**מדינת ישראל**
משרד החינוך

סוג הבחינה: גמר לבתי־ספר לטכנאים ולהנדסאים
מועד הבחינה: אביב תשע"ו, 2016
סמל השאלון: 714911
נספחים: א. דף תשובות
ב. מילון מונחים

# מבני נתונים ויעילות אלגוריתמים

## הוראות לנבחן

**א. משך הבחינה:** ארבע שעות.

**ב. מבנה השאלון ומפתח ההערכה:** בשאלון זה שני פרקים.

| | | |
|---|---|---|
| פרק ראשון | 65 | נקודות |
| פרק שני | 35 | נקודות |
| סה"כ | 100 | נקודות |

**ג. חומר עזר מותר לשימוש:** כל חומר עזר כתוב בכתב־יד או מודפס על נייר.

**ד. הוראות מיוחדות:**

1. את התשובות לשאלות 3 ו־4 יש לרשום **אך ורק** על גבי דף התשובות שבנספח א'.
את התשובות לשאלה 1 ולשאלה 2 יש לרשום במחברת הבחינה.

2. לנוחותך, לשאלון זה מצורף מילון מונחים בשפות עברית, ערבית, אנגלית ורוסית. תוכל להיעזר בו בעת הצורך.

**הוראות למשגיח:**
בתום הבחינה יש לוודא שהנבחנים הדביקו את מדבקת הנבחן שלהם
במקום המיועד לכך בדף התשובות שבנספח א' וצירפו אותו למחברת הבחינה.

**בשאלון זה 47 עמודים ו־4 עמודי נספחים.**

ההנחיות בשאלון זה מנוסחות בלשון זכר,
אך מכוונות הן לנבחנות והן לנבחנים.

**בהצלחה!**

המשך מעבר לדף ◀

# השאלות

## פרק ראשון (65 נקודות)

**ענה על שתי השאלות 1–2 – שאלות חובה.**

### שאלה 1 – שאלת חובה (40 נקודות)

במדינת ישראל כל אזרח ישראלי בן 18 לפחות רשאי להצביע בעבור רשימה (מפלגה) אחת מבין המפלגות המתמודדות לכנסת. הצבעת הבוחר נעשית על־ידי הטלת פתק בקלפי.

על סמך הצבעות הבוחרים מתבצעת **חלוקת המנדטים** (מספר המושבים בכנסת שתקבל כל מפלגה). מספר המנדטים במדינת ישראל הוא 120 .

להלן תהליך **חלוקת המנדטים**.

(שים לב: התהליך המתואר להלן מספק את הנתונים הדרושים לצורך פתרון השאלה גם אם אינו משקף את המציאות כולה.)

עבור כל מפלגה מונים את מספר הקולות שקיבלה, ומחשבים את אחוז הקולות מכלל הקולות שנמנו. הנח כי הקולות שנמנו הם קולות **כשרים בלבד**.

מפלגה תיוצג בכנסת אם אחוז הקולות שקיבלה גבוה מאחוז החסימה, כאשר אחוז החסימה בישראל עומד על 3.25% מכלל הקולות.

**דוגמה:**

בבחירות האחרונות לכנסת נמנו 4,210,884 קולות.
בטבלה 1 שלהלן מוצגים נתונים עבור **חמש מפלגות** שנדגמו באופן אקראי.

בעבור כל מפלגה שנדגמה, הטבלה תכיל את מספר הקולות שקיבלה, את אחוז הקולות הללו מכלל הקולות שנמנו, וכן אם היא עברה את אחוז החסימה.
(אינך נדרש לבדוק את תוצאות החישוב המוצגות בטבלה זו.)

המשך בעמוד 3

| שם המפלגה | מספר הקולות שקיבלה | אחוז הקולות שקיבלה המפלגה מכלל הקולות | האם הרשימה עברה את אחוז החסימה? |
|---|---|---|---|
| הליכוד | 985,408 | 23.40% | כן |
| המחנה הציוני | 786,313 | 18.67% | כן |
| הרשימה המשותפת | 446,583 | 10.61% | כן |
| יחד | 125,158 | 2.97% | לא |
| עלה ירוק | 47,180 | 1.12% | לא |

**טבלה 1**

מפלגות שלא עברו את אחוז החסימה אינן מיוצגות בכנסת. נוסף על כך, הקולות שקיבלו המפלגות הללו הולכים לאיבוד ונגרעים מסך כל הקולות שנמנו.

ניתן להסיק מן הטבלה כי קולותיהן של המפלגות עלה ירוק ו־ יחד ילכו לאיבוד. אי לכך, סך כל הקולות שקיבלו כל המפלגות בישראל אשר עברו את אחוז החסימה יהיה קטן בהכרח מסך כל הקולות שנמנו בתחילה.

בשלב זה נותרות רק המפלגות שעברו את אחוז החסימה, וחלוקת המנדטים (המושבים בכנסת) נעשית ביניהן בלבד, בשני הצעדים האלה:

**צעד 1: חישוב מספר המנדטים למפלגה ה־i**

נסמן ב־$m_i$ את מספר המנדטים של המפלגה ה־i , והוא יחושב באמצעות הנוסחה שלהלן:

$$m_i = \left\lfloor \frac{x_i}{\sum_{j=1}^{s} x_j} \times 120 \right\rfloor$$

כאשר $x_i$ הוא מספר הקולות שקיבלה מפלגה זו, s הוא מספר המפלגות שעברו את אחוז החסימה, והביטוי שבמכנה של השבר הנו סכום הקולות של כל המפלגות שעברו את אחוז החסימה.

שים לב: הסימון $\lfloor x \rfloor$ מציין את הערך המעוגל כלפי מטה של x . לדוגמה:

$$\lfloor 3.7 \rfloor = 3 \qquad \lfloor 3.1 \rfloor = 3 \qquad \lfloor 3 \rfloor = 3$$

המשך בעמוד 4

**צעד 2 : חלוקת המנדטים העודפים בין המפלגות**

אם נותרו מנדטים, כלומר, סך המנדטים שקיבלו כל המפלגות שעברו את אחוז החסימה קטן מ־120, אז יש להמשיך לחלקם בין המפלגות עד שיחולקו כל 120 המנדטים.

לצורך החלוקה יש להגדיר לכל מפלגה **מודד**:

נסמן ב־ level(i) את **המודד** של המפלגה ה־i , והוא יחושב באמצעות הנוסחה שלהלן:

$$\text{level}(i) = \frac{x_i}{m_i + 1}$$

כאשר $x_i$ הוא מספר הקולות שקיבלה המפלגה, ו־$m_i$ הוא מספר המנדטים שקיבלה (חושב בצעד 1).

בהמשך לנתונים שבטבלה 1 , לפניך טבלה 2 אשר כוללת את מספר המנדטים שקיבלו שלוש המפלגות שעברו את אחוז החסימה ואת המודד שלהן.

(אינך נדרש לבדוק את תוצאות החישוב המובאות בטבלה זו).

| שם המפלגה ה־i | מספר הקולות $x_i$ | מספר המנדטים $m_i$ | המודד level(i) |
|---|---|---|---|
| הליכוד | 985,408 | 29 | 32,846.93 |
| המחנה הציוני | 786,313 | 23 | 32,763.04 |
| הרשימה המשותפת | 446,583 | 13 | 31,898.78 |

**טבלה 2**

**עתה נתאר את חלוקת המנדטים העודפים:**

כל עוד קיים לפחות מנדט עודף אחד, בצע:

- בחר את המפלגה עם **המודד** הגבוה ביותר (תסומן ב־A).
- הוסף מנדט אחד למפלגה A .
- חשב מחדש את המודד של המפלגה A (עם המנדט הנוסף שהמפלגה קיבלה).
- הקטן ב־1 את מספר המנדטים העודפים.

המשך בעמוד 5

**דוגמה:**

נניח שנותרו תשעה מנדטים.

לפי טבלה 2 , מפלגת הליכוד היא בעלת המודד הגבוה ביותר ולכן היא מקבלת מנדט נוסף.
עתה יהיו לליכוד 30 מנדטים, המודד החדש של הליכוד יהיה: $\frac{985,408}{31} = 31,787.35$ ,
ומספר המנדטים העודפים יהיה כעת 8 .

תהליך חלוקת המנדטים העודפים חוזר על עצמו עד שכל שמונת המנדטים העודפים מחולקים בין המפלגות.

להלן סיכום של **תהליך חלוקת המנדטים** לכנסת:

1. קליטה ראשונית של הקולות הכשרים ביום הבחירות.
2. חישוב אחוז הקולות שקיבלה מפלגה מכלל הקולות שנמנו.
3. קביעת המפלגות שעברו את אחוז החסימה והמפלגות שלא עברו את אחוז החסימה.
4. חישוב מספר המנדטים לכל מפלגה שעברה את אחוז החסימה.
5. חלוקת המנדטים העודפים (אם ישנם) בין המפלגות.

הנהלת ועדת הבחירות המרכזית לכנסת הקימה מערכת ממוחשבת אשר תומכת בחלוקת המנדטים בין M המפלגות שהציגו את מועמדותן לכנסת, כאשר M הוא משתנה.

המפלגות ממוספרות באופן אקראי מ־0 עד M - 1 (כולל).

