הוכן ע"י אמיר רובינשטיין

מבני נתונים
ומבוא לאלגוריתמים

נושא 7

בעיית הבחירה
Selection problem

בתוכנית

פרק 9 בספר הלימוד

- נגדיר את בעיית הבחירה

- נכיר פתרונות יעילים לבעיית הבחירה:
  - האלגוריתם Random-Select
  - האלגוריתם Select

בעיית הבחירה

הגדרת הבעיה

קלט: קבוצה בת n איברים, ואינדקס $1 \le i \le n$.

פלט: האיבר ה- i הכי קטן, כלומר: האיבר* במקום ה- i בסדר הממוין. * *איבר או האיבר?*

פתרונות לבעיה (אלגוריתמים הפותרים אותה):

- מיון והחזרת האיבר באינדקס i, בזמן $\Theta(n\log n)$.
- קיים פתרון בזמן ליניארי עבור: $i=1$ (מינימום)

$i=n$ (מקסימום)

ולמעשה גם עבור $i=k$ או $i=n-k$ כאשר k קבוע.

```
Minimum(A)   ►A is array
1.   min ← A[1]
2.   for i ← 2 to length[A]
3.       if A[i] < min
4.           min ← A[i]
5.    return  min
```

שאלה: כמה פעמים בממוצע מתבצעת שורה 4 ?

תשובה: הסיכוי שהאיבר $A[k]$ יהיה המינימום של $A[1..k]$ הוא $1/k$

סכום התוחלות של מספר הכניסות לשורה 4 הוא $\frac{1}{2}+\frac{1}{3}+\ldots+\frac{1}{n}=H_n-1=\Theta(\log n)$

בעיית הבחירה

- האם קיים פתרון ליניארי לכל i?

למשל, האם ניתן למצוא את החציון בזמן לינארי?

חציון (median) יוגדר כאיבר ה-$\left\lfloor \frac{n+1}{2} \right\rfloor$ בגודלו.

נציג כעת שני אלגוריתמים:

✓ Random-Select שפותר את הבעיה בזמן ליניארי בממוצע אבל לא במקרה הגרוע

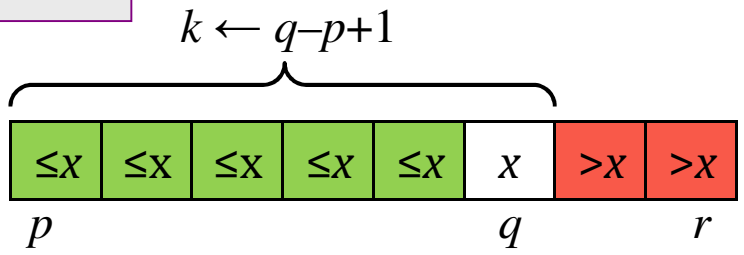
זהו אלגוריתם רקורסיבי אקראי.

✓ Select שפותר את הבעיה בזמן ליניארי במקרה הגרוע (אבל הוא מסובך...)

זהו אלגוריתם רקורסיבי דטרמיניסטי.

Random-Select

```
Random-Select (A, p, r, i)
1.  if  p = r
2.       return  A[p]
3.  q ← Random-Partition (A, p, r)
4.  k ← q–p+1
5.   if i = k
6.      return  A[q]
7.      else if  i < k
8.                  return Random-Select (A, p, q–1, i)
9.            else return Random-Select (A, q+1, r, i–k)
```



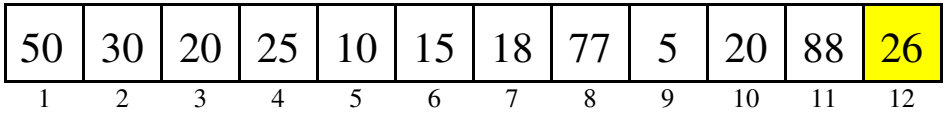
האלגוריתם מבצע חלוקה (עם ציר שנבחר אקראית) של המערך.

- אם "יש מזל", ומספר האיברים k בחלק השמאלי, כולל הציר, הוא בדיוק i, אז סיימנו;
- אחרת, אם $i < k$ אז ממשיכים לחפש את האיבר ה- i בצד שמאל;
- אחרת, ממשיכים לחפש את האיבר ה-$(i–k)$ בצד ימין.

