpha_{ij}^{(k-1)}| = d_{ij}^{(k-1)}$.
2. $|\alpha_{ij}^{(k-1)} \circ \alpha_{kj}^{(k-1)}| = d_{ik}^{(k-1)} + d_{kj}^{(k-1)}$

נתבונן במקרים הבאים:

<u>מקרה 1: הצומת k אינו שייך ל</u> $\alpha_{ij}^{(k)}$

במקרה זה, ברור כי $d_{ij}^{(k)} = d_{ij}^{(k-1)}$

והטענה מתקיימת.

<u>מקרה 2: הצומת k שייך ל</u> $\alpha_{ij}^{(k)}$

לפי הטענה הקודמת, המסלול $\alpha_{ij}^{(k)}$ הוא **פשוט** ומכאן, הצומת k מופיע בו **פעם אחת בלבד**. נסמן

$$\alpha_{ij}^{(k)} = \beta_{ik} \circ \beta_{kj}$$

כאשר β_{ik} מתחיל ב-i ומסתיים ב-k

ו-β_{kj} מתחיל ב-k ומסתיים ב-j .

כל הצמתים האחרים ב-$\alpha_{ij}^{(k)}$ משתייכים ל $1,2,...,k-1$.

עלינו להוכיח כי

$$|\beta_{ik}| = d_{ik}^{(k-1)} = |\alpha_{ik}^{(k-1)}|$$

וכי

$$|\beta_{kj}| = d_{kj}^{(k-1)} = |\alpha_{kj}^{(k-1)}|$$

<u>הוכחה (המשך)</u>

מאחר ש-$\alpha_{ij}^{(k)}$ הוא מסלול מינימלי בגרף $G_{ij}^{(k)}$, והמסלול β_{ik} (ו-β_{kj} בהתאמה) הוא תת מסלול של $\alpha_{ij}^{(k)}$, נובע כי β_{ik} (ו-β_{kj} בהתאמה) הוא מסלול מינימלי בין i ל-j (ובין j ל-k בהתאמה).

מאחר ש-k אינו צומת ביניים באף אחד מן המסלולים האלה נקבל כי

$$|\beta_{ik}| = d_{ik}^{(k-1)} = |\alpha_{ik}^{(k-1)}|$$

וכי

$$|\beta_{kj}| = d_{kj}^{(k-1)} = |\alpha_{kj}^{(k-1)}|$$

מש״ל

הוכחת נכונות (המשך)

המשפט הבא מוכיח כי במקרה שבגרף הקלט יש מעגל שלילי, האלגוריתם מזהה זאת:

משפט 5.2

יהי $G=(V,E,w)$ גרף ממושקל עם פונקצית משקל כללית ונניח שמפעילים על G את אלגוריתם Floyd. יהי α מעגל שלילי ב-G.
נסמן את שני הצמתים הגדולים ביותר בα ב-k_1 וב-k_2 כאשר $k_1 > k_2$.
אזי האלגוריתם יגלה קיום מעגל שלילי לכל המאוחר באיטרציה k_2.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

הוכחה

נניח כי האלגוריתם לא מגלה מעגל שלילי לפני איטרציה k_2.
נראה כעת כי בתום איטרציה k_2-1 מתקיים

$$d^{k_2-1}_{k_1,k_2}+d^{k_2-1}_{k_2,k_1}\le|\alpha|$$

נתבונן בפרוק המעגל α, $\alpha=\beta_1\circ\beta_2$ כאשר β_1 הוא מסלול מכוון מk_1 אל k_2 וβ_2 מסלול מכוון מk_2 אל k_1. צמתי הבינים ב-β_1 וב-β_2 נמנים על הצמתים $1,2,...,k_2-1$, ולכן מתקיים:

$$d^{k_2-1}_{k_1,k_2}\le|\beta_1|$$

$$d^{k_2-1}_{k_2,k_1}\le|\beta_2|$$

© כל הזכויות שמורות לפרופסור עמוס ישראלי

מאחר ש$\beta_1\circ\beta_2=\alpha$ הטענה נובעת.
בזמן האיטרציה הk_2 מתבצע:

$$d^{k_2}_{k_1k_1}\leftarrow\min\left\{d^{k_2-1}_{k_1k_1},d^{k_2-1}_{k_1k_2}+d^{k_2-1}_{k_2k_1}\right\}$$

ומכאן נובע:

$$d^{k_2}_{k_1k_1}\le|\alpha|<0$$

במצב זה, לולאת הבדיקה מודיעה על קיום מעגל שלילי.

מש״ל

© כל הזכויות שמורות לפרופסור עמוס ישראלי

הסיבוכיות

כדי לחשב את הסיבוכיות, נשים לב כי אנו מבצעים 3 לולאות מקוננות ובכל אחת מהן אנו עוברים בדיוק $(|V|)$ פעמים ולכן סיבוכיות האלגוריתם היא $\Theta(|V|^3)$.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

חישוב מרחקים בגרף – סיכום

הטבלה הבאה מסכמת את האלגוריתמים השונים המחשבים את המרחקים ממקור לכל הצמתים :

לא ממושקל מכוון ולא מכוון	BFS	$\Theta(\|V\|+\|E\|)$
ממושקל אי שלילי מכוון ולא מכוון	Dijkstra	$\Theta(\|V\|^2)$ $\Theta(\|E\|\log\|V\|)$
ממושקל כללי מכוון בלבד	Ford	$\Theta(\|V\|\cdot\|E\|)$

© כל הזכויות שמורות לפרופסור עמוס ישראלי

חישוב מרחקים בגרף – סיכום

הטבלה הבאה מסכמת את האלגוריתמים השונים המחשבים את המרחקים בין כל זוג צמתים :

לא ממושקל מכוון ולא מכוון	n פעמים BFS	$\Theta(\|V\|\|E\|)$
ממושקל חיובי מכוון ולא מכוון	n פעמים Dijkstra	$\Theta(\|V\|\|V\|^2)$ $\Theta(\|V\|\|E\|\log\|V\|)$
ממושקל חיובי מכוון ולא מכוון או כללי מכוון בלבד	Floyd	$\Theta(\|V\|^3)$

© כל הזכויות שמורות לפרופסור עמוס ישראלי

הרצאה 5: סיכום

בהרצאה זו, דנו במציאת המרחקים ההדדיים בין כל זוגות הצמתים בגרף בבת אחת.

רוב ההרצאה הוקדש ללימוד אלגוריתם Floyd-Warshall הפותר בעיה זו בסיבוכיות $\Theta(|V|^3)$.

בסיום ההרצאה סיכמנו את נושא המסלולים והמרחקים בגרף על ידי הצגת טבלה בה השוונו את כל האלגוריתמים שלמדנו עד כה.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

הרצאה 6: עצים פורשים בגרף

עץ פורש הוא תת גרף שבו בין כל שני צמתים בגרף קיים מסלול יחיד. בדרך זו, כל עץ פורש מהווה תת גרף חסר יתרות המאפשר תקשורת בין כל שני צמתים בגרף.

בהרצאה זו, נגדיר את המושגים **עץ פורש** ו**עץ פורש מינימום** ונציג את אלגוריתם Prim המקבל כקלט גרף ממושקל ומחשב עץ פורש מינימום של גרף הקלט.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

עצים

גרף לא מכוון G הוא **קשיר (Connected)** אם בין כל שני צמתים ב - G יש מסלול.

גרף G הוא **חסר מעגלים (Acyclic)** אם אין בו אף מעגל.

עץ הוא גרף **לא מכוון, קשיר, וחסר מעגלים.**

© כל הזכויות שמורות לפרופסור עמוס ישראלי

תכונות עצים

משפט

בין כל שני צמתים בעץ **יש מסלול יחיד**.

הוכחה

אם יש שני מסלולים בין זוג צמתים כלשהו אזי בגרף יש **מעגל**.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

משפט

יהי $T=(V,E)$ עץ (גרף קשיר וחסר מעגלים) אזי

$$|E|=|V|-1$$

הוכחה

באינדוקציה על מס' הצמתים $|V|=n$ של T.

בסיס ($|V|=1$)**:** יהי T עץ עם צומת יחיד. הקשת היחידה האפשרית ב T היא לולאה עצמית וברור כי בגרף חסר מעגלים לא תתכן קשת כזו.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

צעד האינדוקציה:

הנחה: נניח כי בכל עץ עם $|V|=n$ צמתים יש $n-1$ קשתות.

הוכחה: יהי T עץ שרירותי עם $n+1$ צמתים. להלן נראה כי ב- T יש n קשתות:

בעץ T יש לפחות שני עלים.

נתבונן בעץ T' המתקבל מ- T על ידי הורדת אחד העלים ביחד עם הקשת המחברת אותו לעץ. בעץ T' יש n צמתים, ולפי הנחת האינדוקציה יש בו $n-1$ קשתות. מכאן נובע באופן מיידי כי בעץ המקורי, T, יש n קשתות.

מש"ל

© כל הזכויות שמורות לפרופסור עמוס ישראלי

עצים פורשים

נתון גרף לא מכוון $G=(V,E)$. **עץ פורש (Spanning Tree)** של G, הוא תת גרף $T(V,E')$ של G המקיים:

1. $T=(V,E')$ הוא עץ (קשיר וחסר מעגלים)
2. ל- G ול- T קבוצת צמתים **זהה**, V.
3. $E'\subseteq E$

תרגיל

מצאו אלגוריתם המקבל כקלט גרף קשיר ומחזיר עץ פורש לגרף הקלט. מהי סיבוכיות האלגוריתם? האם יתכן אלגוריתם יעיל יותר?

© כל הזכויות שמורות לפרופסור עמוס ישראלי

חתך בגרף

יהיה $G=(V,E)$ גרף לא מכוון ויהיו V_1 ו-V_2 שתי תת קבוצות של V המקיימות: $V_1\cup V_2=V$.

ו- $V_1\cap V_2=\Theta$.

נסמן ב $(V_1:V_2)$ את קבוצת הקשתות שצומת אחד שלהן שייך לקבוצה V_1 והצומת השני שייל ל- V_2.

$$(V_1:V_2)=\{(v_1,v_2)\mid v_1\in V_1, v_1\in V_1\}$$

שימו לב: אם נסיר מ-G את כל הקשתות בחתך $(V_1:V_2)$ הגרף יאבד את הקשירות שלו.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

החתך הנקבע על ידי עץ פורש וקשת

יהי $G=(V,E)$ גרף קשיר ויהי $T=(V,E')$ עץ פורש של הגרף G.

אם נסיר מ-T קשת כלשהי $e'\in E'$, העץ T יתחלק לשני תת עצים $T_1=(V_1,E'_1)$ ו-$T_2=(V_2,E'_2)$ וברור כי $V_1\cup V_2=V$ ו-$V_1\cap V_2=\Theta$, כלומר $(V_1:V_2)$ הוא חתך בגרף G.

החתך הזה מכונה: **החתך הנקבע על ידי העץ T והקשת** e'.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

<u>שני משפטים פשוטים</u>

<u>משפט 1 (לא שייך?)</u>

יהי $G=(V,E)$ גרף קשיר ויהי $T=(V,E')$ עץ פורש של הגרף G.

תהי $e_0 \in E'$ קשת כלשהי של העץ T,

יהי $S=(V_1:V_2)$, החתך הנקבע על ידי T ו-e, ותהי $e_0' \in S$, קשת כלשהי בחתך S.

---- יש להוסיף ציור ----

יהי $T'=T-e_0+e_0'$.

במצב זה, הגרף T' הוא עץ פורש של הגרף G.

<u>הוכחה</u>

מיידית.

<u>משפט2 (שייך?)</u>

יהי $G=(V,E)$ גרף קשיר ולא מכוון ויהי $T=(V,E')$ עץ פורש של G.

תהי E^+ קבוצת הקשתות המתקבלת מ-E על ידי הוספת קשת (x,y) אזי:

1. קבוצת הקשתות E^+ מכילה מעגל.
2. תהי E', קבוצת הקשתות המתקבלת מ-E^+ על ידי הורדת קשת (v,u) הנמצאת על המעגל שנסגר על ידי הקשת (x,y). הגרף $T'=(V,E')$ הוא עץ פורש של G.

<u>הוכחה</u>

העץ T פורש את הגרף G ולכן, ב T יש מסלול, α, מן הצומת x אל הצומת y. יהי $T^+=(V,E^+)$ הגרף המתקבל מהוספת הקשת (x,y) לעץ T. בגרף T^+ יש **מעגל** המכיל את (x,y) ואת קשתות המסלול α.

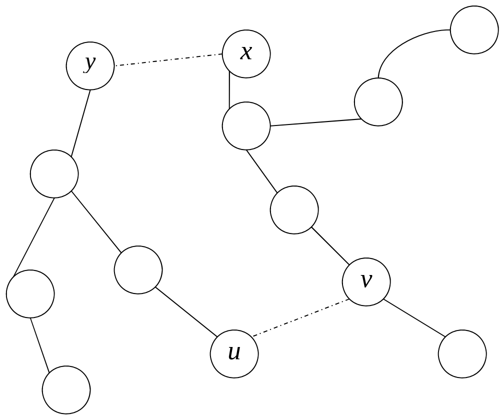
אם נוריד כעת מT^+, אחת מקשתות המעגל, נקבל גרף $T'=(V,E')$. הגרף T' הוא **קשיר**, **חסר מעגלים** והוא מכיל את כל הצמתים בG.

מסקנה: T' הוא **עץ פורש** של G.

מש"ל

<u>דוגמא</u>

העצים T ו - T' מודגמים בציור הבא:



עץ פורש מינימום

נתון גרף G קשיר בו לכל קשת e יש משקל $w(e)$. נגדיר:

1. **משקל של עץ פורש** שווה לסכום משקלי קשתות העץ. כלומר $W(T)=\sum_{e\in E'} w(e)$
2. יהי T עץ פורש של G. העץ T הוא **עץ פורש מינימלי (MST)** **Minimum Spanning Tree** של G, אם עבור כל עץ פורש אחר של G, T_1, מתקיים: $W(T_1)\geq W(T)$

שימו לב: יתכנו מספר עצים פורשים שמשקלם מינימלי.

מוטיבציה

אם משקל כל קשת מעיד על עלות השימוש בקשת, הרי שעץ פורש מינימום מגדיר קבוצה חסרת יתרות שמשקלה מינימלי הפורשת את G. בדרך זו אפשר לקבוע קבוצת קווי תקשורת או קבוצת דרכים המאפשרים/ות תקשורת בין כל חלקי המערכת.

דוגמא

נתונות n ערים וביניהן רשת דרכים, לכל דרך אורך מסוים. רוצים לבחור קבוצת דרכים בעלת אורך כולל מינימלי, אשר מבטיחה קישוריות בין כל שתי ערים נתונות.

הפתרון

נבחר קבוצת דרכים המהווה **עץ פורש מינימום** ב**גרף המושקל** הבא:
צמתי הגרף מייצגים **ערים**.
בין כל שתי **ערים** המחוברות ב**דרך**, נמתח **קשת** שמשקלה שווה ל**אורך הדרך** בין הערים.

אלגוריתמים לחישוב MST

נדון כעת בבעיה החישובית הבאה:
בהינתן גרף $G(V,E)$ חשב עץ פורש מינימום (MST) של G.
בקורס זה, נציג שני **אלגוריתמים חמדניים (Greedy Algorithms)** המקבלים כקלט גרף לא מכוון וקשיר $G=(V,E)$ ומחשבים עץ פורש מינימום של גרף הקלט G. בהרצאה זו נציג את אלגוריתם Prim ובהרצאה הבאה נציג את אלגוריתם Kruskal.

אלגוריתמים חמדניים - תזכורת

החמדנות היא שיטה (Paradigm) חשובה בתורת האלגוריתמים.
באופן כללי ביותר, אלגוריתם חמדני (Greedy Algorithm) הוא אלגוריתם שבו מגדירים לכל צעד של האלגוריתם פונקציית עלות. בכל שלב, האלגוריתם בוחר בצעד שעלותו **מינימלית**.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

אלגוריתם Prim – תאור כללי

אלגוריתם Prim בונה תת-עץ של גרף הקלט G. $T_1=(V_1,E_1)$. בכל שלב של האלגוריתם, נסמן ב-V_2 את קבוצת הצמתים שעדיין לא נכללים ב-T_1. ($V_2=V-V_1$). נתבונן בחתך: $E_{12}=(V_1:V_2)$, המכיל את כל הקשתות המחברות בין צמתי העץ T_1 לבין שאר צמתי הגרף. ברור כי לפחות קשת אחת מקבוצה זו נמצאת בכל עץ פורש של הגרף.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

אלגוריתם Prim – תאור כללי

תת העץ T_1 נבנה ב-$|V|-1$ שלבים: בכל שלב, מוסיפים לעץ T_1 קשת יחידה, (v_1,v_2), כך ש $v_1\in V_1$, ו- $v_2\in V_2$.
לאחר מכן גורעים את הצומת v_2, מן הקבוצה V_2, ומוסיפים אותו לקבוצה V_1.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

פרוט האסטרטגיה החמדנית

איזה קשת מן החתך E_{12} נצרף ל T_1?
לפי השיטה החמדנית עלינו לצרף לעץ הפורש קשת שתמזער את "הפסדינו" המיידיים.
אנו נצרף לעץ שאנו בונים את אחת הקשתות שמשקלן מינימלי מבין הקשתות בקבוצה E_{12}.

הערה

כמובן שיהיה עלינו להוכיח כי העץ שהאלגוריתם בונה הוא עץ פורש מינימום של גרף הקלט.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

אלגוריתם Prim – מבני נתונים

בתחילת ביצוע האלגוריתם, קבוצת הצמתים V_1 תכיל צומת שרירותי u, ממנו מתחילה בניית העץ, קבוצת הקשתות E_1 תאותחל כריקה.

בנוסף לקבוצות צמתים אלה נשתמש בקבוצת הצמתים V_2. בכל שלב בביצוע, הקבוצה V_2 תכיל את צמתי הגרף שלא מוכלים (עדיין) ב-V_1.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

אלגוריתם Prim - הקוד

$V_1 \Leftarrow u$ /* u צומת שרירותי */

$V_2 \Leftarrow V - V_1$

$E_1 \Leftarrow \phi$

do $(|V|-1 \text{ times})$

$E_{12} \Leftarrow (V_1 : V_2)$

$(v_1, v_2) \leftarrow$ קשת מינימלית ב-E_{12}

$V_1 \Leftarrow V_1 + v_2,\ V_2 \Leftarrow V_2 - v_2$

$E_1 \Leftarrow E_1 + (v_1, v_2)$

while $(V_1 \neq V)$

out(T_1)

© כל הזכויות שמורות לפרופסור עמוס ישראלי

אלגוריתם Prim - הוכחת נכונות

משפט 2

בכל שלב בביצוע אלגוריתם Prim העץ $T_1 = (V_1, E_1)$ מוכל בתת עץ של MST של G.

