בית הספר למדעי המחשב המכללה האקדמית נתניה
סמסטר ב' תשע"ז

1.5.2017

מבני נתונים
פתרון תרגיל מס' 5

.1

בשאלה זו הסימון: i mod n

משמעותו: שארית החלוקה של i ב- n.

לדוגמה: 20 mod 8 שווה ל- 4

להלן תוכנית רקורסיבית בשם P1 שמקבלת פרמטרים מערך A ומספר k.

נתח/י את סיבוכיות זמן הריצה (במונחים של ה- O הקטן ביותר שניתן להשגה בשיטות שלמדנו) של התוכנית כתלות ב- n, כאשר n מציין את גודל המערך A שמועבר לפונקציה כפרמטר.

```
P1(A,k)
n = length(A)
if n == 1 return A[1]
if n == 2 return A[2]
  for (i = 1; i ≤  n²  ; i++) {
    A[i mod n ]=A[i+3 mod n]
 }
z=P1(A[1: ⌊2n/3⌋ ],k)· P1(A[1: ⌊3n/4⌋ ],k)
if A[ ⌊n/2⌋ ] ) ≤ k return z · P1(A[1: ⌊n/2⌋ ],k)
return P1( A[1: ⌊2n/3⌋ ],k ) + P1( A[ ⌊n/4⌋ : ⌊3n/4⌋ ],k)
```

פתרון שאלה 1

נסמן ב- $T(n)$ את זמן ביצוע התכנית כתלות ב- n, כאשר n מציין את גודל המערך שמועבר לפונקציה כפרמטר.

נשים לב שהאפשרות שתנאי ה- if לא מתקיים לוקחת יותר זמן מהאפשרות שהתנאי מתקיים. מאחר ואנחנו מנתחים סיבוכיות זמן במקרה הגרוע ביותר נניח שהקלט לתכנית הוא כזה שבכל שלב תנאי ה- if לא מתקיים. נוסחת הנסיגה היא לכן:

$$T(n) \le C_1 + C_2 \cdot n^2 + T(\lfloor 2n/3 \rfloor) + T(\lfloor 3n/4 \rfloor) + T(\lfloor 2n/3 \rfloor) + T(\lfloor 2n/4 \rfloor)$$

בשאלה זו לא נוכל להשתמש בשיטת המסטר ולכן נשתמש בשיטת עץ רקורסיה.

