מדינת ישראל
משרד החינוך

סוג הבחינה: גמר לבתי־ספר לטכנאים ולהנדסאים
מועד הבחינה: אביב תשע"ז, 2017
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	60	נקודות
פרק שני	40	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 2, 3 ו־4 יש לרשום **אך ורק** על גבי דף התשובות שבנספח א'. את התשובות לשאלה 1 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 51 עמודים ו־3 עמודי נספחים.

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

בהצלחה!

המשך מעבר לדף ◀

השאלות

<u>פרק ראשון</u> (60 נקודות)

ענה על <u>שתי</u> השאלות 1–2 – שאלות חובה.

שאלה 1 – שאלת חובה (35 נקודות)

בשאלה זו 8 סעיפים (א'–ח'). עליך לענות על <u>כל</u> הסעיפים.

בעקבות ההתחממות הגלובלית, הוחלט להציב חיישנים במקומות שונים על פני כדור־הארץ, ולמדוד באמצעותם את הטמפרטורה במקומות האלה.

מעוניינים לתכנן מערכת ממוחשבת שתאפשר לעקוב אחר מדידות הטמפרטורה של החיישנים המוצבים בקווי הרוחב השונים, וכן להפיק מידע מן הנתונים השמורים בה.

לפניך מושגים הקשורים לכדור־הארץ:

הקוטב הצפוני – הנקודה הצפונית ביותר על פני כדור־הארץ.

הקוטב הדרומי – הנקודה הדרומית ביותר על פני כדור־הארץ.

קו המשווה – קו דמיוני המקיף את כדור־הארץ ונמצא במרחק שווה מן הקוטב הצפוני ומן הקוטב הדרומי.

קו רוחב – קו דמיוני המקיף את כדור־הארץ ומקביל לקו המשווה.

הנחות יסוד:

1. חיישן מזוהה על־ידי מספר ייחודי אי־שלילי (מפתח).
2. קו רוחב מזוהה על־ידי מספר ייחודי אי־שלילי (מפתח).
3. בשאלה זו נתייחס רק לקווי הרוחב הנמצאים בין קו המשווה ובין הקוטב הצפוני, והם יזוהו על־ידי מספרים חיוביים בין 0 ל־90 , כאשר 0 מייצג את קו המשווה ו־90 מייצג את הקוטב הצפוני.
4. טמפרטורה של חיישן נמדדת במעלות צלזיוס.
5. ייתכן קו רוחב שלא מוצב בו אף חיישן.
6. בכל קו רוחב מוצב חיישן אחד לכל היותר.
7. המספר הכולל של החיישנים המוצבים בכל קווי הרוחב יחד הוא לכל היותר NUM_OF_SENSORS .

המשך בעמוד 3

לפניך תיאור של מבנה הנתונים התומך במימוש הפעולות הנדרשות מן המערכת הממוחשבת.

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

שדה 1: sensorsArr – **מערך חיישנים** בגודל M , כאשר M הוא משתנה.

כל תא במערך זה מייצג חיישן ואמור להכיל את מספר החיישן, הטמפרטורה שנמדדה באמצעות החיישן, ומצביע לצומת **בעץ קווי הרוחב,** אשר מכיל את קו הרוחב שבו מוצב חיישן זה.

(הערה: עץ קווי הרוחב יפורט בשדה 2.)

הנח כי : $M \leq NUM\_OF\_SENSORS$.

מערך זה ממומש טבלת ערבול (גיבוב) – Hash table .

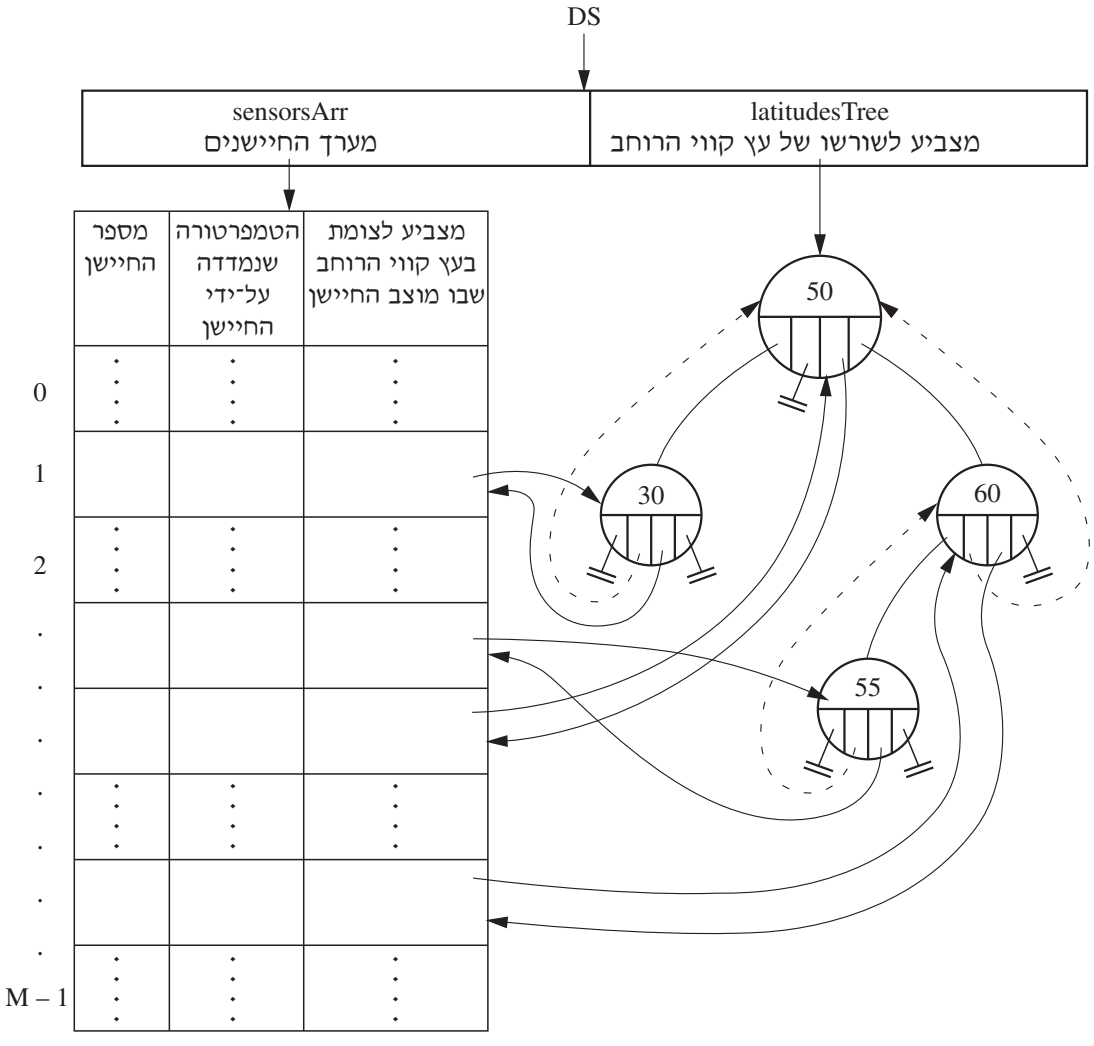
שדה 2: latitudesTree – מצביע לשורשו של **עץ קווי הרוחב,** שהוא **עץ AVL** .

הצמתים בעץ זה מייצגים את קווי הרוחב שבהם מוצבים חיישנים, ולכן בעבור קו רוחב שלא הוצב בו חיישן - לא יהיה קיים כל צומת בעץ.

מפתח החיפוש בעץ זה הוא מספר קו הרוחב.

המשך בעמוד 4

באיור שלהלן מוצג תיאור סכמתי של **"המבנה הראשי"**, המאגד את כל מבני הנתונים שבהם נאחסן את נתוני המערכת הממוחשבת:



איור לשאלה 1

המשך בעמוד 5

להלן הגדרת קבוע בשפת C :

```
#define NUM_OF_SENSORS 1000        // המספר המקסימלי האפשרי של החיישנים
                                   // בכל קווי הרוחב גם יחד
```

ולהלן הגדרת המבנה הראשי בשפת C :

```
typedef struct headType            // טיפוס המבנה הראשי
{
 sensorPtr sensorsArr;             // מערך החיישנים
 latitudePtr latitudesTree;        // מצביע לשורשו של עץ קווי הרוחב
}headder, *headPtr;
```

עתה נפרט את מבנה הנתונים בעבור <u>שדה 1</u> של "המבנה הראשי" – מערך החיישנים.

להלן מבנה של תא במערך החיישנים בשפת C :

```
typedef struct sensorType
{
 int sensorID;                     // מספר החיישן או (-1)
  float tmpr;                      // הטמפרטורה שנמדדה על־ידי החיישן
  struct latitudeType *lptr;       // מצביע לצומת בעץ קווי הרוחב שבו מוצב חיישן זה
 }sensor, *sensorPtr;
```

עתה נפרט את מבנה הנתונים בעבור <u>שדה 2</u> של "המבנה הראשי" – עץ קווי הרוחב.