הסבר:

המשמעות של המשפט הזה היא שבכל שלב של ביצוע האלגוריתם העץ T_1 הוא **תת עץ של עץ פורש מינימום של גרף הקלט.**

אין זה אומר שהעפ״מ הזה מוכר לנו. בהוכחת המשפט, אנו יכולים להסתמך על העובדה שהעפ״ם הזה **קיים.**

© כל הזכויות שמורות לפרופסור עמוס ישראלי

הוכחה

באינדוקציה על מספר הצמתים בגרף:

שימו לב:

אנו מוכיחים את המשפט בשיטת השמורה. נקודת השמורה היא **מיד בתחילת הלולאה.**

בסיס: $|V_1| = 1$

מייד לאחר האתחול, הגרף $T_1 = (V_1, E_1)$ מכיל צומת (שרירותי) יחיד של הגרף G ואין בו אף קשת. מאחר וכל עץ פורש מינימום מכיל את כל הצמתים ב-G, הטענה נובעת.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

<u>צעד האינדוקציה:</u>

נניח כי לאחר הוספת k צמתים ל-V_1 ו-k קשתות לE_1, העץ $T_1=(V_1,E_1)$ הוא תת גרף של MST של גרף הקלט $G=(V,E)$. נכנה עפ"מ זה $T_{min}=(V,E_{min})$.

תהי $e_{min} = (u,v)$ הקשת המינימלית (או אחת הקשתות המינימליות) בקבוצה E_{12}. לפי האלגוריתם, (u,v) היא הקשת הבאה המוספת ל T_1. אם $(u,v)\in T_{min}$, אזי $T_1+(u,v)\subseteq T_{min}$ והמשפט נכון.

נכונות המשפט נובעת מן הטענה הבאה:

© כל הזכויות שמורות לפרופסור עמוס ישראלי

<u>טענה</u>

אם $(u,v)\notin T_{min}$ אזי קיים עץ פורש מינימום **אחר** של G, T'_{min} ומתקיים: $T_1+(u,v)\subseteq T'_{min}$.

<u>הוכחת הטענה</u>

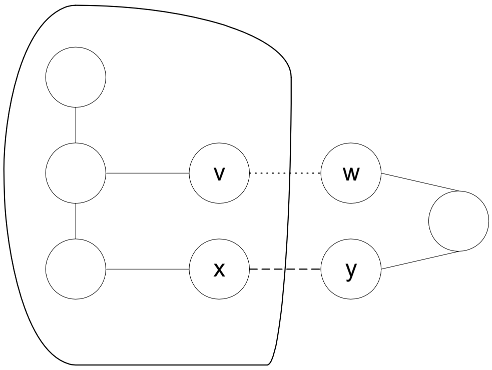
בעץ T_{min} יש מסלול פשוט α בין u לבין v (כי T_{min} הוא עץ פורש). תהי (x,y) הקשת היחידה הנמצאת על α אשר מקיימת: $x\in V_1$ ו- $y\in V-V_1$.

תהי $E'_{min}=E_{min}-(x,y)+(u,v)$.

מן המשפט שהוכחנו קודם, נובע כי הגרף $T'_{min}=(V,E'_{min})$, הוא עץ הפורש את גרף הקלט $G(V,E)$.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

<u>הוכחה (המשך)</u>



<u>הערה:</u> באיור זה, יש לשנות את שמות הצמתים (v,w) ל- (u,v).

© כל הזכויות שמורות לפרופסור עמוס ישראלי

<u>הוכחה (המשך)</u>

הקשת (u,v) נבחרה על ידי האלגוריתם לפני הקשת (x,y) ומכאן נובע כי $w(u,v)\le w(x,y)$.

מעובדה זו נובע כי

$$w(T'_{min})=w(T_{min})+w(u,v)-w(x,y)\le \\ \le w(T_{min})$$

מצד שני T_{min} הוא MST של G ומכאן: $w(T_{min})\le w(T'_{min})$. משני אי השוויונים הללו נובע כי $w(T_{min})=w(T'_{min})$.

מש"ל

© כל הזכויות שמורות לפרופסור עמוס ישראלי

מימוש האלגוריתם - הרעיון

לאחר האתחול מתבצעות $|V|-1$ איטרציות. בכל איטרציה, מוסיפים לעץ הפורש קשת מינימלית, בחתך E_{12} ביחד עם הצומת שלה.
משימתנו היא לזהות אחת מן הקשתות המינימליות בחתך $(V_1:V_2)$, לצרף אותה ואת הצומת בקצה השני לעץ T_1, ולבצע כל זאת **באופן יעיל**.
מספר הקשתות המירבי ב-E_{12}, תלוי במספר הצמתים ב V_1 וב V_2 .
המקרה הגרוע ביותר מתקיים כאשר $|V_1|=|V_2|=n/2$ ובמקרה זה מספר הקשתות האפשרי הוא $\Theta(n^2)$.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

צמצום קבוצת הקשתות E_{12}

אנו רוצים לזהות קשת מינימלית בקבוצה E_{12} ולצרפה לעץ הפורש T_1, בזמן $\Theta(n)$, כדי לקבל זמן ריצה כולל של $\Theta(n^2)$.
כדי לעשות זאת, נשמור רק חלק מן הקשתות בE_{12} תוך הבטחת זיהוי נכון של הקשת המינימלית:
לכל צומת $v \in V_2$, נגדיר את $min(v)$ בתור אחת הקשתות ה**קלות ביותר** הנוגעת בצומת כלשהו בקבוצה V_1.
$min(v) = \min_{u \in V_1}\{w(u,v)\}$. אם אין קשת בין הצומת v לבין אחד הצמתים ב-V_1, הקשת $min(v)$ אינה מוגדרת.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

הגדרת קבוצת הקשתות ME_{12}

בכל איטרציה, אנו נשמור אך ורק את הקבוצה ME_{12} המכילה עבור כל צומת $v \in V_2$ את $min(v)$, שהיא קשת מינימלית בקבוצה E_{12} הנוגעת ב-v, כלומר:

$$ME_{12} = \{\min(v) \mid v \in V_2\}$$

מספר הקשתות ב ME_{12} קטן או שווה מ $|V|-1$.לכן אפשר לאחסן אותה ולטפל בה בסיבוכיות $O(|V|)$.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

מימוש האלגוריתם - מבני נתונים

גרף הקלט יאוחסן בשיטת רשימת השכנויות: צמתי הגרף מאוחסנים במערך צמתים כל שלכל i, $i = 1,2,...,n$, הצומת v_i מאוחסן במקום ה-i במערך הצמתים. אנו מניחים כי לפני תחילת ריצת האלגוריתם, גרף הקלט מאוחסן בזכרון המחשב.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

מימוש האלגוריתם - האתחול

בתחילת ביצוע האלגוריתם, הקבוצה V_1 מכילה צומת יחיד, u, שאר הצמתים נמצאים בV_2. במצב זה, יש לקבוע עבור כל צומת $v_i \in V_2$, את הקשת $min(v_i)$.

בשלב האתחול, קשת כזו מוגדרת רק עבור השכנים של הצומת u. כדי לזהות קשתות אלה, עוברים על כל הקשתות הנוגעות בצומת u, $adj(u)$, ועבור כל קשת $e=(u,v)$ מגדירים $min(v)=(u,v)$.

שימו לב: אם בגרף הקלט אין קשת (u,v), אז בשלב האתחול, הקשת $min(v)$, אינה מוגדרת.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

קוד האיתחול

להלן מוצג קוד שלב האתחול:

$\Pr im - Initialize(V, E, w)$

$V_1 \Leftarrow \{u\}$ /* u שרירותי */

$V_2 \Leftarrow V - V_1$

$E_1 \Leftarrow \phi$

for each $(u,v) \in adj(u)$

  $min(v) \leftarrow (u,v)$

end for

סיבוכיות האתחול היא$\Theta(|V|)$.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

מימוש האלגוריתם - האיטרציות

בכל איטרציה של האלגוריתם, מוסיפים לעץ T_1 קשת מינימלית בין V_1 לבין V_2. כדי לזהות את הקשת המינימלית, e_{min}, עוברים על פני כל הצמתים ב-V_2, ובוחרים מביניהם את הצומת המחובר בקשת מינימלית אל אחד הצמתים ב-V_1, כלומר את הצומת w המקיים:

$$(x,w) \leftarrow \min_{v \in V_2} min(v)$$

לאחר זיהוי הצומת w, מעבירים אותו מ-V_2 אל V_1 ומוסיפים את הקשת (x,w) ל-E_1 . עלות כל איטרציה היא $\Theta(|V|)$

© כל הזכויות שמורות לפרופסור עמוס ישראלי

מימוש האלגוריתם - העדכון

לאחר מעבר הצומת w מ-V_2 אל V_1, צריך לעדכן את ערכי $min(v)$ לכל $v \in V_2$. מאחר שהצומת w, נוסף לקבוצה V_1, עלינו להתיחס לכל הקשתות הנוגעות בצומת w אשר הצומת בצידן השני נמצא בקבוצה V_2.

שימו לב

אין צורך להתיחס לצמתים נוספים (נמקו מדוע).

© כל הזכויות שמורות לפרופסור עמוס ישראלי

מימוש האלגוריתם – העדכון (המשך)

העדכון מתבצע כך:

לכל קשת $(y,w)\in adj(w)$, בודקים:

1. האם הצומת $y\in V_2$
2. האם מתקיים $w(min(y)) > w(y,w)$.

אם שתי התשובות חיוביות, מצאנו קשת קלה יותר המחברת את הצומת y עם V_1 ומבצעים:

$$min(y) \leftarrow (y,w)$$

© כל הזכויות שמורות לפרופסור עמוס ישראלי

מימוש האלגוריתם – העדכון (המשך)

שימו לב: אין צורך לעדכן את $min(w)$, כי בשלב זה מתקיים $w\in V_1$, ולכן הצומת w לא ילקח יותר בחשבון בחיפוש אחר הקשת המינימלית באיטרציות הבאות.

עלות הבדיקה היא

$$\Theta(\deg(w)) = O(|V|)$$

© כל הזכויות שמורות לפרופסור עמוס ישראלי

קוד האיטרציות

$prim_iterations(V,E,w)$

repeat $(|V|-1$ times$)$

$(x,w) \leftarrow \min_{v\in V_2} min(v)$

$V_1 \Leftarrow V_1 + w;\ V_2 \Leftarrow V_2 - w$

$E_1 \Leftarrow E_1 + (x,w)$

for each $(w,v)\in adj(w)$

if $v\in V_2$ **and** $min(v) > (w,v)$

$\min(v) \leftarrow (v,w)$

until $V_1 = V$

© כל הזכויות שמורות לפרופסור עמוס ישראלי

הסיבוכיות

שלב האיתחול כמו גם כל אחת מן האיטרציות דורשים $\Theta(|V|)$ צעדים.

מאחר שיש $|V|-1$ איטרציות סיבוכיות האלגוריתם היא $\Theta(|V|^2)$.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

<u>סיכום</u>

בהרצאה זו, הגדרנו את המושגים **עץ פורש ועץ פורש מינימום** והצגנו את אלגוריתם Prim המקבל כקלט גרף ממושקל $G(V,E,w)$ ומחשב עץ פורש מינימום של גרף הקלט.

בסיום ההרצאה, הראנו איך מממשים את אלגוריתם Prim בסיבוכיות $\Theta\left(|V|^2\right)$.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

הרצאה 7: עץ פורש מינימום בגרף

בהרצאה זו, נדון בהבטים שונים של בעית חישוב MST בגרף ממושקל:
בתחילת ההרצאה, נוכיח כי סיבוכיות הבעיה חסומה מלמטה על ידי מספר הקשתות בגרף $\Omega(|E|)$.
לאחר מכן נציג את אלגוריתם Kruskal.
לסיום ההרצאה, נציג ונפתור את בעיית Union/Find ונראה כי סיבוכיות אלגוריתם Kruskal המשתמש בפתרון Union/Find היא $\Theta(|E|\log|E|)$.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

חסם תחתון לחישוב MST

משפט 7.1

כל אלגוריתם לחישוב MST **חייב** לבחון משקל כל אחת מקשתות הגרף.

הסבר

כמו בכל חסם תחתון, משפט זה הוא טענה על **כל האלגוריתמים האפשריים** לחישוב MST. מסיבה זו, הוכחת המשפט צריכה להניח קיום של אלגוריתם **מופשט** A אשר אינו מקיים את הטענה **ולהגיע לסתירה.**

© כל הזכויות שמורות לפרופסור עמוס ישראלי

פרוט הנחת השלילה

עלינו להניח כי:

1. קיים גרף קלט ממושקל $G=(V,E,w)$.
2. קיימת קשת $e_0\in E$.
3. כאשר האלגוריתם A, מופעל על הגרף G, A מחשב MST של G, בלי שייבחן את הקשת e_0.

באופן אינטואיטיבי, נרצה להניח כי הקשת e_0, היא קשת שמשקלה מינימלי בגרף G, אך בכך נפר את כלליות הנחת השלילה, ולכן הנחה זו אינה קבילה. נראה כעת איך משתמשים בהנחה הכללית ומגיעים לסתירה.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

הוכחה

נניח כי קיים אלגוריתם A, גרף ממושקל קשיר ולא מכוון $G=(V,E,w)$ וקשת $e_0\in E$, כך ש-A מחשב MST של G בלי שיבחן את e_0.
יהי T, העץ פורש מינימום של G, אשר A מייצר. האלגוריתם A אינו בוחן את הקשת e_0. בפרט, הקשת e_0 אינה נכללת בעץ T.
להלן נגדיר את הגרף G' ונוכיח כי האלגוריתם A שוגה בביצוע על הגרף שהגדרנו.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

הגדרת הגרף G'

נתבונן בגרף $G'=(V,E,W')$ אשר זהה לגרף G פרט לפונקצית המשקל w' המוגדרת כך:

$$w'(e)=\begin{cases} w(e) & e\neq e_0 \\ \min_{e\in E} w(e)-1 & e=e_0 \end{cases}$$

הפונקציה w' זהה לפונקציה w עבור כל הקשתות בגרף, למעט הקשת e_0, עליה w' מקבלת ערך מינימלי בגרף G'.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

פרוט הסתירה

יהי T' העץ הפורש של G' אשר A מייצר. האלגוריתם A המופעל על G' מתנהג בדיוק כפי ש-A מתנהג על G, כי A אינו בוחן את הקשת e_0, ו-e_0 מהווה את **ההבדל היחיד** בין G לבין G'.

מכאן, $T'=T$ ובפרט, $e_0 \notin T'$.

מצד שני, e_0 היא קשת **מינימלית יחידה** ב-G' ולכן היא נכללת בכל עץ פורש מינימום של G'. סתירה.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

פרוט הסתירה

הסתירה נובעת מהנחתנו כי האלגוריתם A לא בוחן את הקשת e_0. מאחר שהקשת הזו נבחרה באופן שרירותי, נובע כי לא קיים אלגוריתם לחישוב MST אשר אינו בוחן את כל קשתות גרף הקלט, או במלים אחרות: סיבוכיות **כל אלגוריתם** לחישוב MST בגרף הקלט בוחן את כל קשתות גרף הקלט.

מש״ל

מסקנה

סיבוכיות כל אלגוריתם לחישוב MST חסומה מלמטה ע״י $\Omega(|E|)$.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

דיון

לכאורה, החסם התחתון שהצגנו מעיד על כך שלא יתכן שיפור לאלגוריתם Prim שכן במקרה הגרוע ביותר, בגרף **צפוף**, מתקיים $E=\Theta(|V|^2)$. למעשה, הטיעון הזה אכן מוכיח כי עבור גרפים **צפופים** אלגוריתם Prim הוא אופטימלי. יחד עם זאת, כדאי לחפש אלגוריתמים המשפרים את סיבוכיות אלגוריתם Prim בגרפים בהם מתקיים $|E|<\Theta(|V|^2)$.

להלן נציג את אלגוריתם Kruskal אשר משיג סיבוכיות $\Theta(|E|\log|E|)$ לחישוב MST.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

אלגוריתם Kruskal - תיאור כללי

1. צרף את כל צמתי הגרף אל ה-MST.

שימו לב: בשלב זה אנו מתייחסים אל הMST כגרף בעל $|V|$ רכיבי קשירות. בכל רכיב יש צומת יחיד.

2. עבור על קשתות הגרף בסדר משקלים עולה.

לכל קשת: אם הקשת מחברת שני רכיבי קשירות **שונים** של ה-MST , צרף את הקשת אל ה-MST.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

אלגוריתם Kruskal - הקוד

```
Kruskal(V,E,w)
  V_out ⇐ V;
  E_out ⇐ φ
  SE ⇐ Sort(E);
  for i ← 1 to |SE|
    if T_out + SE[i] has no cycle
       E_out ← E_out + SE[i]
    end-if
    if (|E_out| = |V| − 1)
         return(T_out)
    end-if
  end-for
end (Kruskal)
```

אלגוריתם 7.2: אלגוריתם Kruskal

© כל הזכויות שמורות לפרופסור עמוס ישראלי

שימו לב

התנאי $T_{out} + SE[i]$ has no cycle המופיע בקוד, שקול לדרישה שהקשת $SE[i]$ מחברת שני רכיבי קשירות שונים של הגרף T_{out}.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

נכונות האלגוריתם

משפט הנכונות של אלגוריתם Kruskal זהה למשפט הנכונות של אלגוריתם Prim:

משפט 7.3

בכל שלב בביצוע אלגוריתם Kruskal, הגרף T_{out} הוא תת גרף של MST של גרף הקלט.

