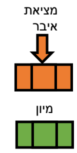
אלגוריתמים וזמני ריצה

MADE BY:TAL KATZ, OMER STERN,YUVAL COHN,AMIT SHAFRAN

1



נתון מערך $A[1..n]$ של מספרים. נגביל את האלגוריתם מיון-הכנסה ל- $(k-1)$ האיטרציות הראשונות של הלולאה החיצונית. כידוע, נקבל את התת-מערך $A[1..k]$ ממוין בסדר עולה (או לא-יורד).

(6 נקי) א. כתוב שגרה (בפסֵידוקוד) המנצלת את תוצאת המיון החלקי כדי לבצע חיפוש של הערך z במערך $A[1..n]$.

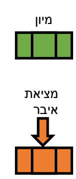
(7 נקי) ב. נסמן $m = n - k$ ונתייחס אל m כאל משתנה בלתי-תלוי ב-n $(0 \leq m \leq n)$. מהו זמן הריצה של השגרה שבסעיף א' במקרה הגרוע כפונקציה של n ושל m?

(6 נקי) ג. עבור אילו ערכים של m כפונקציה של n (בסימון אסימפטוטי) ניתן לבצע את פעולת החיפוש בזמן $O(\lg n)$?

(6 נקי) ד. כמה פעולות חיפוש (כפונקציה של n) ניתן לבצע בזמן כולל $O(n^2)$ כאשר:

$$m = O(\lg n)\ \ ;\ m = O(n/\lg n)\ \ ;\ m = O(n)\ ?$$

2



נתון מערך $A[1..n]$; נבחר איבר ציר x עבור שגרת החלוקה PARTITION ונסמן ב-m את מספר ההופעות של הערך x במערך A.

(7 נקי) א. כתוב שגרת חלוקה חדשה M-PARTITION, המחלקת את המערך A לשלושה תת-מערכים המכילים: הראשון, את כל האיברים הקטנים מ-x; השני, את כל ההופעות של x; השלישי, את כל האיברים הגדולים מ-x. זמן הריצה של השגרה יישאר $O(n)$.

(3 נקי) ב. כתוב גרסה חדשה של מיון-מהיר M-QUICKSORT, המשתמשת בגרסה החדשה של שגרת החלוקה.

(5 נקי) ג. כתוב נוסחת נסיגה עבור זמן הריצה של M-QUICKSORT. אילו פתרונות מתקבלים במקרה הגרוע ובמקרה הטוב? תן הסבר קצר.

(10 נקי) ד. כתוב נוסחת נסיגה עבור זמן הריצה של M-QUICKSORT, בהנחה שבכל שלב של הרקורסיה מתקבל ערך של m שווה בקירוב ל- $n/2$.

אילו פתרונות מתקבלים במקרה הגרוע ובמקרה הטוב?

הערה: מותר להתעלם מבעיית שלמותם של הביטויים השונים.

3 עץ

נתונה טבלת גיבוב $T[1..m]$; כל תא של הטבלה מצביע אל מערך בגודל n. ברצוננו ליישם בכל אחד מ-m המערכים ערימה בינרית. ברצוננו להכניס לטבלת הגיבוב סדרה של n מפתחות בשתי שיטות.

(6 נקי) א. בשיטה הראשונה מכניסים את כל n המפתחות למבנה הנ״ל; אחרי שכולם הוכנסו, בונים את m הערימות. כתוב שגרה המבצעת את הפעולות האלה.

(6 נקי) ב. מהו זמן הריצה של השגרה בסעיף א׳, במקרה הגרוע? האם יתכן שמתקבל זמן ריצה טוב יותר במקרה הממוצע?

(6 נקי) ג. בשיטה השניה מכניסים את n המפתחות למבנה הנ״ל; אחרי כל הכנסת מפתח, מתקנים את הערימה המתאימה. כתוב שגרה שמבצעת את הפעולות האלה.

(7 נקי) ד. מהו זמן הריצה של השגרה בסעיף ג׳, במקרה הגרוע ובמקרה הממוצע?

4 מציאת איבר

נתון מערך $A[m + n]$, כאשר m ו-n משתנים בלתי-תלויים זה בזה. נתונה השגרה הבאה:

What (A, m, n)
- if $n = 1$
  - then return $A[m + 1]$
- $a_1 \leftarrow$ What $(m, \lfloor n/2 \rfloor)$
- $a_2 \leftarrow$ What $(m + \lfloor n/2 \rfloor, \lceil n/2 \rceil)$
- if $a_1 < a_2$
  - then return a_1
  - else return a_2

א. מה מבצעת השגרה? הסבר.

ב. כתוב נוסחת נסיגה עבור זמן הריצה של השגרה. פתור את נוסחת הנסיגה

5 עץ

נתון עץ בינרי T. הרמה ה- d של העץ מוגדרת כאוסף כל הצמתים הנמצאים בעומק d (יחסית לשורש). לדוגמא: הרמה 0 מכילה את השורש, הרמה 1 מכילה את הבנים של השורש ; הרמה $d+1$ מכילה את כל הבנים של הצמתים שברמה d.

סריקה ברמות של העץ T היא פעולה על העץ המחזירה את הצמתים של כל רמה משמאל לימין, החל מהרמה 0 ועד הרמה המכסימלית.

כתבו אלגוריתם המבצע סריקה ברמות של העץ T בזמן $O(n)$ (n הוא מספר הצמתים בעץ).

הסבירו מדוע האלגוריתם שהצעתם פועל נכון.

6 מיון, מציאת איבר

נתונה סדרה של n תת-קטעים

$$[a_1, b_1], [a_2, b_2], \ldots, [a_n, b_n]$$

של הקטע $[0,1]$.

כתבו אלגוריתם המחשב את מספר הזוגות (i, j) , $1 \le i, j \le n$ המקיימים את התנאי $b_i < a_j$.

(במילים אחרות, אנו רוצים לדעת כמה זוגות של תת-קטעים זרים זה לזה קיימים בסדרה.)

האלגוריתם חייב לרוץ בזמן $O(n \lg n)$ במקרה הגרוע.

7 מציאת איבר / מיון

בהינתן מערך S של n מספרים ממשיים שונים זה מזה ומספר ממשי נוסף z:

(10 נק') א. כתבו אלגוריתם שזמן ריצתו $\Theta(n\lg n)$, הסופר את מספר הזוגות (x,y) של איברים ב- S המקיימים את התנאי $x+y\le z$.

(10 נק') ב. כתבו אלגוריתם שזמן ריצתו $\Theta(n^2)$, הסופר את מספר השלשות (u,v,w) של איברים ב- S המקיימות את התנאי $u+v+w=z$.

הערה: מותר להתעלם מההבדל בין זוג סדור / לא סדור ובין שלשה סדורה / לא סדורה (כלומר, מותר לספור את הזוג (x,y) וגם את הזוג (y,x); בדומה עבור שלשות).

אין חובה לכתוב פסידוקוד.

8 עץ

נניח שממשים ערימות בינריות בעצים (בעזרת מצביעים) במקום במערכים.

ברצוננו למזג שתי ערימות שלמות: H_a בת 2^a-1 צמתים ו- H_b בת 2^b-1 צמתים, כך שהתוצאה תהיה גם היא ערימה בינרית בת $n=2^a+2^b-2$ צמתים.

(6 נק') א. כתבו אלגוריתם למיזוג שתי הערימות עבור המקרה $a=b$, שרץ בזמן $O(\lg n)$.

(6 נק') ב. כתבו אלגוריתם למיזוג שתי הערימות עבור המקרה $|a-b|=1$ שרץ בזמן $O(\lg n)$.

(8 נק') ג. כתבו אלגוריתם למיזוג שתי הערימות עבור a,b כלשהם, שרץ בזמן $O(\lg^2 n)$.

הערה: אין חובה לכתוב פסידוקוד.

9 מיון

נתונים שני מערכים $A[1..n]$ ו- $B[1..n]$. נתבונן בשגרה הבאה:

```
ZSORT (A)
   for i ← 1 to n
      do x ← 1
         for j ← 1  to n
            do if A[j] > A[i]
                  then  x ← x + 1
         B[x] ← A[i]
   for i ← n downto 1
      do A[n - i + 1] ← B[i]
```

א. הוכח שהשגרה ממיינת נכון את A.

ב. מהי סיבוכיות הזמן של השגרה? הסבר.

הערה: האיברים של A שונים זה מזה.

10 מיון

נתבונן באלגוריתמים מיון-הכנסה, מיון-מהיר, מיון-ערימה.

איזה משלושת האלגוריתמים הוא היעיל ביותר כאשר:

א. הקלט ממוין מראש בסדר עולה.

ב. הקלט ממוין מראש בסדר יורד.

ג. היעילות נמדדת באמצעות מספר ההחלפות של איברים בלבד (ז״א ההשוואות לא נספרות).

התשובות חייבות להיות מנומקות היטב.

11 עץ

נתון עץ בינרי כלשהו T עם ערכים מספריים (שלמים או ממשיים) המאוחסנים בצמתים.

א. כתוב אלגוריתם יעיל הבודק האם T הוא עץ חיפוש בינרי.

זמן הריצה הדרוש הינו $O(n)$, כאשר n הוא מספר הצמתים של T.

ב. כתוב אלגוריתם יעיל הבודק האם T הוא ערימה. זמן הריצה הדרוש הינו $O(n)$.

12
עץ

א. חשב את הגובה המינימלי ואת הגובה המקסימלי של עץ אדום-שחור בן 7 מפתחות.

ב. צייר שני עצים אדומים-שחורים בני 7 צמתים, כאשר גובהו של הראשון מינימלי וגובהו של השני מקסימלי.

13
מיון
מציאת איבר

א. הצע אלגוריתם, המקבל סדרה של שלמים חיוביים $x_1, x_2,...,x_n$ וקובע האם קיימים שני אינדקסים i ו- j כך ש- $x_i = (x_j)^2$. על האלגוריתם שהצעת להיות בעל תוחלת זמן ריצה $O(n)$.

ב. בהינתן שתי סדרות $X= x_1,...,x_n$ ו- $Y=y_1,...,y_n$, הצע אלגוריתם עם תוחלת זמן ריצה טוב ככל שתוכל, שבודק אם X הוא פרמוטציה של Y.

14
עץ

נתון שסידור ה- Preorder של עץ הינו (משמאל לימין):

F D H L W M R S K L

כמו כן, סידור ה- Inorder הינו:

H D W L F R K S L M

א. צייר עץ בינארי המקיים את שני הסידורים האלו.

ב. נתונים סידור ה- Preorder וסידור ה- Inorder של עץ בינארי מסוים. האם יכול להיות עץ בינארי שונה עם אותם סידורי Preorder ו- Inorder?
נמק את תשובתך!

ג. יהי T עץ חיפוש בינארי ו-X ערך הנמצא בעץ T. יהי T' העץ המתקבל מ-T ע"י הוצאת X והכנסתו חזרה. כלומר: $T' = INSERT(DELETE(T, X), X)$. מתי $T \equiv T'$?

א. תמיד
ב. רק כאשר ל-X יש בן אחד בלבד
ג. רק כאשר X הינו עלה
ד. אף פעם
ה. רק כאשר X הוא האיבר המקסימלי או המינימלי בעץ
ו. אין חוקיות ספציפית

15
עץ

א. בהינתן ערמה כמתואר בספר, הצע אלגוריתם יעיל ככל שתוכל להדפסת כל איברי הערמה שערכם קטן מערך x נתון. נתח את סיבוכיות האלגוריתם שהצעת.