מעצבי המערכת הממוחשבת החליטו על מבנה נתונים שבו ישוכנו נתוני הבחירות לצורך חלוקת המנדטים. לפניך תיאור של **מבנה הנתונים** הזה:

נחזיק מבנה (רשומה) שנכנה אותו בשם "**המבנה הראשי**" המכיל את השדות האלה:

**שדה 1 :** **votesTree** – מצביע לשורשו של **עץ הקולות** שהוא עץ חיפוש בינארי מאוזן (עץ AVL).
עץ זה מכיל מידע על מספר הקולות שקיבלה כל אחת מן המפלגות, כאשר כל צומת בעץ זה מייצג את מספר הקולות שקיבלה מפלגה יחידה.
מפתח החיפוש בעץ זה הוא מספר הקולות.
**שים לב:** בעץ זה משוכנים גם מספר הקולות בעבור מפלגות שלא עברו את אחוז החסימה.

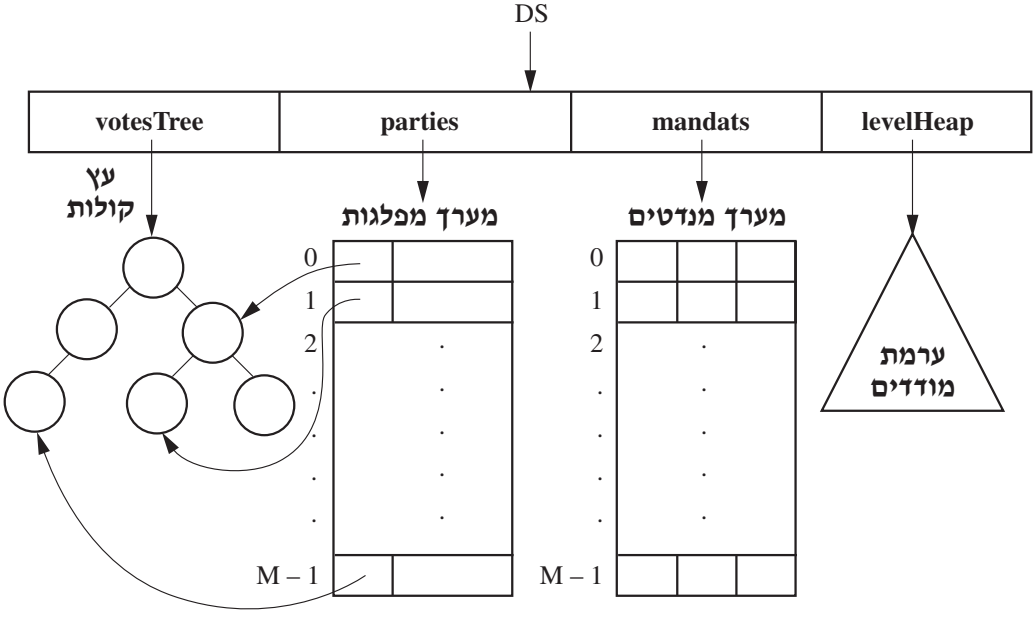
המשך בעמוד 6

**שדה 2 :** **parties** - **מערך מפלגות** בגודל M , כאשר התא ה־i במערך זה לכל $0 \le i < M$ מכיל את פרטי המפלגה ה־i , ומצביע לצומת **בעץ קולות** המכיל את מספר הקולות שקיבלה מפלגה זו.

**שדה 3 :** **mandats** – **מערך מנדטים** בגודל M , כאשר כל תא במערך זה מייצג מפלגה, ומכיל, בין היתר, את מספר המנדטים של מפלגה מסוימת.
מערך זה מכיל מידע בעבור **כל** המפלגות שהתמודדו לכנסת, והוא ממוין בסדר יורד, לפי מספר המנדטים.
**שים לב**: מספר המנדטים של מפלגה שלא עברה את אחוז החסימה יהיה אפס.

**שדה 4 :** **levelHeap** – מצביע לשורשו של עץ שהנו **ערמת מקסימום**. כל איבר בערמה זו מייצג מפלגה, ומכיל, בין היתר, את המודד שלה ואת המיקום (index) של המפלגה הזו במערך **mandats** .
בערמה זו **העדיפות** נקבעת על־פי גובה המודד.
ערמה זו מכונה בשם **ערמת המודדים** ומיוצגת באמצעות **מערך דינמי**.
**שים לב**: ערמת המודדים מכילה מידע **רק** עבור המפלגות שעברו את אחוז החסימה.

**איור א'** שלהלן מציג תיאור סכמתי של **"המבנה הראשי"**:




**איור א' לשאלה 1**

המשך בעמוד 7

"**המבנה הראשי**" מאגד את כל מבני הנתונים שבהם נאחסן את כל נתוני המערכת הממוחשבת.

להלן הגדרת הקבועים בשפת C :

```
#define M 12                    // מספר המפלגות שהציגו מועמדות לכנסת
#define NUM_OF_DELEGATES 120;   // מספר המנדטים בכנסת
#define BLOCK_PERCENT 3.25;     // אחוז החסימה
```

ולהלן הגדרת **המבנה הראשי** בשפת C :

```
typedef struct headderType          // טיפוס המבנה הראשי
{
        voteptr votesTree;          // מצביע לשורשו של עץ הקולות
        party parties[M];           // מערך המפלגות
        mandat mandats[M];          // מערך המנדטים
        levelptr levelHeap;         // מצביע לשורשה של ערמת המודדים
} headder, *headptr;
```

עתה נפרט את מבנה הנתונים בעבור **שדה 1** של "**המבנה הראשי" – עץ הקולות**.

להלן מבנה של צומת בעץ הקולות בשפת C :

```
typedef struct voteType
{
        int partyNum;               // מספר המפלגה
        long voteNum;               // מספר הקולות שקיבלה המפלגה
        struct voteType *vleft;     // מצביע לתת־העץ השמאלי
        struct voteType *vright;    // מצביע לתת־העץ הימני
} vote,*voteptr;
```

המשך בעמוד 8

עתה נפרט את מבנה הנתונים בעבור **שדה 2** של **"המבנה הראשי" – מערך המפלגות**.

להלן מבנה של תא במערך המפלגות בשפת C :

```
typedef struct partyType      //  טיפוס מפלגה
{
        char partyName[15];   //  שם המפלגה
        voteptr  pvt;         //  מצביע לצומת בעץ הקולות שמכיל את מספר הקולות
                              //  שקיבלה מפלגה זו
} party,*partyptr;
```

עתה נפרט את מבנה הנתונים בעבור **שדה 3** של **"המבנה הראשי" – מערך המנדטים**.

להלן מבנה של תא במערך המנדטים בשפת C :

```
typedef struct mandatType
{
        int partNum;          //  מספר המפלגה
        float percent;        //  אחוז הקולות שקיבלה מפלגה זו מכלל הקולות
        int numOfMandats;     //  מספר המנדטים
} mandat,*mandatptr;
```

עתה נפרט את מבנה הנתונים בעבור **שדה 4** של **"המבנה הראשי" – ערמת המודדים**.

להלן מבנה של איבר בערמת המודדים בשפת C :

```
typedef struct levelType  //  טיפוס איבר בערמה
{
        int index;        //  מיקומה של המפלגה במערך המנדטים
        float level;      //  המודד של מפלגה זו
} levelRec,*levelptr;
```

המשך בעמוד 9

**נתונה ספריית פונקציות המכילה, בין היתר, את הפונקציות האלה:**

| | |
|---|---|
| voteptr **insertVotesTree**(voteptr *ct, voteptr p) | פונקציה זו מקבלת שני פרמטרים:<br>**ct** – מצביע לשורשו של **עץ הקולות**,<br>**p** – מצביע **לצומת** (בודד), המייצג מפלגה מסוימת.<br>הפונקציה מוסיפה לעץ הקולות את הצומת שעליו מצביע p , ומחזירה מצביע לצומת שנוסף זה עתה.<br>**הערה:** לאחר הוספת הצומת, עץ הקולות יישאר עץ AVL . |
| void **deleteFromVotesTree**(voteptr *t, voteptr p) | פונקציה זו מקבלת שני פרמטרים:<br>**t** – מצביע לשורשו של **עץ הקולות**,<br>**p** – מצביע לצומת שקיים בעץ הקולות.<br>הפונקציה מוחקת מעץ הקולות את הצומת שעליו מצביע p .<br>**הערה:** לאחר מחיקת הצומת, עץ הקולות יישאר עץ AVL . |
| void **percolateUP**(levelptr t ,int index) | פונקציה זו מקבלת שני פרמטרים:<br>**t** – מצביע לשורשו של עץ המיוצג באמצעות מערך דינמי,<br>**Index** – מיקומו של איבר מסוים בעץ.<br>כל תת־עץ הנמצא בעץ שלשורשו מצביע t הוא ערמה, פרט לתת־העץ ששורשו נמצא במקום ה־index .<br>הפונקציה "מפעפעת" את האיבר שבמקום ה־index במעלה הערמה (כלומר מגלגלת את האיבר הזה כלפי מעלה) עד אשר העץ t הופך לערמה. |
| void **swap**(levelptr p, int index1, int index2) | פונקציה זו מקבלת שלושה פרמטרים:<br>**p** – מצביע לשורש ערמה,<br>**index1** ו־**index2** – המציינים את מיקומם של שני איברים בערמה.<br>הפונקציה מחליפה בין התכנים של שני האיברים האלה. |

**הנח שהפונקציות האלו כתובות וניתן להשתמש בהן בכל הסעיפים הבאים בלי לכתוב אותן מחדש. כמו כן, בעבור כל סעיף תוכל להשתמש בכל פונקציה שמומשה בסעיפים שלפניו.**

המשך בעמוד 10

**להלן הגדרות של משתנים גלובאליים:**

```
headder head;
headptr DS = &head;
long totalVotes=0;          //  סך כל הקולות
long legalVotes=0;          //  סך כל הקולות של המפלגות שעברו את אחוז החסימה
int levelHeapCapacity=0;
int levelHeapSize=0;
int pnum=0;                 //  משמש כאינדקס במערך המנדטים
int addmited = 0;           //  מספר המפלגות שעברו את אחוז החסימה
```

## ענה על כל הסעיפים.

**א.** לפניך פונקציה שכותרתה

```
voteptr addVotes(int i, int vnum)
```

פונקציה זו מקבלת את הפרמטרים:

**i** – מספר המפלגה,

**vnum** – מספר הקולות שיתווספו למפלגה זו.

פונקציה זו מוסיפה את מספר הקולות (vnum) למפלגה ה־i , ומחזירה מצביע לצומת שאליו יתווספו הקולות (vnum) .

