יעל ויצמן 312232655

25/40

<u>ממן 14:</u>

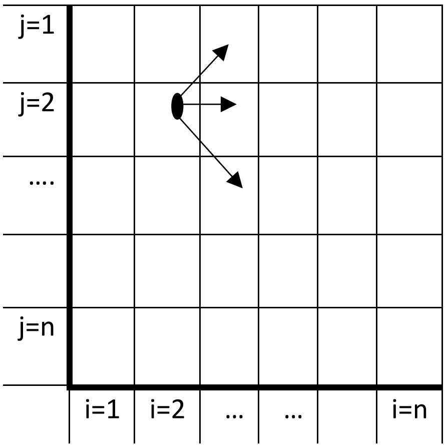
<u>שאלה 1</u>

<u>הגדרות&נתונים</u>

1. S יהיה מוגדר להיות השריג הריבועי (nxn) הנתון.
2. Cij יהיה מוגדר להיות המחיר עבור כל תא I,j בשריג S.
3. הצעדים המותרים: ימינה , ימינה למעלה, ימינה למטה.(כנתון)
4. M – תהיה מוגדרת להיות מטריצה בגודל nxn שבה נכניס את ערכי החישוב שלנו בזמן הריצה.
5. SavePath[] - יהיה מוגדר להיות מערך שבכל תא ותא שלו ישמרו זוג אינדקסים.

<u>אינטואיציה:</u>

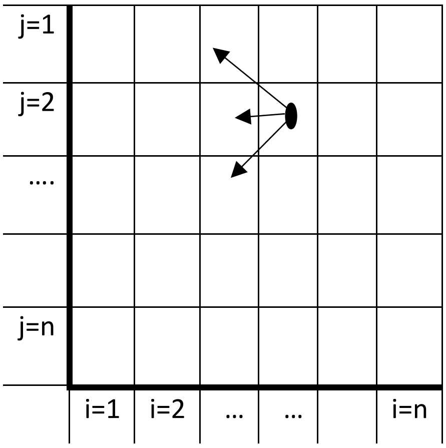
השריג S הנתון מוגדר להיות כך:



- כיוון מסלול - אנו נדרשים בשאלה למצוא את המסלול מהשכבה הימנית ביותר לשמאלית ביותר,
- אנו נדרשים עבור כל ריבוע במטריצה לבדוק 3 מקרים נתונים.

אם נפעיל אלגוריתם שיבדוק את הנדרש בשאלה כאשר הוא יבדוק את הנתונים מריבוע מסוים לשלושה ריבועים מימינו (כמו שמאויר בשירטוט כאן), נקבל אלגוריתם בזבזני, שהרי חישובים שכבר חישב עבור ריבוע מסוים כמה פעמים נוספות.

- ולכן:



- נחשב את האלגוריתם בדרך זו, ובכך מובטח לנו שבכל שלב נשמור את התוצאות שכבר חישבנו, ובכך נמנע מכפילויות.

נסיק מכך שנצטרך לחשוב על נוסחת נסיגה!

נוסחת הנסיגה:

- אם i או j שהתקבלו הן עוקפים את האינדקסים של קצוות המטריצה, אז: OPT[I,j]=∞.
- אחרת חשב: OPT[I,J]=Cij + MIN[OPT[i-1,j-1] + OPT[i-1, j] + OPT[i-1, j+1]]

רעיון האלגוריתם:

.1 נמצא את המסלול המינימלי ממטריצת השריג הנתון ע' נוסחת הנסיגה שהגדרנו קודם.
- נשמור כל ערך של סכום מינימלי שחישבנו במטריצת M .
- בסופו של תהליך אחד מבין האיברים שבעמודה n האחרונה שבמטריצת M , יהיה הסכום המינימלי של מסלול מבין כל המסלולים.

.2 כעת נצטרך לשחזר את המסלול שעשינו -

2.1 נמצא את האינדקס שבו נמצא המסלול המינימלי – ונשמור את (ערכו) + (הערך של האינדקס n) במערך שהגדרנו בהתחלה []savePath (נשמור במקום ה n).