ב. האם מערך ממוין בסדר הפוך הוא ערמה? הסבר את תשובתך.

16
מיון

נתבונן בגירסה של מיון-מיזוג הפועלת באופן הבא:

1) מחלקת את המערך לשלושה שלישים ומפעילה את גירסה זו של מיון-מיזוג על כל שליש באופן רקורסיבי;

2) ממזגת את השליש הראשון עם השני ואת התוצאה עם השלישי.

א. כתבו נוסחת נסיגה עבור המקרה הגרוע ביותר; הסבירו איך מגיעים לנוסחה;

ב. פתרו את הנוסחה וכתבו את התוצאה באמצעות סימון Θ.

17

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

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

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

כתבו אלגוריתם יעיל הקובע עבור כל אחד מ- n הקטעים האם הוא מוכל בתוך אחד הקטעים האחרים. מהי סיבוכיות האלגוריתם?

18 מיון

נתונה מטריצה בגודל $n \times m$ כאשר כל השורות והעמודות שלה ממוינות. בנוסף נתון ערך כלשהו z.

כתבו אלגוריתם יעיל הקובע את מיקומו של z בתוך המטריצה, או מדווח על כישלון החיפוש אם z לא נמצא במטריצה.

19 מיון, מציאת איבר

א. הציגו מערך $A[1...4]$ כך ש- $A[1]=A[2]$, אבל הסדר היחסי של שני איברים אלה בפלט הממוין של מיון-ערימה תלוי בערך של אחד משני האיברים הנוספים שבמערך המקורי.

ב. האם האלגוריתם מיון-ערימה הינו יציב?

20 עץ

עבור כל שני עצי חיפוש בינריים בעלי n צמתים T ו- T', הראו שניתן להגיע מ- T ל- T' באמצעות לכל היותר $2n-2$ רוטציות.

21 מיון

א' (5 נקודות) נתון המערך $[3,0,2,4,5,8,7,6,9]$ כפי שהוא נראה **אחרי** ביצוע שגרת החלוקה PARTITION.

אילו מהאיברים שלו היו יכולים לשמש כאיבר הציר בשגרת החלוקה? נמקו את תשובתכם.

ב' (20 נקודות) שגרת החלוקה PARTITION מופעלת על המערך $A[1..n]$ ויוצרת את המערך $B[1..n]$.

נתון מערך הפלט $B[1..n]$. כתבו אלגוריתם, יעיל ככל שאפשר, למציאת כל האיברים שהיו יכולים לשמש כאיבר הציר בשגרת החלוקה. נתחו את זמן הריצה שלו.

22 עץ

נתון עץ אדום-שחור T ; נסמן ב- ln את גודל התת-עץ השמאלי של השורש וב- rn את גודל התת-עץ הימני של השורש.

האם היחס $ln < 128 \cdot rn$ מתקיים תמיד? הוכיחו או הביאו דוגמה נגדית.

הערה: גודל תת-עץ (בעץ אדום-שחור) הינו מספר הצמתים הפנימיים שלו.

23
מיון

נתונים מערך $A[n]$ של מספרים ממשיים ומספר ממשי נוסף z.

(5 נק') א. כתוב שגרה (בפסֵידוקוד) למציאת אינדקס k $(1 \le k \le n)$ כך שהאיבר $A[k]$ יהיה מינימלי בין כל האיברים $A[i]$ המקיימים $z \le A[i]$; זמן הריצה הנדרש: $O(n)$.

(7 נק') ב. כתוב שגרה (בפסֵידוקוד) לפתרון אותה בעיה, בהנחה הנוספת שהמערך A ממוין בסדר עולה (לא יורד); זמן הריצה הנדרש: $O(\lg n)$.

24
מציאת איבר

נתונה השגרה $MED3$, המוצאת את ערכי המיקום ה- $\lfloor n/3 \rfloor$ וה- $\lfloor 2n/3 \rfloor$ במערך בגודל n בזמן לינארי ($MED3$ פועלת כקופסה שחורה ולא ידוע שום דבר נוסף עליה).

(13 נק') א. כתוב אלגוריתם שמבצע קריאות ל- $MED3$ והמוצא את ערך המיקום ה- k (k נתון, $1 \le k \le n$) בזמן לינארי. הוכח את זמן הריצה.

(7 נק') ב. האם ניתן לכתוב אלגוריתם שרץ בזמן לינארי והמוצא את **כל** ערכי המיקום (מהמינימום ועד למקסימום) באמצעות קריאות ל- $MED3$? הוכח או הפרך.

25
עץ

נתונה **ערימת מינימום** $H[n]$: לכל $i > 1$, $H[Parent(i)] \le H[i]$.

מכל איבר בערימה (פרט לשורש) מחסירים את ערך אביו. מתקבל מערך $D[n]$.

(8 נק') א. באיזה סדר עלינו להחסיר את האבות כך שיתאפשר שחזור הערימה המקורית **ללא** שימוש בזיכרון נוסף? כתוב שגרה לבנית המערך D ושגרה לשחזור המערך H.

(12 נק') ב. איך מתבצעת פעולת $INSERT(H,k)$ (הכנסת המפתח החדש k לערימה H) אם משתמשים בצורה D של הערימה? האם זמן הריצה נשמר? הוכח.

26
עץ

א. נתון עץ חיפוש בינרי T בעל N מפתחות; כל מפתח הוא מחרוזת המכילה m תווים לכל היותר. פעולת השוואה בין מחרוזות מבוססת על הסדר הלקסיקוגרפי ומבצעת מספר השוואות בין תווים כמספר התווים במחרוזת הקצרה יותר ועוד אחת. נתבונן בפעולות הבאות:

$SEARCH\ (T,s)$: חיפוש המחרוזת s בעץ T;

$INSERT\ (T,s)$: הכנסת המחרוזת s לעץ T;

$DELETE\ (T,p)$: מחיקת האיבר שאליו מצביע p מהעץ T.

ידוע שאחרי כל השוואה בין המחרוזת t שבעץ לבין המחרוזת s, מתבצעת השגרה $LCS\text{-}LENGTH(s,t)$, המחשבת את אורך התת-מחרוזת המשותפת הארוכה ביותר של s ושל t (עליך להתייחס אל "מחרוזת" כאל "סדרה" ואל "תת-מחרוזת" כאל "תת-סדרה").

כל שגרה תחזיר את הערך המקסימלי המתקבל מכל הקריאות לשגרה LCS-LENGTH.

מהם זמני הריצה האסימפטוטיים של שלוש הפעולות כפונקציות של m ושל N? הוכח כל טענה.

27 מיון

נתון מערך P באורך $m+n$ של מספרים שלמים. ידוע ש-m המספרים הראשונים שייכים לתחום $\left[1..n^2\right]$ ו-n המספרים האחרונים שייכים לתחום $\left[1..m^3\right]$.

תארו אלגוריתם למיון המערך P בזמן $O(m+n)$.

אין צורך לכתוב פסֵידוקוד.

28 מיון

נתון אלגוריתם מיון M הפועל באופן הבא:

בהינתן מערך A באורך n

- מוצאים את החציון שלו (בעזרת האלגוריתם האופטימלי);
- מבצעים חלוקה סביב החציון;
- ממיינים את החלק השמאלי בעזרת מיון-ערמה;
- מפעילים את האלגוריתם M על החלק הימני של החלוקה באופן רקורסיבי.

(5 נקי) א. הסבירו מדוע האלגוריתם M ממיין נכון את המערך.

(15 נקי) ב. כתבו נוסחת נסיגה עבור האלגוריתם M; פתרו את נוסחת הנסיגה.

(5 נקי) ג. השוו את האלגוריתם M לאלגוריתמי המיון הידועים (מיון-מיזוג, מיון-ערמה, מיון-מהיר) מבחינת הביצועים במקרה הגרוע ובמקרה הממוצע.

29 מיון

נתון מערך A המכיל N מספרים. ידוע שבין N המספרים קיימים רק k ערכים שונים זה מזה $(0<k<N)$; כלומר, חלק מהערכים מופיעים ב-A יותר מפעם אחת. נסמן ב- $m(a)$ את השכיחות של a ב-A; כלומר, $m(a)$ הוא מספר הפעמים שהערך a מופיע במערך A.

(12 נקי) א. כתבו שגרה למציאת איבר a במערך כך שהערך $m(a)$ יהיה מכסימלי; זמן הריצה הנדרש הוא $\Theta(N\cdot\lg k)$.

(13 נקי) ב. נתון בנוסף ערך מספרי z.

כתבו שגרה למציאת שני איברים a ו-b במערך כך שיתקיים התנאי $m(a)\cdot a+m(b)\cdot b=z$; זמן הריצה הנדרש הוא $\Theta(N\cdot\lg k)$.

אין צורך לכתוב פסֵידוקוד.

30 מציאת איבר מיון

נתון אלגוריתם M הפועל באופן הבא:

בהינתן מערך A של n מספרים

- מוצאים את החציון של A (בעזרת האלגוריתם האופטימלי);
- מבצעים את חלוקת המערך סביב החציון;
- בונים ערמת מינימום מתוך $\left\lceil\frac{n}{2}\right\rceil$ האיברים הגדולים;
- מפעילים את האלגוריתם M באופן רקורסיבי על $\left\lfloor\frac{n}{2}\right\rfloor$ האיברים הק

נקרא למבנה המתקבל S.

א. כתבו את האלגוריתם M בפסֵידוקוד.

ב. הוכיחו שזמן הריצה שלו ליניארי.

ג. הראו שעל מבנה הנתונים S ניתן לבצע את הפעולות הבאות:

FIND-OS (S, i): מציאת ערך המיקום ה- $\left(\left\lfloor\frac{n}{2^i}\right\rfloor+1\right)$ של S $\left(1\le i\le\lfloor\lg n\rfloor\right)$;

זמן הריצה $O(1)$;

DEL-OS (S,i): מחיקת ערך המיקום ה- $\left(\left\lfloor\frac{n}{2^i}\right\rfloor+1\right)$ של S $\left(1\le i\le\lfloor\lg n\rfloor\right)$;

זמן הריצה $O(\lg^2 n)$;

INSERT (S,z): הכנסת מפתח חדש z למבנה S; זמן הריצה $O(\lg^2 n)$.

אין צורך לכתוב פסֵידוקוד.

31
מציאת איבר

נתון מערך A באורך n של מספרים ממשיים **חיוביים**, שונים זה מזה.

ברצוננו למצוא שני אינדקסים $1 \le i, j \le n$, כך שמתקיים התנאי $(A[i])^2 = A[j]+1$.

(13 נק׳) א. כתבו שגרה למציאת שני האינדקסים, שזמן ריצתה $O(n\lg n)$ במקרה הגרוע.

(12 נק׳) ב. כתבו שגרה למציאת שני האינדקסים, **שתוחלת** זמן ריצתה $O(n)$.

32
מיון

(12 נק׳) א. נתונה סדרה של n מספרים. כתבו אלגוריתם שזמן ריצתו לינארי, המוצא וממיין את p האיברים הקטנים ביותר של הסדרה. ידוע לנו כי $p \le n/\lg n$.

(13 נק׳) ב. נתונה סדרה של n מספרים שלמים בתחום $[n..n^2+n-1]$. כתבו אלגוריתם שזמן ריצתו לינארי, הממיין את סדרת המספרים.

33
מיון

נתון מספר שלם חיובי **קבוע** c.

נבנה גרסה של האלגוריתם מיון-מיזוג הפועלת באופן הבא:

(1) המערך מחולק ל-c חלקים באורך $\lfloor n/c \rfloor$ או $\lceil n/c \rceil$ כל אחד; על כל חלק מופעלת גרסה זו של מיון-מיזוג באופן רקורסיבי;

(2) c החלקים ממוזגים כדי לקבל מערך ממוין.

