שאלה 1 ו-2
דומות בצורה מחשידה לממ"ן של 305243115
אני פוסל אותן

פתרון ממ"ן 14

אלגוריתמים 20417, סמסטר 2020א

אור חליפה ת.ז. 325373959

שאלה 1:

נתון כי השריג הינו ריבועי מגודל $n \times n$ הצעדים האפשריים מהסבר הפעולות הנתונות הם: ימינה
ימינה - למעלה
ימינה למטה

תחילה אציג את האלגוריתם, אוכיח את נכונותו ואנתח את הסיבוכיות

האלגוריתם:

```
OPT = [n] [n+2]          // נגדיר מערך דו-מימדי, מטריצה מסדר גודל nXn+2
for i from 1 to n
    for j from 1 to n+2
        OPT[i][j] ← ∞

for i from 1 to n
    OPT[i][j] ← c(1,j)

for i from 2 to n
    for j from 1 to n
        OPT[i][j] ← min { OPT(i-1, j-1), OPT(i-1, j), OPT(i-1, j+1) }

min c ← min OPT(n,j)
    if OPT[n][j] = min c
        then j_k ← j

for k from n to 1
    print [k][j_k]
    j_{k-1} ← j_k
    if (OPT [k-1][j_k - 1] + c(k, j_k)) = OPT[k, j_k]
        then j_{k-1} ← j_k + 1
        else if (OPT[k-1, j_k + 1] + c(k, j_k)) = OPT[k, j_k]
            then j_{k-1} ← j_{k+1}
        else j_{k-1} ← j_k
```

Scanned by CamScanner

נכונות האלגוריתם:

הגדרנו מטריצה המגדירה $OPT[i,j]$ את המסלול האופטימלי, שמתחיל בנקודה בשכבה השמאלית ומגיע ל-(i,j) בשכבה הימנית.

$$OPT(i,j) = \begin{cases} \infty & j=0 \text{ או } j=n+1 \\ c(i,j) & i=1 \\ c(i,j) + \min\{OPT(i-1,j-1), OPT(i-1,j), OPT(i-1,j+1)\} & \text{אחרת} \end{cases}$$

הוספת 2 שורות שעזרו לנו לחשב את הערך האופטימלי לא תשפיע על זמן הריצה מכיוון שהן מאותחלות כבר לפני הריצה של האלגוריתם.

האתחול של המטריצה מתרחש בשורות 1-6. מכיוון שהמסלול מתחיל מהעמודה הראשונה הוספנו לכל תא את ערך הנקודה שהוא מייצג, $c(i,j)$.

נרצה למצוא את $\min OPT(n,j)$ המסלול הקצר מבין העונה על הדרישות.
נשחזר את המסלול צעד אחר צעד כך שבכל צעד אנחנו מסתמכים על הצעד הקודם, לכן מדובר בתכנות דינמי.

לאחר מכן, נחשב את ערך המסלול המינימלי לגבי הנקודה הסופית משכבה הימנית כך:
(n,j_n): נבדוק איזה ערך של j מתוך $n \geq j_n \geq 1$ מקיים $\min OPT(n,j) = OPT[n][j_n]$.
בהינתן הנקודה j_k נרצה למצוא מה j_{k-1} מבין האפשרויות: j_k-1, j_k, j_k+1.
נבדוק עבור איזה ערך מתקיים:

$$OPT[k-1][j_{k-1}] + c(k,j_k) = OPT[k,j_k]$$

נקבע את j_{k-1} לפי האפשרות המקיימת את המשוואה.

כי הצעדים מתבצעים בכיוון ימני אזי $\forall(2 \leq k \leq n): i_{k-1} = i_k - 1$

כך נקבל סדרת אינדקסים $[(i_1,j_1),\dots,(i_n,j_n)]$ שמגלה את המסלול האופטימלי.

