חוברת תרגילים מבני נתונים

חוברה ע"י אסתי שטיין

תודות לאיליה זלדנר על העזרה בחלק מהפתרונות

פרק ראשון – סיבוכיות, מחסניות, תורים, מערכים ורשימות מקושרות

שאלה 1

הניחו ש – F ו- G הן פונקציות חיוביות. הוכיחו או הפריכו את הטענות הבאות:

$f(n) = O(g(n)) \rightarrow g(n) = O(f(n))$

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

פיתרון: **הניחו ש G ו F הן פונקציות חיוביות .הוכיחו או הפריכו את הטענות הבאות**
$f(n) = O(g(n)) \rightarrow g(n) = O(f(n))$
$f(n) = \Theta(f(n/2))$

1.*a*.

$$\text{if } f(n) = O(g(n)) \text{ then } \exists\ c_0\ n_0, \forall n \geq n_0\ f(n) \leq c_0 \cdot g(n) \rightarrow \frac{f(n)}{g(n)} \leq c_0$$

$$\text{if } g(n) = O(f(n)) \text{ then } \exists\ c_0\ n_0, \forall n \geq n_0\ g(n) \leq c_1 \cdot f(n) \rightarrow \frac{g(n)}{f(n)} \leq c_1$$

$\Downarrow$

$$false \quad for\ example: \lim_{n\to\infty} f(n) = C \ and \lim_{n\to\infty} g(n) = \infty \rightarrow \lim_{n\to\infty} \frac{f(n)}{g(n)} \leq c_0 \ and \lim_{n\to\infty} \frac{g(n)}{f(n)} = \infty$$

1.*b*.

$$\text{if } f(n) = \theta(f(n/2)) \text{ then } \exists\ c_0\ c_1\ n_0, \forall n \geq n_0\ c_1 \cdot g(n) \leq f(n) \leq c_0 \cdot g(n)$$

$\Downarrow$

$$\text{false for example: } f(n) = 2^n \rightarrow f(\frac{n}{2}) = 2^{\frac{n}{2}} \text{ then } \exists\ c_0\ c_1\ n_0, \forall n \geq n_0\ c_1 \cdot 2^{\frac{n}{2}} \leq 2^n \leq c_0 \cdot 2^{\frac{n}{2}}$$

$$\rightarrow \sqrt[\frac{n}{2}]{c_1} \cdot 2 \leq 2^{\frac{n}{2}} \leq \sqrt[\frac{n}{2}]{c_0} \cdot 2 \rightarrow 2 \leq 2^{\frac{n}{2}} \leq 2 \ and\ it's\ false$$

שאלה 2

לכל זוג פונקציות (F, G) , יש להוכיח האם F מהווה O, Ω, Θ של G. יש לשים לב, כי יותר מתשובה אחת יכולה להיות נכונה.

- $(F, G) = (\sqrt{n}, \log_2 n)$
- $(F, G) = (\log^3_2 n, \log_2 n^3)$
- $(F, G) = (2^n, 2^{n/2})$
- $(F, G) = (\log n!, n \log_2 n)$

פיתרון: **לכל זוג פונקציות (F, G) ,יש להוכיח האם F מהווה Θ, Ω, O של G . יש לשים לב כי יותר מתשובה אחת יכולה להיות נכונה.**

$2.a.\left(\mathrm{F},\ \mathrm{G}\right) = (\sqrt{\mathrm{n}},\ \log_2 \mathrm{n})$

$\exists c_0\ n_0\ \forall n \geq n_0\ \sqrt{\mathrm{n}} \geq c_0 \cdot \log_2 \mathrm{n} \rightarrow \mathrm{F} = \Omega(\mathrm{G})$

$for\ example: c_0 = 1, n_0 = 2 \rightarrow \lim_{n\to\infty} \frac{\sqrt{\mathrm{n}}}{\log_2 \mathrm{n}} \geq 1$

$2.b.\left(\mathrm{F},\ \mathrm{G}\right) = (\ \log^3{}_2 \mathrm{n},\ \log_2 \mathrm{n}^3)$

$\exists c_0\ n_0\ \forall n \geq n_0\ \log^3{}_2 \mathrm{n} \geq c_0 \cdot \log_2 \mathrm{n}^3 = 3c_0 \cdot \log_2 \mathrm{n} \rightarrow \mathrm{F} = \Omega(\mathrm{G})$

$for\ example: c_0 = 1, n_0 = 2 \rightarrow \lim_{n\to\infty} \log^2{}_2 \mathrm{n} \geq 1$

$2.c.\left(\mathrm{F},\ \mathrm{G}\right) = (\ 2^{\mathrm{n}},\ 2^{\frac{\mathrm{n}}{2}})$

$\exists c_0\ n_0\ \forall n \geq n_0\ 2^{\mathrm{n}} \geq c_0 \cdot 2^{\frac{\mathrm{n}}{2}} \rightarrow \mathrm{F} = \Omega(\mathrm{G})$

$for\ example: c_0 = 1, n_0 = 2 \rightarrow \lim_{n\to\infty} 2^{\frac{\mathrm{n}}{2}} \geq 1$

$2.d.\left(\mathrm{F},\ \mathrm{G}\right) = (\ \log \mathrm{n}!,\ \mathrm{n}\log_2 \mathrm{n}\ )$

$\exists c_0\ c_1\ n_0\ \forall n \geq n_0\ c_0 \cdot \mathrm{n}\log_2 \mathrm{n} \leq \log \mathrm{n}! \leq c_1 \cdot \mathrm{n}\log_2 \mathrm{n} \rightarrow \mathrm{F} = \theta(\mathrm{G})$

proof :

$1.\log \mathrm{n}! = \sum_1^n \log(i) \leq \sum_1^n \log(n) = \mathrm{n}\log_2 \mathrm{n} \rightarrow \mathrm{F} = O(\mathrm{G})$

$2.\log \mathrm{n}! = \sum_1^n \log(i) \geq \sum_{\frac{n}{2}}^n \log(i) \geq \sum_{\frac{n}{2}}^n \log(\frac{n}{2}) = \frac{n}{2}\log(\frac{n}{2}) = \frac{1}{2}\mathrm{n}\log_2 \mathrm{n} - \frac{1}{2}\mathrm{n}\log(2) = O(n\log n) \rightarrow \mathrm{F} = \Omega(\mathrm{G})$

$\rightarrow \mathrm{F} = \theta(\mathrm{G})$

שאלה 3

פתרו את המשוואה הבאה:

$T(1) = 1,\ T(n) = T(n/2) + O(n),\ \forall\ n > 1.$

פיתרון:

$$T(1) = 1,\ T(n) = T\left(\frac{n}{2}\right)+O(n), \forall n > 1.$$

$$1 = \frac{n}{2^k} \rightarrow k = \log n$$

$$T(n) = T\left(\frac{n}{2^1}\right)+O\left(\frac{n}{2^0}\right) = T\left(\frac{n}{2^2}\right)+O\left(\frac{n}{2^1}\right)+O\left(\frac{n}{2^0}\right) =$$

$$T(1)+\sum_{i=0}^{\log n-1} O\left(\frac{n}{2^i}\right) = T(1)+\sum_{i=0}^{\log n-1} C_i \frac{n}{2^i} \le T(1)+\sum_{i=0}^{\log n-1} C_{\max} \frac{n}{2^i} =$$

$$1+C_{\max}\cdot n\cdot \sum_{i=0}^{\log n-1}\frac{1}{2^i} = 1+C_{\max}\cdot n\cdot\left(\frac{\frac{1}{2}^{\log n}-1}{\frac{1}{2}-1}\right) = 1+C_{\max}\cdot n\cdot 2 = O(n)$$

שאלה 4

נתונה מחסנית S עם n איברים בתוכה, ונתון תור Q ריק בהתחלה.

יש לתאר אלגוריתם המשתמש ב- Q בכדי לבדוק האם איבר x נמצא בתוך המחסנית S. המגבלה היא שבסיום, המחסנית S צריכה להיות מסודרת באותו אופן כמו לפני תחילת האלגוריתם.

מותר להשתמש רק ב מחסנית S ובתור Q. אין להוסיף מבני נתונים נוספים, אלא רק משתנים בודדים אם יש צורך.

יש לבצע את האלגוריתם ביע ילות הגבוהה ביותר האפשרית, לחשב את הסיבוכיות ולהצדיק אותה מבחינת האלגוריתם.

פיתרון:

```
int counter = 0, flag = 0;
while (!is_empty())
{
	temp = pop();
	counter++;
	enqueue(temp);
	if (x == temp)
	{
		flag = 1;
		break;
	}
}
for (int i=0; i< counter;i++)
	push(dequeue());
for (int i=0; i < counter; i++)
	enqueue(pop());
for (int i=0; i< counter;i++)
	push(dequeue());
if (!flag)
	printf("no X");
else
	printf("X exist ");
```

סיבוכיות זמן : $O(x)+O(x)+O(x)+O(x)=O(4x)=O(x)$
$if\ x \to n\ then\ O(n)$

סיבוכיות מקום נוסף:הוספנו משתנים בודדים + ניצלנו כל פעם במקום שכבר היה שמור לפני כדי לשנות סדר של המשתנים $O(1)$.

שאלה 5

א. תארו אלגוריתם רקורסיבי לחישוב מספר התמורות של המספרים { 1, 2, 3, .. n}, וחשבו את הסיבוכיות.

הכוונה היא ליצור מונה ולאפס אותו, לייצר את כל התמורות האפשריות, ולהגדיל את המונה ב-1 לכל תמורה חדשה שייצרנו. הערך שיקבל המונה, הינו מספר התמורות.

ב. תארו אלגוריתם לא- רקורסיבי לחישוב מספר התמורות של המספרים { 1, 2, 3, .. n}, וחשבו את הסיבוכיות.

פיתרון:

א. ברור לנו שבעצם ביקשו לחשב : $P(n,n)=\frac{n!}{(n-n)!}=n!$

```
int p(n)
{
      return n*p(n-1);
}
```

סיבוכיות זמן : $O(n!)$

סיבוכיות מקום נוסף:רקורסיה שומרת את הכתובת חזרה של כל הרצה של פונקציה $O(n!)$.

ב.

```
int counter = 0;
while (n>0)
{
   counter = counter*n;
   n--;
}
```

סיבוכיות זמן : $O(n!)$

סיבוכיות מקום נוסף:הפעם אין כתובות שצריך לשמור $O(1)$.

שאלה 6

דרגו את הפונקציות הבאות על פי סדר הגדילה שלהן , מהנמוכה עד הגבוהה אסימפטוטית ע "פ $\Theta(n)$. אם יש שתי פונקציות או יותר שוות אנא ציינו זאת.

$\sqrt{2}^{\log n}$	$11n^7-n^4+12n^3-8n$	$\log_8(n^2)$	n^2	$\sqrt{n}$
n^7	$(\log n)^2$	$\log(n!)$	2^{2^n}	$n^{1/\log n}$
$\ln \log n$	$n2^n$	n^4+12n^3-8n	$\ln n$	2^n

פיתרון:

$$n^{\frac{1}{\log n}} = m \to \log(m) = \frac{1}{\log n}\log n = 1 \to m = 2 = \theta(1)$$

$$\log_8(n^2) = \theta(\log(n))$$

$$\ln(\log(n)) = \theta(\log(\log(n)))$$

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

$$\sqrt{n} = \theta(\sqrt{n}), \sqrt{2}^{\log n} = \sqrt{2^{\log n}} = \sqrt{n} = \theta(\sqrt{n})$$

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

$$n^2 = \theta(n^2)$$

$$n^4+12n^3-8n = \theta(n^4)$$

$$n^7 = \theta(n^7), 11n^7-n^4+12n^3-8n = \theta(n^7)$$

$$2^n = \theta(2^n)$$

$$n2^n = \theta(n2^n)$$

$$2^{2^n} = \theta(2^{2^n})$$

שאלה 7

הוכיחו או הפריכו:

a. $\Theta(n^4)+ O(n)= \Theta(n^4)$
b. $\Theta(n^3)-O(n^2)= \Theta(n^3)$
c. $\Omega(\log n)+ \Omega(\log \log n)= \Omega(\log^2 n)$
d. $\sqrt{n} +\log n = O(\log n)$
e. $\log \sqrt{n} = \Omega(\log n)$

פיתרון:

$a.\theta(n^4)+O(n)=\theta(n^4)$

$if\ (\theta(n^4)+O(n)=\theta(n^4))\ then\ \exists\ c_0,c_1,n_0,\forall n\geq n_0\ \ c_1\cdot n^4\leq\theta(n^4)+O(n)\leq\ c_0\cdot n^4$

$\rightarrow c_1\leq\lim_{n\to\infty}\left(\frac{\theta(n^4)}{n^4}+\frac{O(n)}{n^4}\right)=\theta(1)+0\ \leq\ c_0\Rightarrow true$

$b.\theta(n^3)-O(n^2)=\theta(n^3)$

$if\ (\theta(n^3)-O(n^2)=\theta(n^3))\ then\ \exists\ c_0,c_1,n_0,\forall n\geq n_0\ \ c_1\cdot n^3\leq\theta(n^3)-O(n^2)\leq\ c_0\cdot n^3$

$\rightarrow c_1\leq\lim_{n\to\infty}\left(\frac{\theta(n^3)}{n^3}-\frac{O(n^2)}{n^3}\right)=\theta(1)-0\ \leq\ c_0\Rightarrow true$

$c.\Omega(\log n)+\Omega(\log(\log(n)))=\Omega(\log^2 n)$

$if\ (\Omega(\log n)+\Omega(\log(\log(n)))=\Omega(\log^2 n))\ then\ \exists\ c_0,n_0,\forall n\geq n_0\ \ c_1\cdot\log^2 n\leq\Omega(\log n)+\Omega(\log(\log(n))$

$\rightarrow c_1\leq\lim_{n\to\infty}\left(\frac{\Omega(\log n)}{\Omega(\log^2 n))}+\frac{\Omega(\log(\log(n)))}{\Omega(\log^2 n))}\right)=0+0\Rightarrow false$

$d.\sqrt{n}+\log(n)=O(\log(n))$

$if\ (\sqrt{n}+\log(n)=O(\log(n)))\ then\ \exists\ c_0,n_0,\forall n\geq n_0\ \ \sqrt{n}+\log(n)\leq\ c_0\cdot\log(n)$

$\rightarrow\lim_{n\to\infty}\left(\frac{\sqrt{n}}{\log(n)}+\frac{\log(n)}{\log(n)}\right)=\infty+1\ \leq\ c_0\Rightarrow false$

$e.\log(\sqrt{n})=\Omega(\log n)$

$if\ (\log(\sqrt{n})=\Omega(\log n))\ then\ \exists\ c_0,n_0,\forall n\geq n_0\ \ c_0\cdot\log(n)\leq\log(\sqrt{n})$

$\rightarrow c_0\leq\lim_{n\to\infty}\left(\frac{\log(\sqrt{n}}{\log(n)}\right)=\frac{1}{2}\ \ \Rightarrow true$

שאלה 8

מצאו את T(n) בכל אחת מנוסחאות הנסיגה שלהלן . מצאו חסמים הדוקים ככל שתוכלו ונמקו תשובתכם. נתון כי $T(1)=\Theta(1),\ T(2)=\Theta(1)$

א. $T(n)=T(\sqrt{n})+1$

$a.T(n)=T(\sqrt{n})+1$

$n=2^m\rightarrow m=\log(n)\rightarrow T(2^m)=T(2^{\frac{m}{2}})+1$

$S(m)=T(2^m)\rightarrow S(m)=S(\frac{m}{2})+1$

$k=\log(m)\rightarrow S(m)=1+\sum_{i=0}^{\log m-1}1=\log m$

$T(n)=T(2^m)=S(m)=\log m\rightarrow T(n)=\log(\log(n))$

ב. $T(n)=2T(n-1)+O(1)$

$$b.T(n) = 2T(n-1) + O(1)$$

$$k = n \rightarrow T(n) = 2^n T(1) + \sum_{i=0}^{n-1} O(1) \cdot 2^i$$

$$= 2^n + \sum_{i=0}^{n-1} c_i \cdot 2^i \le 2^n + \sum_{i=0}^{n-1} c_{\max} \cdot 2^i$$

$$= 2^n + c_{\max} \sum_{i=0}^{n-1} 2^i = 2^n + c_{\max} \left(\frac{2^n - 1}{2-1} \right) = O(2^n)$$

ג. T(n) = T(n - 1) + n

$$c.T(n) = T(n-1) + n$$

$$k = n \rightarrow T(n) = T(1) + \sum_{i=0}^{n} i$$

$$= \frac{(n+1)n}{2} = O(n^2)$$

ד. T(n) = 2T(n/2) + n/log

$$d.T(n) = 2 \cdot T(\frac{n}{2}) + \frac{n}{\log n}$$

$$k = \log(n) \rightarrow T(n) = 2^{\log(n)} T(1) + \sum_{i=0}^{\log n - 1} \frac{\frac{n}{2^i}}{\log(\frac{n}{2^i})} 2^i$$

$$= n + n \cdot \sum_{i=0}^{\log n - 1} \frac{1}{\log(\frac{n}{2^i})} = n + n \cdot \sum_{i=0}^{\log n - 1} \frac{1}{\log(n) - i} = n + n \cdot \sum_{i=1}^{\log n} \frac{1}{i} = n + n \cdot \left(\int_1^{\log n} \frac{1}{t} \cdot dt + 1 \right) = n + n \cdot \log\log(n) = \theta(n \log(\log(n)))$$

<u>שאלה 9</u>

אפרים שהינו סטודנט במעבדה במבנה נתונים , קיבל לידיו רשימה מקושרת לינארית . ברשימה היו שני סוגי רשומות "שחורות" ו"לבנות". אפרים נדרש להחליט אם יש יותר "לבנות" או "שחורות". לצורך כך, קיבל תקציב מהמכללה בסך 4000 ₪. אפרים החליט לקנות מחשב במחיר 500 ש"ח בכדי שיוכל להשתמש ביתרת כספו לבילויים. אי לכך קיבל מחשב שאינו מסוגל לעשות פעולות אריתמטיות . מלבד זה היה המחשב תקין לגמרי . האם יספיק אפרים לבלות, אחרי שיסיים את התכנית?

כתבו אלגוריתם בשם (Node * L) MajorColor, אשר מקבלת את הרשימה ומחזירה את הצבע שהוא הרוב. האלגוריתם צריך להתבצע בזמן לינארי , ולרוץ על המחשב של אפרים . הניחו שכשנתון מצביע לרשומה Node, החזרת Node.color תדרוש זמן קבוע.

שאלה 10

מה סיבוכיות הזמן של הקטעים הבאים (במושגים של Θ)? נמקו.

א.

```
int f1(int n)
{
 int i,  result = 1
   for (i = 0; i < n; i++  )
         result *= x;
   return result;
}
```

ב.

```
int f2(int n)
{
   if( n == 0)
         return 1;
   if( n % 2 == 0)
           return f2(n/2)* f2(n/2);
   else
           return x*f2(n-1);
```

ג.

```
int f3(int n)
{
   if( n == 0)
         return 1;
   if( n % 2 == 0) {
           tmp = f3(n/2);
           return tmp*tmp;
           }
   else
           return x*f3(n-1);
```

שאלה 11

יש להוכיח או להפריך את הטענה הבאה:

קיימת פונקציה f כך ש $f(n) = \Omega(\log n)$ וגם $(f(n))^2 = O(f(n))$.

לא נכון. נניח בשלילה כי קיימים קבועים c, n_0 כך שלכל $n \geq n_0$ מתקיים

$$(f(n))^2 \leq c\, f(n)$$

אזי לכל $n \geq n_0$ מתקיים $f(n) \leq c$ בסתירה לכך ש- $f(n) = \Omega(\log n)$.

שאלה 12

יהי $P(n)$ פולינום ממעלה k, כלומר $P(n) = a_0 + a_1 n + a_2 n^2 + \ldots + a_k n^k$, כאשר k והמקדמים a_i הם קבועים, המקדם המוביל a_k הוא חיובי, ושאר המקדמים עשויים להיות חיוביים , שליליים או אפס. יש להוכיח כי $P(n) = \Theta(n^k)$.

להוכחת $P(n) = O(n^k)$ ניקח, למשל: $|a_0| + |a_1| + |a_2| + \ldots + |a_{k-1}| + a_k$,$c = 1$,$n_0 = 1$.

להוכחת $P(n) = \Omega(n^k)$ ניקח, למשל: $c = a_k/2$,$n_0 = 2(|a_0| + |a_1|+ |a_2| + \ldots + |a_{k-1}|)/a_k$.
כאשר c ו-n_0 לעיל הם כפי שמופיעים בהגדרות של O ו-Ω.

<u>שאלה 13</u>

יש למצוא חסם עליון אסימפטוטי (O גדול) הדוק ככל שינתן עבור נוסחאות הנסיגה הבאות. יש להציג את דרך הפתרון במלואה.

א. $T(n) = T(n/2) + \log n$

באמצעות שיטת האיטרציה ניתן להראות כי

$T(n) = T(n/2^i) + \log(n/2^{i-1}) + \ldots + \log(n/2)+ \log n = T(n/2^i) + i \log n – (1 + \ldots + (i-1)) = T(n/2^i) + i \log n – i(i-1)/2$

נציב $i = \log n$ (מותר להניח כי i הוא חזקה של 2) ונקבל

$T(n) = T(1) + \log^2 n – (\log n)(\log n – 1)/2 = O(\log^2 n)$

(במקרה זה היה ניתן להגיע לאותו סדר גודל על ידי חסימה נאיבית: $\log n <= \log(n/2^{i-1})$.)

ב. $T(n) = nT(n/2) + 1$

נוכיח באינדוקציה (שיטת ההצבה) כי $T(n) \leq 2n^{(\log n + 1)/2} – 1$ לכל $n \geq 1$ ולכן $T(n) = O(n^{(\log n + 1)/2})$.