(5 נק׳) א. הראו כיצד ניתן לבצע את מיזוג c החלקים בזמן לינארי.

(10 נק׳) ב. כתבו את נוסחת הנסיגה עבור המקרה הגרוע של האלגוריתם (הגרסה החדשה של מיון-מיזוג).

(10 נק׳) ג. פתרו את נוסחת הנסיגה והשוו בין זמני הריצה האסימפטוטיים של שתי הגרסאות של מיון-מיזוג (הגרסה מספר הלימוד והגרסה מהשאלה הזאת).

34
עץ

נתון עץ חיפוש בינרי T; נסמן ב-n את מספר האיברים ב-T.

(10 נקודות)

א׳ כתבו אלגוריתם למציאת שני צמתים x ו- y ב- T, המקיימים את התנאי $key[x]+key[y]=2\cdot key[root[T]]$. זמן הריצה הנדרש של האלגוריתם הינו $\Theta(n)$.

(15 נקודות)

ב׳ נניח עכשיו שהעץ T מאוזן.

לכל שני צמתים x ו-y ב-T, נסמן ב- $p(x,y)$ את האב הקדמון המשותף הנמוך ביותר של x ו-y ב-T (כלומר, x נמצא בתת-עץ השמאלי של $p(x,y)$ ו-y נמצא בתת-עץ הימני של $p(x,y)$, או להיפך).

כתבו אלגוריתם למציאת שני צמתים x ו- y ב- T, המקיימים את התנאי $key[x]+key[y]=2\cdot key[p(x,y)]$. זמן הריצה הנדרש של האלגוריתם הינו $\Theta(n\cdot\lg n)$.

רמז: שימו לב שמספר הרמות ב-T הוא $\Theta(\lg n)$.

35

מציאת איבר

נתון מערך $A[1..n]$ של מספרים.

כתבו אלגוריתם למציאת שלושה אינדקסים $1 \le i < j < k \le n$ כך שמתקיים $A[i] + A[k] = 2 \cdot A[j]$. זמן הריצה הנדרש הינו $\Theta(n^2)$.

36

מציאת איבר

נתונים מערך $A[1..n]$ של מספרים ממשיים ומספר טבעי m המקיים $2m < n$.

כתבו אלגוריתם הבודק האם **קיים** ב-A איבר z המקיים את שני התנאים הבאים:

(1) A מכיל לפחות $n-2m$ איברים קטנים מ-z;

(2) z מופיע יותר מ-m פעמים ב-A.

זמן הריצה הנדרש של האלגוריתם הינו $\Theta(n)$.

37

מציאת איבר

נתון מערך $A[1..n]$ של מספרים שלמים המקיימים את התנאי $0 \le A[i] < 65536$, $i = 1,\dots,n$.

כתבו אלגוריתם למציאת שני אינדקסים $i, j \in \{1,\dots,n\}$ כך שיתקיים $A[i] + A[j] = 65536$;

זמן הריצה הנדרש הינו $O(n)$.

38

נתונים במישור n מלבנים. המלבן ה-i מורכב מכל הנקודות (x, y) המקיימות את התנאים $0 \le x \le x_i$, $0 \le y \le y_i$, $i = 1,\dots,n$.

כתבו אלגוריתם לחישוב שטח איחוד כל n המלבנים; זמן הריצה הנדרש הינו $O(n \cdot \lg n)$.

39

מציאת איבר

נתון מערך $A[1..n]$ של מספרים שלמים. כתבו אלגוריתם שזמן ריצתו **ליניארי**, המחזיר את מספר האיברים במערך השווים לחציון.

40

עץ

נתונה ערמת מינימום H בת n איברים. נניח שכל שורה (רמה) בעץ בערמה ממוינת משמאל לימין, בסדר לא יורד.

א' (10 נקודות) כתבו אלגוריתם לא רקורסיבי המבצע חיפוש אחר ערך נתון z בין איברי הערמה; האלגוריתם יבצע את החיפוש באמצעות קריאה לחיפוש בינרי על כל אחת מהשורות. יש לכתוב את האלגוריתם בפסֵידוקוד.

ב' (10 נקודות) נתחו את זמן הריצה של האלגוריתם.

ג' (5 נקודות) הוכיחו את נכונות האלגוריתם.

41 מיון

ענו לשאלות הבאות ונמקו את תשובותיכם:

א' (13 נקודות) מהו זמן הריצה של האלגוריתם מיון-מהיר על המערך $A=[2,3,...,n,1]$?

ב' (12 נקודות) מהו זמן הריצה של האלגוריתם מיון-מהיר על המערך $A=[1,n,...,3,2]$?

הערה: מדובר באלגוריתם מיון-מהיר המוצג בספר הלימוד.

42 עץ

נתון עץ חיפוש בינרי T; נסמן ב- n את מספר הצמתים וב- h את גובה העץ. נסיר מהעץ צומת כלשהו y; המבנה $T-\{y\}$ מתפרק ל- m $(1\le m\le 3)$ עצי חיפוש בינריים.

א' (10 נקודות) תארו אלגוריתם לאיחוד m העצים הנ"ל לעץ חיפוש בינרי אחד U; זמן הריצה הנדרש: $O(h)$.

ב' (5 נקודות) באלו מקרים הגובה של U קטן מ- h ?

ג' (10 נקודות) מהו הגובה המכסימלי האפשרי של U כפונקציה של h ? באלו מקרים מתקבל גובה זה?

הערה: שימו לב שלצומת y שמוסר מהעץ T יכולים להיות שני בנים ולכן אי אפשר להשתמש באלגוריתם המתואר בספר הלימוד למחיקת צומת מעץ חיפוש בינרי.

43 עץ, מציאת איבר

נתונות שתי רשימות של מספרים ממשיים, S בת m איברים ו- T בת n איברים; בנוסף, נתון מספר ממשי z.

כתבו אלגוריתם הקובע האם קיימים $x\in S$, $y\in T$, כך שמתקיים $x+y=z$. זמן הריצה הנדרש של האלגוריתם הינו $\Theta((m+n)\cdot\lg(\min(m,n)))$.

44 מציאת איבר

א' (10 נקודות) נתון מערך של מספרים $A[1..n]$. ידוע שקיים אינדקס m, $1<m<n$, כך שמתקיימים התנאים $A[1]>...>A[m-1]>A[m]<A[m+1]<...<A[n]$ (כלומר, התת-מערך $A[1..m]$ ממוין בסדר יורד והתת-מערך $A[m..n]$ ממוין בסדר עולה).

כתבו שגרת פסֵידוקוד המבצעת חיפוש אחר האינדקס m והמחזירה אותו; זמן הריצה הנדרש הינו $O(\lg n)$.

ב' (15 נקודות) נתון מערך של מספרים $A[1..n]$. ידוע שקיימים אינדקסים p,q, $1<p<q<n$, כך שמתקיימים התנאים $A[1]>...>A[p-1]>A[p]=...=A[q]<A[q+1]<...<A[n]$ (כלומר, התת-מערך $A[1..p]$ ממוין בסדר יורד והתת-מערך $A[q..n]$ ממוין בסדר עולה).

כתבו שגרת פסֵידוקוד המבצעת חיפוש אחר האינדקסים p,q והמחזירה אותם; זמן הריצה הנדרש הינו $O(\lg n)$.

45

מיון

א' (12 נקודות) נתון מערך $A[1..n]$ של מספרים שלמים המקיימים את התנאים $n^2 \le A[i] \le n^3 + n^2 - 1$ לכל $i = 1,\dots,n$.

כתבו שגרה למיון המערך בזמן ליניארי.

ב' (13 נקודות) נתון מערך $Q[1..n]$ של מספרים ממשיים; ידוע שלא קיימים במערך יותר מ- $\lg^2 n$ ערכים שונים זה מזה.

הראו שניתן להכניס כל איברים המערך לעץ אדום-שחור בזמן $O(n \cdot \lg\lg n)$.

46

מיון

נתונה קבוצה $P = \{p_1,\dots,p_n\}$ של נקודות במישור הממשי. נתונים בנוסף שני מספרים ממשיים a ו- b.

כתבו אלגוריתם למציאת שתי נקודות $p_i, p_j \in P$, $i \ne j$, $p_i = (x_i, y_i)$, $p_j = (x_j, y_j)$, המקיימות את התנאי $|ax_i + by_i| = |ax_j + by_j|$ (או קביעה שאין שתי נקודות כאלה). זמן הריצה הנדרש של האלגוריתם הוא $O(n \cdot \lg n)$.

47

מציאת איבר

נתונים מערך $A[1..n]$ של מספרים ממשיים ומספר טבעי k המקיים את התנאי $3k < n$.

כתבו אלגוריתם הבודק האם **קיים** איבר z ב- A המקיים את שני התנאים הבאים:

(1) A מכיל לפחות $n - 3k$ איברים קטנים מ- z;

(2) z מופיע יותר מ- k פעמים ב- A.

זמן הריצה הנדרש של האלגוריתם הינו $\Theta(n)$.

פתרונות לאלגוריתמים וזמני ריצה

1

א. שגרה לחיפוש הערך z במערך $A[1..n]$:

```
FIND-VALUE (A, n, z)
if  A[1] ≤ z ≤ A[k]
    then  i ← BINARY-SEARCH (A, 1, k, z)
          if  i > 0
             then return A[i]
i ← LINEAR-SEARCH (A, k + 1, n, z)
if  i > 0
   then return A[i]
return 0
```

ב. במקרה הגרוע יתבצעו גם החיפוש הבינרי וגם החיפוש הלינארי, ולכן זמן הריצה יהיה:
$T(m,\ n) = O(\lg k + m) = O(\lg(n-m) + m)$.

ג. עבור $m = O(\lg n)$.

ד. $T(m,n) = O(\lg n) \quad \Leftarrow \quad m = O(\lg n)$

ולכן ניתן לבצע במקרה זה $O(\frac{n^2}{\lg n})$ פעולות חיפוש בזמן כולל של $O(n^2)$.

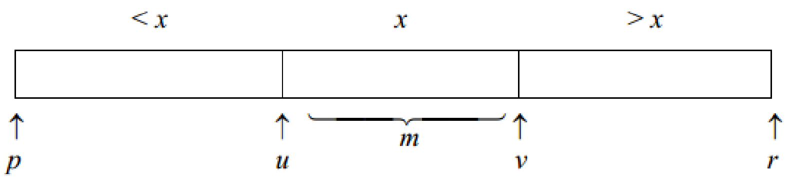
$$m = O(\frac{n}{\lg n}) \quad \Rightarrow \quad T(m,\ n) = O(\lg n + \frac{n}{\lg n}) = O(\frac{n}{\lg n})$$

ולכן ניתן לבצע במקרה זה $O(n \cdot \lg n)$ פעולות חיפוש בזמן כולל של $O(n^2)$.

$$m = O(n) \quad \Rightarrow \quad T(m,n) = O(n)$$

ולכן ניתן לבצע במקרה זה $O(n)$ פעולות חיפוש בזמן כולל של $O(n^2)$.

2



א. הרעיון הוא לקרוא קודם כל ל- PARTITION כש-x הוא איבר החלוקה.

