19.6.08 ⑤

יש לנו n מספרים וילדים לבחור הילד מספר שמגדיל לפחות $\frac{n}{2}$ פעמים.

<u>דרך I:</u> למיין את המערך ב $\Theta(n \log n)$ ואז מכן לבדוק סדרתי של המערך. בדוק למעלה הילד שמספר שחוזר יותר מ $\frac{n}{2}$ פעמים ($\Theta(n)$).

<u>דרך II:</u> אם ידוע לנו שהילד שלנו מספר שיופיע לפחות $\frac{n}{2}$ פעמים אז נילך מספר כמה פעמים הוא מופיע במערך ($\Theta(n)$ עבור כמה פעמים הוא מופיע במערך).

<u>דרך III (שופר):</u> נתבונן תמונה את התנאיין של המערך (אם המערך היה ממיין היינו מוצאים כאן הגדול), שאחרי מכן נספור כמה פעמים הוא מופיע במערך בהתבסס על העובדה שאם יש מספר שיופיע לפחות $\frac{n}{2}$ פעמים במערך אז הוא חייב להיות הגדול (לפחות אחד מהם).

מבחינת סיבוכיות הדרך תהיה לינארית ($\Theta(n)$ סיבוכיות התנאיין + $\Theta(n)$ ספירה).

במבחן נשתמש בכך שנתן לנחקר לצאת בשאלת צורת של מציאת נוקודות נוקודות הזויות ושאנו הולכים יש מעגל שעליו ישלוש $\frac{2\pi}{3}$ מהנקודות (לדוגמה).

בשאלה נתון את המערך של כל שאר הנקודות ואנו נסרוק רחוק מיפוליים מספרים ה $\frac{1}{3}\cdot 2\pi$, $\frac{2}{3}\cdot 2\pi$, $\frac{3}{3}\cdot 2\pi$, $\frac{4}{3}\cdot 2\pi$, $\frac{5}{3}\cdot 2\pi$, $\frac{6}{3}\cdot 2\pi$ (שווה ל 0)

צהו פיתרון בסיבוכיות לינארית $\Theta(n)$,

⊛ המשך התרגול בהקרנת לדף הנלווה.

1	3	5	7
9	11	13	15
17	19	21	23
25	27	29	31

2	3	6	7
10	11	14	15
18	19	22	23
26	27	30	31

4	5	6	7
12	13	14	15
20	21	22	23
28	29	30	31

8	9	10	11
12	13	14	15
24	25	26	27
28	29	30	31

16	17	18	19
20	21	22	23
24	25	26	27
28	29	30	31

בכל קופסא מיתרים מספר בין 1 ל-31
ובכל שלב בהכנסה יש שלושה היגד מבלי לאבד
מבנה ערימה מינימום נשתדל.
במקרה שמכניסים את המינימלי ביותר
במספרים, כל הערכים מיידע את עצמו בהכנסת
הבינארית, כלומר כל מי שברמה הראשונה
יש לו 1 בספרה הראשונה וכן הלאה, כדי לאבד
סידור המרכיבים. מהמקום מוכנסים על תכונת
של ערימה בינארי.

(15) סיבת התקיון מובילה במקרה בכל שבבה
בהתחלה.

ערמות

Operation	Linked List	Binary Heap	Binomial Heap	Fibonacci Heap †	Relaxed Heap
Make-Heap	1	1	1	1	1
Insert	1	log n	log n	1	1
Find-Min	n	1	log n	1	1
Delete-Min	n	log n	log n	log n	log n
Decrease-Key	n	log n	log n	1	1
Merge	1	n	log n	1	1

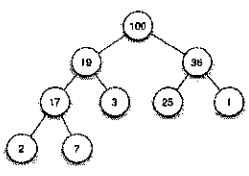
n = number of elements in priority queue

† amortized

2

תזכורת: Heaps

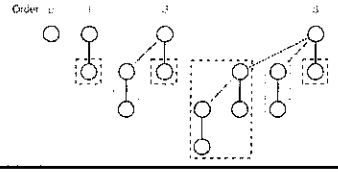
- עץ בינארי מלא
- החוק הבסיסי
  - אם צומת B צאצא של צומת A אזי Key(A)≤Key(B)
- הפעולות הנתמכות
  - Find-min
  - Delete-min
  - Decrease-key
  - Insert
  - Merge



