קורס 20417 סמסטר 2017ב

אודי ליוטשי 036745438

ממ"ן 14

1) ראשית נפתור את הבעיה באמצעות אלגוריתם רקורסיבי רגיל, הרעיון יהיה לצאת מכל איבר מהשכבה הימנית לכיוון השכבה השמאלית (בהתאם לאילוצים) ולבחור את המסלול המינימלי מכל המסלולים האפשריים.

<u>האלגוריתם הרקורסיבי ה-'רגיל'</u>

```
cost (c[][], I ,j)
        if i=0
                return 0
        if j< 1 or j>n
                return ∞
        return c[i][j] + min (cost(c[][], i-1, j-1), cost(c[][], i-1, j), cost(c[][], i-1, j+1))

main
        min = ∞
        for j<-1 to n do
                if(cost(c[][], n, j) < min)
                        min <- cost(c[][],n,j)
        return min
```

<u>הוכחת נכונות</u>

1. תנאי עצירה עוצר את האלגוריתם: בכל צעד הערך של i יורד ב-1 לכן מובטח לנו שהוא יגיע בשלב כלשהו להיות שווה 0 ואז הרקורסיה תעצור.
2. תנאי העצירה מחזיר תשובה נכונה: כש-i מקבל את הערך 0, יצאנו מהשריג ולכן נרצה לקבל ערך 0 מאחר ולא ביצענו צעד. כמו כן כש-j חורג מגבולות השריג נקבל אינסוף שזה למעשה מבטל את המסלול (כי הוא בהכרח לא מינימלי) כנדרש
3. אם הקריאות הרקורסיביות מחזירות תשובה נכונה אז האלגוריתם כולו מחזיר תשובה נכונה: לכל נקודה בשכבה הימנית נקבל את המסלול הכולל אותה עצמה + המסלול הקצר ביותר מהשכבה הסמוכה אליה משמאל לשכבה השמאלית ביותר. ברור אם כך שבחירת הנקודה שבה הסכום הנ"ל מינימלי יתן את עלות המסלול המינימלי כולו.

לאחר שהוכחנו שהאלגוריתם נכון נבנה על בסיס האלגוריתם הנ"ל אלגוריתם תכנון דינאמי יעיל יותר:

לצורך כך נשתמש במטריצה a[][] בגודל NXN, נמלא את a באמצעות האלגוריתם הבא:

```
for i<-1 to n do
        for j<-1 to n do
                if(i = 1)
                        a[i][j] <- c(i,j)
                else if(j=1)
                        a[i][j] = c(i,j) + min(a[i-1][j], a[i-1][j+1])
                else if(j=n)
                        a[i][j] = c(i,j) + min(a[i-1][j-1]. a[i-1][j]
```

```
else
        a[i][j] = c(i,j) + min(a[i-1][j-1]. a[i-1][j], a[i-1][j+1])
```

האלגוריתם מחשב את ערכי כל שכבה בהתבסס כל ערכי השכבה שלשמאלה אשר חושבו כבר.

המחיר המינימלי יהיה הערך המינימלי של a[n][j] כש-j רץ מ-1 עד n.

<u>ניתוח זמן ריצה</u>

לולאה מקוננת של n איטרציות אשר מבצעת Θ(1) פעולות מכאן שזמן הריצה של האלגוריתם יהיה Θ (n^2) כנדרש.

<u>שחזור המסלול המינימלי מתוך a</u>

נמצא את הערך המינימלי של a[n][j] כש-j רץ מ-1 עד n, נשמור את j למשתנה זמני tmp

כעת נריץ את האלגוריתם הבא

```
j <-tmp
i <- n
print (i,j)
while(i>1) do
        if(j=1)
                i,j <- (i,j | a[i][j] = min(a[i-1][j], a[i-1][j+1]))
        else if(j=n)
                i,j <- (i,j | a[i][j] = min(a[i-1][j-1]. a[i-1][j]))
        else
                i,j <- (i,j | a[i][j] = min(a[i-1][j-1]. a[i-1][j], a[i-1][j+1]))
        print (i,j)
        i <- i-1
```

בסיום הריצה נקבל את המסלול המינימלי מהשכבה השמאלית לימנית בקריאת הנקודות בסדר הפוך להדפסתן.

