מבני נתונים – משימה 5

מאור אסייג 318550746

רפאל שטרית 204654891

המחלקה להנדסת חשמל ומחשבים, התכנית להנדסת מחשבים

מבני נתונים 202.1.1011

תשובות

1. ייעול זמן ריצה QuickSort

- **הערה** : בשאלה זו אנו מגדירים "חציון מתמטי" כאיבר האמצעי במערך הממוין.

תיאור כללי של הגישה : כפי שנלמד בהרצאה, $QuickSort$ במקרה הגרוע בעל זמן ריצה של $O(n^2)$ עבור n איברים. ראינו בהרצאה שעל ידי בחירה הסתברותית של ה Pivot באלגוריתמי העזר של $QuickSort$ יכולנו להוריד את תדירות הפעמים שהמקרה הגרוע יקרה, כלומר הצלחנו לעלות כך את זמן הריצה הממוצע. בשאלה זו נציע בחירה חכמה יותר של ה $Pivot$ כך שנוכל באופן וודאי לשפר את ביצועי זמן הריצה במקרה הגרוע ביותר.
לפני כל קריאה ל $Partation$ נחשב את האינדקס של החציון המתמטי באיברי המערך, בהתאם לגבולות המערך שקיבלנו מהקריאה למיון.
בשביל מיון המערך A נקרא ל $QuickSort(A,1,n)$.

הפונקציה QuickSort(A, p, r) ממיינת את המערך A :

1. אם $(p < r)$:
   1.1 נמצא את האינדקס של החציון בעזרת פונקציית עזר $index = find(A,p,r)$.
   1.2 נחליף בין החציון $A[index]$לבין האיבר האחרון במערך $A[r]$.

- מכאן האלגוריתם זהה למה שנלמד בהרצאות

   1.3 $q = Partation(A,p,r)$
   1.4 $QuickSoret(A,p,q-1)$
   1.5 $QuickSort(A,q+1,r$

זמן ריצה : קריאה לפונקציית $find$, כפי שיוכח בהמשך – תעלה $O(n)$. בגלל שכל קריאה ל $Partation$ $O(n)$ בהשוואה לחציון המערך בהתאם לגבולות (כל האיברים הגדולים מהחציון מימינו, והקטנים ממנו משמאלו) – אנו כל הזמן מקטינים את המערך הנקרא ל $QuickSoret$ בחצי, ולכן :

$$T(n) = T_{find}(n) + T_{Partation}(n) + 2T\left(\frac{n}{2}\right)$$

$$= \cdots \doteq \log_2 n * O(n) = n * log_2(n)_{\blacksquare}$$

- נוסחת הנסיגה מניבה $T(n) = T(1) + 2O(n)\sum_{k=1}^{k=i}\frac{1}{2^i}$ כאשר $\frac{1}{2^i} = 1 \to i = \log_2 n$ כאשר T(1) הינו זמן קבוע.

הפונקציה $Find(A,p,r)$ מוצאת את האינדקס של החציון המתמטי במערך A בטווח האינדקסים [p → r] :

- בפונקציה זו נעזרנו באלגוריתם מוכר, $Select$.

1. נחלק את המערך A לתתי מערכים לוגיים, כל תת מערך עם 5 איברים.
2. נמצא את החציון המתמטי בכל תת מערך, נכניס את החציונים למערך עזר B (שגודלו חמישית מגודל A).

3. נקרא רקורסיבית (תנאי עצירה יהיה טווח האינדקסים במערך קטן או שווה ל 5) ל - $Find(B,1,n)$ **בשלבים 1-3** למציאת החציון של החציונים.
4. נריץ חיפוש על המערך המקורי A למציאת האינדקס של הערך המתמטי עבור החציון של החציונים.

