23.5.2016

מבני נתונים
פתרון תרגיל מס׳ 6

.1

כתוב/כתבי פסאודו-קוד של פונקציה בשם **P1**, יעילה ככל האפשר, אשר מקבלת כפרמטר עץ בינארי **T** ומדפיסה עבור כל צומת **x** בעץ את המפתח של הצומת ולאחריו מספר שמציין את מספר הצמתים שנמצאים במסלול שמחבר בין צומת **x** לצומת בעל המפתח הקטן ביותר ביותר בתת העץ השמאלי של **x**, דהינו בתת העץ של **T** ששורשו **left(x)** (שמסומן כ- $T_{left(x)}$).

אין חשיבות לסדר ההדפסה של הצמתים בפלט.

נתח/י את סיבוכיות זמן הריצה של הפונקציה שכתבת כתלות במספר הצמתים בעץ **n**.

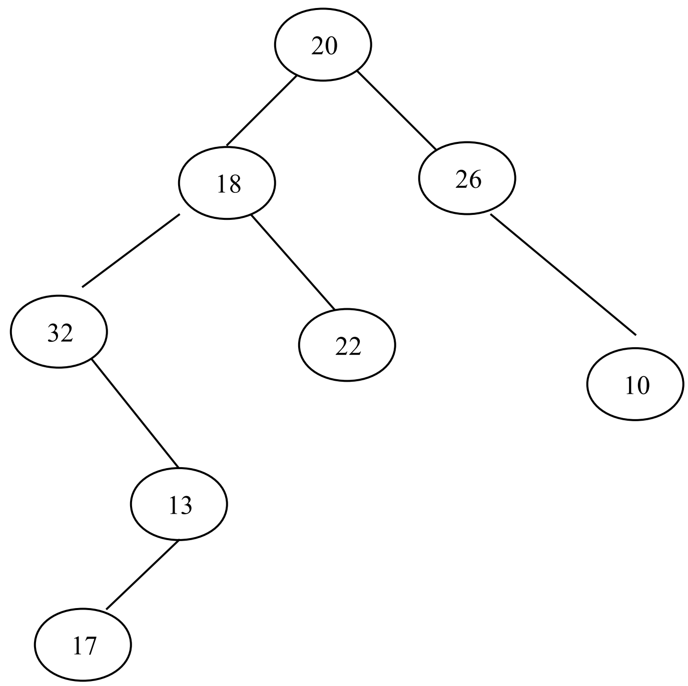
הנחות ודרישות:

- אין להשתמש במבני עזר נוספים.
- מותר להשתמש במספר קבוע של משתנים (כמו למשל **x,y,z**).
- מותר להשתמש בפונקציות עזר, אך יש לכתוב את הפסיאודו-קוד של פונקציות העזר.
- כל צומת **x** בעץ מכיל את השדות הרגילים של עץ בינארי כפי שהוגדר בכיתה.

ראה/י דוגמה בעמוד הבא.

<u>דוגמה:</u>

יהי T עץ בינארי שמתואר בציור הבא :



לאחר הקריאה לפונקציה P1(T) יתקבל הפלט:

20 4 18 3 26 0 32 0 13 2 17 0 22 0 10 0

הסבר לפלט: עבור צומת 20 הודפס המספר 4 כי המסלול בין 20 ל- 13 מכיל 4 צמתים. עבור צומת 26 הודפס המספר 0 כי ל- 26 אין בן שמאלי, וכן הלאה...