מקרה בסיס ($n = 1$): נתון כי $T(1) = 1$. ואכן $T(1) \geq 1 = 1*2^{(\log 1 + 1)/2} – 1$.

מקרה כללי: יהי $n \geq 2$ ונניח כי הוכחנו הטענה לכל $n' < n$. אזי:

$T(n) = nT(n/2) + 1 \leq n (2*(n/2)^{(\log (n/2) + 1)/2} – 1) +1 = 2n (n^{(\log (n/2) + 1)/2}) / 2^{(\log (n/2) + 1)/2} – n + 1 = 2n (n^{(\log n)/2}) / (n^{1/2}) – n + 1 = 2n^{(\log n + 1)/2} – n + 1 \leq 2n^{(\log n + 1)/2} – 1$

כאשר אי השוויון הראשון נובע מהנחת האינדוקציה והאחרון נובע מכך ש $n \geq 2$.

לשירותכם, איך הגענו לניחוש הנ"ל (זה *לא* צריך להופיע בפתרון): בשיטת האיטרציה נקבל

$T(n) = n(n/2)(n/4)\ldots(n/2^{i-1})T(n/2^i) + n(n/2)(n/4)\ldots(n/2^{i-2}) + n(n/2)(n/4)\ldots(n/2^{i-3}) + \ldots + n + 1$

נפתח את המקדם של $T(n/2^i)$

$n(n/2)(n/4)\ldots(n/2^{i-1}) = n^i / 2^{(1 + \ldots + (i-1))}$

נציב $i = \log n$ (שאז $T(n/2^i) = 1$) ונקבל כי האיבר הראשון בסכום הוא:

$n^{\log n} / 2^{\log n (\log n – 1)/2} = n^{\log n} / n^{(\log n – 1)/2} = n^{(\log n + 1)/2}$

כיוון ששאר המחוברים קטנים/שווים לערך זה, מתקבל חסם נאיבי של $O(n^{(\log n + 1)/2} \log n)$ (שהוא יותר טוב, כמובן, מהחסם $O(n^{\log n})$ שמצאתם רובכם), אבל מתוך אבחנה

שהמחוברים קטנים "מהר" (בדומה לסדרה הנדסית) עולה ה"חשד" שלמעשה סדר הגודל של הסכום הוא קטן בהרבה. לחסם מדוייק מגיעים באמצעות ניסוי וטעיה. (כמובן שישנם חסמים נכונים רבים שיתנו אותו סדר גודל; החסם לעיל אינו בהכרח הקל ביותר להוכחה).

שאלה 14

יש למצוא חסם עליון אסימפטוטי הדוק ככל שינתן עבור כל אחד מקטעי הפסאודו- קוד הבאים כפונקציה של n (יש להניח תמיד כי n מספר טבעי). אנא נמקו בבירור את תשובותיכם.

```
1. for i=1, …n
        for j = 1, …, i
        {
                k = n
                while k >2
                        k = ⌊k^(1/3)⌋
        }
```

נשים לב כי בכל לולאת for פנימית מתבצעת פעולה זהה. נתבונן בריצה אחת של לולאה כזו. ניתן לראות כי לאחר p איטרציות של לולאת ה-while, ערכו של k הוא $n^{1/3^p}$. הלולאה תרוץ כל עוד ערך זה > 2, כלומר, כל עוד $3^p < \log n$. לכן מספר הפעמים שלולאת ה-while רצה הוא $O(\log \log n)$. קל לראות כי מספר הפעמים שלולאת ה- for הפנימית רצה הוא $O(n^2)$ ולכן זמן הריצה הכולל הוא $O(n^2 \log \log n)$.

```
2. for i=1, …n
        for j = 1, …, i
        {
                for k = 1, …, j
                        print k
                k = 2
                while k < i
                        k = k^2
        }
```

עבור i ו-j מסויימים, זמן הריצה של לולאת ה- for הפנימית הוא $O(j)$ וזמן הריצה של לולאת ה-while הוא $O(\log \log i)$ (הוכחה בדומה לולאת ה- while בסעיף הקודם). באמצעות סכימה נקבל כי סדר הגודל הוא $O(n \log \log n + n^3) = O(n^3)$ (ניתן לבצע את הסכימה במספר דרכים. שימו לב רק שעבור i ו-j מסויימים אסור להניח שום דבר על היחס בין i ל-j).

שאלה 15

נתונה מטריצה A בגודל mXn המכילה מספרים שלמים, ומספר שלם *k* (m, n, k אינם קבועים לצרכי סיבוכיות).

דרוש מבנה נתונים שיאפשר מימוש הפעולות הבאות:

1. אתחול המבנה – בזמן O(mn).
2. שינוי כניסה כלשהי במטריצה A – בזמן O(*k*).
3. מציאת ריבוע בגודל kXk במטריצה A שסכום איבריו מקסימלי – O(mn).

יש לתאר את המבנה בבירור ו להסביר כיצד לממש את הפעולות הנ "ל באופן נכון ותוך עמידה בדרישות הסיבוכיות.

ישנן מספר וריאציות של אותו פתרון.
בלי הגבלת הכלליות נניח כי $k <= m, n$, כי אחרת אין במטריצה ריבוע k X k.

מבנה הנתונים יכלול:

1. מטריצה $A = (a_{ij})_{i = 1, \ldots, m, j = 1, .., n}$ שתכיל את הנתונים של המטריצה הנתונה
2. מטריצה $B = (b_{ij})_{i = 1, \ldots, m-k+1, j = 1, .., n}$ שתכיל סכומים של k איברים רצופים באותה שורה במטריצה A, כלומר:

$$b_{ij} = a_{ij} + \ldots + a_{i(j+k-1)}$$

3. מטריצה $C = (c_{ij})_{i = 1, \ldots, m, j = 1, .., n-k+1}$ שתכיל סכומים של k איברים רצופים באותה עמודה במטריצה A, כלומר:

$$c_{ij} = a_{ij} + \ldots + a_{(i+k-1)j}$$

<u>אתחול</u>: נאתחל את המטריצה A בערכים הרצויים בזמן O(mn).
אתחול המטריצה B יתבצע באופן הבא:
לכל שורה i נציב בתא b_{i1} את הסכום $a_{i1}+ ,,, + a_{ik}$. חישוב הסכום דורש O(k). לחישוב האיבר b_{ij} עבור $j >= k$ נשים לב כי

$$b_{ij} = a_{i(j-1)} + a_{ij} \ldots + a_{i(j+k-2)} + a_{i(j+k-1)} - a_{i(j-1)} = b_{i(j-1)} + a_{i(j+k-1)} - a_{i(j-1)}$$

מאחר והערך $b_{i(j-1)}$ כבר חושב, נוכח לחשב את b_{ij} בזמן O(1) לכל $j >=k$. מאחר וישנם O(n-k) איברים כאלו, סה"כ זמן הריצה עבור שורה הוא O(n), ומאחר שישנן m שורות, זמן הריצה הכולל הוא O(mn).
אתחול המטריצה C יתבצע באופן מקביל עבור העמודות . סה"כ זמן הריצה של האתחול הוא O(mn).

<u>שינוי כניסה</u>: נשמור את ההפרש בין הערך החדש לערך הקודם. לאחר שינוי הערך במטריצה A עצמה, נצטרך להוסיף את ההפרש לכל ערכי המטריצות B ו-C שסכומם כולל את הכניסה ששונתה. ליתר דיוק, אם שינינו את הכניסה a_{ij} נצטרך להוסיף את ההפרש לאיברים $b_{i(j-k+1)}$, b_{ij} ,…, ולאיברים c_{ij} ,…, $c_{(i-k+1)j}$, אם הם קיימים . בסך הכל נצטרך לשנות לכל היותר 2k איברים, ולכן זמן השינוי הוא O(k).

<u>מציאת ריבוע kXk שסכומו מקסימלי</u>

נסביר כיצד לחשב את סכומי כל הריבועים בזמן O(mn). הריבוע המקסימלי יתקבל באמצעות החזקת משתנים שיחזיקו את הקואורדינטות וסכום הערכים המקסימלי בריבוע שנמצא עד כה, ועדכונם עם כל חישוב.

ראשית נסביר כיצד למצוא את הריבוע המקסימלי מבין אלו שנמצאים בשורות 1-k. נשים לב כי סכום האיברים c_{11} + … + c_{k1} במטריצה C שווה לסכום האיברים בריבוע kXk במטריצה A שהפינה העליונה-שמאלית שלו היא a_{11}. זמן החישוב של סכום זה הוא O(k). עבור שאר הריבועים בשורות 1-k, נוכל לחשב את ערכו של כל ריבוע באמצעות הריבוע הקודם וההפרשים בין סכומי העמודות בריבוע, בדומה לאופן החישוב של סכומי השורות באתחול. לדוגמה, חישוב הריבוע שהפינה העליונה-שמאלית שלו היא a_{12} יתקבל על ידי הוספת ההפרש בין c_{1k} לבין c_{11} לסכום הריבוע הראשון שחושב. על כן חישוב סכומי n הריבועים בשורות 1-k יארך זמן של O(n).

כדי לעבור משורה לשורה, נשתמש בכל פעם בהפרש בין סכומי שורות באורך k שנמצאים במטריצה B. לכן המעבר משורה לשורה יארך O(1). מאחר שישנן m שורות, זמן החישוב הכולל הוא O(mn).

הערות:

1. יש לדאוג לאיזון בין מלל לקוד. הפתרון צריך להיות נכון ומדוייק, אבל גם מובן לקורא אנושי. אין להסתפק לעולם בפסאודו קוד ללא הסבר. מצד שני, תיאור שכולו מילולי עלול להיות לא מספיק מדוייק, ולכן יש לכלול פסאודו קוד או נוסחאות כמו בפתרון הנ"ל. רצוי מאוד להוסיף איורים להמחשה.
2. בחלק מהפתרונות שינוי כניסה התבצע רק במטריצה A עצמה ב-O(1) ואילו חישובי הסכומים נדחו כולם לשלב מציאת ריבוע מקסימלי. לפתרון זה יש יתרון משמעותי ככל שמספר פעולות העדכון רב יותר. שימו לב שבאלגוריתם שלעיל אנחנו מעדכנים סכומים גם אם לא נשתמש בהם בעתיד.

שאלה 16

הוכיחו או הפריכו: (את ההוכחות יש לבצע לפי הגדרת חסם בלבד).

a. $2 \cdot n + 3 \cdot \sqrt{n} = O(n^2)$
b. $\sqrt{n} + \log(n) = O(\log(n))$
c. $\log(\sqrt{n}) = O(\log(n))$
d. $n \cdot \log n = O(n)$
e. $\log(n) = o(\sqrt{n})$

א. נבדוק את הטענה :

$$if\left(2n+3\sqrt{n}=O(n^2)\right)then\left(\ \exists c_0,n_0\ \forall n>n_0\ \ 2n+3\sqrt{n}\le c_0\cdot n^2\right)$$

$$\Rightarrow \frac{2n+3\sqrt{n}}{n^2}\le c_0 \Rightarrow \frac{2}{n}+\frac{3}{n^{\frac{3}{2}}}\le c_0$$

$$for\ example: c_0=1, n_0=100$$

הטענה נכונה

ב.נבדוק את הטענה :

$$if\left(\sqrt{n}+\log n=O(\log n)\right)then\left(\exists c_0,n_0\ \forall n>n_0\ \sqrt{n}+\log n\le c_0\cdot\log n\right)$$

$$\Rightarrow \sqrt{n}\le (c_0-1)\cdot\log n \Rightarrow \frac{\sqrt{n}}{\log n}\le (c_0-1)$$

$$\lim_{n\to\infty}\frac{\sqrt{n}}{\log n}=\lim_{n\to\infty}\frac{\frac{1}{2}\cdot\frac{1}{\sqrt{n}}}{\frac{1}{n\cdot\log_e n}}=\lim_{n\to\infty}\frac{n\cdot\log_e n}{2\sqrt{n}}=\lim_{n\to\infty}\frac{\sqrt{n}\cdot\log_e n}{2}=\infty$$

$$\Rightarrow \infty\le (c_0-1)$$

זאת סתירה .**מה שאומר שהטענה לא נכונה.**

ג. נבדוק את הטענה :

$$if\left(\log\sqrt{n}=O(\log n)\right)then\left(\ \exists c_0,n_0\ \forall n>n_0\ \ \log_2\sqrt{n}\le c_0\left(\log_2 n\right)\right)$$

$$\Rightarrow \frac{1}{2}\log_2 n\le c_0\log_2 n \Rightarrow \frac{1}{2}\le c_0$$

$$for\ example: c_1=1, c_0=10, n_0=1$$

הטענה נכונה

ד. נבדוק את הטענה :

$$if\left(n\cdot\log n=O(n)\right)then\left(\exists c_0,n_0\ \forall n>n_0\ n\cdot\log n\le c_0\cdot n\right)$$

$$\Rightarrow \log n\le c_0$$

$$\lim_{n\to\infty}(\log n)=\infty \Rightarrow$$

$$\infty\le c_0$$

זאת סתירה .**מה שאומר שהטענה לא נכונה.**

ה. נבדוק את הטענה :

$$if\left(\log n=o(\sqrt{n})\right)then\left(\forall c_0,n_0\ \forall n>n_0\ \log n\le c_0\cdot\sqrt{n}\right)$$

$$\Rightarrow n\le 2^{c_0\cdot\sqrt{n}}=\sqrt{n}\le 2^{0.5\cdot c_0\cdot\sqrt{n}}$$

כבר הוכחנו למשל בעזרת לופיטל שתמיד: קבוע בחזקת אין סוף יותר גדול מאין סוף בחזקת קבוע..
מה שאומר שהטענה נכונה.

שאלה 17

מצאו את T(n) בכל אחת מנוסחאות הנסיגה שלהלן . מצאו חסמים הדוקים ככל שתוכלו ונמקו תשובתכם.

1. $T(1)=2, T(n)=\left(T\left(\frac{n}{2}\right)\right)^3$
2. $T(1)=1, T(n)=2\cdot T\left(\frac{n}{4}\right)+\sqrt{n}$

$$2^k = n \Rightarrow k = \log_2 n$$

$$T(1) = 2, T(n) = \left(T(\frac{n}{2}) \right)^3 \Rightarrow \log\left(T(n)\right) = \log\left(\left(T(\frac{n}{2}) \right)^3 \right)$$

$$Z(n) = \log\left(T(n)\right) \Rightarrow Z(n) = 3^1 \cdot Z(\frac{n}{2^1}) = 3^2 \cdot Z(\frac{n}{2^2}) = 3^{\log n} \cdot Z(\frac{n}{2^{\log n}}) = 2 \cdot 3^{\log_2 n}$$

$$T(n) = 2^{Z(n)} = 2^{2 \cdot 3^{\log n}} = O(2^n)$$

$$4^k = n \Rightarrow k = \log_4 n$$

$$T(\frac{n}{4^0}) = 2^1 T(\frac{n}{4^1}) + 2^0 \sqrt{\frac{n}{4^0}} = 2^{\log_4 n} T(\frac{n}{4^{\log_4 n}}) + 2^{\log_4 n - 1} \sqrt{\frac{n}{4^{\log_4 n - 1}}} + \ldots + 2^1 \sqrt{\frac{n}{4^1}} + 2^0 \sqrt{\frac{n}{4^0}} =$$

ב:

$$\sqrt{n} T(1) + \sqrt{n} (\underbrace{\qquad}_{\log_4 n}) \cdot \log_4 n \cdot \sqrt{n} = O(\sqrt{n} \cdot \log n)$$

שאלה 18_

נתונה התכנית הבאה:

```
int foo(int n)
{
 if (n<42)
        return 1;
 if(n is odd)  {
        for i=1:n
                print i;
        return foo(n-1);
        }
 if(n is even) {
        for i=1: √n
                print i;
        return foo(n/2);
        }
}
```

א.מהי נוסחת נסיגה עבור הפונקציה?

ב.יש לחשב את סיבוכיות הזמן במושג של Ω בלבד(חסם התחתון).

$$T(n)=\begin{cases} n<42 & 1 \\ n\ is\ odd & O(n)+T(n-1) \\ n\ is\ even & O(\sqrt{n})+T(\frac{n}{2}) \end{cases}$$

א.

ב. התרחיש הכי טוב (חסם תחתון) שהמספר בכל האיטרציה יהיה זוגי.
נוסחת הנסיגה שלנו תיראה כך:

$$T(n)=O(\sqrt{\frac{n}{2^0}})+T(\frac{n}{2^1})=T(\frac{n}{2^{\log n}})+O(\sqrt{\frac{n}{2^0}})+O(\sqrt{\frac{n}{2^1}})+...+O(\sqrt{\frac{n}{2^{\log n-1}}})$$

$$=1+c_0\sqrt{\frac{n}{2^0}}+c_1\sqrt{\frac{n}{2^1}}+...+c_{\log n-1}\sqrt{\frac{n}{2^{\log n-1}}}=1+c_{MAX}\cdot\sqrt{n}(\sqrt{\frac{1}{2^0}}+\sqrt{\frac{1}{2^1}}+...+\sqrt{\frac{1}{2^{\log n-1}}})$$

$$=1+c_{MAX}\cdot\sqrt{n}(\frac{1}{2^{0\cdot 0.5}}+\frac{1}{2^{1\cdot 0.5}}+...+\frac{1}{2^{(\log n-1)\cdot 0.5}})=1+c_{MAX}\cdot\sqrt{n}(\frac{1}{\sqrt{2}^0}+\frac{1}{\sqrt{2}^1}+...+\frac{1}{\sqrt{2}^{(\log n-1)}})$$

$$=1+c_{MAX}\cdot\sqrt{n}(\frac{1}{\sqrt{2}^0}\cdot\frac{\left(\frac{1}{\sqrt{2}}\right)^{\log n-1}-1}{\frac{1}{\sqrt{2}}-1})=1+(\frac{\sqrt{2}}{\sqrt{2}-1})\cdot c_{MAX}\cdot\sqrt{n}=O(\sqrt{n})$$

שאלה 19

מה סיבוכיות הזמן של הקטעים הבאים(במושגים של Θ)?נמקו.

א.

```
void func( int n )
{
  int i, j, k, x;

  x = 0;
  for ( i=0; i < n; i++ )
     for ( j=i; j < n; j++ )
          for ( k=j; k>i; k-- )
                   x++;
}
```

ב.

```
void func2( int n )
{
   if ( n<=0 )
      return 1;
   else
       if (n % 2 )
```

```
        return func2( n/2 );
    else
        return func2( n – 1 );
}
```

א.

$$first\ \text{iteration}: \sum_{i=0}^{n} n \quad second\ \text{iteration}: \sum_{i=1}^{n} n$$

...

$$n-1\ \text{iteration}: \sum_{i=n-1}^{n} n$$

$$\Rightarrow \sum_{i=0}^{n} n + \sum_{i=1}^{n} n + \ldots + \sum_{i=n-1}^{n} n + \sum_{i=n}^{n} n = 0\cdot 1 + 1\cdot 2 + 2\cdot 3 + \ldots + (n-1)\cdot n =$$

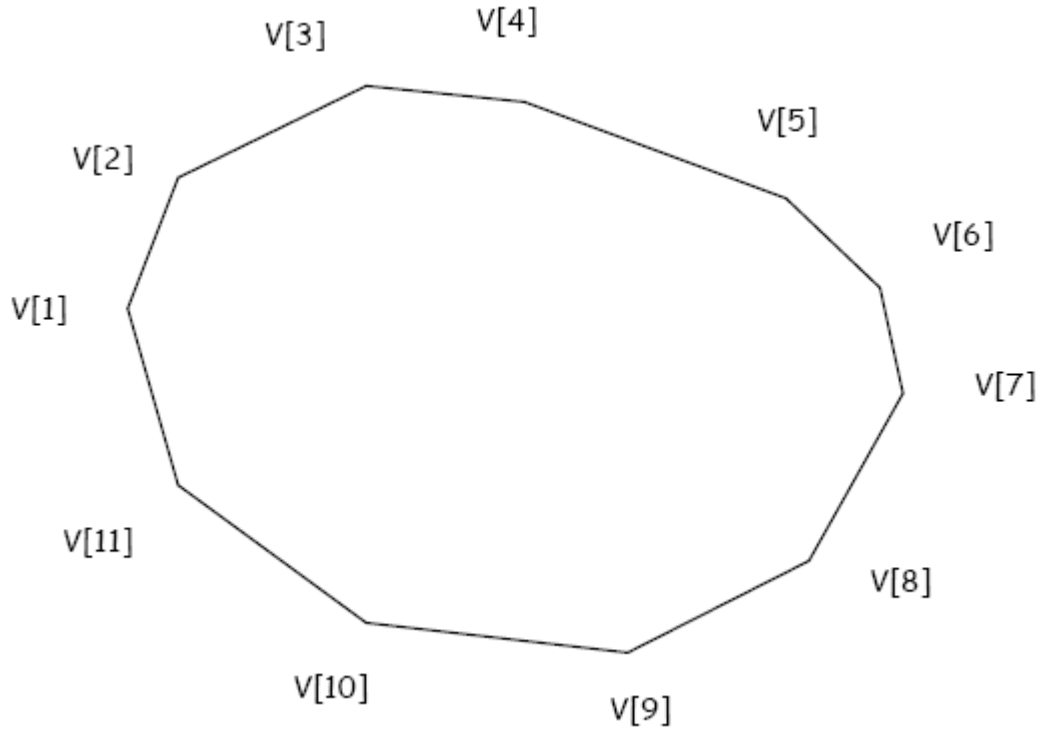
$$\sum_{i=1}^{n} (i-1)i = \sum_{i=1}^{n} i^2 - \sum_{i=1}^{n} i = \frac{n(n+1)(2n+1)}{6} - \frac{n(n+1)}{2} = \frac{n^3 - n}{3} = \theta(n^3)$$

ב.

$$T(n) = \left\{ \begin{array}{l} T(\frac{n}{2}) + \theta(1)\ \ odd \\ T(n-1) + \theta(1)\ \ even \end{array} \right\} = W.C.\ 2 \cdot \theta(1) \cdot \log n = \theta(\log n)$$

שאלה_20

מצולע יקרא קמור אם כל הזוויות הפנימיות שלו קטנות מ-180 מעלות ,ושום צלע אינה חותכת את השאר .
נתון מצולע קמור ע"י n הנקודות שלו במערך V באורך n [V[1...n .כל נקודה במצולע תתואר באמצעות שתי קואורדינאטות x ו - y.
נתון ש V[1] – הוא הצומת עם הערך המינימלי עבור x וכל שאר הצמתים מסודרים בסדר הפוך לכיוון השעון .ניתן להניח שכל הקואורדינטות של x ו הקואורדינאטות של y הם זרות.