Scanned by CamScanner

חישוב סיבוכיות:

בשורות 1-10 ממלאים ע"י מטריצה מסדר גודל $n(n+2)=O(n^2)$

בשורה 10-12 מוצאים מינימום בכל n הערכים, $O(n)$, סורקים את הערכים ומשווים אותם למינימום $O(n)$.

בשורות 20-23 מבצעים n פעמים מספר קבוע של פעולות $O(n)$.

ולכן הסיבוכיות היא $O(n^2)$.

Scanned by CamScanner

שאלה 2:

נמיין את התיבות לפי הרוחב שלהן, כאשר לכל שתי תיבות i, j
$j > i \Longleftarrow w(j) > w(i)$

```
1.  HEAP-SORT(A) //

2.  for i from 1 to n
3.      OPT[i] = null
4.      PREV[i] = null

5.  for i from 1 to n
6.      for j from 1 to n

7.          OPT[i] <- max{ h(i), max(h(i) + OPT(j)) }

8.          if (h(i) > max (h(i) + OPT(j)))

9.              then PREV[i] <- null
10.             else PREV[i] <- j , PREV[j] <- null

11. max h <- max OPT(i)

12. for i from 1 to n
13.     if (min h = OPT(i))
14.     then max i <- i

15. while PREV[i] ≠ null
16.     do print PREV[i] , i <- PREV[i]
```

נכונות והסבר האלגוריתם:

תחילה נבצע מיון של התיבות לפי הרוחב שלהם ולפי היחס כי $j > i \Longleftarrow w(j) > w(i)$
נגדיר את OPT להיות גובה המגדל היציב הגבוה ביותר כאשר תיבה i בראשו.

כעת נרצה לחשב את OPT: שני אפשרויות: א. התיבה i עומדת לבדה וגובהה $h(i)$
ב. התיבה i היא חלק ממגדל יציב וגובהה המקסימלי
יוגדר כך: $\max(h(i) + OPT(j))$

בשורות 5-8 נבצע בחירה המתייחסת ל-2 האפשרויות בכך שנחשב את 2 הערכים ונבחר את הגדול מבין השניים.
נמצא כי:

$$OPT(i) = \max(h(i), \max(h(i) + OPT(j)))$$

התיבות היחידות שאנחנו שמים מעל תיבה כלשהי קטנות ממנה באורך וברוחב ובכך המגדל נשאר יציב.
בכל רמה נוסיף את גובה התיבה לשיקול הגובה המתאים ונתבסס רק על קריאות קודמות של OPT.

בעזרת PREV נשמרת התיבה בה בחרנו את i, שורות 9-10.

בשורה 11 נמצא את הגובה המקסימלי OPT ובכך גילינו את התיבה בראש המגדל.

בשורות 12-16 אנחנו סורקים את המערך PREV כדי לשחזר את רצף התיבות. נקבל בסוף התהליך את סדר התיבות
בצורה הפוכה:
נמצא אינדקס i עבורו $OPT(i)$ מקסימלי, נדפיס את הערך $PREV[i]$ ונשמור את i להיות $PREV[i]$
(שורות 15-16).

Scanned by CamScanner

חישוב סיבוכיות:

שורה 1: מיון האיברים לפי הרוחב בשימוש HEAP-SORT אורך $O(n\lg n)$.

שורות 2-4: הגדרת 2 מערכים בגודל n. $2n = O(n)$.

שורות 5-10: הכנסה ל- n תאים, במקרה הגרוע נצטרך את הערך האופטימלי לפי 2 קריטריונים כי מיינו רק את הרוחב ולא את הגובה ולכן $O(n^2)$

שורות 11-14: ביצוע פעולות מספר זמן קבוע של n איברים: $O(n)$

שורות 15-16: סריקת המערך PREV, במקרה הגרוע נצטרך לסרוק את כולו. למערך n איברים ולכן: $O(n)$.

בסה"כ: $O(n^2)$.