<u>פתרון</u>

```
P1(T)
x=root(T)
if (x==null) {return}
y.min=key(x)
if (left(x)==null && right(x)==null) {
     y.distance_to_min=1;
     print key(x), 0
     return y
   }
if (left(x)==null) {
      y2=P1(T_right(x))
      y.min=min{key(x),y2.min};
      y.distance_to_min=y2.distance_to_min+1
      if key(x)< y2.min { y.distance_to_min=1 }
      print key(x),0
      return y
    }
if (right(x)==null) {
      y1=P1(T_left(x))
      y.min=min{key(x),y1.min};
      y.distance_to_min=y1.distance_to_min+1
      if key(x)< y1.min { y.distance_to_min=1 }
      print key(x),y1.distance_to_min+1
      return y
   }
y1=P1(T_left(x))
y2=P1(T_right(x))
y.min=min{key(x),y1.min,y2.min}
if (y.min==key(x)} {y.distance_to_min=1}
if (y.min==y1.min} {y.distance_to_min=y1.distance_to_min+1}
if (y.min==y2.min} {y.distance_to_min=y2.distance_to_min+1}
print key(x),y1.distance_to_min+1; return y
```

2.

<u>הגדרה</u>: נגדיר <u>עץ בינארי עם צבעים אדום-לבן</u>, כעץ בינארי שבו לכל צומת x יש בנוסף לשדות הרגילים, שדה color(x) ששווה ל- red אם הצומת אדום או white אם הצומת לבן.

<u>הגדרה:</u> עבור צומת x בעץ בינארי T נגדיר את <u>תת העץ השמאלי של x</u> כתת העץ של T שמורכב מהבן השמאלי של x וכל הצמתים שמתחתיו. במילים אחרות, תת העץ השמאלי של x שווה ל- $T_{left(x)}$ וכולל גם את left(x). באופן דומה נגדיר את <u>תת העץ הימני של x</u> כתת העץ של T שמורכב מהבן הימני של x וכל הצמתים שמתחתיו.

<u>הגדרה</u>: נגדיר שצומת x בעץ בינארי עם צבעים אדום-לבן T הוא צומת <u>טוב</u> אם לצומת יש צאצא אדום בתת העץ השמאלי שלו וצאצא לבן בתת העץ הימני שלו והמפתח של הצאצא האדום המקסמאלי בתת העץ השמאלי שלו גדול יותר מהמפתח של הצאצא הלבן המקסימאלי בתת עץ הימני שלו.

כתוב/כתבי פסאודו-קוד של פונקציה בשם P2, <u>יעילה ככל האפשר</u>, אשר מקבלת כפרמטר עץ בינארי עם צבעים אדום-לבן T ומדפיסה עבור כל צומת טוב x בעץ T את המפתח של הצומת ולאחריו מספר שמציין את מספר הצמתים במסלול שמחבר בין הצומת האדום המקסימאלי בתת העץ השמאלי של x לצומת הלבן המקסימאלי בתת העץ הימני של x.

אין חשיבות לסדר ההדפסה של הצמתים בפלט.

נתח/י את סיבוכיות זמן הריצה של הפונקציה שכתבת כתלות במספר הצמתים בעץ n.