א. תנו אלגוריתם המוצא את הקואורדינאטה עם ה x - המקסימאלי בזמן של $O(\log n)$
ב. תנו אלגוריתם המוצא את הקואורדינאטה עם ה y - המקסימאלי בזמן של $O(\log n)$

```
int findX(int szie, Node *arr)
{
	int mod = size/2+1;
	int left = 0;
	int right = size;

	while (true)
	{
		if (arr[mid].x < arr[mid-1].x && arr[mid].x >
	arr[mid+1].x)
		{
			right = mid;
			mid = (left + right)/2 + 1;
			continue;
		}
		if (arr[mid].x > arr[mid-1].x && arr[mid].x <
	arr[mid+1].x)
		{
			left = mid;
			mid = (left + right)/2 + 1;
			continue;
		}
		break;
	}
	return mid;
}
```

כמובן שלא עושים רקורסיה, כי זה מעלה את הסיבוכיות מקום לסיבוכיות לוגריתמית.
חיפוש בינארי: **סיבוכיות זמן** $O(\log n)$

$O(1)$ **סיבוכיות מקום נוסף** : מספר קבוע של משתנים:

ב.

האלגוריתם יהיה מבוסס על חיפוש בינארי.

```
int findY(int szie, Node *arr)
{
	int mod = size/2+1;
	int left = 0;
	int right = size;

	while (true)
	{
		if ((arr[mid].y < arr[mid-1].y && arr[mid].x >
		arr[mid+1].y)|| (arr[mid].x > arr[mid-1].x && arr[mid].x
		< arr[mid+1].x && arr[mid].x > arr[mid+1].x) ||
		(arr[mid].y < arr[mid-1].y && arr[mid].y < arr[mid+1].y))
		{
			right = mid;
			mid = (left + right)/2 + 1;
			continue;
		}
		if (arr[mid].y > arr[mid-1].y && arr[mid].y <
	arr[mid+1].y)
		{
			left = mid;
			mid = (left + right)/2 + 1;
			continue;
		}
		break;
	}
	return mid;
}
```

.X אלה גם את Yבהבדל ל סעיף א' חשוב לבדוק לא רק את
כמובן שלא עושים רקורסיה, כי זה מעלה את הסיבוכיות מקום לסיבוכיות לוגריתמית.

$O(\log n)$ **סיבוכיות זמן** :חיפוש בינארי

$O(1)$ **סיבוכיות מקום נוסף** : מספר קבוע של משתנים:

פרק שני – מבני נתונים בסיסיים, עצים, עצי חיפוש

שאלה 1

הציעו מבנה נתונים התומך בפעולות push ו- pop כפי שהן מוגדרות עבור מחסנית , וכן בפעולה find-min המחזירה את האיבר בעל הערך הנמוך ביותר במבנה. כל הפעולות הם בסיבוכיות זמן - $O(1)$ במקרה הגרוע.

הרעיון להוסיף לכל שדה משתנה MIN וכך כל צומת יודע מי היה מינימום לפניו.בגלל שמדובר במחסנית – אנחנו יכול לחזות את סדר ההוצאה של הצמתים וכך נוכל לשמר נכון את המידע על המינימום בזמן הנוכחי.

```
#include <stdio.h>
#define STACK_SIZE 100
int top = 0;
typedef struct{
	int min;
	int value;
} node, *Node;

node S[STACK_SIZE];

void push(int k){
	if (top == STACK_SIZE){
		printf("full");
		return;
	}
	S[top].value = k;
	if(top==0)
		S[top].min = k;
	else if (S[top-1].min > k)
		S[top].min = k;
	else
		S[top].min = S[top-1].min;
	top++;
	return;
}

int pop(){
	if (top == 0){
		printf("empty");
		return 0;
	}
	top--;
	return S[top].value;
}

int min(){
	if (top == 0){
		printf("empty");
		return 0;
	}
	else
		return S[top].min;
}
```

סיבוכיות זמן: כל הפעולות מבוצעות בזמן קבוע $O(1)$

סיבוכיות מקום נוסף $O(n)$: לכל צומת במחסנית רגילה הוספנו משתנה מינימום

שאלה 2

רשימה מקושרת "רגישת חיפוש" מוגדרת כרשימה מקושרת רגילה עם header כך שהכנסות מתבצעות בראש הרשימה וכאשר ניגשים אל איבר באמצעות פעולת find הוא מועבר לראש הרשימה (בלי לשנות את הסדר בין שאר איברי הרשימה).

1. נניח שנתונה הרשימה הבאה:

header--> 1 --> 2 --> 3 --> 4 --> 5

תארו מהי התוצאה של הפעלת הפעולות הבאות (משמאל לימין)

find(3); insert(6); delete(2); find(4)

2. ממשו את הפעולות (כתבו פסאודו-קוד) insert, delete ו- find עבור רשימה מקושרת רגישת חיפוש.

3. נניח שרשימה מכילה איבר x כזה שבממוצע $\frac{1}{4}$ מפעולות ה-find מתבצעות עליו, כלומר, מתוך K פעולות find, K/4 הן פעולות find(x) ו 3/4 K הן פעולות find על איברים שונים מ – x.

הוכיחו שהזמן הממוצע של פעולת find(x) הוא $O(1)$,
כאשר אורך הרשימה – N ומספר הפעולות – K ונתון ש- $K = \Omega(N)$.

```
#include "stdio.h"
#include "malloc.h"
#include "conio.h"

typedef struct node *Node;
typedef struct node {
	Node next;
	int value;
} node, *Node;

Node head;

void printList(){
	Node temp = head;
	printf("\nprintList\n");
	printf("\n");
	while(temp != 0){
		printf("%d ", temp->value);
		temp = temp->next;
	}
	printf("\n");
}

	int deleteNode(int value){
	Node temp, temp2 = head;
	printf("\ndelete %d\n", value);
	if (temp2 == 0){
		printf("\nnot found\n");
		return 0;
	}
	if(temp2->value == value){
		printf("\nfound\n");
		if(temp2->next == NULL)
			head = 0;
		else
			head=temp2->next;
		free(temp2);
		return 1;
	}
```

```
	temp = temp2->next;
	while(temp != 0){
		if(temp->value == value){
			printf("\nfound\n");
			temp2->next = temp->next;
			free(temp);
			return 1;
		}
		temp2 = temp2->next;
		temp = temp->next;
	}
	printf("\nfound\n");
	return 0;
}

void insert(int value){
	Node temp;
	printf("\ninsert %d\n", value);
	temp = (Node)malloc(sizeof(node));
	temp->value = value;
	temp->next = head;
	head = temp;
}

int find(int value){
	printf("\nfind %d\n", value);
	if (!deleteNode(value))
	return 0;
	insert(value);
	return 1;
}

int main(){
	int i;
	for (i = 5;i>0;i--)
		insert(i);
	printList();
	find(3);
	printList();
	insert(6);
	printList();
	deleteNode(2);
	printList();
	find(4);
	printList();
	getch();
	return 0;
}}
```

```
insert 5
insert 4
insert 3
insert 2
insert 1
printList
1    2    3    4    5
find 3
delete 3
found
insert 3
printList
3    1    2    4    5
insert 6
printList
6    3    1    2    4    5
delete 2
found
printList
6    3    1    4    5
find 4
delete 4
found
insert 4
printList
4    6    3    1    5
```

3.2

$$first\ time\ find(x) = O(n)$$

$$\frac{\sum_{i=2}^{\frac{K}{4}-m} find(x) = O(\frac{3K}{4}) + \sum_{i=\frac{K}{4}-m}^{\frac{K}{4}-m} find(x) = O(m)}{\frac{K}{4}} = \frac{\frac{C_1 3K}{4} + C_2 m}{\frac{K}{4}} = C_1 3 + \frac{C_2 4m}{K}$$

$$if\ (\lim_{K\to\infty} \frac{4m}{K} = 0)\ then: C_1 3 + \frac{C_2 4m}{K} = \theta(1)$$

$$if\ (\lim_{K\to\infty} \frac{4m}{K} = C) then := C_1 3 + \frac{C_2 4m}{K} = \theta(1)$$

שאלה 3

המספרים 1,2,3,4,5,6 מגיעים לקלט בסדר עולה, ומוכנסים למבנה נתונים D. הוצאת מספר מ-D מעבירה אותו לפלט. ניתן להוציא מ – D מספר עוד לפני שכל המספרים נכנסו ל - D . אילו מהסדרות הבאות יכולות להתקבל בפלט (משמאל לימין) כאשר:

א. D הוא תור.

ב. D הוא מחסנית.

ג. D הוא דו-תור.

הסדרות:

1. 5,2,6,3,4,1
2. 2,4,3,6,5,1
3. 1,5,2,4,3,6
4. 4,2,1,3,5,6
5. 1,2,5,4,6,3

אם סדרה יכולה להתקבל ממבנה הנתונים D, הראו את רצף הפעולות הדרוש על מבנה הנתונים.

א. אין פלט שיכול להיות פלט של תור.הפלט של תור לפי כל תרחיש תמיד יהיה : $1,2,3,4,5,6$

ב. פלטים שיכולים להיות במחסנית :

פלט מס' 3 :

push(1), *push*(2), *pop*(), *push*(3), *push*(4), *pop*(), *pop*(), *push*(5), *push*(6), *pop*(), *pop*(), *pop*()

2,4,3,6,5,1

פלט מס' 5 :

push(1), *pop*(), *push*(2), *pop*(), *push*(3), *push*(4), *push*(5), *pop*(), *pop*(), *push*(6), *pop*(), *pop*()

$1,2,5,4,6,3$

ג. פלטים שיכולים להיות פלטים של דו-תור הם :

פלט 2 :

enQueue(Tail,1),enQueue(Tail,2),deQueue(Tail), enQueue(Tail,3),
enQueue(Tail,4),deQueue(Tail),deQueue(Tail), enQueue(Tail,5),
enQueue(Tail,6),deQueue(Tail),deQueue(Tail),deQueue(Tail)

$2,4,3,6,5,1$

פלט 3 :

enQueue(Tail,1),deQueue(Tail),enQueue(Tail,2), enQueue(Tail,3), enQueue(Tail,4),enQueue(Tail,5),
deQueue(Tail), deQueue(Haid),deQueue(Tail), deQueue(Tail),enQueue(Tail,6),deQueue(Tail)
1,5,2,4,3,6

פלט 4 :

enQueue(Tail,1),enQueue(Head,2), enQueue(Tail,3), enQueue(Tail,4),deQueue(Tail),deQueue(Haid),
deQueue(Head),deQueue(Tail), enQueue(Tail,5), deQueue(Tail), enQueue(Tail,6),deQueue(Tail)
4,2,1,3,5,6

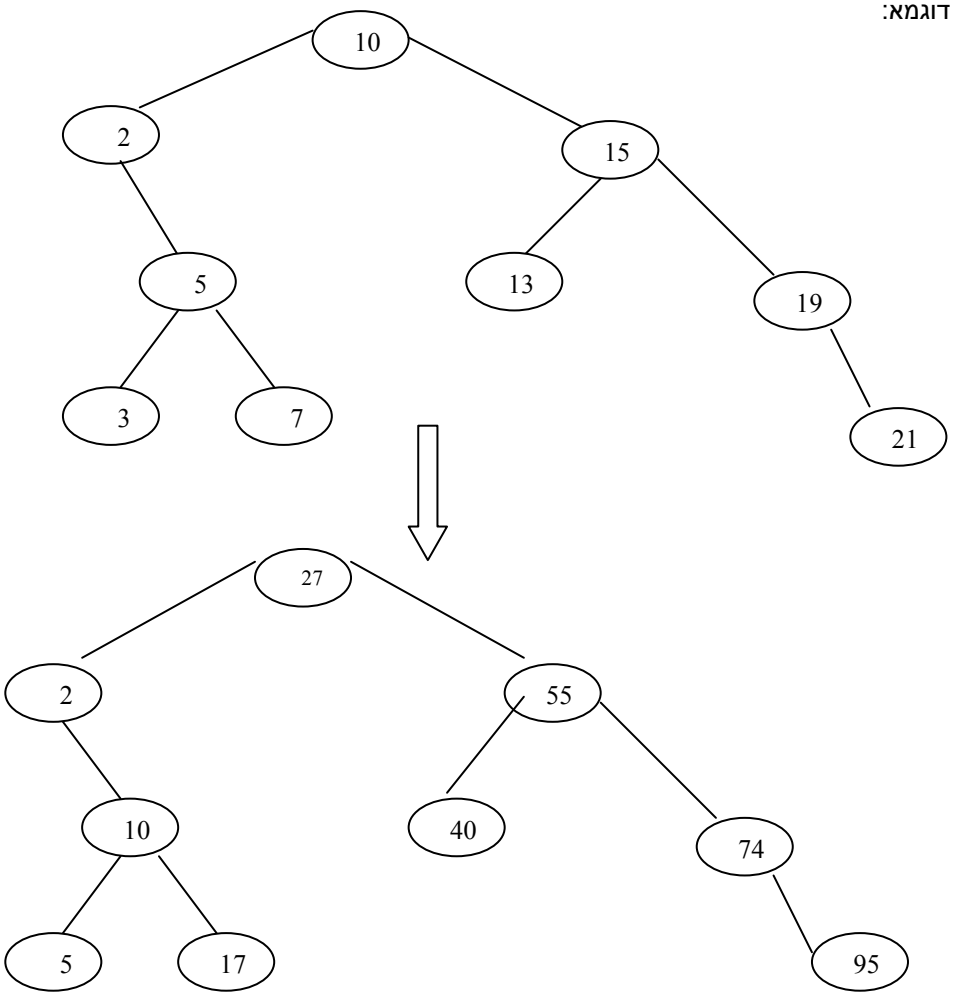
פלט 5 :

enQueue(Tail,1),deQueue(Tail), enQueue(Tail,2), deQueue(Tail),enQueue(Tail,3), enQueue(Tail,4),
enQueue(Tail,5), deQueue(Tail), deQueue(Tail),enQueue(Tail,6),deQueue(Tail), deQueue(Tail)
1,2,5,4,6,3

<u>שאלה 4</u>

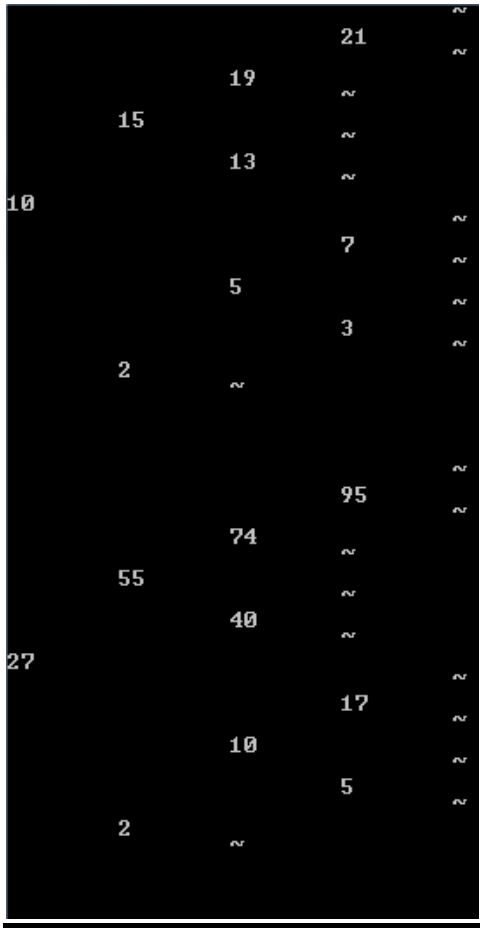
נתון עץ חיפוש בינארי בעל n צמתים. לכל צומת מפתח ייחודי (אין כפילויות). עליכם להציע אלגוריתם (פסאודו-קוד) שיבנה עץ בינארי חדש בעל מבנה זהה לעץ המקורי כך שהמפתח של כל צומת בעץ החדש יהיה סכום של כל המפתחות בעץ המקורי שקטנים או שווים לערך המפתח ב צומת המתאים בעץ המקורי. העץ המקורי חייב להישאר ללא שינוי. סיבוכיות הזמן הנדרשת $O(n)$.

דוגמא:



```
#include "stdio.h"
#include "malloc.h"
#include "conio.h"

typedef struct tree *Tree;

typedef struct tree
{
	Tree left;
	Tree right;
	int key;
} tree, *Tree;

Tree head;

int counter = 0;

Tree solTree;

void padding ( char ch, int n )
{
	int i;

	for ( i = 0; i < n; i++ )
		putchar ( ch );
}
void printTree ( Tree root, int level )
{

	if ( root == NULL )
	{
		padding ( '\t', level );
		puts ( "~" );
	}
	else
	{
		printTree ( root->right, level + 1 );
		padding ( '\t', level );
		printf ( "%d\n", root->key );
		printTree ( root->left, level + 1 );
	}
}


Tree InorderSumTree(Tree parent)
{
	Tree node;

	if ( parent == 0)
		return 0;
	node = (Tree ) malloc(sizeof(tree));

	node->left = InorderSumTree(parent->left);
	node->key = parent->key + counter;
	counter = node->key;
	node->right = InorderSumTree(parent->right);
	return node;
}
```

```
Tree Insert_Element(int key, Tree parent)
{
	if(!parent)
	{
		parent = (Tree ) malloc(sizeof(tree));
		parent->key = key;
		parent->left = 0;
		parent->right = 0;
		return parent;
	}

	if(key < parent->key)
		parent->left = Insert_Element(key, parent->left);

	if(key > parent->key)
		parent->right = Insert_Element(key, parent->right);

	return parent;
}



int main()
{
	head = Insert_Element(10, head);
	head = Insert_Element(2, head);
	head = Insert_Element(15, head);
	head = Insert_Element(5, head);
	head = Insert_Element(13, head);
	head = Insert_Element(19, head);
	head = Insert_Element(3, head);
	head = Insert_Element(7, head);
	head = Insert_Element(21, head);

	printTree(head, 0);
	_getch();

	solTree = InorderSumTree(head);
	printf("\n\n\n");
	printTree(solTree, 0);
	_getch();

	return 0;
}
```



סיבוכיות זמן העתקה מתבצעת בזמן הסריקה כך שזה עדיין עומד בסיבוכיות זמן ליניארית. $O(n)$

סיבוכיות מקום נוסף :רקורסיה בזמן סריקה + השכפול של העץ מעלה את הסיבוכיות מקום. $O(n)$

<u>שאלה 5</u>

קלט: מצביע לראש עץ חיפוש בינארי

פלט: עבור כל רמה זוגית בעץ יש להדפיס את הצמתים הנמצאים באותה רמה מגדול לקטן.

סדר ההדפסה יבוצע ע"פ סדר הרמות, ז"א : שורש העץ הנמצא ברמה 0 יודפס ראשון, הצמתים ברמה 1 לא יודפסו (הבנים של השורש), הצמתים שברמה 2 יודפסו מגדול לקטן וכו'.....

נדרש לכתוב קוד בשפת C יעיל ככל האפשר (מבחינת סיבוכיות מקום ומבחינת סיבוכיות זמן)

הרעיון לבצע סוג של BFS , כך שכל פעם נדחוף לתור את הבנים של הבנים של הצומת (מימין לשמאל בשביל המיון) ובשליפה מהתור נקבל את הצמתים לפי גבהים ולפי גודל.