להלן מבנה של צומת בעץ קווי הרוחב בשפת C :

```
typedef struct latitudeType
{
  int lat;                         // מספר קו הרוחב
  struct latitudeType *left;       // מצביע לתת־העץ השמאלי
  struct latitudeType *parent;     // מצביע להורה של הצומת
  sensorPtr sptr;                  // מצביע לתא במערך החיישנים שבו משוכן חיישן מסוים המוצב
                                   // בקו רוחב זה
  struct latitudeType *right;      // מצביע לתת־העץ הימני
 } latitude ,*latitudePtr
```

המשך בעמוד 6

להלן הגדרות התקפות לכל הסעיפים שיבואו בהמשך:

```
typedef enum {FAILURE, SUCCESS, INVALID_INPUT, ALLOCATION_ERROR,
DUPLICATE_ID} statusType;

typedef enum {FALSE,TRUE} boolean;

typedef enum {GET, PUT} modType;
```

נתונה ספריית פונקציות, המכילה, בין היתר, את הפונקציות שלהלן:

int **hash**(int num, modType mode)	פונקצייה זו מקבלת שני פרמטרים: num – מספר חיישן, ו־mode . **כאשר ערכו של mode הוא PUT** : אם num לא קיים בטבלת הערבול, אז הפונקצייה מחזירה מיקום בטבלת הערבול להכנסת חיישן זה, אחרת - היא מחזירה את הערך (–1) . **הערה:** הנח כי במקרה של הכנסת מספר, תמיד יימצא בעבורו מקום בטבלת הערבול. **כאשר ערכו של mode הוא GET** : הפונקצייה מחפשת בטבלת הערבול את החיישן שמספרו num . אם היא מוצאת אותו, היא מחזירה את מיקומו בטבלה זו, אחרת - היא מחזירה את הערך (–1) .

המשך בעמוד 7

void **insertLatitudesTree**(latitudePtr *pL, latitudePtr t)	הפונקצייה מקבלת שני פרמטרים: pL - מצביע למצביע לשורשו של עץ קווי הרוחב, t - מצביע לצומת (בודד) המייצג קו רוחב מסוים. הפונקצייה מוסיפה את הצומת שעליו מצביע t לעץ קווי הרוחב שעל שורשו מצביע pL . **הערות:** 1. לאחר הוספת קו רוחב, עץ קווי הרוחב יישאר **עץ AVL מיוחד**. 2. פונקצייה זו מעדכנת כנדרש את שדה ה־parent של הצומת שעליו מצביע t .
void **deleteFromLatitudesTree**(latitudePtr *pL, latitudePtr t)	פונקצייה זו מקבלת שני פרמטרים: pL – מצביע לשורשו של עץ קווי הרוחב, t – מצביע לצומת בעץ קווי הרוחב, אשר שעל שורשו מצביע pL . הפונקצייה מוחקת מעץ קווי הרוחב את הצומת שעליו מצביע t . **הערות**: 1. לאחר מחיקת קו הרוחב, עץ קווי הרוחב יישאר **עץ AVL מיוחד**. 2. פונקצייה זו מעדכנת את שדות ה־parent שהשתנו בעקבות המחיקה.
statusType **changeSensorTemp**(int ID, float tmpr)	פונקצייה זו מקבלת שני פרמטרים: ID - מספר חיישן, tmpr - טמפרטורה חדשה שנמדדה באמצעות חיישן זה. אם החיישן שמספרו ID לא נמצא במערך החיישנים, אז הפונקצייה מחזירה את הערך INVALID_INPUT , אחרת - היא מעדכנת את שדה הטמפרטורה של חיישן זה ל־tmpr , ומחזירה את הערך SUCCESS .

המשך בעמוד 8

latitudePtr **findUpper**(latitudePtr pL, int key)	פונקצייה זו מקבלת שני פרמטרים: pL - מצביע לשורשו של עץ קווי הרוחב key - מספר קו רוחב מסוים. אם קיים בעץ, שעל שורשו מצביע pL , צומת שמכיל את קו הרוחב המקסימלי מבין כל קווי הרוחב הקטנים או השווים לקו הרוחב הנתון (key) , אז הפונקצייה מחזירה מצביע לצומת הזה בעץ. אחרת – הפונקצייה מחזירה את הערך NULL . **דוגמה:** בעבור קווי הרוחב: 52, 66, 34, 7, 85, 10, 30 ובעבור key = 40 , הפונקצייה תחזיר מצביע לצומת המכיל את הערך 34 , שהוא קו הרוחב המקסימלי מבין קווי הרוחב הקטנים או השווים ל־40 (34, 7, 10, 30) .

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

להלן הגדרות של משתנים גלובאליים, ופונקצייה היוצרת את מערך החיישנים ומאתחלת אותו:

```
headder head;

headPtr DS = &head;

void init(void)
{
   DS->sensorsArr = malloc(sizeof(sensor) * M);      // M הוגדר בעמוד 3

   for(i = 0; i < M; i++)

   {

       DS->sensorsArr[i].sensorID = -1;

       DS->sensorsArr[i].tmpr = 0;

       DS->sensorsArr[i].lptr = NULL;

   }
}
```

המשך בעמוד 9

ענה על הסעיפים שלהלן:

א. לפניך פונקצייה שכותרתה:

```
statusType addSensor(int ID, int lat, float tmpr)
```

פונקצייה זו מקבלת את הפרמטרים האלה:

ID – מספר חיישן,

lat – מספר קו הרוחב שבו יוצב חיישן זה,

tmpr – טמפרטורה שנמדדה באמצעות החיישן שמספרו ID .

פונקצייה זו אמורה להוסיף את החיישן שמספרו ID למערכת הממוחשבת.

הנח כי: $0 \leq ID < M$.

הפונקצייה מחזירה ערך מטיפוס statusType כמפורט בטבלה שלהלן:

ALLOCATION_ERROR	אם קיימת בעיה בהקצאת זיכרון
DUPLICATE_ID	אם ID כבר קיים בטבלת הערבול
INVALID_INPUT	אם מספר קו הרוחב lat הוא מספר שלילי, או גדול מ־90
SUCCESS	אם החיישן ID נוסף בהצלחה למערכת הממוחשבת

המשך בעמוד 10

בפונקצייה 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.

```
statusType addSensor(int ID, int lat, float tmpr)
{
   statusType status = SUCCESS ;
   latitudePtr L;
   int i;
   if (lat < 0 || lat > 90) return INVALID_INPUT;
   L = malloc(sizeof(latitude));
   if (L == NULL) return ALLOCATION_ERROR;
   i = _____(1)_____;
   if (i == -1) return DUPLICATE_ID;
   DS->sensorsArr[i].sensorID = ID;
   DS->sensorsArr[i].tmpr = tmpr;
   DS->sensorsArr[i].lptr =_____(2)_____;
   L->lat = lat;
   L->left = NULL;
   L->right = NULL;
   L->parent = NULL;
   L->sptr =_____(3)_____;
   _____(4)_____;
   return status;
}
```

המשך בעמוד 11

ב. מהו זמן הריצה הממוצע של הפונקצייה שבסעיף א', כאשר n מציין את מספר החיישנים במערכת הממוחשבת?

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

ג. לפניך פונקצייה שכותרתה:

```
statusType removeSensor(int ID)
```

פונקצייה זו אמורה להסיר את החיישן שמספרו ID מן המערכת הממוחשבת.

הפונקצייה מחזירה ערך מטיפוס statusType כמפורט בטבלה שלהלן:

INVALID_INPUT	אם הפונקצייה hash מחזירה את הערך (1–)
SUCCESS	אם החיישן שמספרו ID הוסר מן המערכת הממוחשבת

בפונקצייה חסרים **שני** ביטויים, המסומנים במספרים בין סוגריים עגולים. רשום במחברת הבחינה את מספרי הביטויים החסרים (1)–(2), בסדר עולה, וכתוב ליד כל מספר את הביטוי החסר שהוא מייצג.

```
statusType removeSensor(int ID)
{
   latitudePtr L;
   int indx;
   statusType status = SUCCESS;
   indx = _____(1)_____;
   if (indx == -1) status = INVALID_INPUT;
   else
   {
    L = DS->sensorsArr[indx].lptr;
    _____(2)_____;
```

המשך בעמוד 12

```
    DS->sensorsArr[indx].sensorID = -1;
    DS->sensorsArr[indx].tmpr = 0 ;
    DS->sensorsArr[indx].lptr = NULL;
   }
   return status;
}
```

ד. לפניך פונקצייה שכותרתה:

```
latitudePtr findLower(latitudePtr pL, int key)
```

פונקצייה זו מקבלת שני פרמטרים:

pL – מצביע לשורשו של עץ קווי הרוחב,

key – מספר קו רוחב כלשהו שלא בהכרח מוצבים בו חיישנים.

אם קיים בעץ, שעל שורשו מצביע pL , צומת שמכיל את קו הרוחב המינימלי מבין כל קווי הרוחב הגדולים או שווים לקו הרוחב הנתון (key) , אז הפונקצייה מחזירה מצביע לצומת הזה בעץ. אחרת - הפונקצייה מחזירה את הערך NULL .

דוגמה: בעבור קווי הרוחב האלה: 72, 61 , 34 , 7 , 85 , 66 , 30 , ובעבור key = 50 , פונקצייה זו תחזיר מצביע לצומת המכיל את הערך 61 , שהוא המינימלי מבין כל קווי הרוחב הגדולים או השווים ל־50 (72 , 61 , 85 , 66) .

המשך בעמוד 13

בפונקצייה 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.

```
latitudePtr findLower(latitudePtr pL, int key)
{
   latitudePtr L, L1;
   int dist, d;
   L = pL;
   L1 = NULL;
   dist = 90;   // המשתנה מאותחל לקו הרוחב שמספרו 90 - הקוטב הצפוני
   while(L)
   {
      d = (L->lat) - key;
      if(_____(1)_____ && d < dist)
      {
         dist = d;
         _____(2)_____;
      }
      L = (L->lat > key)?_____(3)_____ : _____(4)_____;
   }
   return L1;
}
```

המשך בעמוד 14

ה. מהי סיבוכיות זמן הריצה של האלגוריתם המוצג בסעיף ד', כאשר n מציין את מספר קווי הרוחב במערכת הממוחשבת?