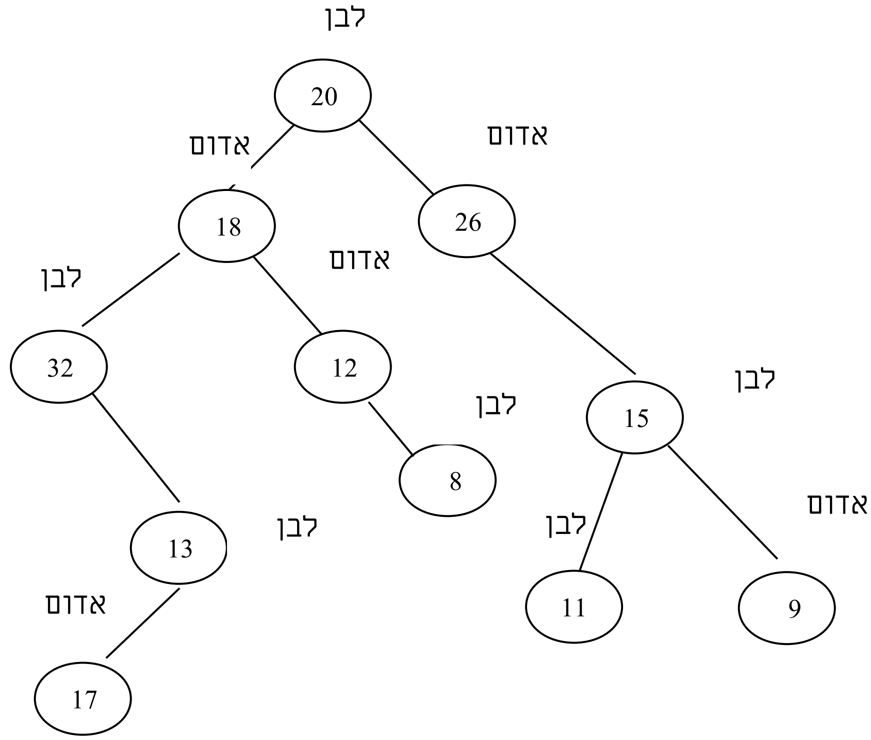
<u>הנחות ודרישות:</u>

- אין להשתמש במבני עזר נוספים.
- מותר להשתמש במספר קבוע של משתנים (כמו למשל x,y,z).
- מותר להשתמש בפונקציות עזר, אך יש לכתוב את הפסיאודו-קוד של פונקציות העזר.
- כל צומת x בעץ מכיל את השדות הרגילים של עץ בינארי כפי שהוגדר בכיתה ובנוסף מכיל את השדה color(x) שתואר לעיל.

ראה/י דוגמה בעמוד הבא.

<u>דוגמה:</u>

יהי T עץ בינארי עם צבעים אדום-לבן שמתואר בציור הבא :



לאחר הקריאה לפונקציה P2(T) יתקבל הפלט:

20 4 18 6

<u>הסבר לפלט</u>: עבור צומת 20 הצאצא האדום המקסימאלי בתת העץ השמאלי שלו הוא 18 והצאצא הלבן המקסימאלי בתת העץ הימני שלו הוא 15 ומאחר ו- 18 גדול מ- 15 הצומת 20 הוא צומת טוב ומודפס המפתח שלו ולאחר מכן המספר 4 שמתאר את מספר הצמתים שנמצאים במסלול שמחבר את צומת 18 לצומת 15.

עבור צומת 18 הצאצא האדום המקסימאלי בתת העץ השמאלי שלו הוא 17 והצאצא הלבן המקסימאלי בתת העץ הימני שלו הוא 8 ומאחר ו 17 גדול מ- 8 הצומת 18 הוא צומת טוב ומודפס המפתח שלו והמספר 6 כי מספר הצמתים שנמצאים במסלול שמחבר את צומת 17 לצומת 8 הוא 6.

לגבי שאר הצמתים הם אינם טובים כי לכל אחד מהם או שאין צאצא אדום בתת העץ השמאלי או שאין צאצא לבן בתת העץ הימני.