2.2 נחשב את ערך המסלול מהסוף להתחלה בעזרת נוסחת הנסיגה שהגדרנו, כאשר כל זוג אינדקסים שנמצא נשמור אותם במערך []savePath (כמובן עם אינדקסים רצים מn-1 להתחלה)

האלגוריתם:

(מספור השורות הוא בחפיפה למספור השורות שברעיון האלגוריתם)

1. עבור i=1 עד n
   - אם i=1 אז מ j=1 עד n בצע: M[I, j] = S[I, j] .
   - אחרת, חשב את נוסחת הנסיגה שהגדרנו כך ש: M[I, j] = OPT[I, j].
2. עבור 1=<j<=n חשב את [M[n, j]]min_path = MIN.

2.1 עבור j=1 עד n : אם: M[n, j] = min_path אז נשמור את ערכו במערך savePath[ind]

2.2 עבור i=n-1 כל עוד i<0:

נחשב : ind= $\min(M[i,j], M[i,j+1], M[i,j-1]) + c[i,j]$

נשמור במערך []savePath את הערך ind.

נכונות האלגוריתם:

שורה 1 – *חישוב נוסחת הנסיגה- על פי מה שהראיתי בשלב האינטואיציה ניתן לראות שנוסחת הנסיגה נכונה ועדיין מקיימת את התנאים שנדרשו בשאלה + היא תיתן את המינימלי מבניהם כי מותר רק 3 צעדים ואנו בפירוש בוחרים את המינימלי.
*בנוסף נוסחת הנסיגה תמיד תעצור מפני שאנו מקטינים תמיד ב1 את i, הנוסחא עוברת עבור אינדקס i מ 1 עד n ובודקת מי המינימלי בשורה i-1 – ולכן תמיד יהיה לה ערכים מאיברים קודמים. לא מספיק פורמלי
שורה 2 – לאחר שחישבנו את המסלולים נותר למצוא מהעמודה האחרונה בM את הסכום המינימלי ואז לשחזר את המסלול על ידי בדיקה של פעולות קבועות.

לא הבנתי כיצד את משחזרת את המסלול באלג'. אני לא רואה שחזור מסלול אלא רק שחזור ערך המסלול

חישוב זמן ריצה:

- על מנת למצוא את המסלול המינימלי ממטריצת השריג ע' נוסחת הנסיגה שהגדרנו – נעבור על כל תאי המטריצה ועבור כל תא נבצע כמה פעולות שהן שוות ערך לקבוע ולכן זמן הריצה יהיה כגודל המטריצה $O(n^2)$.
- על מנת למצוא את הסכום והאינדקס שבו נמצא ערך המסלול המינימלי נצטרך לעבור על כל העמודה שבמקום ה n - ולכן זמן הריצה יהיה – O(n).
- חישוב שיחזור המסלול ידרוש מאיתנו בדיקה על ידי ביצוע פעולות קבועות ולכן יקח O(n).

קיבלנו שזמן הריצה הוא $O(n^2)$.

שאלה 2

הגדרות&נתונים:

1. נתונות n תיבות מלבניות במערך נתון BOX+ מידע על גובה אורך ורוחב.
2. אנו נדרשים לבנות את גובה המגדל כך שיקיים : שיהיה המקסימלי ביותר + יציב.
3. OPT(i) - נגדיר מערך OPT בגודל n, שבו בכל OPT[i] יהיה בכל זמן נתון של ריצת האלגוריתם את הגובה המקסימלי שיכול להיות לתיבה באינדקס i , כאשר התיבה הזאת היא בראש המגדל.
4. Results- נגדיר מערך בשם results בגודל n, שבו כל results[i] יאמר לנו:
   * i – זה האינדקס של התיבה באינדקס i של המערך המקור,
   * result[i] - יהיה הערך של התיבה שנמצאת מתחת ל i – אם התיבה i היא בתחתית המגדל אז results[i]=0.