זמין הריצה במקרה הזה הוא Θ (n) מכיוון שמתבצע מספר קבוע של פעולות עבור כל ערך של i כש-i רץ מ-n עד 2.

(2

<u>האלגוריתם</u>

תחילה נמיין את n הקופסאות לפי שטח הבסיס (מכפלת אורך ברוחב) בסדר יורד.

כעת נגדיר את הפונקציה $\mathrm{HT}(i)$ להיות המגדל הגבוה ביותר שניתן לבנות כך שהתיבה i נמצאת

בפסגתו . נשים לב שמגדל זה יכול להכיל אך ורק תיבות בעלות שטח בסיס גדול מ i– לפי הנדרש בשאלה.

מכאן שהפונקציה תקיים את הנוסחה הרקורסיבית הבאה :

$\mathrm{HT}(i)$ = max $\{\{\mathrm{HT}(j) \mid j<i \text{ and } wj<wi \text{ and } lj<li\} + hi\}$

נחשב ביטוי זה לכל $1 \leq i \leq n$ ונאחסן את התוצאות במערך(בגודל n) נוסיף שדה נוסף לאיבר במערך שבו נאחסן את אינדקס התיבה הקודמת במגדל ש-i בראשו (ערכים אלה יעודכנו במהלך הריצה עד שיתקבל המגדל המקסימלי לאותו i) כשנגיע למצב שנשארה רק תיבה אחת נשים בשדה התיבה הבאה null.

<u>שחזור המגדל מהמערך שהתקבל</u>

נעבור על המערך ונמצא את הערך המקסימלי, זהו גובה המגדל המקסימלי שהתקבל.

נדפיס את אינדקס התיבה שבה מצאנו את הערך המקסימלי ונקפוץ לערך התיבה הקודמת של אותו איבר במערך, נחזור על הפעולה עד שבשדה התיבה הקודמת נקבל null.

סדר התיבות במגדל המקסימלי יהיה בדיוק הפוך לסדר שהודפס .

<u>הוכחת נכונות</u>

הפונקציה $\mathrm{HT}(i)$ תחפש את השילוב שיצור את המגדל המקסימלי עבור תיבה i בפסגה, הפונקציה שוקלת עבור כל i רק תיבות עם שטח פנים גדול יותר ולכן לא יתכן שהאלגוריתם ישבץ תחתיה תיבה קטנה יותר בסתירה לנדרש.

הדרישה למקסימום דואגת שנקבל את המגדל הגבוה ביותר האפשרי.

שילוב שתי הדרישות מבטיח לנו מגדל יציב בגובה מקסימלי עבור על תיבה שנציב בפסגה.

מבין כל המגדלים הללו אנו בוחרים את זה הגבוה ביותר מכאן שנקבל את המגדל המקסימלי היציב ביותר שניתן לבנות מהתיבות הנתונות כנדרש.

<u>סיבוכיות</u>

1. מיון $\Theta(n \cdot \log n)$ - לפי שיטות מיון נפוצות
2. חישוב מקסימום עבור תיבה - חישוב מקסימום לתיבה יהיה במקרה הגרוע תלוי בכל התיבות בקבוצה ולכן יקח O(n), חישוב זה מתבצע עבור כל תיבה ומכאן נקבל Θ(n^2)
3. סריקת המערך למציאת מקסימום – Θ(n)

סה"כ זמן ריצה: $\Theta(n\text{^}2)$

(3

א.

$q(x) = x_{j+1} - x$

$r(x) = x_i - x$

$s(x) = x_{j+1} - x_i$

ב.