פתרון

P2(T)

```
x=root(T)
if (x==null) {y.distance_to_max_red=0
               y.distance_to_max_white=0
               y.max_red=0; y.max_white=0
               return y}
y1=P2(T_left(x))
y2=P2(T_right(x))
if (color(x)==red) {
    y.max_red=key(x)
    y.distance_to_max_red=1
    if (y1.distance_to_max_red!=0 && key(x)< y1.max_red){
        y.max_red=y1.max_red
        y.distance_to_max_red=y1.distance_to_max_red+1
       }
   if (y2.distance_to_max_red!=0 && y.max_red < y2.max_red){
        y.max_red=y2.max_red
        y.distance_to_max_red=y2.distance_to_max_red+1
       }
   y.distance_to_max_white=0
   if (y1.distance_to_max_white!=0){
     y.max_white=y1.max_white
     y.distance_to_max_white=y1.distance_to_max_white+1
   }
   if (y2.distance_to_max_white!=0){
    if (y1.distance_to_max_white==0 || y2.max_white > y1.max_white){
          y.max_white=y2.max_white
          y.distance_to_max_white=y2.distance_to_max_white+1
       }
     }
   if (y.distance_to_max_white==0) {y.max_white=0}
}
```

```
if (color(x)==white) {
    y.max_white=key(x)
    y.distance_to_max_white=1
    if (y1.distance_to_max_white!=0 && key(x)< y1.max_white){
        y.max_white=y1.max_white
        y.distance_to_max_white=y1.distance_to_max_white+1
       }
    if (y2.distance_to_max_white!=0 && y.max < y2.max_white){
        y.max_white=y2.max_white
        y.distance_to_max_white=y2.distance_to_max_white+1
       }
   y.distance_to_max_red=0
   if (y1.distance_to_max_red!=0){
     y.max_red=y1.max_red
     y.distance_to_max_red=y1.distance_to_max_red+1
   }
   if (y2.distance_to_max_red!=0){
    if (y1.distance_to_max_red==0 || y2.max_red > y1.max_red){
          y.max_red=y2.max_red
          y.distance_to_max_red=y2.distance_to_max_red+1
       }
     }
   if (y.distance_to_max_red==0) {y.max_red=0}
}
if (y1.distance_to_max_red!=0 &&
    y2.distance_to_max_white!=0 &&
    y1.max_red > y2.max_white ) {
      print key(x),
        y1.distance_to_max_red+y2.distance_to_max_white+1
    }
return y
```

3.

לצורך שאלה זו יש להניח שמותר להשתמש בפונקציות העזר הבאות:

Create(L) - יוצרת רשימה מקושרת ריקה
Insert(L,x) - מוסיפה איבר **x** לרשימה מקושרת **L**
Print(L) - מדפיסה את כל המפתחות של כל האיברים שנמצאים ברשימה מקושרת **L**
Merge(L,L_1,L_2) יוצרת רשימה מקושרת חדשה **L** שמכילה את השרשור של הרשימות מקושרות L_1 ו- L_2.

נניח שסיבוכיות הזמן של קריאה לכל אחת מפונקציות העזר הנ"ל היא O(1).

<u>הגדרה</u>: נגדיר <u>עץ בינארי עם צבעים אדום-לבן</u>, כעץ בינארי שבו לכל צומת **x** יש בנוסף לשדות הרגילים, שדה**color(x)** ששווה ל- **red** אם הצומת אדום או **white** אם הצומת לבן.

כתוב/כתבי פסאודו-קוד של פונקציה בשם **P3**, <u>יעילה ככל האפשר</u>, אשר מקבלת כפרמטר עץ בינארי עם צבעים אדום-לבן **T** ומדפיסה את מספר הצמתים שנמצאים במסלול הארוך ביותר שמחבר שני צמתים לבנים כלשהם בעץ. בנוסף למספר הצמתים שבמסלול, הפונקציה מדפיסה גם את רשימת כל הצמתים שבמסלול.

אין חשיבות לסדר ההדפסה של הצמתים בפלט.