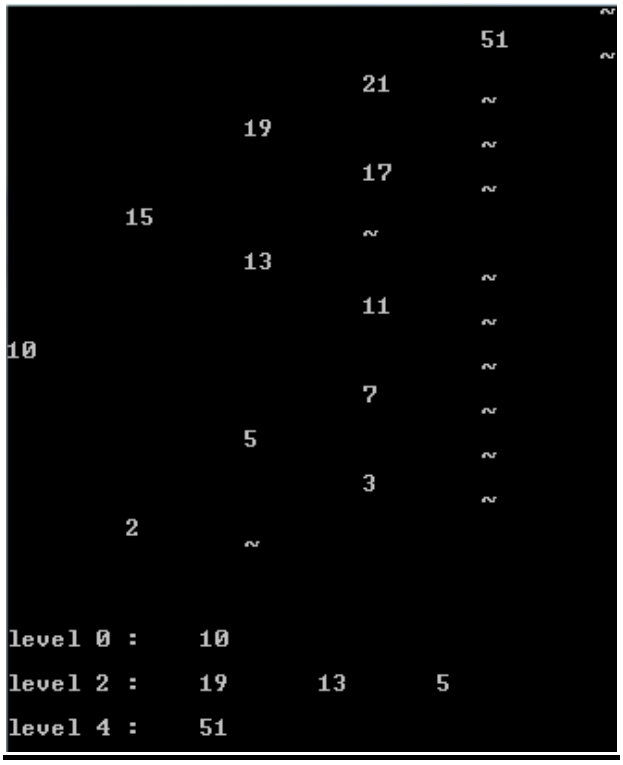
```
#include "stdio.h"
#include "malloc.h"
#include "conio.h"

typedef struct tree *Tree;
typedef struct node *Node;
typedef struct queue *Queue;

typedef struct tree
{
	Tree left;
	Tree right;
	int key;
```

```
} tree, *Tree;

typedef struct node
{
	Tree key;
	int level;
} node, *Node;

typedef struct queue
{
	Queue next;
	Queue prev;
	Node key;
} queue, *Queue;

Tree head;
Queue queueHead;
Queue queueTail;

int counter = 0;

void padding ( char ch, int n )
{
	int i;

	for ( i = 0; i < n; i++ )
		putchar ( ch );
}

void printTree ( Tree root, int level )
{

	if ( root == NULL )
	{
		padding ( '\t', level );
		puts ( "~" );
	}
	else
	{
		printTree ( root->right, level + 1 );
		padding ( '\t', level );
		printf ( "%d\n", root->key );
		printTree ( root->left, level + 1 );
	}
}

int empty()
{
	if (queueTail == 0)
		return 1;
	return 0;
}

void enqueue(Tree parent, int level)
{
	Queue temp;
	Node key;

	key = (Node)malloc(sizeof(node));
	key->key = parent;
	key->level = level;
```

```
	temp = (Queue)malloc(sizeof(queue));
	temp->key = key;
	temp->next = 0;
	if ( queueTail != 0)
		queueTail->next = temp;
	temp->prev = queueTail;
	queueTail = temp;

	if (queueHead == 0)
		queueHead = queueTail;
}

Node dequeue()
{
	Node temp;
	Queue tempNode;

	temp = queueHead->key;
	if(queueHead->next == 0 )
	{
		free(queueTail);
		queueTail = 0;
		queueHead = 0;
		return temp;
	}
	queueHead->next->prev = 0;
	tempNode = queueHead->next;
	free(queueHead);
	queueHead = tempNode;

	return temp;
}

void   printTree2()
{
	Node temp;

	if (empty( ))
		return;


	temp = dequeue();

	if (temp->level > counter)
	{
		counter = temp->level;
		printf ("\n\nlevel %d : ",counter);
	}
	printf( "   %d   ", temp->key->key);

	if( temp->key->right)
	{
		if(temp->key->right->right)
			enqueue(temp->key->right->right, temp->level+2);
		if(temp->key->right->left)
			enqueue(temp->key->right->left, temp->level+2);
	}

	if( temp->key->left)
	{
		if(temp->key->left->right)
```

```
                        enqueue(temp->key->left->right, temp->level+2);
                if(temp->key->left->left)
                        enqueue(temp->key->left->left, temp->level+2);
        }
        printTree2( );
}


void    printTreeByLevels ()
{
        if( head == 0)
                return;
        enqueue(head, 0);
        printf("level 0 : ");
        printTree2();
}

// insert node in BST
Tree Insert_Element(int key, Tree parent)
{
        if(!parent)
        {
                parent = (Tree ) malloc(sizeof(tree));
                parent->key = key;
                parent->left = 0;
                parent->right = 0;
                return parent;
        }

        if(key < parent->key)
                parent->left = Insert_Element(key, parent->left);

        if(key > parent->key)
                parent->right = Insert_Element(key, parent->right);

        return parent;
}

// main function
int main()
{

        // creat tree
        head = Insert_Element(10, head);
        head = Insert_Element(2, head);
        head = Insert_Element(15, head);
        head = Insert_Element(5, head);
        head = Insert_Element(13, head);
        head = Insert_Element(19, head);
        head = Insert_Element(3, head);
        head = Insert_Element(7, head);
        head = Insert_Element(21, head);
        head = Insert_Element(51, head);
        head = Insert_Element(17, head);
        head = Insert_Element(11, head);

        // print tree
        printTree(head, 0);
        _getch();

        printf("\n\n\n");
```

```
        //sol print : output + number of operations
        printTreeByLevels();
        _getch();

        return 0;
}
```



סיבוכיות זמן על כל צומת עוברים מספר קבוע של פעמים $O(n)$

סיבוכיות מקום נוסף :רקורסיה בזמן סריקה + תור עזר ששומר את הצמתים + שדה נוסף לכל צומת . $O(n)$

שאלה 6

נתון $T(1) = 1$ ולכל $n \geq 2$ $T(n) = 2T(n/2) + 6n - 1$

הוכיחו או הפריכו: $T(n) = O(n^2)$

$$2^k = n \Rightarrow k = \log_2 n$$
$$T(1) = 1$$
$$T(n) = 2^1 T(\frac{n}{2^1}) + \frac{2^0 6n}{2^0} - 1 = 2^1\left(2^1 T(\frac{n}{2^2}) + \frac{2^0 6n}{2^1} - 1\right) + \frac{2^0 6n}{2^0} - 1 = 2^2 T(\frac{n}{2^2}) + \frac{2^1 6n}{2^1} - 2^1 + \frac{2^0 6n}{2^0} - 2^0$$
$$= 2^2\left(2^1 T(\frac{n}{2^3}) + \frac{2^1 6n}{2^3} - 1\right) + \frac{2^1 6n}{2^1} - 2^1 + \frac{2^0 6n}{2^0} - 2^0 = 2^3 T(\frac{n}{2^3}) + \frac{2^3 6n}{2^3} - 2^2 + \frac{2^1 6n}{2^1} - 2^1 + \frac{2^0 6n}{2^0} - 2^0$$
$$= 2^{\log n} T(\frac{n}{2^{\log n}}) + \frac{2^{\log(n-1)} 6n}{2^{\log(n-1)}} + \ldots + \frac{2^3 6n}{2^3} + \frac{2^1 6n}{2^1} + \frac{2^0 6n}{2^0} - 2^{\log(n-2)} - \ldots - 2^2 - 2^1 - 2^0$$
$$= n + 6n \cdot \log n - \frac{2^{\log(n-1)} - 1}{1} = n + 6n \cdot \log n - n = 6n \cdot \log n$$
$$if\left(6n \cdot \log n = O(n^2)\right) then\left(\exists c_0, n_0 \; \forall n > n_0 \;\; 6n \cdot \log n \leq c_0 \cdot n^2\right)$$
$$\Rightarrow \frac{6 \log n}{n} \leq c_0$$
$$for\ example: c_0 = 1, n_0 = 100$$

הטענה נכונה

שאלה 7

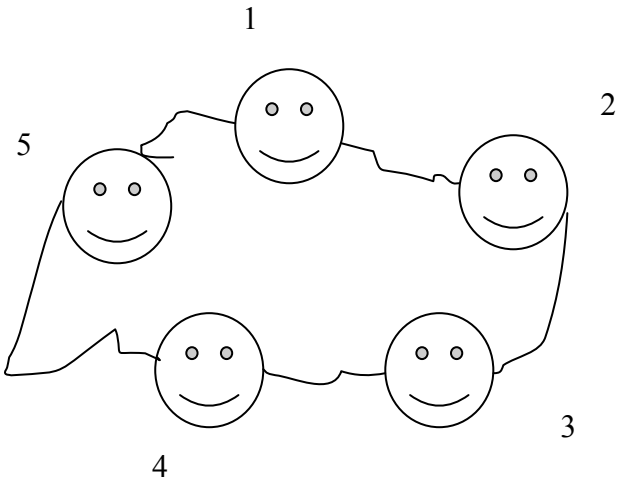
כתבו פסאודו- קוד של אלגוריתם לינארי שמקבל שני תורים ומחזיר חיבור (שרשור) של התור הראשון עם ההופכי (reverse) של התור השני. ניתן להשתמש רק בפעולות אבסטרקטיות (אין להניח מימוש מסוים).

```
ReverseAndCombine(Q1, Q2)
        return Combine(Q1, Reverse(Q2))

Reverse(Q)
        S ← makeStack()
        while !Q.IsEmpty()
                S.push(Q.dequeue())
        while !S.IsEmpty()
                Q.equeue(S.pop())
        return Q

Combine(Q1, Q2)
        while !Q2.IsEmpty()
                Q1.equeue(Q2.dequeue())
return Q1
```

שאלה 8

n ילדים עומדים במעגל ואוחזים ידיים כל אחד נותן ידיים לשני הילדים משני צידיו . מגרילים מספר k ($k < n$ וקבוע למשחק). מתחילים מהילד שהוחלט שהוא הראשון , סופרים לפי כיוון השעון , והילד מספר k יוצא

מהמשחק. ממשיכים באותו אופן מהילד הבא (בכל פעם מוציאים את הילד ה –k בספירה) עד שנותר ילד אחד והוא המנצח.



עבור k = 3 , סדר ההוצאה (מתחילים מילד מספר 1) יהיה: 3, 1, 5, 2 ו- 4 מנצח.
מוצעים שני מימושים למעגל:

א. מערך באורך n+1 של משתנים בוליאניים (1 – הילד משחק, 0 הילד יצא מהמשחק).
יש צורך לעשות איטראציות, עד שנשארים עם איבר אחד – בכל איטראציה יש לנו 2/3 מהקודמת, $1=2/3n\cdot(2/3)^{i-1}$, הפיתרון לוגריתמי, ולכן זמן המעבר על המערך הינו $O(n\cdot\log n)$.

ב. רשימה מקושרת מעגלית חד-כיוונית המכילה את הילדים שעדיין משחקים.
כאן צריך לעשות סכום של סידרה הנדסית עד האיבר ה- $(2/3)^i$. מתוך החישוב הקודם רואים שהוא שווה ל – n/1. לכן הסכום יהיה : $n\cdot(1-(2/3)^i)/(1-2/3)$ שזה יוצא 3n-3 מכאן $O(n)$.

פרטו כל מימוש ותארו במילים כיצד תבוצע ההוצאה.
מהי סיבוכיות ההוצאה בכל אחד מהמימושים עד שנותר רק ילד אחד.

שאלה 9

דרוש מבנה נתונים שיחזיק מספרים טבעיים משני צבעים, מספרים ירוקים ומספרים אדומים. מספרים וצבעים יכולים להופיע יותר מפעם אחת. אין הגבלה על מספר האיברים.
יש להציע מבנה נתונים שיעמוד בדרישות הסיבוכיות הבאות: סיבוכיות מקום לינארית וסיבוכיות זמן קבועה לכל הפעולות.

Init() אתחול מבנה הנתונים
InsertGreen() הכנס מספר ירוק למבנה.
InsertRed() הכנס מספר אדום למבנה.
DeleteGreen() הוצא את המספר הירוק האחרון שהוכנס למבנה.
DeleteRed() הוצא את המספר הירוק האחרון שהוכנס למבנה.
GetGreen() החזר את המספר הירוק האחרון שהוכנס למבנה, אם אין אברים ירוקים החזר -1
GetRed() החזר את המספר הירוק האחרון שהוכנס למבנה, אם אין אברים אדומים החזר -1
Inverse() הפוך את כל הירוקים לאדומים וההפך.
Join() אחד את כל האברים לסוג אחד, ז"א אין יותר משמעות לאדומים וירוקים עכשיו. אם מפעילים פעולת Insert() אין משמעות לצבע, אם מפעילים Delete() מוציאים את האחרון ללא משמעות לצבע. פעולת Get תחזיר את האחרון ללא משמעות לצבע, פעולת Inverse() לא תשפיע ול- Join() לא תהיה משמעות.
Split() פצל את האברים המבנה לשניים כך שמחצית תהיה אדומים ומחצית ירוקים. הפעולה חסרת משמעות אם המבנה כבר מפוצל.

תשובה:

נשתמש בשתי רשימות מקושרות, דו כיווניות, אחת לכל צבע. סדר ההכנסה של איברים מאותו צבע נשמר ע"י הסדר ברשימה עצמה. בנוסף יש לשמור את סדר ההכנסה בין כל האיברים ללא קשר לצבעם. נשמור בנוסף מצביעים בין האיברים בלי קשר לצבע, לפי סדר הכנסה שלהם. נשמור מצביע לאיבר האמצעי med שהוא אמצע הרשימה כולה (ללא קשר לצבע). בנוסף נחזיק משתנה שישמור את ההפרש בין מספר האיברים מעל ה-med ומתחתיו (-1, 0, 1). ונשמור ביט הפיכת צבעים לinverse וביט אחוד.

()Init אתחול מבנה הנתונים
()InsertGreen הכנס מספר ירוק למבנה.
()InsertRed הכנס מספר אדום למבנה.
()DeleteGreen הוצא את המספר הירוק האחרון שהוכנס למבנה.
()DeleteRed הוצא את המספר הירוק האחרון שהוכנס למבנה.
()GetGreen החזר את המספר הירוק האחרון שהוכנס למבנה, אם אין אברים ירוקים החזר 1-
()GetRed החזר את המספר הירוק האחרון שהוכנס למבנה, אם אין אברים אדומים החזר 1-
()Inverse הפוך את כל הירוקים לאדומים וההפך.
()Join אחד את כל האברים לסוג אחד, ז"א אין יותר משמעות לאדומים וירוקים עכשיו. אם מפעילים פעולת ()Insert אין משמעות לצבע, אם מפעילים ()Delete מוציאים את האחרון ללא משמעות לצבע. פעולת Get תחזיר את האחרון ללא משמעות לצבע, פעולת ()Inverse לא תשפיע ול- ()Join לא תהיה משמעות.
()Split פצל את האברים המבנה לשניים כך שמחצית תהיה אדומים ומחצית ירוקים. הפעולה חסרת משמעות אם המבנה כבר מפוצל.

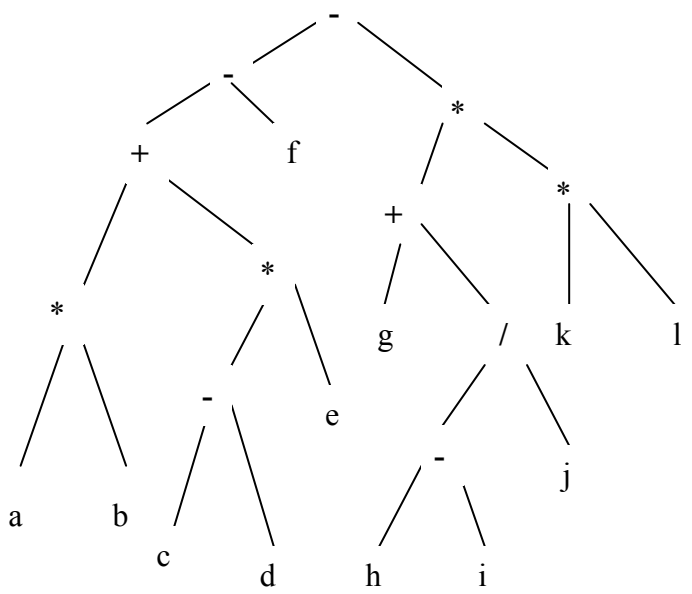
<u>שאלה 10</u>

כידוע ישנם שלושה סוגי סריקות של עצים: תחילי (preorder), תוכי (inorder), סופי (postorder). בד"כ כותבים ביטויים מתמטיים בכתיב תוכי. מחשבון HP מקבל קלט כביטוי סופי, שפות כמו Lisp ו-Scheme עובדות עם ביטויים תחיליים. פונקציות כתובות כביטוי תחילי, קודם שם הפונקציה ואז הארגומנטים. הדוגמא הבאה היא של עץ ביטויים. הפעולות מיוצגות בצמתים פנימיים, והערכים בעלים. סריקה תוכית של העץ תיתן את הביטוי הבא:

a * b + c – d * e – f- g + h – i / j *k * l

בכדי שסדר העדיפויות יהיה ברור ניתן לשים סוגריים. הסוגריים באים רק בכדי להראות את סדר הפעולות אבל אינם חלק מהעץ.

((((a * b) + ((c – d) * e)) – f) – ((g + ((h – i) / j)) * (k * l)))



כתבו אלגוריתם (בפסאודו – קוד) לשחזור ביטוי אריתמטי סופי מתוך ביטוי הסוגריים התוכי, כשהוא מצוין בסוגריים באופן מלא. על האלגוריתם להיות ליניארי במספר הארגומנטים והפעולות שלו.

יש כמה אפשרויות. אחד מהם הוא לייצר את העץ ומתוכו את הסופי.

ייצור העץ ייקח זמן לינארי בגלל שהביטוי מהווה עץ בפני עצמו.

ממנו לייצר ביטוי סופי – אותו זמן.

שאלה 11

בהינתן עץ בינארי, נגדיר אב קדמון משותף צעיר ביותר (LCA – Lowest Common Ancestor) לצמתים u, v בתור הצומת הנמוך ביותר (הכי קרוב לעלים), אשר גם u וגם v הם צאצאיו. תנו אלגוריתם יעיל אשר מקבל מצביעים לשני הצמתים הללו ומצביע לראש העץ, ומחזיר את ה LCA(u, v).
מה תהיה הסיבוכיות של האלגוריתם? הוכיחו.
ניתן לשחזר את הערכים מתוך המצביעים.
זמן – כגובה העץ.
הוכחה - בכל שלב יורד רמה ולא עולה – לכן לכל היותר גובה העץ.

```
Node *GetLCA(int a, int b)
{
   Node *    node = root;

   if (root == 0)
      return;

   while(node)
   {
      if (node->key < a && node->key < b)
         node = node->right;
      else if (node->key > a && node->key > b)
         node = node->left;
      else
         return node;
   }

   return 0;
}
```

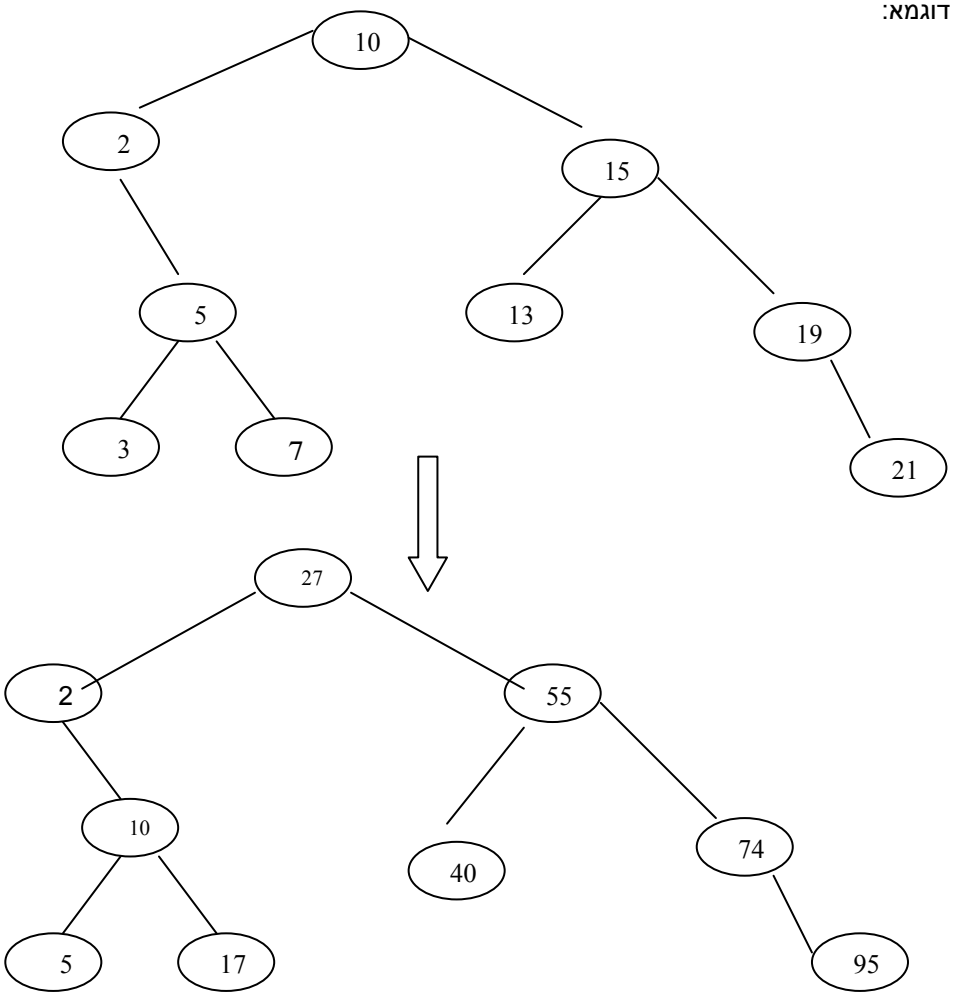
שאלה 12

הציעו מבנה נתונים התומך בפעולות push ו- pop כפי שהן מוגדרות עבור מחסנית , וכן בפעולה find-min המחזירה את האיבר בעל הערך הנמוך ביותר במבנה. כל הפעולות הם בסיבוכיות זמן - O(1) במקרה הגרוע.

שאלה 13

נתון עץ חיפוש בינארי בעל n צמתים. לכל צומת מפתח ייחודי (אין כפילויות). עליכם להציע אלגוריתם (פסאודו-קוד) שיבנה עץ בינארי **חדש** בעל מבנה זהה לעץ המקורי כך שהמפתח של כל צומת בעץ החדש יהיה סכום של כל המפתחות בעץ המקורי שקטנים או שווים לערך המפתח בצומת המתאים בעץ המקורי . העץ המקורי חייב להישאר ללא שינוי. סיבוכיות הזמן הנדרשת $O(n)$.

דוגמא:



שאלה 14

יהא T עץ בעל n צמתים. האב הקדמון המשותף הנמוך ביותר LCA (Lowest Common Ancestor) של שני צמתים u ו- v מוגדר כצומת w ב T שהיא האב הקדמון של u ו-v ונמצאת במרחק הקצר ביותר משניהם. בהינתן שני הצמתים u ו- v , כתבו אלגוריתם יעיל שימצא את ה LCA עבור שני הצמתים. מה תהיה סיבוכיות הזמן? נמקו.

מבנה כל צומת:

flag	
left	right

1. בצע את הפעולות הבאות בinorder:
   - כל עוד t שונה מnull תשווה האם t שווה לv:
     - אם לא המשך להתקדם,
     - אם כן תחזור ולאורך כל צמתי החזרה שים 1 בt->flag.
2. בצע inorder על u:
   - כל עוד t שונה מnull תשווה האם t שווה לu:
     - אם לא המשך להתקדם,
     - אם כן: כל עוד u->flag שונה מ1:
       - t יצביע על האבא שלו,
       - אם u->flag הוא 1 החזר את u.