Random-Select

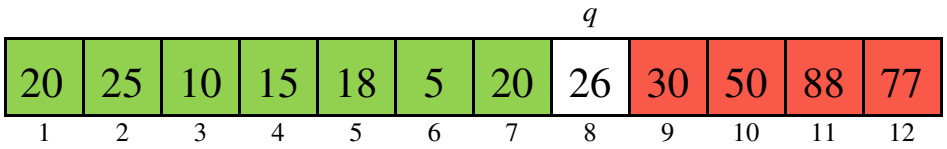
<u>הדגמה</u>

מחפשים את האיבר ה- 4

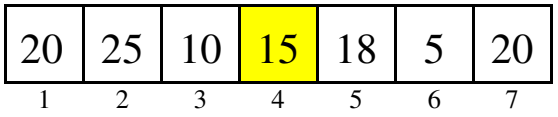


נניח שהציר שנבחר אקראית הוא 26.

תוצאת החלוקה:

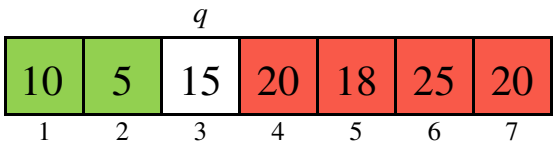


מחפשים את האיבר ה- 4 בשמאל.

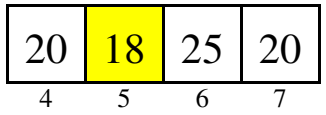


נניח שהציר הוא 15.

תוצאת החלוקה:

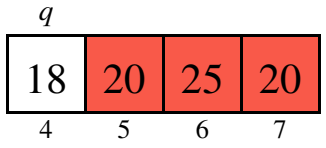


מחפשים את האיבר ה- 1= 4-3 בימין.



נניח שהציר הוא 18

תוצאת החלוקה:



עוצרים ומחזירים את 18.

Random-Select

ניתוח זמנים

מקרה טוב: חלוקה אחת ואחריה עוצרים $T(n) = \Theta(n)$

מקרה גרוע: חלוקה ביחס $0{:}n\text{-}1$ בכל שלב, והמשך הרקורסיה עם החלק הגדול.

$T(n) = T(n\text{-}1) + \Theta(n) = \Theta(n^2)$

ממוצע (הוכחה בספר הלימוד): $\Theta(n)$

לסיכום, מצאנו לבעיית הבחירה פתרון ליניארי בממוצע, אך לא במקרה הגרוע.

מקרה פרטי - מציאת חציון

שאלה: כיצד נמצא חציון של מערך בגודל n בסיבוכיות זמן ליניארית בממוצע?

תשובה: נקרא ל- $\text{Random-Select}(A, 1, n, \lfloor \frac{n+1}{2} \rfloor)$

Select

הרעיון לשיפור הוצע בשנת 1973, במאמר:

- M. Blum, R.W. Floyd, V. Pratt, R. Rivest and R. Tarjan, "Time bounds for selection," *J. Comput. System Sci.* 7 (1973) 448-461.

<u>הרעיון</u>: כדי לשפר את זמן הריצה, צריך לדאוג שהחלוקות יהיו מאוזנות יחסית.

- לא בהכרח ביחס 1:1, אבל כפי שנראה, לכל הגרוע ביחס 3:7 בערך.
- נראה שיטה לבחור את איבר הציר באופן "חכם" יותר, שיבטיח זאת.
- נשתמש בגרסה שונה מעט של חלוקה, שמקבלת איבר ציר ומחלקת לפיו.
  נקרא לגרסה זו Partition-With-Pivot.