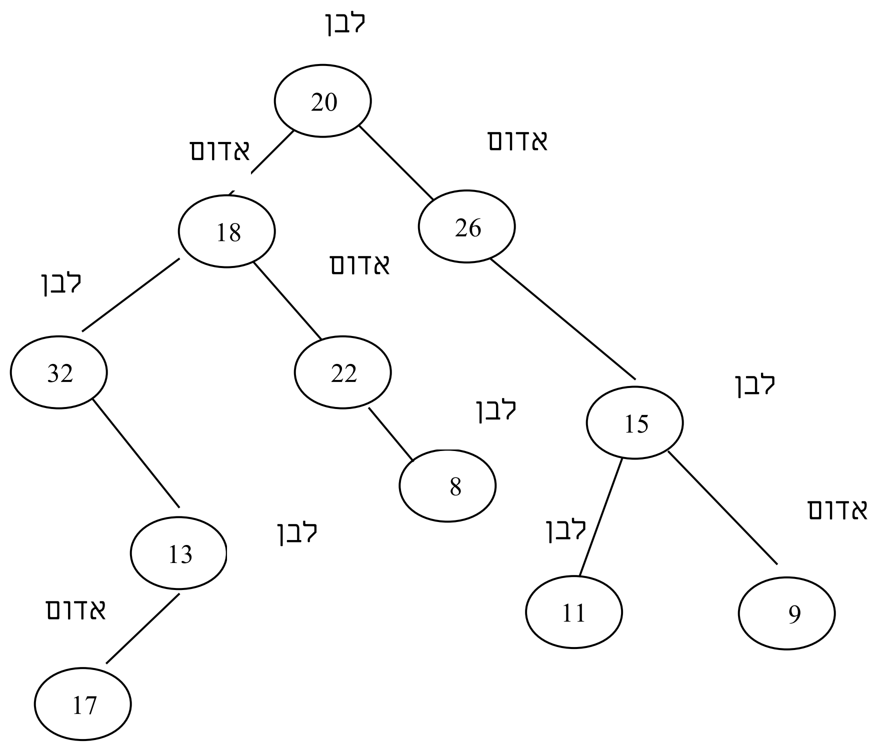
נתח/י את סיבוכיות זמן הריצה של הפונקציה שכתבת כתלות במספר הצמתים בעץ n.

<u>הנחות ודרישות:</u>
- אין להשתמש במבני עזר נוספים.
- מותר להשתמש במספר קבוע של משתנים (כמו למשל **x,y,z**).
- מותר להשתמש בפונקציות עזר , אך (פרט לפונקציות העזר שתוארו בתחילת השאלה) יש לכתוב את הפסיאודו-קוד של פונקציות העזר.
- כל צומת **x** בעץ מכיל את השדות הרגילים של עץ בינארי כפי שהוגדר בכיתה ובנוסף מכיל את השדה **color(x)** שתואר לעיל.

<u>הדרכה:</u>

כדי להדפיס את הצמתים שבמסלול יש ליצור רשימה מקושרת שמכילה את כל הצמתים שבמסלול ולהדפיס אותה, בעזרת פונקציות העזר שמתוארות למעלה.

דוגמה 1:
יהי T עץ בינארי עם צבעים אדום-לבן שמתואר בציור הבא :



לאחר הקריאה לפונקציה P3(T) יתקבל הפלט:

7 13 32 18 20 26 15 11

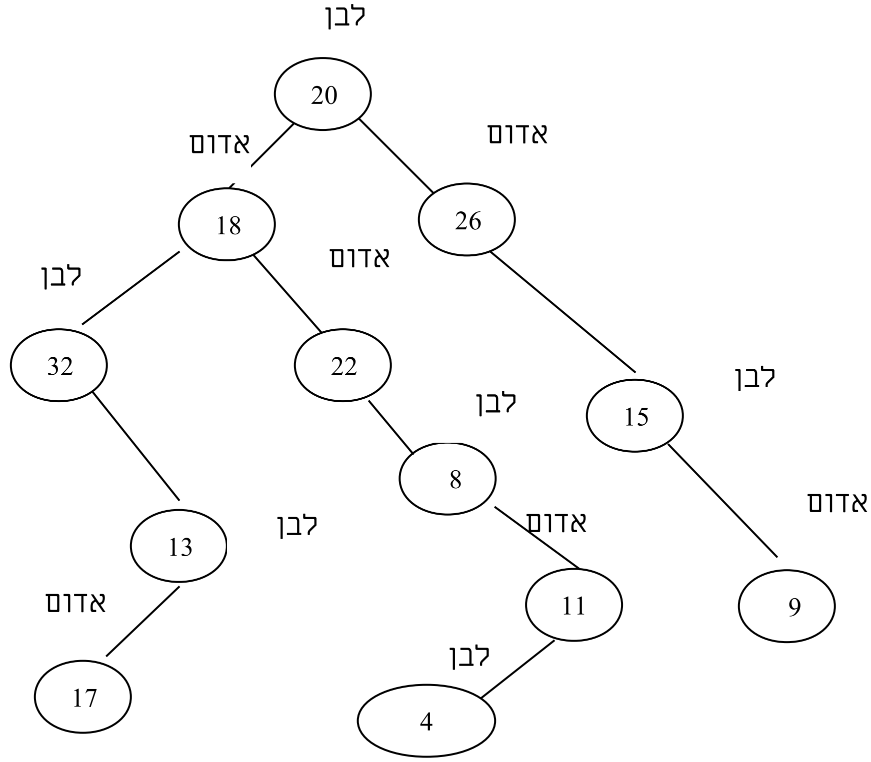
<u>הסבר לפלט:</u> אורך המסלול הארוך ביותר בין שני צמתים לבנים בעץ הוא 7, המסלול זה הוא המסלול המחבר בין שני הצמתים הלבנים 13 ו- 11 ורשימת הצמתים במסלול היא:

13 32 18 20 26 15 11

ראה/י דוגמה נוספת בעמוד הבא:

<u>דוגמה 2:</u>

יהי T עץ בינארי עם צבעים אדום-לבן שמתואר בציור הבא :



לאחר הקריאה לפונקציה P3(T) יתקבל הפלט:

7 13 32 18 22 8 11 4

<u>הסבר לפלט:</u> אורך המסלול הארוך ביותר בין שני צמתים לבנים בעץ הוא 7, מסלול זה הוא המסלול המחבר בין שני הצמתים הלבנים 13 ו- 4 ורשימת הצמתים במסלול היא:

13 32 18 22 8 11 4

פתרון

```
P3(T)
y=P3.1(T)
if (y.dist_of_two_whites==0) { print 0 }
else {print y.dist_of_two_whites
      print(y.list_of_two_whites)
      }
P3.1(T)
x=root(T)
if (x==null) {
              create(L1);creat(L2);
              y.list_of_two_whites=L1
              y.dist_of_two_whites=0
              y.list_root_to_white=L2
              y.dist_root_to_white=0
              return y
             }
y1=P3.1(T_left(x))
y2=P3.1(T_right(x))
create(L)
n=0
if (color(x)==white || y1.dist_root_to_white !=0 ||
    y2.dist_root_to_white !=0 ) {
    if (y1.dist_root_to_white >= y2.dist_root_to_white){
      L=y1.list_root_to_white
      insert(L,x)
      n=y1.list_root_to_white+1
    }
    if (y1.dist_root_to_white < y2.dist_root_to_white){
      L=y2.list_root_to_white
      insert(L,x)
      n=y2.list_root_to_white+1
    }
```

```
 }
y.dist_root_to_white=n
y.list_root_to_white=L
create(L1)
n1=0
if (color(x)=white ||
   (y1.dist_root_to_white!=0 && y2.dist_root_to_white!=0){
    n1=y1.dist_root_to_white+y2.dist_root_to_white+1
    L1=merge(y1.list_root_to_white,y2.list_root_to_white)
    insert(L1,x)
}
max=max{y1.dist_of_two_whites,y2.dist_of_two_whites,n1}
if (max==y1.dist_of_two_whites){
   L1=y1.list_of_two_whites
   n1=y1.dist_of_two_whites
  }
if (max==y2.dist_of_two_whites){
   L1=y2.list_of_two_whites
   n1=y2.dist_of_two_whites
  }
y.dist_of_two_whites=n1
y.list_of_two_whites=L1
return y
```