לאחר מכן נתקן את שני התת-מערכים $A[p..\ q-1]$ ו- $A[q+1..r]$, כך שב- $A[p..q-1]$ כל האיברים השווים ל-x יהיו בצד ימין, וב- $A[q+1..r]$ כל האיברים השווים ל-x יהיו בצד שמאל.

```
M-PARTITION (A, p, r)
q ← PARTITION (A, p, r)
i ← p − 1 , j ← q                ▸ fixing the left sub-array
while i < j
   do repeat  j ← j − 1
      until   A[j] ≠ x
      repeat  i ← i + 1
      until   A[i] = x
      if  i < j
         then exchange (A[i], A[j])
u ← j
i ← q , j ← r + 1                ▸ fixing the right sub-array
while i < j
   do repeat  j ← j − 1
      until   A[j] = x
      repeat  i ← i + 1
      until   A[i] ≠ x
      if  i < j
         then exchange (A[i], A[j])
v ← i
return u, v

M-QUICKSORT (A, p, r)
if p < r
   then  u, v ← M-PARTITION (A, p, r)
         M-QUICKSORT (A, p, u)
         M-QUICKSORT (A, v, r)
```

ב. הגרסה החדשה של מיון-מהיר:

```
M-QUICKSORT (A, p, r)
if p < r
   then  u, v ← M-PARTITION (A, p, r)
         M-QUICKSORT (A, p, u)
         M-QUICKSORT (A, v, r)
```

ג. נסמן ב- n_L וב- n_R את גודל התת-מערך השמאלי וגודל התת-מערך הימני, בהתאמה.

כלומר: $n_L + n_R = n - m$

נוסחת הנסיגה: $T(n) = T(n_L) + T(n_R) + O(n)$

המקרה הגרוע: $m = 1,\ n_L = 1,\ n_R = n-2$ (ללא הגבלת הכלליות)

$T(n) = T(n-2) + O(n) = O(n^2)$

המקרה הטוב: $m = n,\ n_L = 0,\ n_R = 0$

$T(n) = O(n)$

ד. כאשר $m \cong \frac{n}{2}$ אז גם $n_L + n_R \cong \frac{n}{2}$.

המקרה הגרוע: $n_L = 1,\quad n_R \cong \frac{n}{2}$ (ללא הגבלת הכלליות)

$T(n) = T(\frac{n}{2}) + O(n) = O(n)$

המקרה הטוב: $n_L = n_R \cong \frac{n}{4}$

$T(n) = 2T(\frac{n}{4}) + O(n) = O(n)$

א. צריך לבצע HASH-INSERT n פעמים, ולאחר מכן לקרוא ל- BUILD-MAX-HEAP m פעמים:

```
for  i ← 1 to n
   do  read (k)
       HASH-INSERT (T[h(k)], k)        ▸ insert to the array at index h(k)
for  i ← 1 to m
   do BUILD-MAX-HEAP (T[i])
```

ב. במקרה הגרוע כל n האיברים יוכנסו לאותו מערך.

הכנסת האיברים לטבלה: $O(n)$

בניית הערימה: $O(n)$

סה"כ: $O(n)$

במקרה הממוצע יהיו m ערימות בעלות $\frac{n}{m}$ איברים כל אחת.

הכנסת האיברים לטבלה: $O(n)$

בניית הערימות: $m \cdot O(\frac{n}{m}) = O(n)$

סה"כ: $O(n)$

ג. במקרה זה צריך לקרוא לשגרה MAX-HEAP-INSERT n פעמים:

```
for i ← 1 to n
   do  read (k)
       MAX-HEAP-INSERT (T[h(k)], k)
```

ד. במקרה הגרוע כל n האיברים יוכנסו לאותו מערך.

הכנסת איבר בודד: $O(1)+O(\lg n)=O(\lg n)$

סה"כ: $n \cdot O(\lg n) = O(n \cdot \lg n)$

במקרה הממוצע יהיו m ערימות בעלות $\frac{n}{m}$ איברים כל אחת.

הכנסת איבר בודד: $O(1)+O(\lg \frac{n}{m})=O(\lg \frac{n}{m})$

סה"כ: $n \cdot O(\lg \frac{n}{m}) = O(n \cdot \lg \frac{n}{m})$

(א) השגרה מחזירה את המינימום בתת-מערך $A[m+1..m+n]$

(ב) הפרמטר m נשאר ללא שינוי; מתקבלת נוסחת הנסיגה

$$T(n) = \begin{cases} \Theta(1), & n = 1 \\ 2T(n/2) + \Theta(1), & n > 1 \end{cases}$$

שפתרונה (שיטת האב, מקרה 1) הוא $T(n) = \Theta(n)$.

5

נשתמש בתור עזר Q שיכיל את האיבר הבא לסריקה על-פי רמות:

```
BFS(T)
1    ENQUEUE(Q, root[T])
2    while not ISEMPTY(Q)
3        do    p ← DEQUEUE(Q)
4              if left[p] ≠ NIL
5                   then ENQUEUE(Q, left[p])
6              if right[p] ≠ NIL
7                   then ENQUEUE(Q, right[p])
8              print key[p]
```

האלגוריתם מכניס לתור את שורש העץ, ולאחר מכן מבצע לולאה המסתיימת כאשר התור מתרוקן. בכל איטרציה של הלולאה מוציאים מהתור את האיבר שבראש התור, מדפיסים אותו ומכניסים לתור את שני בניו (אם הם קיימים), הבן השמאלי ואחריו הבן הימני.

נוכיח את נכונות האלגוריתם.

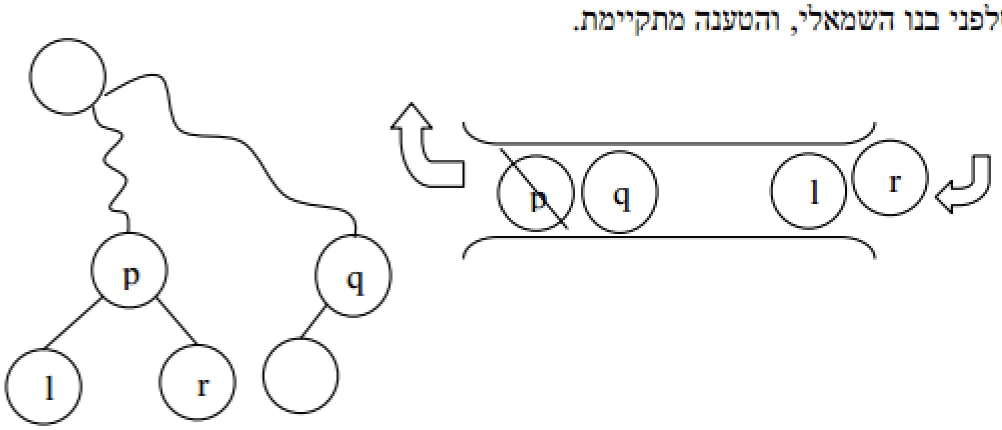
טענה: לפני כל איטרציה של הלולאה בשורה 2, התור מכיל את הצמתים מהצומת שבראש התור ועד לצומת שלפני בנו השמאלי, בסדר המוגדר על-ידי סריקה לפי רמות.

נוכיחאת הטענה באינדוקציה על מספר האיטרציות.

לפני האיטרציה הראשונה, התור מכיל אך ורק את השורש והטענה מתקיימת.

נניח כעת כי הטענה מתקיימת לפני איטרציה i והתור מכיל את צמתי העץ בסריקה לפי רמות החל בצומת שבראש התור (להלן p) ועד לצומת שלפני בנו השמאלי בסריקה לפי רמות. נפריד לשני מקרים:

1. p אינו הצומת האחרון ברמה שלו. נסמן ב- q את האיבר הנמצא בתור אחרי p. עפ"י הנחת האינדוקציה, זהו האיבר הנמצא מימין ל- p בעץ. אחרי הוצאת p מהתור מכניסים לתור את שני בניו. על-פי הנחת האינדוקציה, התור לפני הוצאת p הכיל את כל הצמתים בעץ שבין p לבין הצומת שלפני בנו השמאלי של p (בסדר המוגדר על-ידי סריקה לפי רמות). כעת, בניו של p נמצאים בעץ משמאל לבניו של q (ראו באיור), ולכן לפני האיטרציה הבאה התור יכיל את כל הצמתים בעץ שבין q לבין הצומת שלפני בנו השמאלי, והטענה מתקיימת.



5 המשך

2. p הוא הצומת האחרון ברמה שלו. במקרה זה בניו של p הם האחרונים ברמה שלהם. נסמן ב- q את האיבר הנמצא בתור אחרי p. עפ"י הנחת האינדוקציה, זהו האיבר הראשון ברמה הבאה בעץ והתור מכיל את כל הצמתים בעץ שבין p לבין הצומת שלפני בנו השמאלי של p (בסדר המוגדר על-ידי סריקה לפי רמות). שני בניו של q הם הראשונים ברמה שמתחת לבנים של p, ולכן לאחר הוצאת p והכנסת שני בניו התור יכיל את כל הצמתים בעץ שבין q לבין הצומת שלפני בנו השמאלי. כלומר, הטענה מתקיימת.

קיבלנו שאיברי התור מסודרים בדיוק בסדר המוגדר ע"י סריקת לפי רמות. מכיוון שהאיברים מוצאים מהתור ומודפסים זה אחר זה, האלגוריתם מבצע סריקה ברמות של העץ.

נשים לב שכל צומת בעץ מוכנס לתור פעם אחת ומוצא מהתור פעם אחת, ולכן זמן הריצה של האלגוריתם הוא $O(n)$, כנדרש.

6

נניח שכל הנקודות שונות זו מזו ולכל i $a_i < b_i$.

תיאור האלגוריתם:

1. נבנה מערך בן $2n$ תאים, כאשר כל תא מכיל שני שדות – *value* ו- *type*. השדה *value* יכיל את ערך הנקודה, והשדה *type* יכיל את הערך a-type אם הנקודה היא מסוג a_i ואת הערך b-type אם היא מסוג b_i.
2. נמיין את המערך לפי השדה *value* (למשל, באמצעות מיון-ערמה).
3. נעבור על המערך משמאל לימין ונשמור את מספר הנקודות מסוג b_i שחלפנו על פניהן במשתנה *BCounter*. בכל פעם שנגיע לנקודה מסוג a_i, נוסיף את *BCounter* למספר הזוגות של תת-קטעים זרים, מפני שהנקודה a_i גדולה מכל הנקודות מסוג b_i שלפניה.

להלן האלגוריתם:

```
COUNTDISJOINT(A, B)
1   n ← length[A]
2   for i ← 1 to n
3       do    value[C[i]] ← A[i]
4             type[C[i]] ← a-type
5             value[C[n+i]] ← B[i]
6             type[C[n+i]] ← b-type
7   HEAPSORT(C)
8   BCounter ← 0
9   PairCounter ← 0
10  for i ← 2 to n−1
11      do if type[C[i]] = b-type
12            then Bcounter ← Bcounter + 1
13            else PairCounter ← PairCounter + Bcounter
14  return PairCounter
```

ננתח את סיבוכיות זמן הריצה של האלגוריתם:

זמן הריצה של לולאת ה-for בשורה 2 הוא $O(n)$, מכיוון שבכל איטרציה של הלולאה מבוצע מספר קבוע של פעולות.

זמן הריצה של שורה 8 הוא:

$$O(2n\lg 2n) = O(n\lg 2 + n\lg n) = O(n\lg n)$$

זמן הריצה של לולאת ה-for בשורה 10 הוא גם-כן $O(n)$.