3

תזכורת: Binomial Heaps

- תחילה נגדיר Binomial Tree:
  - עץ בינומי מדרגה 0 מכיל צומת יחידה
  - עץ בינומי מדרגה k הוא
    - בעל שורש מדרגה k
    - ילדיו הינם עצים בינומיים מדרגות k-1,k-2,...,0 (בסדר זה)
    - עץ בינומי מדרגה k מכיל 2^k צמתים והינו בגובה k



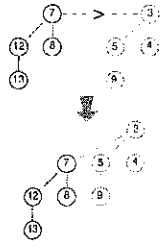
4

כידוע נוכל לממש ערוץ עדיפות כמו ערימה
על בינומי עם n איברים
ונוכל לבצע את כל הפעולות ביעילות
וכל הפעולות שם על מתנים
יכיון שגם שם על מתנים.

(מ"ו) בשכתוב הפיסקה נשתמש בסיס ואחר.
(בהמשך תחת "סדרת בינומי שמנים").

תזכורת: Binomial Heaps

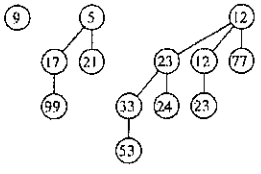
- איחוד שני עצים בינומיים מסדר k-1
  - ניצור עץ בינומי מסדר k ע"י תלייה אחד העצים כבן השמאלי ביותר של העץ השני



5

תזכורת: Binomial Heaps

- הגדרת Binomial Heap
  - סט עצים בינומיים המקיימים את התכונות הבאות
    - כל עץ מקיים את תכונת minimum-heap (כל צאצא גדול מההורה שלו)
    - עבור כל סדר k של עץ בינומי, יש 0 או 1 עצים כאלו ב-heap

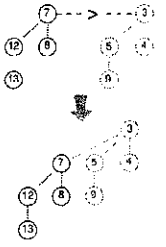


6

תזכורת: Binomial Heaps

- פעולת Merge של שני עצים מדרגה k

```
function mergeTree(p, q)
    if p.root <= q.root
        return p.addSubTree(q)
    else
        return q.addSubTree(p)
end
```



תזכורת: Binomial Heaps

- פעולת Insert
  - ניצור heap חדש המכיל את האבר החדש, ונבצע merge בין שני ה-heaps
- פעולת minimum
  - עלינו לחפש את הערך המינימלי מבין שורשי העצים בheap
- פעולת delete-min
  - מצא את האבר ומחק אותו
  - הפוך את בנין ל-binomial heap ומזג את שני ה-heaps
- פעולת Decrease-min
  - בדומה לפעולות ב-heap רגיל
- פעולת Delete
  - שנה את הערך ל-∞- ובצע delete-min

Properties of binomial trees

1) $|B_k| = 2^k$
2) $\text{degree}(\text{root}(B_k)) = k$
3) $\text{depth}(B_k) = k$

==> The degree and depth of a binomial tree with at most n nodes is at most log(n).

Define the rank of B_k to be k

Binomial heaps (ops cont.)

Basic operation is meld(h1,h2):

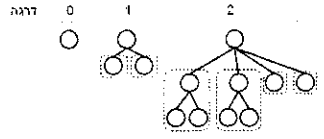
Like addition of binary numbers.

	B_5	B_4		B_2	B_1	
h1:		B_4	B_3		B_1	B_0
h2:		B_4	B_3			B_0 +
	B_5	B_4		B_2		

10

נגדיר עצים בינומיים "שמנים" בצורה הבאה:

- עץ בינומי "שמן" מדרגה 0 מכיל צומת אחד בלבד.
- עץ בינומי "שמן" מדרגה k ניתן לבנות משלושה עצים בינומיים "שמנים" מדרגה k-1 כאשר נתלה שניים מהם על העץ השלישי.



תארו מבנה נתונים אנלוגי לערמה בינומית (binomial heap) המשתמש בעצים "שמנים".

11

ערמה בינומית "שמנה" מוגדרת בצורה הבאה:
בערמה יש n עצים בינומיים "שמנים" כאשר יש לכל היותר 2 עצים מדרגה k, לכל k. שורשי העצים מחוברים ביניהם ברשימה מקושרת. הפעולות מוגדרות בדומה לערמה בינומית רגילה וכאשר יש צורך לעשות merge בין העצים (יש יותר משני עצים מדרגה k) בונים משלושה עצים מדרגה k עץ יחיד מדרגה k+1.