זמן ריצה : שלב 1 יעלה זמן לינארי לקביעת המקטעים הלוגיים, $O(n)$.
שלב 2 עבור כל תת מערך זהו זמן קבוע למציאת החציון המתמטי במערך של 5 איברים, ובסה"כ עבור $\frac{n}{5}$ תתי מערכים כאלו נקבל $O(n)$.
שלב 3 ישנה קריאה רקורסיבית לשלבים 1 -3 ובשלב 4 ישנו חיפוש לינארי על המערך A למציאת האינדקס של האיבר המייצג את החציון.

$$T(n) = O(n) + \frac{n}{5} * O(c) + T_{step\ 1to3}\left(\frac{n}{5}\right) + O(n) =$$

$$= c_2 O(n) + \ T_{step\ 1to3}\left(\frac{n}{5}\right) = \cdots = T(1) + \ c_2 O(n) * \sum_{k=1}^{k=i} \frac{1}{5^i} =$$

$$= \mathrm{O}(1) + c_2 O(n)\left[\frac{\frac{1}{5^{\log_5 n}} - 1}{\frac{1}{5} - 1}\right] = O(1) + c_2 O(n)\left[\frac{\frac{1}{n} - 1}{-\frac{4}{5}}\right] =$$

$$= O(1) + c_2 O(n)\left[\frac{\frac{1}{n} - 1}{-\frac{4}{5}}\right] = \frac{4}{5} c_2 O(n) + O(1) = O(n)_{\blacksquare}$$

2. חיפוש במערך בסדר עולה שסובב

תיאור כללי של הגישה : בסיוע גישת $Divide\ and\ Conquer$ נמצא את האינדקס שמייצג את האיבר הכי גדול במערך ("נקודת התפר" בין תחילת המערך לסופו), לאחר מכן נקרא לחיפוש בינארי עבור הצד השמאלי מנקודת התפר, וחיפוש בינארי מימין לנקודת התפר.

$Find2(A,x)$

הפונקציה מוצאת את האינדקס של הערך x במערך A :

1. נמצא את האינדקס של נקודת התפר בעזרת קריאה של $index = find1(A,1,n)$
2. נקרא לחיפוש בינארי על צד ימין של נקודת התפר ועל צד שמאל של נקודת התפר
$Temp1 = BinarySearch(A,1,index,x)$
$Temp2 = BinarySearch(A,index,n,x)$
3. נחזיר את מה שלא $null$ מבין $Temp2$ ו $Temp1$.

זמן ריצה : שלב 1 נקרא לפונקציה $find1$ יעלה $O(log_2 n)$ כפי שנראה בהמשך, בשלב 2 כל חיפוש בינארי יקח $O(log_2 n)$.

$c_1 \in (0,1)$

$$T(n) = O(\log_2 n) + T_{BinarySearch}(n * c_1) + T_{BinarySearch}\left(n * (1 - c_1)\right) = 3O(log_2 n) = O(log_2 n)_{\blacksquare}$$

$Find1(A,Left,Right\ x)$

הפונקציה מוצאת את נקודת התפר במערך A :

1. $If\ (\ A[Right]\ <\ A[Left]\ )$
   1.1 $if\ (\ Right\ +\ 1\ =\ Left\ )$
      1.1.1 החזר $Right$.
      1.1.2 אחרת $Temp3 = null$.
2. נקטין את הבעיה ל 2 חלקים קטנים יותר
$$Temp1 = Find1(A,\frac{Right}{2}+1,Right,x)$$
$$Temp2 = Find1(A,Left,\frac{Right}{2},x)$$
3. החזר את מה שלא $null$ מבין $Temp3$, $Temp1$, $Temp2$.
אם כולם $null$ החזר $null$.

זמן ריצה : האלגוריתם מבוסס על חיפוש בינארי ולכן יעלה $T(n) = O(log_2 n)$.

3. טבלת יאנג

א. עם האיברים $\{2,3,4,5,8,9,12,14,16\}$

2	3	4	5
8	9	12	14
16	2	∞	∞
∞	∞	∞	∞

ב. הוכיחו כי טבלת יאנג בגודל nxm ריקה אם $Y[1,1] = \infty$.