בסה"כ סיבוכיות זמן הריצה של האלגוריתם היא:

$$O(n) + O(n\lg n) + O(n) = O(n\lg n)$$

סעיף א:

נשתמש בגרסה שונה מעט של חיפוש בינרי: במקרה שהאיבר שמחפשים אינו נמצא במערך הממוין, אז השגרה תחזיר את האינדקס של האיבר הכי גדול שקטן ממנו. אם אין כזה, היא תחזיר 0. זמן הריצה של גרסה זו הוא גם-כן $O(\lg n)$.

נתאר את האלגוריתם הדרוש:
1. נמיין את S באמצעות מיון אופטימלי. (למשל מיון-ערמה)
2. עבור כל איבר $S[i]$ ב-S, נחפש את $z-S[i]$ בתוך S באמצעות שגרת החיפוש המורחבת, ונסכום את כל האינדקסים שקיבלנו.
3. נחזיר את הסכום.

נסביר מדוע האלגוריתם מבצע את הדרוש וננתח את סיבוכיות הזמן שלו:
שגרת החיפוש תחזיר את האינדקס הגדול ביותר j שעבורו $S[j] \le z-S[i]$. כלומר, $S[j]+S[i] \le z$. מכיוון שלכל $j^* \le j$ מתקיים $S[j^*] \le S[j]$, נקבל שיש בדיוק j איברים ב-S שעבורם $S[j]+S[i] \le z$. אם נסכום עבור כל i את האינדקסים הללו, נקבל את התוצאה הדרושה.

הערה: בספירה יכולים להיות גם זוגות מהצורה (j,j). ניתן להרחיב את האלגוריתם כך שיטפל במקרה קצה זה בלי להשפיע על הסיבוכיות.

סיבוכיות הזמן של המיון בשלב 1 היא $\Theta(n\lg n)$.
בשלב 2 קוראים לשגרת העזר n פעמים, ולכן סיבוכיות הזמן של שלב 2 היא $\Theta(n\lg n)$.
בסה"כ סיבוכיות הזמן של האלגוריתם היא $\Theta(n\lg n)$, כנדרש.

סעיף ב:

ראשית, נתאר שגרה שסופרת במערך ממוין A את מספר הזוגות (a,b) המקיימים $a+b=c$ עבור c נתון.

1. נצביע על שני קצות המערך.
2. נבצע עד אשר שני המצביעים נפגשים:
   2.1. אם סכום האיברים שמצביעים עליהם גדול מ-c, נזיז את המצביע הימני שמאלה.
   2.2. אם הסכום קטן מ-c, נזיז את המצביע השמאלי ימינה.
   2.3. אם הסכום שווה ל-c, נגדיל את המונה ונזיז את המצביע הימני שמאלה.

נסביר את פעולת השגרה וננתח את סיבוכיותה:
אם הסכום גדול מ-c: כל איבר הנמצא מימין למצביע הימני גדול יותר מהאיבר שעליו מצביע המצביע הימני, ולכן לא ייתכן שהסכום שלו ושל האיבר שעליו מצביע המצביע השמאלי יהיה שווה ל-c. לכן מזיזים את המצביע הימני שמאלה.
בצורה דומה מטפלים במקרה שהסכום קטן מ-c.
אם הסכום שווה ל-c, אז מצאנו זוג אחד, וממשיכים לסרוק את המערך בחיפוש אחר זוגות נוספים. (הבחירה להזיז את המצביע הימני שמאלה היא שרירותית.)

ניתוח האלגוריתם פשוט: בסה"כ עוברים על כל איבר במערך פעם אחת פעם אחת – עם המצביע הימני או עם השמאלי. מכיוון שבכל איטרציה מבצעים מספר קבוע של פעולות, הרי שסיבוכיות האלגוריתם היא $\Theta(n)$, כאשר $n = length[A]$.

וכעת, נתאר אלגוריתם לבעיה הנתונה:

1. נמיין את המערך באמצעות מיון אופטימלי. (למשל מיון-ערמה)
2. עבור כל איבר $S[i]$, נספור את מספר הזוגות במערך שסכומם $z-S[i]$ ונסכם את כל המספרים הללו.
3. נחזיר את הסכום שקיבלנו.

שוב, יהיו זוגות לא חוקיים שספרנו, אבל ניתן לטפל במקרי קצה אלו בלי לשנות את מהות האלגוריתם.

כעת ננתח את סיבוכיות זמן הריצה:
שלב 1 – $\Theta(n\lg n)$.
שלב 2 – $\Theta(n^2)$.
שלב 3 – $O(1)$
בסה"כ סיבוכיות זמן הריצה היא $\Theta(n^2)$, כנדרש.

<u>סעיף א:</u>

1. ניקח את העלה הכי ימני ב- H_b ונוציא אותו מהערמה.
   נשים לב, שניתן להגיע לאיבר האחרון ב- H_b בזמן $O(b)$. (כיצד?)
2. נשריש בו את H_a כתת-עץ שמאלי, ואת H_b כתת-עץ ימני.
3. "נערמם" את הערמה שקיבלנו החל מהשורש (כלומר, נקרא לשגרה MAX-HEAPIFY עם האינדקס 1).

נשים לב שאחרי שלב 2 מתקבל עץ בינרי כמעט שלם, שכן H_a, H_b היו עצים שלמים.
כמו-כן, H_a, H_b היו ערמות חוקיות, ולכן לאחר הערמום נקבל ערמה חוקית.

סיבוכיות שלב 1 היא $O(\lg n)$, סיבוכיות שלב 2 היא $O(1)$ וסיבוכיות שלב 3 היא $O(\lg n)$.
בסה"כ סיבוכיות האלגוריתם היא $O(\lg n)$.

<u>סעיף ב:</u>

נניח ללא הגבלת הכלליות כי $a = b+1$.
1. הפעם, ניקח את העלה הכי ימני ב- H_a ונוציא אותו מהערמה.
2. נשריש בו את H_a כתת-עץ שמאלי, ואת H_b כתת-עץ ימני.
3. נערמם את הערמה שקיבלנו מהשורש.

שוב, נשים לב שאחרי שלב 2 מתקבל עץ בינרי כמעט שלם. זה נובע מכך שהגובה של H_a גדול ב-1 מהגובה של H_b. לכן, אחרי שמסירים את העלה הכי ימני מ- H_a ו"מחברים" את שתי הערמות באופן שתואר לעיל, מתקבל עץ כמעט שלם שגובהו גדול ב-1 מהגובה של H_a. לאחר שנבצע ערמום, נקבל ערמה חוקית שמהווה מיזוג של שתי הערמות המקוריות.

ניתוח זמן הריצה:
שלב 1 – $O(\lg n)$.
שלב 2 – $O(1)$.
שלב 3 – $O(\lg n)$.

לכן בסה"כ נקבל שזמן הריצה הוא $O(\lg n)$.

<u>סעיף ג:</u>

שוב, נניח ללא הגבלת הכלליות כי $a \ge b$.

אם $a = b$, נשתמש בסעיף א.

אם $a = b+1$, נשתמש בסעיף ב.

אחרת נפעל באופן הבא:

מהשורש של H_a נפנה שמאלה $a-b$ פעמים.

מגיעים לשורש של ערמה בגובה b. נסמן אותה ב- A_1. נמזג את A_1 עם H_b באמצעות האלגוריתם מסעיף א'. הערמה הממוזגת (להלן, M) מחליפה את A_1 ב- H_a.

נסמן ב- x את האיבר הנמצא בשורש של M. נשים לב, ש- H_a הוא עץ כמעט שלם בגובה $a+1$, אולם ייתכן ש- x הגיע מ- H_b ולכן ייתכן שהאבות הקדמונים של x ב- H_a קטנים ממנו (כמובן, ייתכן גם שהם קטנים מאיברים נוספים שהגיעו מ- H_b).

כדי להפוך את H_a לערמה חוקית, נעלה עם x במעלה הערמה, עד שנגיע לאיבר שאינו קטן מ- x.

בכל שלב בלולאה נחליף את x עם אביו, ולאחר מכן נמצא מקום מתאים בערמה עבור האב.

מציאת המקום המתאים עבור האב תתבצע באופן הבא: נשווה בין האב לבין בנו השמאלי, ונחליף ביניהם אם הבן השמאלי גדול יותר. כאשר נגיע עם האב לשורש של M נקרא ל- MAX-HEAPIFY.

סיבוכיות הזמן: ל- x יש $a-b$ אבות הקדמונים בערמה ולכן נצטרך לחזור על התהליך $a-b$ פעמים לכל היותר. הזמן הדרוש למציאת מקום בערמה עבור כל אב קדמון של x הוא $O(\lg n)$, כגובה הערימה.

לפיכך, סיבוכיות הזמן של האלגוריתם היא: $(a-b)\cdot O(\lg n) = O(\lg^2 n)$

(כדי לראות את הדברים בצורה מוחשית, כדאי לצייר דוגמה. למשל, עם $a=5$ ו- $b=3$.)

9 א. בשלב הראשון, השגרה מחשבת לכל איבר $A[i]$ את מספר האיברים **הגדולים** ממנו במערך (ועוד אחד), ומציבה מספר זה במשתנה x. לאחר מכן $A[i]$ מוצב במקום ה-x במערך B.

המערך B המתקבל הוא ממוין בסדר יורד, מפני שבמקום $B[1]$ נמצא האיבר הגדול ביותר ב-A, במקום $B[2]$ נמצא האיבר השני בגודלו ב-A וכך הלאה.

בשלב השני, השגרה מעתיקה את איברי B חזרה למערך A בסדר הפוך, ולכן מתקבל ב-A מערך ממוין.

ב. סיבוכיות הזמן של השגרה היא $O(n^2)$, מפני שהשלב הראשון בשגרה מורכב משתי לולאות מקוננות, שכל אחת מהן מתבצעת בדיוק n פעמים.

10 א. **קלט ממוין בסדר עולה**: מיון-הכנסה יהיה היעיל ביותר, מפני שזמן הריצה שלו יהיה $O(n)$.

זמן הריצה של מיון-מהיר יהיה $\Theta(n^2)$ ושל מיון-ערימה $\Theta(n \lg n)$.

(זמן הריצה של מיון-ערימה הוא **תמיד** $\Theta(n \lg n)$ – ראו את תרגילים 6.4-4 ו- 6.4-5 בספר.)

ב. **קלט ממוין בסדר יורד**: מיון-ערימה יהיה היעיל ביותר – זמן ריצה של $\Theta(n \lg n)$.

זמני הריצה של מיון-הכנסה ושל מיון-מהיר יהיו $\Theta(n^2)$.

ג. **כאשר סופרים רק החלפות**:

קלט ממוין בסדר עולה: מיון-הכנסה יהיה היעיל ביותר – במהלך המיון לא תתבצע אף החלפה.

קלט ממוין בסדר יורד: מיון-ערימה יהיה היעיל ביותר – במהלך המיון יתבצעו $\Theta(n \lg n)$ החלפות. במיון-הכנסה יתבצעו $\Theta(n^2)$ החלפות.

במיון מהיר – בכל רמה זוגית של עץ הרקורסיה (החל מרמה 0) תתבצע החלפה אחת ויוחזר הערך $q = 1$; בכל רמה אי-זוגית יתבצעו $r - p + 1$ החלפות ויוחזר הערך $q = r$.

מספר ההחלפות הכולל יהיה לפיכך: $1 + (n-1) + 1 + (n-3) + \ldots + 1 = \Theta(n^2)$.

11 א. אלגוריתם איטרטיבי הבודק אם עץ בינרי T הוא עץ חיפוש בינרי:

```
CHECK-BST (T)
x ← TREE-MINIMUM (T)
i ← 1, bst ← TRUE
while i ≤ n – 1 and bst = TRUE
    do y ← TREE-SUCCESSOR (x)
        if key[x] ≤ key[y]
            then i ← i +1
                x ← y
            else bst ← FALSE
return bst
```

לפי תרגיל 12.2-7 בספר, זמן הריצה של האלגוריתם הוא $\Theta(n)$.