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

ו. לפניך פונקצייה שכותרתה :

```
latitudePtr treeSuccesor(latitudePtr p)
```

פונקצייה זו מקבלת את p , שהוא מצביע לצומת כלשהו בעץ **קווי הרוחב**.

הפונקצייה מחזירה מצביע לצומת העוקב, בסדר inorder , לצומת שעליו מצביע p. אם אין לו צומת עוקב – הפונקצייה מחזירה את הערך NULL .

בפונקצייה 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.

```
latitudePtr treeSuccesor(latitudePtr p)
{
 latitudePtr temp;
 if(p == NULL) return NULL;
 if(p->right)
 {
      temp = p->right;
      while(________(1)________) ________(2)________;
      return temp;
  }
 temp = p;
```

המשך בעמוד 15

```
 while((temp->parent) && (_________(3)_________)) // האם הצומת הוא בן ימני

    temp = temp->parent;

 return _________(4)_________;

}
```

1. לפניך פונקצייה שכותרתה:

```
statusType averageTemp(latitudePtr lp, int lat1, int lat2, float
*avr)
```

הפונקצייה מקבלת את הפרמטרים האלה:

- lp – מצביע לשורשו של עץ קווי הרוחב,
- lat1 – מספר קו רוחב מסוים, שלא בהכרח מוצבים בו חיישנים,
- lat2 – מספר קו רוחב מסוים, שלא בהכרח מוצבים בו חיישנים.

הפונקציה מחזירה, באמצעות המשתנה avr , את הטמפרטורה הממוצעת שנמדדה בקווי הרוחב אשר נמצאים בטווח שבין lat1 ו־lat2 (כולל lat1 ו־lat2).

הפונקצייה מחזירה ערך מטיפוס statusType , כמפורט בטבלה שלהלן:

FAILURE	אם לא מוצבים חיישנים בין lat1 ובין lat2 (כולל lat1 ו־lat2)
INVALID_INPUT	אם lat1 או lat2 הוא מספר שלילי או גדול מ־90, או אם lat1 > lat2 , או אם עץ קווי הרוחב ריק
SUCCESS	אם חישוב הטמפרטורה הממוצעת הצליח

המשך בעמוד 16

בפונקצייה 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.

```
statusType averageTemp(latitudePtr lp, int lat1,int lat2,float *avr)
{
   statusType status = SUCCESS;
   latitudePtr L1, L2, L;
   float sum;
   int n;
   if(lat1 < 0 || lat2 < 0 || lp == NULL || lat1 > lat2 || lat2 > 90
   || lat1 > 90)
         status = INVALID_INPUT;
   else
  {
      n = 0;
      sum = 0;
      L1 = _________(1)_________;
      if(L1 == NULL)
            status = FAILURE;
      else
         if(L1->lat > lat2)
               status = FAILURE;
         else
          {
            L2 = ___________(2)__________;
            L = L1;
```

המשך בעמוד 17

```
            while(L != L2)
            {
               sum += L->sptr->tmpr;
               n = n+1;
               L =_____________(3)_____________;
            }
            _____________(4)_____________;
            _____________(5)_____________;
            *avr = sum / n;
         }
   }
return status;
}
```

ח. מהי סיבוכיות זמן הריצה של האלגוריתם המוצג בסעיף ז', כאשר n מציין את מספר קווי הרוחב במערכת הממוחשבת?

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

המשך בעמוד 18

שאלה 2 – שאלת חובה (25 נקודות)

בשאלה זו 10 סעיפים (א'–י'). עליך לענות על <u>כל</u> הסעיפים.

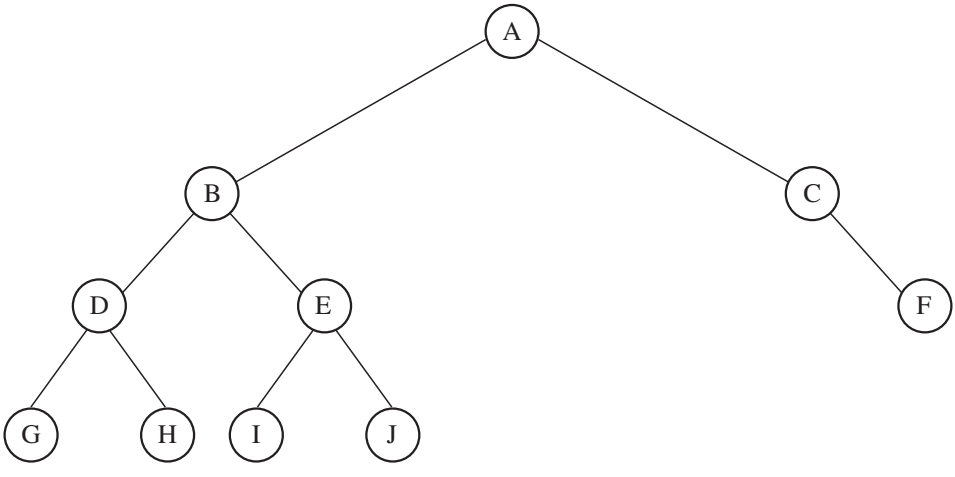
<u>נגדיר</u>:

עץ בינארי למהדרין הוא עץ בינארי, אשר הדרגה של כל צומת בו היא או 0 או 2, כלומר כל צומת בו הוא או עלה או שיש לו בדיוק שני בנים.

עץ בינארי כמעט שלם (almost complete binary tree) הוא עץ בינארי למהדרין, שעבורו קיים מספר שלם אי־שלילי, k, כך שיתקיימו שני התנאים האלה:

1. כל העלים בעץ הם ברמה ה־k או ברמה ה־k+1 .
2. אם לצומת בעץ יש צאצא ימני שהוא עלה ברמה ה־k+1 – אז כל העלים מבין צאצאיו השמאליים, שהם גם עלים, יהיו גם הם ברמה ה־k+1 .

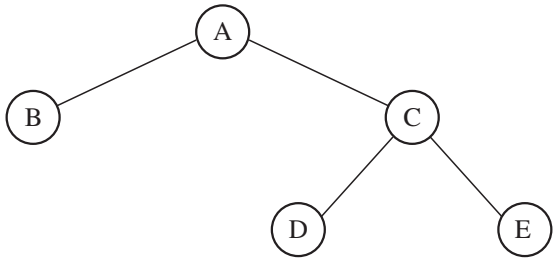
באיור א' שלהלן מוצג עץ בינארי שאינו עץ בינארי למהדרין, משום שדרגתו של הצומת C היא 1 , ולכן הוא גם לא עץ בינארי כמעט שלם.



איור א' לשאלה 2

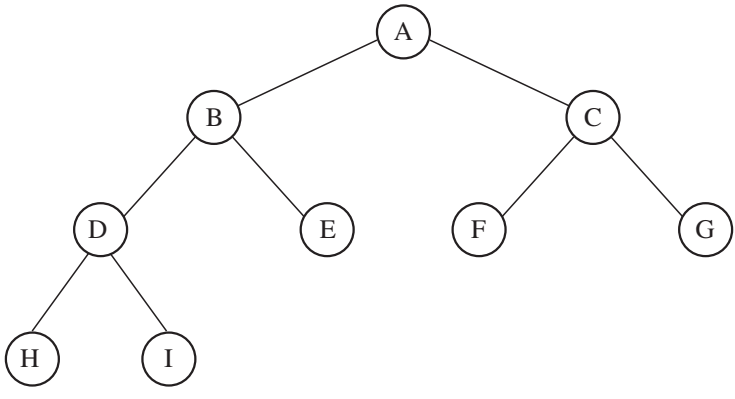
המשך בעמוד 18

באיור ב' שלהלן מוצג עץ בינארי למהדרין אשר מקיים את תנאי 1 , משום שכל עליו הם או ברמה 1 (k = 1) או ברמה 2 (k+1 = 2) . עם זאת, בעץ הזה לא מתקיים תנאי 2 משום שלצומת A יש צאצא ימני, שהוא עלה ברמה 2 (למשל הצומת D) , אך יש לו גם צאצא שמאלי, שהוא עלה ברמה 1 (הצומת B) . לפיכך, העץ המוצג באיור ב' אינו עץ בינארי כמעט שלם.



איור ב' לשאלה 2

באיור ג' שלהלן מוצג עץ בינארי למהדרין, המקיים את התנאים 1 ו־2 , ולכן הוא גם עץ בינארי כמעט שלם.