לכל $n>1$ קיים D כך ש-

$$C_1 + C_2 \cdot n^2 \le D \cdot n^2$$

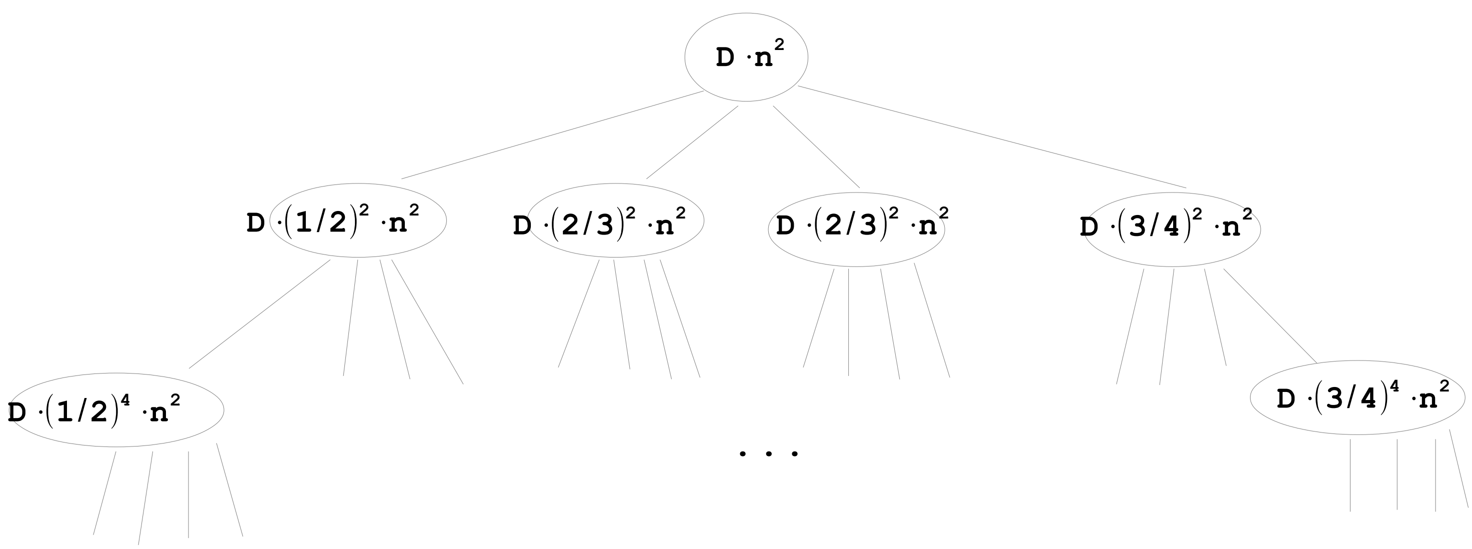
ולכן

$$(1)\quad T(n) \le D \cdot n^2 + T(2n/4) + T(2n/3) + T(2n/3) + T(3n/4)$$

ששקול ל-

$$(2)\quad T(n) \le T(n/2) + T(2n/3) + T(2n/3) + T(3n/4) + D \cdot n^2$$

עץ הרקורסיה המתאים לאי שוויון הנ"ל נראה כך:



נסמן ב- q את המספר שמתקבל מהחישוב הבא:

(3) $q=(1/2)^2+(2/3)^2+(2/3)^2+(3/4)^2$

סכום האיברים ברמה הראשונה בעץ קטן או שווה ל $D \cdot n^2$

סכום האיברים ברמה השניה בעץ קטן או שווה ל $D \cdot q \cdot n^2$

סכום האיברים ברמה השלישית בעץ קטן או שווה ל $D \cdot q^2 \cdot n^2$

וכן הלאה.

נסמן ב- x את מספר הרמות בעץ.

לכן נקבל ש-

$$(4) \quad T(n) \le D \cdot n^2 \cdot (1+q+q^2+\ldots+q^{x-1})$$

האיבר הימני ביותר ברמה 1 בעץ התקבל כתוצאה מהצבה של $T(n)$ בנוסחת הנסיגה.

האיבר הימני ביותר ברמה 2 בעץ התקבל כתוצאה מהצבה של $T(3n/4)$ בנוסחת הנסיגה.

האיבר הימני ביותר ברמה 3 בעץ התקבל כתוצאה מהצבה של $T((3/4)^2 n)$ בנוסחת הנסיגה.

...

האיבר הימני ביותר ברמה x בעץ התקבל כתוצאה מהצבה של $T((3/4)^{x-1} n)$ בנוסחת הנסיגה.

מאחר והאיבר הימני ביותר ברמה x בעץ גדול מ- 0 הוא התקבל כתוצאה מהצבה של מספר שגדול או שווה ל- 1 בנוסחת הנסיגה. ולכן:

$$(3/4)^{x-1} n \ge 1$$

ששקול ל-

$$(4/3)^{x-1} \le n$$

ששקול ל-

$$x-1 \le \log_{4/3} n$$

ששקול ל-

$$(5) \quad x \le 1+\log_{4/3} n$$

מ- (5) ו- (4) נקבל:

$$(6) \quad T(n) \le D \cdot n^2 \cdot (1+q+q^2+\ldots+q^{\log_{4/3} n})$$

ששקול ל-

$$(7)\qquad T(n)\le D\cdot n^2\cdot(q^{\log_{4/3}n+1}-1)/(q-1)$$

ולכן:

$$(8)\qquad T(n)\le D\cdot n^2\cdot(q^{\log_{4/3}n+1})/(q-1)$$

ששקול ל-

$$(9)\qquad T(n)\le D\cdot n^2\cdot q/(q-1)\cdot(q^{\log_{4/3}n})$$

ששקול ל-

$$(10)\qquad T(n)\le D\cdot n^2\cdot q/(q-1)\cdot(q^{\frac{\log_q n}{\log_q 4/3}})$$

ששקול ל-

$$(11)\qquad T(n)\le D\cdot n^2\cdot q/(q-1)\cdot(n^{\frac{1}{\log_q 4/3}})=\theta(n^{2+\frac{1}{\log_q 4/3}})$$

ולכן הראנו ש-

$$(12)\qquad T(n)=O(n^{2+\frac{1}{\log_q 4/3}})$$

כאשר q הוא הערך שחושב ב- (3).