**הנח כי** : $0 \le i < M$

בפונקציה חסרים **ארבעה** ביטויים, המסומנים במספרים בין סוגריים עגולים. רשום במחברת הבחינה את מספרי הביטויים החסרים (1)–(4) , בסדר עולה, וכתוב ליד כל מספר את הביטוי החסר שהוא מייצג.

המשך בעמוד 11

```
voteptr addVotes(int i,int vnum)
{
   voteptr vts;
   vts = malloc(sizeof(vote));
   vts->partyNum = i;
   vts->voteNum = vnum;
   vts->vleft = NULL;
   vts->vright = NULL;
   if (DS->parties[i].pvt == NULL)
            ________(1)________= insertVotesTree
                                (&(DS->votesTree),vts);
   else
         {
            vts->voteNum +=________(2)________;
            ________(3)________;
            DS->parties[i].pvt =________(4)________;
         }
   totalVotes += vnum;
   return vts;
}
```

**ב.** מהי סיבוכיות זמן הריצה של האלגוריתם המוצג בסעיף א', כאשר $n$ מציין את מספר המפלגות שהשתתפו בבחירות לכנסת?

1. $O(n)$
2. $O(log^2 n)$
3. $O(l)$
4. $O(log\ n)$

המשך בעמוד 12

**ג.** לפניך פונקציה **רקורסיבית** שכותרתה:

```
void results(voteptr p)
```

פונקציה זו מקבלת את הפרמטר p אשר מצביע לשורשו של עץ הקולות.

בעבור כל אחת מן המפלגות, הפונקציה משכנת במערך המנדטים את מספר המפלגה ואת אחוז הקולות שקיבלה מפלגה זו מכלל הקולות שנמנו.

הפונקציה דואגת שמערך המנדטים יהיה ממוין בסדר יורד על פי אחוז זה.

כמו־כן, הפונקציה מעדכנת את מספר המפלגות שעברו את אחוז החסימה וכן את סך כל הקולות על־ידי גריעת מספר הקולות של המפלגות שלא עברו את אחוז החסימה.
**הנח כי** המשתנה legalVotes כבר קיבל את ערכו של המשתנה totalVotes **לפני** הזימון של הפונקציה results .

בפונקציה חסרים **ארבעה** ביטויים, המסומנים במספרים בין סוגריים עגולים. רשום במחברת הבחינה את מספרי הביטויים החסרים (1)–(4) , בסדר עולה, וכתוב ליד כל מספר את הביטוי החסר שהוא מייצג.

```
void results(voteptr p)
{
      long  vnum;
      mandatptr  mdt;
      float  percent;
      float block = BLOCK_PERCENT;// אחוז החסימה
      mdt = DS->mandats;
      if(p)
         {
            ________(1)________;
            vnum = p->voteNum;
            percent = (float)vnum/totalVotes *100;
            if(percent > block)
            {
                 ________(2)________;
            }


              else   legalVotes =________(3)________;
            mdt[pnum].partNum = p->partyNum;//להזכירך pnum הוא משתנה גלובאלי
            mdt[pnum].numOfMandats=0;
            mdt[pnum].percent= percent;
            pnum = pnum+1;
            ________(4)________;
         }
}
```

המשך בעמוד 13

**ד.** מהי סיבוכיות זמן הריצה של האלגוריתם המוצג בסעיף ג', כאשר $n$ מציין את מספר המפלגות שהשתתפו בבחירות לכנסת?

1. $O(n)$
2. $O(log^2 n)$
3. $O(l)$
4. $O(log\ n)$

**ה.** לפניך פונקציה **רקורסיבית** שכותרתה:

```
void percolateDown(levelptr t, int index,int size)
```

פונקציה זו מקבלת את הפרמטרים:

**t** – מצביע לשורשו של עץ המיוצג באמצעות מערך,

**index** – מיקומו של איבר מסוים בעץ,

**size** – מספר האיברים בעץ.

כל תת־עץ הנמצא בעץ שלשורשו מצביע t הוא ערמה, פרט לתת־העץ ששורשו נמצא במקום ה־index .

הפונקציה "מחליקה" את האיבר שבמקום ה־index במורד הערמה (כלומר מגלגלת את האיבר הזה כלפי מטה) עד אשר העץ t הופך לערמה.

המשך בעמוד 14

בפונקציה חסרים **שלושה** ביטויים, המסומנים במספרים בין סוגריים עגולים. רשום במחברת הבחינה את מספרי הביטויים החסרים (1)–(3) , בסדר עולה, וכתוב ליד כל מספר את הביטוי החסר שהוא מייצג.

```
void percolateDown(levelptr  t, int index, int  size)
{
      int j;
      j = 2*index+1;
      if(j < size)
         {
            if ((j < size - 1) && (________(1)________))j++;
            if (t[index].level < t[j].level)
                  {
                        ________(2)________;
                        ________(3)________;
                  }
         }
}
```

**1.** מהי סיבוכיות זמן הריצה בעבור הזימון percolateDown(t,0,n) ?

1. $O(n)$
2. $O(log^2 n)$
3. $O(l)$
4. $O(log\ n)$

המשך בעמוד 15

ז. לפניך פונקציה שכותרתה:

```
void insertLevelsHeap(levelptr *ps, levelptr p,int *capacity, int *size)
```

פונקציה זו מקבלת את הפרמטרים:

**ps** – מצביע לראש ערמת המודדים,

**p** – מצביע לצומת בודד,

**capacity** – כתובתו של משתנה המכיל את גודלו של המערך המייצג את הערמה.

**size** - כתובתו של משתנה שמייצג את מספר האיברים שבערמה זו.

הפונקציה מוסיפה את הצומת שעליו מצביע p לערמה שעליה מצביע ps , כך שלאחר ההוספה ps יצביע לראש ערמת המודדים שהיא ערמת מקסימום.

אם גודלה של הערמה שווה לגודל המערך שבו משוכנת הערמה אזי הפונקציה מרחיבה את המערך כך שגודלו יהיה פי 3 ועוד 2 .

בפונקציה חסרים **ארבעה** ביטויים, המסומנים במספרים בין סוגריים עגולים. רשום במחברת הבחינה את מספרי הביטויים החסרים (1)–(4) , בסדר עולה, וכתוב ליד כל מספר את הביטוי החסר שהוא מייצג.

```
void insertLevelsHeap(levelptr *ps, levelptr p,int *capacity, int *size)
{
      if (*ps == NULL)
         {
            *capacity = 0;
            *size = 0;
         }
      if(_________(1)_________)
         {
            *capacity =_________(2)_________;
            *ps = realloc(*ps,sizeof(levelRec)*(*capacity));
         }
         (*ps)[*size].index = p->index;
         (*ps)[*size].level =p->level;
         _________(3)_________;
         _________(4)_________;
}
```

המשך בעמוד 16

**ח.** לפניך פונקציה שכותרתה:

```
void deleteLevel(levelptr *t, int index, int *size)
```

פונקציה זו מקבלת את הפרמטרים:

**t** – מצביע לראש ערמת המודדים,

**index** – מיקומו של איבר בערמה זו,

**size** – כתובתו של משתנה שמייצג את מספר האיברים שבערמה זו.

הפונקציה מבטלת את האיבר שבערמה במיקום index ומעדכנת את מספר האיברים בערמה.

**הערה:** לאחר ביטול האיבר, ערמת המודדים תישאר ערמת מקסימום.

**בפונקציה** חסרים **שלושה** ביטויים, המסומנים במספרים בין סוגריים עגולים. רשום במחברת הבחינה את מספרי הביטויים החסרים (1)–(3) , בסדר עולה, וכתוב ליד כל מספר את הביטוי החסר שהוא מייצג.

```
void deleteLevel(levelptr *t, int index, int *size)
{
      levelptr p = *t;
      swap(p,index,*size-1);
      ________(1)________;
      if ((index > 0) && (________(2)________))
            percolateUP(p,index);
      else
            ________(3)________;
}
```

המשך בעמוד 17

**ט.** מהי סיבוכיות זמן הריצה בעבור הזימון **deleteLevel** (t,0,n) ?

1. $O(n)$
2. $O(log^2 n)$
3. $O(l)$
4. $O(log\ n)$

**י.** לפניך פונקציה שכותרתה:

```
void additions(int total)
```

הפונקציה מקבלת את הפרמטר total שהוא מספר המנדטים **העודפים**.