← נניח בשלילה שהטבלה לא ריקה, אזי קיים אינדקס i_x ואינדקס i_y בתחום כך ש $Y[i_x, i_y] \geq Y[1,1]$ (על פי הגדרת הטבלה), אך $Y[1,1] = \infty$ לפי הנתון ולכן לא קיימים איברים שיקיימו תנאי זה, ובפרט לא קיימים אינדקסים כאלו ולכן הטבלה ריקה\מלאה ב ∞.

ג. הוצאת האיבר המינימלי מטבלה בגודל $n * m$.

$ExtractMin$

1. נחזיר את $Y[1,1]$.
2. הכנס את $number$ ל $Y[1,1]$.
3. נחלחל את הערך בטבלה ע"י קריאה ל $SpecialInsert(\infty, 1,1)$.

זמן ריצה : שלב 1 יעלה $O(1)$ ושלב 2 יעלה כפי שיפורט בהמשך $O(m+n)$ ובסה"כ $O(m+n)$ $T(n) = O(m+n)$.

$SpecialInsert(number, tempX, tempY)$

תיאור כללי – נבצע השוואה עם השכן הימני והשכן התחתון, במידה ושתיהם גדולים מהאיבר הנוכחי נחליף את האיבר הנוכחי עם הגדול מבין השכנים הנ"ל, אחרת נחליף עם השכן שגדול מהאיבר הנוכחי.

1. אם $tempX < n$ וגם $tempY < n$

1.1 אם $Y[tempX, tempY] > Y[tempX+1, tempY]$ או $Y[tempX, tempY] > Y[tempX, tempY+1]$

1.1.1 החלף את $Y[tempX, tempY]$ עם $\max(Y[tempX, tempY+1], Y[tempX+1, tempY])$

1.1.2 קרא לפונקציה עם הערכים החדשים של $number$ $SpecialInsert(number, new\ X, new\ Y)$

2. אחרת אם $tempX < n$ וגם $tempY > n$

2.1 אם $Y[tempX, tempY] > Y[tempX+1, tempY]$

2.1.1 החלף את $Y[tempX, tempY]$ עם $Y[tempX+1, tempY]$

2.1.2 קרא לפונקציה עם הערכים החדשים של $number$ $SpecialInsert(number, tempX+1, tempY)$

3. אחרת אם $tempY < n$ וגם $tempX > n$

3.1 אם $Y[tempX, tempY] > Y[tempX, tempY+1]$

3.1.1 החלף את $Y[tempX, tempY]$ עם $Y[tempX, tempY+1]$

3.1.2 קרא לפונקציה עם הערכים החדשים של $number$ $SpecialInsert(number, tempX, tempY+1)$

זמן ריצה : במקרה הגרוע נכניס איבר שיהווה את הערך הכי גדול בטבלה, ולכן ה"חלחול" שלו בטבלה ייערך במקרה הגרוע כמספר השורות והעמודות (ימינה עד הסוף ואז למטה עד הסוף או הפוך) ובסה"כ $T(n) = O(m+n)$.

ד. הכנסת איבר, הפונקציה תיקרא עם $Insert(number, n, m)$

תיאור כללי – נבצע השוואה עם השכן השמאלי והשכן שמעליו, במידה ושתיהם קטנים מהאיבר הנוכחי נחליף את האיבר הנוכחי עם הקטן מבין השכנים הנ"ל, אחרת נחליף עם השכן הקטן מהאיבר הנוכחי.

$Insert(number, tempX, tempY)$

1. הכנס את $number$ ל $Y[tempX, tempY]$.
2. בצע את הלוגיקה של $SpecialInsert$ בצורה "הפוכה" - כאשר אנו דואגים לא להגיע לקצוות ($tempX > 0, TempY > 0$) ומבצעים החלפה עם המינימום מבין השכן העליון והשכן השמאלי.