שאלה 15

נתון עץ T שבו מאוחסנים מספרים בכל צומת . כתבו אלגוריתם ליניאר י בגודל העץ, שימצא וידפיס את סכום האבות הקדמונים של כל צומת . סכום אבות קדמונים מוגדר כסכום המספרים , בצומת עצמו, ובכל ההורים הקדמונים שלו.

פיתרון:

```
#include<stdio.h>
#include<conio.h>
#include <stdlib.h>
#define s_size 50

typedef struct node{
      int  value;
      int sum;
      struct node *right,*left;
      int fleft;
      int fright;
}node;

typedef struct stack{
      node n;
}stack;
stack s[s_size];
int top=0;



struct node* insert (struct node *p,int num);
//void remove (struct node *p,int num);
void inorder_rec (struct node *p);
void inorder_reg (struct node *p);
void postorder_rec (struct node *p);
void postorder_reg (struct node *p);
```

```
void preorder_rec (struct node *p);
struct node*  pop( );
struct node*  find(struct node *p,int num);
void push(struct node *p);
int find_min(struct node *p);
int find_max(struct node *p);
int find_dis (struct node *p,int left, int right);
struct node * get_lca(struct node *p,int left,int right);
void preorder_sum (struct node *p,int sum);

int main()
{
	node *head_ptr=NULL;
	node *print=NULL;


	head_ptr=insert(head_ptr,8);
	head_ptr=insert(head_ptr,20);
	head_ptr=insert(head_ptr,11);
	head_ptr=insert(head_ptr,3);
	head_ptr=insert(head_ptr,7);
	head_ptr=insert(head_ptr,1);
	printf("\ninorder_rec:\n");
	inorder_rec(head_ptr);
	printf("\ninorder_reg\n");
	inorder_reg(head_ptr);
	printf("\npostorder_rec:\n");
	postorder_rec(head_ptr);
	printf("\npostorder_reg:\n");
	postorder_reg(head_ptr);
	printf("\npreorder_rec:\n");
	preorder_rec(head_ptr);
	printf("\nwhat found:");
	print=find(head_ptr,10);
	if (print!=NULL)
	printf("\n%d",print->value);
	printf("\nthe min is: %d",find_min(head_ptr));
	printf("\nthe max is: %d",find_max(head_ptr));
	print=get_lca(head_ptr,1,20);
	if (print!=NULL)
	printf("\n\nthe lca is: %d", print->value);
	print=get_lca(head_ptr,1,8);
	if (print!=NULL)
	printf("\n\nthe lca is: %d", print->value);
	printf("\n\nthe fathers sum is:");
	preorder_sum(head_ptr,0);
	printf ("\n\nthe dis is: %d",find_dis(head_ptr,3,20));



getch();
return 0;
}

struct node* insert (struct node *p,int num)
{


	if (p==NULL)
	{
		p = malloc( sizeof( struct node ) );
```

```
            p->value=num;
            p->fleft=1;
          p->fright=1;
            p->sum=0;
          p->left=NULL;
          p->right=NULL;
      }

      else{if (num<p->value) p->left=insert(p->left,num);
      else p->right=insert (p->right,num);}

      return p;
}
struct node*  find(struct node *p,int num)
{
      if (p==NULL)
      {printf("value not found"); return NULL;}
      if (p->value==num)return p;
      if (num<p->value) return find(p->left,num);
      if (num>p->value) return find (p->right,num);
}
int find_min(struct node *p)
{
      while(p->left!=NULL)
            p=p->left;

      return p->value;
}
int find_max(struct node *p)
{
      while(p->right!=NULL)
            p=p->right;

      return p->value;
}



void inorder_rec (struct node *p)
{
      if (p==NULL) return;
      else {
            inorder_rec(p->left);
            printf (" %d ", p->value);
            inorder_rec(p->right);
            return;}
}
void preorder_rec (struct node *p)
{
            if (p==NULL) return;
      else {
            printf (" %d ", p->value);
            preorder_rec(p->left);
            preorder_rec(p->right);
            return;}
}

void push(struct node *p)
{
      if (top==s_size) printf ("stack is full");
      s[top].n=*p;
```

```
	top++;
	return;
}
void postorder_rec (struct node *p)
{	if (p==NULL) return;
	else {
		postorder_rec(p->left);
		postorder_rec(p->right);
		printf (" %d ", p->value);
		return;}
}

struct node*  pop( )
{

  if(top == 0)
  {
	printf("\n\t STACK EMPTY...\n\n");
		return 0;
  }
    else
      {
       top--;

         return &(s[top].n);
    }
}

void inorder_reg (struct node *p)
{
	while (1){
	while (p!=NULL    )
	{
		push(p);
		p=p->left;
	}
	if (top==0) break;
	p=pop();
	printf (" %d ", p->value);
	p=p->right;}
}

void postorder_reg (struct node *p)
{

	while(1){
		while (p!=NULL && p->fleft)
		{
			p->fleft=0;
			push(p);
			p=p->left;
		}
		if (top==0) break;
		p=pop();
		if (p->right!=NULL && p->fright)
		{
			p->fright=0;
			push(p);
			p=p->right;
		}
		else printf (" %d ", p->value);}
```

```
}

/*void remove (struct node *p,int num)
{
	node *print=NULL;
	if (print=find(p,num)==NULL) printf("\nno such value");
	if (print->left==NULL && print->right==NULL) //leaf
		free (print);
	if (print->left && print->right==NULL) //left son*/

struct node * get_lca(struct node *p,int left,int right)
{
	if (p==NULL) return NULL;
	if (find(p,left)==NULL || find(p,right)==NULL)return NULL;

	while (p)
	{
		if (p->value<left && p->value<right)
			p=p->right;
		else if (p->value>left && p->value>right)
			p=p->left;
		else return p;
	}

	return NULL;
}

void preorder_sum (struct node *p,int sum)
{
		if (p==NULL) return;
	else {
		printf (" %d ", p->value+sum);
		preorder_sum(p->left,p->value+sum);
		preorder_sum(p->right,p->value+sum);
		return;}
}

int find_dis (struct node *p,int left, int right)
{
	int flag=1;
	int dis=0;
	node* common=NULL;
	node* temp=NULL;
	common=get_lca(p,left,right);
	temp=common;

	//search left
	while (flag)
	{
		if (common->value>left)
		{
			common=common->left;
			dis++;
		}
		else if (common->value<left)
		{
			common=common->right;
			dis++;
		}
		else flag=0;
	}
```

```
	flag=1;
	common=temp;
	//search right
		while (flag)
	{
		if (common->value>right)
		{
			common=common->left;
			dis++;
		}
		else if (common->value<right)
		{
			common=common->right;
			dis++;
		}
		else flag=0;
		}
		return dis;
}
```

שאלה 16

נדרש לממש מערך n-מימדי שמימדיו הם $(N_1,N_2,...N_n)$ (כלומר הקואורדינטה הכי "נמוכה" היא $(0,0,...0)$ והקואורדינטה הכי "גבוהה" היא $(N_1-1,N_2-1,...N_n-1)$).

לצורך פתרון הבעיה הוצע להשתמש במערך חד-מימדי (ווקטור) באורך $\prod_{j=1}^{n} N_j$ ולתרגם כל גישה לקואורדינטה $(i_1,i_2,...i_n)$ במערך ה-n-מימדי לאינדקס I במערך החד-מימדי בצורה הבאה:

$$I=\sum_{j=1}^{n}(i_j \cdot \prod_{k=j+1}^{n} N_k).$$

א) האם מספיק לבדוק האם הקואורדינטה $(i_1,i_2,...i_n)$ היא חוקית ע"י הבדיקה $0 \le I < \prod_{j=1}^{n} N_j$?

ב) האם ניתן לתרגם אינדקס I במערך החד-מימדי לקואורדינטה $(i_1,i_2,...i_n)$ במערך ה-n-מימדי באופן חד ערכי? אם כן, הסבירו בקצרה איך זה אפשרי. אם לא, תנו דוגמה.

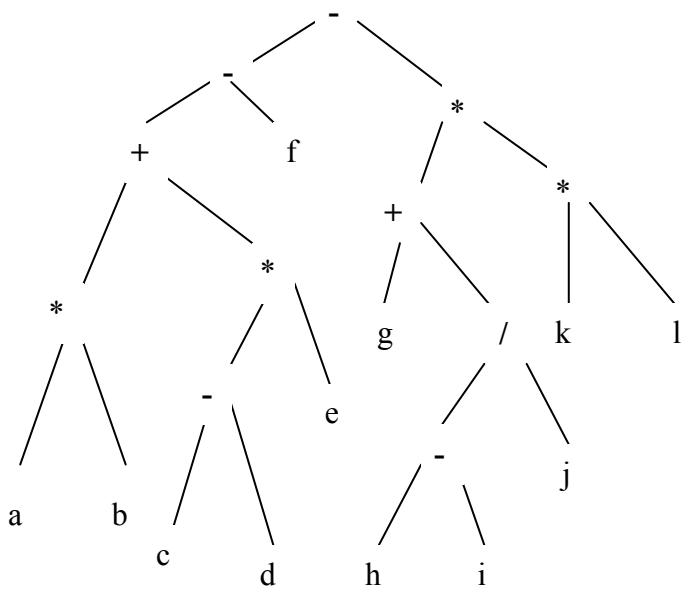
שאלה 17

כידוע ישנם שלושה סוגי סריקות של עצים: תחילי (preorder), תוכי (inorder), סופי (postorder). בד"כ כותבים ביטויים מתמטיים בכתיב תוכי. מחשבון HP מקבל קלט כביטוי סופי, שפות כמו Lisp ו-Scheme עובדות עם ביטויים תחיליים. פונקציות כתובות כביטוי תחילי, קודם שם הפונקציה ואז הארגומנטים. הדוגמא הבאה היא של עץ ביטויים. הפעולות מיוצגות בצמתים פנימיים, והערכים בעלים. סריקה תוכית של העץ תיתן את הביטוי הבא:

a * b + c - d * e - f- g + h - i / j *k * l

בכדי שסדר העדיפויות יהיה ברור ניתן לשים סוגריים. הסוגריים באים רק בכדי להראות את סדר הפעולות אבל אינם חלק מהעץ.

((((a * b) + ((c - d) * e)) - f) - ((g + ((h - i) / j)) * (k * l)))



א. האם ניתן לשחזר את העץ מתוך הסריקה התוכית שנתונה למעלה (עם הסוגריים). אם כן, יש לתת אלגוריתם לשחזור העץ. אם לא יש להוכיח מדוע לא ניתן (לתת דוגמא נגדית).

ב. יש לתת את הסריקה התחילית והסופית של עץ הביטויים שלמעלה.

ג. יש לכתוב פונקציה בשם CalcTree המקבלת מצביע לעץ T, ומחזירה את הערך שצריך להיות בראש העץ. הפונקציה תהיה חלק מהתכנית המורכבת באופן הבא:

1. הגדרת צומת בעץ:

```
typedef struct node {
        union t {
                char   oper;
                int    number;
                };
        int    leaf;     // leaf == 1 (int in union), ==0 (oper in nonleaf)
         struct node * left, right;
} NODE;
```

2. פונקציה שתקרא ביטוי תוכי (עם סוגריים) – ותבנה ממנו עץ.

```
NODE * BuildTree( char *expression)
```

3. הפונקציה תהיה לא רקורסיבית, ותשתמש במחסנית בכדי לחשב את הביטוי שנמצא בעץ.

4. יש לכתוב את **כל התכנית** (ממש ב- C). כולל כל הפונקציות המוזכרות לעיל.

<u>שאלה 18</u>

נתון מערך באורך n, כאשר m האברים הראשונים במערך ממוינים בסדר עולה , ויתר האברים (n-m) מסומנים כריקים. m אינו ידוע.

תארו אלגוריתם המחפש אם איבר בעל ערך k קיים במערך. יש לעשות זאת בסיבוכיות של log m (ולא log n שהיא טריוויאלית).

שאלה 19

בהינתן צומת u נסמן ב-Tree-Successor(u) את העוקב בסדר in-order של u ונסמן ב--Tree Predecessor(u) את הקודם ל-u (שוב ב in-order.

הוכח או הפרך.

א. יהי T עץ חיפוש בינארי ויהי u צומת בעץ שאינו עלה אז $\mathrm{Tree}-\mathrm{Successor}(u)$ הוא עלה

ב. יהי T עץ חיפוש בינארי ויהי u צומת בעץ שאינו עלה אז $Tree-Predecessor(u)$ הוא עלה

ג. פעולת המחיקה מעץ חיפוש בינארי היא חילופית (כלומר העץ המתקבל ממחיקת x ולאחריו y הוא אותו העץ כמו למחוק את y ולאחריו את x)

ד. אם לצומת בעץ חיפוש בינארי שני בנים, אזי לצומת העוקב (בסדר inorder) אין בן שמאלי ולצומת הקודם (בסדר inorder) אין בן ימני.

שאלה 20

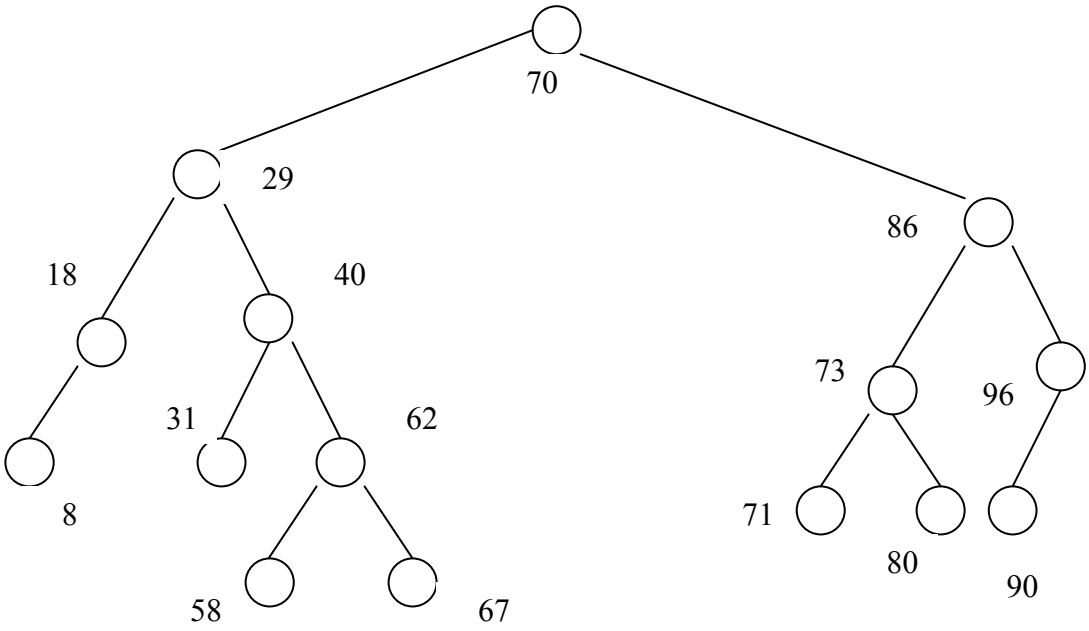
נתון מערך $T[1..n]$ ממוין של שלמים שונים זה מזה.

כתבו אלגוריתם המחפש אינדקס i כך ש-: $T[i]=i$. השגרה תחזיר את i אם הוא קיים, או 1- אחרת. נדרש זמן ריצה $\Theta(\log(n))$.

פרק שלישי – עצים מאוזנים

שאלה 1

נתון עץ AVL :



א. יש לצייר את העץ לאחר הכנסת הערך 68 לעץ שלמעלה.

ב. יש לצייר את העץ לאחר הכנסת הערך 84 לעץ שלמעלה.

ג. יש לענות לכל אחת מהטענות האם היא נכונה או לא ולנמק במספר שורות .

נתון עץ AVL לא ריק אליו הוכנס הערך k שלא היה בו קודם:

i. לאחר ההכנסה לא ייתכן ש- k יופיע בשורש העץ.

ייתכן מצב בו k+1 ימצא בראש העץ, ו- k-1 יהיה הבן השמאלי שלו, ואז k יופיע בראש העץ אחרי גלגול LR.

ii. אם k הוא בראש העץ אזי הוא חייב להיות הערך האמצעי (הווה אומר שההפרש ביו מספר האברים הקטנים ממנו ומספר האברים הגדולים ממנו הוא לכל היותר 1).

לא נכון, ניתן למצוא הרבה דוגמאות של עץ מאוזן שלא מקיים את הכתוב לעיל

iii. אם במהלך ההכנסה בוצע גלגול LL אזי k הוא הערך הקטן ביותר בעץ.

לא נכון, יכול להיות גלגול LL על תת עץ מסוים (אפילו בצד ימין)

iv. אם במהלך ההכנסה בוצע גלגול RR אזי לא ייתכן ש- k הוא האיבר הקטן ביותר בעץ.

נכון, כיוון שהכנסנו ערך ימינה לצומת, וכיוון שהעץ הוא עץ חיפוש, שורש תת העץ עליו ביצענו גלגול זה, חייב להיות קטן מ – k.

v. לאחר ההכנסה האיבר המופיע בשורש העץ הוא האיבר האמצעי .

שאלה 2

נקודות בריבוע היחידה במישור נתונות ע"י זוג הקואורדינטות שלהן (x,y).

יש להציע מבנה נתונים התומך בכל הפעולות הבאות: (מספר הנקודות בריבוע הוא n)

א. insert(x,y) הכנסת הנקודה ב – $O(\log n)$.

ב. delete(x, y) הוצאת הנקודה ב – $O(\log n)$.

ג. line(m) הדפסת כל הנקודות (x_i,y_i) במבנה הנמצאות על הישר $x_i+y_i=m$. בזמן $O(k+\log n)$ כאשר k הוא מספר נקודות הפלט.

רמז: אפשר לשים לב שהישרים עליהם שואלים בפעולת line הינם מקבילים, וניתן לסדר אותם על פי הערך של m.

ניתן לסדר עץ חיפוש מאוזן ע"פ הערך של m. על כל צומת מסוג m יהיה תלוי עץ חיפוש מאוזן ע"פ ערכי x.
הכנסת הנקודה, תגרור הכנסת ה – m החדש לעץ הראשי (או מציאתו), ואז הכנסת הנקודה ע"פ ערך x שלה לתוך תת העץ של m.
הוצאת נקודה תגרור מציאת הm. אם מצאנו מוצאים את הנקודה ומוציאים אותה (אם ישנה כזו). אם זו האחרונה בתת העץ של m, מוציאים את הצומת m מהעץ.
שתי הפעולות האחרונות לכל היותר $2\cdot\log n$.
פעולת line תהיה מציאת הצומת של m, והדפסת תת העץ שלה.

<u>שאלה 3</u>

במערכת תכנון תהליכים קיימים בכל רגע נתון לכל היותר n תהליכים ולכל אחד מהם יש מזהה id ועדיפות priority משלו. לכל תהליך המזהה והעדיפות הם מספרים שלמים בתחום [0..n-1]

יש להציע מבנה נתונים התומך בפעולות הבאות: (בהתחלה המערכת ריקה)

א. insert(id) יש להוסיף למערכת תהליך עם מזהה id ועדיפות 0 (אין שני תהליכים עם אותו id).

ב. delmax() יש להוציא מהמערכת תהליך עם עדיפות מקסימאלי ת. אם יש כמה כאילו ש להוציא אחד מהם שרירותית.

ג. inc(id) - יש להגדיל את העדיפות של תהליך id ב – 1. (העדיפות לא גולשת מעל n).

ד. dec(id) – יש להקטין את העדיפות של תהליך id ב – 1. (העדיפות לא יורדת מתחת ל 0).

ה. print(k) – יש להדפיס את k התהליכים עם העדיפות הגבוהה ביותר.

ו. find_priority() – יש למצוא את כל התהליכים עם עדיפות priority ולהדפיס אותם.

סיבוכיות הזמן:

- insert(id), delmax(), inc(id), dec(id) - סיבוכיות $O(1)$ במקרה הגרוע.
- print(k), find_priority() - $O(k)$ במקרה הגרוע כאשר k הוא מספר התהליכים המודפסים.

סיבוכיות מקום: $O(n)$.

שני מערכים:

א. אחד של עדיפויות עם רשימה מקושרת דו כיוונית, לכל ה – id ים, כל פעם מוסיפים צומת בתחילת הרשימה הדו כיוונית, ומצביעים חזרה לעדיפות המתאימה.

ב. של ה- id עם מצביע לעדיפות המתאימה.

מוסיפים תא מקסימום שגדל כל פעם שעוברים אותו.

הכנסה: מוסיפים לעדיפות 0, לתחילת הרשימה המקושרת שלה והפוך חזרה מה- id אליה.

הוצאת מקס. : ע"פ התא, מוציאים את הראשון שמוצבע.

הגדלה: מנתקים מהרשימה המתאימה, מחברים לבאה בתור, ומעדכנים את המקס. אם צריך.

הקטנה:כנ"ל

הדפסה : ע"פ המקסימום.

מציאת עדיפות: הדפסת הרשימה המקושרת התלויה עליו.

שאלה 4

נתונים שני עצים ריקים – עץ AVL ועץ 2-3. מבצעים על שניהם את הפעולות הבאות:

1) insert(1), insert(2), …, insert(10) (סה"כ 10 הכנסות)
2) delete(4)

א. יש לצייר את כל אחד מהעצים לפני ואחרי ההוצאה.

ב. כעת הניחו שהיו רק פעולות הכנסה של מספרים 1,2,…,n בסדר עולה ומספרן גדול (מ 20-).

ג. באיזה עץ יש יותר עלים? באיזה עץ יש יותר צמתים פנימיים? הסבירו.

ד. כמה בנים יש לרוב הצמתים הפנימיים של עץ 2-3 (2 בנים, 3 בנים או שלא ניתן לקבוע בוודאות)? הסבירו.

ה. איזה מהגלגולים התבצעו לרוב בעץ AVL בזמן ההכנסות? הסבירו.