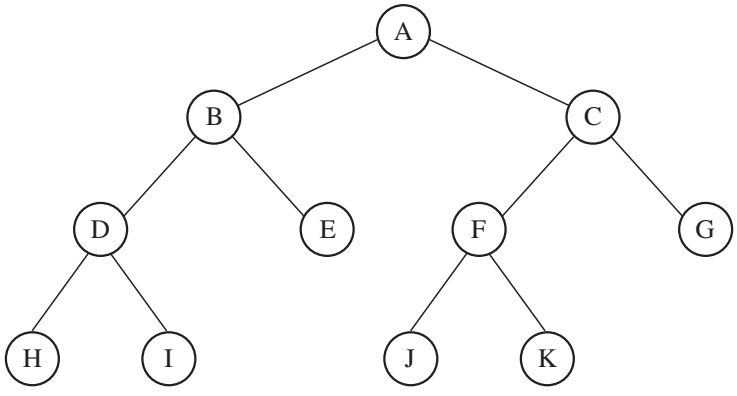


איור ג' לשאלה 2

המשך בעמוד 20

ענה על הסעיפים א' ו־ב' שלהלן:

א. נתון העץ הבינארי הזה:



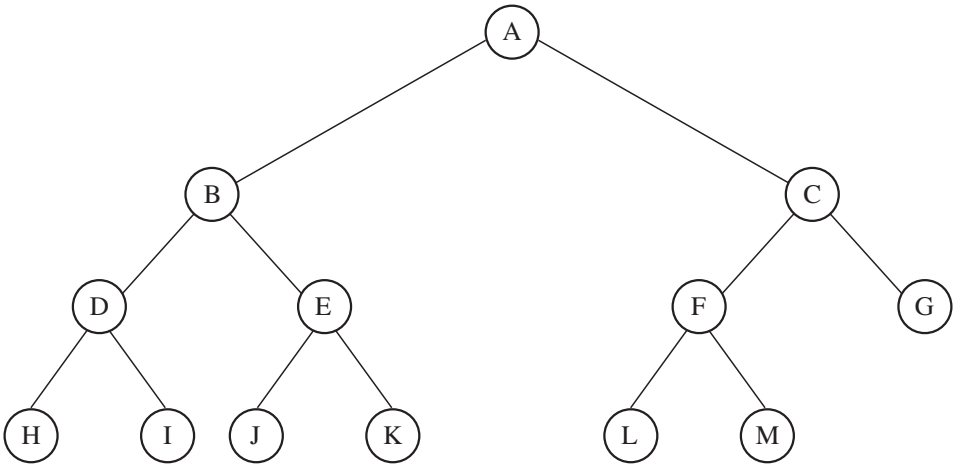
איור ד' לשאלה 2

בחר את ההיגד הנכון מבין ההיגדים שלהלן:

1. העץ הנתון הוא לא עץ בינארי למהדרין
2. העץ הנתון הוא עץ בינארי כמעט שלם
3. העץ הנתון הוא לא עץ בינארי כמעט שלם, כיוון שתנאי 1 לא מתקיים בו
4. העץ הנתון הוא לא עץ בינארי כמעט שלם, כיוון שתנאי 2 לא מתקיים בו

המשך בעמוד 21

ב. נתון העץ הבינארי הזה:



איור ה' לשאלה 2

בחר את ההיגד הנכון מבין ההיגדים שלהלן:

1. העץ הנתון הוא לא עץ בינארי כמעט שלם, כיוון שתנאי 1 לא מתקיים בו
2. העץ הנתון הוא לא עץ בינארי כמעט שלם, כיוון שתנאי 2 לא מתקיים בו
3. העץ הנתון הוא עץ בינארי כמעט שלם
4. העץ הנתון הוא לא עץ בינארי למהדרין

המשך בעמוד 22

להלן ייצוג של עץ בינארי כמעט שלם באמצעות **מערך**:

נשכֵּן את העץ הבינארי הכמעט שלם במערך, כדלקמן:

- את שורש העץ נמקם במערך באינדקס 1 .
- את יתר צומתי העץ נשכֵּן במערך באופן הזה:

עבור כל צומת בעץ, שישוכן במערך באינדקס i $(i > 0)$:

בנו השמאלי ישוכן במערך באינדקס 2*i , ובנו הימני ישוכן במערך באינדקס 2*i+1 .

איור ו' שלהלן מתאר את תמונת המערך שבו משוכן העץ הבינארי הכמעט שלם המוצג באיור ג'.

0	
1	A
2	B
3	C
4	D
5	E
6	F
7	G
8	H
9	I

איור ו' לשאלה 2

להלן כמה טענות שהן נכונות עבור עץ בינארי כמעט שלם המשוכן במערך החל מאינדקס 1 . כמו כן, אין צורך להשתמש בשדות father , left או right .

1. בעבור הצמתים המשוכנים במערך באינדקסים 2*i ו־2*i+1, אביהם ישוכן באינדקס ה־i .
2. שורשו של עץ בינארי כמעט שלם נמצא באינדקס 1 במערך.
3. אם לעץ בינארי כמעט שלם יש m עלים, אזי יש בו 2*m–1 צמתים.
4. הצומת השמאלי ביותר של הרמה ה־i בעץ משוכן במערך באינדקס ה־2^i .

המשך בעמוד 23

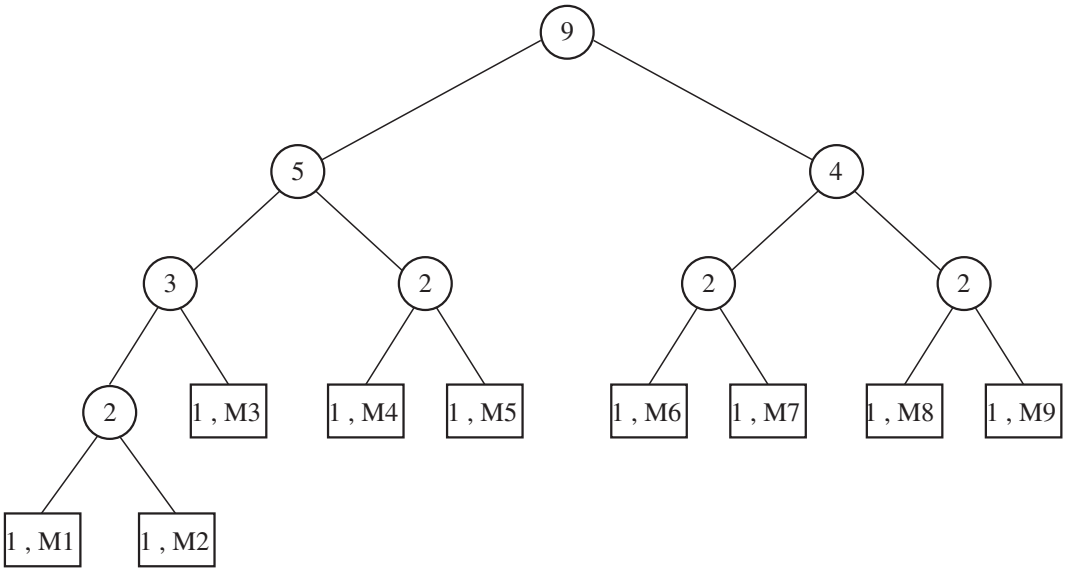
בסעיפים ג'–י' התייחס לבעיה שלהלן:

יש לכתוב תכנית אשר קולטת רשימה של שמות חיילים העומדים במעגל בסדר נתון, החל מחייל מסוים שנקבע כראשון במעגל, ועד החייל האחרון במעגל. הקלט מסתיים במחרוזת end , המציינת את סופו. קטע התכנית משכן את שמות החיילים בעלים של **עץ בינארי כמעט שלם** המיוצג באמצעות מערך, כמתואר לעיל, באופן הזה:

יהי TOTAL מספר החיילים. לעץ הבינארי הכמעט שלם יהיו TOTAL עלים המייצגים את החיילים, ולכן יהיו לו 2*TOTAL–1 צמתים.

כל **צומת** בעץ מכיל מספר שלם, שהוא כמות העלים שיש בתת־העץ ששורשו הוא הצומת הזה. נוסף על כך, כל **עלה** מכיל את שם החייל שהוא מייצג.

איור ז' שלהלן מתאר את העץ עבור TOTAL = 9 . העלים מוצגים כריבועים, וצמתים שאינם עלים – כעיגולים. שמות החיילים מסומנים מ־M1 עד M9 , כאשר N1 הוא החייל הראשון במעגל, ו־M9 הוא החייל האחרון במעגל. נכנה את העץ הזה בשם: **R_ Tree** .



איור ז' לשאלה 2

המשך בעמוד 24

להלן הגדרות של קבועים והצהרות של משתני עזר בשפת C:

```
#define TOTAL 9          //מספר העלים בעץ
#define MAXNODES 17      // סה"כ מספר הצמתים בעץ שהוא 2*TOTAL–1
int i, p, q , twotomax ;
int remain ;
```

להלן הגדרת מבנה של תא במערך שבו משוכן העץ. כל תא מייצג צומת בעץ, בשפת C :

```
typedef  struct  nodeType
{
   char name[20];       //עבור צומת שהוא עלה – שדה זה יכיל את שם החייל
                        //אחרת – שדה זה יכיל רווח
   int count;           //מספר העלים בתת־העץ ששורשו הוא הצומת הזה
 } node,*nodeptr;
```

להלן הצהרה של המערך שבו נשכן את העץ הבינארי הכמעט שלם:

```
node tree[MAXNODES+1];
```

לפניך קטע קוד שבונה את העץ R_Tree .

סעיפים ג'–ו' שלאחריו מתייחסים לקטע הקוד הזה.

קטע הקוד הזה קולט שמות של חיילים ומשכן אותם בעץ **R_Tree** , המיוצג באמצעות מערך, כמתואר באיור ו'.