.2

נגדיר מבנה נתונים – "רשימת רשימות" L בעל התכונות הבאות :

- L (בהמשך נקרא לה "הרשימה הראשית") היא רשימה מקושרת דו-כיוונית, רגילה לא ממוינת עם זנב. דהינו, head(L) מצביע לראש הרשימה הראשית L ולכל איבר x ברשימה הראשית L ישנם השדות הבאים: list(x),key(x),next(x),prev(x).
- לכל איבר x ברשימה הראשית L יש שדה list(x) שמצביע לרשימה מקושרת דו-כיוונית מעגלית לא ממוינת וללא זנב <u>לא ריקה</u> Q_x (הרשימה של האיבר x). כל איבר y ברשימה Q_x מכיל את השדות: key(y),key1(y),info(y),next(y),prev(y).
לרשימות מהסוג Q_x נקרא בהמשך "הרשימות המשניות". הערך של השדה key1(y) של איבר y ברשימה Q_x שווה לערך של key(x).
- האיברים ברשימה הראשית L מקימים את התכונה הבאה: הערך של המפתח של איבר x ברשימה הראשית L שווה לערך של השדה key1(y) עבור כל איבר y שנמצא ברשימה Q_x .

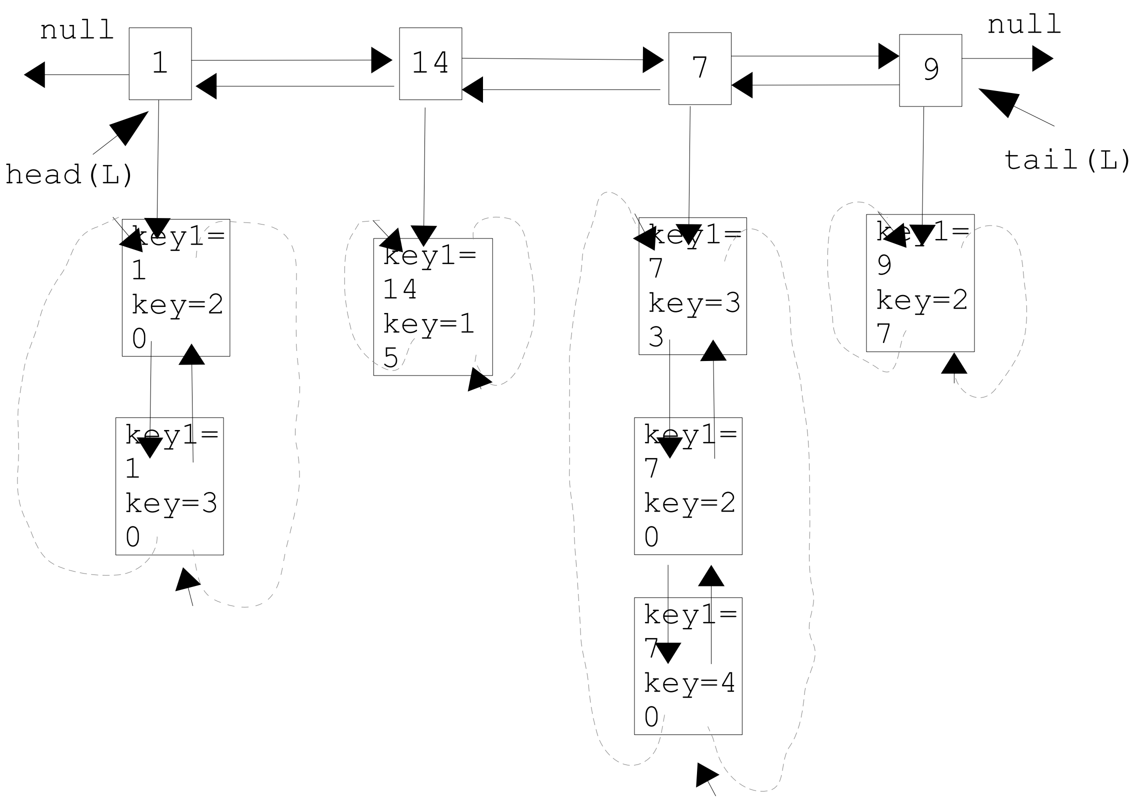
ניתן להניח שמפתחות כל האיברים ברשימה הראשית L שונים זה מזה וכן מפתחות כל האיברים ברשימות המשניות שונים זה מזה.

