מבני נתונים ומבוא לאלגוריתמים
20407

ממן 14

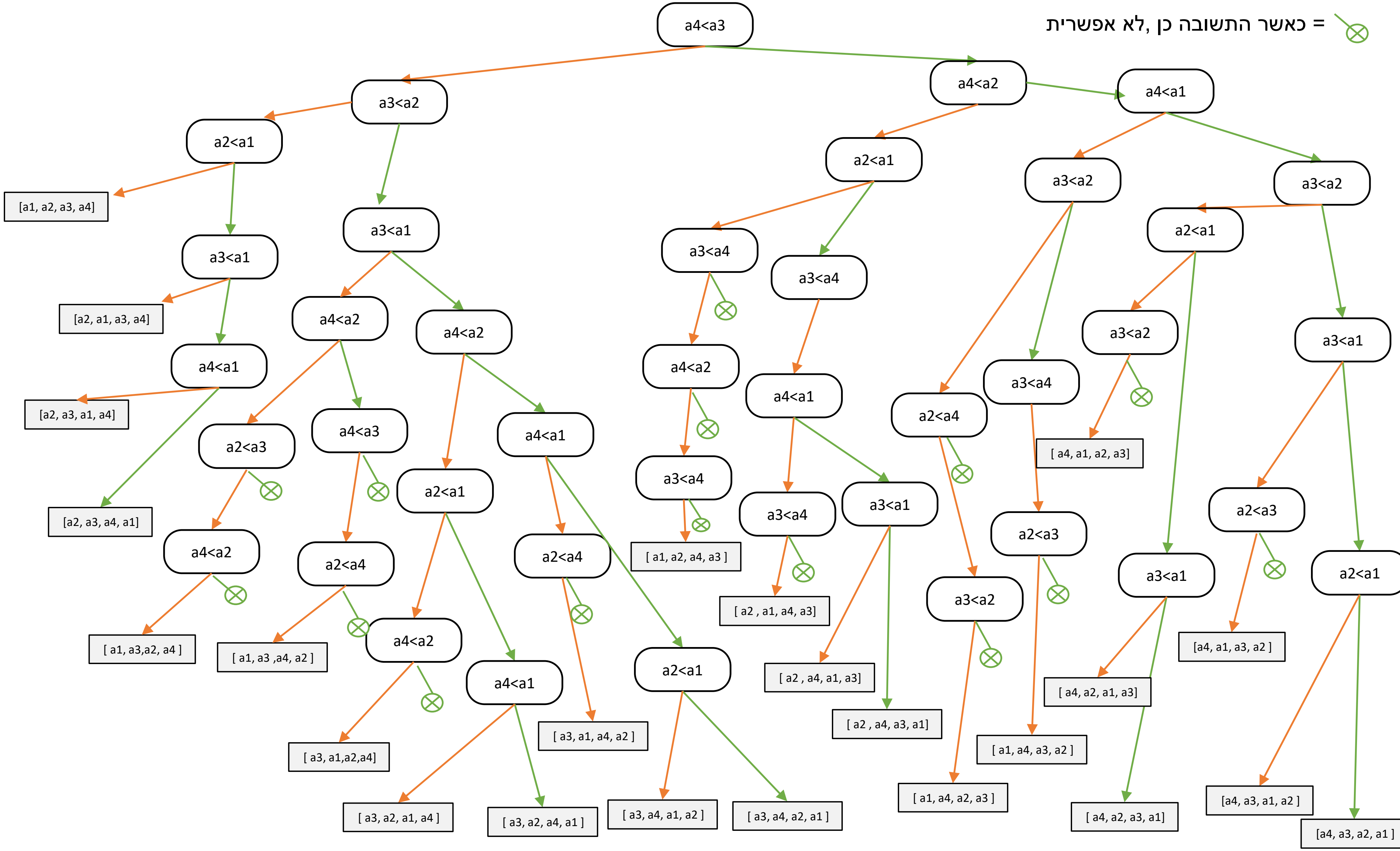
מוגש ע"י אורנית כהן גינדי

ממן 14 שאלה 1 סעיף א

בסדר

עץ החלטה עבור מיון בועות לארבע איברים: המערך: [a1, a2, a3, a4], ירוק מסמן - כן, וכתום- לא

= כאשר התשובה כן, לא אפשרית



ממן 14 שאלה 1 המשך

ב. עומק עץ החלטה הנו כמספר ההשוואות שהאלגוריתם מבצע במקרה הגרוע.
אלגוריתם מיון מיזוג מבצע תיטא של n lgn במקרה הגרוע ועל כן זהו עומקו של עץ ההחלטה עבור מיון זה.

