עבודה במבני נתונים 5

מגישים: ניב לוי 204416846 דין אברג'יל 308365881

שאלה 1

למדנו כבר שהסיבה לכך שמיון-מהיר אינו רץ בזמן O(nlogn) היא בשל בחירה לא טובה של Pivot. בחירת ה-pivot מתבצעת ע"י הפונקציה partition ובמקרה הגרוע היא בוחרת את הpivot להיות האיבר הראשון או האחרון במערך הקלט. לעומת זאת אם נבחר את ה-pivot להיות החציון של המערך בכל פעם נוכל להקטין את מספר הקריאות לפונקציה partition משמעותית.

בפונקציה partition נבחר את החציון באופן הבא: את בחירת החציון נבצע ע"י אלגוריתם Deterministic Select שלמדנו: נחלק את המערך לחמישיות, נמיין אותן, נמצא את החציון של כל חמישייה ובאופן רקורסיבי נחזיר את חציון החציונים. זמן ריצה O(n) כפי שנלמד בכיתה.

כעת לאחר שבחרנו את החציון להיות הpivot אפשר לומר שבכל קריאה רקורסיבית ל-quickSort אנחנו עובדים עם 2 תתי מערכים בגודל n/2. כלומר, נקבל את נוסחת הנסיגה: $T(n) = 2T\left(\frac{n}{2}\right) + O(n)$ משיטת המאסטר נקבל שזמן הריצה הכולל הוא O(nlogn).

שאלה 2

findSpinnedElement (array a, int size, int key)

```
Pivot = findPivot(a, 1, size)
If (pivot does not exist) // which means that the array is sorted
     Return binarySearch(a, 1, size, key)
If (a[pivot] == key)
      Return pivot
LeftArray = a[1,…,pivot]
RightArray = a[pivot+1,…,size]
If (a[1] < key)
      Return binarySearch(leftArray, leftArray.size, key)
      Else return binarySearch(rightArray, rightArray.size, key)
```

```
findPivot (array, int low, int high)
1. Middle = (low+high)/2
2. If (array[middle] > array[middle+1]
3.    Return middle
4. If (array[middle] < array[middle-1])
5.    Return middle-1
6. If (array[low] > array[mid]) // it means pivot is at [1,…,mid-1]
7.    Return findPivot(array, low, middle-1)
8. Else return findPivot(array, middle+1, high)
```

<u>רעיון האלגוריתם:</u>

נחפש את הפיווט (הוא האיבר שמשני צידיו קיימים מספרים הגדולים ממנו), אחרי שנמצא אותו נבדוק: אם הוא המפתח נחזיר אותו, אחרת נחלק לשני מערכים משני צידיו (הם בהכרח ממויינים). נבדוק האם המפתח גדול מהאיבר הראשון במערך, אם כן הוא במערך השמאלי, אחרת הוא במערך הימני. בהתאם לכך נבצע חיפוש בינארי במערך המתאים.

<u>ניתוח זמן ריצה:</u>

- במציאת הpivot אנחנו מבצעים בדיקות שלוקחות זמן קבוע, וכל קריאה רקורסיבית קוראת לפונקציה עם חצי מהאיברים במערך. כלומר נקבל נוסחאת נסיגה $T(n) = T\left(\frac{n}{2}\right) + O(1)$. כלומר זמן הריצה הוא O(logn).
- לאחר שמצאנו את ה-pivot אנחנו מקבלים שני מערכים המכילים חלק מאיברי המערך ועליהם אנחנו מבצעים חיפוש בינארי. אפשר להגיד שחיפוש בינארי מקבל an מתוך n האיברים במערך כאשר 1 > a > 0 כלומר זמן הריצה של החיפוש בינארי הוא O(log(an)) שזה למעשה O(logn).

לכן בסה"כ זמן הריצה של האלגוריתם הנ"ל הוא log(n).

<u>שאלה 3</u>

<u>סעיף א':</u>

2	4	9	∞
3	8	16	∞
5	14	∞	∞
12	∞	∞	∞

<u>סעיף ב':</u>

נניח כי Y[1,1] = ∞ אז לכל i,j (אינדקס שורה ועמודה) מתקיים מהגדרת טבלת יאנג:

$Y[i,j] \geq Y[1,j] \geq Y[1,1] = \infty$ כי איברים גדלים ככל שהאינדקסים יותר גדולים. כמובן שאין מספר הגדול מאינסוף ולכן לכל i,j אינדקסים אפשריים במערך מתקיים $Y[i,j] = \infty$ ולכן כל האיברים במטריצה הם אינסוף ומהגדרת טבלת יאנג טבלה זו הינה טבלה ריקה.

נניח כי Y[m,n] < ∞ ידוע כי Y[m,n] הוא האיבר הגדול ביותר במטריצה כיוון שמהווה את המקום האחרון של השורה והעמודה האחרונה. לכל i,j (אינדקס שורה ועמודה) מתקיים מהגדרת טבלת יאנג:

$Y[i,j] \leq Y[m,j] \leq Y[m,n] < \infty$ ולכן $Y[i,j] < \infty$. לכן בהכרח כל איבר בטבלת יאנג קטן מאינסוף, כלומר מספר סופי ולכן הטבלה מלאה.