ו. בהינתן עץ AVL חוקי כקלט, האם תמיד ניתן לבנות עץ זהה (כולל המבנה הפנימי שלו) ע"י סדרת פעולות insert ו-delete המופעלות על עץ ריק? אם התשובה היא "כן", הציעו אלגוריתם העושה זאת והסבירו מדוע הוא נכון. אם התשובה היא "לא", תנו דוגמא של עץ AVL חוקי, אותו אי אפשר לבנות ע"י סדרת פעולות insert ו-delete בלבד והסבירו למה אי אפשר.

ניתן לבנות בשכבות עם תור.
מכניסים את השורש לתור
מוציאים את הצומת הראשונה, ומכניסים במקומה את ילדיה לתור.

שאלה 5 – שאלת תכנות

יש לתאר מבנה נתונים + פירוט אלגוריתמים ולהוכיח סיבוכיות זמן ומקום . אח"כ יש לתכנת.

בצבא של מדינת ליליפוט ישנן משימות לביצוע כאשר לכל משימה ניתנת עדיפות . לכל משימה ניתן מספר זהות שהוא ייחודי למשימה, אבל לכמה משימות יכולה להיות אותה רמת עדיפות .
ניתן להניח כי אין הגבלה על מספר המשימות או על גודל העדיפות של משימה (כמובן שמסיבות טכניות, יש מגבלה במחשב, אבל לצורך החישובים התייחסו למספרים כגודל בלתי מוגבל). המספרים המיוצגים שלמים.

מבנה הנתונים צריך לתמוך בפעולות הבאות:

Init(k) – אתחול מבנה הנתונים. הפונקציה מקבלת מספר טבעי k ששימושו יוסבר בהמשך. הפונקציה תחזיר * void , מצביע למבנה הנתונים שנבחר. אם האתחול נכשל יוחזר NULL . סיבוכיות O(1).

New_Task(id, p) - יצירת משימה חדשה בצבא שהמזהה שלה הוא id (בתנאי שלא קיימת כזו) והעדיפות הינה p. הפעולה תחזיר את ה- id , אך אם קיימת פעולה עם id כזה יוחזר 1-. סיבוכיות O(log n).

Change_Priority(id, p) - שינוי עדיפות של המשימה id לעדיפות חדשה והיא p . אם ישנה משימה כזו יוחזר ה- id שלה, אחרת יוחזר 1-. סיבוכיות O(log n).

Remove_Task(id) – הסרת המשימה id . אם ישנה משימה כזו יוחזר ה- id שלה, אחרת יוחזר 1-. סיבוכיות O(log n).

Get_Med_Prio() - לצורך סטטיסטיקה ביקש גוליבר לדעת את ה- id של המשימה שעדיפותה היא החציון מבין כל העדיפויות, וה- id שלה הוא המינימאלי מבין כל המשימות עם העדיפות הזו. אם אין משימה כזו, (המבנה ריק) יוחזר 1-. סיבוכיות O(1).

להזכירכם, חציון הינו מספר שמחצית המספרים קטנים או שווים לו. החציון יילקח על ערכי העדיפויות ללא תלות במספר המשימות עם אותה עדיפות.

Next_Task() – פעולה זו תחזיר את ה- id של המשימה הבאה לביצוע, מתוך המשימות הממתינות במבנה. **המשימה עם העדיפות הגבוהה ביותר, ומתוכה עם ה-id המינימאלי מבין כל אילו עם העדיפות הגבוהה ביותר, אלא אם קיימת משימה ותיקה**. משימה "ותיקה" היא משימה שמאז שנכנסה למבנה בוצע k או יותר פעמים Next_Task(). יש לבחור את המשימה ה"ותיקה" ביותר מבין כל המשימות ה"ותיקות", אם ישנן כאילו. ז.א. זו שנכנסה ראשונה. על הפונקציה להחזיר את ה- id של משימה זו שאמורה להתבצע עכשיו, או -1 אם אין כזו. סיבוכיות O(1) .

סיבוכיות המקום של מבנה הנתונים הוא O(n).

<u>שאלה 6 – שאלת תכנות</u>

יש לתאר מבנה נתונים + פירוט אלגוריתמים ולהוכיח סיבוכיות זמן ומקום . אח"כ יש לתכנת.

במוסד הלימודים "נורס גראודה" החליטו לנסות שיטת דירוג חדשה לכיתות לפי הציונים של התלמידים .
ב"נורס גראודה" ישנם n סטודנטים, המחולקים ל-k כיתות.
לכל כיתה ישנו Class_Number - מס' הכיתה ו-Class_Index – מס' התלמידים המצטיינים בכיתה.
תלמיד מוגדר כמצטיין אם ממוצע הציונים שלו הוא 85 לפחות.

יש להציע מבנה נתונים שיתמוך בשאילתות הבאות. להלן רשימת השאילתות בהם תומך המבנה:

<u>Init()</u> :
<u>קלט</u> : רשימת k כיתות. (לאו דווקא מספרים רצופים)
איתחול מבנה נתונים.
סיבוכיות זמן $O(nlog(n))$.

<u>Range</u>
<u>קלט</u> : טווח (a,b)(שני מספרים שלמים,a<b).
החזרת מס' הכיתות בהן Class_Index הוא בטווח הנתון.
סיבוכיות זמן $O(log(k))$.

<u>Level_Up</u>
<u>קלט</u> : Class_Index.
יש להעביר את התלמיד המצטיין עם הממוצע הכי גבוה מכיתה עם Class_Index נתון לכיתה עם Class_Index הקטן ביותר שגדול ממש מ-Class_Index הנתון.
אם לכמה כיתות יש אותו Class_Index,יש לבחור בכיתה עם מס' הכיתה (Class_Number)הגדול ביותר ולהעביר את התלמיד לכיתה עם מס' הכיתה הקטן ביותר.
סיבוכיות זמן $O(log(k) + log(n1) + log(n2))$,כאשר n1 ו-n2 – מס' התלמידים בשתי הכיתות.

<u>Merge</u>
<u>קלט</u> : Class_Index.
יש לאחד שתי כיתות לכיתה אחת. יש לאחד את הכיתה עם Class_Index נתון, עם הכיתה בעלת Class_Index הקטן ביותר שגדול ממש מ-Class_Index הנתון.
אם לכמה כיתות יש אותו Class_Index,אין לבצע דבר.
סיבוכיות זמן$O(log(k) + n1 + n2)$,כאשר n1 ו-n2 – מס' התלמידים בשתי הכיתות.

<u>End</u>:
משחרר את כל המבנים ולא מדפיס כלום.
סיבוכיות זמן $O(n)$.

<u>הערות:</u>
סיבוכיות המקום של המבנה הינה $O(n)$.יש להחליט לבד לגבי מקום נוסף.

פיתרון:

סיבוכיות המקום של כל הפעולות היא $O(n)$

int init(char *input) $O(n\log n)$

הסבר על הפונקציה

הקובץ מקבל את הכתובת של הקובץ שבו יש רשימה של כיתות (בסדר שרירותי) עם שדה: מספר כיתה ורשימת סטודנטים (בסדר שרירותי) כל סטודנט מכיל שדות : שם , מספר ו ממוצע.
עבור כל כיתה נעבור על רשימת סטודנטים שלה ונמנה את מספר המצטיינים . נבנה עץ AVL כשהמפתח הראשי הוא "מספר מצטיינים בכיתה" ומפתח משני הוא : " מספר כיתה" ומכל צומת יש מצביע לעץ שהמפתח הראשי : "ממוצע של סטודנט" ומפתח משני : "מספר סטודנט" .
נרוץ על עץ כיתות לפי INORDER ונעדכן לכל צומת את מספר הצמתים עד עליו.
הפונקציה מחזירה 1 אם הצליחה , אחרת 0.

int _Range(Min,Max) $O(\log(k))$

הסבר על הפונקציה

נרוץ על עץ כיתות ונגיע עם מצביע ל מפתח ראשי MIN ומפתח משני הכי קטן.
נרוץ עם מצביע שני על עץ כיתות ונגיע למפתח ראשי MAX ומפתח משני הכי גדול.
בכל צומת יש שדה של מספר צמתים עד עליו ב INORDER.
נחסר את השדה מ ה MAX את ה MIN . נקבל את מספר כיתות בטווח.
נחזיר את התשובה.

int Level_Up (int class_index) $O(\log(k*n_1*n_2))$

נרוץ על עץ כיתות ונגיע עם מצביע ל מפתח class_index משם נרוץ בעץ של הסטודנטים למפתח ראשי ומשני הכי גדול ונשלוף אותו מהעץ ונוסיף אותו לעץ שמצביע עליו כיתה עם מפתח ראשי class_index+1. נעדכן את המפתח הראשי של הכיתות ששינינו.

int Merge(int class_index)

נרוץ על עץ כיתות ונגיע עם מצביע ל מפתח class_index, נבדוק שיש אחד כזה .נרוץ על עץ הסטודנטים של הכיתה ב INORDER ונכניס אותם למערך A. נרוץ על עץ הסטודנטים של כיתה class_index+1 ב INORDER ונכניס אותם למערך B. נעשה מערך C ממוין שיכלול איחוד בין 2 המערכים ממוינים(בעזרת 2 מצביעים).ממערך ממוין C ניצור עץ כמעת שלם בצורה רקורסיבית : אמצע מערך הוא השורש ,בן שמאלי שלו הוא אמצע של מערך מצד שמאל של השורש ובן ימני הוא אמצע של מערך ימני מהשורש וכ"ו.נצרף את העץ לכיתה class_index+1 , נמחק את הכיתה המיותרת ואת עצים המיותרים+ נעדכן את המפתח הראשי בעץ כיתתות.

int End() $O(n)$

הסבר על הפונקציה

נרוץ על כל העצים רקורסיבית ונשחרר את המבנים.

שאלה 7 – תרגיל תכנות:

יש לתאר מבנה נתונים + פירוט אלגוריתמים ולהוכיח סיבוכיות זמן ומקום. אח"כ יש לתכנת.

בתרגיל זה נבנה מבנה נתונים אשר יסייע לספריה במעקב אחר השאלות ספרים.
הספריה תשמור מידע עבור כל משאיל ועבור כל ספר.
לכל ספר יש שם ספר, שם מחבר, מספר סידורי (מס"ד) אשר יינתן ע"י הספרן וזמן השאלה מירבי מותר. ייתכנן יותר מעותק אחד של ספר, אך לא ייתכן מס"ד זהה לשני ספרים שונים. כל מס"ד יהיה מספר שלם הגדול מ 0.
זמן ההשאלה המירבי של ספר יכול להיות יום, שלושה ימים או שבוע.
לכל מנוי יש שם פרטי, שם משפחה ומספר תעודת זהות (ייחודי).
לכל מנוי מותר לשאול לכל היותר שלושה ספרים בו זמנית . מנוי אשר מאחר בהחזרת ספר לא יכול ל שאול ספר חדש.

על התוכנית שלכם לתמוך בפעולות הבאות:

- אתחול ספריה ללא ספרים וללא מנויים.
- קניית ספר חדש.
- מכירת ספר קיים (הוצאה ממבנה הנתונים).
- הוספת מנוי חדש.
- ביטול מנוי קיים.
- השאלת ספר למנוי.
- "מעבר יום" (העברת התאריך יום אחד קדימה).
- הדפסה של כל המנויים הנמצאים באיחור.

לצורך ניתוח הסיבוכיות, n הוא מספר הספרים הנוכחי וm הוא מספר המנויים הנוכחי.
עליכם לממש את הפונקציות הבאות בקובץ הנקרא בשם library.c. כמו כן, מותר להוסיף עוד שני קבצי עזר – sorted_ds.h ו sorted_ds.c.

<u>תיאור הפונקציות:</u>

אתחול
שם הפונקציה – Init
קלט – אין
פלט – מצביע למבנה הנתונים אם הצליח, מצביע ל NULL(0) אם לא הצליח.
סיבוכיות זמן נדרשת – $O(1)$.

קניית ספר
שם הפונקציה –AddBook
קלט – מצביע למבנה הנתונים, מס"ד, שם ספר, שם מחבר וזמן השאלה.
פלט – ERROR1 אם מס"ד קיים, ERROR2 קלט לא תקין. אם הקנייה הצליחה מחזיר SUCCESS.
סיבוכיות זמן נדרשת – $O(\log(n))$.

מכירת ספר קיים
שם הפונקציה –SellBook
קלט - מצביע למבנה הנתונים, מס"ד.
פלט – ERROR1 אם מס "ד לא קיים , ERROR2 הספר בהשאלה , אם המכירה הצליחה מחזיר SUCCESS.
סיבוכיות זמן נדרשת – $O(\log(n))$.

הוספת מנוי
שם הפונקציה –AddMember
קלט – מצביע למבנה הנתונים, מספר תעודת זהות, שם פרטי ושם משפחה.
פלט – SUCCESS הפונקציה הוסיפה את המנוי , ERROR1 אם מספר תעודת זהות קיים , ERROR2 קלט לא תקין.
סיבוכיות זמן נדרשת – $O(\log(m))$.

ביטול מנוי קיים
שם הפונקציה –RemoveMember
קלט – מצביע למבנה הנתונים, מספר תעודת זהות.
פלט – SUCCESS הפונקציה מחקה את המנוי , ERROR1 אם מס"ד לא קיים, ERROR2 המנוי מחזיק ספרים.
סיבוכיות זמן נדרשת – $O(\log(m))$.

השאלת ספר למנוי
שם הפונקציה –LoanBook
קלט – מצביע למבנה הנתונים, מספר תעודת זהות ומס"ד של הספר.
פלט – SUCCESS ביצענו השאלה תקינה, ERROR1 אם מספר תעודת זהות לא קיים , ERROR2 אם מס"ד לא קיים , ERROR3 המנוי מחזיק ספר באיחור , ERROR4 המנוי מחזיק שלושה ספרים (ושלושתם לא באיחור).
סיבוכיות זמן נדרשת – $O(\log(m)+\log(n))$.

מעבר יום
שם הפונקציה –NextDay
קלט – מצביע למבנה הנתונים.
פלט – מספר הספרים שעברו לאיחור.
סיבוכיות זמן נדרשת – $O(1)$.

הדפסה של כל המנויים שנמצאים באיחור.
שם הפונקציה –MemberReturnLate
קלט – מצביע למבנה הנתונים.
פלט – רשימה מקושרת של מספרי תעודת זהות של כל המנויים שמחזיקים ספרים באיחור . מותר שמנוי יופיע ברשימה יותר מפעם אחת, כמספר הספרים שהוא מחזיק באיחור.
סיבוכיות זמן נדרשת – $O(k)$. k הוא אורך הרשימה המודפסת.

הפתרון שלנו מורכב ממשתנה תאריך, שני עצי AVL ותשע רשימות מקושרות.
עצי הAVL הם בדיוק כפי שנלמדו בכיתה.
הרשימות המקושרות הן רשימות חד כיווניות, כאשר הגישה לאיברים מתבצת בעזרת מצביע לראש הרשימה ולסוף הרשימה. כמו כן, כל רשימה מחזיקה מונה של מספר האיברים ברשימה.

- עץ AVL ראשון הוא עץ ספרים . העץ ממוין לפי מ ס"ד של הספר , כאשר עבור כל ספר שומרים את כל הפרטים שלו וכן שומרים מספר ת "ז של המנוי שמחזיק את הספר (0 אם הספר איננו מושאל).
- עץ AVL שני יהיה של מנויים. העץ ממוין לפי מספר ת"ז של המנוי. לכל מנוי שומרים את הפרטים של המנוי וכן שומרים פרטים של שלושה ספרים אותם ה מנוי מחזיק. עבור כל ספר שהמנוי מחזיק שומרים מס "ד של הספר (0 אם לא משאיל ספר), וכן את התאריך האחרון אשר בו מותר **להחזיר** את הספר ללא עונש.
- נחזיק שמונה רשימות שנמספר אותן 0..7 . הרשימה ה i תחזיק את מספרי המנויים שמותר להם להחזיר ספר עוד (בדיוק) i ימים.
- הרשימה התשיעית תחזיק את מספרי המנויים אשר מחזיקים ספרים באיחור.

כעת נתאר בקצרה את הפעולות על מבנה הנתונים:

1. *אתחול*: נאתחל את משנה היום להיות 1. שני עצי AVL נאתחל ריקים וכן נאתחל תשע רשימות ריקות (בגודל 0).
2. *קניית ספר*: בודקים אם הספר (מס"ד) איננו נמצא עדיין בעץ הס פרים. אם לא, אז נוסיף את הספר לעץ הספרים.
3. *מכירת ספר*: מחפשים את הספר בעץ הספרים . אם הספר נמצא , בודקים אם הספר מושאל. אם הספר לא מושאל לאף מנוי, אז מוציאים את הספר מעץ הספרים.
4. *הוספת מנוי*: מוסיפים את המנוי לעץ המנויים אם הוא איננו קיים עדיין.
5. *ביטול מנוי*: חיפוש המנוי בעץ המנויים . אם המנוי נמצא, בודקים אם המנוי מחזיק ספר (אחד או יותר). אם המנוי לא מחזיק אף ספר, מוציאים את המנוי מעץ המנויים.
6. *השאלת ספר ע"י מנוי*: מחפשים את הספר בעץ הספרים . בודקים שהספר איננו מושאל לאף מנוי וזוכרים את זמן ההשאלה המותר לספר (חלק מהפר טים של הספר). כעת מבצעים חיפוש של המנוי בעץ המנויים . אם המנוי קיים , מוודאים שהמנוי איננו מחזיק ספר שזמן ההחזרה שלו קטן מהתאריך של היום (אם זמן ההחזרה קטן יותר , סימן שכבר היה עלינו להחזיר את הספר !). אם המנוי מחזיק פחות משלושה ספרים , מוסיפים את הספר אותו משא ילים לתוך נתוני המנוי וכן רושמים עבור הספר את התאריך בו יש להחזיר אותו. השלב הבא הוא עדכון בעץ הספרים שהשאלנו את הספר למנוי הנתון . שלב אחרון בהשאלת הספר היא הוספת מספר המנוי לרשימת הספרים שיש להחזיר בעוד i ימים (i הוא זמן ההשאלה המותר לספר).
7. *מעבר יום*: שלב ראשון, מגדילים את מונה היום ב 1. שלב שני , זוכרים כמה ספרים נמצאים ברשימה 0 ומוסיפים את רשימה 0 לרשימת המנויים באיחור (שימו לב שאיחוד של שתי רשימות מתבצע ב $O(1)$ זמן). כעת עוברים על רשימות 1..7 ומעבירים אותן יום אחד קדימה. השלב הבא הוא לאתחל את רשימה 7 להיות רשימה ריקה. כעת נותר רק להחזיר את גודל רשימה 0 ששמרנו בצד.
8. *הדפסת מנויים באיחור*: הדפסה של רשימת המנויים באיחור.
9. *סיום עבודה*: פשוט משחררים את כל הרשימות (עם כל האיברים) וכן שני עצי AVL.

שאלה 8:

לאולימפיאדה הקרובה נרשמים שחיינים לנבחרת ארה"ב. כל שחיין שנרשם מוסר את זמן השחייה שלו בדקות. רוצים לבנות מבנה נתונים שיכיל את רשימת השחיינים כשהמפתח הוא זמן השחייה שלו. בנוסף רוצים לחלק את השחיינים לקבוצות ע"פ ההישגים שלהם.
יש לבצע את הפונקציות הבאות על מבנה הנתונים:

Init : אתחול מבנה הנתונים בזמן קבוע. אין להדפיס כלום.

AddSwimmerAch(intMin) : מכניס את זמן השחייה (בדקות שלמות) של שחיין חדש לקבוצה. אם יש כבר זמן כזה, לא מכניסים שחיין נוסף עם אותם השגים, ומדפיסים '0', אחרת מדפיסים '1'.

RemSwimmerAch(intMin) : מחליטים לבטל שחיין עם הישג כנ"ל. אם יש כזה, מבטלים ומדפיסים '1', אחרת מדפיסים '0'.

AddInterval(intMin) : מוסיפים נקודת זמן t_i שיחלק את הקבוצה באופן הבא:

$$t_1 < t_2 < t_3 < \ldots < t_k$$

כאשר השלמים הללו (נקודות הזמן) מחלקים את הקבוצה לסקציות כך ש:

$S_0 = \{x | x < t_1\}$, $S_1 = \{x | t_1 \le x < t_2\}$, .. , $S_{k-1} = \{x | t_{k-1} \le x < t_k\}$, $S_k = \{x | t_k \le x\}$,

כך שכך זמן y שהוכנס מציין סקציה יחידנית בין שתי נקודות זמן אליה הוא שייך. אם $y < t_1$ או $y \ge t_k$ אזי y מציין את הסקציות S_0 או S_k בהתאמה. אם עדיין לא הוכנסו נקודות זמן, אזי הסקציה היחידה שקיימת היא S_0
כאשר מכניסים נקודת זמן, מחלקים את הסקציה שמכילה אותה לשתי סקציות וקבוצת השחיינים שהזמנים שייכים לסקציה שלפני החלוקה, מתחלקת לשתי קבוצות ע"פ נקודת הזמן שנוספה.
אם כבר קיימת נקודת זמן כזו, מדפיסים '0', אחרת מדפיסים '1'.

RemInterval(intMin) : אם קיימת נקודת זמן כזו הסר אותה, והחזר '1', אחרת החזר '0'.

AvgOdd(y) : יש לזהות את הסקציה אליה שייך y, למיין את המספרים (הזמנים) באותה סקציה, ולהדפיס את **ממוצע הזמנים במקומות האי-זוגיים לאותה סקציה** (כאשר הזמן הראשון בתוך הסקציה הנ"ל יחשב ראשון, דהיינו אי-זוגי).

End : משחרר את כל ההקצאות והנתונים. זמן במקרה הגרוע $O(n + k)$, כאשר n הוא גודל הקבוצה ו- k הינו מספר נקודות הזמן. אין להדפיס כלום. חובה לשחרר את כל מבנה הנתונים בכדי לעבור את התרגיל !!

סיבוכיות הזמן של AddSwimmerAch ושל RemSwimmerAch תהיה $O(\log k + \log l)$, כאשר k הוא מספר נקודות הזמן ו – l הוא גודל (מספר הזמנים) הסקציה אליה שייך השחיין.