כתוב/י פסאודו-קוד של פונקציה בשם P4(L,k1,k2) יעילה ככל האפשר שמקבלת כפרמטרים רשימת רשימות L (כפי שהוגדר למעלה), ושני מספרים k1,k2 מחפשת את האיבר y שנמצא ברשימה משנית כלשהי ומקיים key1(y)=k1 ו- key(y)=k2. במידה והאיבר y לא קיים הפונקציה מודיעה על שגיאה. במידה והאיבר y קיים, הפונקציה מוציאה אותו מהרשימה המשנית בה הוא נמצא ומעבירה אותו לתחילתה רשימה המשנית Q_x של האיבר x שמקיים שהמפתח שלו גדול יותר מכל המפתחות של כל האיברים ברשימה הראשית.

נתח/י את סיבוכיות הזמן של הפונקציה P2 שכתבת כתלות ב- n ו- m כאשר n מציין את מספר האיברים ברשימה הראשית ו- m מציין את מספר האיברים ברשימה המשנית הגדולה ביותר מבין כל הרשימות המשניות.

מותר להשתמש בפונקציות עזר, אך יש לפרט את הפסיאודו קוד של פונקציות העזר.

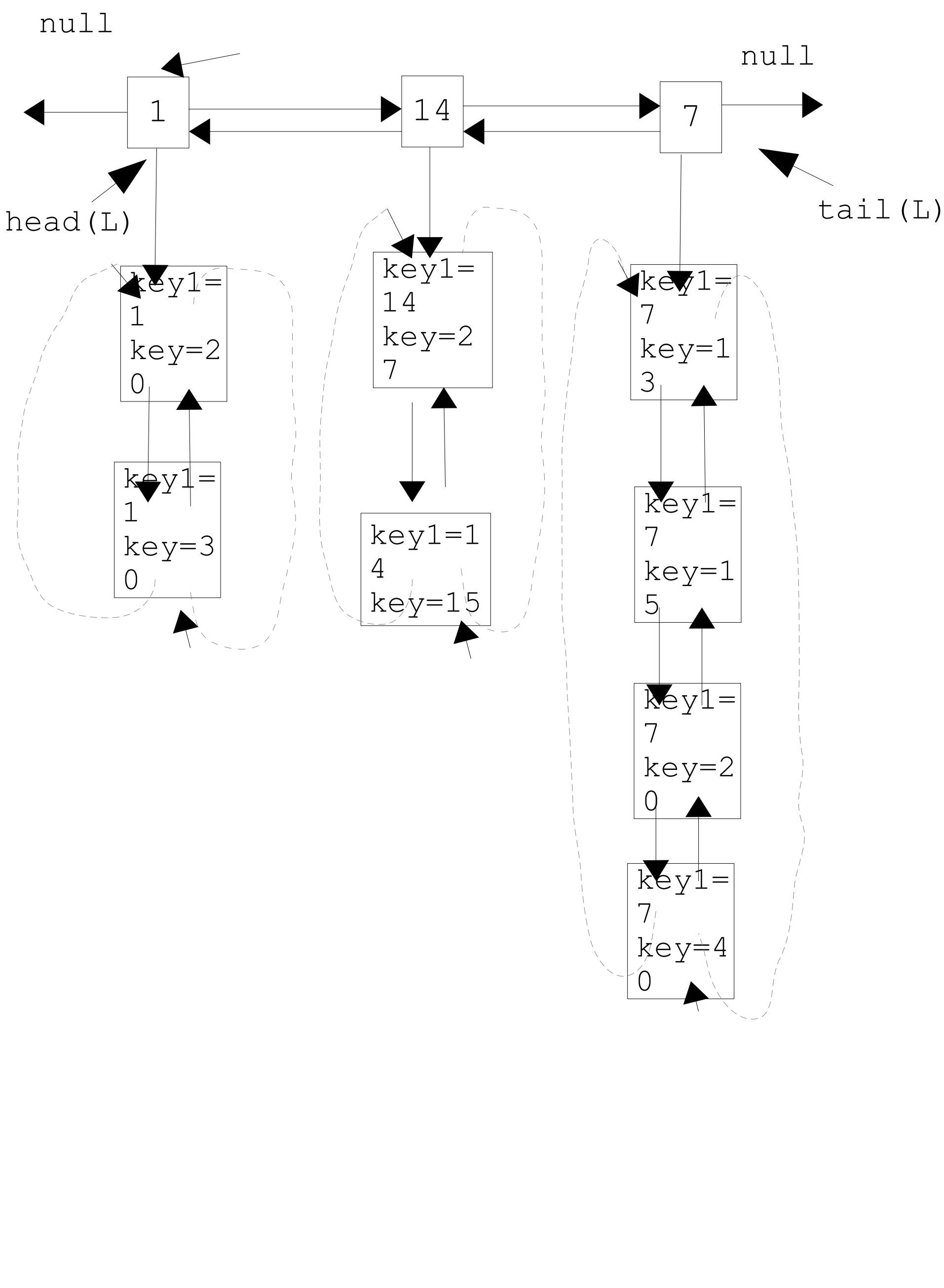
לדוגמה, נניח שנתונה רשימת הרשימות L הבאה:



לאחר הקריאה לפונקציה P2(L,4,11) תתקבל הודעה : איבר לא קיים.

לאחר הקריאה לפונקציה P2(L,9,27) האיבר שהמפתחות שלו הם key1=9 key=27 יוצא מהרשימה המשנית של האיבר שהמפתח שלו הוא 9 ויועבר לתחילת רשימה המשנית של האיבר שהמפתח שלו הוא 14. מאחר ועבור האיבר ברשימה הראשית שהמפתח שלו הוא 9 אין איברים ברשימה המשנית, הוא יוצא מהרשימה הראשית.

הרשימה שתתקבל לאחר הפעולה הנ"ל היא:


null
null
1
14
7
head(L)
tail(L)
key1=
1
key=2
0
key1=
14
key=2
7
key1=
7
key=1
3
key1=
1
key=3
0
key1=1
4
key=15
key1=
7
key=1
5
key1=
7
key=2
0
key1=
7
key=4
0

פתרון שאלה 2

```
P2(L,k1,k2)
x=head(L)
if (x==NULL) { print "element does not exist"; return }
while true {
 if (key(x)==k1) { P2.1(L,k1,k2,x); return}
 if (next(x)==NULL) { print "element does not exist"
                          return }
 x-next(x)
 }

P2.1(L,k1,k2,x)
z=list(x)
z1=list(x)
while true {
 if (key(z)==k2) { P4.2(L,k1,k2,x,z); return}
 if (next(z)==z1) { print "element does not exist"
                     return }
 z=next(z)
 }
```

```
P2.2(L,k1,k2,x,z)
w=max(L)
delete_list2(Qx,z)
if (w==x) {
  insert_list2(Qx,z)
  return
}
if (head(Qx)==NULL) {delete_list1(L,x)}
t=new element of secondary linked list
key1(t)=key(w)
key(t)=k2
insert_list2(Qw,t)

max(L)
max=head(L)
max_key=key(max)
x=head(L)
while true {
 if (key(x)>max_key) { max=x; max_key=key(x)}
 if (next(x)==NULL) { return max }
 x=next(x)
 }

insert_list2(L,s)
x=head(L)
if (x==NULL) {head(L}=s; next(s)=s; prev(s)=s;
              return}
head(L)=s
next(s)=x
prev(s)=prev(x)
```

delete_list1(L,s)

```
if (head(L)==s) && (tail(L)==s) {
   head(L)=null; tail(L)==null; return
   }
if (head(L)==s) { head(L)=next(s) }
if (tail(L)==s) { tail(L)=prev(s) }
prev(next(s))=prev(s);next(prev(s))=next(s)
```

delete_list2(L,s)

```
if (next(s)==s) {
   head(L)=null; return
   }
if (head(L)==s) { head(L)=next(s) }
prev(next(s))=prev(s); next(prev(s))=next(s)
```