הפונקציה מחלקת את total (מספר המנדטים **העודפים**) בין כל המפלגות שעברו את אחוז החסימה כמפורט בעמודים 2-3 שלעיל.

בפונקציה חסרים **חמישה** ביטויים, המסומנים במספרים בין סוגריים עגולים. רשום במחברת הבחינה את מספרי הביטויים החסרים (1)–(5) , בסדר עולה, וכתוב ליד כל מספר את הביטוי החסר שהוא מייצג.

```
void additions(int total)
{
      int max;
      int  level, mandats, partyNum, numOfVotes;
      levelptr lvl;
      while(total)
         {
            max = DS->levelHeap->index;
            deleteLevel(&(DS->levelHeap),0,&levelHeapSize);
            mandats =_________(1)_________;
            partyNum = DS->mandats[max].partNum;
```

המשך בעמוד 18

```
            numOfVotes =_________(2)_________;

            level=_________(3)_________;

            lvl = malloc(sizeof(levelRec));

            lvl->index =_________(4)_________;

            lvl->level = level;

            _________(5)_________;

            total = total-1 ;

        }

}
```

**י"א.** מהי סיבוכיות זמן הריצה בעבור הזימון **additions(k)** , בהנחה ש־$n$ מפלגות עברו את אחוז החסימה בבחירות לכנסת?

1. $O(n \log n)$
2. $O(k \log n)$
3. $O(k \log k)$
4. $O(n)$

**י"ב.** לפניך פונקציה שכותרתה:

```
void calculateMandats(int total)
```

הפונקציה מקבלת את הפרמטר total שהוא מספר המנדטים לחלוקה.

הפונקציה מבצעת חלוקה של **כל** total המנדטים בין כל המפלגות שעברו את אחוז החסימה (addmited) , כמתואר בפתיח השאלה, וגם מטפלת בחלוקת המנדטים העודפים (אם ישנם).

נוסף על כך, הפונקציה משכנת במערך המנדטים, בעבור כל מפלגה שעברה את אחוז החסימה, את אחוז הקולות שקיבלה מפלגה זו מכלל הקולות של המפלגות שעברו את אחוז החסימה, ואת מספר המנדטים שלה. כמו כן, היא בונה את **ערמת המודדים**.

המשך בעמוד 19

בפונקציה חסרים **חמישה** ביטויים, המסומנים במספרים בין סוגריים עגולים. רשום במחברת הבחינה את מספרי הביטויים החסרים (1)–(5) , בסדר עולה, וכתוב ליד כל מספר את הביטוי החסר שהוא מייצג.

```
void calculateMandats(int total)
{
     int  n,partyNo, i, num;
     long  nvotes;
     levelptr  lvl;
     float prcnt,level;
     num = NUM_OF_DELEGATES; // 120 מספר המנדטים בכנסת הוא
     lvl = malloc(sizeof(levelRec));
     for(i=0;i<addmited;i++)
        {
           partyNo = DS->mandats[i].partNum;
           nvotes = DS->parties[partyNo].pvt->voteNum;
           DS->mandats[i].percent = (float)nvotes/legalVotes *100;
           prcnt = DS->mandats[i].percent;
           n = num*prcnt/100 ;
           total =_________(1)_________;
           _________(2)_________ = n;
           level=_________(3)_________;
           lvl->index = i;
           lvl->level = level;
           _________(4)_________;
        }
     _________(5)_________;
}
```

המשך בעמוד 20

**שאלה 2 – שאלת חובה (25 נקודות)**

**בשאלה זו שישה סעיפים. עליך לענות על כל הסעיפים.**

נתונה ספרייה הכוללת הגדרות ופונקציות לטיפול במחסנית (Stack) , ובה ההגדרות והפונקציות שלהלן:

**הגדרות:**

```
#define STACK_MAX_SIZE 100 // גודל מקסימלי של מחסנית
typedef  int  stack_item;
typedef struct
{
     int top;
     stack_item data[STACK_MAX_SIZE];
} stack;
```

**הפונקציות:**

| | |
|---|---|
| void init(stack *s) | פונקציה זו מאתחלת את המחסנית s להיות ריקה. |
| void push(stack *s , stack_item x) | פונקציה זו מוסיפה את האיבר x למחסנית s<br>**הנחה:** המחסנית s מאותחלת ואינה מלאה. |
| stack_item pop(stack *s) | פונקציה זו שולפת את האיבר שבראש המחסנית s ומחזירה את ערכו.<br>**הנחה:** המחסנית s מאותחלת ואינה ריקה. |
| stack_item top(stack *s) | פונקציה זו מחזירה את ערכו של האיבר שבראש המחסנית s מבלי להוציאו מהמחסנית.<br>**הנחה**: המחסנית s מאותחלת ואינה ריקה. |
| int isEmpty(stack *s) | פונקציה זו מחזירה את הערך 1 אם המחסנית s ריקה, אחרת - היא מחזירה את הערך 0 .<br>**הנחה:** המחסנית s מאותחלת. |

**הנח שהפונקציות האלו כתובות וניתן להשתמש בהן בכל הסעיפים הבאים בלי לכתוב אותן מחדש.**

**כמו כן, בעבור כל סעיף תוכל להשתמש בכל פונקציה שמומשה בסעיפים שלפניו.**

המשך בעמוד 21

נתונה ההצהרה הזאת:

```
stack *s ;
```

**ענה על <u>כל</u> הסעיפים א'-ו'.**

**א.** לפניך פונקציה המקבלת מחסנית s המכילה מספרים שלמים הגדולים מ־0 . הפונקציה מחזירה את הסכום הגדול ביותר של שני איברים סמוכים שבמחסנית s .

**הנח כי:**

1. במחסנית s יש שני איברים לפחות.
2. בתום ביצוע הפונקציה, המחסנית s נשארת ללא שינוי.

**דוגמה:** בעבור המחסנית s שלהלן:

| |
|---|
| 2 |
| 8 |
| 4 |
| 13 |
| 9 |
| 11 |
| 4 |
| 1 |

**s**

סכום האיברים הסמוכים 13 ו־9 גדול מסכום כל שני איברים סמוכים שבמחסנית s , ולכן הפונקציה תחזיר את המספר 22 .

המשך בעמוד 22

בפונקציה חסרים **חמישה** ביטויים, המסומנים במספרים בין סוגריים עגולים. רשום במחברת הבחינה את מספרי הביטויים החסרים (1)–(5) , בסדר עולה, וכתוב ליד כל מספר את הביטוי החסר שהוא מייצג.

```
int maxsum(stack *s)
{
      int max=0;
      int  x,y;
      stack stemp;
      ________(1)________;
      while (!isEmpty(s))
          {
             x=pop(s);
             if(________(2)________)
             {
                    y=________(3)________;
                   if(x+y>max)
                          max=x+y;
             }
             push(&stemp,x);
          }
      while(________(4)________)
          {
             ________(5)________;
          }
      return max;
}
```

המשך בעמוד 23

**ב.** לפניך פונקציה המקבלת שתי מחסניות s1 ו־s2 , המכילות מספרים שלמים הגדולים מ־0 .

הפונקציה תחזיר את סכומם של זוג האיברים הסמוכים, **הקרוב ביותר** לראש המחסנית s1 , שהוא גדול מסכום של כל זוג איברים סמוכים במחסנית s2 .

אם אין זוג כזה – הפונקציה תחזיר את הערך 0 .

**הנחה:** בכל אחת מהמחסניות ישנם שני איברים לפחות.

**הערה:** בתום ביצוע הפונקציה, המחסנית s2 נשארת ללא שינוי. אולם המחסנית s1 לא תכיל בהכרח את כל האיברים שהיו בה לפני הפעלת הפונקציה.

**דוגמה:**

בעבור המחסניות s1 ו־s2 שלהלן:

| s2 |
|---|
| 2 |
| 8 |
| 4 |
| 13 |
| 4 |
| 1 |
| 9 |
| 11 |

| s1 |
|---|
| 7 |
| 9 |
| 12 |
| 8 |
| 14 |
| 6 |
| 7 |

סכום המספרים 9 ו־12 במחסנית s1 גדול מסכום כל שני איברים סמוכים במחסנית s2 , ולכן הפונקציה תחזיר 21 .

**שים לב:** במחסנית s1 קיים זוג איברים סמוכים שסכומם גדול מ־21 , אבל מאחר שהוא נמצא עמוק יותר במחסנית, הפונקציה תחזיר 21 .