סיבוכיות הזמן של AddInterval ושל RemInterval תהיה $O(\log k + \log l)$, כאשר k הוא מספר נקודות הזמן ו – l הוא גודל (מספר הזמנים) הסקציה לפני החלוקה, או לפני האיחוד.

סיבוכיות הזמן של AvgOdd תהיה $O(\log k)$.

סיבוכיות מקום של מבנה הנתונים $O(k + n)$.

סיבוכיות המקום הנוסף של כל הפעולות תהיה $O(\log k + \log n)$, כאשר n הוא גודל הקבוצה כולה, ו –K הוא מספר נקודות הזמן.

דוגמא:

Command:	**Print:**
Init	
AvgOdd 4	0
AddSwimmerAch 3	1
AddSwimmerAch 1	1
AvgOdd 7	1
AddInterval 3	1

```
AddInterval 7                   1
AvgOdd 7                        0
AvgOdd 6                        3
AvgOdd 2                        1
AvgOdd 3                        3
AddSwimmerAch 5                 1
AddSwimmerAch 4                 1
AvgOdd 6                        4
ReSwimmerAch 6                  0
RemSwimmerAch 4                 1
AvgOdd 2                        1
AvgOdd 3                        4
RemSwimmerAch 3                 1
AvgOdd 5                        5
AvgOdd 4                        5
RemSwimmerAch  5                1
AvgOdd 5                        0
AvgOdd 3                        0
RemInterval 3                   1
AvgOdd 5                        1
RemSwimmerAch 1                 1
AvgOdd -3                       0
End
```

יש מספר פתרונות לתרגיל הזה,למשל עץ B+ מדרגה 3 שבכיתה נלמד איך עושים ODD_SUM או עץ AVL שכל צומת היא עץ AVL קטן. נפרסם פתרון של 2 עצי AVL שזה לדעתי פתרון הפשוט יותר.

מאוד חשוב להדגיש שזה הסבר מאוד כללי רק בשביל להעביר את הרעיון של הפתרון ולא פתרון רשמי.

כדי לא להסתבך עם ADT אני ממליץ לכם(לא חובה) להגדיר מבנה כללי (שמשותף לכל העצים) ובפנים לשמור מבנה של תכונות העץ.

```
typedef struct node *AvlTree;
typedef struct NodeInfo
{
	int swimmers[5];
	AvlTree  linkedIntervalToSwimmersTreeRoot;
	AvlTree  linkedIntervalToSwimmersTreeMin;
	AvlTree  linkedIntervalToSwimmersTreeMax;
}*NODEINFO;

// left,right, flag: for avl tree and info struct, key( or time or
interval)
typedef struct node
{
	NODEINFO Info;
	int key;
	int Flag;
	AvlTree Left;
	AvlTree Right;
}*AvlTree
	;
```

```
// trees height
int H[2];
// avlTree[0] = interval tree
// avlTree[1] = time tree
  AvlTree avlTree[2];
```

הרעיון של התוכנית:נגדיר 2 עצים.אחד אינטרוולים ואחד זמנים(של שחיינים) .כל צומת בעץ אינטרוולים מצביע על צומת בעץ זמנים שמייצגת את שורש קבוצת הזמנים(של האינטרוול) וכל תכונות הרגילות של עץ(מפתח, מצביע על בן ימני ועל בן שמאלי).
כל צומת בעץ זמנים מחזיקה משתנה שסופר את מספר הפעמים הוא השתנה(זוגי לאי-זוגי והפוך).כל צומת תחזיק סכום אי זוגיים וסכום זוגיים של כל הצמתים תחתיו,ומספר הצמתים שיהיו בענף השמאלי שלו וסך הצמתים שיהיו בענף הימני שלו(כדי לזהות בזמן הכנסה\מחיקה אם זה צומת זוגי או אי-זוגי),כדי לא לפגוע בסיבוכיות הוספתי מצביעים ל MAX ו MINכדי לא לרוץ על זמנים של אינטרוול אחר.
הבעיה שאסור לעדכן את כל האברים(זה לא עומד בסיבוכיות),אז עושים עדכון רק לצמתים שאתם עוברים מהצומת(שהוספתם\מחקתם) ועד השורש(הרי שורש זה הדבר היחידי שמעניין אותנו) וכמובן שכל צומת שעוברים(רק כשעולים ימינה) מוסיפים 1 למספר שינוי ומשנים את סכום זוגיים בסכום אי זוגיים,כך השורש יהיה תמיד מעודכן.
ולא צריך לדאוג לצמתים שלא עודכנו כי כל פעם שניגש לצומת בתוך העץ נעלה עד השורש כדי לבדוק מה הסכום שנשתמש(זוגיים או אי זוגיים).
לגבי מחיקה והוספה של אינטרוולים הפתרון אפילו יותר פשוט כי זה להוסיף אברים שכולם יותר גדולים מצמתים של אינטרוול ישן, לכן כל מה שעושים זה הולכים לשורש עץ (אם צמתים הגדולים) ומשנים את סכום זוגי ואי-זוגי לפי הצורך).

$O(1)$ **boolean Init()**

הסבר על הפונקציה

אתחול של כל המשתנים בזמן קבוע.

$O(\log k + \log l)$ **boolean AddSwimmerAch(intMin)**

$O(\log k + \log l)$ **boolean RemSwimmerAch(intMin)**

הסבר על הפונקציות

רצים על עץ אינטרוולים עד הצומת הנכונה (הצומת יותר קטנה שווה לזמן ו צומת שאחריה ב INORDER יותר גדולה) מגיעים דרך מצביע לשורש (תת עץ שחיינים) של זמני אינטרוול ועושים הוספה\מחיקה לפי חוקי AVL ולפי ההסבר שבהתחלה רצים על תת העץ לשורש ומעדכנים את הסכומים .

$O(\log k + \log l)$ **boolean Interval(intMin)**

$O(\log k + \log l)$ **boolean RemInterval(intMin)**

הסבר על הפונקציות

רצים על עץ אינטרוולים עד הצומת עושים מחיקה\הוספה ועושים עדכונים לפי חוקי AVL ולפי ההסבר למעלה מעדכנים את הסכום של סך הצמתים באינטרוול.

$O(\log k)$ **int *AvgOdd(intMin)***

הסבר על הפונקציה

מחפשים את הצומת בעץ האינטרוולים ,מגיעים דרך מצביעה לשורש (תת עץ זמנים)של זמני האינטרוול .ועושים חישוב ממוצע(לפי השדות (מספר צמתים וסכום אי זוגיים).

$O(n+k)$ **boolean End()**

הסבר על הפונקציה

רצים על הצים (למשל ב INORDER) ומשחררים את כל הצמתים .

סיבוכיות מקום של כל המבנה $O(n+k)$ **+ סיבוכיות מקום נוסף** $O(\log n + logk)$ **בגלל הרקורסיה.**

פרק רביעי – ערבול, קבוצות זרות, ערימה, ציונים, select

שאלה 1:
נתונה רשימה בת N שורות כאשר בכל שורה נתונים שני מספרים המציינים זמן התחלה וזמן סיום של פעולות שצריכות להיות מבוצעות על ידי K רובוטים, כאשר כל רובוט יכול לבצע פעולה אחת בו זמנית.

א. יש לתאר אלגוריתם ומבנה נתונים יעילים ביותר לקריאת הרשימה במעבר בודד תוך חלוקת הפעולות בצורה אופטימלית בין הרובוטים והכנת סדר התחלת וסיום הפעולות של כל רובוט . מהו זמן ביצוע האלגוריתם?

ב. באיזה מקרה K רובוטים לא יוכלו לעמוד בביצוע המשימה, האם וכיצד ניתן למנוע בעיה זו במקרים מסוימים.

א. מוציאים את k האיברים הראשונים, מכניסים לערימת מינימום ממוינת ע"פ זמן סיום. מוציאים את הראשון ומחליפים בבא בתור – Nlog k.
ב. יותר מ- k משימות חופפות בזמנים.

שאלה 2:
נתונה הקבוצה S, בכל אחד מהסעיפים יש להציע מבני נתונים לביצוע פעולות על קבוצות זרות:

פעולה	זמן הפעולה	union	find
איבר מינימאלי ב-S	O(1)	כמו בהרצאה	כמו בהרצאה
הדפסת האיברים בין p ל-q מתוך S	O(n)	כמו בהרצאה	כמו בהרצאה
הדפסת האיברים בין p ל-q מתוך S (ניתן להניח שיש מספר קטן O(1) של אברים)	O(log n)	O(log n)	O(log n)
הכנסת איבר x לתוך S	O(1)	כמו בהרצאה	כמו בהרצאה
פרק את S לשתי קבוצות באופן שרירותי, כך שאחת מכילה k איברים, והשניה |S|-k	O(log |S|)	O(log |S|)	O(log |S|)

1. אפשר לשמור מצביע אל האלמנט המינימלי בכל קבוצה. Union צריך לעדכן מצביע זה.
2. ניתן לממש ע"י חיפוש של p ו-q ומציאת אלמנטי הביניים.
3. ניתן לשמור איברי קבוצות בעצים ממוינים מאוזנים rank trees. אלמנטים עוקבים יוחזקו גם ברשימות (עבור פשטות בסריקות ההדפסה – הרשימות לא הכרחיות).
4. דורשת עדכון מצביע במערך האלמנטים (מניחים שיש תמיד מקום במערך זה).

5. ניתן לשמור איברי קבוצות בעצים ממוינים מאוזנים. ה-Split דורש איחוד תת-עצים מסוימים. ה- k הוא שרירותי, ולכן ניתן לבחור k כזה שיהיה ניתן לאזן אותו בקלות.

שאלה 3:
נתונה תוכנית מיון בשיטת QuickSort הבאה:

```
int partition(int *a, int left, int right, int pivot) {
        int i,j;
        for(i=left, j=right; i<=j; ) {
                for ( ; a[i] < pivot; i++) ;
                for ( ; a[j] > pivot; j--) ;
                if (i<= j) {
                        swap(a, i, j)
                        i++; j--;
                }
        }
        return i;
}

swap(int *a, int i, int j) {
        int temp;
        temp = a[i];
        a[i] = a[j];
        a[j] = temp;
}

void quick_sort(int *a, int left, int right) {
        int i, j;
        int pivot;
        if (left >= right) return;
        pivot = a[left];
        i = partition(a, left, right, pivot);

        quick_sort(a, left, i-1);
        quick_sort(a, i, right);
}
```

לתוכנית נותנים למיין מערך a שמכיל את המספרים 1, 2, 3, 4, 5, 6, 7, 8.
מהו סדר התחלתי של המספרים במערך שעבורו עומק הרקורסיה מינימלי ?

For example: 5,6,8,7,1,2,4,3
After the first step
=> 3,4,2,1 and 7,8,6,5
After the second step
=> 1,2 and 4,3 and 5,6 and 8,7

שאלה 4:

נתונה ערימה H הממומשת כעץ (לא כמערך), אבל להבדיל ממימוש הסטנדרטי, אין בעץ מצביעים מבנים לאב. יש להראות איך אפשר, בהינתן מצביע לשורש העץ ומספר הצמתים ב-H, לבצע פעולות סטנדרטיות של ערימות (insert ו- del_min) בסיבוכיות זמן O(log n).

insert :

בעזרת ייצוג בינארי של מספר הצמתים + 1 נוכל לדעת מה המסלול למקום של הצומת החדש בעץ כמעט שלם. נשמור את כל הצמתים במסלול כך שנוכל לבצע סיור מלמטה למעלה כשמבצעים sift-up.

Del_min:

- מציאת הצומת האחרון מעץ כמעט שלם – ייצוג בינארי
- החלפת הצמתים ו- sift-down – כמו במימוש הסטנדרטי

שאלה 5:

בהינתן k רשימות ממוינות , יש לתאר אלגוריתם שימזג את הרשימות הללו ב – $O(n \cdot \log k)$, כאשר n הוא מספר האברים סה"כ.

בונים ערימה מה – k הראשונים, ואז כל פעם מוציאים את המינימלי ומכניסים את הבא . הבניית $O(k)$ואח"כ כל איבר נכנס פעם אחת ויוצא פעם אחת סה"כ $O(n \log k)$.

שאלה 6:

נתונים מערך A של n איברים ומספר z.

א. כתבו שגרה יעילה בזמן המוצאת את זוג המספרים (x, y) ב-A המקיים x + y = z. רשמו זמן ריצה וסיבוכיות המקום של השגרה.

ב. פתרו את הבעיה בסעיף א ' על ידי שימוש במבני נתונים תור עדיפויות . ניתן להשתמש ב $O(1)$ זכרון (נוסף לתורי העדיפויות).

א.פתרון:

```
FindPairs(A, z)
        Heap_sort(A)
        i=0
        j=length[A]-1
        while(i<j )
                if (A[i]+A[j]<z)
                        i = i+1
                else if (A[i]+A[j]>z)
                        j = j-1
                else
                        return < A[i], A[j] >
        return "not found"
```

זמן ריצה O(nlogn + n) = O(nlogn)

ב.פתרון:

```
FindPairs(A, z)
    maxPQ ← makeMaxPQ(A)
    minPQ ← makeMinPQ(A)
    min ← minPQ.ExtractMin()
    max ← maxPQ.ExtractMax()
    while (min != max)
        if (min + max < z)
            min ← minPQ.ExtractMin()
        else
            if (min + max > z)
                max ← maxPQ.ExtractMax()
            else
                return < min, max >
    return "not found"
```

זמן ריצה: $O(2n + 2\log n + n\log n) = O(n\log n)$
זמן ריצה זה מתקבל, אם תור העדיפויות ממומש באמצעות ערימה.

שאלה 7:
נתונות n הרצאות שונות. כל הרצאה מוגדרת על ידי זמן התחלה (S) וזמן סיום שלה (E). $[S_i, E_i]$, $i=1,2,\ldots,n$
כתבו אלגוריתם יעיל הקובע האם ניתן לשבץ את כל ההרצאות הנתונות באותה כיתה .

פתרון 1:

```
notColide(A)
    Build_Heap(A)            //the key for comparison is s_i
    tmp1 ← Extract_Min(A)
    for i ← 2 to n
        tmp2 ← Extract_Min(A)
        if (tmp1.E > tmp2.S)
                return false
        tmp1 ← tmp2
    return true
```

פעולת Extract_Min לוקחת $O(\log n)$ ולכן זמן הריצה של האלגוריתם הוא $O(n+n\log n)=O(n\log n)$.

פתרון 2:

```
Lectures(A)
Heap_sort(A)
for i=0 to i<A.lengthA]-1
        if  A[i].E ≥ A[i+1].S
                return false
return true
```

גם כאן זמן הריצה הוא כזמן המיון של Heap_sort: O(nlogn).

שאלה 8:

נתון עץ חיפוש בינארי T עם n קודקודים וגובה h. הפונקציה הבאה מדפיסה k איברים הקטנים של T.

```
kSmallest(T,  k):
    x ← T.minElement()
    print (key(x))
    i = 1
    while ( i < k )
        x ← T.successor(x)
        print (key(x))
    i ← i+1
```

חשבו את זמן ריצה של kSmallest כפונקציה של h ו-k. רשמו חסם עליון הטוב ביותר.

פתרון:

זמן הריצה של minElement() הוא O(h), וכך הוא גם זמן הריצה של successor, את השגרה הראשונה אנו מפעילים פעם אחת, את השגרה השנייה k-1, כך שהחסם הראשוני שעולה הוא O(k*h).
אולם נשים לב כי מציאת k האיברים הקטנים מתבצעת כמו מעבר inorder על העץ. מכוון שנבקר O(1) פעמים בכל קודקוד ואורך המסלול המקסימלי מקודקוד לעוקב שלו הוא h, זמן ריצה הוא O(h+k).
O(h+k) = O(max(h,k))

שאלה 9:

כתבו אלגוריתם אשר בהינתן עץ חיפוש בינארי הופך אותו לעץ חיפוש בינארי כמעט שלם . מה יהיה זמן ריצה וסיבוכיות מקום של האלגוריתם?

פתרון:

- סרוק את העץ inorder לקבלת הערכים ממויינים
- בנה עץ חיפוש בינארי כמעט שלם T באופן הבא:
  - אתחל n קודקודים עם ערך null (כמס' קודקודי העץ שקיבלנו)
  - קבע את אחד מהם להיות שורש T
  - אכלס את הקודקודים הריקים בעץ, משמאל לימין
    (כמו בערימה, כאשר התא ה-i מצביע ל- 2i ו- 2i+1)
- הכנס את הערכים הממויינים על פי הסדר מהקטן לגדול לעץ , כמו בסריקת inorder
  (ראשית מאכלסים את תת עץ שמאלי, אחריו השורש ואז תת עץ ימני, עבור כל צומת, כשמתחילים עם הכי שמאלית ועמוקה)

זמן ריצה:

סריקת inorder על העץ - O(n)

בניית העץ - O(n)

הכנסה לעץ - O(n)

סה"כ - O(n)

שאלה 10

הציעו מבנה נתונים לטיפול במספרים שלמים אי-שליליים התומך בפעולות הבאות:

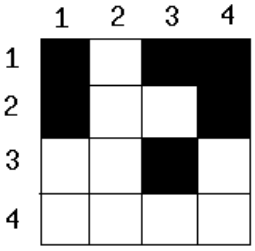
- init(A) – בהינתן מערך A לאתחל את מבנה הנתונים בזמן O(n) כאשר n זה מספר האיברים ב- A.
- insert(x) – להכניס את x למבנה הנתונים. סיבוכיות הזמן – O(log n), כאשר n זה מספר האיברים במבנה הנתונים כרגע.
- Find_till(y) – להדפיס את כל האיברים במבנה הנתונים שקטנים או שווים ל-y. סיבוכיות הזמן – O(k), כאשר k זה מספר איברים כאלה.

שאלה 11

נתונה קבוצה S של $\sqrt{n}$ מספרים שלמים בתחום $[-2^l, 2^l]$ ($l \geq 30$ טבעי). כעת אנו ממלאים מערך בן n איברים, כאשר התוכן של כל אחד מ-n התאים הוא איבר כלשהו מהקבוצה S. הציעו אלגוריתם (דטרמיניסטי או אקראי) למיון המערך העובד בסיבוכיות O(n) בממוצע. כל פעולה על מספר או זוג מספרים בני $l+1$ ביטים דורשת צעד יחיד.

שאלה 12

נתון נייר משבצות בגודל $n \times n$. n ידוע מראש אך אינו קבוע לצורך חישוב הסיבוכיות . הצע מבנה נתונים המאפשר לבצע את הפעולות הבאות בסיבוכיות טובה ככל האפשר . לגבי הפעולות עבורן צוינה דרישה , אין לעבור סיבוכיות זאת.



- Init – אתחל מבנה נתונים , כאשר הדף ריק (כל המשבצות לבנות). הסיבוכיות זמן הנדרשת: O(n^2)
- Color(x,y) – צבע בשחור את המשבצת (x,y).
- Shape(x1, y1, x2, y2) – האם זוג המשבצות (x1,y1) ו- (x2,y2) שייכות לאותה צורה ? שתי משב צות שייכות לאותה צורה אם קיימת ביניהן סדרת משבצות שחורות "נוגעות" (כולל אלכסון). משבצות לבנות אינן שייכות לאף צורה. לדוגמה, (1,3) ו- (3,3) שייכות לאותה צורה, לעומת זאת (1,3) ו- (1,1) לא שייכות לאותה צורה.
- Which(x,y) – החזר רשימה של כל המשבצות השייכות לצורה ש- (x,y) שייכת אליה.
- Num – כמה צורות יש כרגע על הדף? סיבוכיות זמן נדרשת: O(1).

תארו את מבנה המוצע ואת הפעולות הנ "ל. נתחו את הסיבוכיות של המקרה הגרוע ביותר . מהי סיבוכיות המשוערכת של סדרת הפעולות המכילה: אתחול, C פעולות Color, S פעולות Shape ו-N פעולות Num?

שאלה 13

נתונים שני מערכים ממוינים A ו- B שגודלם n ו- m בהתאם

הציעו אלגוריתם אשר מוצא את החציון המשותף של A ו-B. במילים אחרות, האלגוריתם צריך להחזיר מספר x כזה שמספר האיברים (ב-A וב-B יחד) שקטנים או שווים ל-x שווה למספר האיברים שגדולים או שווים ל-x.

סיבוכיות: O(log(n)+log(m))

שאלה 14

יש למיין n רשומות כאשר תחום המפתחות הוא m, ומספר המפתחות השונים הוא p.

יש להתייחס לכל מקרה בנפרד ולנתח סיבוכיות.

א. $m = O(n^k)$, p לא ידוע. יש להתייחס למקרים השונים של k.

ב. $p = O(\log n)$, כאשר m הוא תחום כל המספרים הממשיים.

ג. $p = O(n)$, כאשר m הוא תחום כל המספרים הממשיים.

שאלה 15

יש להציע אלגוריתם יעיל ככל שתוכלו, אשר ימיין מערך שמכיל לכל היותר m איברים אשר אינם במקומם במערך ממוין.

דוגמא: n=11, m= 3.

מערך הקלט (משמאל לימין) 1, 2, 3, 11, 5, 6, 4, 9, 8, 12, 13

מערך הפלט (משמאל לימין) 1, 2, 3, 4, 5, 6, 8, 9, 11, 12, 13

א. יש לנתח את סיבוכיות המקום והזמן כאשר m ידוע וקבוע מראש.

ב. יש לנתח את סיבוכיות המקום והזמן כאשר m לא ידוע וקבוע מראש.

ג. מאיזו נקודה עדיף להשתמש במיון quick-sort רגיל ?