רעיון האלגוריתם:

1. נמיין את התיבות על ידי חישוב עבור כל תיבה i : נבדוק את ערך ה Base Area= BA=L(i)*W(i) , וכך נמיין מהגדול לקטן כך שבאינדקס 0 יהיה את הBA הגדול ביותר שמצאנו, ובאינדקס n יהיה את ה BA הקטן ביותר שמצאנו. שלב זה יקדם אותנו לפתור את הבעיה בקלות יותר עבור הדרישה שעבור כל תיבה i מימדי התיבה שמתחתיו יהיו גדולים יותר.
2. ניצור את שני המערכים: -* OPT – לתוכו נכניס בשלב ראשוני עבור כל אינדקס i את הגובה של תיבה באינדקס i.
   * results – לתוכו נכניס עבור כל i , results[i]=i.
3. כעת, על מנת לחשב את OPT(i) נחשוב קודם על הרעיון, אמרנו ש OPT(i) הוא ראש המגדל היציב, ולכן כל מי שמתחתיו חייב להיות גדול ממנו בממדים. ולכן יכולים להיות שני מקרים:
   * או שאנו במצב שהמגדל הוא התיבה הבודדת i לבד – ואז הגובה המקסימלי הוא h(i).
   * או שהתיבה i היא חלק מגדל יציב ואז נרצה את h(i) + הגובה המקסימלי שהושג עד כה עבור OPT(j) כאשר j קטן יותר מ i (1≤j<i).
4. מכאן הגענו להגדרת הנוסחה:

$$\boxed{OPT(i)=Max\{\ h(i),\ h(i)+OPT(J)\ \},\ \text{כאשר}\ 1\le j<i}$$

5. לאחר מכן נחפש את האיבר באינדקס i , המקסימלי מהמערך max – הוא יביע לנו שני דברים:
   *נדע שהתיבה i הנתונה היא בראש המגדל היציב.
   * הערך שרשום ב max[i] הוא הגובה המקסימלי למגדל היציב שהאלגוריתם מצא.
6. נוציא מהמערך results את האינדקסים של התיבות שבונות לנו את המגדל – האינדקסים יובאו כך שהאינדקס שהודפס ראשון יביע את התיבה שבראש המגדל ואנידקס שיודפס לאחריו יביא את זה שמתחתיו..... עד לאינדקס שיודפס אחרון שיתן לנו את התיבה שתהיה בבסיס המגדל.

האלגוריתם:

1.נמיין את כל התיבות לפי הערךBA שלהן, כך שבזמן המיון אלגוריתם המיון יבדוק עבור כל תיבה את BOX[i].Length*BOX[i].Wide ולפי זה ימיין את התיבות מהBA המקסימלי לקטן .

2. עבור i=1 עד n :
   * OPT[i] = BOX[i].Height
   * results[i]=i

3. עבור i=1 עד n :

עבור j=0 כל עוד j<i :

*אם (BOX[i].Length<BOX[j].Length && BOX[i].Wide<BOX[j].Wide)

אז: OPT[i] = Max{ h(i), h(i) + OPT(j) } // ביצוע נוסחת הנסיגה

* אם (h(i)+OPT(j) > h(i)) // ישי במערך זה לפי מה שהגדרנו בהדגש שאם i הוא בבסיס אז results[i]=0

אז results[i] = j

4. עבור i=1 עד n :

Length_max = MAX{ OPT (i), 1≤i≤1 } // חפש את האיבר המקסימלי

5.עבור i=Length_max עד 0
אם results[i]≥0
Final_Results<- i
i= results[i]
אם results[i]=0 צא מהלולאה

6. החזר את Final_Results

<u>נכונות האלגוריתם:</u>
<u>נכונות עבור שורה 3</u> – אנו מחשבים את ערכי OPT, על בסיס התנאי
*שאם מדובר במגדל בפני עצמו אם הוא עומד לבדו וכמובן שיעמוד בדרישות הממדים.
* אם מדובר במקרה שתיבה i תהיה מעל תיבה j אז:
- קודם כל נבדוק אם היא עומדת בדרישות BOX[i].Length<BOX[j].Length && BOX[i].Wide<BOX[j].Wide
ורק אם היא אכן גדולה בממדיה מתיבה i , אז תיבה i תהיה מעליה.
* האלגוריתם מקיים שתמיד i<j ולכן לא ניתקל בשלב של OPT שעדיין לא התחלנו אותו.
* בכל איטרציה אנו שמים במערך OPT את הגובה המתאים המיקום המתאים.