שימו לב

משפט 7.3 מהווה שמורה, המתקיימת בכל מעבר בתחילת לולאת ה-for. אנו נוכיח את המשפט באינדוקציה על המעברים בלולאת האלגוריתם. כדי למנוע סרבול, לא נתייחס למעברים בלולאה, בהם העץ T_{out} לא משתנה.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

<u>הוכחה</u>

באינדוקציה על מספר המעברים בלולאה הראשית באלגוריתם (שוה למספר הקשתות שצורפו עד כה לעץ הנבנה):

<u>בסיס -</u>

עלינו להוכיח כי לפני ביצוע המעבר הראשון בלולאת האלגוריתם, הגרף T_{out}, המכיל את כל צמתי G, ולא מכיל אף קשת מ-G הוא תת גרף של MST של גרף הקלט G. הטענה נובעת מהגדרת MST כתת גרף המכיל את כל צמתי הגרף.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

<u>צעד האינדוקציה -</u>

נניח כי לאחר הוספת k קשתות, הגרף $T_{out}=(V,E_{out})$ הוא תת גרף של MST של גרף הקלט $G=(V,E)$.

יהי $T_{min}=(V,E_{min})$ ה-MST המכיל את T_{out}.

תהי e הקשת הבאה המוספת ל T_{out}.

אם $e\in E_{min}$, אזי $T_{out}+e\subseteq T_{min}$ והמשפט נכון. נניח אם כן כי $e\notin E_{min}$, נתבונן ב- $T_{min}+e$, הגרף הנוצר מצרוף הקשת e לעץ הפורש T_{min}. בגרף הזה יש מעגל σ. תהי $e'\neq e$ קשת ב-σ המוכלת ב-T_{min} אך לא מוכלת ב-T_{out}.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

<u>צעד האינדוקציה (המשך)</u>

תמיד יש קשת כזאת כי כל קשתות σ פרט לקשת e, כבר נמצאות ב- T_{out} האלגוריתם לא היה מצרף את הקשת e ל- T_{out}. נתבונן בגרף T'_{min}, הגרף המתקבל מצרוף e והוצאת e' מן העץ T_{min}. לפי **משפט 6.4** הגרף T'_{min} הוא עץ הפורש את G.

נראה כעת כי $w(T'_{min})\le w(T_{min})$

בזמן שהאלגוריתם מצרף את הקשת e ל-T_{out}, האלגוריתם יכול לצרף גם את e' ל-T_{out}. מכן נובע כי $w(e')\ge w(e)$.

ומכאן,

© כל הזכויות שמורות לפרופסור עמוס ישראלי

$$w(T'_{min}) =:$$
$$w(T_{min})+w(e)-w(e')\le w(T_{min})$$

מצד שני, T_{min} הוא MST של הגרף G, ולכן $w(T'_{min})\ge w(T_{min})$ ומכאן $w(T'_{min})=w(T_{min})$, ו-$w(e')=w(e)$

כלומר, העץ T'_{min} הוא MST נוסף של גרף הקלט G.

מש״ל

© כל הזכויות שמורות לפרופסור עמוס ישראלי

<u>מוטיבציה לבעיית Union/Find</u>

האתגר העיקרי שאלגוריתם Kruskal מציב בפנינו הוא: כיצד לבדוק ביעילות האם קשת נתונה (u,v) מחברת שני רכיבי קשירות שונים בגרף T_{out}, או לחילופין, האם הקשת (u,v) **<u>סוגרת מעגל</u>** בגרף T_{out}.

להלן נתמודד עם אתגר זה בשלבים הבאים:

1. נציג בעיה חישובית חדשה ושמה Union/Find.
2. נציג מחדש את אלגוריתם Kruskal תוך שימוש בפתרון הבעיה, כדי לקבל סיבוכיות משופרת.
3. נציג פתרון יעיל לבעיה שהצגנו.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

<u>בעיית Union/Find (U/F)</u>

בבעיה זו נתונים n איברים נבדלים הנמצאים בתחילה ב-n קבוצות שונות.
בבעיה זו נדרש מימוש יעיל ככל האפשר לשתי השיטות הבאות:

1. השיטה $Find$ המקבלת כקלט אחד האיברים **<u>ומזהה את הקבוצה</u>** שבה נמצא האיבר בקלט.
2. השיטה $Union$ המקבלת כקלט שתי קבוצות **<u>ומאחדת את הקבוצות שבקלט לקבוצה אחת</u>**

© כל הזכויות שמורות לפרופסור עמוס ישראלי

<u>שימו לב</u>

נהוג להסתכל על מימוש השיטות $Union$ ו-$Find$ כבעיה חישובית אחת ושמה $Union / Find$ או בקיצור U/F .

© כל הזכויות שמורות לפרופסור עמוס ישראלי

<u>הצגה מחודשת של אלגוריתם Kruskal תוך שימוש בשיטות *Union* ו-*Find*</u>

אלגוריתם Kruskal בונה MST של גרף הקלט, T_{out}. העץ T_{out}, נבנה בתהליך המתחיל מצרוף צמתי הגרף ל-n רכיבי קשירות.

לאחר מכן, קשתות העץ נבחנות בסדר משקלים עולה ועבור כל קשת e, האלגוריתם מחליט האם לצרף את הקשת e לעץ T_{out}.

כדי להשתמש בשיטות $Union$ ו-$Find$ במימוש אלגוריתם Kruskal, נתבונן בכל רכיב קשירות **<u>כקבוצת צמתים.</u>**

© כל הזכויות שמורות לפרופסור עמוס ישראלי

הצגה מחודשת של אלגוריתם Kruskal

תוך שימוש בשיטות Union ו-Find

נתבונן מקרוב בשלב צירוף בתהליך המתבצע במעבר על הקשתות הממוינות:

עבור כל קשת $e=(u,v)$ עלינו לבצע את שתי המשימות הבאות:

1. **לזהות (Find)** את רכיבי הקשירות של הצמתים u ו-v בגרף T_{out}.
2. אם הרכיב שבו נמצא הצומת u שונה מן הרכיב שבו נמצא הצומת v, האלגוריתם **מאחד (Union)** את שני הרכיבים הללו ומצרף את הקשת (u,v) לקבוצת הקשתות E_{out}.

מימוש Kruskal עם Union/Find

```
Kruskal(V, E, w)
 V_out ⇐ V; E_out ⇐ Θ
 SE ⇐ Sort(E)
 for i ← 1 to |E|
    (u,v) ← SE[i]
    fu ← Find(u)
    fv ← Find(v)
    if (fu ≠ fv)
        Union(fu, fv)
        E_out ← E_out + (u,v)
    endif
    if (|E_out| = |V| − 1)
        return(T_out)
    endif
 endfor
end (Kruskal)
```

אלגוריתם 7.4: מימוש אלגוריתם Kruskal באמצעות *Union* ו-*Find*

פתרון בעיית U/F

להלן נציג מימוש של השיטות *Union* ו-*Find*, עבור n צמתים. המימוש מאפשר טיפול ב- $n-1$ קריאות לשגרת *Union*, אשר ביניהן שזורות m קריאות לשגרת *Find* (m הוא מספר טבעי כלשהו. סיבוכיות המימוש שנציג היא $O(m+n\log n)$. כלומר, מספר הצעדים הכולל בביצוע כל הקריאות המתוארות חסום מלמעלה על ידי $O(m+n\log n)$.

שימו לב

קיימים מימושים יעילים יותר.

הצגת המימוש

נניח כי מספר האיברים המשתתפים בתהליך הוא n. איברי הקבוצה ייוצגו על ידי $1,2,...,n$.

ייצוג קבוצות

כל קבוצה תיקרא בשם של אחד מאיברי הקבוצה.

בתחילת התהליך, כל איבר יהיה בקבוצה נפרדת, כלומר האיבר i, $1\le i\le n$, יהיה האיבר היחיד בקבוצה ששמה יהיה גם הוא i.

הצגת המימוש (המשך)

המימוש שנציג משתמש בשלושה מערכים:

1. המערך $Comp$ - כאשר $Comp[i]$,
   $1 \le i \le n$, מאחסן את הקבוצה אליה משתייך האיבר i.
2. המערך $Size$ - כאשר $Size[i]$, מאחסן את גודל הקבוצה i.
3. המערך $Elts$ - מאחסן מצביע אל רשימת מקושרת של איברי הקבוצה i.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

הצגת המימוש (המשך)

שימו לב

שם המערך המייצג את השתייכות האיברים לקבוצות שונות נבחר ל-*Comp*, כי באלגוריתם Kruskal, כל קבוצה מייצגת רכיב קשירות (Component).

© כל הזכויות שמורות לפרופסור עמוס ישראלי

אתחול U/F

בתחילת התהליך, האיבר i, $1 \le i \le n$, הוא האיבר היחיד בקבוצה ששמה i.

אלגוריתם 7.5 מציג את אתחול המערכים:

$UFInit(n)$

for $i \leftarrow 1$ **to** n

$\quad Comp[i] \leftarrow i$

$\quad Size[i] \leftarrow 1$

$\quad Elts[i] \leftarrow insertList(i)$

end for

אלגוריתם 7.5: אתחול U/F

© כל הזכויות שמורות לפרופסור עמוס ישראלי

מימוש השיטה *Find*

מימוש *find* הוא מיידי ומוצג באלגוריתם 7.6:

$Find(i)$

return $comp(i)$

אלגוריתם 7.6: מימוש השיטה *Find*

© כל הזכויות שמורות לפרופסור עמוס ישראלי

מימוש השיטה *Union*

השיטה $Union(i,j)$ מאחדת את הקבוצה ששמה i עם הקבוצה ששמה הוא j. כדי לבצע זאת, מעבירים את איברי אחת הקבוצות אל הקבוצה השניה. כדי לבצע זאת **ביעילות** מעבירים את איברי הקבוצה שמספר איבריה **קטן יותר** אל הקבוצה השניה. אם בשתי הקבוצות יש אותו מספר איברים, מעבירים איברי אחת מהן אל השניה.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

מימוש *Union* - הקוד

$union(k,l)$
 If $Size[k] \ge Size[l]$
  $large \leftarrow k;\ small \leftarrow l$
 else
  $large \leftarrow l;\ small \leftarrow k$
 foreach $m \in Elts[small]$
  $Comp[m] \leftarrow large$
  $Elts[large] \leftarrow Elts[large] \circ m$
 end foreach
 $Size[large] \leftarrow$
  $Size[large] + Size[small]$
end

אלגוריתם 7.7: מימוש *Union*

© כל הזכויות שמורות לפרופסור עמוס ישראלי

שימו לב

אין צורך לעדכן את $size[small]$, כי הקבוצה $small$ לא קיימת יותר.

למה 7.8

במימוש זה, סיבוכיות ביצוע $Union$ שווה למספר האיברים בקבוצה הקטנה מבין השתיים.

הוכחה

מיידית.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

סיבוכיות $|V|$-1 פעולות *Union*

השימוש בשיטות U/F מתאר תהליך שבו n קבוצות מאוחדות לקבוצה אחת על ידי ביצוע $n-1$ קריאות לשגרת $Union$. סיבוכיות התהליך שווה לסכום מספר הצעדים המתבצע על ידי כל הקריאות. ברור שככל שהקבוצות המאוחדות גדולות יותר, מספר הצעדים באיחוד הקבוצות גדול יותר. להלן נציג שתי דוגמאות : בדומה הראשונה, סיבוכיות ביצוע $n-1$ היא לינארית. בדוגמה השניה, הסיבוכיות היא $\Theta(n\log n)$.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

דוגמאות

1. אם בכל פעולת איחוד קבוצות, מאחדים יחידון (קבוצה בעלת איבר יחיד) לתוך קבוצה i, כל איבר, פרט לאיבר יחיד, עובר מקבוצה לקבוצה **פעם אחת בלבד**. במצב זה, סיבוכיות כל התהליך היא $\Theta(|V|)$.
2. אם כל פעם מאחדים **שתי קבוצות שוות בגודלן**, כל איבר עובר בדיוק $\log|V|$ פעמים מקבוצה לקבוצה. במצב זה, סיבוכיות כל התהליך היא $\Theta(|V|\log|V|)$.

למה 7.9

כל איבר מועבר מקבוצה לקבוצה לכל היותר $O(\log n)$ פעמים.

הוכחת הלמה

יהי v איבר שרירותי. נוכיח באינדוקציה כי לאחר שהאיבר v עובר מקבוצה לקבוצה בפעם ה-i, מספר האיברים בקבוצה אליה האיבר v עבר הוא לפחות 2^i.

בסיס (i=0)

בתחילת ביצוע האלגוריתם, לפני המעבר הראשון, מספר האיברים בכל קבוצה, ובפרט בקבוצה בה נמצא האיבר v, הוא 1.

שלב האינדוקציה

נניח כי מייד לפני המעבר ה$i+1$, v נמצא בקבוצה שמספר איבריה הוא לפחות 2^i. לפי האלגוריתם, v מועבר אל קבוצה שגדלה **גדול או שווה** מן הגודל של הקבוצה של v לפני המעבר, כלומר מאחדים שתי קבוצות שגודל כל אחת מהן הוא לפחות 2^i. מכאן נובע: גודל הקבוצה המאוחדת **גדול או שווה** 2^{i+1}.

מש״ל

תוצאה 7.10

כל איבר עובר מקבוצה לקבוצה לכל היותר $\log n$ פעמים.

הוכחה

יהי v איבר שרירותי. לאחר ש-v עובר בפעם ה-n, מקבוצה לקבוצה, גודל הקבוצה של v **גדול או שווה** מ-$n = 2^{\log n}$. במצב זה, הקבוצה של v, כבר מכילה את **כל האיברים** והתהליך הסתיים.

הסיבוכיות הכוללת

למה 7.11

מספר הצעדים הכללי הנדרש לאיחוד כל הרכיבים במהלך אלגוריתם Kruskal הוא $\Theta(|V|\log|V|)$.

הוכחה

במימוש שהצגנו, כל רכיב מיוצג על ידי קבוצת הצמתים שלו. לפי **למה 2**,לכל צומת v, מספר הפעמים ש-v **עובר מרכיב לרכיב** חסום מלמעלה על ידי $\log|V|$. מאחר שיש $|V|$ צמתים, הטענה נובעת באופן מיידי.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

סיבוכיות אלגוריתם kruskal

לאלגוריתם Kruskal שלושה שלבים:

1. **איתחול** - הסיבוכיות היא $\Theta(|V|)$.
2. **מיון הקשתות** - הסיבוכיות היא $\Theta(|E|\log|E|)$.
3. **צרוף קשתות לעץ הפורש** - לשלב זה שני חלקים שצעדיהם שזורים זה בזה:

3.1 **מעבר על כל הקשתות** - הסיבוכיות היא $\Theta(|E|)$.

3.2 **איחוד רכיבים** – בכל פעם שאנו מוסיפים קשת ל-MST עלינו לאחד בין שני רכיבים.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

סיבוכיות אלגוריתם kruskal (המשך)

סיבוככיות המיון היא כאמור $\Theta(|E|\log|E|)$.

אם נשתמש באלגוריתם Union/Find לביצוע האיחודים, סיבוכיות האיחודים היא $O(|V|\log|V|)$ והסיבוכיות הכוללת של שלב המעבר על הקשתות היא $\Theta(|E|+|V|\log|V|)$.

מכאן נקבל כי סיבוכיות האלגוריתם כולו היא $\Theta(|E|\log|E|)$.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

סיכום הרצאה 7

בהרצאה זו, דנו בהבטים שונים של בעית חישוב MST בגרף ממושקל:

1. הוכחנו כי סיבוכיות הבעיה חסומה מלמטה על ידי מספר הקשתות בגרף $\Omega(|E|)$.
2. הצגנו את אלגוריתם Kruskal.

הצגנו את בעית Union/Find ופתרנו אותה בסיבוכיות $O(m+n\log n)$.

הפתרון מאפשר מימוש אלגוריתם Kruskal בסיבוכיות $\Theta(|E|\log|E|)$.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

<u>הרצאה 8 - חיפוש לעומק</u>

בהרצאה זו נציג את האלגוריתם **חיפוש לעומק** הקרוי באנגלית **Depth First Search (DFS)**.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

<u>מוטיבציה</u>

אלגוריתם DFS מהווה בסיס לפתרון כמה בעיות אלגוריתמיות בסיסיות, ביניהן:

1. זיהוי צמתי הפרדה ופירוק גרף הקלט לרכיבים אי פריקים (דו קשירים) (בגרף לא מכוון).
2. מיון טופולוגי בגרף מכוון.
3. זיהוי רכיבים קשירים היטב בגרף מכוון.

בהרצאה זו, נלמד את האלגוריתם ותכונותיו, ובהרצאה הבאה נלמד אלגוריתם לזיהוי צמתי הפרדה בגרף לא מכוון. האלגוריתם לזיהוי צמתי הפרדה מבוסס על DFS.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

<u>חיפוש לעומק -</u>
<u>Depth First Search (DFS)</u>

חיפוש לעומק, **Depth First Search**, הוא אלגוריתם המבצע סיור בגרף. אפשר לדמות את ביצוע האלגוריתם לתנועה של סוכן על פני גרף הקלט. הסוכן מתחיל את תנועתו מצומת שורש נתון, ונע מצומת לצומת. בכל שלב בביצוע, הסוכן מתקדם לעבר צמתים חדשים, ״עמוקים״ ככל האפשר, וזאת כל עוד יש צמתים כאלה. כאשר אין אפשרות להתקדם אל צומת חדש - הסוכן נסוג לאחור צעד יחיד, ומנסה לחדש את ההתקדמות. ההחלטה לאיזה שכן להתקדם מבוססת על **מידע מקומי** בלבד.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

<u>חיפוש לעומק - תאור</u>

הקלט לאלגוריתם הוא גרף $G=(V,E)$ וצומת $r\in V$. האלגוריתם נפתח בשגרת **אתחול** $DFSInit$ אשר דומה לשיטה $BFSInit$.

לאחר האתחול מתבצעת קריאה ראשונה לשיטה הרקורסיבית $DFSVisit(r)$. השיטה הזאת מבצעת את החיפוש עצמו.