שאלה 16

בעיר חוביזה קיימת מערכת גני ילדים מפותחת וממוחשבת. פרנסי העיר חוביזה, החליטו לפתח מבנה נתונים יעיל אשר ישמור את כל הנתונים עבור הגנים והילדים בעיר.
מבנה הנתונים יאפשר ביצוע הפעולות הבאות:

1. INIT() – אתחל את מבנה הנתונים ללא גנים וילדים.
2. ARRIVE(CHILD, LEVEL) – הוסיפו את CHILD כילד חדש בעיר למאגר הילדים ברמה LEVEL שעדיין לא בגן. רמה יכולה להיות: גנון, טרום חובה, חובה.
3. ESTABLISH(C, CHILD) הוסף גן חדש C לקבוצת הגנים, והוסף את CHILD לגן. אם CHILD נמצא בגן אחר הוא עוזב את הגן האחר.
4. THROW(CHILD)- CHILD עוזב את הגן, ועובר למאגר הילדים שלא הולכים לגן (בנתיים).
5. LEAVE(CHILD) – CHILD עוזב את הגן והעיר, ועובר לעיר אחרת, ויוצא ממאגר הנתונים.
6. CLOSE(C) גן C נסגר. (למרות שלא כל כך נעים לראות את זה), וכל הילדים חוזרים למאגר הילדים שלא הולכים לגן.
7. JOIN(C, CHILD) – CHILD מצטרף לגן C. אם CHILD שייך לגן אחר, הוא עוזב את הגן האחר.
8. MERGE(C, D, E) – גן C וגן D מתאחדים, לגן חדש בשם E.

מבנה הנתונים יאפשר את ביצוע השאילתות הבאות:

א. WHERE(CHILD) – באיזה גן משחק CHILD?
ב. OLDER(C) – מי הילד הכי ותיק בגן C? (עבור השאלה, נניח שכל המידע המוכנס למבנה הנתונים באותו סדר שבו הוא קורה. אם הוספנו ילד 1P לפני ילד 2P למבנה הנתונים, סימן שהילד 1P הצטרף לגן לפני הילד 2P). אם גן C התאחד עם גן D, הילד הכי ותיק בגן המאוחד E , יהיה הוותיק ביותר מבין הגן C והגן D.
ג. CHILDREN(C) – החזר את רשימת הילדים בגן C.
ד. EXTERNAL ( ) – אחוז הילדים שלא שייכים לשום גן בעיר חוביזה.

1. מה תהיה סיבוכיות המקום והזמן במקרה הגרוע, עבור כל פעולה ושאילתא.
2. מהי הסיבוכיות המשוערכת, במקרה הכי גרוע.

רמז: פיתרון יעיל יכול להשתמש בערימה, קבוצות זרות ורשימה ממוינת.

שאלה 17:

האלגוריתם הבא מוצע כתחליף לאלגוריתם של BuildHeap בכדי לבנות ערימת מקסימום על מערך בן n איברים. האלגוריתם עובד מלמעלה למטה (ז"א מהאיבר הראשון לכיוון האחרון), ובכל פעם מפעפע את הערך כלפי מעלה. ז"א אם הערך הנוכחי גדול מאביו הרי שיתחלף איתו וכך הלאה.

א. האם האלגוריתם נכון? נמקו.
ב. מה תהיה סיבוכיות האלגוריתם? נמקו.

שאלה 18:

S היא קבוצה של n נקודות במישור.
כל הנקודות ב-S הן מהצורה (a,b) כאשר a ו b הם מספרים שלמים ומתקיים $-n \le a \le n$ ו-
$-k \le b \le k$ עבור קבוע k כלשהו.
תאר אלגוריתם למיון n הנקודות ב-S לפי ריבוע המרחק שלהם מהראשית (ריבוע המרחק של (a,b) מהראשית הוא a^2+b^2). על האלגוריתם שתיארת לרוץ בזמן $O(n)$ במקרה הגרוע.

הערה: ניתן למיין את הערכים גם לפי המרחק מהראשית ($\left|(a,b)\right| = \sqrt{a^2+b^2}$).

שאלה 19:

תארו מבנה נתונים השומר קבוצה S של איברים, כאשר לכל איבר יש מפתח, ומאפשר את הפעולות הבאות:

- Insert(x) - הוסף את האיבר x ל-S.
- Find13() - מצא את האיבר ב-S שנמצא במקום $\lfloor n/3 \rfloor$ בסדר הממוין של S לפי המפתחות, כאשר n הוא מספר אברי S ברגע זה.
  (כלומר, יש למצוא את האיבר x עבורו יש $\lfloor n/3 \rfloor - 1$ איברים ב-S עם מפתח קטן מהמפתח של x, ויש $n - \lfloor n/3 \rfloor$ איברים עם מפתח גדול מהמפתח של x).
- Find23() - מצא את האיבר ב-S שנמצא במקום $\lfloor 2n/3 \rfloor$ בסדר הממוין של S לפי המפתחות.

על פעולת ה-Insert להתבצע בזמן $O(\log n)$ ועל פעולות Find13 ו-Find23 להתבצע בזמן קבוע.

שאלה 20:

בבית ספר באזור רצו לבדוק אם כל הילדים בגדוד ה ' של הצופים הם מכיתה ה'1 בלבד בשכבה. כתבו פסאודו-קוד של אלגוריתם יעיל הבודק האם גדוד הצופים הינו תת-קבוצה של כיתה ה'1.
מהו זמן ריצה הממוצע של האלגוריתם?
מהו זמן ריצה הגרוע של האלגוריתם?

פרק חמישי – מחרוזות, גרפים וכל מיני..

שאלה 1:

נתונה מחרוזת s מעל א"ב קבוע כלשהו.

יש לתאר אלגוריתם המדפיס מחרוזת t המקיימת את התכונה הבאה:

t היא תת-מחרוזת הארוכה ביותר של s שמופיעה ב-s יותר מפעם אחת.מבין כל תת-המחרוזות שמקיימות את התכונה הנ"ל,תודפס המחרוזת הקטנה ביותר לקסיקוגרפית.אם אין ל-s תת-מחרוזת שמופיעה ב-s יותר מפעם אחת,האלגוריתם לא מדפיס כלום.

על האלגוריתם לרוץ בזמן $O(|s|)$, כאשר $|s|$ - הוא אורך המחרוזת s.

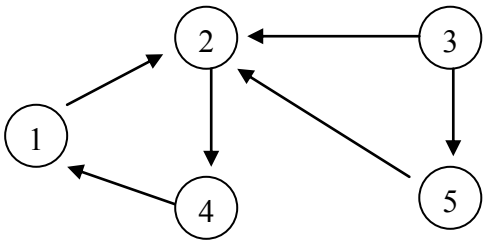
לדוגמא,עבור המחרוזת s= ababcabcc, האלגוריתם ידפיס abc .

עבור המחרוזת s= ababa, האלגוריתם ידפיס aba.

שאלה 2:

נתון, גרף מכוון עם קודקודים ממוספרים מ 1 עד N . הפלט יהיה, מערך שבו האינדקס הוא מספר הצומת ולכל צומת בתא יהיה רשום מספר הצומת המקסימאלי שהיא יכולה להגיע אליה (לא כולל הצומת עצמה). על הפיתרון להיות יעיל בסיבוכיות זמן ומקום.

לדוגמא, עבור הגרף הבא (N = 5) :



מערך התוצאה יהיה:

1	2	3	4	5
4	4	5	2	4

שאלה 3:

ערימת מינימום בדרגה t הינה ערימה שמספר הילדים של כל קודקוד לכל היותר t (t אינו קבוע). אם t = 2 זוהי ערימת מינימום כפי שלמדנו בהרצאה.

ניתן להניח שהערימה תקינה ואין צורך לערוך בדיקות.

א. בהינתן אינדקס i של איבר בערימה, רשמו פונקציה לחישוב אינדקס הילד ה –k ו ההורה. נמקו!!

```
calcIndex( i, t) :  i * t + k + 1
calcIndex_father( i) :  (i – 1) / t;
```

ב. רשמו אלגוריתם (פסאודו-קוד) של פעולת insert(x) ו- del_min( ).

```
Insert(x) :
A[n] <- x;
n++;
i <- n -1;
while( i > 0) {
        if( A[i] < A[calcIndex_father(i)] {
                swap( A[i], A[calcIndex_father(i)] );
                i <- calcIndex_father(i);
                }
        else
                i <- 0;
```

```
        }

    Del_min() :
    x = A[0];
    A[0] <- A[n-1];
    n--;
    i <- 0
    l <- 1;
    r <- min( t, n-1);
    while( l <= n-1) {
            m <-index of minimal{ A[l], ...,A[r]}
            if( A[i] , A[m])
                    return x;
            swap( A[i], A[m]);
            i <- m;
            l <- calcIndex( i, 0);
            r <- min( calcIndex( i, t-1), n-1);
            }
    return x;
```

ג. מה יהיו זמני הריצה של כל אחד מהאלגוריתמים במונחים של t ו – n. נמקו!!

Insert(x) : $\log_t n$

Del_min: $\log_t n \cdot t$

ד. נתון כי זמני הריצה הם $\log t \cdot \log n$. הסבירו את השינויים באלגוריתמים. תוספת מקום מותרת $O(n)$.

בכל קודקוד נחזיק ערימת מינ . קטנה של ערכי הבנים . בכל איבר בערימה הקטנה יהיה רשום אינדקס הילד המתאים לו . בכל ילד נחזיק מצביע לאינדקס שלו בערימה הקטנה של אביו . בכל עדכון של איבר בערימה הכללית, נעדכן את ערכו בערימה הקטנה של האב ע"י מחיקה באמצעות האינדקס והחזרה של הערך המעודכן.
העדכון מתבצע עבור שני איברים בכל swap.
זמן כללי לעדכון ערך $O(\log t)$.
עבור ההכנסה:
נבצע הוספת איברים כמו בסעיף ב תוך כדי עדכון העירמות הקטנות . עדכון כל ערימה קטנה הוא לוגריתמי ב – t ולכן זמן הריצה הוא $O(\log_t n \cdot \log t)$.
עבור הוצאת המינימלי:
נבצע הוצאת מינימום כמו בסעיף ב תוך כדי עדכון הערימות הקטנות מציאת המינ . תיקח זמן קבוע. עדכון כל ערימה קטנה הוא לוגריתמי ב- t ולכן זמן הריצה הוא כאמור.

שאלה 4:

נתונים n מספרים שלמים בתחום 1..k. אחרי זמן עיבוד של $O(n + k)$ יש להיות מוכנים לענות לשאלה: כמה מספרים יש בתחום a..b בזמן קבוע. (עבור כל תחום של a..b).

```
Range(A, k, a, b)
  Preprocess :
    for i ← 1 to k
      C[i] ← = 0
    for j ← 1 to n
      C[A[j]] ← C[A[j]] + 1  // C[i] = the number of appearances of i in A.
    for i ← 2 to k
      C[i] ← C[i] + C[i-1]   // C[i] = the number of elements in A that are ≤ i
  return (C[b] - C[a-1])
```

שאלה 5:

נתונים n מספרים בתחום a .. a+e . יש למיין את המספרים הנ"ל בזמן ליניארי, בהנחה שהמספרים מפוזרים אחיד על התחום הנ"ל. הראו שבמקרה הגרוע זמן המיון יהיה $O(n \log n)$.

Solution 1:

Use bucket sort over the range [x,x+d] with the following changes:

1. The elements in each bucket are stored in a RB tree (instead of a linked list)
2. In the last stage, concatenate all the *inorder* visits of all the buckets one after another.

Time Complexity:

Let n_i be the number of elements in the tree in bucket i.

1. Inserting the n elements into the buckets takes $O(n_1\log n_1 + n_2\log n_2 + ... + n_n\log n_n)$
   - ♦ When the keys are uniformly distributed $n_i = O(1)$ for every i , hence $O(n_1\log n_1 + n_2\log n_2 + ... + n_n\log n_n) \le c(n_1 + n_2 + ... + n_n) = cn$, where c is a constant.
   - ♦ In the worst distribution cases: $O(n_1\log n_1 + n_2\log n_2 + ... + n_n\log n_n) \le O(n_1\log n + n_2\log n + ... + n_n\log n) = O((n_1 + n_2 + ... + n_n)(\log n)) = O(n\log n)$
2. *Inorder* traversals of all buckets takes $O(n_1 + n_2 + ... + n_n) = O(n)$
3. Concatenation of all *inorder* traversal lists takes $O(n)$

The algorithm runs in $O(n)$ time for uniformly distributed keys and $O(n\log n)$ in the worst distribution case.

Solution 2:

Execute in parallel the following two algorithms:

1. original bucket sort
2. Any sort algorithm that takes $O(n\log n)$

Stop when one of the algorithms has stopped and return the sorted elements

שאלה 6:

בשאלה זו אנו עוסקים בקבוצות זרות הממומשות ע"י עצים הפוכים עם כיווץ מסלולים. הראנו בהרצאה כי ביצוע n פעולות (union, find) לוקח זמן ארוך יותר מליניארי במספר הפעולות.

א. הוכיחו כי ביצוע n הפעולות בסדר הבא: קודם כל פעולות ה find ואז כל פעולות ה – union ייקחו זמן לינארי.

זה קל, כי אם קודם עושים את כל הfind זה ייקח זמן קבוע, כי כולם צמתים בודדים, ואח"כ את כל ה-union ממילא ייקח זמן קבוע.

ב. הוכיחו כי ביצוע n הפעולות בסדר הבא: קודם כל פעולות ה union ואז כל פעולות ה – find ייקחו זמן לינארי.

כל פעולת union לוקחת זמן קבוע , לכן m פעולות כאילו יקחו זמן $O(m)$.

אח"כ בפעולת ה- find נעשה פעם אחת על כל צומת, ז"א אם הגובה יהיה לוגריתמי אזי אחרי הכיווץ נקבל את כל האיברים הללו משורשרים מתחת לשורש ישירות.

ז"א אם נפנה לצומת בעומק d אזי הזמן יהיה $O(d)$, אבל אז ל d-1 צמתים מעליו הזמן יהיה $O(1)$ כייון שהם יתלו מיידית מתחת לשורש. לכן בכלליות כל d צמתים "ישלמו" $O(d)$ פעולות.

שאלה 7:

נתונה קבוצה S של m מחרוזות מעל ה א "ב Σ= a,b. דהיינו : $S = \{ S_1, S_2, .. , S_m \}$ כאשר סכום אורכי המחרוזות שווה ל – n . יש לתאר אלגוריתם העובד בזמן לינארי באורך המחרוזות , המחזיר את המחרוזות שמופיעות כתת מחרוזות בתוך מחרוזות אחרות.
ניתן להשתמש באלגוריתמים שנלמדו בכיתה, אך יש לתאר במדויק את הקלט, סיבוכיות הזמן, והפלט.
דוגמא: עבור הקלט $S = \{$ aabaa, abaa, aba, baa$\}$. הפלט יהיה aba, baa .
משרשרים את כל המחרוזות , ומכניסים אותם לעץ סיומות ע "פ האלגוריתם של הקופסא השחורה כאשר מקצצים את הסיומות שלא שייכות (כמו שעשינו בהרצאה). זמן לינארי בסכום אורכי המחרוזות.
אח"כ עוברים עם מחרוזת S_i , מחפשים אותה בעץ , אם מצאנו אותה , והסיומת מכילה $\$_j$ כאשר $j \neq i$ אזי המחרוזת S_i היא תת מחרוזת של המחרוזת S_j . זמן לינארי = סכום אורכי המחרוזות.

שאלה 8:

נתון הגרף לא מכוון ממושקל. יש להפעיל את האלגוריתם של Prim למציאת עץ פורש מינימלי, ע"י שימוש ברשימת סמיכויות. שרטטו את רשימת הסמיכויות עבור הגרף ואת הגרף אחרי כל שלב .
מה תהיה סיבוכיות האלגוריתם ?

	A	B	C	D	E	F	G
A	0	7	8	9	0	0	0
B	7	0	0	5	1	0	0
C	8	0	0	4	0	6	0
D	9	5	4	0	2	3	11
E	0	1	0	2	0	0	12
F	0	0	6	3	0	0	10
G	0	0	0	11	12	10	0

שאלה 9:

מטריצת הקשירויות של גרף מכוון G=(V,E) היא מטריצה B שמימדיה |E|*|V| וערכיה מוגדרים באופן הבא:

B(i,j) = -1 אם הקשת j יוצאת מהצומת i.
B(i,j) = 1 אם הקשת j נכנסת לצומת i.
B(i,j) = 0 אם הקשת j אינה נוגעת בצומת i.

יש לתאר ולהסביר מה מייצגים איברי מטריצת המכפלה BB^T כאשר B^T היא המטריצה ה- transpose של B.

הערות:
את הפתרון יש לחלק לשני מקרים: אברי האלכסון ושאר אברי המטריצה.
שימו לב! קשתות (i,j) ו-(j,i) הינן שתי קשתות שונות.

שאלה 10:

הגדרה : *k-עץ* הוא גרף קשיר אשר ניתן להוריד ממנו k קשתות בדיוק כך שיתקבל עץ.

א.יש לכתוב אלגוריתם שמקבל כקלט גרף קשיר לא מכוון G=(V,E) עם פונקצית משקל על הקשתות,ומוצא בו k-עץ פורש מינימאלי.
(כלומר,תת-גרף שהוא k-עץ הפורש את כל צמתי הגרף, ושמשקלו מינימאלי ביחס לכל k העצים הפורשים את הגרף G).
מהי סיבוכיות הזמן של האלגוריתם?

ב.הוכח או הפרך: האם בגרף ,שהוא k-עץ, יש בהכרח בדיוק k מעגלים פשוטים?
(מעגל פשוט – מסלול מעגלי שכל הקדקדים לאורך המסלול שונים זה מזה).

שאלה 11:
הוכח או הפרך : מיון QuickSort הינו מיון יציב.

שאלה 12:
נתונות k מחרוזות המייצגות גנום מעל הא"ב $\{A,T,C,G\}$. אורך כל המחרוזת - לכל היותר n.
(לדוגמא, $AATCG$ - הינה מחרוזת באורך 5.)
רישא של מחרוזת S הינה תת-מחרוזת המתחילה בתו הראשון של S.

א.יש לתאר מבנה נתונים ואלגוריתם המקבל כקלט את המחרוזות $S_1,.....,S_k$ ומספר שלם m, ומוציא כפלט את מספר כל הרישות המשותפות לכל הפחות ל –m מחרוזות בסיבוכיות זמן $O(n*k)$.

ב.יש לתאר מבנה נתונים ואלגוריתם המקבל כקלט את המחרוזות $S_1,.....,S_k$ ומספר שלם m , ומוציא כפלט את מספר כל תתי המחרוזות המשותפות לכל הפחות ל - m מחרוזות בסיבוכיות זמן $O(n^2*k)$.

שאלה 13:
נתון גרף מכוון $G(V,E)$. ידוע שהגרף פשוט: אין בו לולאות עצמיות (אין קשת מצומת לעצמו), ואין יותר מקשת אחת מצומת כלשהו u, אל צומת אחר v.
נגדיר "בור" כצומת אשר יש אליו קשתות נכנסות אך אין מימנו יוצאות. "בור שלם" הנו צומת "בור" שמספר הקשתות הנכנסות אליו הנו n-1 ($n=|V|$).

א. תאר כיצד נראה גרף עם צמתים שהם "בור שלם" תחת ייצוג ע"י רשימת סמיכויות.
ב. תאר את אותו הגרף ע"י מטריצת סמיכויות.
ג. כיצד ניתן למצוא אם בגרף יש צמתים שהם "בור שלם" תחת ייצוג ע"י מטריצת סמיכויות ובסיבוכיות זמן $O(n)$?
ד. האם ניתן למצוא זאת גם תחת ייצוג ע"י רשימת סמיכויות ובאותה סיבוכיות זמן? אם כן, כיצד תעשו זאת, ואם לא – מדוע, ומה כן ניתן להסיק לגבי הימצאות "בור שלם" בסיבוכיות זו?

1. השאלה הבאה עוסקת בפונקציות ערבול (Hash functions).
נתונה פונקצית הערבול $h(k) = k \bmod m$. עליכם להכניס את המפתחות הבאים:
10,22,31,4,15,28,17,88,59 (משמאל לימין) לטבלת ערבול בגודל *m=11* עם הפונקציה הנתונה.

א. באמצעות מיעון פתוח (Open Addressing).
ב. באמצעות שרשראות.
ג. בעזרת ערבול כפול (rehashing עם שתי פונקציות) עם פונקצית ערבול שנייה
$h_2(k) = 1 + (k \bmod (m-1))$.

כעת בצעו הוצאה של 15,59,88 (משמאל לימין) עבור שלושת המקרים.

שאלה 14:
הציעו שיטת אחסון לאלמנטים המופיעים בשרשראות ה- Hash בתוך הטבלה עצמה (שיטת chain hashing). ניתן להניח שכל כניסה בטבלה יכולה להכיל דגל ומספר מצביעים.
האם ניתן להשיג סיבוכיות של $O(1)$ באמצעות רשימות חד כיווניות בלבד?
יש לתאר באופן גרפי את הפיתרון המוצע.

שאלה 15:
בהינתן k רשימות ממוינות, הציעו דרך למזג אותן בסיבוכיות $O(n \cdot \log k)$, כאשר n הוא מספר האלמנטים הכולל במערכת.

שאלה 16:
יש להציע מבנה נתונים בו מוחזקות n מחרוזות בינאריות, כל אחת באורך m. למחרוזת s נסמן ב- w(s) את מספר הפעמים ש – s הוכנסה למבנה.
ניתן לבצע את הפעולות הבאות על מבנה הנתונים:

א. insert(s) – הכניסו מחרוזת s למבנה. אם s כבר מופיעה יש להוסיף 1 ל – w(s). הסיבוכיות הנדרשת הינה $O(m)$.
ב. search(s) – חיפוש s במבנה. הסיבוכיות הנדרשת $O(m)$.
ג. max(r, t) – הקלט הוא שתי מחרוזות r ו – t. בין כל המחרוזות שנמצאות במבנה ומקומן בסדר הלקסיקוגרפי הוא בין r ל – t יש למצוא את s כך ש – w(s) מקסימלי – הסיבוכיות הנדרשת $O(m)$.