שים לב: העץ שבאיור ז' נבנה בעבור תשעה חיילים, ואילו קטע הקוד יבנה את העץ **R_Tree** בעבור TOTAL חיילים.

```
   twotomax = 1;
   while (twotomax < TOTAL) twotomax *= 2;
   printf("\n");
```

המשך בעמוד 25

```
for(i = twotomax; _____(1)_____; i++)
{
   printf("\nEnter name: ");
   scanf("%s",tree[i].name);
   tree[i].count = 1;
}
for(i = _____(2)_____; i < twotomax; i++)
{
   printf("\nEnter name: ");
   scanf("%s",tree[i].name);
   tree[i].count = 1;
}
for(_____(3)_____)
{
   tree[i].count = tree[2*i].count + tree[2*i+1].count;
}
```

בקטע הקוד חסרים **שלושה** ביטויים, המסומנים במספרים בין סוגריים עגולים. בכל אחד מן הסעיפים שלהלן, בחר את הביטוי החסר מבין ארבע האפשרויות הנתונות, והקף בעיגול את הספרה המייצגת אותו בדף התשובות שבנספח א'.

ג. הביטוי החסר (1) הוא:

1. i <= 2*TOTAL
2. i <= 2*TOTAL–1
3. i <= TOTAL
4. i <= twotomax–TOTAL

המשך בעמוד 26

ד. הביטוי החסר (2) הוא:

1. TOTAL
2. 2*TOTAL
3. 2*TOTAL+1
4. twotomax–TOTAL

ה. הביטוי החסר (3) הוא:

1. i = TOTAL–1; i>0; i– –
2. i = 1; i <=TOTAL; i++
3. i = twotomax–1; i>0; i– –
4. i = 2*TOTAL–1; i > TOTAL; i– –

ו. מהי סיבוכיות זמן הריצה של קטע הקוד הבונה את העץ R_Tree כפונקצייה של m , כאשר m מציין את מספר העלים בעץ?

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

אחרי שבנינו את העץ הבינארי הכמעט שלם, והחיילים משוכנים בעלים שלו, יש לכתוב אלגוריתם המקבל כקלט מספר שלם וחיובי – n , ומדפיס את שם החייל הנמצא במעגל במקום ה־n־י (החל מן החייל הראשון במעגל). לשם כך, יש לרדת בעץ, החל מן הצומת שהוא שורש העץ, לשם מציאת העלה שבו משוכן שמו של החייל המבוקש.

להלן נתאר את אופן פעולת האלגוריתם בעבור n = 13 ו־ TOTAL = 9 (ראה איור ח' בהתאם):

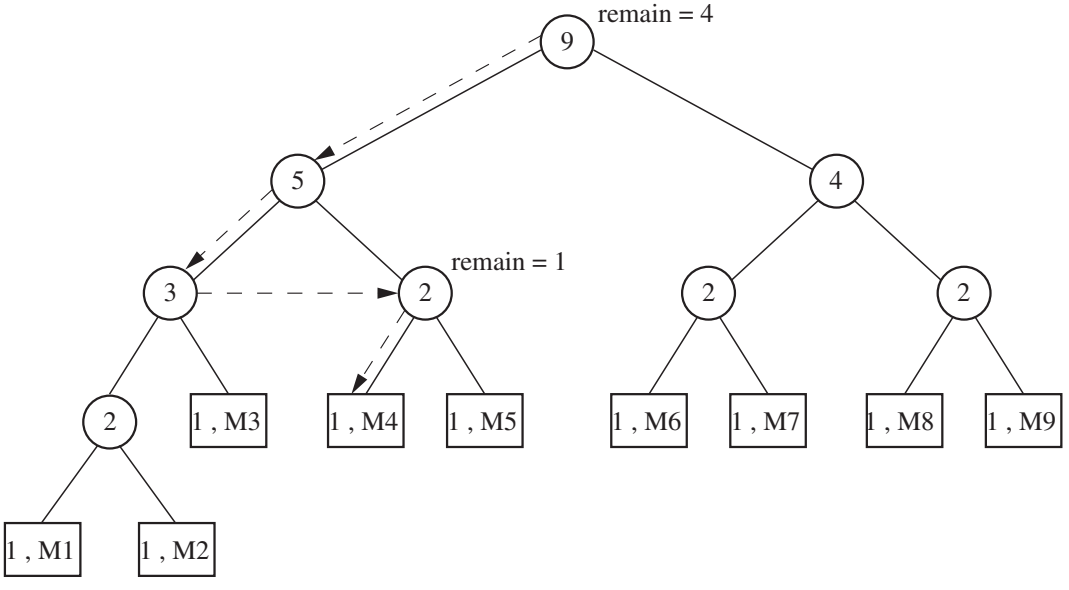
- מאחר שערכו של n הוא 13 , וגדול מערכו של המונה שבשורש העץ, שהוא 9 , עלינו למצוא את החייל **הרביעי** מתחילת המעגל (שארית החלוקה של 13 ב־9).
- נרד לבנו השמאלי של השורש. מאחר שערכו של המונה של הבן השמאלי הוא 5 , והוא גדול מן המספר שנותר למנות, שהוא 4 , אזי החייל המבוקש מצוי בהכרח בתת־העץ הזה.

המשך בעמוד 27

- נרד שוב לבנו השמאלי של הצומת הזה. ערך המונה שבו הוא 3 , והוא קטן מן המספר שיש למנות, שהוא 4 . פירוש הדבר: החייל המבוקש לא מצוי בתת־העץ השמאלי של הצומת שבו ערך המונה הוא 5 , אלא בתת־העץ הימני של הצומת שבו ערך המונה הוא 5 .
- נפחית מן המספר שנותר למנות, שהוא 4 , את מספר החיילים המיוצגים על־ידי תת־העץ השמאלי, שהוא 3 , ונעבור לתת־העץ הימני של הצומת שבו ערך המונה הוא 5 .

  נמנו אפוא שלושה חיילים, ונותר למנות עוד חייל אחד.
- נרד לבן השמאלי, M4 , שהוא העלה המייצג את החייל שאת שמו יש להדפיס.

הקווים המקווקווים שבאיור ח' להלן מייצגים את המסלול למציאת העלה M4 **, שתואר לעיל.**



איור ח' לשאלה 2

קטע הקוד שלהלן מממש את האלגוריתם שתואר לעיל, ומתבסס על העץ **R_Tree** שנבנה לעיל. קטע הקוד מדפיס את שם החייל המבוקש.

בקטע הקוד חסרים **שלושה** ביטויים, המסומנים במספרים בין סוגריים עגולים. בכל אחד מן הסעיפים שלהלן, בחר את הביטוי החסר מבין ארבע האפשרויות הנתונות, והקף בעיגול את הספרה המייצגת אותו בדף התשובות שבנספח א'.

המשך בעמוד 28

להלן קטע הקוד :

```
p = 1;
remain = (n-1)%tree[p].count + 1;
while(_____(1)_____)
  {
      p = _____(2)_____;
      if (remain > tree[p].count)
        {
           remain - = tree[p].count;
           _____(3)_____;
         }
  }
printf("%s\t",tree[p].name);
```

ז. הביטוי החסר (1) הוא:

1. tree[p].count > remain
2. tree[p].count > 1
3. remain > 1
4. tree > 1

ח. הביטוי החסר (2) הוא:

1. p/2
2. p*2+1
3. p*2
4. p++

המשך בעמוד 29

ט. הביטוי החסר (3) הוא:

1. p/2
2. p*2+1
3. p++
4. p*2

י. מהי סיבוכיות זמן הריצה של קטע הקוד כפונקצייה של m , כאשר m מציין את מספר העלים בעץ?

1. $\Theta(\log m)$
2. $\Theta\left(\frac{m}{\log m}\right)$
3. $\Theta(m \log m)$
4. $\Theta(m)$

המשך בעמוד 30

פרק שני (40 נקודות)

ענה על שאלה אחת מבין השאלות 3–4 .

שאלה 3 (40 נקודות)

בשאלה זו 21 סעיפים שאינם תלויים זה בזה. 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. $T(n) = 8T(n/2) + 27n^3$ היא פונקציית זמן הריצה של אלגוריתם מסוים, הפועל על קלט שגודלו n . מהי סיבוכיות זמן הריצה של האלגוריתם?

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

ב. $T(n) = 7T(n/7) + 49n\log^2 n$ היא פונקציית זמן הריצה של אלגוריתם מסוים, הפועל על קלט שגודלו n . מהי סיבוכיות זמן הריצה של האלגוריתם?

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

ג. $T(n) = 8T(n-1)-16T(n-2) + 4^n \cdot 7 + (5n-7)\cdot 3^n + n^2$ היא פונקציית זמן הריצה של אלגוריתם מסוים, הפועל על קלט שגודלו n . מהי סיבוכיות זמן הריצה של האלגוריתם?

1. $\Theta(4^n)$
2. $\Theta(n^2 4^n)$
3. $\Theta(n4^n)$
4. $\Theta(n^3 \cdot 12^n)$

המשך בעמוד 31

ד. $T(n) = 2T\left(3\sqrt{\log_3 n}\right) + \log_2 \log_3 n$ היא פונקציית זמן הריצה של אלגוריתם מסוים, הפועל על קלט שגודלו n . מהי סיבוכיות זמן הריצה של האלגוריתם?

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

ה. $$T(n) = 4^n + 12\cdot\sum_{i=1}^{n-2} T(i)$$
$$T(1) = 1$$

היא פונקציית זמן הריצה של אלגוריתם מסוים, הפועל על קלט שגודלו n . מהי סיבוכיות זמן הריצה של האלגוריתם?