לאחר שביצוע השיטה $DFSVisit(r)$ מסתיים, האלגוריתם בודק האם קיים צומת אליו הסיור לא הגיע (וזה ייתכן רק בגרף לא קשיר, או בגרף מכוון). אם קיים צומת כזה, נניח v, האלגוריתם קורא שוב לשיטה $DFSvisit(v)$.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

<u>אלגוריתם DFS - הקוד</u>

$DFS(G,r)$
  $DFSInit(G)$
  $DFSVisit(r)$
 while exists unvisited node v
   $DFSVisit(v)$
 End while
end (DFS)

<u>אלגוריתם 8.1: DFS</u>

© כל הזכויות שמורות לפרופסור עמוס ישראלי

<u>תויות זמן - Time Stamps</u>

במהלך פעולת אלגוריתם DFS על גרף כלשהו, אנו מתיחסים לכל קריאה לשיטה $DFSVisit$, ולכל חזרה משיטה זו כפעימה בשעון, המאותחל לזמן 0. לכל צומת $v \in V$, מרווח הפעולה בצומת v, הוא פרק ה״זמן״ בו מתבצעת השיטה $DFSVisit(v)$. כל צומת v מקבל שתי תוויות זמן (time stamps) המציינות את:

1. **<u>זמן הגילוי</u>** של v, $d[v]$, הוא ה״זמן״ היחסי בו התגלה הצומת v והטיפול בו **התחיל**.
2. **<u>זמן הנטישה</u>** של v, $f[v]$, הוא הזמן בו הטיפול בצומת v **נגמר**.

מספרן הכולל של התוויות הוא אם כן $2|V|$.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

<u>חיפוש לעומק – מבנה הנתונים</u>

עבור כל צומת, $v \in V$, המשתנים, $d[v]$, ו- $f[v]$, מאחסנים את תוויות הזמן של v. בנוסף, הצומת v משתמש במשתנה $color[v]$, שתפקידו דומה לתפקידו ב-BFS, ובמשתנה $\pi[v]$ המאחסן את הצומת ממנו ״התגלה״ הצומת v. המשתנה $time$ הוא משתנה גלובלי (סטטי) המסייע בקביעת ערכי תויות הזמן.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

<u>אתחול DFS</u>

השיטה $DFSInit(G)$ מופיעה להלן:

$DFSInit(G)$
for each $v \in V$
  $\pi[v] \leftarrow Null$
  $color[v] \leftarrow W$
$time \leftarrow 1$
end ($DFSInit$)

<u>אלגוריתם 8.2: השיטה *DFSInit*</u>

© כל הזכויות שמורות לפרופסור עמוס ישראלי

השיטה *DFSVisit* - תיאור

השיטה הרקורסיבית *DFGSVisit* מבצעת את הסיור בגרף. הסיור מתחיל על ידי הקריאה $DFSVisit(r)$. נתאר כעת את ביצוע השיטה *DFSVisit* על צומת כלשהו u.

כאשר הביצוע מתחיל, צבעו של v הוא לבן. בתחילת הביצוע, זמן הגילוי של u מתעדכן לפי ערכו של המשתנה *time*, ערכו של *time* מוגדל ב-1, וצבעו של u משתנה לאפור.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

תיאור השיטה *DFSVisit* (המשך)

לאחר מכן, האלגוריתם עובר על כל אחד משכניו של הצומת u: לכל צומת חדש v, שצבעו לבן, ערכו של המשתנה $\pi[v]$ נקבע ל-u. לאחר מכן מתבצעת הקריאה הרקורסיבית $DFSvisit(v)$.

מכאן נובע כי אם הקריאה לשיטה $DFSvisit(v)$, ארעה במהלך ביצוע השיטה $DFSvisit(u)$, אז $u = \pi[v]$.

הקשת $(\pi[v], v)$, מכוונת מ-$\pi[v]$ אל v, היא הקשת דרכה v **מתגלה** והיא גם הקשת דרכה מתבצעת **נסיגה** מן הצומת v לאחר סיום הסיור.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

השיטה *DFSVisit* - הקוד

```
DFSvisit(u)   /* התקדמות אל u */
  d[u] ← time++
  color[u] ← G
  for each v ∈ Adj[u] do
    if color[v] = W    /* v is new */
        π[v] ← u
        DFSvisit(v) /* התקדמות אל v */
      else   /* v is old - נסיגה מיידית מ v */
      endif
  end foreach
  f[u] ← time++
  color[u] ← B /* נסיגה מ u */
End /* DFS */
```

אלגוריתם 8.3: השיטה *DFSVisit*

© כל הזכויות שמורות לפרופסור עמוס ישראלי

חיפוש לעומק - סיבוכיות

לכל $v \in V$, השיטה $DFSvisit(v)$ מתבצעת **בדיוק פעם אחת** (הוכיחו זאת).

כמו כן, אפשר להראות כי האלגוריתם עובר על כל קשת בגרף **בדיוק פעמיים**.

לכן סיבוכיות הזמן הכוללת של DFS היא $\Theta(|E|+|V|)$.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

<u>תכונות חיפוש לעומק</u>

להלן נציג מספר תכונות של אלגוריתם DFS. נפתח בהגדרה כללית של עץ מושרש:

<u>הגדרה 8.4: עץ מושרש</u>

יהי $T=(V,E)$ עץ ויהיו u ו-v צמתים בעץ T. הצומת v הוא **צאצא** של הצומת u אם קיים ב-T **מסלול מכוון** מ-u אל v, במצב זה, הצומת u הוא **אב קדמון** של v.

יהי $r\in V$ צומת כלשהו ב-T. אם כל צומת $v\in V-r$, הוא צאצא של הצומת r, אז T הוא **עץ מושרש** והצומת r הוא **השורש** של T.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

יער מושרש הוא אוסף של עצים מושרשים.

<u>הגדרה 8.5: קשתות עץ</u>

נתבונן באוסף הקשתות המכוונות מן הצורה $(\pi[v],v)$. כל קשת כזו מתאימה לקריאה רקורסיבית של השיטה $DFSVisit(v)$. מכאן נובע כי אוסף הקשתות הזה משקף בדיוק את עץ הקריאות הרקורסיביות של השגרה $DFSVisit(r)$. כל קשת כזו נקראת **קשת עץ (tree-edge)** .

© כל הזכויות שמורות לפרופסור עמוס ישראלי

<u>הגדרה 8.6: יער הקודמים</u>

אוסף קשתות העץ יוצר משרה את **יער הקודמים** G_π. זהו יער שכל אחד מן העצים בו **מושרש.**

קבוצת הצמתים של G_π היא $V_\pi=V$.

קבוצת הקשתות של G_π היא

$$E_\pi=\{(\pi[v],v):v\in V\}$$

<u>שימו לב</u>

מבנה יער הקודמים תלוי כמובן במהנה גרף הקלט, אך הוא נקבע על ידי **ביצוע האלגוריתם**. למעשה מבנה יער הקודמים תלוי ב:

1. השורש r ממנו מתחיל הסיור.
2. סדר הצמתים ברשימת השכנויות של כל צומת.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

<u>תכונות היער</u> G_π

נדון כעת בתכונות היער G_π. מטרתנו היא לאפיין את הצאצאים ביער G_π של כל אחד מן הצמתים בגרף G.

<u>אבחנה 8.7</u>

יהי $G=(V,E)$ גרף קשיר ולא מכוון בו בוצע DFS. לכל צומת v מתקיים:

$$1\le d[v]< f[v]\le 2|V|$$

<u>הוכחה</u>

נובעת ישירות מן הקוד.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

<u>משפט 8.8: (תכונת הסוגריים)</u>

יהי $G=(V,E)$ גרף מכוון או בלתי מכוון. לכל ביצוע של DFS על G, נגדיר לכל צומת w את הקטע $(d[w],f[w])$ כ**מרווח הפעולה של w**.

יהיו u ו-v צמתים ב-G, בכל ביצוע של DFS על G יתקיים אחד משלושת התנאים הבאים:

1. מרווחי הפעולה של u ו - v זרים.
2. מרווח הפעולה של v מכיל את מרווח הפעולה של u.
3. מרווח הפעולה של u מכיל את מרווח הפעולה של v.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

<u>הסבר</u>

משמעות המשפט היא שלכל שני צמתים $u,v\in V$ מתקיימת אחת משתי האפשרויות הבאות:

1. אחד הצמתים (נניח u) מתגלה ונעזב לפני שהצומת השני מתגלה או
2. הצומת u מתגלה, אחריו מתגלה הצומת v, לאחר מכן v נעזב ולבסוף u נעזב.

במלים אחרות: לא יתכן שצומת u מתגלה, אחריו מתגלה v, לאחר מכן u נעזב, ולבסוף v נעזב.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

<u>הוכחת משפט 8.8</u>

בלי הגבלת כלליות, נניח כי $d[u]<d[v]$. ידוע גם (**אבחנה 8.8**) כי $d[u]<f[u]$ וכי $d[v]<f[v]$.

נתבונן ביחס הקדימות בין $f[u]$ לבין $d[v]$:

<u>מקרה א׳</u>:

$f[u]<d[v]$ במקרה זה, ברור כי $d[u]<f[u]<d[v]<f[v]$, כלומר מרווחי הפעולה הם זרים.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

<u>הוכחת משפט 8.8 (המשך)</u>

<u>מקרה ב׳</u>: $f[u]>d[v]$

במקרה זה, הצומת v מתגלה בין הרגע בו הצומת u מתגלה לבין הרגע בו הטיפול בצומת u נגמר. מצב זה ייתכן, רק אם הקריאה לשיטה $DFSVisit(v)$ בוצעה **<u>במהלך</u>** הביצוע של $DFSVisit(u)$, כלומר זוהי קריאה רקורסיבית. במקרה כזה ברור כי ביצוע $DFSVisit(v)$ **<u>יסתיים</u>** לפני ביצוע $DFSVisit(u)$. כלומר, מרווח הביצוע של u **<u>מכיל</u>** את מרווח הביצוע של v.

מש״ל

© כל הזכויות שמורות לפרופסור עמוס ישראלי

תוצאה 8.9

צומת v הוא צאצא של צומת u ביער הקודמים G_π, אם ורק אם $d[u] < d[v] < f[v] < f[u]$.

הוכחה

מיידית **ממשפט 8.8**.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

משפט 8.9 (משפט המסלול הלבן)

יהי $G=(V,E)$ גרף מכוון או לא מכוון עליו בוצע DFS. צומת v הוא צאצא של צומת u ביער G_π, אם ורק אם כאשר u מתגלה, בזמן $d[u]$, קיים בגרף הקלט G מסלול של צמתים **לבנים** מן הצומת u אל הצומת v.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

הוכחת משפט 8.9

כיוון ראשון:

נניח כי v הוא צאצא של u ביער הקודמים G_π, כלומר קיים ענף ב-G_π, $u=u_0,u_1,..,u_l=v$. במקרה זה, לכל $1\le i\le l$, הקריאה $DFSVisit(u_i)$, היא קריאה רקורסיבית, המתבצעת במהלך $DFSVisit(u_{i-1})$, ומכאן נובע כי

$$d[u]=d[u_0]<\ldots<d[u_l]=d[v]$$

ובפרט, בזמן $d[u]$, הצבע של $u_1,u_2,\ldots,u_l=v$, הוא לבן.

מש״ל

© כל הזכויות שמורות לפרופסור עמוס ישראלי

הוכחת משפט 8.9 (המשך)

כיוון שני:

נניח בשלילה כי בזמן $d[u]$ קיים מסלול לבן α מ-u אל v, אך v אינו צאצא של u ביער הקודמים G_π. נניח, בלי הגבלת הכלליות כי v הוא הצומת הראשון במסלול α שאינו צאצא של הצומת u, אחרת נבחר בתור v את הצומת הקרוב ביותר ל-u על המסלול α שאינו צאצא של u. יהי w הצומת הקודם ל-v במסלול α. (שימו לב: יתכן כי $w=u$).

© כל הזכויות שמורות לפרופסור עמוס ישראלי

הוכחת משפט 8.9 (המשך)

מתוצאה 8.7 נובע כי $f[w] \le f[u]$.
נשים לב לכך ש-v מתגלה רק לאחר ש-u מתגלה, אך לפני סיום הטיפול ב-w (כי v שכן של w). מכאן נובע כי

$$d[u] < d[v] < f[w] \le f[u]$$

ממשפט 8.8 נובע כי הקטע $(d[v], f[v])$ מוכל כולו בקטע $(d[u], f[u])$.
מתוצאה 8.7 נובע כי v הוא צאצא של u, בסתירה להנחתנו.

מש״ל

© כל הזכויות שמורות לפרופסור עמוס ישראלי

סוגי הקשתות – קשת אחורית

תהי $e = (u,v)$ קשת בגרף $G = (V,E)$ שבוצע בו DFS.
הקשת $e = (u,v)$ (מכוונת מ-u אל v) היא **קשת אחורית (back edge)** אם הצומת u הוא **צאצא** של הצומת v ביער הקודמים G_π. במקרה זה, צבעו של הצומת v בזמן המעבר הראשון על פני הקשת e הוא **אפור.**

© כל הזכויות שמורות לפרופסור עמוס ישראלי

תוצאה 8.10

יהי $G = (V,E)$ גרף לא מכוון וקשיר שבוצע בו חיפוש לעומק החל מצומת $r \in V$. אזי יער הקודמים G_π הוא **עץ מושרש בצומת** r.

הוכחה

הגרף G **הוא קשיר ולכן** לכל צומת $v \in V$, יש מסלול בין r ובין v.
בתחילת ביצוע האלגוריתם, כל הצמתים בגרף לבנים ולכן ממשפט המסלול הלבן נובע כי כל צמתי הגרף הם צאצאים של הצומת r ביער הקודמים G_π.

מש״ל

© כל הזכויות שמורות לפרופסור עמוס ישראלי

משפט 8.11

בחיפוש בעומק בגרף לא מכוון, כל קשת של הגרף היא קשת עץ או קשת אחורית.

הוכחה

תהי (u,v) קשת שרירותית ב G ונניח בלי הגבלת כלליות כי u מתגלה לפני v כלומר, $d[u] < d[v]$. מאחר שבמהלך $DFSVisit[u]$ האלגוריתם עובר על כל השכנים של u, אז ברור ש-v מתגלה ונעזב בטרם $DFSVisit[u]$ מסתיים.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

הוכחת משפט 8.11 (המשך)

נתבונן במקרים הבאים :

מקרה 1: $u = \pi[v]$

במקרה זה, (u,v) היא קשת עץ המכוונת מu אל v.

מקרה 2: $u \neq \pi[v]$

במקרה זה בטרם ניסוג מ-v עלינו לעבור בקשת (u,v). מעבר זה הוא **המעבר הראשון** ב-(u,v) וכיוונה יהיה מv אל u. מאחר שהנחנו $d[u] < d[v]$, נובע כי (u,v) היא קשת אחורית.

מש״ל

© כל הזכויות שמורות לפרופסור עמוס ישראלי

חיפוש לעומק בגרף מכוון

בנוסף לקשת עץ ולקשת אחורית שכבר הוגדרו, בחיפוש לעומק בגרף מכוון קיימים עוד שני סוגי קשתות המתאפיינים כשלהלן :

1. **קשת קדימה (forward edge):** אם הצומת u הוא **אב קדמון** של הצומת v ביער הקודמים G_Π.
2. **קשת חוצה (cross edge):** אם e אינה קשת מאחד הסוגים הקודמים.

****** להוסיף איורים *****

© כל הזכויות שמורות לפרופסור עמוס ישראלי

סיווג הקשתות במהלך האלגוריתם

במהלך ביצוע DFS אפשר לזהות את הסוג של כל קשת $e = (u,v)$ המכוונת מן הצומת u אל הצומת v לפי צבעו של v :

אם v הוא **לבן**, e היא **קשת עץ**.

אם v הוא **אפור**, e היא **קשת אחורית**.

אם v הוא **שחור**, e היא **קשת קדימה** או **קשת חוצה**.

במקרה האחרון, כדי לקבוע את סוגה של הקשת e יש לבדוק :

אם $f[v] > d[u]$ e היא **קשת קדימה.**

אם $f[v] < d[u]$ e היא **קשת חוצה.**

© כל הזכויות שמורות לפרופסור עמוס ישראלי

חיפוש לעומק - סיכום

בהרצאה זו, הגדרנו את בעית ה**סיור בגרף** והצגנו את אלגוריתם DFS .

© כל הזכויות שמורות לפרופסור עמוס ישראלי

הרצאה 9: אלגוריתם לזיהוי צמתי הפרדה ורכיבים אי פריקים

בהרצאה זו נמשיך את הדיון באלגוריתם חיפוש לעומק ובשימושיו.
עיקר ההרצאה יוקדש לאלגוריתם לזיהוי של צמתי הפרדה ורכיבים אי-פריקים בגרף לא מכוון.
בתחילת ההרצאה, נגדיר צמתי הפרדה ורכיבים אי-פריקים בגרף ונסביר את חשיבות המושגים הללו.
לאחר מכן, נוכיח מספר תכונות של צמתי הפרדה ונשתמש בתכונות אלה כדי לבנות אלגוריתם, המבוסס על חיפוש לעומק, לזיהוי צמתי הפרדה ורכיבים אי-פריקים.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

צמתי הפרדה בגרף

יהי $G=(V,E)$ גרף לא מכוון קשיר.
צומת $v \in V$ הוא **צומת הפרדה (באנגלית Separation vertex או Cut Point)**, אם הסרת v ביחד עם כל הקשתות הנוגעות בו הופכת את G לגרף **לא קשיר**.
כל צומת הפרדה הוא נקודה קריטית בגרף שכן הסרתו מפרה את הקשירות.
**** יש לצרף איור ***

© כל הזכויות שמורות לפרופסור עמוס ישראלי

צמתי הפרדה (המשך)

הגדרה שקולה: צומת v הוא **צומת הפרדה** בגרף לא מכוון $G=(V,E)$ אם קיימים שני צמתים $u, w \in V$ כך שכל מסלול בין u ל-w עובר דרך v.
יהי v צומת הפרדה. הצמתים u ו-v אשר כל מסלול ביניהם עובר בv נקראים **עדים** ל-v.

תרגיל

הוכיחו את שקילות שתי ההגדרות לצומת הפרדה.
הדרכה: יש להוכיח כי v מקיים את הגדרה 1 אסם v מקיים את הגדרה 2.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

רכיבים אי-פריקים בגרף

גרף G הוא **אי-פריק** אם אין בו צמתי הפרדה.
רכיב אי-פריק בגרף G הוא תת גרף **אי-פריק מכסימלי**. כלומר זהו תת גרף **אי-פריק** של G ואשר אינו מוכל **ממש** בשום תת גרף אי-פריק אחר של G.
שימו לב: אם v הוא צומת הפרדה אזי הוא מוכל בכמה רכיבים אי-פריקים (לפחות שניים) של G.
לעומת זאת כל קשת של G מוכלת בדיוק ברכיב אי-פריק יחיד.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

אלגוריתם לזיהוי צמתי הפרדה ורכיבים אי-פריקים

האלגוריתם שנציג, מושג על ידי **חיזוק** אלגוריתם DFS כדי לזהות את צמתי ההפרדה והרכיבים ה אי-פריקים בגרף לא מכוון נתון.