המשך בעמוד 24

בפונקציה חסרים **ארבעה** ביטויים, המסומנים במספרים בין סוגריים עגולים. רשום במחברת הבחינה את מספרי הביטויים החסרים (1)–(4) , בסדר עולה, וכתוב ליד כל מספר את הביטוי החסר שהוא מייצג.

```
int topbig(stack *s1, stack *s2)
{
     int  x,y;
     int max_________(1)_________;
     while (!isEmpty(s1))
         {
            x=_________(2)_________;
            if (!isEmpty(s1))
            {
                y=_________(3)_________;
                if (x+y > max)_________(4)_________;
            }
         }
     return 0;
}
```

**ג.** מהי סיבוכיות זמן הריצה של האלגוריתם המוצג בסעיף ב', בהנחה שמספר האיברים בכל אחת מן המחסניות הוא $n$ ?

1. $O(n)$
2. $O(log^2 n)$
3. $O(l)$
4. $O(n^2)$

המשך בעמוד 25

**ד.** לפניך פונקציה **רקורסיבית** בשם isbig אשר מקבלת ארבעה פרמטרים: מחסנית s , שני מספרים שלמים (x ו־y) ומשתנה flag2 .

אם x + y גדול מהסכום של כל זוג איברים סמוכים שבמחסנית s אזי הפונקציה תחזיר את הערך 1 ; אחרת – הפונקציה תחזיר את הערך 0 .

אם קיימים במחסנית s שני איברים סמוכים שסכומם גדול או שווה ל־x + y אזי flag2 יקבל את הערך 1 ; אחרת – הוא יקבל את הערך 0 .

**הנחות:**

- במחסנית s יש שני איברים לפחות.
- בתום ביצוע הפונקציה, המחסנית s נשארת ללא שינוי.
- בעת זימון הפונקציה **isbig** ערכו ההתחלתי של המשתנה flag2 הוא 0 .

**דוגמה:** בעבור המחסנית s שלהלן:

| |
|---|
| 2 |
| 8 |
| 4 |
| 13 |
| 4 |
| 1 |
| 9 |
| 11 |

**s**

אם x = 3 ו־ y = 6 תחזיר הפונקציה את הערך 0 .

אם x = 9 ו־ y = 16 תחזיר הפונקציה את הערך 1 .

המשך בעמוד 26

בפונקציה חסרים **חמישה** ביטויים, המסומנים במספרים בין סוגריים עגולים. רשום במחברת הבחינה את מספרי הביטויים החסרים (1)–(5) , בסדר עולה, וכתוב ליד כל מספר את הביטוי החסר שהוא מייצג.

```
int isbig(stack *s, int x, int y, int flag2)
{
      int flag;
      if  (isEmpty(s) &&________(1)________)  flag=1;
      else
          if ((isEmpty(s) &&________(2)________))  flag=0;
          else
             {
                  int mis1=pop(s);
                  if(!isEmpty(s))
                    {
                          int mis2=top(s);
                          if(x+y<= mis1+mis2)
                             {
                                flag=0;
                                flag2=________(3)________;
                             }
                    }
                    flag=________(4)________;
                    ________(5)________;
             }
      return flag;
}
```

המשך בעמוד 27

**ה.** לפניך פונקציה **רקורסיבית**, המקבלת שתי מחסניות s1 ו־s2 שמכילות מספרים שלמים הגדולים מ־0 .

הפונקציה תחזיר את הסכום של זוג האיברים הסמוכים **הקרוב ביותר** לראש המחסנית s1 , אשר גדול מסכום כל זוג איברים סמוכים במחסנית s2 .

אם אין זוג כזה – הפונקציה תחזיר את הערך 0 .

**הנחה:** בכל אחת מהמחסניות ישנם שני איברים לפחות.

בפונקציה חסרים **שלושה** ביטויים, המסומנים במספרים בין סוגריים עגולים. רשום במחברת הבחינה את מספרי הביטויים החסרים (1)–(3) , בסדר עולה, וכתוב ליד כל מספר את הביטוי החסר שהוא מייצג.

```
int firstbig(stack *s1, stack *s2)
{
      if(________(1)________)  return 0;
      else
          {
             int  x=pop(s1);
             if(!isEmpty(s1))
                 {
                     int  y=top(s1);
                     if(________(2)________)
                            return x+y;
                 }
                 ________(3)________;
          }
}
```

המשך בעמוד 28

**1.** מהי סיבוכיות זמן הריצה של האלגוריתם המוצג בסעיף ד', בהנחה שמספר האיברים בכל אחת מהמחסניות הוא $n$ ?

1. $O(n)$
2. $O(log^2 n)$
3. $O(l)$
4. $O(n^2)$

המשך בעמוד 29

## פרק שני (35 נקודות)

**ענה על אחת מבין השאלות 3–4 (לכל שאלה – 35 נקודות).**

### שאלה 3 (35 נקודות)

**בשאלה זו עשרים סעיפים. עליך לענות על כל הסעיפים.**
**בכל סעיף נתונות ארבע תשובות, שרק אחת מהן נכונה. בכל סעיף בחר את התשובה הנכונה, והקף בעיגול את הספרה המייצגת אותה בדף התשובות שבנספח א'.**

**א.** $T(n) = 81\,T(n/9) + 7n^4 \log n$ היא פונקציית זמן הריצה של אלגוריתם הפועל על קלט שגודלו $n$. מהי סיבוכיות זמן הריצה של אלגוריתם זה?

1. $\Theta(n^2 \log^2 n)$
2. $\Theta(n^6 \log n)$
3. $\Theta(n^4 \log^2 n)$
4. $\Theta(n^4 \log n)$

**ב.** $T(n) = 81\,T(n/9) + n^2 \log n$ היא פונקציית זמן הריצה של אלגוריתם הפועל על קלט שגודלו $n$. מהי סיבוכיות זמן הריצה של אלגוריתם זה?

1. $\Theta(n^2 \log n)$
2. $\Theta(n^2)$
3. $\Theta(n^2 \log^2 n)$
4. $\Theta(n^4 \log^2 n)$

**ג.** $T(n) = 6\,T(n-1) - 9\,T(n-2) + n \cdot 3^n + n^2 \cdot 2^n$ היא פונקציית זמן הריצה של אלגוריתם הפועל על קלט שגודלו $n$. מהי סיבוכיות זמן הריצה של אלגוריתם זה?

1. $\Theta(n^2\, 2^n)$
2. $\Theta(n^3\, 3^n)$
3. $\Theta(n\, 3^n)$
4. $\Theta(n^3\, 6^n)$

המשך בעמוד 30

**ד.** $T(n) = T(n/3) + T(2n/3) + cn$ היא פונקציית זמן הריצה של אלגוריתם הפועל על קלט שגודלו $n$ . מהי סיבוכיות זמן הריצה של אלגוריתם זה?

1. $\Theta(n \log n)$
2. $\Theta(\log n)$
3. $\Theta(n)$
4. $\Theta(\log^2 n)$

**בסעיפים ה'–ו' התייחס להגדרות האלה:**

$T(n)$ ו־$S(n)$ שלהלן, מתארות פונקציות זמן ריצה של שני אלגוריתמים.

$$T(n) = (T(n-1))^2 \quad T(1) = 2$$

$$S(n) = \sum_{i=1}^{\log n} \log T(i)$$

**ה.** מהי סיבוכיות זמן הריצה של האלגוריתם שפונקציות זמן הריצה שלו היא $T(n)$ ?

1. $\Theta(2^{2^n})$
2. $\Theta(\log n)$
3. $\Theta(2^n)$
4. $\Theta(n^2)$

**ו.** מהי סיבוכיות זמן הריצה של האלגוריתם שפונקציית זמן הריצה שלו היא $S(n)$ ?

1. $\Theta(n \log n)$
2. $\Theta(n)$
3. $\Theta(n^2)$
4. $\Theta(\log n)$

המשך בעמוד 31

**7.** נתון קטע הקוד הזה:

```
void Tr(int n)
{
      int k, x;
      if(n>1)
          {
             x=n;
             while(x>1)
                 x=x- n/10;
             k=2;
             while(k<n)
                 k=k*2;
             Tr(n-1);
          }
}
```

מהי סיבוכיות זמן הריצה של קטע קוד זה כפונקציה של n ?

1. $\Theta(n^2 \log n)$
2. $\Theta(\log^3 n)$
3. $\Theta(\log n!)$
4. $\Theta(n)$

המשך בעמוד 32

**ח.** להלן הפונקציות האלה:

$f_2(n) = O(g_2(n))$ $f_1(n) = O(g_1(n))$ בהנחה שלכל n מתקיים: $f_1(n) \geq f_2(n)$ .

**סטודנט A** טוען כי: $f_1(n) - f_2(n) = O(min(g_1(n), g_2(n)))$ .

**סטודנט B** טוען כי: $\frac{f_1(n)}{f_2(n)} = O(\frac{g_1(n)}{g_2(n)})$ .

בחר את ההיגד הנכון מבין ההיגדים שלהלן:

1. שני הסטודנטים צודקים בטענתם.
2. רק סטודנט A צודק.
3. רק סטודנט B צודק.
4. שני הסטודנטים שוגים בטענתם.