1. $\Theta(n4^n)$
2. $\Theta(4^n)$
3. $\Theta(n^2)$
4. $\Theta(n^2 4^n)$

המשך בעמוד 32

1. לפניך קטע קוד:

```
a = 2 ;
while (a <= n)
   {
      for (k = 1; k <= n;  k++)
       {
          b = n ;
          while (b > 1)
             {
                 S ;
                 b = b / 2 ;
                 y = 1;
                 z = 0;
                 while (y < n)
                    {
                       y = y + n / 7;
                       for (j = 1; j <= n; j++) z++;
                    }
             }
       }
      a = a * a * a ;
   }
```

נוסף על קטע הקוד, נתון כי:

- S הוא משפט פשוט, והזמן הדרוש לביצועו הוא $\Theta(1)$
- S אינו משנה את הערכים של: k, a, y, z, j , n ו־b
- כל המשתנים בקטע הקוד הם מטיפוס שלם

המשך בעמוד 33

מהי סיבוכיות זמן הריצה של קטע הקוד הנתון כפונקצייה של n ?

שים לב: $log^2 n = log\ n \cdot log\ n$

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

2. לפניך קטע קוד:

```
for(i = 2; i <= n; i++)
{
      x = 2;
      while(x < i)
      {
            x = x*2;
      }
      while(x > 2)
      {
            x = sqrt(x);
      }
}
```

הנח כי כל המשתנים בקטע הם מטיפוס שלם.

מהי סיבוכיות זמן הריצה של קטע הקוד הנתון כפונקצייה של n ?

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

המשך בעמוד 34

בסעיפים ח'–י' התייחס לבעיה שלהלן:

לפניך טענה ולאחריה נתונה ההוכחה של הטענה.

נתונות שתי פונקציות מונוטוניות עולות f ו־g.

הטענה היא: **אם** החסם התחתון של $g(n)$ הוא קבוע, **וגם** $f(g(n)) = O(n)$, **וגם** $f(n) = \Omega(n)$, **אזי** $g(n) = O(n)$.

הוכחה:

קיימים קבועים $c1$, $n1$ כך שלכל $n \geq n1$ מתקיים: $f(g(n)) \leq c1n$.

נוסף על כך, קיימים קבועים $c2$, $n2$, כך שלכל $n \geq n2$ מתקיים $f(n) \geq c2n$.

נבחר ____(1)____ = $n0$, ואז לכל $n \geq n0$ מתקיים: $c1n \leq f(g(n)) \leq$ _____ (2) _____ .

מסקנה:

לכל $n \geq n0$ מתקיים _____ (3) _____ - מש"ל.

בהוכחה זו חסרים **שלושה** ביטויים, המסומנים במספרים בין סוגריים עגולים. בכל אחד מן הסעיפים שלהלן, בחר את הביטוי החסר מבין ארבע האפשרויות הנתונות, והקף בעיגול את הספרה המייצגת אותו בדף התשובות שבנספח א'.

ח. הביטוי החסר (1) הוא:

1. $\min\{n1, n2\}$
2. $\min\{c1, n1, c2, n2\}$
3. $\max\{n1, n2\}$
4. $\max\{c1, n1, c2, n2\}$

המשך בעמוד 35

ט. הביטוי החסר (2) הוא:

1. $c_2 n$
2. $c_2 \cdot \frac{f(n)}{g(n)}$
3. $c_2 g(n)$
4. $c_2 log(g(n))$

י. הביטוי החסר (3) הוא:

1. $g(n) \leq (c1 * c2)\, n$
2. $g(n) \leq c1\, n$
3. $g(n) \leq (c2 / c1)\, n$
4. $g(n) \leq (c1 / c2)\, n$

י"א. יש למיין את הפונקציות שלהלן על־פי הסיבוכיות שלהן:

$$n,\ n!,\ 2^n,\ n^n,\ 2^{log_3 n},\ n^{\frac{2n+1}{n+1}}$$

ויש למצוא סידור g1 , g2 , g3 , g4 , g5 , g6 כך ש־

$$g1 = \Omega(g2),\ g2 = \Omega(g3),\ g3 = \Omega(g4),\ g4 = \Omega(g5),\ g5 = \Omega(g6)$$

1. $g_1(n) = n^n,\ g_2(n) = n!,\ g_3(n) = 2^n,\ g_4(n) = n^{\frac{2n+1}{n+1}},\ g_5(n) = n,\ g_6(n) = 2^{log_3 n}$
2. $g_1(n) = n^n,\ g_2(n) = n^{\frac{2n+1}{n+1}},\ g_3(n) = n!,\ g_4(n) = 2^n,\ g_5(n) = 2^{log_3 n},\ g_6(n) = n$
3. $g_1(n) = n^n,\ g_2(n) = n!,\ g_3(n) = 2^n,\ g_4(n) = 2^{log_3 n},\ g_5(n) = n^{\frac{2n+1}{n+1}},\ g_6(n) = n$
4. $g_1(n) = n!,\ g_2(n) = n^n,\ g_3(n) = n^{\frac{2n+1}{n+1}},\ g_4(n) = 2^n,\ g_5(n) = n,\ g_6(n) = 2^{log_3 n}$

המשך בעמוד 36

בסעיפים י"ב–י"ז הנך רשאי להשתמש בפונקציות שלהלן:

שם הפונקצייה	תיאור הפונקצייה	סיבוכיות זמן הריצה של הפונקצייה במקרה הגרוע ביותר
Sort(S)	פונקצייה הממיינת את איברי הקבוצה S , ומחזירה את הקבוצה S בצורה ממוינת, בסדר עולה.	$O(n \log n)$
Select(S, k)	פונקצייה 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)	פונקצייה המעתיקה את הסדרה S , בת n האיברים, לשתי סדרות, S1 ו־S2 , כך שאיבריה יחולקו באופן הזה: לסדרה S1 יועתקו האיברים הקטנים מ־z או השווים לו, ולסדרה S2 יועתקו האיברים הגדולים מ־z . **שים לב:** הסדרות S1 ו־S2 אינן בהכרח ממוינות.	$O(n)$

הנח שהפונקציות האלו כתובות, וכי ניתן להשתמש בהן בכל הסעיפים הבאים בלי לכתוב אותן מחדש.

בסעיפים י"ב–י"ז התייחס לבעיה שלהלן:

נתונה קבוצה סופית S , המכילה n מספרים שלמים חיוביים, שונים זה מזה, ואיבריה משוכנים במערך A , החל באינדקס 1 וכלה באינדקס ה־n־י (כולל).

נוסף על כך, נתון מספר k , כך ש: $1 \le k \le n$.

המשך בעמוד 37

הנחה: כל הפעולות המתמטיות, כגון חיבור, כפל והשוואה, ניתנות לביצוע בזמן $O(1)$.

לפניך אלגוריתם **יעיל** בשם TR2 , אשר מחזיר את הערך "no" , אם קיימת בקבוצה S תת־קבוצה כלשהי T , המכילה בדיוק k מספרים, וסכום איבריה של T יהיה קטן מ־k^3 ; אחרת – אם בעבור **כל** תת־קבוצה T של S מתקיים: $|T| = k \,,\ \sum_{t \in T} t \geq k^3$ – האלגוריתם יחזיר את הערך "yes" .

שים לב:

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

האלגוריתם:

TR2(A, n)

צעד 1: הצב במשתנה x את הערך ___(1)___ .

צעד 2: Partition(A, ____(2)____ , B, C) .

צעד 3: Sum ← 0 .

צעד 4: עבור i , המקבל ערכים החל מ־1 ועד ___(3)___ (כולל), בצע:

4.1: Sum ← Sum +____(4)____ .

צעד 5: אם ___(5)___ אזי עצור והחזר "no" .

צעד 6: החזר "yes" .

באלגוריתם הנתון חסרים **חמישה** ביטויים, המסומנים במספרים בין סוגריים עגולים. בכל אחד מן הסעיפים שלהלן, בחר את הביטוי החסר מבין ארבע האפשרויות הנתונות, והקף בעיגול את הספרה המייצגת אותו בדף התשובות שבנספח א'.

י"ב. הביטוי החסר (1) הוא:

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

המשך בעמוד 38

י"ג. הביטוי החסר (2) הוא:

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

י"ד. הביטוי החסר (3) הוא:

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

ט"ו. הביטוי החסר (4) הוא:

1. $\text{select}(A, i)$
2. $\text{select}\left(A, \left\lceil \frac{i+1}{2} \right\rceil\right)$
3. $A[i]$
4. $A[\text{select}(A, i)]$

ט"ז. הביטוי החסר (5) הוא:

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

המשך בעמוד 39

י"ז. מהי סיבוכיות זמן הריצה של האלגוריתם TR2 ? (בחר את החסם ההדוק ביותר.)

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

סעיפים י"ח–כ"א מתייחסים לפונקציות שלהלן:

הפונקצייה **Extract_increasing** שלהלן מקבלת מחסנית בשם S_from , אשר מכילה את האיברים $a_1 , a_2 , a_3, ... a_i , a_{i+1} , ... , a_n$, באופן בו האיבר a_1 נמצא בראש המחסנית. הפונקצייה מעתיקה למחסנית אחרת בשם S_to את האיברים $a_1 , a_2 , a_3 , ... a_i$, המהווים סדרה עולה, כלומר $a_1 \le a_2 \le a_3 \le .... \le a_i$.
(a_{i+1} לא מועתק למחסנית S_to , מכיוון ש- $a_i > a_{i+1}$.)

אם כל איברי המחסנית S_from מהווים סדרה עולה, אז כל איבריה יועתקו למחסנית S_to .

```
Extract_increasing(S_from, S_to)
{
     CreateEmptyStack(S_to);       // ריקון המחסנית S_to שהתקבלה כפרמטר
     if not IsEmpty(S_from)
      {
          Push(S_to, pop(S_from));
          while(______(1)______)
           {
                Push(S_to, pop(S_from));
           }
       }
     return (S_to);
}
```

המשך בעמוד 40

נתונה מחסנית Input , שבה איברים מטיפוס שלם (int) , ונוסף על כך, נתונות שלוש מחסניות ריקות: S1 , S2 ו־S3 .