בטרם ניגש אל האלגוריתם נתבונן בכמה דוגמאות וננסה להסיק מסקנות לגבי היחסים בין מבנה עץ DFS בגרף G, לבין מיקומם של צמתי ההפרדה בגרף G.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

כללים לזיהוי צמתי הפרדה

בהנתן גרף G, אפשר לבצע בו DFS. מבנה העץ המתקבל תלוי בגרף הקלט G ובצומת ממנו החיפוש מתחיל.
הכללים הבאים ישמשו אותנו באלגוריתם לזיהוי צמתי הפרדה:

כלל השורש

יהי T עץ DFS בגרף G. השורש r של העץ T הוא צומת הפרדה, אםם דרגת השורש בעץ T, (כלומר מספר הקשתות בעץ T הנוגעות בשורש) גדולה מ-1.
נסמן את דרגת הצומת r בעץ T על ידי $\deg_T r$.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

הוכחה

כיוון ראשון: יהי T עץ DFS עם שורש r ונניח כי ה שורש r הוא צומת הפרדה בגרף G. עלינו להוכיח כי $\deg_T r > 1$.
מאחר ש-r הוא צומת הפרדה בגרף G, ל-r יש שני עדים $u, v \in V$. העץ T פורש את G , ולכן יש בו מסלול (לא מכוון) בין הצומת u לצומת v. מסלול זה, חייב לעבור דרך r ומכאן דרגת היציאה של הצומת r בעץ T, גדולה מ-1.

כיוון שני: יהי r השורש בעץ DFS T ונניח כי $\deg_T r > 1$. עלינו להוכיח כי השורש r הוא צומת הפרדה בגרף G.

הוכחה (המשך)

© כל הזכויות שמורות לפרופסור עמוס ישראלי

נתבונן בשני צמתים u ו-v שמרחק כל אחד מהם מ-r, בעץ T הוא 1. בלי הגבלת כלליות נניח כי $d[v] > d[u]$.
נניח בשלילה כי r אינו צומת הפרדה ב- G. במקרה זה, יש מסלול α בין מסלול בין הצמתים u ו-v אשר אינו עובר בצומת r. מכאן נובע כי בזמן $d[u]$ המסלול α הוא מסלול לבן בין הצומת u לצומת v. ממשפט המסלול הלבן נובע כי הצומת v, הואו צאצא של הצומת u בעץ T. בסתירה להנחתנו כי המרחק בין r ל-v בעץ T הוא 1. **סתירה**.

מש״ל

כלל העלה

© כל הזכויות שמורות לפרופסור עמוס ישראלי

עלה של עץ DFS אינו צומת הפרדה.

הוכחה

יהי v עלה בעץ החיפוש ונניח בשלילה כי v הוא גם צומת הפרדה. יהיו u וw שני עדים לv.

העץ T פורש את G ולכן בT יש מסלול (לא מכוון) בין u לw. הצומת v חייב להיות על המסלול הזה. מכאן: דרגת היציאה של v בעץ T גדולה מ 0.

סתירה כי הנחנו שv עלה.

מש"ל

© כל הזכויות שמורות לפרופסור עמוס ישראלי

טיפול בצמתי בינים

עד כה טיפלנו בשורש העץ ובעלים שלו. נותר לנו לטפל בצמתים הנותרים - צמתי הביניים.

גרף מכוון אשר גרף התשתית שלו הוא עץ נקרא **עץ מכוון.**

שורש של עץ מכוון $T(V,E)$ הוא צומת $s\in V$ המקיים: לכל $v\in V$, בT יש מסלול מכוון מ s אל v.

יהי $T=(V,E')$, עץ DFS בגרף $G=(V,E)$, כאשר $E'\subseteq E$. כבר הוכחנו כי אם נכוון את קשתות T בכיוון בו הן נעברו בפעם הראשונה במהלך החיפוש, נקבל כי T הוא עץ המושרש בצומת התחלת החיפוש s.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

צמתי ביניים (המשך)

יהיו $v,u\in V$ צמתים בעץ מושרש T. אם יש מסלול מu אל v נאמר כי u הוא **אב קדמון** של v וv הוא **צאצא** של u.

צומת v בעץ DFS T, הוא **צומת ביניים** אם v אינו שורש ואינו עלה.

קשת אחורית $e=(w,u)$ המכוונת מw אל u נקראת **חולפת על פני** v אם u הוא **אב קדמון** של v וw הוא **צאצא** של v, (ושל u).

© כל הזכויות שמורות לפרופסור עמוס ישראלי

כלל צומת הבינים

יהי v צומת ביניים (צומת שאינו שורש או עלה) בעץ DFS T, בגרף G. הצומת v הוא צומת הפרדה אמם ל- v יש בן, u, בעץ T, והבן u מקיים:

יהי T_1, תת העץ, של T המושרש ב-u, אזי לא קיימת קשת אחורית היוצאת מצומת ב-T_1 אשר חולפת על פני v.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

הוכחה

כיוון ראשון: נניח כי צומת הביניים v הוא צומת הפרדה. עלינו להוכיח כי לצומת v קיים בן u (בעץ T) כך שלא קיימת קשת אחורית אשר יוצאת מתת העץ T_1, המושרש בצומת u, אשר חולפת על פני הצומת v. יהיו x ו-y שני עדים של v. בלי הגבלת כלליות נניח כי $d[x]<d[y]$.

נתבונן במקרים הבאים :

מקרה 1: הצומת x נמצא בין השורש r לבין הצומת v.

מקרה 2: אף אחד מן הצמתים x ו y לא נמצא בין השורש r לבין הצומת v.

(**הערה:** כאן יש לאייר את שני המקרים.)

© כל הזכויות שמורות לפרופסור עמוס ישראלי

הוכחת מקרה 1

נתבונן במסלול (היחיד) בעץ T המחבר את x עם y. זהו מסלול מכוון מ x אל y העובר דרך v, כי x וy הם **עדים** של v. יהי u הצומת הראשון במסלול לאחר v ויהי T_1 תת העץ של T המושרש ב-u. הצומת y נמצא בעץ T_1.

עלינו להראות כי מ T_1 לא יוצאת קשת אשר חולפת על פני v.

נניח בשלילה כי קיימת קשת כזו, $e=(z,w)$, כאשר z הוא צומת בT_1 וw הוא צומת על המסלול מ-r אל v.

נתבונן כעת במסלול המורכב מתת המסלולים הבאים :

© כל הזכויות שמורות לפרופסור עמוס ישראלי

1. מ y אל z - יש מסלול כזה כי שניהם שייכים ל T_1.
2. הקשת $e=(z,w)$.
3. מ w אל x - יש מסלול כזה, כי שני הצמתים נמצאים על המסלול ב T מ r אל v.

המסלול המורכב משלושת תת המסלולים האלה, מחבר את y עם x אך אינו עובר בv. **סתירה** לכך ש-x ו- y הם **עדים** לצומת v.

מש״ל

© כל הזכויות שמורות לפרופסור עמוס ישראלי

הוכחת מקרה 2:

במקרה זה, יש ב T מסלול, α, בין הצומת x לצומת y כי T **פורש** את G. המסלול α עובר ב-v, כי x ו-y הם **עדים** של v. יהיו u_1 ו u_2, הצמתים משני צדדיו של הצומת v במסלול α, יהי T_1 (T_2) תת העץ של T המושרש ב z (w) והמכיל את x (y).

עלינו להוכיח כי **לא קיימת** קשת (אחורית) היוצאת מ-T_1 **או לא קיימת** קשת אחורית היוצאת מ-T_2 וחולפת על פני הצומת v. נניח בשלילה כי קימות שתי קשתות כאלה $e_1=(z_1,w_1)$ ו $e_2=(z_2,w_2)$ אשר חולפות על פני v.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

הוכחת מקרה 2 (המשך):

נתבונן כעת במסלול המורכב מתת המסלולים הבאים:

1. מ x אל z_1 - יש מסלול כזה כי שניהם שייכים ל T_1.
2. הקשת $e_1=(z_1,w_1)$.
3. מ w_1 אל w_2 - יש מסלול כזה, כי שני הצמתים נמצאים על המסלול בT מ r אל v.
4. הקשת $e_2=(z_2,w_2)$.
5. מ z_2 אל y - יש מסלול כזה כי שניהם שייכים ל T_2.

המסלול המורכב מחמשת תת המסלולים האלה, מחבר את x עם y אך אינו עובר בv. **סתירה** לכך שx ו y הם **עדים** ל v. **מש"ל**

© כל הזכויות שמורות לפרופסור עמוס ישראלי

כיוון שני: יהי T_1 תת עץ של עץ DFS, T בגרף G. נניח כי T_1 מושרש בצומת u, הקשת מv אל u היא קשת של T, וכי אין אף קשת אחורית מצומת בT_1 החולפת על פני v. יהי w אביו של v בעץ T.

נראה כעת כי כל המסלולים בין w ל u בגרף G, עוברים בצומת v, כלומר v הוא צומת הפרדה בG.

נניח בשלילה כי בG יש מסלול, α, העובר בין w לu, אשר אינו מכיל את v. תהי יהי x הצומת הראשון במסלול α אשר נמצא בין r לבין v.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

שימו לב: המסלול α נגמר ב w ולכן הצומת x הוא הצומת u, או צומת אחר הנמצא בין r לבין u. תהי $e=(y,x)$ הקשת בα המגיעה לצומת x.

במהלך ביצוע DFS, מרכז הפעילות מתקדם מ x אל w (אם $x \neq w$), משם אל v ואחר כך אל u. לאחר מכן, מרכז הפעילות ממשיך להתקדם, והוא מגיע אל כל הצמתים הנגישים מ u, בטרם יסוג חזרה וכל הצמתים הללו שייכים לעץ T_1 המושרש בu.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

בפרט, מרכז הפעילות מגיע אל y ובזמן שהוא בy, הקשת $e=(y,x)$ נסרקת ומרכז הפעילות נסוג מייד, מפני שצבעו של x הוא **אפור.**

במלים אחרות: הקשת e היא קשת אחורית היוצאת מצומת בT_1 וחולפת על פני v. **סתירה**.

מכאן אנו מסיקים: כל מסלולים בG, בין w לבין u עוברים דרך הצומת v ולכן v הוא **צומת הפרדה** בG.

מש"ל

© כל הזכויות שמורות לפרופסור עמוס ישראלי

<u>מסלול עוקף וצומת נמוך</u>

המשפט שהוכחנו, מהווה תנאי הכרחי ומספיק לקביעה האם צומת בינים בעץ DFS הוא צומת הפרדה.

כדי לאפשר זיהוי יעיל, נגדיר **מסלול עוקף וצומת נמוך** . לאחר מכן, נוכיח גרסה נוספת, נוחה יותר לחישוב, של כלל צומת הביניים.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

<u>מסלול עוקף - הגדרה</u>

יהי $G(V,E)$ גרף לא מכוון בו בוצע DFS, יהי T עץ ה DFS ויהי $v \in V$ צומת בינים ב T.

מסלול עוקף מv, הוא מסלול מכוון, α, המתחיל בv והמקיים אחד מן התנאים הבאים:

1. המסלול α הוא ריק.
2. בα אין קשתות עץ ויש קשת אחורית אחת.
3. בα יש מספר קשתות עץ, בכיוון מעלה העץ, ואחריהן קשת אחורית אחת.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

<u>ערך נמוך - הגדרה</u>

יהי $G(V,E)$ גרף לא מכוון בו בוצע DFS, יהי T עץ ה DFS ויהי $v \in V$ צומת בינים ב T.

לכל צומת $v \in V$, נגדיר את **הערך הנמוך** של v, המסומן ב $low[v]$, כצומת בעל זמן הגילוי, $d[v]$, המינימלי אליו אפשר להגיע מv על ידי מסלול עוקף.

המשפט הבא, מסביר את הקשר בין הערך הנמוך של v, $low[v]$, לבין זיהויו של v כצומת הפרדה:

© כל הזכויות שמורות לפרופסור עמוס ישראלי

<u>משפט</u>

יהי $G(V,E)$ גרף לא מכוון בו בוצע DFS. יהי v צומת בינים בעץ החיפוש T. הצומת v הוא **צומת הפרדה** אם ורק אם בT יש קשת (v,u) המכוונת מ v אל u. ומתקיים $low(u) \ge d[v]$.

<u>הוכחה</u>

$\Leftarrow$לפי המשפט שהוכחנו, אם v הוא צומת הפרדה אשר הוא צומת בינים בעץ T , אז בעץ T יש תת עץ, T_1 המושרש בu, $e=(v,u)$ היא קשת בעץ T ואין קשת אחורית היוצאת מT_1 וחולפת על פני v. בתנאים אלה,

$low[u] \ge d[v]$ **מש״ל**

© כל הזכויות שמורות לפרופסור עמוס ישראלי

$\Rightarrow$ נניח כעת כי $e=(v,u)$ היא
קשת בעץ T ומתקיים
$low[u] \ge d[v]$.
עלינו להוכיח כי v הוא צומת הפרדה
בG. נניח בשלילה כי v אינו צומת
הפרדה ויהי T_1 תת העץ של T
המושרש בצומת u. לפי המשפט
שהוכחנו, יש קשת אחורית
$e=(y,x)$ אשר חולפת על פני v,
כלומר: y הוא צומת בT_1, וx נמצא
על המסלול מr אל v ומתקיים:
$d[w]<d[v]$. במצב זה,
$low[u]=d[x]<d[v]$.

סתירה **מש״ל**

© כל הזכויות שמורות לפרופסור עמוס ישראלי

<u>כללים לזיהוי צמתי הפרדה</u>

האלגוריתם שנציג לזיהוי צמתי הפרדה ורכיבים אי-פריקים, מסתמך על המשפטים שהוכחנו. כדי לעשות זאת, האלגוריתם מבצע גרסה מחוזקת של DFS. בטרם נציג את האלגוריתם, נציג מחדש את שלושת הכללים שהוכחנו לזיהוי צמתי הפרדה.
אנו מניחים כי $G(V,E)$ הוא גרף לא מכוון שמבצעים בו חיפוש לעומק החל מצומת r וכי T הוא עץ החיפוש.
הכללים שהוכחנו הם:

© כל הזכויות שמורות לפרופסור עמוס ישראלי

<u>כלל השורש</u>

שורש r של עץ DFS T הוא צומת הפרדה, אם $\deg_T r > 1$.

<u>כלל העלה</u>

עלה של עץ DFS אינו צומת הפרדה.

<u>כלל צומת הבינים</u>

יהי v צומת בינים בעץ החיפוש T.
הצומת v הוא **צומת הפרדה** אם ורק אם בT יש קשת (v,u) המכוונת מv אל u. ומתקיים

$$low(u) \ge d[v]$$

© כל הזכויות שמורות לפרופסור עמוס ישראלי

<u>חישוב הצומת הנמוך</u>

האלגוריתם לזיהוי צמתי הפרדה משתמש בDFS ומחשב לכל צומת את הערך הנמוך כדלהלן:

1. כאשר מגלים צומת v וקובעים את זמן הגילוי, מבצעים גם:
$low[v] \leftarrow d[v]$,
זיהינו מסלול עוקף **ריק**.
2. כאשר מסיירים בקשת אחורית, (v,u) מעדכנים:
$$low[v] \leftarrow \min\{low[v], d[u]\}$$
זיהינו מסלול עוקף ללא קשתות עץ ובו קשת אחורית **יחידה.**

© כל הזכויות שמורות לפרופסור עמוס ישראלי

חישוב הצומת הנמוך (המשך)

3. כאשר נסוגים לאורך קשת עץ (v,u) מבצעים:

$$low[v] \leftarrow \min\{low[v], low[u]\}$$

המסלול העוקף שזיהינו, מתחיל בקשת (v,u) וממשיך במסלול העוקף אשר קבע את הצומת הנמוך של u.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

זיהוי צמתי הפרדה – זיהוי צמתי ביניים

האלגוריתם לזיהוי צמתי הפרדה, מבצע DFS ובנוסף, מחשב עבור כל צומת, את הערך הנמוך. הפעם האחרונה שהאלגוריתם נמצא בצומת u היא כאשר נסוגים מu אל v, לאורך קשת עץ (v,u).
באותו רגע, החישוב של הערך הנמוך של הצומת u כבר הושלם ואפשר לבדוק, האם

$$low[u] \geq d[v]$$

אם התשובה חיובית, מכריזים על v כעל צומת הפרדה.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

זיהוי צמתי הפרדה – זיהוי שורש העץ

השורש של עץ DFS הוא צומת הפרדה אם דרגתו (בעץ) >1. כדי לזהות את השורש כצומת הפרדה, מוסיפים לכל צומת v, מונה למספר הבנים של vת $dout[v]$.
בכל פעם שמתקדמים מן השורש, בודקים האם $dout[v] > 0$ ואם כן, שורש העץ מזוהה כצומת הפרדה.