## בסעיפים ט'–י' התייחס לפונקציות שלהלן:

להלן הפונקציות האלה:

$$f(n) = O(n^2) \qquad g(n) = \Omega(\sqrt{n}) \qquad h(n) = O(\log n)$$

**ט.** בחר את ההיגד **שאינו** נכון מבין ההיגדים שלהלן:

1. $\frac{f(n)}{h(n)} = \Omega\left(\frac{1}{\log n}\right)$
2. $h(n) \cdot g(n) = \Omega(\sqrt{n})$
3. $2^{h(n)} = O(n)$
4. $\frac{f(n)}{h(n)} = O(n^2 \log n)$

המשך בעמוד 33

**י.** בחר את ההיגד הנכון מבין ההיגדים שלהלן:

1. $\frac{f(n)+h(n)}{g(n)} = O(\log n)$
2. $(f(n)+g(n))^2 = \Omega(n)$
3. $f(n)\cdot h(n) = O(n \log n)$
4. $f(n)\cdot g(n) = O(n^2\sqrt{n})$

**בסעיפים י"א–כ' הנך רשאי להשתמש בפונקציות האלה:**

| שם הפונקציה | תיאור הפונקציה | סיבוכיות זמן הריצה של הפונקציה במקרה הגרוע ביותר |
|---|---|---|
| Sort(S) | פונקציה הממיינת את איבריה של הקבוצה S ומחזירה אותה ממוינת על־פי סדר עולה. | $O(n \log n)$ |
| Select(S,k) | פונקציה המוצאת ומחזירה את האיבר ה־k הקטן ביותר מבין איברי הקבוצה S , כלומר היא מוצאת את ערך המיקום ה־k בקבוצה S . | $O(n)$ |
| Partition(S,z,S1,S2) | פונקציה המחלקת את הסדרה S בת n האיברים לשתי סדרות S1 ו־S2 באופן הבא:<br>בסדרה S1 יהיו האיברים הקטנים מ־z או השווים לו, ובסדרה S2 יהיו האיברים הגדולים מ־z .<br>**שים לב:** הסדרות S1 ו־S2 , אינן בהכרח ממוינות. | $O(n)$ |

**הנח שהפונקציות האלו כתובות וניתן להשתמש בהן בכל הסעיפים הבאים בלי לכתוב אותן מחדש.**

**כמו כן, בעבור כל סעיף תוכל להשתמש בכל פונקציה שמומשה בסעיפים שלפניו.**

המשך בעמוד 34

**בסעיפים י"א–י"ג התייחס לבעיה שלהלן:**

נגדיר **איבר רוב** כאיבר שמופיע במערך בגודל n יותר מ־n/2 פעמים.

נתון מערך A המכיל $n$ מספרים טבעיים כאשר $n$ הוא מספר אי־זוגי.

לפניך אלגוריתם יעיל אשר בודק אם קיים **איבר רוב** במערך A .

אם קיים – האלגוריתם יחזיר את המספר שהוא איבר הרוב במערך A ;

אחרת – האלגוריתם יחזיר את הערך 1- .

**האלגוריתם:**

**צעד 1 :** _____(1)_____

**צעד 2 :** _____(2)_____

**צעד 3 :** _____(3)_____

באלגוריתם הנתון חסרים **שלושה** ביטויים, המסומנים במספרים בין סוגריים עגולים.

התשובה הנכונה עבור כל אחד מהביטויים החסרים מופיעה בסעיפים האלה:

**י"א.** הביטוי החסר (1) הוא:

1. $M = \text{select}(A, \frac{n+1}{2})$
2. $\text{partition}(A, \frac{n+1}{2}, A1, A2)$
3. $M = \text{select}(A, \frac{n}{2})$
4. $\text{Sort}(A)$

**י"ב.** הביטוי החסר (2) הוא:

1. L מקבל את מספר ההופעות של M במערך A
2. L1 מקבל את מספר המספרים שבמערך A הקטנים מ־M או שווים לו
3. L2 מקבל את מספר המספרים שבמערך A הגדולים מ־M או שווים לו
4. L3 מקבל את מספר ההופעות של הערך השכיח במערך הממוין A

המשך בעמוד 35

**י"ג.** הביטוי החסר (3) הוא:

1. אם $L1 \geq \frac{n+1}{2}$ אזי החזר את המספר $\frac{L1}{2}$ , אחרת – החזר את הערך 1-
2. אם $L \geq \frac{n+1}{2}$ אזי החזר את המספר M , אחרת – החזר את הערך 1-
3. אם $L2 \geq \frac{n+1}{2}$ אזי החזר את המספר $L_2$ , אחרת – החזר את הערך 1-
4. אם $L3 \geq \frac{n+1}{2}$ אזי החזר את המספר השכיח במערך A ,
אחרת – החזר את הערך 1-

**י"ד.** מהי סיבוכיות זמן הריצה של האלגוריתם הנתון?

1. $\Theta(n \log n)$
2. $\Theta(\log n)$
3. $\Theta(n)$
4. $\Theta(n^2)$

## בסעיפים ט"ו–כ' התייחס לבעיה שלהלן:

נתונה קבוצה סופית S , המכילה n מספרים שלמים חיוביים ואיבריה משוכנים במערך A , החל באינדקס 1 ועד לאינדקס ה־n (כולל).

**הנח כי** כל הפעולות המתמטיות, כגון חיבור, כפל והשוואה, ניתנות לביצוע בזמן $O(1)$ .

לפניך אלגוריתם יעיל בשם TR1 , אשר מחזיר את הערך "no" אם קיימת בקבוצה S תת־קבוצה כלשהי T , כך שסכום איבריה של T יהיה קטן מ־$|T|^3$;

אחרת – אם בעבור **כל** תת־קבוצה T של S מתקיים: $\sum_{t \in T} t \geq |T|^3$ , האלגוריתם יחזיר את הערך "yes" .

**שים לב:**

$|T|$ מציין את מספר האיברים שבקבוצה T .

המשך בעמוד 36

**האלגוריתם:**

**TR1(A, n)**

**צעד 1 :** _____(1)_____ .

**צעד 2 :** Sum ← 0 .

**צעד 3 :** עבור k המקבל ערכים החל מ־1 ועד _____(2)_____ בצע:

**3.1** Sum ← Sum + _____(3)_____

**3.2** אם _____(4)_____ אז עצור והחזר "no"

**צעד 4 :** החזר "yes".

באלגוריתם הנתון חסרים **ארבעה** ביטויים, המסומנים במספרים בין סוגריים עגולים.

התשובה הנכונה עבור כל אחד מהביטויים החסרים מופיעה בסעיפים האלה:

**ט"ו.** הביטוי החסר (1) הוא:

1. x = select(A,k)
2. $x = select\left(A, \left\lceil \frac{n+1}{2} \right\rceil\right)$
3. x = select(A,1)
4. Sort(A)

**ט"ז.** הביטוי החסר (2) הוא:

1. k
2. x
3. n
4. $k^3$

המשך בעמוד 37

**י"ז.** הביטוי החסר (3) הוא:

1. $x = select(A,k)$
2. $select\left(A,\left\lceil \frac{k+1}{2} \right\rceil\right)$
3. $A[k]$
4. $A[Select(A,k)]$

**י"ח.** הביטוי החסר (4) הוא:

1. $Sum < k$
2. $Sum < k^3$
3. $Sum < k^3 + Select(S,k)$
4. $Sum < k^3 - Select(S,k)$

**י"ט.** מהי סיבוכיות זמן הריצה של האלגוריתם TR1 ?

1. $\Theta(n \log n)$
2. $\Theta(\log n)$
3. $\Theta(l)$
4. $\Theta(n)$

**כ.** בהנחה שהקבוצה הנתונה S מכילה n מספרים שלמים וחיוביים, וכל איבר בה גדול או שווה ל־ 1 וקטן או שווה ל־ $n^2$ ; מה תהיה כעת סיבוכיות זמן הריצה של האלגוריתם TR1 ?

1. $\Theta(n)$
2. $\Theta(n^2 \log n)$
3. $\Theta(n^2)$
4. $\Theta(n \log n)$

המשך בעמוד 38

**שאלה 4 (35 נקודות)**

**בשאלה זו עשרים סעיפים. עליך לענות על כל הסעיפים.**
**בכל סעיף נתונות ארבע תשובות, שרק אחת מהן נכונה. בכל סעיף בחר את התשובה הנכונה, והקף בעיגול את הספרה המייצגת אותה בדף התשובות שבנספח א'.**

להלן מבנה של צומת בעץ חיפוש בינארי בשפת C :

```
typedef struct node
  {
     int info;                  //  שדה מידע
     struct node *left;         //  מצביע לבן שמאלי
     struct node *right;        //  מצביע לבן ימני
  }tree_node, *tree_ptr ;

tree_ptr root ;
```

**בסעיפים א'–ח' התייחס לבעיה שלהלן:**

לפניך הגדרה חדשה:

**עץ מיוחד** הוא עץ חיפוש בינארי מאוזן, שבו הערך המוחלט של ההפרש בין מספר צומתי תת־העץ השמאלי ובין מספר צומתי תת־העץ הימני – קטן או שווה 1 .
כמו כן, תת־העץ השמאלי שלו ותת־העץ הימני שלו הם **עצים מיוחדים** גם כן.
**הערה:** עץ ריק נחשב גם הוא לעץ מיוחד.

נתון מערך A המכיל n מספרים טבעיים הגדולים מ־0 וממוין לפי סדר עולה.

מערך A הוא מערך **גלובלי**.

לפניך פונקציה **רקורסיבית** בעלת סיבוכיות זמן ריצה $\Theta(n)$ , אשר מקבלת שני אינדקסים i ו־j במערך A .

הפונקציה בונה **עץ מיוחד** מן המספרים שבמערך הזה, מן המספר שבאינדקס i ועד למספר שבאינדקס j (כולל), כאשר לשורשו של עץ זה מצביע t .