זמן ריצה : במקרה הגרוע נכניס איבר שיהווה את הערך הכי גדול בטבלה, ולכן ה"חלחול" שלו בטבלה ייערך במקרה הגרוע כמספר השורות והעמודות (שמאלה עד הסוף ואז למעלה עד 0 או הפוך) ובסה"כ $T(n) = O(m + n)$.

ה. מיון n^2 איברים באמצעות טבלת יאנג בגודל $n * n$

1. נבצע הכנסה לטבלת יאנג n^2 מספרים
2. נבצע n^2 פעמים $ExtarctMin$ לקבלת המספרים בסדר עולה

זמן ריצה : הכנסה לטבלת יאנג ו $ExtractMin$ יעלו $O(n + n) = O(n)$. ולכן בסה"כ נקבל :

$$T(n^2) = n^2 * O(n) + n^2 * O(n) = O(n^3)$$

I. בדיקה האם מספר $number$ נתון קיים בטבלת יאנג n * m

תיאור כללי – נדמה הכנסה לטבלת יאנג, נקבע ל$number$ אינדקסים לוגיים היידמו את איבר הקצה בטבלה $[n,m]$ ונבצע את אותה הלוגיקה כמו הכנסה (להחליף עם השכן הקטן ביותר מבין השכן העליון והשכן השמאלי) – תוך כדי בדיקה האם ביקרנו בערך השווה ל $number$, אם הפעולה המדמה הכנסה הסתיימה ולא ביקרנו בערך השווה לו נחזיר $false$.

דגש – האינדקסים מתחילים בטבלה מ [1,1].

1. נקבע אינדקסים לוגיים $tempX = m$, $tempY = n$.
2. כל עוד $tempX \geq 1$ וגם $tempY \geq 1$
   - **2.1** **אם** $tempX > 1$ וגם $tempY > 1$
     - **2.1.1** בדוק האם השכן העליון או השכן השמאלי שווים לי, אם כן החזר $true$.
     - **2.1.2** אחרת, אם אחד השכנים קטן מהערך של האיבר הנוכחי החלף עם השכן הקטן ביותר מבין השכן השמאלי והשכן העליון ועדכן את האינדקסים הלוגיים בהתאם.
   - **2.2** **אחרת אם** $tempX > 1$ וגם $tempY = 1$
     - **2.2.1** בדוק האם השכן השמאלי שווה לאיבר הנוכחי, אם כן החזר $true$.
     - **2.2.2** אחרת, אם השכן השמאלי קטן מהאיבר הנוכחי החלף ביניהם ועדכן את האינדקסים הלוגיים בהתאם.
   - **2.3** **אחרת אם** $tempX = 1$ וגם $tempY > 1$
     - **2.3.1** בדוק האם השכן העליון שווה לאיבר הנוכחי, אם כן החזר $true$.
     - **2.3.2** אחרת, אם השכן העליון קטן מהאיבר הנוכחי החלף ביניהם ועדכן את האינדקסים הלוגיים בהתאם.
3. החזר **false** .

זמן ריצה : במקרה הגרוע נכניס איבר שיהווה את הערך הכי קטן בטבלה, ולכן ה"חלחול" שלו בטבלה במקרה הגרוע ייערך

כמספר השורות והעמודות (ימינה עד הסוף ואז למטה עד הסוף או הפוך) ובסה"כ $T(n) = O(m+n)$.

4. weighted median

א. הוכיחו כי עבור $X_1, X_2 \dots, X_n$ עם משקלים זהים $W_i = \frac{1}{n}$ החציון במערך הינו החציון המשקלי.

← החציון במערך הינו האינדקס האמצעי, כלומר $X_{\frac{n}{2}}$.

נניח שזהו האינדקס של החציון המשקלי, נבדוק זאת :

$$\sum_{i=1}^{i=\frac{n}{2}} \frac{1}{n} = \frac{1}{n} * \frac{n}{2} = \frac{1}{2} \triangleq wieghted\ median \blacksquare$$

ב. מציאת ה $wieghted\ median$ עבור n איברים

- ניתן לחשוב על הנתונים כאל 2 מערכים (האינטגרציה בין המערכים זהה לזו של איברים סדורים), מערך ה X_i ומערך של W_i כאשר $i \in [1, n]$.