שאלה 3

סעיף א': נגדיר 3 פולינומים לינאריים:

$$\begin{cases} f(x) = ax+b \\ g(x) = cx+d \\ h(x) = kx+v \end{cases}$$

לפי הנתונים בשאלה, פולינום מקבל את אותם הערכים כל עוד מכילים את אותן הנקודות, ולכן עבור הנקודות:

$$x_i \ , \ x_{j+1}$$

$$P_{i,j+1}(x_i) = P_{i,j}(x_i) \qquad \text{(1) סימון}$$

$$P_{i,j+1}(x_{j+1}) = P_{i+1,j+1}(x_{j+1}) \qquad \text{(2) סימון}$$

תנאי (1) מתקיים $\Longleftrightarrow$ א. $g(x_i)=0 \Longleftarrow g(x) = (x-x_i)\ ,\ c\overset{\text{הנחה}}{=}1$

ב. $\dfrac{f(x_i)}{h(x_i)} = 1$

תנאי (2) מתקיים $\Longleftrightarrow$ א. $f(x_{j+1}) = 0 \Longleftarrow f(x) = (x-x_{j+1})\ ,\ a\overset{\text{הנחה}}{=}1$

ב. $\dfrac{g(x_{j+1})}{h(x_{j+1})} = -1$

סה"כ, $\dfrac{f(x_i)}{h(x_i)} = 1 \ \wedge \ \dfrac{g(x_{j+1})}{h(x_{j+1})} = -1 \Longleftarrow$

$$\frac{g(x_{j+1})}{h(x_{j+1})} + \frac{f(x_i)}{h(x_i)} = 0 \Longrightarrow kx_i^2 + 2kx_ix_{j+1} - kx_{j+1}^2 = -k(x_i+x_{j+1})^2 = 0$$

$$\Longrightarrow \begin{cases} -k=0 \Longleftrightarrow k=0 \\ (x_i+x_{j+1})^2 = 0 \Longleftrightarrow x_i = -x_{j+1} \end{cases}$$

יש 2 פתרונות, אך נוכל לבחור 2 נקודות שלא יקיימו $x_i = -x_{j+1}$ ולכן $k=0$

נקבל

$$\begin{cases} f(x) = x_i - x = r(x) \\ g(x) = x - x_{j+1} = q(x) \\ h(x) = x_i - x_{j+1} = s(x) \end{cases}$$

המקיימים את הדרישה

Scanned by CamScanner

סעיף ב:

```
1. OPT ← [n][n]
2. for i from 1 to n
     3. do OPT[i][j] ← y_i
4. for k from 2 to n
     5. do i ← 1
     6. for j from k to n
          7. do OPT[i][j+1] ← ((x_{j+1} - x) OPT[i][j] - (x_j - x) OPT[i+1][j+1] (x)) / (x_{j+1} - x_i)
          8. i ← i+1
9. return OPT[1, n]
```

$$OPT[i][j+1] \leftarrow \frac{(x_{j+1}-x)\,OPT[i][j]-(x_j-x)\,OPT[i+1][j+1]\,(x)}{(x_{j+1}-x_i)}$$

הוכחת נכונות והסבר האלגוריתם:

הנכונות מתבססת על נתוני השאלה והנוסחה בסעיף א'
בשורות 2-3 מילאנו את האלכסון הראשי של מטריצת OPT
בשורות 4-8 נחשב את הערכים שבאלכסון המטריצה ונמלא בהתחלה את האלכסון שמתחיל ב $(1,2)$ ומסתיים ב $(n-1,n)$ ואז את האלכסון שמתחיל ב $(1,k)$ ומסתיים ב $(n-k+1,n)$
שכן נ"י כל איטרציה לערכים שחושבו בריצה הקודמות ב OPT.

כלומר OPT הוא מערך דו-מימדי, מטריצה, מסדר גודל $n \times n$ כאשר $OPT[i][j]$
הוא האינטרפולציה של הנקודות $(x_i, y_i), \dots, (x_j, y_j)$.