הקוד הסופי מופיע בשקף הבא.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

הקוד הישן

$DFS-visit(u)$ /* סייר ב u */
$d[u] \leftarrow time++$

for each ($v \in Adj[u]$ **and** $v \neq \Pi[u]$)
  if ($d[v]=0$) /* v is **new** */

    $DFS-visit(v)$ /* סייר בu */

  else /* v is not **new** */

  endif
endfor
$f[u] \leftarrow time++$
end /* **נסיגה מ** v */

© כל הזכויות שמורות לפרופסור עמוס ישראלי

תוספת קוד לזיהוי השורש כצומת הפרדה

להלן (במסגרות) טיפול בשורש העץ:

$DFS-visit(u)$ /* סייר ב u */
$d[u] \leftarrow time++$
$degout[u] \leftarrow 0$
$low[u] \leftarrow d[u]$;
foreach $(v \in Adj[u])$ **and** $(v \neq \Pi[u])$
  if $d[v]=0$ /* v is **new** */
    if $d[u]=1$ **and** $degout[u]>0$
      u **(root) is** a CP
    endif
    $degout[u]++$
    $\Pi[v] \leftarrow u$
    $DFS-visit(v)$ /* סייר ב v */

  else /* v is not **new** */

  endif
end foreach
$f[u] \leftarrow time++$
end /* u **נסיגה מ** */

© כל הזכויות שמורות לפרופסור עמוס ישראלי

תוספת קוד לזיהוי צומת בינים כצומת הפרדה

להלן (במסגרת) טיפול בצומת ביניים:

$DFS-visit(u)$ /* סייר ב- u */
$d[u] \leftarrow time++$
$degout[u] \leftarrow 0; low[u] \leftarrow d[u]$
foreach $(v \in Adj[u])$ **and** $(v \neq \Pi[u])$
  if $d[v]=0$ /* v is **new** */
    if $d[u]=1$ **and** $degout[u]>0$
      u **(root) is** a CP
    endif
    $degout[u]++; \quad \Pi[v] \leftarrow u;$
    $DFS-visit(v)$ /* סייר ב v */
    if $d[u]>1$ **and** $low[v] \geq d[u]$
      u **(intermediate) is a** CP
    endif
    $low[u] \leftarrow \min(low[u], low[v])$
  else /* v **is not new** */
    $low[u] \leftarrow \min(low[u], d[v])$
  endif
end foreach
$f[u] \leftarrow time++$
end /* u **נסיגה מ** */

© כל הזכויות שמורות לפרופסור עמוס ישראלי

נכונות האלגוריתם

נכונות האלגוריתם נובעת מן המשפטים שהוכחנו.

סיבוכיות האלגוריתם

מאחר שכל התוספות לאלגוריתם הן של פעולות קונסטנטיות. סיבוכיות האלגוריתם לזיהוי צמתי הפרדה ורכיבים אי-פריקים זהה לסיבוכיות DFS כלומר:

$$O(|E|+|V|)$$

© כל הזכויות שמורות לפרופסור עמוס ישראלי

מציאת צמתים ברכיבים אי-פריקים

כל צומת הפרדה נמצא במספר רכיבים אי-פריקים. קשתות הרכיב הן כל הקשתות **ששני הצמתים** שלהם נמצאים באותו רכיב אי-פריק.

כדי לחלק את צמתי הגרף G לרכיבים אי-פריקים משתמשים במחסנית צמתים לפי הכללים הבאים:

1. מכניסים את שורש העץ למחסנית.
2. כאשר מגלים צומת חדש, מכניסים את הצומת למחסנית.
3. כאשר מגלים קשת אחורית ומבצעים נסיגה מיידית, לא משנים את מצב המחסנית.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

4. כאשר נסוגים אל צומת אשר אינו צומת הפרדה, לא מבצעים דבר.
5. כאשר נסוגים מ-v אל u ומזהים כי u הוא צומת הפרדה, מוציאים את כל הצמתים במחסנית **עד** v (כולל v) מן המחסנית ומדפיסים אותם **ביחד עם** u, זהו הרכיב האי-פריק שגילינו. **שימו לב:** הצומת u נשאר במחסנית.
6. כאשר האלגוריתם מסתיים, פולטים את כל הצמתים הנמצאים במחסנית. זהו הרכיב האי-פריק האחרון.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

מציאת קשתות ברכיבים אי-פריקים

קשתות הגרף מתחלקות לרכיבים אי-פריקים באופן יחיד. לכל קשת רכיב משלה. אם רוצים לזהות קשתות של רכיב משתמשים במחסנית קשתות. בכל פעם שמסיירים בקשת **מכניסים אותה למחסנית**.

כאשר נסוגים אל צומת הפרדה v לאורך קשת עץ (v,w) **מוציאים מן המחסנית** את כל הקשתות עד הקשת (v,w) כל הקשתות שהוצאו מהוות רכיב אי-פריק יחיד.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

סיכום

בהרצאה זו, הגדרנו את המושגים **צמתי הפרדה** ו**רכיבים אי-פריקים** בגרף לא מכוון.

לאחר שהגדרנו מושגים אלה, למדנו איך לחזק את אלגוריתם חיפוש לעומק כדי לזהות את הרכיבים האי-פריקים של גרף לא מכוון.

סיבוכיות האלגוריתם הסופי, זהה לסיבוכיות חיפוש לעומק.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

הרצאה 10: רשתות זרימה

בעיות זרימה מטפלות בתופעות מעבר של ״סחורות״ מסוג כלשהו, כגון:

1. זרימת נוזלים בצנרת.
2. העברת סחורות ברשת כבישים.
3. העברת חבילות (Packets) ברשת.
4. ועוד.

בהרצאה זו נגדיר רשתות זרימה ופונקציות זרימה ונוכיח קיום של מספר תכונות של רשתות כאלה.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

אלגוריתמים ב׳ הרצאה מס׳ 10 - פרופסור עמוס ישראלי

רשת זרימה – מוטיבציה

בטיפול בבעיות זרימה, איננו מתענינים **בזמן המעבר** במערכת אלא אך ורק ב**קיבול המערכת** אשר מוגדר כ**מספר יחידות הזרימה (למשל מכוניות)** המירבי אשר יכול לעבור במערכת במהלך יחידת זמן נתונה (למשל שעה).

אנו נטפל במקרה פשוט ביותר בו יש זרימה **ממקור** יחיד ומגיעה אל **יעד** יחיד. ברצוננו לפתח אלגוריתם שיקבל כקלט את נתוני המערכת, ויחשב את כמות הסחורות הכוללת שאפשר להעביר מן המקור אל הבור ביחידת זמן נתונה.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

אלגוריתמים ב׳ הרצאה מס׳ 10 - פרופסור עמוס ישראלי

זמן מעבר בכביש וקיבול הכביש

בהנתן כביש כלשהו, ביצועי הכביש והתועלת שהוא מביא נקבעים על ידי שני פרמטרים חשובים:

1. **זמן המעבר -** הזמן הנדרש למכונית לעבור מתחילת הכביש ועד סופו.
2. **קיבול הכביש -** מספר המכוניות ליחידת זמן אשר יכולות לעבור בכביש.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

אלגוריתמים ב׳ הרצאה מס׳ 10 - פרופסור עמוס ישראלי

זמן מעבר בכביש וקיבול הכביש

לדוגמא מכונית צריכה בערך שעה לנסיעה מחיפה אל תל -אביב. שעה זו היא **זמן המעבר** מחיפה אל תל-אביב. במשך שעה זו, מספר רב של מכוניות, אשר יצאו מחיפה מוקדם יותר, יגיעו אל תל-אביב. מספר המכוניות המירבי, אשר יגיע אל תל אביב הוא **קיבול הכביש**.

שימו לב: הקיבול יכול להיות שונה בחלקי כביש שונים, לדוגמא, מספר המכוניות לדקה בקטע ישר, גדול בהרבה ממספר המכוניות לדקה בסיבוב.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

<u>זמן מעבר במערכת וקיבול המערכת</u>

כאשר אנו מנסים להכליל את מושגי זמן המעבר והקיבול מכביש יחיד למערכת שלמה יש לשים לב כי לעתים, קיימים במערכת מספר נתיבים מן ממקור ליעד נתונים.

זמן המעבר במערכת, מן המקור אל היעד, תלוי בנתיב המסוים אשר נבחר. במקרה כזה, אפשר לדון בזמן מעבר **מינימלי** או בזמן מעבר **ממוצע**.

קיבול המערכת, שווה למספר המכוניות המירבי, אשר יכול להגיע אל היעד במשך יחידת זמן שנקבעת מראש ותלוי בקיבולי כל הכבישים במערכת.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

לדוגמא: האלגוריתם יקבל את מספר המכוניות שיכולות לעבור בכל אחד מכבישי המערכת, ויחשב את מספר המכוניות המירבי שיכול להגיע אל הבור במשך שעה אחת.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

<u>רשתות זרימה – מידול</u>

אנו נמדל (נציג בעזרת מודל) תופעות זרימה על ידי גרף הקרוי **רשת זרימה**. במהלך המידול, נשתמש בדוגמא של רשת תחבורה, המעבר לרשת תקשורת, למערכת צינורות או כל מערכת דומה הוא פשוט ביותר.

כל **צומת** במערכת הכבישים, ימודל על ידי **צומת בגרף**.

כל **כביש** במערכת הכבישים, ימודל על ידי **קשת בגרף**.

1. **כמות המכוניות** שיכולה לעבור בכביש מסוים, תמודל על ידי **פונקצית קיבול** של הקשת הממדלת את הכביש.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

<u>פונקצית הקיבול</u>

לכל כביש קיים חסם עליון על מספר המכוניות שיכול לעבור בכביש במשך שעה. הדרך שבה אפשר למדוד זאת היא:

1. לדאוג להזרמת מכוניות מירבית בכביש.
2. כאשר התנועה בכביש בעיצומה, יש להציב מונה בקצה הכביש ולספור כמה מכוניות עברו על פני המונה במשך שעה.
3. מספר המכוניות שעברו הוא מספר המכוניות המירבי שהכביש יכול להעביר במשך שעה.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

רשתות זרימה - הגדרה

רשת זרימה, $G=(V,E,c,s,t)$ כוללת:

1. גרף מכוון חסר לולאות עצמיות וקשתות מקבילות $G=(V,E)$.
2. לכל קשת $e\in E$, מוגדרת **פונקציית קיבול** $c(e)$, $c:E\to R$ כאשר לכל $e\in E$ מתקיים $c(e)>0$.
3. **צומת מקור** s המקיים $\deg_{in}(s)=0$.
4. **צומת בור** t המקיים $\deg_{out}(t)=0$.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

פונקציית זרימה

בידינו נמצא עכשיו מודל של הרשת. בכל רשת כבישים אפשר להשתמש בהרבה אופנים: לעתים אין תנועה בכלל, לעתים התנועה דלילה ולעתים היא צפופה.

פוונקציית זרימה מאפשרת לנו לתאר מצבים שונים של התנועה. אנו מניחים כי פונקצית הזרימה, מציינת עבור כל קשת e, את מספר יחידות הזרימה העוברות בe ביחידת זמן נתונה.

לאחר שנגדיר **פונקציית זרימה חוקית וזרימה כוללת** נראה איך אפשר לחשב את **הזרימה המירבית (קיבול המערכת)**.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

פונקציית זרימה - הערה

אנו מטפלים אך ורק במצב בו הזרימה ברשת **יציבה**. במצב זה, לכל קשת e, מספר יחידות הזרימה אשר **נכנסות אל** e ביחידת זמן מסוימת, **שווה** למספר יחידות הזרימהת אשר **יוצאות מ** e במשך אותה יחידת זמן.

איננו מטפלים בתהליך המתחיל כאשר הרשת **ריקה** ואשר במהלכו הרשת **מתמלאת**.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

פונקציית זרימה - הגדרה

בהנתן רשת זרימה $G=(V,E,s,t,c)$, כמות הזרימה בכל קשת נתונה על ידי **פונקצית זרימה** $f:E\to R^{+}$.

לכל קשת, $e\in E$, $f(e)$ מציינת את מספר יחידות הזרימה העוברות ב e, במשך יחידת זמן קבועה מראש. יחידה זו, יכולה להיות שניה, דקה, יממה או כל יחידה אחרת שבה התנועה נשארת **יציבה**.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

חוקי הזרימה

פונקציה $f : E \to R^+$ תיקרא פונקציית **זרימה חוקית** אם מתקיימים חוק הקשת וחוק הצומת:

חוק הקשת

לכל קשת $e \in E$, $0 \le f(e) \le c(e)$.

כלומר, הזרימה בקשת אינה עולה על קיבול הקשת.

חוק הצומת

לכל צומת $v \in V - \{s,t\}$

$$\sum_{e=(u,v)} f(e) = \sum_{e=(v,w)} f(e) \qquad (1)$$

כלומר, סכום הזרימה על הקשתות הנכנסות לv שווה לסכום הזרימה על קשתות היוצאות מv.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

הזרימה הכוללת ברשת

בהינתן רשת זרימה $G = (V,E,s,t,c)$ ופונקציית זרימה בG, f, הזרימה הכוללת בG תוגדר על ידי

$$F_f = \sum_{e=(v,t)} f(e) \qquad (2)$$

כלומר הזרימה הכוללת שווה למספר יחידות הזרימה המגיעות אל הבור.

הבעיה החישובית שנדון בה היא:

בהנתן רשת זרימה $G = (V,E,s,t,c)$ מצא פונקצית זרימה חוקית, f, כך שהזרימה הכללית ברשת F_f היא מירבית.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

חתכים ברשת זרימה

אינטואיטיבית, חוק הצומת גורם לכך שאין אפשרות ל**צבירת יחידות זרימה** בשום חלק של רשת הזרימה ולכן, כאשר הרשת במצב יציב מספר יחידות הזרימה הנכנס אל הרשת (במקור) **שווה** למספר יחידות הזרימה היוצא ממנה (בבור).

להלן נגדיר את מושג ה**חתך** ברשת זרימה ונוכיח כי הטענה האינטואיטיבית שניסחנו נכונה לא רק עבור **המקור והבור** אלא עבור כל חתך **ברשת**.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

חתכים ברשת זרימה - הגדרה

תהי $G = (V,E,c,s,t)$ רשת זרימה ותהי $S \subset V$ תת קבוצה של V המקיימת: $s \in S$, $t \notin S$.

שימו לב: החתך מחלק את V לשתי קבוצות **זרות**: S ו- $\bar{S} = V - S$ כאשר דורשים: $s \in S$, $t \in V - S$.

נסמן ב $(S : \bar{S})$ את קבוצת הקשתות מצמתים בS אל צמתים ב$V - S$.

נסמן ב $(\bar{S} : S)$ את קבוצת הקשתות מצמתים ב $V - S$ אל צמתים ב S.

קבוצת הקשתות המאוחדת, בשני הכיוונים, נקראת: **החתך המוגדר על ידי S.**

© כל הזכויות שמורות לפרופסור עמוס ישראלי

קיבול של חתכים

תהי $G(V,E,s,t,c)$ רשת זרימה ותהי f פונקצית זרימה ברשת.

לכל חתך S נגדיר את **הקיבול של S** ככמות הזרימה המירבית הניתנת להזרמה מן הצמתים בS, אל הצמתים ב$V-S$ כלומר:

$$c(S)=\sum_{e\in(S:\bar{S})} c(e)$$

© כל הזכויות שמורות לפרופסור עמוס ישראלי

זרימה בחתכים

תהי $G(V,E,s,t,c)$ רשת זרימה ותהי f פונקצית זרימה ברשת.

לכל חתך S, הזרימה f גורמת למעבר זרימה בין הצמתים ב-S לבין הצמתים ב-$\bar{S}$. נגדיר את הזרימה בחתך $(S,\bar{S})$, $f(S,\bar{S})$, כסכום הזרימה בקשתות מצמתים בS אל צמתים ב$\bar{S}$ **פחות** סכום הזרימה בקשתות מצמתים ב$\bar{S}$ אל צמתים בS.

$$f(S:\bar{S})=\sum_{e\in(S:\bar{S})} f(e)-\sum_{e\in(\bar{S}:S)} f(e)$$

נשים לב כי הזרימה הכוללת ברשת מוגדרת כזרימה בחתך $(V-t:t)$.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

למות החתך

שתי הלמות הבאות מביאות שתי תכונות בסיסיות של הקיבול והזרימה בחתכי הרשת.

הלמה הראשונה מראה כי לכל חתך ברשת, $(S,\bar{S})$, ולכל פונקצית זרימה f, הקיבול של $(S,\bar{S})$, הוא **חסם עליון** לזרימה ב-$(S,\bar{S})$.

הלמה השניה מראה כי לכל חתך ברשת $(S,\bar{S})$, הזרימה בחתך $(S,\bar{S})$ שוה לזרימה הכוללת ברשת, F_f.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

למה 1

לכל חתך S, $f(S:\bar{S})\le c(S:\bar{S})$.

הוכחה

ההוכחה נובעת מיידית מן ההגדרות ומחוק הקשת:

$$f(S:\bar{S})=\sum_{e\in(S:\bar{S})} f(e)-\sum_{e\in(\bar{S}:S)} f(e)$$

$$\le \sum_{e\in(S:\bar{S})} c(e)-0=c(S)$$

מש״ל

© כל הזכויות שמורות לפרופסור עמוס ישראלי

<u>למה 2</u>

לכל חתך S, $S \subset V$, ולכל פונקצית זרימה חוקית f מתקיים:

$$f\left(S:\bar{S}\right) \sum_{e\in\left(S:\bar{S}\right)} f(e) - \sum_{e\in\left(\bar{S}:S\right)} f(e) = F_f$$

© כל הזכויות שמורות לפרופסור עמוס ישראלי

<u>הוכחה</u>

לכל אחד מן הצמתים ב $\bar{S}$, נציב משוואה כדלהלן:

עבור הצומת t, נשתמש בהגדרת הזרימה הכוללת ברשת:

$$F_f = \sum_{e=(v,t)} f(e)$$

כל אחד משאר הצמתים ב $\bar{S}$ מקיים את חוק הצומת

$$\sum_{e=(u,v)} f(e) = \sum_{e=(v,w)} f(e)$$

ואם נעביר מאגף אחד לשני נקבל, עבור כי כל צומת $v \in \bar{S} - t$ מקיים:

$$0 = \sum_{e=(u,v)} f(e) - \sum_{e=(v,w)} f(e)$$

© כל הזכויות שמורות לפרופסור עמוס ישראלי

<u>הוכחה - המשך</u>

אם נסכם את כל המשוואות הללו נקבל:

באגף השמאלי נקבל $F(f)$ כמבוקש.

באגף הימני נקבל:

1. כל קשת מצומת ב$\bar{S}$ אל צומת ב$\bar{S}$ תופיע פעמיים בסימנים הפוכים ולכן תבוטל.
2. כל קשת מצומת ב$\bar{S}$ אל צומת בS תופיע אך ורק בסימן -.
3. כל קשת מצומת בS אל צומת ב $\bar{S}$ תופיע אך ורק בסימן + .

מש״ל

© כל הזכויות שמורות לפרופסור עמוס ישראלי

<u>מסקנה</u>

תהי f פונקציית זרימה ברשת $G=(V,E,c,s,t)$. אם מוצאים חתך ברשת S המקיים:

$$F(S) = F(f) = c(S)$$

אזי, הזרימה $F(f)$ היא מירבית.