הנחה: האיברים שנמצאים במחסנית Input אינם ממוינים.

הפונקצייה **Sort** שלהלן ממיינת את האיברים הנמצאים במחסנית Input , כך שהסדרה הממוינת תימצא במחסנית S1 (האיבר הקטן ביותר יימצא בתחתית המחסנית S1) .

שים לב: פונקצייה זו משתמשת בפונקצייה **Extract_increasing** שהוצגה בעמוד הקודם.

```
Stack Sort(Stack Input)
{
     CreateEmptyStack(S1);
     CreateEmptyStack(S2);
     CreateEmptyStack(S3);
     Extract_Increasing(Input, S1);
     while(not IsEmpty(Input)
     {
         Extract_Increasing(Input, S2);
         while(______(2)______)
         {
            /*בשלב זה המטרה היא מיזוג האיברים שבמחסניות S1 ו־S2 למחסנית S3 , כך שבתחתית S3
                                    יימצא האיבר הגדול מבין האיברים שבמחסניות S1 ו־S2  */
              if (______(3)______)  push(S3, pop(S1));
               else   push(S3, pop(S2);
          }
          ______(4)______ ;
          while (not IsEmpty(S2)) push(S3, pop(S2));
          if(not IsEmpty(S3)) push(S1, pop(S3));
      }
    return S1;
}
```

המשך בעמוד 41

בשתי הפונקציות הללו חסרים **ארבעה** ביטויים, המסומנים במספרים בין סוגריים עגולים. בכל אחד מן הסעיפים שלהלן, בחר את הביטוי החסר מבין ארבע האפשרויות הנתונות, והקף בעיגול את הספרה המייצגת אותו בדף התשובות שבנספח א'.

י"ח. הביטוי החסר (1) הוא:

1. not IsEmpty(S_from) || StackTop(S_to) <= StackTop(S_From)
2. not IsEmpty(S_from) && StackTop(S_to) <= StackTop(S_From)
3. not IsEmpty(S_from) || StackTop(S_to) >= StackTop(S_from)
4. not IsEmpty(S_from) && StackTop(S_to) >= StackTop(S_from)

י"ט. הביטוי החסר (2) הוא:

1. not IsEmpty(S2) && not IsEmpty(S1)
2. not IsEmpty(Input)
3. StackTop(S1) >= StackTop(S2)
4. not IsEmpty(S1) || not IsEmpty(S2)

כ. הביטוי החסר (3) הוא:

1. StackTop(S1) <= StackTop(S2)
2. StackTop(S1) >= StackTop(S2)
3. not IsEmpty(S1) && not IsEmpty(S2)
4. Pop(S1) <= StackTop(S2)

כ"א. הביטוי החסר (4) הוא:

1. while(not IsEmpty(S1)) push(S2, pop(S1))
2. while(not IsEmpty(S3)) push(S1, pop(S3))
3. while(not IsEmpty(S1)) push(S3, pop(S1))
4. while(not IsEmpty(S2)) push(S1, pop(S2))

המשך בעמוד 42

שאלה 4 (40 נקודות)

בשאלה זו 21 סעיפים שאינם תלויים זה בזה. 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'.

סעיפים א'–י' מתייחסים לבעיה שלהלן:

יש להציע מבנה נתונים שיענה על דרישותיו של לקוח שבבעלותו מפעל.

מבנה הנתונים ישמור מידע על המוצרים שבמפעל ועל איכותם, כאשר לכל מוצר יש מספר זהות ייחודי (תחום מספרי הזהות אינו חסום), ואיכות המוצרים היא מספר שלם בין 0 ל־5 .

נוסף על כך, מבנה הנתונים יתמוך בפעולות האלה:

שם הפעולה	תיאור הפעולה
Init()	אתחול מבנה נתונים ריק
Insert(id, quality)	הכנסת מוצר בעל מספר זיהוי ייחודי id ואיכות quality למבנה הנתונים
Delete(id)	מחיקת מוצר בעל מספר זיהוי ייחודי id ממבנה הנתונים, אם הוא נמצא בו
MedianQuality( )	מציאת האיכות החציונית של המוצרים במבנה הנתונים החציון מוגדר כאיבר ה־$\left\lceil \frac{n}{2} \right\rceil$ **בגודלו** **שימו לב:** על הפעולה להחזיר את ערך **האיכות החציונית** ולא את האיבר בעל איכות זו
AvgQuality( )	מציאת האיכות הממוצעת של המוצרים במבנה הנתונים
JunkWorst( )	מחיקת כל האיברים שאיכותם היא 0 ממבנה הנתונים הפעולה מחזירה עץ שמכיל את כל האיברים שנמחקו **הערה:** הפעולה אינה מטפלת בשחרור זיכרון
Printout(i)	הדפסת האיברים שאיכותם היא i בסדר עולה, על פי מספר ה־id שלהם

המשך בעמוד 43

סטודנט הציע את מבנה הנתונים שלהלן כפתרון **יעיל** לביצוע הפעולות שתוארו לעיל.

המבנה כולל:

- מערך A בגודל 6 , שכל תא בו מצביע על שורשו של עץ AVL , כאשר התא A[i] מצביע על שורשו של עץ AVL שבו משוכנים רק המוצרים שאיכותם היא i לכל $0 \le i \le 5$.

  מפתח החיפוש בעץ כזה הוא id של המוצר. כל מוצר חייב להופיע באחד משבעת העצים הללו.

- num0 – יכיל את כמות המוצרים שאיכותם היא 0 .
- num1_5 - יכיל את כמות המוצרים שאיכותם שונה מ־0 .
- sumall - יכיל את סכום האיכויות של כל המוצרים שבמבנה הנתונים.
- מערך Q בגודל 6 , שבו התא ה־i־י יכיל את כמות המוצרים שנמצאים במבנה הנתונים, אשר איכותם קטנה או שווה i .

דוגמה: אם במבנה הנתונים משוכנים המוצרים האלה:

id	1000	1111	1112	2222	4444	4445
quality	0	1	1	2	4	4

אז MedianQuality() יחזיר 1 , ו־AvgQuality() יחזיר 2 , שהוא ממוצע כל האיכויות.

כמו כן, המערך Q ייראה כך:

quality	0	1	2	3	4	5
Q[i]-Q[i–1]	1	3	4	4	6	6

ניתן לראות, כי עבור האיכות i , כאשר i>0 , כמות המוצרים המופיעים במבנה נתונים מאיכות זו היא: Q[i]-Q[i–1] .
לדוגמה: אין כלל מוצרים שאיכותם היא 3 , ואכן עבור $i = 3 : Q[3]–Q[2] = 4–4 = 0$.

המשך בעמוד 44

על סמך מבני הנתונים האלו, ענה על השאלות שלהלן:

בכל אחד מן הסעיפים שלהלן, בחר את התשובה הנכונה מבין ארבע האפשרויות הנתונות, והקף בעיגול את הספרה המייצגת אותה בדף התשובות שבנספח א'.

א. מהי סיבוכיות זמן הריצה של הפעולה Insert(id, quality) ? (בחר את החסם ההדוק ביותר.)

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

ב. שני תלמידים הציעו מימושים שונים לפעולה Insert(id, quality) .

ההצעה של תלמיד א': יש לבצע הכנסת id בעץ ה־AVL , אשר על שורשו מצביע A[quality] , בלבד.

ההצעה של תלמיד ב': יש לבצע הכנסת id בעץ ה־AVL , אשר על שורשו מצביע A[quality] , ונוסף על כך, יש לעדכן את המשתנים num0 , num1_5 ו־sumall , בהתאם לערך שבמשתנה quality .

איזו מבין שתי ההצעות הללו מממשת את הפעולה Insert(id, quality) בשלמותה?

1. הצעתו של תלמיד א' בלבד
2. הצעתו של תלמיד ב' בלבד
3. אף אחת מבין שתי ההצעות
4. שתי ההצעות. (התוספת שמציע תלמיד ב' אינה תורמת דבר ולכן הן שקולות)

ג. מהי סיבוכיות זמן הריצה של הפעולה Delete(id) ? (בחר את החסם ההדוק ביותר.)

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

המשך בעמוד 45

ד. מהי סיבוכיות זמן הריצה של הפעולה MedianQuality( ) ? (בחר את החסם ההדוק ביותר.)

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

ה. מהי סיבוכיות זמן הריצה של הפעולה AvgQuality( ) ? (בחר את החסם ההדוק ביותר.)

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

ו. מהי סיבוכיות זמן הריצה של הפעולה JunkWorst( ) ? (בחר את החסם ההדוק ביותר.)

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

ז. מהי סיבוכיות זמן הריצה של הפעולה Printout(i) ? (בחר את החסם ההדוק ביותר.)

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

המשך בעמוד 46

בסעיפים **ח'–י'** התייחס למימוש הפעולה MedianQuality( ) שלהלן:

```
step 1 : p= _____(1)_____

step 2 :  for (i=0;  i<=5;  i++)

                if (_____(2)_____) return _____(3)_____ ;
```

במימוש זה חסרים **שלושה** ביטויים, המסומנים במספרים בין סוגריים עגולים. בכל אחד מן הסעיפים שלהלן, בחר את הביטוי החסר מבין ארבע האפשרויות הנתונות, והקף בעיגול את הספרה המייצגת אותו בדף התשובות שבנספח א'.