1. נפעיל על המערך W_i את אלגוריתם המיון $mergeSort$ כפי שנלמד בכיתה, תוך כדי אבחנה שאם החלפנו איברים במערך W_i נחליף את האיברים עם אותם אינדקסים באופן מקבילי במערך X_i.

- נשים לב ש $mergeSort$ שומר על יציבות, תכונה הכרחית למציאת החציון המשקלי על פי הגדרתו.

2. נבצע סריקה על המערך W_i תוך כדי צבירת המשקלים עד שנגיע למשקל חצי ונחזיר את האיבר במערך X_i המתאים לאינדקס שמצאנו.

זמן ריצה : מיון מערך W_i באמצעות $mergeSort$ יעלה $O(n*\log_2 n)$ וסריקה על איברי אותו מערך וצבירת הערך הרצוי תעלה במקרה הגרועה סדר גודל של $O(n)$ ולכן **בסה"כ** נקבל כרצוי :

$$T(n) = O(n) + O(n*\log_2 n) = O(n*\log_2 n)_{\blacksquare}$$

ג. מציאת ה $wieghted\ median$ עבור n איברים ב $O(n)$

- הפונקציה נקראת עם הערכים $(A,0)$ מוצאת את החציון המשקלי בעזרת אלגוריתם המוצא חציון במערך ב $\theta(n)$ במערך A :

$$Find(A,w)$$

1. אם גודל המערך A שווה ל 1 החזר את האיבר הבודד.
2. ניצור מערך עזר B, מערך עזר C, משתנה עזר W_b
   - B – ישמור איברים הקטנים מהחציון של A.
     C – ישמור איברים הגדולים מהחציון של A.
     W_b – יצבור את המשקל בהוספה ל B.
3. מצא את החציון של A באמצעות אלגוריתם העזר.
4. עבור כל איבר במערך A, בצע :
   4.1 אם האיבר הנוכחי קטן מהחציון של A, הכנס ל B והוסף את משקלו ל W_b.
   4.2 אחרת הכנס את האיבר ל C.
5. אם $W_b + W > \frac{1}{2}$
   - החציון המשקלי יהיה מתוך ההגדרה במערך B
   5.1 קרא ל $Find(B,W)$
6. אחרת קרא ל $Find(C,W_b)$

זמן ריצה : בכל קריאה לפונקציה אנו עוברים על כל איברי המערך שהתקבל, ומכיוון שאנו מחלקים לפי החציון מתוך ההגדרה בכל קריאה אנו חוצים את האיברים אותם נותר לנו לנתח, על כן :

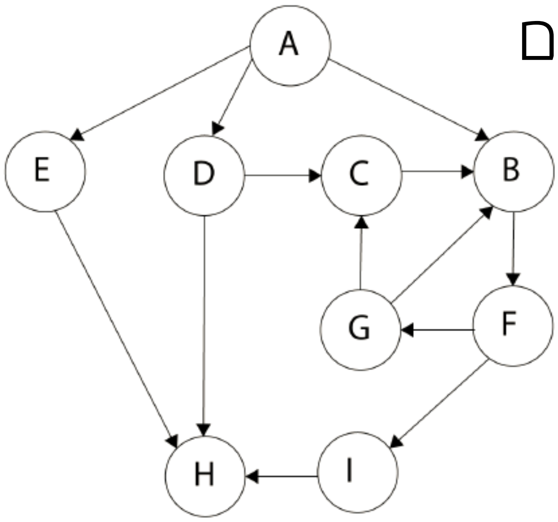
$$T(n) = O(n) + T\left(\frac{n}{2}\right) = \cdots = \sum_{k=0}^{k=i} \frac{n}{2^i} + T(1) =$$

$$= \sum_{k=0}^{k=\log_2 n} \frac{n}{2^i} + O(1) = (n * \frac{\left(\frac{1}{2}\right)^{\log_2 n} - 1}{\frac{1}{2} - 1}) + O(1) =$$