בסדר

ג.אורך המסלול הקצר ביותר בין השורש לעלה במיון הכנסה הנו כמספר ההשוואות במקרה הטוב ביותר. מצב זה הוא כאשר הקלט כבר ממוין , ובכל איטרציה מתבצעת השוואה לאיבר המקסימלי בתת המערך הממוין , סה"כ N-1 השוואות – וזהו אורך המסלול הקצר ביותר בעץ ההחלטה למיון זה.

מה קורה במקרה זה באלגוריתם?

אנו יודעים כי במקרה הטוב המערך כבר ממוין. במקרה זה התנאי של לולאת ה whileבשורה 5 (עמוד 21 בספר הלימוד) לא יתקיים כלל ולכן מספר ההשוואות יהיה n-1
(או 2n-2אם סופרים את ההשוואה i>0שבודקת שלא הגענו לתחילת המערך).

ד. ניתן לשנות כל אלגוריתם מבוסס השוואות כך שאורך המסלול הקצר יהיה N-1, ע"י הוספת בדיקה התחלתית של האם הקלט ממוין. בבדיקה זו יתבצעו N-1 השוואות וברגע הראשון בו מתגלה שהקלט אינו ממוין , יתחיל לפעול אלגוריתם המיון (ז"א מכל נקודת השוואה, אם הסדר לא ממוין אז יתפצל ממנה עץ החלטה שמייצג את אלגוריתם המיון)

בסדר

ממן 14 שאלה 2

בסדר

א.ייתכן.
זמן הריצה במקרה הגרוע, גרוע יותר מהמיון הטוב ביותר, וגרוע יותר תמיד אפשר לעשות. ואת המקרה הטוב הסברתי בשאלה הקודמת, שניתן (ואיך) להפוך כל אלגוריתם מבוסס השוואות להיות N-1 (לינארי)במקרה הטוב

ב.לא יתכן.
לכל אלגוריתם מיון מבוסס השוואות מתאים עץ החלטה והעומק של העץ מבטא את זמן הריצה של האלגוריתם.
באופן כללי עומק של עץ עם T עלים, הנו לכל הפחות logT.
כמות העלים הכוללת היא n! (n גודל הקלט), כמות הצמתים בעומק O(n) (עומק לינארי) הנה 2^{cn} (c קבוע)

לפי טענת הפרופסור, לפחות חצי מכמות הצמתים בעומק לינארי הם עלים , מכאן שצריך להתקיים:
½ =< 2^{cn} /n! ומתקבלת סתירה לכך שהביטוי שואף ל0.

בסדר

ממן 14 שאלה 3

נתונה הסדרה S באורך n, המכילה ערכים בתחום: [n+1 .. n+p] , p ו- n שלמים ובלתי תלויים

א. מטרת השגרה היא למצוא את הערך השכיח בסדרה S , ז"א שמספר מופעיו בסדרה, הוא הגדול ביותר. לשם כך השגרה נעזרת במערך מונים counts שגודלו p (כגודל ההפרש של כל האיברים בS , מהערך n) וכל אינדקס בו מיצג ערך בסדרה S. במערך המונים הערכים באינדקסים מקודמים בכל איטרציה, לפי הערך שנמצא בS במעבר אחד על הסדרה . חיפוש לינארי אחר הערך המקסימלי במערך המונים מניב את התוצאה הרצויה.

בעצם יש כאן שימוש במיון מניה,
מה הקריטריונים שתנאי השאלה עומדים בהם , כך שאפשר להשתמש המיון מניה?
האם האלגוריתם כמו שהוא כתוב אכן עושה את מה שהוא אמור לעשות? 2- נקודות לשאלה

```
Most-Frequent( S )
comment:  'counts' is another array of size  p, initiated with zeros
for  i ← 1 to  n
      currentVal ← S[ i ]  - n
      counts[ currentVal ] =  counts[ currentVal ] + 1
max =  1
for i ← 2 to  p
     if counts [ i ] > counts [ max ]
              then max ←  i
return max + n
```

לולאת for ראשונה רצה לכל אורכה של הרשימה S ← O(n)
לולאת for שניה רצה לאורך מערך המונים counts שבאורך p ← O(p)
סה"כ זמן ריצה כמבוקש: O(n+p) בסדר

ממן 14 שאלה 3 - המשך

ב. מטרת השגרה: למצוא a ו-b ב- S, ששכיחותם גבוהה מ-1 וסכומם שווה ל- z נתון , ב- O(n+p).
בתחילת השגרה , נבנה מערך מונים אשר מכיל את שכיחויות ערכי S (את הערכים של S (פחות הערך n כדי לחסוך מקום) מייצגים האינדקסים של מערך המונים) ואחרי כן השגרה בודקת בלולאה על מערך המונים עבור כל ערך אם המשלים שלו לZ קיים ואם לשניהם שכיחות גבוהה מ-1
השגרה מחזירה זוג מספרים שהם ערכים בS ומקיימים את הדרישות. אם לא נמצאו ערכים להחזרה , יוחזר null