<u>סעיף ג':</u>

נשים לב שהאיבר הקטן במערך הוא במקום ה1,1, כלומר נוציא אותו ולאחר מכן נדרש לעדכן את טבלת יאנג כך שתשמר ההגדרה שכל איבר גדול מהאיבר הקודם בשורה או העמודה. האיברים הפוטנציאלים להחליף את האיבר במקום ה-i,j הם או האיבר במקום ה-i+1,j או האיבר במקום ה-I,j+1 נבחר את המינימלי מביניהם ונחליף במקום האיבר ה-I,j ככה נמשיך עד שנגיע לסוף המערך או עד שנגיע לאיבר שגדול מהאיבר ה-I,j (כלומר עידכנו את כל האיברים).

1. נשמור את האיבר במקום ה-1,1 במשתנה X (הוא האיבר המינימלי).
2. יהיו i=1 ו-j=1 אינדקסים.
3. כל עוד i+1 < m וגם j+1<n וגם Y[i,j] < Y[i+1,j] נבצע:
   a. Y[I,j] = min (Y[i+1,j], Y[i,j+1]).
   b. I = I +1
   c. J = j+1
4. נחזיר את X.

במקרה הגרוע אנחנו עוברים על המערך מהנקודה 1,1 ועד סוף השורה שזה n-1 השוואות ולאחר מכן יורדים בעמודה האחרונה עד לסופה ומבצעים עוד m השוואות. כלומר נגיע לסה"כ O(m+n-1). באופן סימטרי אפשר גם ללכת הפוך. שאר הפעולות הן פעולות קבועות ולכן זמן הריצה הכולל הוא O(m+n).

סעיף ד':

1. אם ניתן להכניס איבר לטבלה בהכרח Y[m,n] ריק ולכן נכניס את האיבר החדש למקום זה.
2. יהיו i=m ו-j=n אינדקסים.
3. נרוץ על השורה כל עוד Y[i-1,j] גדול מ-Y[I,j].
4. ברגע שהגענו לאיבר כזה (אם בכלל), נרוץ על העמודה שבה עצרנו כל עוד Y[I,j-1] גדול מ-Y[I,j].
5. ברגע שעצרנו נגיע למקום המדויק שאליו נכניס את האיבר. נכניס אותו ונמשיך באופן רקורסיבי על האיבר שאותו החלפנו.
6. אם ברקורסיה נגיע למצב בו Y[i-1,j] ו-Y[I,j-1] קטנים מ-Y[I,j] נעצור את הרקורסיה.

כמו בסעיף הקודם, אנחנו רצים עם האיבר שאותו רוצים להכניס במקרה הגרוע עד למקום ה-Y[1,1] כלומר O(m+n-1) שזה הוא O(m+n). אחרת נמשיך במהלך הדרך עם איבר אחר עד שנגיע למקום ה-Y[1,1] או שנגיע למצב בו כל האיברים מסודרים במקומם שזה גם O(m+n).

סעיף ה':

1. ניצור מערך חדש A בגודל n^2.
2. כל עוד הטבלה לא ריקה נבצע Extract Min על הטבלה.
3. את האיבר שקיבלנו נכניס למקום הבא הפנוי במערך A.

נבצע את Extract Min $O(n^2)$ פעמים כי יש במטריצה n*n n^2 איברים. ועבור כל איבר כזה זמן הריצה של Extract Min הוא O(n) לכן בסה"כ נקבל זמן ריצה של $O(n^3)$.

סעיף ו':

נניח שהאיבר שאותו מחפשים הוא *k*.

1. נבצע סריקה מהאיבר *Y[1,1]* כל עוד *Y[i+1,j]* גדול מ-*k* לכל *I<n* (סריקה על השורה)
2. ברגע שעצרנו (או שהגענו לתא האחרון) נבצע סריקה מאותו איבר כל עוד *Y[I,j+1]* גדול מ-*k* לכל *j<n*.
3. כעת אנחנו יודעים שכל מי שמימין לאיבר גדול ממנו ומי שמעליו קטן ממנו לכן נותר לבדוק רק את האיברים שנמצאים באותה שורה משמאלו. נבצע סריקה מאותו איבר כל עוד *Y[i-1,j]* גדול מ-*k*.

אם במהלך אף אחד מהשלבים לא מצאנו את *k* אז הוא בהכרח לא בטבלת יאנג.

נשים לב שבמקרה הגרוע נרוץ על כל השורה הראשונה *O(m)* ואחר כך על כל העמודה האחרונה *O(n)* ולבסוף נרוץ על כל האיברים שנותרו בשורה אליה הגענו *O(m-1)*. לכן בסה"כ זמן הריצה הוא *O(2m+n-1)* שזה *O(m+n)*.