אתחלנו את כל $OPT[i][j]$ להיות פולינומי האינטרפולציה של הנקודה (x_i, y_i)

לאחר מילוי המערך התשובה הסופית, האינטרפולציה של כל n הנקודות תמצא בתא $OPT[1][n]$.

חישוב סיבוכיות:

שורה 1: יצירת מערך - בזמן קבוע

שורות 2-3: ריצה על אלכסון ראשי של מטריצה מסדר גודל $n \times n$ - $O(n)$.

שורות 4-8: n איטרציות: הלולאה הפנימית תלוי במקרה הגרוע $n-1$ פעמים ושם נבצע הכל באמד מן האיטרציות פעולות בזמן קבוע

$$O(1) \cdot (n-1)n = O(n^2)$$

שורה 9. החזרה $O(1)$.

בסה"כ $O(n^2)$.

Scanned by CamScanner

סעיף ג:

נחשב את הנקודות:

$$\begin{aligned} P(-2) &= 46 \\ P(-1) &= 2 \\ P(0) &= 0 \\ P(1) &= 10 \\ P(2) &= 98 \end{aligned} \quad \Rightarrow \quad (-2,46)\ ;\ (-1,2)\ ;\ (0,0)\ ;\ (1,10)\ ;\ (2,98)$$

ארצה לראות כי יוחזר $p(x)$ מהאלגוריתם.

יש לנו 5 נקודות אז $n=5$.

נאתחל תחילה מערך בגודל $5X5$.

שורות 2-3 יניבו:

$$OPT = \begin{bmatrix} 46 & & & & \\ & 2 & & & \\ & & 0 & & \\ & & & 10 & \\ & & & & 98 \end{bmatrix}$$

נחשב את $OPT[1][5]$: (שורות 8-1, לפי סדר הפעולה הרקורסיבי):

$$OPT[1][2] = \frac{(-x+x_2)\,OPT[1][1] - (-x+x_1)\,OPT[2,2]}{-x_1+x_2} = \frac{(-x+1)46-(-x-2)2}{1} =$$

$$= -46x-46+2x+4 = -44x-42$$

באופן דומה:

$$OPT[2][3] = \frac{2(-x+0)-0(-x-1)}{1} = -2x$$

$$OPT[3][4] = 10x\ ,\quad OPT[4][5] = 88x-78\ ,\quad OPT[1][3] = 21x^2+19x,$$

$$OPT[2][4] = 6x^2+4x\ ,\quad OPT[3][5] = 39x^2-29x.$$

$$OPT[1][4] = \frac{(21x^2+19x)(-x+1)-(6x^2+4x)(-x-2)}{3} = \frac{-15x^3+18x^2+27x}{3} =$$

$$= -5x^3+6x^2+9x$$

$$OPT[2][5] = \frac{(6x^2+4x)(-x+2)-(39x^2-29x)(-x-1)}{3} = \frac{33x^3+18x^2-21x}{3} =$$

$$= 11x^3+6x^2-7x$$

ולסיום:

$$OPT[1][5] = \frac{OPT[1][4](-x+x_5) - OPT[2][5](-x+x_1)}{-x_1+x_5} =$$

$$= \frac{(-5x^3+6x^2+9x)(-x+2) - (11x^3+6x^2-7x)(-x-2)}{4} = \frac{16x^4+12x^3+8x^2+4x}{4} =$$

$$= 4x^4+3x^3+2x^2+x = x+2x^2+3x^3+4x^4 = P(x)$$

כלומר האלגוריתם אכן מחזיר את $P(x)$.

Scanned by CamScanner

שאלה 4.

בסה"כ תשובה טובה ומתומצתת 20\15

סעיף א':

האלגוריתם מחשב את המרחקים המינימליים מהקודקוד r לשאר הקודקודים בגרף G.
כלומר, מחשב את המרחק הקצר ביותר מבין הצלעות מ-v אל שאר הצמתים בגרף
כאשר הנתון ה-$A[v]$ נמצא המרחק המינימלי בין הצומת v ל-r.