לולאת for בשורה 1: רצה על S ← O(n)
לולאת for בשורה 5: רצה על counts ← O(P)
סה"כ זמן ריצה O(n+p)

```
    FIND-AB( S, z )
    comment:  'counts' is another array of size  p, initiated with
    zeros and 'sorted'  contains the sorted S
1    for  i  ← 1 to  n
2          currentVal = S[ i ]  - n
3          counts[ currentVal ] =  counts[ currentVal ] + 1
4    target ← z - n
5    for  i ← 1 to p
6          a  ←  i
7          b  ←  target - a
8          if   b  <= p  (commen: if b in range of counts)
9          then  if   counts [a ] > 1  and  counts[ b] >1
10                   then return <a+n , b+n>
11   return null
```

הפתרון בסדר, הערות דומות לסעיף א

ממן 14 שאלה 4

א. נשתמש בשתי מחסניות , אחת בשם S שתכיל את המספרים כולם, והשניה בשם mins שתכיל היסטורית ערכי המינימום , ז"א ערכי המינימום לפי סדר הכנסם למחסנית numbers.
בנוסף לכל אלה, נשמור את הערך S.d שיאותחל ב-0. שמירת הערך S.d בעצם מסייעת לפעולה ADD לפעול בO(1) כיון שהפעולה ADD עצמה תהיה רק עדכון של הערך S.d , והוספת S.d לערכי המחסנית תתבצע רק בזמן הוצאתם מהמחסנית (POP).
(אילו רצינו לשנות ממש את כל אברי המחסנית היה עלינו לעבור על כולם ואז זמן הריצה היה O(n))

בפעולה PUSH נכניס ערך חדש למערך וכאשר הוא יצא ממנו, הוא יצא בתוספת S.d אוטומטית. כדי להמנע משינוי לא רצוי זה, נפחית ממנו את הערך S.d לפני הכנסתו למחסנית.

מתודות הנקראות באותיות קטנות, הנן המתודות **הרגילות של מחסנית רגילה**. באותיות גדולות , מתודות של **מבנה הנתונים S.**

הפעולה POP(S) מחזירה ערך מSm , פלוס הערך S.d . אם S.d =0 הערך המוחזר יהיה ללא שינוי.

```
PUSH(S , x)
S.push(x - S.d)
if mins is empty
   then mins.push(x - S.d)
else min ← mins.pop() +S.d
      mins.push(temp - S.d)
      if x ≤ min
        then  mins.push(x - S.d)
```

```
POP(S)
if S is empty
   then return NIL
num ← S.pop() + S.d
min ← mins.pop() + S.d
if min != null
   then if  min != num
            then  mins.push(min - S.d)
```

```
MIN(S)
if mins is empty
   then return NIL
min ← mins.pop() + S.d
mins.push(min - S.d)
return min
```

```
ADD(S , d)
S.d ← S.d + d
```

בסדר

השתמשתי ב ≤ כדי שהסטורית המינימומים תכיל עותקים של מינימום שחוזר על עצמו ב S

ב. נניח בשלילה שניתן למחוק את המינימום בזמן קבוע.
פעולת הסרת המינימום היא פעולה שניתן באמצעותה לבנות אלגוריתם מיון מבוסס השוואות. למשל כמו במיון ערימה. אם נוציא מינימומים בזה אחר זה נקבל מיון. אם נפעיל את פעולת הוצאת המינימום n על כל הקלט לצורך מיון, נקבל סיבוכיות של O(n)
וזה סותר אתהחסם התחתון O(nlgn) שקיים על מיונים מבוססי השוואות .
מכאן שהפעולה של הסרת המינימום אינה אפשרית בזמן קבוע

בסדר

ממן 14 שאלה 5

א. בסדר

$h(k) = k \bmod m \rightarrow h(1111) = 1111 \bmod 127 = 95$

ב.

$A = \frac{\sqrt{2}}{2}$ $kA = 1111 * \frac{\sqrt{2}}{2} = 785.59563 \rightarrow$

$h(1111) = 59563 * 127 = 75.64550 \rightarrow h(1111) = 75$

בסדר

ג. לפי הסכמה המתוארת בשאלה, j הוא מונה האיטרציות של חיפוש המפתח ואילו i הוא האינדקס של המפתח k בטבלת הגיבוב.