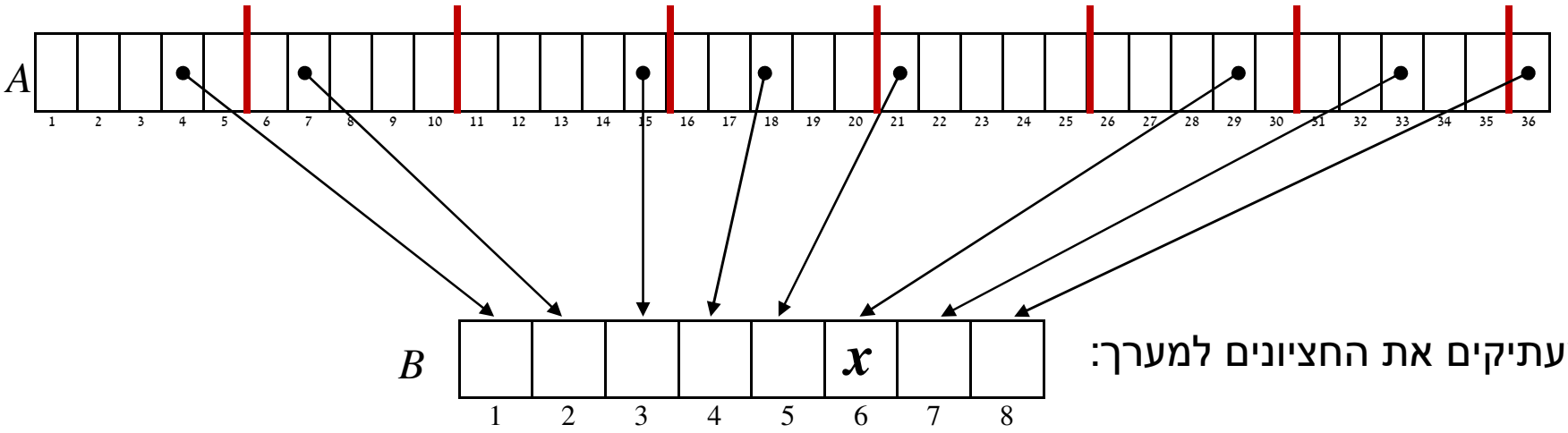
<u>רעיון בחירת הציר</u>:

שיטת חציון החציונים.

Select

הרעיון

מחלקים לחמישיות, ומוצאים חציון של כל חמישייה (למשל ע"י מיונה):



מעתיקים את החציונים למערך:

ממציאי השיטה הוכיחו כי החציון של החציונים הללו (החציון של B) הוא איבר ציר טוב לחלוקת A.

האם אתם מכירים שיטה יעילה למציאת חציון?

קריאה רקורסיבית ל- Select ! ובדוגמה זו, יש לקרוא ל- Select(B,1,8,4).

חציון החציונים שמצאנו (מסומן x) הוא הציר לחלוקה של A.

ההמשך כמו Random-Select.

Select

האלגוריתם:

```
Select (A, p, r, i)
1.  if  p = r
2.        return A[p]
3.   x ← Choose-Pivot (A, p, r)
4.   q ← Partition-With-Pivot (A, p, r, x)
5.   k ← q–p+1
6.   if i = k
7.        return  A[q]
8.     else if  i < k
9.                 return  Select (A, p, q–1, i)
10.          else return  Select (A, q+1, r, i–k)
```

Choose-Pivot (A, p, r)

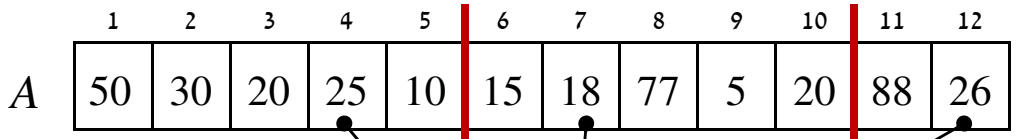
1. נמצא את החציון של כל חמישיית איברים רצופים (ועוד אולי השארית בסוף) למשל ע"י מיון כל קבוצה.
   נעתיק חציונים אלו למערך עזר B (גודלו $\lceil n/5 \rceil$).
2. נמצא ונחזיר את החציון של B (חציון החציונים). נעשה זאת ע"י הפעלת Select על B.

Select

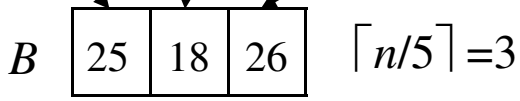
<u>הדגמה</u>

מחפשים את האיבר ה- 4.

המערך המקורי:

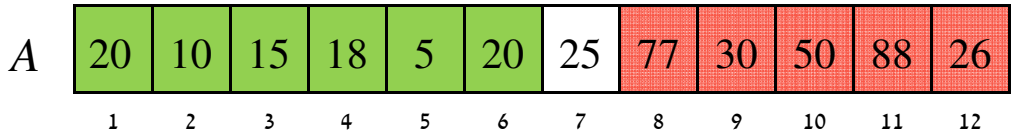


מערך החציונים:



מציאת חציון החציונים מתבצעת ישירות ע"י מיון (ובלי רקורסיה), כיוון שגודלו של $B \geq 5$.