הערה: אפשר גם לבצע סריקה תוכית רגילה של העץ ולבדוק אם המערך המתקבל הוא ממוין.

ב. אלגוריתם רקורסיבי הבודק אם עץ בינרי T מקיים את תכונת הערימה:

```
CHECK-HEAP (T)
if T = NIL
  then return TRUE
  else if (left[T] ≠ NIL and key[left[T]] > key[T]) or
        (right[T] ≠ NIL and key[right[T]] > key[T])
        then return FALSE
        else return CHECK-HEAP (left[T]) and CHECK-HEAP (right[T])
```

בכל צומת בעץ מתבצעות שתי קריאות רקורסיביות ולכן זמן הריצה הוא $\Theta(n)$.

12

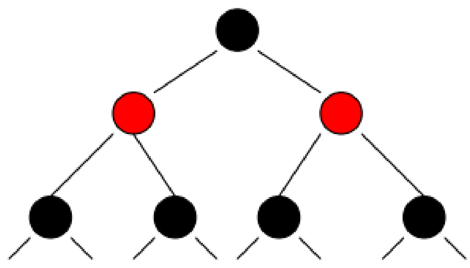
א. הגובה של עץ אדום-שחור המכיל שבעה צמתים פנימיים יהיה מינימלי כאשר העץ יהיה עץ בינרי **שלם**. גובה העץ במקרה זה הוא $\lfloor \lg_2 n \rfloor = \lfloor \lg_2 15 \rfloor = 3$. (ראו את הציור בסעיף ב').

n מציין כאן את מספר כל צמתי העץ, כולל העלים.

הגובה המקסימלי של עץ אדום-שחור המכיל שבעה צמתים פנימיים הוא 4 (ראו בסעיף ב').

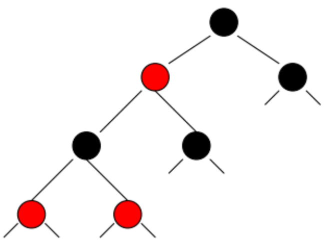
גובה העץ לא יכול להיות גדול מ-4, מפני שאז התנאי הנזכר בתרגיל 13.1-5 בספר לא יתקיים עבור השורש.

ב. דוגמה לעץ אדום-שחור המכיל שבעה צמתים פנימיים שגובהו מינימלי :



גובה העץ : 3

דוגמה לעץ אדום-שחור המכיל שבעה צמתים פנימיים שגובהו מקסימלי :



גובה העץ : 4

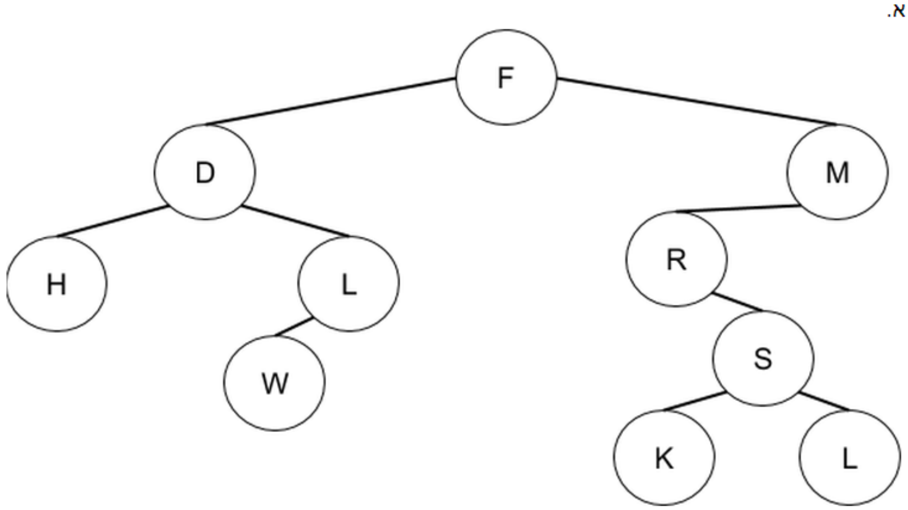
13

א. נייצג את קלט המספרים השלמים במערך A, ונשתמש במערך עזר B שגודלו יותר מ max{x1...,xn}.
נעבור בריצה O(n) על איברי המערך A ונשמור כל איבר xi במקום B[xi]. B[xi]
נעבור בריצה נוספת O(n) על איברי המערך A, אם xj∈N וגם B[xj]=xj אז החזר אמת.
*הסבר ודגשים - לא איתחלנו את המערך B והשתמשנו בשיטת גיבוב כדי לוודא גם שהערך חוקי.
האלגוריתם מבצע בסיבוכיות O(n)

ב. נייצג כל סדרה במערך X ו Y בהתאמה ונשתמש במערך A שגודלו יותר מ max{x1,..,xn,y1,..,yn}.
נעבור בריצה O(n) על איברי המערך X ונוסיף כל איבר xi מ X למחסנית שנמצאת ב A[xi].
נעבור בריצה נוספת O(n) על איברי המערך Y, אם המחסנית ב A[yi] לא ריקה וגם A[yi]=yi אז
הוצא איבר מהמחסנית והמשך ל i+1, אחרת החזר 'שקר'.
אם סיימנו לרוץ על כל איברי Y אז החזר 'אמת'

14

א.



*הלכתי על הסידור בPreorder ובניתי את העץ לפי רמזים מinorder

ב' אין עץ בינארי שונה עם אותם סידורים preorder ו inorder מכיוון שהאלגוריתם הרקורסיבי מחלק בכל שלב את הבעיה לתתי עצים, כלומר F המופיע ראשון בסידור הpreorder אומר שF שורש תת העץ ובסידור הinorder F מחלק באופן מוחלט את הצמתים בסידור inorder לתת העץ הימני ותת העץ השמאלי לפי כל הצמתים שמימין לF וכל הצמתים שמשמאל בהתאמה.

ג' התשובה: רק כאשר X הינו עלה. מחיקה של עלה מעח"ב לא משנה את מבנה העץ וביצוע חיפוש המיקום והכנסת הערך תתבצע לאותו המקום. בכל מקרה אחר שX לא עלה הכנסתו מחדש תגרום לו להיות עלה.

15 א' במקרה הגרוע ביותר הערך xהוא הערך שבשורש, במקרה זה בכל מקרה נאלץ לרוץ על כל מבנה הנתונים (ערימה) כדי להדפיס את כל האברים (כולם קטנים מהשורש) לכן אסימפטוטית הסיבוכיות היא O(n)ואפשר פשוט לעבור על המערך (שעליו ממומשת הערימה) ולהדפיס את האיבר אם הוא קטן מ.x

ב' מערך ממוין בסדר הפוך מקיים את כל תכונות הערימה, שכן כל צומת בערימה גדול משני בניו.

16 $T(n) = 3T(n/3) + O(n)$

א' אזי נפתור את נוסחאת הנסיגה לפי נוסחאת האב $a = 3 \quad b = 3 \qquad n^{log_3 3} = n$

ב' לכן זה מתאים למקרה השני של נוסחאת האב והפתרון הוא $T(n) = \Theta(nlogn)$

17 נגדיר לצורך האלגוריתם מבנה נתונים בסיסי המבוסס על עץ א"ש ונרחיב כך שכל צומת בעץ יכיל גם את הערך b המינימאלי בתת העץ השמאלי והערך b המינימאלי בתת העץ הימני.

נגדיר סדר מלא עבור תת הקטעים בצורה הבאה:

נסמן תת קטע $x < y$ אם $a_x < a_y$ או אם $a_x = a_y$ וגם $b_x > b_y$.

נכניס את n תתי הקטעים למבנה הנתונים החדש $O(nlogn)$ כל הכנסה של איבר לעץ א"ש ע"פ הפעולות הידועות ונוסיף שמירה על הערכים הb המינימאלי של תתי העצים בכל צומת. בביצוע רוטציה נעביר את הערכים המתאים ב $O(1)$ ובעת הוספה של צומת בעלה נבצע ריצה לשורש העץ ותיקון כל הערכים שבדרך $O(logn)$.

1. נכניס עותק של הרשימה לבסיס הנתונים החדש $O(n) * O(logn) = O(nlogn)$.
2. עבור כל קטע נבצע ריצה מהשורש למציאת האיבר המינימאלי שגדול מהקטע הנבדק (למעשה אנו רצים בעח"ב מאוזן עד לקטע הנבדק עצמו) $O(logn)$
3. בכל שלב בריצה של שלב 2, נקבע ב $O(1)$ אם קיים קטע המכיל את הקטע הנבדק בתת העץ הימני ע"י בדיקה עם הערך המינימאלי של b בתת העץ השמאלי.

הרחבה לגבי הבדיקה - אם הערך a הנבדק קטן מהערך a בצומת אז הצומת לא יכולה להכיל קטע שמכיל את הקטע הנבדק וגם בתת עץ הימני (שמכיל ערכי a גדולים או שווים) לא יכול להיות קטע המכיל את הקטע הנבדק, לכן נחפש בתת עץ השמאלי.

אם הערך a הנבדק גדול מהערך aבצומת אז הצומת מתאימה רק אם הערך b המינימאלי בתת העץ השמאלי קטן או שווה לערך b הנבדק (כי ממילא ערכי a רק קטנים כהולכים בצעדים שמאלה). אם לא מצאנו נמשיך לחפש בתת העץ הימני.

החיפוש יסתיים כשאנו מוצאים. אם מצאנו בתא שהוא ממש הקטע הנבדק אז נסיק שאין קטע החלקי לקטע הנבדק. אחרת יש.

סך הכל אנו מבצעים n פעמים ריצות על עח"ב א"ש ולכן הסיבוכיות הכוללת היא $O(nlogn)$

18 מכיוון שלא נתון לנו אם הכיוונים של המיונים אחידים (בניגוד למה שהיה באופק)

זמן הריצה המינימלי יהיה min(nlogm,mlogn כלומר חיפוש בינארי על כל שורה\עמודה

אם נתון לנו כי הם ממויינים באותו סדר אז ניתן ללכת מהפינה בח+m על ידי בדיקה של האיבר הפינתי עם הערך גדול או קטן ואז לטייל ככה על המערך

19 א. $A[4] = 0 \quad A[3] = 1 \quad A[1] = A[2] = 2$ אזי ע"פ האלגוריתם למיון ערימה בעמוד 113 בספר נחליף בכל שלב בין ראש הערימה לסופה ונבנה את הערימה המצומצמת. נקבל בפלט הממוין

$A[4] = 0, A[3] = 1, A[2] = 2, A[1] = 2$

מצד שני, עבור המערך $A[4] = 0 \quad A[3] = 2 \quad A[1] = A[2] = 2$ (מכיוון שהאלגוריתם maxHeapify) בוחר קודם את הבן השמאלי אם הוא גדול שווה) נקבל בפלט הממוין:

$A[4] = 0, A[2] = 2, A[3] = 2, A[1] = 2$

ב. סעיף ב' מוכיח ע"י דוגמא נגדית שמיון ערימה אינו יציב (כי הוא לא שומר על המיקום היחסי של ערכים שווים).

20 הרי שכל פעולת רוטציה שמאלה היא הפיכה ע"י פעולת רוטציה ימינה.
בנוסף כל פעולת רוטציה שמאלה מגדילה את מספר הצמתים במסלול השמאלי ב1.
נתבונן באלגוריתם:

1. כל עוד יש בן ימני, בצע רוטציה שמאלה.
2. אם אין בן ימני אז חזור לשורה 1 עם הבן השמאלי.
3. אם אין בן שמאלי סיים.