<u>הוכחה</u>

לפי למה 1 הזרימה בכל אחד מחתכי הרשת שווה ל f. בפרט, הזרימה בחתך $\left(S:\bar{S}\right)$ שווה ל f. לפי למה 2, הזרימה בחתך $\left(S:\bar{S}\right)$ חסומה מלמעלה על ידי $c(S)$ ולכן, אין אפשרות לזרימה שערכה גדול יותר

מש״ל

© כל הזכויות שמורות לפרופסור עמוס ישראלי

זיהוי זרימה מירבית

הלמה האחרונה נותנת לנו מכשיר לבדיקה האם פונקציית זרימה נתונה היא זרימה מירבית.

הגדרה: תהי רשת $G=(V,E,c,s,t)$ זרימה עם פונקצית זרימה f. חתך S ברשת הוא **חתך רווי** אם מתקיים $f(S:\bar{S})\leq c(S:\bar{S})$.

שימו לב: בתנאים אלה, לא קיימת פונקצית זרימה g המקיימת $F_g > F_f$.
(הוכיחו זאת!).

© כל הזכויות שמורות לפרופסור עמוס ישראלי

זיהוי זרימה מירבית (המשך)

אפשר להוכיח גם את ההיפך כלומר:

משפט

אם f היא פונקצית זרימה מירבית ברשת $G=(V,E,c,s,t)$ אזי ב-G יש חתך רווי.

ההוכחה למשפט זה תנתן בהמשך.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

אלגוריתם ישיר

בדרך זו יש בידינו אלגוריתם ישיר לקביעת הזרימה המירבית ברשת: נעבור על כל חתך ברשת S ונחשב את הקיבול שלו. החתך בעל הקיבול המינימלי יקבע את הזרימה המירבית ברשת. **תרגיל:** מה סיבוכיות האלגוריתם?

רמז: זכרו, כל קבוצת צמתים המכילה את s ואשר אינה מכילה את t מגדירה חתך.

סיכום

בהרצאה זו, הצגנו את מודל רשתות הזרימה ואת תכונותיהן.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

הרצאה 11:האלגוריתם של Ford וFullkerson

בהרצאה זו נמשיך את טיפולנו ברשתות זרימה ונלמד את אלגוריתם Ford-Fulkerson לחישוב זרימה מירבית ברשתות אלה.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

קיבולים אפקטיביים ומסלולי שיפור

תהי $G(V,E,c,s,t)$ רשת זרימה ותהי f פונקציית זרימה. לכל קשת $e \in E$, **הקיבול האפקטיבי של** e שווה לכמות הזרימה שאפשר להוסיף לe כלומר:

$$c(e)-f(e)$$

קשת רוויה היא קשת שקיבולה האפקטיבי הוא 0.

מסלול שיפור הוא מסלול מכוון מs אל t שבו אין אף קשת רוויה.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

שיפור זרימה לאורך מסלול שיפור

נניח כי f היא פונקצית זרימה, α הוא מסלול שיפור וv הוא צומת ביניים בα. תהינה $e_1=\overline{(u,v)}$ ו $e_2=\overline{(v,w)}$ קשתות בα. הזרימה f מקיימת את **חוק הצומת** ולכן, כמות הזרימה הנכנסת אל v שווה לכמות הזרימה היוצאת ממנו.

אם נוסיף לזרימה בe_1 ובe_2 אותה כמות זרימה, עבור v, חוק הצומת ישמר. מכאן, אם נוסיף את אותה כמות הזרימה, לכל הקשתות בα, חוק הצומת ישמר בכל צמתי הרשת.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

שיפור זרימה לאורך מסלול שיפור

בהנתן פונקצית זרימה f ומסלול שיפור α, אפשר **לחזק** את הזרימה f, על ידי הגדלת הזרימה בכמות שווה בכל אחת מקשתות המסלול α.

צואר הבקבוק של מסלול שיפור היא הקשת שקיבולה האפקטיבי הוא מינימלי.

השיפור המירבי האפשרי לאורך α שווה לקיבול האפקטיבי של צואר הבקבוק. בדרך זו, צואר הבקבוק של α הופך לקשת רוויה.

מאחר שמסלול השיפור מחבר את s עם t, הזרימה הכללית ברשת עולה בקיבול של צואר הבקבוק של α.

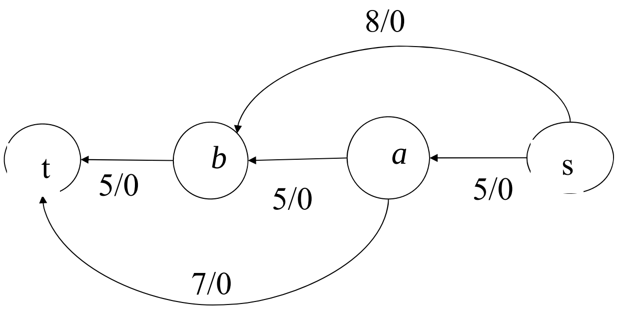
© כל הזכויות שמורות לפרופסור עמוס ישראלי

קשתות שימושיות לאחור

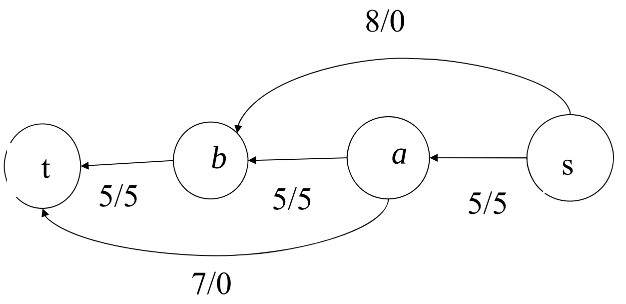
לעתים יתכן מצב בו אין ברשת אף מסלול שיפור ובעת ובעונה אחת, אין ברשת אף חתך רווי. מצב זה נגרם כאשר הזרימה ברשת נקבעה על ידי מספר מסלולי שיפור בדרך שבה בכל אחד מן המסלולים, בין s לבין t, יש קשת רוויה, אולם הקשתות הרוויות כולן, אינן משלימות חתך רווי.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

לדוגמא נתבונן ברשת הבאה:



תחילה נזרים 5 יחידות במסלול (s,a,bb,t) ונקבל:



© כל הזכויות שמורות לפרופסור עמוס ישראלי

המשך שיפור הסרימה

במצב זה אין ברשת חתך רווי, כלומר **הזרימה אינה מכסימלית**. מצד שני אין ברשת אף מסלול שיפור:

במסלול s,a,t, הקשת (s,a) רוויה.

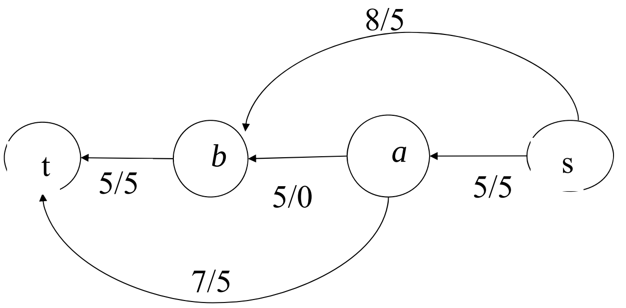
ובמסלול s,b,t הקשת (b,t) רוויה.

כדי לשפר זרימה אנו יכולים:

1. להזרים 5 יחידות בקשת (i,b).
2. נוצר עודף זרימה 5 בצומת b.
3. **להזרים לאחור** (להפסיק להזרים) 5 יחידות בקשת (a,b).
4. נוצר עודף זרימה 5 צומת a.
5. להזרים 5 יחידות בקשת (a,t).

ונקבל את המצב הבא:

© כל הזכויות שמורות לפרופסור עמוס ישראלי



במצב זה, החתך $(\{s,b\}:\{b,t\})$ הוא רווי (בדקו זאת) והזרימה ברשת מרבית.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

הסבר

הצלחנו לשפר זרימה, על ידי הורדת זרימה בקשת מסוימת.

עלינו למצוא דרך להתיחס גם אל שיפורי זרימה שבמהלכם בחלק מן הקשתות **מקטינים את הזרימה** ובלבד ש**הזרימה הכוללת** ברשת **תגדל**.

להלן נגדיר קשתות שימושיות לפנים ולאחור ואחר כך נגדיר **מסלול שיפור זרימה באופן פורמלי:**

© כל הזכויות שמורות לפרופסור עמוס ישראלי

קשתות שימושיות

קשת e תיקרא **שימושית לפנים** אם אפשר להגביר את הזרימה בה כלומר:

$$c(e) > f(e)$$

הקיבול האפקטיבי לפנים של e הוא

$$c(e) - f(e)$$

קשת e תיקרא **שימושית לאחור** אם אפשר להקטין את הזרימה בה כלומר:

$$f(e) > 0$$

הקיבול האפקטיבי לאחור של e הוא $f(e)$.

שימו לב: אם e היא קשת המקיימת $c(e) > f(e) > 0$ אז e היא גם שימושית לפנים וגם שימושית לאחור.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

מסלולי שיפור - הגדרה מחודשת

מסלול שיפור זרימה הוא סדרת צמתים $s, v_1, \ldots, v_{l-1}, t$. בין כל שני צמתים במסלול יש **קשת** ומתקיים:

1. הקשת הראשונה בסדרה, (s, v_1), יוצאת מן הצומת s.
2. הקשת האחרונה במסלול, (v_{l-1}, t), נכנסת אל הצומת t.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

3. כל קשת אחרת בסדרה (v_{i-1}, v_i) מקיימת:
   אם הקשת (v_{i-1}, v_i) מכוונת מהצומת v_{i-1} אל הצומת v_i אז היא **שימושית לפנים.**
   אם הקשת (v_{i-1}, v_i) מכוונת מהצומת v_i אל הצומת v_{i-1}. אז היא **שימושית לאחור**.

צואר הבקבוק של מסלול שיפור היא הקשת שקיבולה האפקטיבי הוא מינימלי.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

האלגוריתם של Ford ו Fulkerson

לכל קשת $e \in E$ קבע זרימה התחלתית
$f(e) \leftarrow 0$

חזור

1. מצא מסלול שיפור מ s אל t על ידי ביצוע BFS על כל הקשתות השימושיות.
2. שפר את הזרימה לאורך מסלול השיפור שמצאת.

עד שלא קיים מסלול שיפור נוסף.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

סיום האלגוריתם

אם הקיבולים והזרימות הם אך ורק מספרים טבעיים, אפשר להוכיח כי האלגוריתם מסתיים, שכן בכל איטרציה, הזרימה הכללית משתפרת במספר שלם. לאחר מספר סופי של איטרציות, מתקבלת זרימה כללית השווה לקיבול החתך המינימלי. זרימה כזו היא מירבית ואינה ניתנת לשיפור. במקרה זה, הסיבוכיות יכולה להיות תלויה בערך הקיבולים, ולא רק בגודל הרשת. אם מתירים קיבולים ממשיים כלשהם, סיום האלגוריתם, כפי שהוגדר עד כה, אינו מובטח.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

נכונות - קיום החתך הרווי ומציאתו

כדי להוכיח כי האלגוריתם נכון, עלינו להוכיח כי לאחר סיום האלגוריתם, הזרימה הכוללת היא מירבית.
טענה זו נובעת מיידית מן הלמה הבאה:

למה

לאחר סיום אלגוריתם Ford ו Fullkerson, יש ברשת חתך רווי.

שימו-לב: כפי שהוכחנו, הזרימה בחתך הרווי שווה לזרימה הכוללת בגרף.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

הוכחה

נבצע BFS על הרשת החל מ s ונשתמש רק בקשתות שימושיות. לא יתכן שנגיע לt, כי אין ברשת מסלולי שיפור.
הקבוצה S מכילה את כל הצמתים שהגענו אליהם במהלך החיפוש. כל הקשתות מ S אל $\bar{S}$ אינן שימושיות ולכן $\left(S : \bar{S}\right)$ הוא חתך רווי.

מש״ל

© כל הזכויות שמורות לפרופסור עמוס ישראלי

<u>סיבוכיות</u>

נגדיר אורך של מסלול שיפור, α, כמספר הקשתות ב α.

קרפ (Karp) ואדמונדס (Edmonds) הוכיחו כי אפשר להבטיח את סיום האלגוריתם על ידי מציאת מסלולי השיפור, בסדר עולה של אורכיהם.

להלן נוכיח כי בדרך זו (על ידי הפעלת BFS על כל הקשתות השימושיות, סיבוכיות האלגוריתם היא

$$\Theta(|V||E|^2)$$

© כל הזכויות שמורות לפרופסור עמוס ישראלי

<u>הרשת השיורית (Residual Network)</u>

תהי $G=(V,E,c,s,t)$ רשת זרימה.

הרשת השיורית $G_f=(V,E_f,c_f,s,t)$ היא רשת זרימה המוגדרת על אותו גרף אולם קבוצת הקשתות E_f מכילה רק **קשתות שימושיות** והקיבול השיורי c_f מוגדר בהתאם.

בהנתן רשת שיורית $G_f=(V,E_f,c_f,s,t)$ נגדיר את **המרחק השיורי** $\delta_f(u,v)$ כמספר הקשתות במסלול מינימלי בין u לבין v.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

<u>למה</u>

אם מריצים את אלגוריתם אדמונדס קרפ אז לכל צומת $v\in V-s,t$, המרחק מ s אל v $\delta_f(u,v)$, גדל לאחר כל שיפור זרימה.

<u>הוכחה</u>

נניח בשלילה כי קיים צעד שיפור המשפר את הזרימה מ f ל f' וקיימים צמתים שהמרחק אליהם קטן, כלומר $\delta_f(s,v)>\delta_{f'}(s,v)$ עבור צומת כלשהו v.

תהי D קבוצת הצמתים שמרחקם השיורי מן הצומת s גדל ויהי v ב D שהמרחק השיורי אליו $\delta_{f'}(s,v)$ הוא מינימלי מבין הצמתים ב D.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

במצב זה מתקיים:

$$\delta_{f'}(s,u)<\delta_{f'}(s,v)\Rightarrow\delta_f(s,u)\le\delta_{f'}(s,u)$$

יהי p', מסלול מינימלי ב G' מ s אל v, ויהי u הצומת הקודם ל v במסלול p'. ברור כי $(u,v)\in E_{f'}$.

ולכן:

$\delta_{f'}(s,u)=\delta_{f'}(s,v)-1$.

מן המסקנה הקודמת נובע כי $\delta_f(s,u)\le\delta_{f'}(s,u)$.

נתבונן במקרים הבאים:

© כל הזכויות שמורות לפרופסור עמוס ישראלי

מקרה 1: $f(u,v)<c(u,v)$

במקרה זה

$$\begin{aligned}\delta_f(s,v) &\le \delta_f(s,u)+1\\ &\le \delta_{f'}(s,u)+1\\ &= \delta_{f'}(s,v)\end{aligned}$$

סתירה.

מקרה 2: $f(u,v)=c(u,v)$

במקרה זה, הקשת (u,v) מופיעה ב E_f אך ורק כקשת שימושית לאחור.

מכאן נובע:

1. $(v,u)\in E_f$.
2. $(u,v)\notin E_f$.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

מאחר שנתון כי $(u,v)\in E_{f'}$, אפשר להסיק כי הזרימה בקשת (u,v) ירדה במהלך המעבר מפונקצית הזרימה f לפונקצית הזרימה f'. במצב זה אפשר להסיק כי הקשת (v,u) נמצאת על מסלול השיפור שהאלגוריתם בחר במעבר מן הזרימה f לזרימה f'.
מאחר שהאלגוריתם בוחר את מסלולי השיפור על ידי ביצוע BFS, אפשר להסיק כי הקשת (v,u) נמצאת על מסלול קצר ביותר מן הצומת s אל הצומת u ברשת השיורית G_f.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

מכאן נובע כי

$$\delta_f(s,u)=\delta_f(s,v)+1$$

במקרה זה מתקיים:

$$\begin{aligned}\delta_f(s,v) &= \delta_f(s,u)-1\\ &\le \delta_{f'}(s,u)-1\\ &\le \delta_{f'}(s,v)-2\\ &< \delta_{f'}(s,v)\end{aligned}$$

סתירה.

מאחר שהגענו לסתירה, אפשר להסיק כי לכל $v\in V$ מתקיים

$$\delta_f(s,v)\le\delta_{f'}(s,v)$$

מש״ל

© כל הזכויות שמורות לפרופסור עמוס ישראלי

משפט

מספר מסלולי השיפור באלגוריתם אדמונדס קרפ הוא לכל היותר $\Theta(|V||E|)$.

הוכחה

קשת (u,v) ברשת השיורית G_f היא **קריטית (צואר בקבוק)** במסלול שיפור **קצר ביותר** p' אם לאחר השיפור היא נמחקת מן הקשת השיורית.
בכל איטרציה של האלגוריתם מוצאים מסלול שיפור ובכל מסלול שיפור יש לפחות קשת קריטית אחת.
אנו נחסום את מספר האיטרציות שהאלגוריתם מבצע (ואת סיבוכיות

© כל הזכויות שמורות לפרופסור עמוס ישראלי

האלגוריתם) על ידי חישוב חסם על מספר הפעמים בהם כל קשת יכולה להיות קריטית.

תהי $E \in (u,v)$ קשת כלשהי ברשת ותהי G_f הרשת השיורית בה הקשת (u,v) היא קריטית בפעם הראשונה.

במצב זה מתקיים :

$.\delta_f(s,v) = \delta_f(s,u) + 1$

כל קשת שהיתה קריטית **מפסיקה להופיע ברשת השיורית** עד שהזרימה בה **יורדת.** (כתוצאה מהזרמה לאחור).

תהי $G_{f'}$ הרשת השיורית שבה מתרחש השיפור בעקבותיו הקשת (u,v) מופיעה שוב ברשת השיורית.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

בתנאים אלה הקשת (v,u) מופיעה על מסלול השיפור המחושב על ידי האלגוריתם במהלך החישוב על $G_{f'}$,

ומתקיים $.\delta_{f'}(s,u) = \delta_{f'}(s,v) + 1$

מאחר שלפי הלמה

$\delta_f(s,v) \le \delta_{f'}(s,v)$

נקבל

$$\begin{aligned} \delta_{f'}(s,u) &= \delta_{f'}(s,v) + 1 \\ &\ge \delta_f(s,v) + 1 \\ &= \delta_f(s,u) + 2 \end{aligned}$$

נקבל כי בכל פעם ש (u,v) הופכת להיות קריטית המרחק של u מן המקור גדל **לפחות** ב2.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

מכאן : הקשת (u,v) יכולה להיות קריטית לכל היותר $\Theta(|V|)$ פעמים.