המשך בעמוד 39

```
void Q4(int i, int j, tree_ptr *t)
      {
         int m ;
         if (i <= j)
               if (i = = j)
                  {
                    *t = (tree_ptr)malloc(sizeof (tree_node)) ;
                    (*t)-> info =_______(1)_______;
                    (*t)-> left = (*t)-> right =_______(2)_______;
                  }
               else
                  {
                    m =_______(3)_______;
                    *t = (tree_ptr) malloc (sizeof (tree_node)) ;
                    (*t)-> info =_______(4)_______;
                    _______(5)_______;
                    _______(6)_______;
                  }
      }
```

בפונקציה זו חסרים **שישה** ביטויים, המסומנים במספרים בין סוגריים עגולים.

התשובה הנכונה עבור כל אחד מהביטויים החסרים מופיעה בסעיפים האלה:

**א.** הביטוי החסר (1) **לא** יכול להיות:

1. A[i]
2. A[j]
3. A[(i+j)/2]
4. A[(j–i)/2]

המשך בעמוד 40

**ב.** הביטוי החסר (2) הוא:

1. t
2. Q4(i, j, &((*t)->left))
3. NULL
4. Q4(i, j, &((*t)->right))

**ג.** הביטוי החסר (3) הוא:

1. A[(i+j)/2]
2. (i+j)/2
3. (j–i)/2
4. j–i

**ד.** הביטוי החסר (4) הוא:

1. A[(i+j)/2]
2. A[j]
3. A[i]
4. A[(m–i)]

**ה.** הביטוי החסר (5) הוא:

1. Q4(i, m–1, &((*t)->left))
2. Q4(m+1, j,&((*t)->left))
3. Q4(i, m–1, &((*t)->right))
4. Q4((j–m)/2, j, &((*t)->right))

המשך בעמוד 41

**ו.** הביטוי החסר (6) הוא:

1. Q4(i, m–1, &((*t)->left))
2. Q4(m+1, j, &((*t)-> right))
3. Q4(i, m–1, &((*t)->right))
4. Q4(i, (m–i)/2, &((*t)->left))

**ז.** נסמן ב־$T(n)$ את פונקציית זמן הריצה של הזימון Q4(0, n–1, &root) .

הפונקציה T(n) היא:

1. $T(n) = 2T(n/2) + c$
2. $T(n) = 2T(n-1) + c$
3. $T(n) = 2T(n/2) + c \cdot n$
4. $T(n) = 2T(n-1) + c \cdot n$

**ח.** נתון מערך B של n מספרים טבעיים הגדולים מ־0 וקטנים מ־3n .

לפניך אלגוריתם **יעיל** אשר בונה מאיברי מערך B **עץ מיוחד** שלשורשו מצביע root .

**האלגוריתם:**

**צעד 1:** בצע _____(1)_____ על המערך B .

**צעד 2:** הפעל את Q4(0, n–1, &root) על המערך B .

באלגוריתם הזה חסר ביטוי אחד, המסומן ב־(1) .

הביטוי החסר (1) הוא:

1. מיון מנייה (counting sort)
2. מיון מהיר (quick sort)
3. מיון בסיס (radix sort)
4. מיון ערמה (heap sort)

המשך בעמוד 42

**ט.** סיבוכיות זמן הריצה של האלגוריתם הנתון בסעיף ח' היא:

1. $O(\log n)$
2. $O(n)$
3. $O(n \log n)$
4. $O(n \log^2 n)$

## בסעיפים י'–י"ב התייחס לבעיה שלהלן:

סטודנט A סיפר לסטודנט B , שמצא "אלגוריתם מיוחד" בעל סיבוכיות זמן ריצה O(n) , אשר מקבל ערמה בת n איברים וממיר אותה ל"עץ המיוחד" שתואר לעיל.

סטודנט B השיב לסטודנט A , שלא ייתכן שקיים אלגוריתם כזה, וניסה לנמק את טענתו בעזרת ההוכחה שלהלן:

**ההוכחה:**

נניח בשלילה שקיים אלגוריתם כזה.

עתה נבצע ____(1)____

סיבוכיות זמן הריצה שקיבלנו עד כה היא: ____(2)____

לאור זאת, לטענתו של הסטודנט B :____(3)____

בהוכחה זו חסרים **שלושה** ביטויים, המסומנים במספרים בין סוגריים עגולים.

התשובה הנכונה עבור כל אחד מהביטויים החסרים מופיעה בסעיפים האלה:

**י.** הביטוי החסר (1) הוא:

1. סריקת preorder
2. סריקת inorder
3. מיון heap sort
4. מיון מנייה (counting sort) או מיון בסיס (radix sort)

המשך בעמוד 43

**י"א.** הביטוי החסר (2) הוא:

1. $O(n^2)$
2. $O(\log n)$
3. $O(n)$
4. $O(n \log n)$

**י"ב.** הביטוי החסר (3) הוא:

1. מתקבלת סתירה, מכיוון שזמן הריצה של המיון הוא $O(\log n)$ , והרי אין אפשרות למיין בזמן ריצה הקטן מ־$O(n)$ .
2. מתקבלת סתירה, מכיוון שזמן הריצה של המיון הוא $O(\log n)$ , והרי אין אפשרות למיין במודל השוואות בזמן ריצה הקטן מ־$O(n)$ .
3. מתקבלת סתירה, מכיוון שזמן הריצה של המיון הוא $O(n)$ והרי אין אפשרות למיין בזמן ריצה הקטן מ־$O(n \log n)$ .
4. מתקבלת סתירה, מכיוון שזמן הריצה של האלגוריתם החדש הוא $O(n)$ , והרי אין אפשרות למיין במודל השוואות בזמן ריצה הקטן מ־$O(n \log n)$ .

## בסעיפים י"ג–י"ד התייחס לבעיה שלהלן:

נתון מערך A שגודלו n .

לפניך אלגוריתם, בעל זמן הריצה הממוצע המהיר ביותר, אשר מחפש במערך שלושה איברים x, y ו־z כך ש־ $x + y = z$ .
אם מצא שלושה איברים כאלה – יחזיר עבורם "כן".
אם אין שלשה שכזו, האלגוריתם יחזיר "לא".

**האלגוריתם:**

**צעד 1 :** הכנס את כל הסכומים המתקבלים מזוגות כל המספרים שבמערך A למבנה הנתונים ____(1)____ , שבו זמן הריצה הממוצע להכנסת סכום של זוג אחד יהיה $O(1)$ .

**צעד 2 :** עבור כל איבר t שבמערך A , אם t אינו נמצא ב־____(1)____ אזי החזר "לא".

**צעד 3 :** החזר "כן".

המשך בעמוד 44

באלגוריתם הזה חסר ביטוי אחד המסומן ב־(1) .

**י"ג.** הביטוי החסר (1) הוא:

1. עץ מאוזן AVL
2. ערמת מינימום
3. מחסנית
4. טבלת גיבוב/ערבול (hashing)

**י"ד.** אלגוריתם זה פועל בזמן ממוצע:

1. $O(n^2 \log n)$
2. $O(n^3)$
3. $O(n^2)$
4. $O(n^3 \log n)$

**ט"ו.** תהי S קבוצה ובה n מספרים שלמים השונים זה מזה.

ברצוננו להתאים עבור קבוצה זו מבנה נתונים שיקיים את שני התנאים שלהלן:

- מבנה הנתונים ייבנה בזמן $O(n)$ .
- מבנה הנתונים יתמוך בזמן $O(\log n)$ בכל אחת מן הפעולות האלה:

extract_min – חילוץ האיבר המינימלי מן הקבוצה S ,
insert(k) – הוספת הערך k לקבוצה S ,
delete(k) – מחיקת הערך k מן הקבוצה S .

מבנה הנתונים המתאים יכיל:

1. עץ מאוזן AVL וטבלת גיבוב/ערבול (hashing)
2. ערמת מינימום וטבלת גיבוב/ערבול (hashing)
3. ערמת מינימום בלבד
4. טבלת גיבוב/ערבול (hashing) בלבד

המשך בעמוד 45

**בסעיפים ט"ז–י"ט התייחס לבעיה שלהלן:**

יהי $G = (V, E)$ גרף מכוון ללא מעגלים (DAG) עם משקולות ממשיים על הקשתות, כלומר $w: E \to R$.

לפניך אלגוריתם יעיל למציאת משקל המסלול הקצר ביותר מ־s ל־t המכיל לפחות שלוש קשתות, כאשר s ו־t הם קדקודים בגרף הנתון.

**האלגוריתם:**

**צעד 1 :** בהינתן $G = (V, E)$ בנה גרף חדש $G' = (V', E')$ באופן הבא:

$V' = \{v_1, v_2, v_3, v_4 \mid for\ each\ \ v \in V\}$ ו־

$E' = \{(v_1, u_2), (v_2, u_3),$ ____(1)____ $\mid (v, u) \in E\}$

$\forall e = (v_i, u_j) \in E'\ \ w(e) = \{w(v, u) \mid (v, u) \in E\ \ and\ (v_i, u_j) \in E'\}$

**צעד 2 :** הרץ את האלגוריתם ____(2)____ מקדקוד $S_1$ .

**צעד 3 :** החזר את משקל המסלול הקצר המבוקש מקדקוד $S_1$ אל ____(3)____ .

באלגוריתם זה חסרים **שלושה** ביטויים, המסומנים במספרים בין סוגריים עגולים.