<u>האלגוריתם</u>

נשתמש במטריצה [][]a בגודל NXN לשמירת תוצאות החישוב.

for i<-1 to n do

a[i][i] = y_i

for i<-2 to n do

for j<-1 to n-i+1

$$a[j][j+i-1] = \frac{(x_{j+i-1} - x)a[j][j+i-2] - (x_j - x)a[j+1][j+i-1]}{x_{j+i-1} - x_j}$$

return a[1][n]

נכונות האלגוריתם מתבססת על נכונות האינטרפולציה והנוסחה מסעיף א שקיבלנו כנתון.

<u>סיבוכיות</u>

אתחול מטריצה - O(n)

חישוב ערכי המטריצה הנותרים ע"י לולאה מקוננת בגודל n, פעולת החישוב נעשית בזמן קבוע מאחר והיא מתבססת על ערכים שכבר חושבו – O(n^2)

סה"כ O(n^2)

ג.

i=j=1	i=j=2	i=j=3	i=j=4	i=j=5
-2,46	-1,2	0,0	1,10	2,98

i\j	1	2	3	4	5
1	46	-44x -42	$21x^2 + 19x$	$-5x^3 + 6x^2 + 9x$	$x + 2x^2 + 3x^3 + 4x^4$
2		2	-2x	$6x^2 + 4x$	$11x^3 + 6x^2 - 7x$
3			0	10x	$39x^2 - 29x$
4				10	88x-78
5					98

i=2;j=1..4

j=1 :

$$a[1][2] = \frac{(x2 - x)a[1][1] - (x1 - x)a[2][2]}{x2 - x1} = \frac{(-1 - x) \cdot 46 - (-2 - x) \cdot 2}{-1 - (-2)} = -44x - 42$$

j=2 :

$$a[2][3] = \frac{(x3 - x)a[2][2] - (x2 - x)a[3][3]}{x3 - x2} = \frac{(0 - x) \cdot 2 - (-1 - x) \cdot 0}{0 - (-1)} = -2x$$

j=3 :

$$a[3][4] = \frac{(x4 - x)\, a[3][3] - (x3 - x)\, a[4][4]}{x4 - x3} = \frac{(1 - x) \cdot 0 - (0 - x) \cdot 10}{1 - 0} = 10x$$

j=4 :

$$a[4][5] = \frac{(x5 - x)\, a[4][4] - (x4 - x)\, a[5][5]}{x5 - x4} = \frac{(2 - x) \cdot 10 - (1 - x) \cdot 98}{2 - 1} = 88x - 78$$

i=3;j=1..3

j=1:

$$a[1][3] = \frac{(x3-x)\, a[1][2]-(x1-x)\, a[2][3]}{x3-x1} = \frac{(0-x)\cdot(-44x-42)-(-2-x)\cdot(-2x)}{0-(-2)} = 21x^2 + 19x$$

j=2:

$$a[2][4] = \frac{(x4-x)a[2][3]-(x2-x)\, a[3][4]}{x4-x2} = \frac{(1-x)\cdot(-2x)-(-1-x)\cdot(10x)}{1-(-1)} = 6x^2 + 4x$$

j=3:

$$a[3][5] = \frac{(x5-x)a[3][4]-(x3-x)\ a[4][5]}{x5-x3} = \frac{(2-x)\cdot(10x)-(0-x)\cdot(88x-78)}{2-0} = 39x^2 - 29x$$

i=4;j=1..2

j=1:

$$a[1][4] = \frac{(x4-x)a[1][3]-(x1-x)a[2][4]}{x4-x1} = \frac{(1-x)\cdot(21x^2+19x)-(-2-x)\cdot(6x^2+4x)}{1-(-2)} = -5x^3+6x^2+9x$$

j=2:

$$a[2][5] = \frac{(x5-x)a[2][4]-(x2-x)a[3][5]}{x5-x2} = \frac{(2-x)\cdot(6x^2+4x)-(-1-x)\cdot(39x^2-29x)}{2-(-1)} = 11x^3+6x^2-7x$$

i=5;j=1

$$a[1][5] = \frac{(x5-x)a[1][4]-(x1-x)\ a[2][5]}{x5-x1} = \frac{(2-x)\cdot(-5x^3+6x^2+9x)-(-2-x)\cdot(11x^3+6x^2-7x)}{2-(-2)}$$
$$= x+2x^2+3x^3+4x^4$$

return [1][5] כנדרש

4)

א. האלגוריתם מחשב מרחקים מינימליים מהקודקוד r לשאר הקודקודים בגרף G.