טענת האינדוקציה: שבסוף האיטרציה ה-i של הלולאה החיצונית, $A[v]$ יכיל את הערך '∞' אם"ם אין מסלול באורך i לכמות מ-r ל-v. אחרת יכיל משקל של מסלול כלשהו מ-r ל-v ומשקל זה יהיה קטן או שווה למשקל כל המסלולים מ-r ל-v שאורכם לכל היותר i.

בסיס האינדוקציה: לפני הריצה הראשונה של הלולאה החיצונית, הצומת היחיד שיש אליו מסלול באורך 0 הוא r עצמו ומשקל המסלול הוא 0, לשאר הצמתים אין מסלול באורך 0 הוא r עצמו ומשקל המסלול הוא 0. לשאר הצמתים אין מסלול באורך 0. לכן האתחול תואם את הטענה.

צעד האינדוקציה: נניח נכונות עבור n ונוכיח עבור $n+1$.

לפי הנחת האינדוקציה לכל $v \in V$, $A[v]$ מכיל משקל של מסלול מ-v ל-r שהוא קטן או שווה למשקל כל המסלולים מ-v ל-r שאורכם לכל היותר n.

נניח בשלילה כי הערך $A[v]$ מפר את טענת האינדוקציה, כלומר, בסוף האיטרציה ה-$n+1$ יש מסלול באורך $n+1$ מ-r ל-v שמשקלו קטן מ-$A[v]$.

אבל, במקרה הזה, הקשת (u,v) היא האחרונה במסלול. אבל מכיוון שיש מסלול אל u באורך n לפי הנחת האינדוקציה, משקל המסלול יהיה קטן או שווה ל-$A[u]+c((u,v))$ בסוף האיטרציה ה-$n+1$. וזאת בסתירה להנחת השלילה.

עד פה הוכחת ש[A[v מכיל ערך קטן שווה לערך מסלול קל ביותר מ-r ל-v

נניח בשלילה כי סוף $A[v]$ אינו משקל של מסלול מ-r ל-v.
אז הקשת (u,v), תקיים כי $A[u]$ יכול משקל מסלול $A[v]=A[u]+(u,v)$ השווה למשקל המסלול מ-r ל-v, בסתירה להנחת השלילה.

הפסקה הזו קצת בלבלה אותי,
באיזה הקשר היא מוכחת?
כמו"כ בטענת האינדוקציה יש אמ"מ
ונדמה שבפסקה זו מוכיחים רק כיוון אחד
לא ברור מי זו הקשת (u,v) שמוזכרת כאן.

לכן, $A[v]$ יכיל את המרחק של המסלול המינימלי מ-r ל-v.

הקביעה כאן לא נשענת על שום דבר.
צריך להוכיח זאת.
למשל: הראנו ש[A[v מכיל ערך קטן שווה לערך מסלול קל ביותר מ-r ל-v
כמו"כ מתקיים ש-[A[v מכיל ערך גדול שווה לערך מסלול קל ביותר מ-r ל-v כי בפסקה הקודמת הראנו שהוא מכיל ערך של מסלול כלשהו. -5

סעיף ב':

לפי סעיף א': באיטרציה ה-n לא יהיו עדכונים והריצה תסתיים כי המרחקים מחושבים לאחר $n-1$ איטרציות. כלומר $B(n)=n$

למשל, גרף שבאלגוריתם יעדכן באיטרציה ה-k את הערך $A[n-k]$ והאיטרציה n לא תעדכן דבר:

$$V=(v_1, v_2, \dots, v_n) \qquad E=\{(v_i, v_{i+1}) \mid i \in [1,n]\}$$

סעיף ג':

דוגמה לסדרת גרפים כזו הם גרפים שבהם כל $n-1$ הקודקודים מחוברים רק ל-r.
במקרה הזה רק הסריקה הראשונה תעדכן כבר את כל הערכים והשנייה כבר לא תעדכן דבר.

Scanned by CamScanner