התשובה הנכונה עבור כל אחד מהביטויים החסרים מופיעה בסעיפים האלה:

**ט"ז.** הביטוי החסר (1) הוא:

1. $(v_3, u_4), (v_4, u_4)$
2. $(v_3, u_3)$ בלבד
3. $(v_3, u_4), (v_4, u_3)$
4. $(v_3, u_4)$ בלבד

המשך בעמוד 46

**י"ז.** הביטוי החסר (2) הוא:

1. DAG-shortest-paths
2. BFS
3. מיון טופולוגי
4. DFS

**י"ח.** הביטוי החסר (3) הוא:

1. $t_1$
2. $t_4$
3. $t_3$
4. $t_2$

**י"ט.** סיבוכיות זמן הריצה של האלגוריתם הנתון היא:

1. $O(|V| * |E|)$
2. $O(|V| + |E|)$
3. $O(V^3)$
4. $O(E^2)$

המשך בעמוד 47

**כ.** יהי $G = (V, E)$ גרף קשיר ולא מכוון, ויהי $T$ עץ המתקבל מהרצת DFS על הגרף $G$ החל מהקדקוד s כאשר $s \in V$ .

לפניך שתי טענות.

I. **סטודנט A טוען:** עומקו של העץ $T$ הוא לפחות כמו עומקו של כל עץ המתקבל מהרצת BFS על הגרף $G$ החל מהקדקוד $s$ .

II. **סטודנט B טוען:** דרגתו של הקדקוד $s$ בעץ $T$ זהה בהכרח לדרגתו של הקדקוד $s$ בעץ כלשהו המתקבל מהרצת BFS על הגרף $G$ החל מהקדקוד $s$ .

בחר את ההיגד הנכון מבין ארבעת ההיגדים שלהלן:

1. רק סטודנט B צודק.
2. שני הסטודנטים צודקים בטענותיהם.
3. רק סטודנט A צודק.
4. שני הסטודנטים שוגים בטענותיהם.

**הדבק את מדבקת הנבחן במקום המיועד לכך בדף התשובות שבנספח א' והדק אותו למחברת הבחינה.**

**בהצלחה!**

# נספח א': דף תשובות לשאלות 3–4

## לשאלון 714911, אביב תשע"ו

מקום למדבקת נבחן

הדבק את מדבקת הנבחן שלך במקום המיועד לכך, והדק את הדף הזה אל מחברת הבחינה שלך.

הקף בעיגול את הספרה המייצגת את התשובה הנכונה לכל סעיף.

**שאלה 3**

| | | | | |
|---|---|---|---|---|
| סעיף א | 1 | 2 | 3 | 4 |
| סעיף ב | 1 | 2 | 3 | 4 |
| סעיף ג | 1 | 2 | 3 | 4 |
| סעיף ד | 1 | 2 | 3 | 4 |
| סעיף ה | 1 | 2 | 3 | 4 |
| סעיף ו | 1 | 2 | 3 | 4 |
| סעיף ז | 1 | 2 | 3 | 4 |
| סעיף ח | 1 | 2 | 3 | 4 |
| סעיף ט | 1 | 2 | 3 | 4 |
| סעיף י | 1 | 2 | 3 | 4 |
| סעיף י"א | 1 | 2 | 3 | 4 |
| סעיף י"ב | 1 | 2 | 3 | 4 |
| סעיף י"ג | 1 | 2 | 3 | 4 |
| סעיף י"ד | 1 | 2 | 3 | 4 |
| סעיף ט"ו | 1 | 2 | 3 | 4 |
| סעיף ט"ז | 1 | 2 | 3 | 4 |
| סעיף י"ז | 1 | 2 | 3 | 4 |
| סעיף י"ח | 1 | 2 | 3 | 4 |
| סעיף י"ט | 1 | 2 | 3 | 4 |
| סעיף כ | 1 | 2 | 3 | 4 |

**שאלה 4**

| | | | | |
|---|---|---|---|---|
| סעיף א | 1 | 2 | 3 | 4 |
| סעיף ב | 1 | 2 | 3 | 4 |
| סעיף ג | 1 | 2 | 3 | 4 |
| סעיף ד | 1 | 2 | 3 | 4 |
| סעיף ה | 1 | 2 | 3 | 4 |
| סעיף ו | 1 | 2 | 3 | 4 |
| סעיף ז | 1 | 2 | 3 | 4 |
| סעיף ח | 1 | 2 | 3 | 4 |
| סעיף ט | 1 | 2 | 3 | 4 |
| סעיף י | 1 | 2 | 3 | 4 |
| סעיף י"א | 1 | 2 | 3 | 4 |
| סעיף י"ב | 1 | 2 | 3 | 4 |
| סעיף י"ג | 1 | 2 | 3 | 4 |
| סעיף י"ד | 1 | 2 | 3 | 4 |
| סעיף ט"ו | 1 | 2 | 3 | 4 |
| סעיף ט"ז | 1 | 2 | 3 | 4 |
| סעיף י"ז | 1 | 2 | 3 | 4 |
| סעיף י"ח | 1 | 2 | 3 | 4 |
| סעיף י"ט | 1 | 2 | 3 | 4 |
| סעיף כ | 1 | 2 | 3 | 4 |

# נספח ב': מילון מונחים (3 עמודים)

## לשאלון 714911, אביב תשע"ו

| המונח | תרגום המונח | | |
|---|---|---|---|
| | ערבית | רוסית | אנגלית |
| אחזור | استرجاع | Возврат, извлечение | retrieval |
| איבר | مُتَغيِّر/عضو | Элемент | item |
| אציקלי | اسيكليك | Ацикличный | acyclic |
| אקראי | عشوائي | Случайный | random |
| אתחול | قيمة بدائيّة | Инициализация | initialization |
| גמ"ל – גרף מכוון ללא מעגלים | رَسْم مُوَجِّه بدون دوائر | Направленный ацикличный график | connected graph with no circles |
| גרף מכוון קשיר ופשוט | رَسْم مُوَجِّه وبسيط | Направленный, связный и простой график | simple connected graph |
| גרף-על | رسم رئيسيّ | График, получаемый путем стягивания каждой компоненты сильной связности в одну вершину | supergraph |
| דרגת כניסה/יציאה | درجة الدخول / الخروج | Степень вершины (входная, выходная) | in-degree, out-degree |
| הון | أموال | Капитал | fund |
| זמן ריצה | مدّة التنفيذ | Время работы | run time |
| חציון | الوسيط | Медиана | median |
| טבלת ערבול (גיבוב) | جدول الخلط | Хеш-таблица | hash table |
| טיפוס | نوع | Тип | type |
| מונוטוני | منبسط | Монотонный | monotonous |
| מחסנית | باغة | Стек | stack |
| מטריצת סמיכויות | جدول الحدود الزميلة | Матрица смежности | adjacency matrix |
| מיון טופולוגי | تصنيف | Топологическая сортировка | topological sorting |

המשך בעמוד 2

| המונח | תרגום המונח | | |
|---|---|---|---|
| | ערבית | רוסית | אנגלית |
| מסלול | مسار | Путь | path |
| מערך דינמי | مصفوفة غير ثابتة | Динамический массив | dynamic array |
| מצביע | مُؤَشِّر | Указатель | pointer |
| משתנה גלובלי | مُتَغيِّر عامّ | Глобальная переменная | global variable |
| סדרה | سلسلة | Последователь-ность | series |
| סיבוכיות | تعقيد | Сложность (вычислений) | complexity |
| עדיפות | أولويّة | Приоритет | preferential |
| עץ חיפוש בינארי מאוזן | شجرة بحث ثنائيّ متوازنة | сбалансированное двоичное дерево поиска | balanced binary search tree |
| עץ פורש | شجرة الامتداد | Остовное дерево | spanning tree |
| ערך מוחלט | قيمة مُطْلَقة | модуль | absolute value |
| ערמה | كومة | Куча | heap |
| ערמה בינארית | كومة ثنائيّة | Двоичная куча | binary heap |
| פונקציה רקורסיבית | دالّة أو عمليّة تراجعيّة | Рекурсивная функция | recursive function |
| פונקציית משקל | دالّة لقياس الوزن | Весовая функция | weight function |
| צומת | مُفْتَرَق | Узел | node |
| קדקוד | رأس | вершина | vertex |
| קשת | وصلة | Дуга | arc |
| רק"ח – רכיב קשיר חזק | مُرَكِّب مرتبط قويّ | компонента сильной связности | strong connected component |
| רשומה | سِجِلّ | Запись (элемент структуры данных) | record |
| רשימה מקושרת חד־כיוונית מעגלית | سلسلة موصولة باتّجاه واحد بشكل دائريّ | Кольцевой однонаправленный связный список | circular one direction connected linked list |

המשך בעמוד 3

| המונח | תרגום המונח | | |
|---|---|---|---|
| | ערבית | רוסית | אנגלית |
| שגרה | إجراء | Функция, рутина | procedure |
| שדה קישור | حقل رابط | Поле, содержащее ссылку | linking field |
| שורש | جذر | Корень | root |
| תת־עץ | شجرة فرعيّة | Поддерево | sub-tree |