Partition-With-Pivot של A סביב x=25:



וממשיכים לחפש את האיבר ה- 4 בצד שמאל: Select(A,1,6,4)

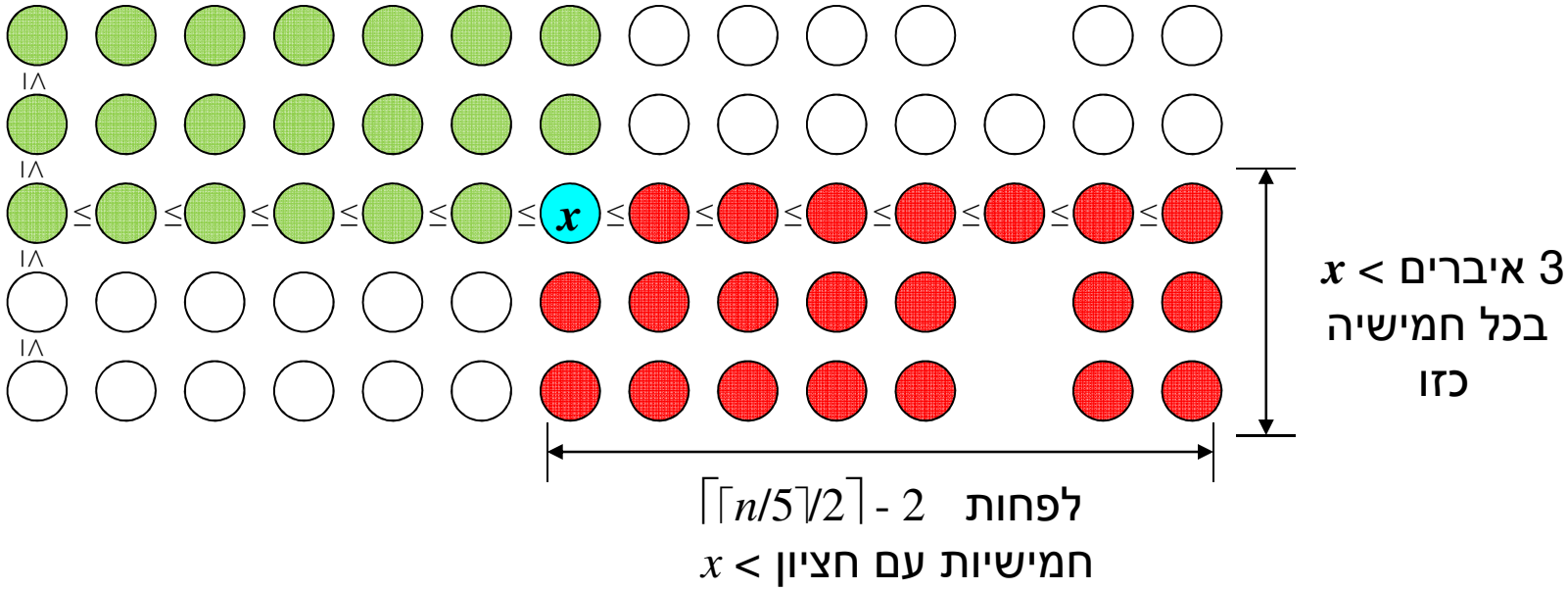
Select – ניתוח זמנים

ניתוח זמנים במקרה הגרוע

נניח לשם פשטות שכל האיברים שונים זה מזה.

לצורך הבהירות נסדר את החמישיות באיור הבא לפי גודל החציון שלהן, משמאל לימין

(כל חמישיה ממוינת מלמעלה למטה):



– בכל חמישייה שהחציון שלה גדול מהציר x יש בדיוק 3 איברים > x.

– כמה חמישיות כאלו יש? מכיוון ש- x הוא החציון של חציוני החמישיות, מחציתם > x.