הרי שהאלגוריתם רץ על הפאה השמאלית של העץ ומסיימת כאשר העץ במצב של שרוך.
מכיוון שכל פעולת רוטציה מגדילה את הפאה השמאלית ב1 אזי שמבוצעות n-1 רוטציות.
אם נבצע את סדר הרוטציות ימינה בסדר הפוך נקבל חזרה את אותו העח"ב.
בצורה זו ניתן להגיע מT לשרוך ומT' לשרוך ולכן גם משרוך לT' כלומר מT לT' ב2n-2 רוטציות.

21 **א'** אחרי ביצוע החלוקה, כל איברי המערך הקטנים מאיבר הציר חייבים להימצא לפניו, וכל איברי המערך הגדולים ממנו חייבים להימצא אחריו. האיברים $<4;5;9>$ מקיימים את הנדרש.

ב' סורקים את המערך B משמאל לימין ; כל איבר $B[i]$ שהוא מכסימום בתת-מערך $B[1..i]$ מועתק למערך עזר L. סורקים את המערך B מימין לשמאל ; כל איבר $B[i]$ שהוא מינימום בתת-מערך $B[i..n]$ מועתק למערך עזר R. מחזירים את כל האיברים הנמצאים גם ב- L וגם ב- R. מכיוון שהמערך L ממוין בסדר עולה והמערך R ממוין בסדר יורד, אפשר למצוא את האיברים האלה בזמן $O(n)$. זמן הריצה הכולל: $O(n)$.

22

נבנה את העץ האדום-שחור הבא: התת-עץ הימני הוא עץ שלם בגובה h ; כל הצמתים שלו שחורים. התת-עץ השמאלי הוא עץ שלם בגובה $2h$; הצמתים שלו צבועים לסירוגין, כל הצמתים ברמות הזוגיות שחורים, כל הצמתים ברמות האי-זוגיות אדומים. העץ המתקבל הינו עץ אדום-שחור חוקי.

גודל התת-עץ הימני הוא $rn = 2^h - 1$; גודל התת-עץ השמאלי הוא $ln = 2^{2h} - 1$.

אם מתקיים התנאי $ln < 128 \cdot rn$, אז $2^{2h} - 1 < 128 \cdot (2^h - 1) < 2^{h+7} - 1$; כלומר, $h < 7$.

מסקנה: התנאי הנתון לא מתקיים תמיד.

23

א

```
LINEAR-MIN (A, z)
   min ← ∞ ,   k ← 0
   for i ← 1 to n
      do if A[i] ≥ z and min > A[i]
            then k ← i
                 min ← A[i]
   if k > 0
      then return k
      else return "not found"
```

השגרה רצה בזמן $O(n)$.

ב

```
BINARY-MIN (A, z)
   if A[n] < z
      then return "not found"
   low ← 1,  high ← n
   while low ≤ high
      do if A[low] ≥ z
            then return low
         if A[high] < z
            then return high+1
         mid ← (low + high)/2
         if A[mid] = z
            then return mid
            else if A[mid] > z
                    then high ← mid-1
                    else low ← mid+1
```

השגרה פועלת בדומה לחיפוש הבינארי ורצה בזמן $O(lgn)$.

24

(א) נכתוב אלגוריתם רקורסיבי. הקריאה לשגרה MED3 מחזירה את ערכי המיקום ה- $\lfloor n/3 \rfloor$ וה- $\lceil 2n/3 \rceil$.

- אם $k=\lfloor n/3 \rfloor$ או $k=\lceil 2n/3 \rceil$, סיימנו; אחרת,
- אם $k<\lfloor n/3 \rfloor$, מבצעים חלוקה סביב ערך המיקום ה- $\lfloor n/3 \rfloor$ וקוראים ל- MED3 עבור השליש הראשון של המערך; אחרת,
- אם $k>\lceil 2n/3 \rceil$, מבצעים חלוקה סביב ערך המיקום ה- $\lceil 2n/3 \rceil$ וקוראים ל- MED3 עבור השליש האחרון של המערך; אחרת,
- מבצעים את שתי פעולות החלוקה וקוראים ל- MED3 עבור השליש האמצעי של המערך.

נוסחת הנסיגה של הרקורסיה:

$$T(n) = T(n/3) + \Theta(n)$$

פתרון הנוסחה (שיטת האב, מקרה 3, הפונקציה $f(n)=n$ רגולרית) הוא:

$$T(n) = \Theta(n).$$

(ב) לא יכולנו לכתוב אלגוריתם המחזיר את כל ערכי המיקום, היינו מקבלים מיון של המערך, דבר שלא ניתן לבצע בזמן ליניארי.

25

(א) במעבר מ- H ל- D חייבים לרוץ מהעלים אל השורש:

```
for i ← n downto 2
    do H[i] ← D[i] ← H[i] − H[Parent(i)]
```

בשחזור של H מתוך D חייבים לרוץ בכיוון ההפוך:

```
for i ← 2 to n
    do H[i] ← D[i] + H[Parent(i)]
```

השורש לא משתנה: $D[1]=H[1]$.

(ב) מוסיפים את המפתח החדש לסוף המערך. מסמנים את המסלול מהאיבר החדש כלפי מעלה, עד השורש (ניתן להשתמש במערך של מצביעים בגודל $\lceil \lg n \rceil$); זמן: $O(\lg n)$.
יורדים מהשורש לאורך המסלול המסומן ומשחזרים את הערכים המקוריים של H (לכל איבר במסלול וגם לבן השני שלו, אם הוא קיים); זמן: $O(\lg n)$.
מבצעים את תיקון הערימה כלפי מעלה על המסלול המסומן; זמן: $O(\lg n)$.
עולים לאורך המסלול המסומן ומחשבים את ההפרשים (הערכים של D) עבור כל איבר על המסלול (וגם עבור אחיו, אם הוא קיים); זמן: $O(\lg n)$.

26

(א) כל פעולת השוואה בין s לבין מפתח t בעץ מתבצעת בזמן $O(m)$.
הפעולה LCS-LENGTH(s,t) מתבצעת בזמן $O(m^2)$.

SEARCH(T,s): מבצעת $O(N)$ פעולות השוואה בין s לבין מפתחות בעץ; זמן הריצה הוא:

$$O(m\cdot N + m^2\cdot N) = O(m^2\cdot N)$$

INSERT(T,s): כמו במקרה החיפוש; זמן הריצה הוא: $O(m^2\cdot N)$

DELETE(T,p): לא מבצעת פעולות השוואה בין מפתחות בעץ; זמן הריצה: $O(N)$.

27

$\textsc{Radix-Sort} + \textsc{Counting-Sort}$ (בדומה לשאלה ז-18, עמ' 126 במדריך למידה).

28

א מחלקים את המערך לשני חלקים ככה שכל איבר בחלק הראשון קטן שווה לכל איבר בחלק השני. ממיינים את החלק הראשון עם מיון ערמה ואת החלק השני רקורסיבית ויוצא ששני החלקים ממויינים.

ב נוסחת הנסיגה היא

$$T(n) = T\left(\frac{n}{2}\right) + \Theta(n) + \Theta\left(\frac{n}{2}\lg\frac{n}{2}\right) = T\left(\frac{n}{2}\right) + \Theta(n\lg n)$$

אפשר לפתור אותה עם שיטת האב מקרה 3 כאשר $c = \frac{1}{2}$. מקבלים $T(n) = \Theta(n\lg n)$.

ג במקרה הגרוע מיון מהיר הוא היחיד שרץ ב-$\Theta(n^2)$. במקרה הממוצע כולם רצים ב-$\Theta(n\lg n)$.

29

א בונים עץ אדום שחור מכל הערכים **השונים** ב-A. בגלל שיש k ערכים שונים, יהיו בעץ k צמתים ולכן הגובה שלו הוא $O(\lg k)$. N הכנסות לעץ כזה לוקחות $O(N\lg k)$. לכל צומת x בעץ נשמור שדה $size$ שמכיל את מספר הפעמים שהמפתח של x מופיע בקלט - $m(key[x])$. שומרים בצד זוג מספרים: המספר שהופיע הכי הרבה עד עכשיו וכמות הפעמים שהופיע. בכל הכנסה בהתאם לשדה ה-$size$ של הצומת המוכנס מעדכנים את זוג השדות.

ב משתמשים בעץ מסעיף א' כדי לבנות עץ אדום-שחור נוסף שמכיל את המכפלה $key[x]\cdot size[x]$ לכל צומת x בעץ המקורי. בכל צומת שומרים רשימה מקושרת $keys$ של המפתחות.

סורקים את העץ שבנינו ולכל צומת x מחפשים את $y = z - key[x]$. אם $key[y] \neq key[x]$ אז מחזירים את האיברים הראשונים ב-$keys[x], keys[y]$. אם $key[y] = key[x]$ אז צריך לוודא שיש יותר מאיבר אחד ב-$keys[x]$, אחרת האיבר שמצאנו הוא אותו אחד.

זמן ריצה: $O(N\lg k) + O(k\lg k) + O(k) = O(N\lg k)$.

30

ב 4 גובה עץ הרקורסיה הוא $\lg n$ כאשר עלות הרמה ה-k היא $O(\frac{n}{2^k})$:

$$\sum_{k=0}^{\lg n}\frac{n}{2^k} = O(n)$$

ג 4 אחרי החלוקה המערך מחולק לשני חצאים ככה שכל איבר בחצי הראשון קטן מכל איבר בחצי השני. בחצי השני בונים ערמת מינימום ולכן האיבר הראשון שם הוא המינימום בחצי השני והוא גדול מכל האיברים בחצי הראשון, כלומר הוא ערך המיקום ה-$\lfloor\frac{n}{2^i}\rfloor + 1$...

$\textsc{Find-Os}$ - מחזירים את $A[\lfloor\frac{n}{2^i}\rfloor + 1]$.

$\textsc{Del-Os}$ - מוציאים את השורש של הערמה שמתחילה ב-$\lfloor\frac{n}{2^i}\rfloor + 1$.
לכל $k = i+1, \ldots, \lfloor\lg n\rfloor$ מוציאים את שורש הערמה שמתחילה ב-$\lfloor\frac{n}{2^k}\rfloor + 1$ ומכניסים אותו לערמה שמתחילה ב-$\lfloor\frac{n}{2^{k-1}}\rfloor + 1$ - סוג של "בעבוע". כל הפעולות על הערמות לוקחות $O(\lg n)$ ויש $\lg n$ ערמות, סה"כ מקבלים $O(\lg^2 n)$.

31

א ממיינים את המערך ב-$\Theta(n\lg n)$. לכל איבר x במערך מחפשים בינארית את $x^2 - 1$. סה"כ n חיפושים בעלות $\Theta(\lg n) \Leftarrow \Theta(n\lg n)$.

ב מכניסים לטבלת גיבוב (שומרים גם את האינדקס המקורי של האיבר) ולכל איבר בטבלה מחפשים אותו דבר כמו בסעיף הקודם. סה"כ n חיפושים בעלות $\Theta(1) \Leftarrow \Theta(n)$.