12

תארו פעולת meld של שתי ערמות בינומיות "שמנות" שהגדרתם בסעיף הקודם.

פעולת meld הינה מיזוג בין שתי ערמות בינומיות "שמנות". הדבר מקביל לחיבור שני מספרים בבסיס 3 (כמו שבערמות בינומיות הדבר היה מקביל לחיבור מס' בבסיס 2). נתחיל מדרגה 0 ונוסיף את העצים בערמה אחת לשנייה. אם יש סה"כ 3 או יותר עצים מדרגה 0 בערמה החדשה נמזג שלושה ליצירת עץ יחיד מדרגה 1. כעת נעבור לדרגה הבאה (1) ונמשיך כך...

- הראו שאין מימוש heap שתומך בפעולות extract-min ו-insert בזמן $o(\log n)$ תחת הנחות מודל ההשוואות.
- פתרון
  - נבצע n פעולות הכנסה בזמן $n \cdot o(\log n)$
  - נבצע n פעולות הוצאת מינימום בזמן $n \cdot o(\log n)$
  - סה"כ $o(n \cdot \log n)$
- למדנו כי מיון n אברים הוא $\Omega(n \cdot \log n)$, סתירה

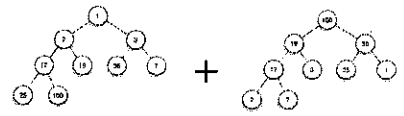
תרגיל 3 – Median Heap

- ממשו מבנה נתונים התומך בפעולות
  - insert בזמן O(logn)
  - extract-median בזמן O(logn)
  - find-median בזמן O(1)

תרגיל 3 - פתרון

Min-heap

Max-heap



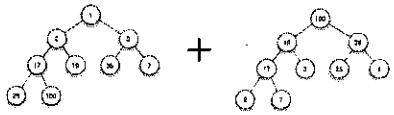
האברים הגדולים (מהחציון)

האברים הקטנים (עד החציון)

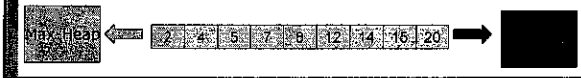
תרגיל 3 - פתרון

Min-heap

Max-heap



- נשתמש ב-max-heap ו-min-heap
- n/2 הערכים הגדולים ביותר יישמרו ב-max-heap
- השאר יישמרו ב-min-heap
- החציון תמיד נמצא בשורש של אחד מהם



תרגיל 3 - פתרון

- Find-median
  - If (size(minheap)>size(maxheap))
    - return getmin(minheap)
  - Else
    - return getmax(maxheap)

O(1)

- Insert(x)
  - If (x<getmin(minheap))
    - Insert(maxheap,x)
  - Else
    - Insert(minheap,x)
  - If (abs(size(minheap)-size(maxheap))>1)
    - Balance heaps (move root from bigger heap to smaller heap)

O(logn)

- Extract-Median
  - Extract median from the max-heap or min-heap...

O(logn)

תארו מבנה נתונים התומך בפעולות הבאות על קבוצה S מתחום סדור מלא

- Median(S): מחזיר את החציון ב-S (אם מס' אברים זוגי – חציון תחתון)
- Min(S): מחזיר את האבר הקטן ביותר ב-S
- Max(S): מחזיר את האבר הגדול ביותר ב-S
- Insert(x,S): מוסיף אבר x ל-S
- Delete(x,S): מניחים ש-x נמצא לפני הפעולה ב-S. הפעולה מוציאה את x מ-S

על הפעולות median,min,max לקחת O(1) זמן במקרה הגרוע, ועל הפעולות insert ו-delete לקחת O(logn) זמן במקרה הגרוע.

שומרים median-heap כמו שלמדנו בכתה. כלומר, שתי ערימות: אחת ערימת-מקסימום ואחת ערימת מינימום, כאשר כל איבר בערימת המינימום גדול מכל איבר בערימת המקסימום, ובנוסף הגודל של ערימת המקסימום שווה או קטן בדיוק 1 מהגודל של ערימת המינימום. בפעולת insert או delete מכניסים/מוציאים את האיבר מ-/ל- ערימה המתאימה, ואם צריך מעבירים איבר מערימה אחת לאחרת כדי לאזן את הגדלים. ה-median תמיד יהיה האיבר המינימלי בערימת המינימום. בנוסף, מחזיקים ערימת מינימום וערימת מקסימום שמחזיקות את כל האיברים, כדי לענות על שאילתות ה-min וה- max