$$= n * \frac{\frac{1}{n} - 1}{-\frac{1}{2}} + O(1) = \frac{1-n}{-\frac{1}{2}} + O(1) = 2n - 2 + O(1)$$

$$= O(n)_\blacksquare$$

5. גרפים



א. סדר ביקור BFS החל מקודקוד A
אלגוריתם חיפוש לרוחב

$$A \to B \to D \to E \to F \to C \to H \to G \to I$$

- ניתן להחליף את הסדר של H עם C ואת G עם I, אך נהוג לתת עדיפות לאותיות בסדר הלקסיקוגרפי.

ב. סדר ביקור DFS החל מקודקוד A
אלגוריתם חיפוש לעומק

$$A \to B \to F \to I \to H \to G \to C \to D \to E$$

6. גרפים – מסלולים קצרים

נתון גרף לא מכוון $G(V,E)$, שני צמתים $t,s \in V$ וקשת $e=(u_1,u_2)\in E$

א. האם הקשת *e* נמצאת על כל המסלולים הקצרים ביותר בין *s* ל *t* ?

אלגוריתם כללי :

1. נשתמש באלגוריתם BFS על הקודקוד s ליצירת עץ חיפוש רוחבי כשהשורש הינו s. במקרה זה נוכל לדעת את המרחק הקצר ביותר לקודקוד t, על ידי גישה ל $t.d$.
2. נמחק את הקשת e ונריץ שוב BFS על הקודקוד s.
3. נשווה את $t.d$ הנוכחי ל $t.d$ משלב 1.

- $t.d$ הינו מינמלי לפי BFS, ולכן אם ערכו השתנה אזי בהכרח הקשת e הינה חלק מכל המסלולים הקצרים.

ניתוח זמן ריצה :

הרצת BFS על G תעלה $O(|V|+|E|)$, חיפוש t בעץ החיפוש תלוי בשיטת המיפוי של הקודקודים (נניח שיטת רשימת סמיכויות) $O(1)$, **ובסה"כ** :

$$T\big(G(V,E)\big)=2*O(|V|+|E|)+O(1)=O(|V|+|E|)_\blacksquare$$

ב. האם הקשת *e* נמצאת על מסלול קצר כלשהו בין *s* ל *t* ?

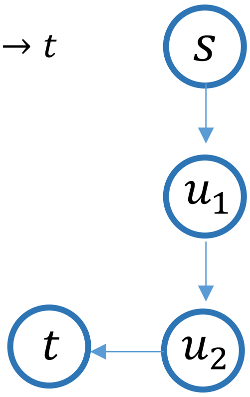
אלגוריתם כללי :

1. נשתמש באלגוריתם BFS על הקודקוד s ליצירת עץ חיפוש רוחבי כשהשורש הינו s. במקרה זה נוכל לדעת את המרחק הקצר ביותר לקודקוד t, על ידי גישה ל $temp1=t.d$.
2. נמצא את המרחק מהקשת e ל s ע"י $temp2=u_2.d$.
3. נריץ BFS על u_2 ונחשב את המרחק של הקשת e מ t $temp3=t.d$
4. e יהיה על מסלול קצר כלשהו במידה ויתקיים : $temp1=temp2+temp3$

$temp1 : s \rightarrow u_1 \rightarrow u_2 \rightarrow t$

$temp2 : s \rightarrow u_1 \rightarrow u_2$

$temp3 : t \rightarrow u_2$



ניתוח זמן ריצה :

הרצת BFS על G תעלה $O(|V|+|E|)$, חיפוש t בעץ החיפוש תלוי בשיטת המיפוי של הקודקודים (נניח שיטת רשימת סמיכויות) $O(1)$, **ובסה"כ** :

$$T\big(G(V,E)\big) = 2 * O(|V|+|E|) + O(1) = O(|V|+|E|)_{\blacksquare}$$

Until next time, thank you.