נוכיח באינדוקציה שבסוף האיטרציה ה - iית של הלולאה החיצונית A[v] יכיל ערך של אינסוף רק אם אין מסלול באורך i לפחות 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.

<u>בסיס האינדוקציה</u>: 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 שיסתור את הערך ∞, לכן האתחול של A תואם את הטענה .

<u>צעד האינדוקציה</u>: נניח נכונות עבור k ונוכיח נכונות עבור k+1:

לפי הנחת האינדוקציה לכל v∈V , A[v] מכיל משקל של מסלול כלשהו מ-r ל-v ומשקל זה קטן או שווה

למשקל כל המסלולים מ-r ל-v שאורכם לכל היותר k .

נניח בשלילה שהערך A[v] יסתור את טענת האינדוקציה בסוף האיטרציה ה- k+1, כלומר יש מסלול באורך k+1 מ-r ל-v שמשקלו קטן מ-A[v].

אך במקרה הזה תהי (u,v) הקשת האחרונה במסלול כזה, אל u ישנו המסלול באורך k ללא הקשת (u,v) ולפי הנחת האינדוקציה A[u] קטן או שווה למשקל המסלול הזה, לכן A[v] יהיה קטן או שווה ל(A[u] + c((u,v בסוף האיטרציה ה- k+1 בסתירה להנחה בשלילה . מכאן שהנחת השלילה שלנו שגויה וA[v] מכיל ערך נכון בסוף האיטרציה ה- k+1.

נניח בשלילה שערכו של A[v] ≠∞ אינו משקלו של מסלול מ-r ל-v. לפי הנחת האינדוקציה A[v] עודכן אם כך באיטרציה הנוכחית. נניח לה"כ ש-v הוא הקודקוד הראשון עבורו ישנה הפרה לאחר העדכון.

אזי תהי קשת e=(u,v) אשר בסריקה שלה עודכן A[v]. לפי ההנחות לעיל A[u] הכיל משקל מסלול כלשהו מ-r ל-u ו- A[v] מכיל את נערך של A[u]+c(e) ולכן A[v] מכיל את משקל המסלול מ-r ל-v בסתירה להנחת השלילה.

אנו יודעים שקיים בהכרח מסלול מינימלי מ-r ל-v שאורכו n-1 לכל היותר, לכן A[v] יכיל בסיום n-1 איטרציות ערך קטן או שווה למשקל המסלול המינימלי, מצד שני, כל ערך סופי של A[v] שקול למשקל של מסלול כלשהו מ-r ל-v ולכן A[v] יכיל את המרחק של המסלול המינימלי מ-r ל-v.

ב. לפי סעיף א המרחקים ב-A מחושבים לאחר n-1 איטרציות ולכן באיטרציה ה-nית לא יהיו עדכונים והריצה תסתיים, כלומר, B(n) = n.

דוגמא לסדרת גרפים כזו היא גרפים שהם מסלול שבהם משקל כל הצלעות שווה:

$V = (v_1, v_2, \ldots, v_{n-1}, v_{n=r})$

$E = \{(v_i, v_{i+1}) \mid 1 \leq i < n\}$

בגרפים מסוג אלה יעדכן האלגוריתם באיטרציה ה-iית את הערך A[n-i] והאיטרציה ה-n לא תעדכן.

ג. דוגמא לסדרת גרפים כזו הם גרפים שבהם כל n-1 הקודקודים מחוברים ל-r ולו בלבד. במקרה כזה האיטרציה הראשונה תעדכן את כל ערכי A והאיטרציה השניה לא תעדכן דבר משום שהערך שחושב באיטרציה הקודמת הוא אכן המסלול המינימלי.