שאלה 4

סעיף א'

אם n זוגי: ה-median נמצא בתא $\frac{n}{2}$ (האינדקסים מתחילים מ-1) ולכן לפניו יש $\frac{n}{2}-1$ איברים ואחריו $\frac{n}{2}$

איברים. כמו כן כל איבר שוקל $\frac{1}{n}$ ולכן סכום משקל האיברים שלפני median הוא
$\left(\frac{n}{2}-1\right)*\frac{1}{n}=\frac{n}{2}*\frac{1}{n}-\frac{1}{n}=\frac{1}{2}-\frac{1}{n}$ ומכיוון ש-n חיובי אז קבלנו סכום קטן מחצי. סכום משקל האיברים שאחרי median הוא $\frac{n}{2}*\frac{1}{n}=\frac{1}{2}$ כנדרש.

אם n אי זוגי: ה-median נמצא בתא $\frac{n+1}{2}$ (האינדקסים מתחילים מ-1) ולכן לפניו יש $\frac{n+1}{2}-1$ איברים ואחריו $\frac{n+1}{2}$

איברים. כמו כן כל איבר שוקל $\frac{1}{n}$ ולכן סכום משקל האיברים שלפני median שווה לסכום של האיברים אחריו והוא $\left(\frac{n+1}{2}-1\right)*\frac{1}{n}=\frac{n}{2}*\frac{1}{n}+\frac{1}{2}*\frac{1}{n}-\frac{1}{n}=\frac{1}{2}-\frac{1}{2n}$ ומכיוון ש-n חיובי אז קבלנו סכום קטן מחצי כנדרש.

סעיף ב

- בצע merge sort עבור n האיברים לפי משקלם.
- אתחל משתנה עזר sum=0. אתחל משתנה המייצג אינדקס i=1.
- כל עוד sum< 0.5
  - Sum = sum + i.weighted
  - I++
- החזר את i.weighted

זמן ריצה: merge sort – O(n log n) . הלולאה מתבצעת לכל היותר n פעמים ובתוכה מספר פעולות קבועות ולכן זמן הריצה שלה הוא O(n) ובסה"כ O(n log n).

סעיף ג'

getWeithtedMedian(array)

- מצא את ה- median ב- array
- Sum1=0, sum2=0
- עבור על n האיברים ועבור כל איבר בצע:
  - אם i.weighted < median.weighted
    - Sum1 = sum1 + i.weighted
  - אחרת
    - Sum2 = sum2 + i.weighted
- אם sum1<0.5 וגם sum2<=0.5
  - החזר את median
- אחרת אם sum1 >0.5
  - עבור על n האיברים והכנס למערך חדש newArray כל איבר המקיים i.weighted<median.weighted

- החזר את getWeithtedMedian(newArray)
- אחרת
  - עבור על n האיברים והכנס למערך חדש newArray כל איבר המקיים i.weighted>median.weighted
  - החזר את getWeithtedMedian(newArray)

זמן ריצה: פונקציה רקורסיבית ובה מספר פעולות קבועות ו- median – O(n) , ניתן לראות כי בכל קריאה רקורסיבית מקטינים את המערך בחצי ולכן נקבל $T(n) = T(\frac{n}{2}) + n + 1$ לכן לפי שיטת המאסטר T(n) = O(n)

<u>שאלה 5</u>

סדר הריצה לפי BFS הוא: A-B-D-E-F-H-C-G-I

סדר הריצה לפי DFS הוא: A-B-F-G-C-I-H-D-E

<u>שאלה 6</u>

סעיף א:

- נריץ BFS החל מקדקוד s.
- נשמור במשתנה עזר shortest את המרחק מ-t ל-s (t.d).
- נשמיט מגרף G את הצלע e.
- נריץ BFS על הגרף המתקבל החל מקדקוד s
- אם t.d = sortest אז e לא נמצאת על כל המסלולים הקצרים ביותר בין s ו- t. אחרת היא כן.

זמן ריצה בהתאם לסדר הפעולות לעיל:

- משתמשים ברשימת שכנויות אז כפי שנלמד בכתה O(V+E).
- O(1)
- מדובר ברשימת שכנויות ולכן גודל רשימת השכנויות של u1 הוא לכל היותר V ולכן כדי למחוק מהרשימה את u2 יש לעבור על הרשימה ולכן O(V). כדי למחוק את u1 מהרשימה של u2 באותו אופן נקבל O(V) ובסה"כ O(V).
- משתמשים ברשימת שכנויות אז כפי שנלמד בכתה O(V+E).
- O(1)

בסה"כ קבלנו O(V+E).

סעיף ב':

- נריץ BFS החל מקדקוד s.
- d1s = u1.d, d2s = u2.d
- נריץ BFS החל מקדקוד t.
- אם d1s + u2.d +1 = s.d או d2s + u1.d +1 = s.d החזר true . אחרת החזר false.

זמן ריצה בהתאם לסדר הפעולות לעיל:

- משתמשים ברשימת שכנויות אז כפי שנלמד בכתה $O(V+E)$.
- $O(1)$
- משתמשים ברשימת שכנויות אז כפי שנלמד בכתה $O(V+E)$.
- $O(1)$

בסה"כ קבלנו $O(V+E)$.