32

א מוצאים עם $\textsc{Select}$ את האיבר ה-p וקוראים ל-$\textsc{Partition}$ כשהוא ה-$pivot$. אז ממיינים עם מיון מיזוג/ערימה את התת מערך שכל איבריו קטנים מ-p. סה"כ יוצא:

$$O(p\lg p) = O\left(\frac{n}{\lg n}(\lg n - \lg\lg n)\right) = O\left(n\left(1 - \underset{\substack{\downarrow\\0}}{\frac{\lg\lg n}{\lg n}}\right)\right) = O(n)$$

ב $\textsc{Radix-Sort} + \textsc{Counting-Sort}$ (בדומה לשאלה ז-18, עמ' 126 במדריך למידה).

33 **א** ממזגים את כל c הקטעים בבת אחד כמו במיון מיזוג (אלא שכאן יש לנו c קטעים במקום 2). צריך לעבור על n איברים ובכל שלב לבצע $c = O(1)$ בדיקות (כדי למצוא את המינימום מבין c החלקים), סה"כ יוצא $\Theta(n)$.

ב + ג פותרים את הנוסחה $T(n) = c \cdot T(\frac{n}{c}) + \Theta(n)$ עם שיטת האב מקרה 2. מקבלים $T(n) = \Theta(n \lg n)$ ולכן אין הבדל בין שתי הגרסאות.

34 **א** מנהלים שתי סריקות תוכיות במקביל: אחת רגילה על תת העץ השמאלי של השורש ואחת הפוכה על תת העץ הימני של השורש (הולכים ימינה לפני שהולכים שמאלה). הראשונה תתן לנו רשימה של מפתחות בסדר עולה והשנייה בסדר יורד. בכל שלב סוכמים את שני הצמתים הנוכחיים. אם הסכום שווה ל-$2 \cdot key[root[T]]$ סיימנו. אם הוא קטן מקדמים את הסריקה הראשונה, אחרת את השנייה. האלגוריתם רץ ב-$O(n)$ ודורש שתי מחסניות בגודל $O(\lg n)$.

ב מריצים את האלגוריתם מסעיף א' לפי רמות (קודם הצמתים בעומק 0, אחרי זה 1, 2...). לשורש יש $\Theta(n)$ עבודה, לכל אחד מהבנים $\Theta(\frac{n}{2})$ וכך הלאה...

זמן הריצה הוא: $T(n) = 2 \cdot T(\frac{n}{2}) + \Theta(n) = \Theta(n \lg n)$.

35 מתחילים במיון המערך. לכל $1 < j < n$ רצים עם שני אינדקסים, אחד בראש המערך והשני בסופו ובודקים אם הסכום שלהם שווה ל-$2 \cdot A[j]$. במידה וכן אז סיימנו. אם הוא קטן מקדמים את האינדקס הראשון. אם הוא גדול מזיזים את האחרון אחד אחורה. מקדמים את j כאשר אחד האינדקסים מגיע אליו ומתחילים את התהליך שוב.

המיון הראשוני לוקח $\Theta(n \lg n)$. בכל שלב זוג האינדקסים משמאל ומימין ל-j עוברים על $\Theta(n)$ איברים, ו-j גם כן, לכן זמן הריצה הכולל הוא $\Theta(n^2)$.

36 המועמדים האפשריים לתנאי הראשון הם כל האיברים שערך המיקום שלהם גדול מ-$n-2m$. כדי שאיבר כלשהו יהיה מועמד לתנאי השני, הוא צריך להיות ערך המיקום ה-$n-m$ או ה-n. אם כך מוצאים את ערך המיקום ה-$n-m$ עם Select ועושים עוד מעבר על המערך כדי למנות את מספר המופעים שלו. אם הוא מופיע יותר מ-m פעמים אז סיימנו, אחרת עושים אותו דבר לערך המיקום ה-n. סה"כ קוראים פעמיים ל-Select ועושים עוד שני מעברים על המערך לכן זמן הריצה הוא $\Theta(n)$.

37 ממיינים את המערך עם Counting-Sort ורצים עם שני אינדקסים מההתחלה ומהסוף. אם הסכום שלהם גדול מ-65535 מזיזים את האינדקס השני אחד אחורה. אם הסכום קטן אז הראשון זז אחד קדימה. מסיימים כשהסכום שווה או שהאינדקסים נפגשים/עברו אחד את השני.

38 ממיינים את הזוגות לפי x. אם יש יותר מזוג נקודות אחד עם אותו ערך x, לוקחים את האחד עם ערך ה-y המקסימלי.

שומרים במשתנה $area$ את השטח שנצבר עד כה. האלגוריתם רץ על הזוגות הממויינים תוך שהוא שומר במשתנה $max = (x, y)$ את הנקודה עם ה-y המקסימלי. המלבן ה-i מטופל בצורה הבאה:

- אם $y_i \leq y_{i-1}$ אז $area += (x_i - x_{i-1}) \cdot y_i$.
- אחרת אם $y_i > max_y$ אז $area = x_i \cdot y_i$ (מעדכנים את max).
- אחרת $area = x_i \cdot max_y - (x_i - max_x) \cdot max_y$.

39 מוצאים את החציון עם Select ורצים על המערך וסופרים כמה איברים שווים לו.

40 **א'** גובה הערמה הוא $\lfloor \lg n \rfloor$; מספר השורות הינו $1+\lfloor \lg n \rfloor$. בכל שורה מבצעים חיפוש בינרי.

```
HEAP-LEVEL-SEARCH(H, n, z)
l ← 0
while l < ⌊lg n⌋
  do i ← BINARY-SEARCH(A, 2^l, 2^(l+1) − 1, z)
     if i ≠ NIL
        then return i
        else l ← l + 1
i ← BINARY-SEARCH(A, 2^l, n, z)
if i ≠ NIL
  then return i
else return NIL
```

ב' זמן הריצה הינו $(1+\lfloor \lg n \rfloor)\cdot O(\lg n)=O(\lg^2 n)$.

ג' שמורת הלולאה היא: לפני האיטרציה l ($0\le l\le \lfloor \lg n \rfloor$), אם האיבר z נמצא בערמה, אזי הוא נמצא באחת השורות $l,\dots,\lfloor \lg n \rfloor$.

41 **א'** בקריאה הראשונה, איבר הציר 1 מתחלף עם האיבר הראשון 2; מתקבל התת-מערך $A=[3,\dots,n,2]$. התהליך חוזר על עצמו $n-1$ פעמים; לכן, זמן הריצה של האלגוריתם הינו $\Theta(n^2)$.

ב' בקריאה הראשונה, איבר הציר 2 מתחלף עם האיבר השני n ; מתקבל התת-מערך $A=[n-1,\dots,3,n]$. בקריאה השניה, אין שינויים במערך; מתקבל התת-מערך $A=[n-1,\dots,3]$. מערך זה הוא ממוין בסדר יורד; לכן, זמן הריצה של האלגוריתם הינו $\Theta(n^2)$.

42 **א'** יהא x הבן השמאלי של y ויהא z הבן הימני של y. אם אביו p של y קיים, מחברים את z כבן של p במקומו של y. אחר-כך, אם z קיים, מחברים את x כבנו השמאלי של הצומת המינימלי בתת-עץ המושרש ב-z. אם z אינו קיים, מחברים את x כבן של p במקומו של y. כל התהליך מתבצע בזמן $O(h)$.

ב' הגובה של U קטן מ-h במקרים הבאים: כל מסלול באורך h מהשורש לעלה מסתיים בעלה של התת-עץ המושרש ב-z ; או, z לא קיים וכל מסלול כזה מסתיים בעלה של התת-עץ המושרש ב-x.

ג' הגובה המכסימלי האפשרי של U כפונקציה של h הינו $2h-2$. זה קורה כאשר y הוא השורש של העץ, גובהו של כל אחד מהתת-עצים המושרשים ב-x וב-z הוא $h-1$, והצומת המינימלי בתת-עץ המושרש ב-z הוא עלה נמוך ביותר.

43 נעתיק את שתי הרשימות למערכים (באותם שמות S ו- T).
נניח כי $n < m$. ממיינים את המערך S (מיון מיזוג או מיון ערמה); זמן ריצה $\Theta(m \cdot \lg m)$. לכל איבר $y \in T$ נבצע חיפוש בינרי במערך S אחר הערך $x = z - y$; זמן ריצה $\Theta(n \cdot \lg m)$. סה"כ $\Theta((m+n) \cdot \lg m)$.

אם $m > n$, דומה.

44 **א'** (10 נקודות) נתון מערך של מספרים $A[1..n]$. ידוע שקיים אינדקס m, $1 < m < n$, כך שמתקיימים התנאים $A[1] > ... > A[m-1] > A[m] < A[m+1] < ... < A[n]$ (כלומר, התת-מערך $A[1..m]$ ממוין בסדר יורד והתת-מערך $A[m..n]$ ממוין בסדר עולה).
כתבו שגרת פסֵידוקוד המבצעת חיפוש אחר האינדקס m והמחזירה אותו; זמן הריצה הנדרש הינו $O(\lg n)$.

ב' (15 נקודות) נתון מערך של מספרים $A[1..n]$. ידוע שקיימים אינדקסים p,q, $1 < p < q < n$, כך שמתקיימים התנאים $A[1] > ... > A[p-1] > A[p] = ... = A[q] < A[q+1] < ... < A[n]$ (כלומר, התת-מערך $A[1..p]$ ממוין בסדר יורד והתת-מערך $A[q..n]$ ממוין בסדר עולה).
כתבו שגרת פסֵידוקוד המבצעת חיפוש אחר האינדקסים p,q והמחזירה אותם; זמן הריצה הנדרש הינו $O(\lg n)$.

א' השגרה בפסֵידוקוד:

```
FIND-MINIMUM(A)
left ← 1
right ← length[A]
while left < right
   do m ← (left + right)/2
      if m = left
         then return m
      else if A[m] < A[m+1]
         then right ← m
         else left ← m
```

45

א' בונים את המערך $B[1..n]$: $B[i] = A[i] - n^2$. אז מתקיים $0 \le B[i] \le n^3 - 1$ לכל $i = 1,...,n$.
ניתן לייצג כל איבר של B כמספר בן שלוש ספרות בבסיס n. משתמשים במיון בסיס מעל מיון-מניה.

ב' בונים את העץ האדום-שחור בהתחשב בכפילויות של המפתחות. מתקבל עץ בגודל $O(\lg^2 n)$ שגובהו $O(\lg(\lg^2 n)) = O(\lg \lg n)$.

46 נבנה את המערך $C[1..n]$; נגדיר $C[i]=|ax_i+by_i|$, $i=1,...,n$ (כאשר $p_i=(x_i,y_i)$). ממינים את המערך C באמצעות אלגוריתם מיון אופטימלי ; אחר-כך, בודקים, בזמן לינארי, אם קיימים במערך C שני ערכים בעלי ערך זהה. זמן הריצה הכולל הינו $O(n\cdot\lg n)$.

47 אילו המערך A היה ממוין, כל המופעים של z היו מופיעים ברצף והתחלתו של הרצף אחרי המיקום $n-3k$; רצף כזה, באורך גדול מ- k , חייב לכלול את המיקום $n-2k$ או את המיקום $n-k$.

לכן, האלגוריתם המוצע יפעל בצורה הבאה :

נמצא את ערך המיקום ה- $n-2k$ בעזרת האלגוריתם SELECT ; נבדוק אם ערך זה מופיע k פעמים לפחות במערך A ; אם כן, ערך זה הוא הערך z הנדרש ; אחרת,

נמצא את ערך המיקום ה- $n-k$ בעזרת האלגוריתם SELECT ; נבדוק אם ערך זה מופיע k פעמים לפחות במערך A ; אם כן, ערך זה הוא הערך z הנדרש ; אחרת, הערך הנדרש לא קיים.

זמן הריצה הכולל : $\Theta(n)$.