<u>נכונות עבור שורה 4</u> – אנו מחפשים את האיבר המקסימלי עי מעבר איטרטיבי על איברי המערך.
<u>נכונות עבור שורה 5</u> – מהמיקום של האיבר המקסימלי אנו שומרים את האינדקסים של התיבות ששמרנו במערך results.
עד שנגיע לאינדקס שהוא בסיס המבנה, (נשמור את ערכו ורק אז נצא מהלולאה).

<u>חישוב זמן ריצה:</u>
מיון מערך התיבות שהתקבל יהיה – O(nlogn) .
הכנסת הערכים ההתחלתיים ל OPT, Results – O(n)
חישוב OPT עבור כל הערכים יקח – במקרה הגרוע נקרא n פעמים ועבור כל פעם נבדוק n-1 פעמים **ולכן-** $O(n^2)$.
מציאת המקסימום מהמערךOPT – מעבר איטרטיבי על n האיברים – O(n)
ולכן סך הכל זמן הריצה יהיה: T(n) = $O(n^2)$

שאלה 3:

<u>הגדרות&נתונים:</u>

1. אינטרפולציה- התאמת פונקציה המתאימה ביותר להתנהגות של נקודות דגימה שנלקחו ובכך לדעת מה קורה בכל נקודת זמן לאורך הניסוי.
2. נתון הפולינום :

$$p(x) = a_0 + a_1x + a_2x^2 + \cdots \dots\dots + a_{i-1}x^{i-1} + \underbrace{a_ix^i + a_{i+1}x^{i+1} + \dots\dots + a_{j-1}x^{j-1} + a_jx^j} + a_{j+1}x^{j+1} + \dots a_{n-1}x^{n-1}$$

$\downarrow$

פולינום האינטרפולציה של הנקודות $(x_i, y_i) \dots\dots\dots\dots (x_j, y_j)$ עבור כל i≤j .

3. מישום שנתון ששלושת הפולינומים הם מדרגה 1 או 2 נגדירים כך:

* **A** = q(x)=ax+b

* **B** = r(x) = cx+d

* **C** = s(x)= ex+f

<u>אינטואיציה:</u>

נחשוב על התהליך מהשלב ההתחלתי עד להגעה לפיתרון:

נתונה לנו הנוסחא: $p_{i,j+1} = \frac{q(x)*p_{i,j}(x) - r(x)*p_{i+1,j+1}(x)}{s(x)}$

לשם הנוחות והבהירות נשתמש בסימונים שהגדרנו עבור 3 הפולינומים הפשוטים ונקבל את הנוסחא:

$$p_{i,j+1} = \frac{\boldsymbol{A}*p_{i,j}(x) - \boldsymbol{B}*p_{i+1,j+1}(x)}{\boldsymbol{C}}$$

$\downarrow$

ניגש כעת לראות מה קורה בתוך הפולינום הנתון:

* מיהו $p_{i,j}$? $p_{i,j}(x_i) = y_i$ (על פי הגדרת אינטרפולציה שעבור אותם נקודות שיקבל, ישיג את אותה התוצאה) $\leftarrow$ $p_{i,j+1}(x_i) = p_{i,j}(x_i)$ && $p_{i,j+1}(x_{j+1}) = p_{i+1,j+1}(x_{j+1})$

$\downarrow$

ולכן נוכל לכתוב את הנוסחא הנתונה כך:

$$p_{i,j+1} = \frac{\boldsymbol{A}*p_{i,j+1}(x_i) - \boldsymbol{B}*p_{i,j+1}(x_{j+1})}{\boldsymbol{C}}$$

$\downarrow$

כעת על מנת לקבל שיווין , ברור לכל ספק שנצטרך:

* שהפולינום B יהיה שווה ל-0 כאשר ניתן לו את הערך x_i, B=r(x) ולכן $r(x_i) = x_i - x_i = 0$ ולכן: $r(x) = x - x_i$

* הפולינום A יהיה שווה ל0 אם ניתן לו את הערך x_{j+1} , A=q(x) ולכן $q(x_{j+1}) = x_{j+1} - x_{j+1} = 0$ ולכן: $q(x) = x - x_{j+1}$
נקבל שעבור A = 1 תוכל להתקיים הנוסחא הנתונה.