$i = h(k, j) = (i + j) \bmod m = (h(k, j-1) + j) \bmod m$

אחרי מספר איטרציות חיפוש, נראה שחלק מהביטוי מתפתח לסכום סדרה חשבונית :

$j = 0 \quad i_1 = h(k, 0) = h'(k)$

$j = 1 \quad i_2 = h(k, 1) = (h'(k) + 1) \bmod m$

$j = 2 \quad i_3 = h(k, 2) = ((h'(k) + 1) \bmod m + 2) \bmod m = (h'(k) + 1 + 2) \bmod m$

$j = 3 \quad i_4 = h(k, 3) = ((h'(k) + 1 + 2) \bmod m + 3) \bmod m = (h'(k) \underline{\mathbf{+ 1 + 2 + 3}}) \bmod m$

טענה: עבור j כללי (j<m) : $i_j = h(k, j) = (h'(k) + \sum_{t=0}^{j} t) \bmod m$

בסיס: $j=0 \quad i_0 = h(k,0) = (h'(k) + \sum_{t=0}^{0} t)\bmod m = (h'(k) + 0) \bmod m = h'(k)$

צעד: נניח שהטענה נכונה עבור j

נוכיח באינדוקציה שהטענה נכונה עבור j+1 :

$i_{j+1} = h(k, j+1) = (h(k, j) + j +1) \bmod m$

$\rightarrow h(k, j+1) = ((h'(k) + \sum_{t=0}^{j} t)\bmod m + j+1) \bmod m = \boxed{(h'(k) + \sum_{t=0}^{j+1} t) \bmod m \quad \text{מש"ל}}$

כעת נציב את הפתרון לנוסחת סכום סדרה חשבונית:

$\sum_{t=0}^{j} t = j(j+1)/2$

$\rightarrow h(k , j) = (h'(k) + j(j+1)/2) \bmod m = (h'(k) + (j^2 +j)/2) \bmod m = \mathbf{(h'(k) + j/2 + j^2/2) \bmod m}$

בסדר

פונקציית הגיבוב הכללית של הבדיקה הריבועית נראית כך:

$h(k, i) =(h'(k) + c_1 i + c_2 i^2) \bmod m$

מכאן נראה שעבור הקבועים $c_1 = c_2 = ½$ הסכמה המתוארת בשאלה, אכן מקיימת מקרה פרטי של סכמת "הבדיקה הריבועית"

ממן 14 שאלה 5 - המשך

ד. הצבה בפונקציית הגיבוב מסעיף קודם: $h(k, j) = (h(k, j-1) + j) \bmod m$

k=222 , j=0 → h’(k) = k mod m = 222 mod 127 = 95

k =1111 , j=0 → h’(1111) = 1111 mod 127 = 95 → place 95 is occupied by key 222.
j=1 → h(1111, 1) = (h(1111,0) + 1) mod 127= (95 +1) mod 127 = **96**

בסדר

k=6699, j=0 → h’(6699) = 6699 mod 127 = 95 → place 95 is occupied by key 222
j=1 → h(6699,1) = (h(6699,0) +1) mod d → (95 +1)mod 127 =96 → place 96 is occupied by key 1111
j=2 → h(6699,2) = (h(6699,1) + 2) mod d → (96 +2)mod 127 = **98**

כעת עם הקבועים c1 =c2=1/2 :

k=222 , j=0 → h(222,0) = (h’(222) + 0/2 +0^2/2) mod 127 = h’(k) = 222 mod 127 = **95**

k =1111 , j=0 → h(1111,0) = (h’(1111) + 0/2 +0^2/2) mod 127 = 1111 mod 127 = 95 → place 95 is occupied by key 222.
j=1 → h(1111, 1) = (95 + 1/2 + 1^2/2) mod 127 = **96**

k=6699, j=0 → h(6699,0) = (h’(6699) + 0/2 +0^2/2) mod 127 = 95 → place 95 is occupied by key 222
j=1 → h(6699,1) = (h’(6699) + 1/2 +1^2/2) mod 127 = (95 +1/2 +1/2) mod 127 = 96 → place 96 is occupied by key 1111
j=2 → h(6699,2) = (h’(6699) + 2/2 +2^2/2) mod 127 = (95 +1 +2) mod 127= **98**

ה.

k=222 , kA = 222 * $\frac{\sqrt{2}}{2}$ = 156.97770 →
h(222) = 0.97770 * 127 = 124.16858 → **h(222) = 124**

בסדר

k =1111 , kA = 1111 *$\frac{\sqrt{2}}{2}$ = 785.59563 →
h(1111) = 0.59563 * 127 =75.64550 → **h(1111) = 75**

k=6699 , kA = 6699 * $\frac{\sqrt{2}}{2}$ = 4736.90832 →
h(6699) = 0.90832 * 127 = 115.35755 → **h(6699) = 115**