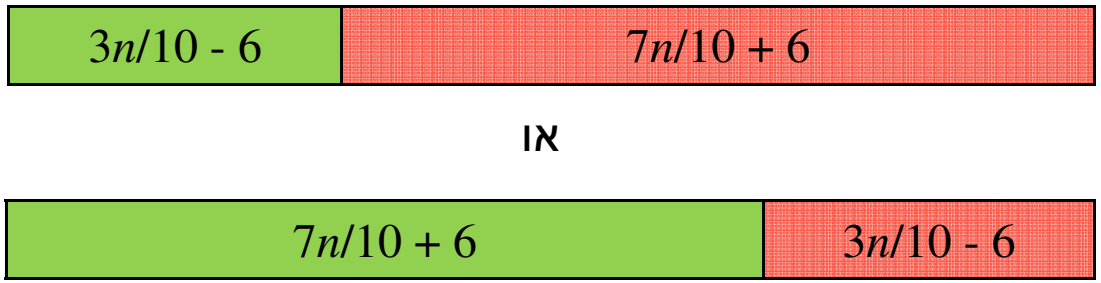
נוריד מזה את הקבוצה בעלת פחות מ- 5 איברים (אם ישנה), והחמישיה הכוללת את x עצמו.

– מכאן שיש ב- A לפחות $3(\lceil\lceil n/5\rceil/2\rceil - 2) \geq 3n/10 - 6$ איברים > x, כלומר כ- 30% לפחות.

Select – ניתוח זמנים

באופן דומה מראים שיש ב- A לפחות $3n/10 - 6$ איברים $< x$.

במילים אחרות, גם בחלוקה הגרועה ביותר, בחלק הקטן לפחות $3n/10 - 6$ איברים ובחלק הגדול לכל היותר $7n/10 + 6$ איברים.



- כלומר הבחירה ה"חכמה" של x כחציון החציונים מבטיחה שהחלוקה הגרועה ביותר האפשרית היא ביחס של 3:7 בערך.
- ה"מחיר" ששילמנו - מציאתו של x דורשת זמן: חלוקת המערך לחמישיות, מציאת חציונים, וקריאה רקורסיבית.

Select – ניתוח זמנים

חישוב עלות מציאת חציון החציונים (ChoosePivot):

- מציאת חציון של כל חמישייה דורשת $\Theta(1)$ זמן (מדוע?).

  סה"כ מציאת $\lceil n/5 \rceil$ חציונים דורשת אם כן $\Theta(n)$ זמן.

- לאחר מכן ישנה קריאה רקורסיבית ל- Select על מערך בגודל $\lceil n/5 \rceil$.

הנוסחה שמתארת את זמן הריצה הכולל של Select במקרה הגרוע היא אם כן:

$$T(n) = T(\lceil n/5 \rceil) + T(7n/10 + 6) + \Theta(n)$$

מציאת החציונים + חלוקה

ניתן להוכיח בשיטת ההצבה ש- $T(n)= \Theta(n)$ (פרטים בספר הלימוד, עמוד 159)

ניתן גם לראות זאת ע"י הזנחת קבועים וערכי תקרה:

$$T(n) = T(n/5) + T(7n/10) + \Theta(n)$$

$$\alpha + \beta < 1$$

כלומר כעת בידינו אלגוריתם ליניארי במקרה הגרוע לפתרון בעיית הבחירה.

Select – סיבוכיות זיכרון

ומהי סיבוכיות הזיכרון הנוסף של Select?

דרישות הזיכרון הנוסף של Select מתבטאות ב:

- הקצאת מערכי החציונים
- עומק מחסנית הרקורסיה

עומק מחסנית הרקורסיה הוא לוגריתמי (מדוע?) ולכן זניח אסימפטוטית לעומת הקצאת מערכי החציונים, שגודלם ליניארי בגודל הקלט*.

נוסחת הנסיגה המתארת את סיבוכיות הזיכרון של Select עבור קלט בגודל n היא אם כן:

$$S(n) = \max\{S(\lceil n/5 \rceil) , S(7n/10 + 6)\} + \lceil n/5 \rceil$$

$$= S(7n/10 + 6) + \lceil n/5 \rceil$$

➔ $S(n) = \Theta(n)$

*שאלה

כיצד אפשר לוותר על הקצאת מערכי החציונים?
מה סיבוכיות הזיכרון הנוסף עם שיפור כזה?

מקרה פרטי - מציאת חציון

שאלה: כיצד נמצא חציון של מערך בגודל n בסיבוכיות זמן ליניארית?

תשובה: נקרא ל- $\text{Select}(A, 1, n, \left\lfloor \frac{n+1}{2} \right\rfloor)$

הערה: אמנם סיבוכיות הזמן היא ליניארית, אבל הקבועים החבויים בה הם גדולים למדיי.
לכן, מבחינה מעשית, לפעמים עדיף להשתמש ב- Random-Select, או אפילו למיין ולהחזיר את האיבר האמצעי.