מכיון שהחשבון נכון לכל קשת ובכל מסלול שיפור לפחות קשת אחת הופכת להיות לא קריטית. מספר מסלולי השיפור האפשרי הוא $.O(|E||V|)$

מש״ל.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

הרצאה 12: זיווג בגרף דו-צדדי

(Matching in Bi Partite Graphs)

גרף דו צדדי הוא גרף $B=(V,E)$ המקיים:

אפשר לחלק את הקבוצה V לשתי קבוצות זרות U ו- W וכל קשת בגרף מחברת בין צומת בקבוצה U, לבין צומת בקבוצה W. במלים אחרות:

1. $V=U\cup W$, $U\cap W=\phi$.
2. לכל $e\in E$,

$e=(u,w)$, $u\in U$, $w\in W$

© כל הזכויות שמורות לפרופסור עמוס ישראלי

זיווג (Matching)

זיווג בגרף $G=(V,E)$ הוא קבוצת קשתות $M\subseteq E$ אשר אינן חולקות אף צומת משותף.

זיווג מכסימום

(Maximum Matching)

הוא זיווג שמספר הקשתות בו הוא מרבי.

זיווג מכסימלי (Maximal Matching)

הוא זיווג שאי אפשר להוסיף לו אף קשת מהגרף.

שימו לב להבדל בין שתי ההגדרות.

בהרצאה זו נציג אלגוריתם לחישוב זיווג מכסימום בגרף דו-צדדי.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

האלגוריתם שנציג, משתמש באלגוריתם זרימה כשגרה.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

קיבולים אפקטיביים ומסלולי שיפור

תהי $G(V,E,c,s,t)$ רשת זרימה ותהי f פונקציית זרימה. לכל קשת $e\in E$, **הקיבול האפקטיבי של** e שווה לכמות הזרימה שאפשר להוסיף ל- e כלומר:

$$c(e)-f(e)$$

קשת רוויה היא קשת שקיבולה האפקטיבי הוא 0.

מסלול שיפור הוא מסלול מכוון מ- s אל t שבו אין אף קשת רוויה.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

שיפור זרימה לאורך מסלול שיפור

נניח כי f היא פונקצית זרימה, α הוא מסלול שיפור וv הוא צומת ביניים בα. תהינה $e_1=\overline{(u,v)}$ ו $e_2=\overline{(v,w)}$ קשתות בα. הזרימה f מקיימת את **חוק הצומת** ולכן, כמות הזרימה הנכנסת אל v שווה לכמות הזרימה היוצאת ממנו.

אם נוסיף לזרימה בe_1 ובe_2 אותה כמות זרימה, עבור v, חוק הצומת ישמר. מכאן, אם נוסיף את אותה כמות הזרימה, לכל הקשתות בα, חוק הצומת ישמר בכל צמתי הרשת.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

שיפור זרימה לאורך מסלול שיפור

בהנתן פונקצית זרימה f ומסלול שיפור α, אפשר **לחזק** את הזרימה f, על ידי הגדלת הזרימה בכמות שווה בכל אחת מקשתות המסלול α.

צואר הבקבוק של מסלול שיפור היא הקשת שקיבולה האפקטיבי הוא מינימלי.

השיפור המירבי האפשרי לאורך α שווה לקיבול האפקטיבי של צואר הבקבוק. בדרך זו, צואר הבקבוק של α הופך לקשת רוויה.

מאחר שמסלול השיפור מחבר את s עם t, הזרימה הכללית ברשת עולה בקיבול של צואר הבקבוק של α.

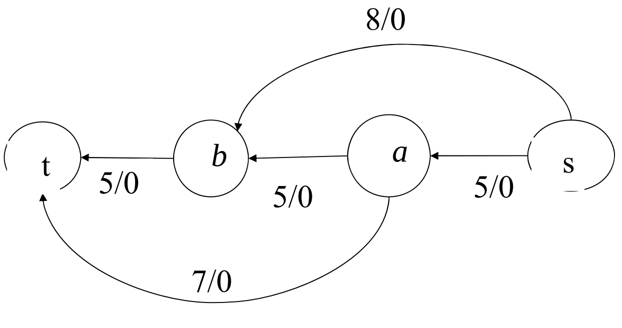
© כל הזכויות שמורות לפרופסור עמוס ישראלי

קשתות שימושיות לאחור

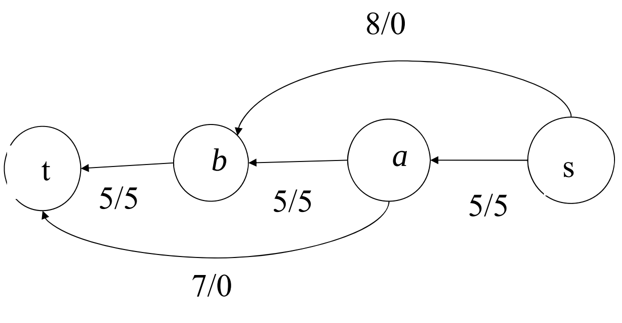
לעתים יתכן מצב בו אין ברשת אף מסלול שיפור ובעת ובעונה אחת, אין ברשת אף חתך רווי. מצב זה נגרם כאשר הזרימה ברשת נקבעה על ידי מספר מסלולי שיפור בדרך שבה בכל אחד מן המסלולים, בין s לבין t, יש קשת רוויה, אולם הקשתות הרוויות כולן, אינן משלימות חתך רווי.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

לדוגמא נתבונן ברשת הבאה:



תחילה נזרים 5 יחידות במסלול (s,a,bb,t) ונקבל:



© כל הזכויות שמורות לפרופסור עמוס ישראלי

המשך שיפור הסרימה

במצב זה אין ברשת חתך רווי, כלומר **הזרימה אינה מכסימלית**. מצד שני אין ברשת אף מסלול שיפור:

במסלול s,a,t, הקשת (s,a) רוויה.

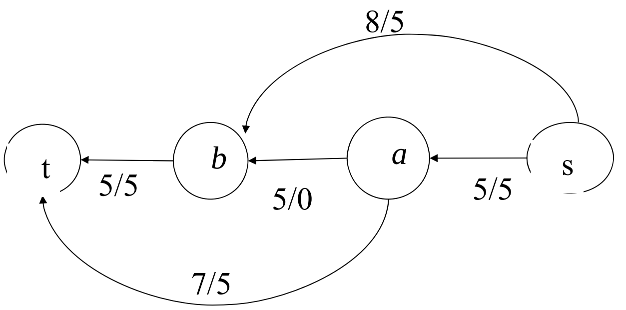
ובמסלול s,b,t הקשת (b,t) רוויה.

כדי לשפר זרימה אנו יכולים:

1. להזרים 5 יחידות בקשת (i,b).
2. נוצר עודף זרימה 5 בצומת b.
3. **להזרים לאחור** (להפסיק להזרים) 5 יחידות בקשת (a,b).
4. נוצר עודף זרימה 5 צומת a.
5. להזרים 5 יחידות בקשת (a,t).

ונקבל את המצב הבא:

© כל הזכויות שמורות לפרופסור עמוס ישראלי



במצב זה, החתך $(\{s,b\}:\{b,t\})$ הוא רווי (בדקו זאת) והזרימה ברשת מרבית.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

הסבר

הצלחנו לשפר זרימה, על ידי הורדת זרימה בקשת מסוימת.

עלינו למצוא דרך להתיחס גם אל שיפורי זרימה שבמהלכם בחלק מן הקשתות **מקטינים את הזרימה** ובלבד ש**הזרימה הכוללת** ברשת **תגדל**.

להלן נגדיר קשתות שימושיות לפנים ולאחור ואחר כך נגדיר **מסלול שיפור זרימה באופן פורמלי:**

© כל הזכויות שמורות לפרופסור עמוס ישראלי

קשתות שימושיות

קשת e תיקרא **שימושית לפנים** אם אפשר להגביר את הזרימה בה כלומר:

$$c(e) > f(e)$$

הקיבול האפקטיבי לפנים של e הוא

$$c(e) - f(e)$$

קשת e תיקרא **שימושית לאחור** אם אפשר להקטין את הזרימה בה כלומר:

$$f(e) > 0$$

הקיבול האפקטיבי לאחור של e הוא $f(e)$.

שימו לב: אם e היא קשת המקיימת $c(e) > f(e) > 0$ אז e היא גם שימושית לפנים וגם שימושית לאחור.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

מסלולי שיפור - הגדרה מחודשת

מסלול שיפור זרימה הוא סדרת צמתים $s, v_1, \ldots, v_{l-1}, t$. בין כל שני צמתים במסלול יש **קשת** ומתקיים:

1. הקשת הראשונה בסדרה, (s, v_1), יוצאת מן הצומת s.
2. הקשת האחרונה במסלול, (v_{l-1}, t), נכנסת אל הצומת t.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

3. כל קשת אחרת בסדרה (v_{i-1}, v_i) מקיימת:
   אם הקשת (v_{i-1}, v_i) מכוונת מהצומת v_{i-1} אל הצומת v_i אז היא **שימושית לפנים.**
   אם הקשת (v_{i-1}, v_i) מכוונת מהצומת v_i אל הצומת v_{i-1}. אז היא **שימושית לאחור**.

צואר הבקבוק של מסלול שיפור היא הקשת שקיבולה האפקטיבי הוא מינימלי.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

האלגוריתם של Ford ו Fulkerson

לכל קשת $e \in E$ קבע זרימה התחלתית $f(e) \leftarrow 0$

חזור

1. מצא מסלול שיפור מs אל t על ידי ביצוע BFS על כל הקשתות השימושיות.
2. שפר את הזרימה לאורך מסלול השיפור שמצאת.

עד שלא קיים מסלול שיפור נוסף.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

סיום האלגוריתם

אם הקיבולים והזרימות הם אך ורק מספרים טבעיים, אפשר להוכיח כי האלגוריתם מסתיים, שכן בכל איטרציה, הזרימה הכללית משתפרת במספר שלם. לאחר מספר סופי של איטרציות, מתקבלת זרימה כללית השווה לקיבול החתך המינימלי. זרימה כזו היא מירבית ואינה ניתנת לשיפור. במקרה זה, הסיבוכיות יכולה להיות תלויה בערך הקיבולים, ולא רק בגודל הרשת. אם מתירים קיבולים ממשיים כלשהם, סיום האלגוריתם, כפי שהוגדר עד כה, אינו מובטח.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

<u>נכונות - קיום החתך הרווי ומציאתו</u>

כדי להוכיח כי האלגוריתם נכון, עלינו להוכיח כי לאחר סיום האלגוריתם, הזרימה הכוללת היא מירבית.
טענה זו נובעת מיידית מן הלמה הבאה:

<u>למה</u>

לאחר סיום אלגוריתם Ford ו Fullkerson, יש ברשת חתך רווי.
שימו-לב: כפי שהוכחנו, הזרימה בחתך הרווי שווה לזרימה הכוללת בגרף.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

<u>הוכחה</u>

נבצע BFS על הרשת החל מs ונשתמש רק בקשתות שימושיות. לא יתכן שנגיע לt, כי אין ברשת מסלולי שיפור.
הקבוצה S מכילה את כל הצמתים שהגענו אליהם במהלך החיפוש. כל הקשתות מS אל $\bar{S}$ אינן שימושיות ולכן $(S:\bar{S})$ הוא חתך רווי.

מש״ל

© כל הזכויות שמורות לפרופסור עמוס ישראלי

<u>סיבוכיות</u>

נגדיר אורך של מסלול שיפור, α, כמספר הקשתות ב α.
קרפ (Karp) ואדמונדס (Edmonds) הוכיחו כי אפשר להבטיח את סיום האלגוריתם על ידי מציאת מסלולי השיפור, בסדר עולה של אורכיהם.
להלן נוכיח כי בדרך זו (על ידי הפעלת BFS על כל הקשתות השימושיות, סיבוכיות האלגוריתם היא

$$\Theta(|V||E|^2)$$

© כל הזכויות שמורות לפרופסור עמוס ישראלי

<u>הרשת השיורית (Residual Network)</u>

תהי $G=(V,E,c,s,t)$ רשת זרימה.
הרשת השיורית $G_f=(V,E_f,c_f,s,t)$ היא רשת זרימה המוגדרת על אותו גרף אולם קבוצת הקשתות E_f מכילה רק **קשתות שימושיות** והקיבול השיורי c_f מוגדר בהתאם.
בהנתן רשת שיורית $G_f=(V,E_f,c_f,s,t)$ נגדיר את **המרחק השיורי** $\delta_f(u,v)$ כמספר הקשתות במסלול מינימלי בין u לבין v.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

למה

אם מריצים את אלגוריתם אדמונדס קרפ אז לכל צומת $v \in V - s,t$, המרחק מ s אל v $\delta_f(u,v)$, גדל לאחר כל שיפור זרימה.

הוכחה

נניח בשלילה כי קיים צעד שיפור המשפר את הזרימה מ f ל f' וקיימים צמתים שהמרחק אליהם קטן, כלומר $\delta_f(s,v) > \delta_{f'}(s,v)$ עבור צומת כלשהו v.

תהי D קבוצת הצמתים שמרחקם השיורי מן הצומת s גדל ויהי v ב D שהמרחק השיורי אליו $\delta_{f'}(s,v)$ הוא מינימלי מבין הצמתים ב D.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

במצב זה מתקיים:

$$\delta_{f'}(s,u) < \delta_{f'}(s,v) \Rightarrow \delta_f(s,u) \le \delta_{f'}(s,u)$$

יהי p', מסלול מינימלי ב G' מ s אל v , ויהי u הצומת הקודם ל v במסלול p'. ברור כי $(u,v) \in E_{f'}$.

ולכן:

$\delta_{f'}(s,u) = \delta_{f'}(s,v) - 1$.

מן המסקנה הקודמת נובע כי $\delta_f(s,u) \le \delta_{f'}(s,u)$.

נתבונן במקרים הבאים:

© כל הזכויות שמורות לפרופסור עמוס ישראלי

מקרה 1: $f(u,v) < c(u,v)$

במקרה זה

$$\begin{aligned}\delta_f(s,v) &\le \delta_f(s,u)+1 \\ &\le \delta_{f'}(s,u)+1 \\ &= \delta_{f'}(s,v)\end{aligned}$$

סתירה.

מקרה 2: $f(u,v) = c(u,v)$

במקרה זה, הקשת (u,v) מופיעה ב E_f אך ורק כקשת שימושית לאחור.

מכאן נובע:

1. $(v,u) \in E_f$.
2. $(u,v) \notin E_f$.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

מאחר שנתון כי $(u,v) \in E_{f'}$, אפשר להסיק כי הזרימה בקשת (u,v) ירדה במהלך המעבר מפונקצית הזרימה f לפונקצית הזרימה f'. במצב זה אפשר להסיק כי הקשת (v,u) נמצאת על מסלול השיפור שהאלגוריתם בחר במעבר מן הזרימה f לזרימה f'.

מאחר שהאלגוריתם בוחר את מסלולי השיפור על ידי ביצוע BFS, אפשר להסיק כי הקשת (v,u) נמצאת על מסלול קצר ביותר מן הצומת s אל הצומת u ברשת השיורית G_f.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

מכאן נובע כי

$$\delta_f(s,u)=\delta_f(s,v)+1$$

במקרה זה מתקיים:

$$\begin{aligned}\delta_f(s,v)&=\delta_f(s,u)-1\\&\le\delta_{f'}(s,u)-1\\&\le\delta_{f'}(s,v)-2\\&<\delta_{f'}(s,v)\end{aligned}$$

סתירה.

מאחר שהגענו לסתירה, אפשר להסיק כי לכל $v\in V$ מתקיים

$$\delta_f(s,v)\le\delta_{f'}(s,v)$$

מש״ל

© כל הזכויות שמורות לפרופסור עמוס ישראלי

משפט

מספר מסלולי השיפור באלגוריתם אדמונדס קרפ הוא לכל היותר $\Theta(|V||E|)$.

הוכחה

קשת (u,v) ברשת השיורית G_f היא **קריטית (צואר בקבוק)** במסלול שיפור **קצר ביותר** p' אם לאחר השיפור היא נמחקת מן הקשת השיורית.

בכל איטרציה של האלגוריתם מוצאים מסלול שיפור ובכל מסלול שיפור יש לפחות קשת קריטית אחת.

אנו נחסום את מספר האיטרציות שהאלגוריתם מבצע (ואת סיבוכיות

© כל הזכויות שמורות לפרופסור עמוס ישראלי

האלגוריתם) על ידי חישוב חסם על מספר הפעמים בהם כל קשת יכולה להיות קריטית.

תהי $(u,v)\in E$ קשת כלשהי ברשת ותהי G_f הרשת השיורית בה הקשת (u,v) היא קריטית בפעם הראשונה.

במצב זה מתקיים:

$\delta_f(s,v)=\delta_f(s,u)+1$.

כל קשת שהיתה קריטית **מפסיקה להופיע ברשת השיורית** עד שהזרימה בה **יורדת.** (כתוצאה מהזרמה לאחור).

תהי $G_{f'}$ הרשת השיורית שבה מתרחש השיפור בעקבותיו הקשת (u,v) מופיעה שוב ברשת השיורית.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

בתנאים אלה הקשת (v,u) מופיעה על מסלול השיפור המחושב על ידי האלגוריתם במהלך החישוב על $G_{f'}$,

ומתקיים $\delta_{f'}(s,u)=\delta_{f'}(s,v)+1$.

מאחר שלפי הלמה

$$\delta_f(s,v)\le\delta_{f'}(s,v)$$

נקבל

$$\begin{aligned}\delta_{f'}(s,u)&=\delta_{f'}(s,v)+1\\&\ge\delta_f(s,v)+1\\&=\delta_f(s,u)+2\end{aligned}$$

נקבל כי בכל פעם ש (u,v) הופכת להיות קריטית המרחק של u מן המקור גדל **לפחות** ב2.

© כל הזכויות שמורות לפרופסור עמוס ישראלי

מכאן: הקשת (u,v) יכולה להיות קריטית לכל היותר $\Theta(|V|)$ פעמים. מכיון שהחשבון נכון לכל קשת ובכל מסלול שיפור לפחות קשת אחת הופכת להיות לא קריטית. מספר מסלולי השיפור האפשרי הוא $O(|E||V|)$.

מש״ל.

© כל הזכויות שמורות לפרופסור עמוס ישראלי