ערימות - תורי עדיפויות – תרגילים עם פתרונות

שאלה 1

מהו זמן הריצה של מיון ערמה על קלט בו כל הערכים זהים?

פתרון:

O(n), כי אחרי כל הוצאה של המקסימום, הגלגול כלפי מטה יסתיים אחרי רמה אחת (האלגוריתם יגלה שהשורש לא קטן מבניו ויסיים).

שאלה 2

נתונות שתי ערמות A_1 ו-A_2 בגודל n_1 ו-n_2. נניח ש- $n_1 \geq n_2$ ושכל איבר של A_1 גדול מכל איבר של A_2. הסבירו איך למזג את שתי הערמות לתוך ערמה אחת בזמן ריצה $O(n_2)$.

פתרון:

נכניס את אברי A_2 כעלים לערמה A_1. בגלל ש-A_1 גדולה יותר כל אברי A_2 יכנסו כעלים, ובגלל ושכל איבר של A_1 גדול מכל איבר של A_2, מובטח שלא יהיה צורך בגלגול כלפי מעלה.

שאלה 3

נתונה ערימת-מינימום H המקיימת את התנאי הנוסף: עבור כל צומת $x \in H$, כל המפתחות בתת-עץ השמאלי של x קטנים מ- (או שווים ל-) כל המפתחות בתת-עץ הימני של x.

א. מה מתקבל מהסריקה התחילית של H (כעץ בינרי)?

ב. בהינתן מערך כלשהו A, המכיל n איברים, תארו שגרה לבנית ערימת-מינימום, מאברי המערך, המקיימת את התנאי הנוסף שנזכר לעיל. נתחו את זמן הריצה של השגרה במקרה הגרוע.

פתרון:

א. הסריקה עוברת על המפתחות בסדר ממוין (מהקטן לגדול).

ב. נמיין את אברי A. נכניס את אברי המערך ל-H בסדר תחילי. זמן הריצה של recbuild הוא O(n) כי הוא מבצע סריקה בסדר תחילי. זמן הריצה של build נקבע ע"י זמן הריצה של המיון: O(nlgn).

```
build(H,A)
sort(A)
recbuild(H,1,A,1)
```

```
recbuild(H,i,A,j)
if i<=length[H]
then
     H[i]←A[j]
     j←recbuild(H,2i,A,j+1)
     j←recbuild(H,2i+1,A,j)
return j
```

שאלה 4

בהינתן רשימה של n תת-קטעים של $[0,1]$:

$$[a_i, b_i], \qquad 0 \le a_i < b_i \le 1,$$

$$i = 1, 2, \dots, n,$$

כתבו אלגוריתם יעיל הקובע האם קיימת נקודה ב- $[0,1]$ שאינה שייכת לאף אחד מ- n התת-קטעים. מהי סיבוכיות האלגוריתם?

פתרון:

נמיין את הקטעים לפי נקודת ההתחלה שלהם. נעבור על הקטעים ונשמור את נקודת הסיום הגדולה ביותר שמצאנו עד כה. אם קטע מתחיל אחרי נקודת הסיום הגדולה ביותר שמצאנו עד גילינו "פער" בו יש נקודות שלא שייכות לאף קטע.

```
Sort the segments according to their starting point
B←0
for i←1 to n do
  if aᵢ>B
  then return true
  B←max(bᵢ,B)
return B=1
```

שאלה 5

כתבו אלגוריתם, לא רקורסיבי, שמקבל מערך a שמייצג ערמה. וסורק את הערמה בסדר תחילי.

רמז: האלגוריתם ישתמש במחסנית.

פתרון:

```
if size[a]>0
then push(s,1)
while not empty(s) do
     i←pop(s)
     print(a[i])
     if 2i+1<size[a] then push(s,2i+1)
     if 2i<size[a] then push(s,2i)
```

שאלה 6

נוסיף להגדרת הערמה את התנאי: לכל צומת שהוא אב לשני בנים, ערך הבן השמאלי גדול או שווה לערך הבן הימני.

בהינתן ערמה H המקיימת את התנאי הנוסף, תארו שגרה (אין צורך לכתוב פסידוקוד) שתבצע את הפעולה הבאה (עם שמירת התכונה החדשה): הגדלת האיבר $H[i]$ בערך d ($d>0$) ותיקון הערמה. זמן הריצה יהיה $O(lgn)$.

פתרון:

מגדילים את ערך הצמת **i** , ומבצעים את שגרת התיקון הבאה:

כל עוד לא הגענו לשרש:

אם הוא בן ימני וגדול מאחיו מחליפים בינו לבין אחיו.

משווים אותו עם אביו: אם הוא גדול מאביו מחליפים ביניהם. אם הוא לא גדול מאביו: סיימנו.

שאלה 7

נתונה ערמה H המקיימת את התנאי הנוסף: עבור כל צומת $x \in H$, כל המפתחות בתת-עץ השמאלי של x קטנים מ- (או שווים ל-) כל המפתחות בתת-עץ הימני של x.

מה מתקבל מהסריקה בסדר סופי של H?

פתרון:

מקבלים את רשימת הצמתים ממוינת מהקטן לגדול.

שאלה 8

A *d*-ary heap is like a binary heap, but (with one possible exception) non-leaf nodes have *d* children instead of 2 children.

How would you represent a *d*-ary heap in an array? Where will the parent and the leftmost son of the node in index *i* be?

פתרון:

כמו בערמה בינרית, הצמתים יאוחסנו כרצף של ערכים במערך.

The parent of *i* is $\left\lfloor \frac{i+d-2}{d} \right\rfloor$

The leftmost son of *i* is $(i-1)d+2$

שאלה 9

a. Write an **efficient** function that searches for x in an array a[] - that represents a minimum heap. The size of the array is n.
   If x is in the array, the function should return 1. If x is not in the array, the function should return 0.
   The header of the function is:
   int search(int[] a, int n, int x)

b. What is the asymptotic running time of function *search*?
c. If a[] would represent a maximum heap (instead of a minimum heap), what changes would you need to make in function search?

פתרון:

a.
```
int search(int[] a, int n, int x)
{
        for (int i=0;i<n;i++)
                if (a[i]==x)
                        return 1;
        return 0;
}
```
b. The running time is O(n)
c. No change is needed if the heap is a maximum heap.