מיון מהיר הדטרמיניסטי

תרגיל 9.3-3 מספר הלימוד

כיצד ניתן לשפר את מיון מהיר, כך שזמן הריצה שלו על מערך בגודל n יהיה $\Theta(n\log n)$

במקרה הגרוע?

פתרון:
בכל שלב שבו צריך לבחור איבר ציר לחלוקה, הוא ייבחר כחציון של תת-המערך הרלוונטי, באמצעות קריאה ל- Select.

$$T(n) = 2T(n/2) + \Theta(n) = \Theta(n\log n)$$

Select + Partition

שאלות חזרה

1. כיצד ניתן למצוא את המינימום של מערך באמצעות קריאה ל- Select? ואת המקסימום? האם דרך זו יעילה מבחינה תיאורטית ומבחינה מעשית?

2. עבור אילו גדלים של מערכים Choose-Pivot לא צריכה לקרוא שוב ל- Select ? מה עושה Choose-Pivot במקרה זה ?

3. נסו לעקוב אחר מהלך הקריאות הרקורסיביות של Select על מערך A בגודל 31.

4. את Partition-With-Pivot אפשר לממש ע"י קריאה ל- Partition ה"רגילה". כיצד?

תשובות לשאלות חזרה

1. מינימום: Select$(A, 1, n, 1)$. מקסימום: Select $(A, 1, n, n)$.
מבחינה תיאורטית, זמן הריצה בשני המקרים הוא ליניארי ב- n. אבל כפי שראינו, ניתן למצוא מינימום ומקסימום גם ללא שימוש ב- Select, ע"י מעבר פשוט על המערך. גם זה פתרן ליניארי, אבל עם קבועים קטנים בהרבה. לכן מבחינה מעשית הוא עדיף.

2. עבור מערך בגודל קטן מספיק (למשל 5 או קבוע אחר כלשהו), אפשר פשוט להחזיר את החציון שלו (ע"י מיון).

3. מערך החציונים B יהיה בגודל 7, ולכן מציאת החציון שלו תגרור קריאה נוספת ל- Select.
B יחולק לחמישיות, וייווצר מערך חציונים נוסף C בגודל 2. החציון x' של C יוחזר ללא קריאות נוספות ל- Select, וישמש לחלוקה של B. כעת החיפוש אחר החציון של B ימשיך בחלק שמאל או ימין של החלוקה (או במקרה הטוב יסתיים מייד). רק לאחר מציאת החציון x של B הוא יוחזר וישמש לחלוקה של A, ואחרי חלוקת A נמשיך לחפש את האיבר ה- i של A בצד שמאל או ימין של החלוקה (או במקרה הטוב נסיים מייד).

4. מחליפים בין x לבין האיבר במקום האחרון, ואז פשוט קוראים ל- Partition.

תרגילים

תרגילים נוספים

1. נגדיר <u>איבר רוב</u> במערך בגודל n כאיבר שמופיע יותר מ- $n/2$ פעמים.

הציעו אלגוריתם, שבהינתן מערך בגודל n מוצא איבר רוב, אם קיים כזה, ואחרת מודיע שלא קיים איבר רוב.

2. נתון מערך A בגודל n של איברים כלשהם.

א. הראו אלגוריתם להדפסת $\lfloor\sqrt{n}\rfloor$ האיברים הקטנים במערך בסדר ממוין, בזמן ליניארי במקרה הגרוע. נמקו מדוע לא ניתן לפתור את הבעיה בזמן $o(n)$.

ב. הוכיחו כי לא ניתן להדפיס את $\lfloor n/2\rfloor$ האיברים הקטנים במערך בסדר ממוין, בזמן $o(n\log n)$.

3. נתון מערך ובו m מספרים שונים זה מזה כלשהם.
הציעו מבנה נתונים לביצוע הפעולות הבאות, תוך עמידה בדרישות סיבוכיות הזמן:

- Init אתחול מבנה הנתונים ב- $O(m)$.
- Insert(x) הוספת x למבנה ב- $O(\log n)$, n מספר האיברים במבנה בעת ביצוע הפעולה.
- Find-Mid הדפסת ערך החציון ב- $O(1)$.
- Del-Mid הוצאת החציון מהמבנה ב- $O(\log n)$.