ח. הביטוי החסר (1) הוא:

1. $\left\lfloor \frac{num0 + 4 \cdot numl\_5}{5} \right\rfloor$
2. $\left\lfloor \frac{num0 + numl\_5}{2} \right\rfloor$
3. מצביע לעץ שמכיל את המפתח בעל האיכות החציונית
4. $A\left[\left\lfloor \frac{num0 + numl\_5}{2} \right\rfloor\right]$

ט. הביטוי החסר (2) הוא:

1. $A[i] < p$
2. $A[i] >= p$
3. $Q[i] \geq p$
4. $Q\left[\left\lfloor \frac{num0 + numl\_5}{2} \right\rfloor\right] \geq p$

י. הביטוי החסר (3) הוא:

1. $A[i]$
2. $Q\left[\left\lfloor \frac{num0 + numl\_5}{2} \right\rfloor\right]$
3. i
4. $Q[i]$

המשך בעמוד 47

בסעיפים י"א–י"ג התייחס לבעיה שלהלן:

נתון מערך A המכיל n מספרים שלמים. בהינתן מספר x המופיע ב־A , השכיחות של x במערך A תהיה מספר המופעים של x במערך.

יש ליצור מערך **ממוין** B בגודל n , שיכיל רק את השכיחות של כל אחד מהמספרים המופיעים ב־A .

לפניך אלגוריתם שפותר את הבעיה בתוחלת זמן הריצה הטובה ביותר.

אלגוריתם:

צעד 1: עבור כל איבר $y \in A$, בצע:

1.1 הכנס את y לתוך מבנה הנתונים ____ (1)____ .

צעד 2: העתק את שכיחות האיברים המשוכנים בתוך מבנה נתונים ____ (1)____ למערך B .

צעד 3: מיין את המערך B על־ידי ____(2)____ .

באלגוריתם הזה חסרים **שני** ביטויים, המסומנים במספרים בין סוגריים עגולים. בכל אחד מן הסעיפים שלהלן, בחר את הביטוי החסר מבין ארבע האפשרויות הנתונות, והקף בעיגול את הספרה המייצגת אותו בדף התשובות שבנספח א'.

י"א. הביטוי החסר (1) הוא:

1. עץ חיפוש בינארי.
ערכים כפולים נשמור באותו צומת בעץ על־ידי מונה. כלומר, כל צומת יכיל ערך נתון במערך ואת שכיחותו (מספר המופעים שלו במערך). בעץ חיפוש בינארי זה לא יהיו שני צמתים **שונים** שיכילו את אותו הערך.

2. טבלת Hash בגודל n (פתרון התנגשויות הוא על־ידי רשימות מקושרות).
ערכים כפולים נשמור באותו איבר ברשימה על־ידי מונה. כלומר, כל צומת ברשימה יכיל ערך נתון במערך ואת שכיחותו (מספר המופעים שלו במערך). ברשימה זו לא יהיו שני צמתים **שונים** שיכילו את אותו הערך.

3. רשימה דו־כיוונית ממוינת.
ערכים כפולים נשמור באותו צומת ברשימה על־ידי מונה. כלומר, כל צומת ברשימה הזו יכיל ערך נתון במערך ואת שכיחותו (מספר המופעים שלו במערך). ברשימה זו לא יהיו שני צמתים **שונים** שיכילו את אותו הערך.

4. עץ מאוזן AVL .
ערכים כפולים נשמור באותו צומת בעץ על־ידי מונה. כלומר, כל צומת יכיל ערך נתון במערך ואת שכיחותו (מספר המופעים שלו במערך). בעץ המאוזן (AVL) לא יהיו שני צמתים **שונים** שיכילו את אותו הערך.

המשך בעמוד 48

י"ב. הביטוי החסר (2) הוא:

1. מיון מהיר (quick sort)
2. מיון מנייה (counting sort)
3. מיון מיזוג (merge sort)
4. מיון עֶרמה (heap sort)

י"ג. תוחלת זמן הריצה של האלגוריתם הנתון היא:

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

בסעיפים י"ד–ט"ז התייחס לבעיה שלהלן:

סטודנט A סיפר לסטודנט B , שמצא אלגוריתם מיוחד שמקבל ערמה בת n איברים, וממיר אותה לעץ מאוזן AVL בזמן ריצה $O(n)$.

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

ההוכחה:

נניח בשלילה שקיים אלגוריתם כפי שמציע סטודנט A .

נפעיל את האלגוריתם וניצור עץ AVL .

עתה נבצע _____(1)_____ .

סיבוכיות זמן הריצה שקיבלנו עד כה היא: _____(2)_____ .

לאור זאת, לטענתו של הסטודנט B , _____(3)_____ .

המשך בעמוד 49

בהוכחה זו חסרים **שלושה** ביטויים, המסומנים במספרים בין סוגריים עגולים. בכל אחד מן הסעיפים שלהלן, בחר את הביטוי החסר מבין ארבע האפשרויות הנתונות, והקף בעיגול את הספרה המייצגת אותו בדף התשובות שבנספח א'.

י"ד. הביטוי החסר (1) הוא:

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

ט"ו. הביטוי החסר (2) הוא: (בחר את החסם ההדוק ביותר.)

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

ט"ז. 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 הנו $O(n)$, והרי אין אפשרות למיין במודל השוואות בזמן ריצה הקטן מ־$O(n \log n)$.

המשך בעמוד 50

בסעיפים י"ז–כ"א התייחס לבעיה שלהלן:

יהי $G = (V, E)$ גרף מכוון, ויהי $u \in V$ קדקוד כלשהו בגרף.

לפניך אלגוריתם **יעיל**, המדפיס את אורך המעגל הקטן ביותר, מבין כל המעגלים שאורכם הוא אי־זוגי, אשר הקדקוד u משתתף בו. אם לא קיים מעגל כזה – האלגוריתם ייתן הודעה על כך.

האלגוריתם

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

$V^* = V \cup V' \qquad V' = \{v' \mid v \in V\}$

$E^* = \{\_\_\_(1)\_\_\_ \mid (u \to v) \in E\}$

צעד 2: הרץ על הגרף G^* את אלגוריתם _____(2)_____ החל מן הקדקוד u ,

_____ (3)_____.

צעד 3: אם האלגוריתם _____(2)_____ הסתיים ולא הגעת ל _____(4)_____ ,

– הודע כי לא קיים מעגל כנדרש.

באלגוריתם הזה חסרים **ארבעה** ביטויים, המסומנים במספרים בין סוגריים עגולים. בכל אחד מן הסעיפים שלהלן, בחר את הביטוי החסר מבין ארבע האפשרויות הנתונות, והקף בעיגול את הספרה המייצגת אותו בדף התשובות שבנספח א'.

י"ז. הביטוי החסר (1) הוא:

1. $(u \to v'),(u' \to v)$
2. $(u' \to v')$
3. $(v \to u')$
4. $(u \to v'),(v' \to u)$

י"ח. הביטוי החסר (2) הוא:

1. DFS
2. BFS
3. מיון טופולוגי
4. בלמן־פורד

המשך בעמוד 51

י"ט. הביטוי החסר (3) הוא:

1. החזר את d[u] , וסיים
2. אם הגעת לקדקוד u במהלך ההרצה, אזי החזר את d[u] וסיים
3. אם הגעת לקדקוד 'u במהלך ההרצה, אזי החזר את d['u] וסיים
4. החזר את d['u] , וסיים

כ'. הביטוי החסר (4) הוא:

1. קדקוד שכן לקדקוד u
2. קדקוד שכן לקדקוד 'u
3. קדקוד u
4. קדקוד 'u

כ"א. סיבוכיות זמן הריצה של האלגוריתם הנתון היא:

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

בהצלחה!

נספח א': דף תשובות לשאלות 2, 3 ו־4

לשאלון 714911, אביב תשע"ז

מקום למדבקת נבחן

הדבק את מדבקת הנבחן שלך במקום המיועד לכך, והדק את הדף הזה למחברת הבחינה שלך.
הקף בעיגול את הספרה המייצגת את התשובה הנכונה לכל סעיף.

שאלה 2

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

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
סעיף כ"א	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
סעיף כ"א	1	2	3	4

נספח ב': מילון מונחים (2 עמודים)

לשאלון 714911, אביב תשע"ז

המונח	תרגום המונח		
	ערבית	רוסית	אנגלית
אחזור	استرجاع	Возврат, извлечение	retrieval
איבר	مُتَغيِّر/عضو	Элемент	item
אציקלי	اسيكليك	Ациклический	acyclic
אקראי	عشوائي	Случайный	random
אתחול	قيمة بدائيّة	Инициализация	initialization
דרגת כניסה/יציאה	درجة الدخول/ الخروج	Степень вершины (входная, выходная)	in-degree, out-degree
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
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	Последовательность	series
סיבוכיות	تعقيد	Сложность (вычислений)	complexity
עדיפות	أولويّة	Приоритет	preference

המשך בעמוד 2

המונח	תרגום המונח		
	ערבית	רוסית	אנגלית
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
עֵרמה	كومة	Куча	heap
עֵרמה 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	مُرَكَّب مرتبط قويّ	компонента сильной связности	strong connected component
רשומה	سِجِلّ	Запись (элемент структуры данных)	record
שדה קישור	حقل رابط	Поле, содержащее ссылку	linking field
שורש	جذر	Корень	root
תת-עץ	شجرة فرعيّة	Поддерево	sub-tree

המשך בעמוד 2