תארו תחילה מה מכיל המימוש שלכם, ואח"כ הסבירו כיצד מתבצעת כל פעולה ומדוע היא עומדת בדרישות הסיבוכיות.

פתרון 1

ניסיון ראשון

מיון כלשהו - $\Omega(n\log n)$, ומעבר על המערך כדי לבדוק האם יש $n/2$ איברים רצופים זהים – $\Theta(n)$.

סה"כ: $\Omega(n\log n)$ (החסם ההדוק תלוי בבחירת המיון).

ניסיון שני

נשים לב שאם קיים איבר רוב, אז הוא החציון.

נמצא בעזרת SELECT את החציון, ואז נעבור על המערך ונספור כמה פעמים מופיע בו החציון. אם הוא מופיע יותר מ- $n/2$ פעמים נחזיר אותו. אחרת נודיע שאין איבר רוב.

סיבוכיות: $\Theta(n)$

פתרון 2

א. נפעיל את Select למציאת האיבר ה- $\lfloor\sqrt{n}\rfloor$ בגודלו: $\text{Select}(A, 1, n, \lfloor\sqrt{n}\rfloor)$

כעת $\lfloor\sqrt{n}\rfloor$ האיברים הקטנים של A ממוקמים בשמאל המערך.

נמיין אותם בעזרת מיון בועות למשל.

סיבוכיות הזמן היא: $\Theta(n)+\Theta(\lfloor\sqrt{n}\rfloor^2)=\Theta(n)$

חייבים לעבור על כל n איברי המערך לכן לא ניתן לפתור את הבעיה בזמן $o(n)$.

ב. נניח בשלילה שניתן לעשות זאת. כלומר קיים אלגוריתם, נקרא לו Alg, שמדפיס את מחצית האיברים הקטנים של מערך נתון בסדר ממוין.

אז ניתן גם למיין כל מערך מאורך n בזמן $o(n\log n)$ (וזו כמובן סתירה לחסם התחתון למיון):

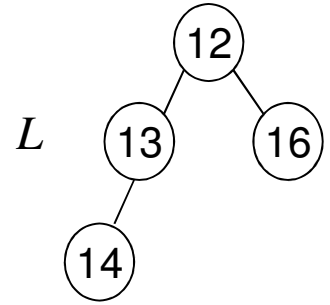
- בהינתן מערך הקלט, נוסיף לו מימין n פעמים "אינסוף" (מעשית, אפשר למצוא תחילה את המקסימום, ולהוסיף n איברים ששווים למקסימום ועוד אחד). שלב זה רץ בזמן $\Theta(n)$.
- כעת נקרא לאלגוריתם Alg על המערך ה"מוכפל", ונקבל את איברי המערך המקורי ממוינים.

שימו לב שלא מספיק להראות שהדרך מסעיף א' לא עובדת כאן.

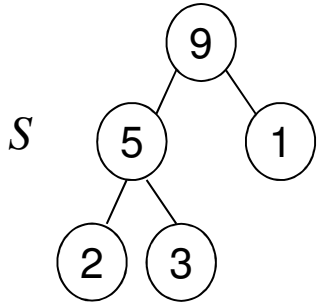
פתרון 3

מבנה הנתונים:

ערימת מינימום L של $\lfloor n/2 \rfloor$ האיברים הגדולים



ערימת מקסימום S של $\lceil n/2 \rceil$ האיברים הקטנים



מימוש הפעולות:

– Init נפעיל את Select למציאת החציון (כולל חלוקת האיברים סביבו). כעת, נבנה את שתי הערימות הנ"ל משני חצאי המערך, בעזרת שתי קריאות ל- Build-Heap.

זמן: $\Theta(m) + 2\cdot\Theta(m/2) = \Theta(m)$.

– Find-Mid נחזיר את ערך השורש של S (כלומר את $S[1]$). זמן: $\Theta(1)$.

– Insert(x) אם $x \le S[1]$, נכניס אותו ל- S, אחרת ל- L. כעת:
אם $|L|=|S|-2$ נמחק (Extract-Max) את השורש של S ונכניס (Heap-Insert) אותו ל- L.
אחרת אם $|L|=|S|+1$ נמחק את השורש של L ונכניס אותו ל- S.

– Del-Mid נמחק את השורש של S. אם $|L|=|S|+1$ נפעל כמתואר לעיל.

זמן: שתי הפעולות האחרונות במקרה הגרוע: $\Theta(3\log(n/2)) = \Theta(\log n)$