* הפולינום C יהיה שווה ל-1 כאשר ניתן לו את הערך x_i, C=s(x) ולכן $s(x_i) = 1$

נתחיל בשלבים למציאת C, כמו שראינו קודם עבור x_i נקבל מהנוסחא $p_{i,j+1} = \frac{\boldsymbol{A}*p_{i,j+1}(x_i) - \boldsymbol{B}*p_{i,j+1}(x_{j+1})}{\boldsymbol{C}}$

$p_{i,j+1}(x_i) = \frac{\boldsymbol{A}*p_{i,j+1}(x_i)}{\boldsymbol{C}}$ (כי B שווה ל 0 כאשר x_i ניתן), ולכן על מנת שתתקיים הנוסחא

שבשאלה נצטרך: $1 = \frac{\boldsymbol{A}}{\boldsymbol{C}}$, $1 = \frac{\boldsymbol{q(x_i)}}{\boldsymbol{s(x_i)}}$.

אותו הדבר עבור $p_{i,j+1}(x_{j+1})$ יתקיים $\frac{r(x_{j+1})}{s(x_{j+1})} = -1$ ולכן נקבל:

$$\frac{r(x_{j+1})}{s(x_{j+1})} + \frac{q(x_i)}{s(x_i)} = 0 \rightarrow -ex_i^2 + 2ex_ix_{j+1} - ex_{j+1}^2 = -e(x_i + x_{j+1})^2 = 0$$

נגיע לפיתרון s(x)= $x_i - x_{j+1}$ דרך פסילת שני אופציות הפיתרון למשוואה שראינו ע"י שנבחר את הפיתרון **e=0** ,
ונציב $\frac{q(x_i)}{s(x_i)} = 1$.

מכאן , שקיבלנו והוכחנו שהפולינומים המקיימים את הנוסחא הנתונה:

$$\boldsymbol{r(x) = x_i - x}$$
$$\boldsymbol{q(x) = x - x_{j+1}}$$
$$\boldsymbol{s(x) = x_i - x_{j+1}}$$

נוסחת הנסיגה:

$$p_{i,j+1} = \frac{(x_{j+1}-x)p_{i,j}(x)-(x_i-x)p_{i+1,j+1}(x)}{x_{j+1}-x_i}$$

ב. מסעיף א' ראינו והוכחנו את הנסיגה עבור בעיית האינטרפולציה :

<u>רעיון האלגוריתם:</u>

1. נגדיר מערך OPT[I, j] בגודל nxn שיהיה פיתרון האינטרפולציה עבור הנקודות שהוגדרו בשאלה: $(x_i, y_i) \dots\dots\dots\dots (x_j, y_j)$ עבור כל i≤j.
2. כפי שאמרנו קודם, (על פי הגדרת אינטרפולציה שעבור אותם נקודות שיקבל, ישיג את אותה התוצאה) ← $p_{i,j+1}(x_{j+1}) = p_{i+1,j+1}(x_{j+1})$ && $p_{i,j+1}(x_i) = p_{i,j}(x_i)$

   ולכן נאתחל את המטריצה בכל המקומות של OPT[I,i] להיות שווה ל y_i.
3. עבור שאר ערכי j<i נשתמש בנוסחת הנסיגה שהגדרנו בסעיף הקודם.
4. נחזיר את OPT[1,n] שהוא הפיתרון של כל פולינום האינטרפולציה

<u>האלגוריתם:</u>

1. עבור i=0 עד i=n
   OPT[I,i]= y_i
2. עבור k=2 עד n
   *i=1
   * עבור j=k עד n
   בצע את נוסחת הנסיגה: $OPT[_{i,j+1}] = \frac{(x_{j+1}-x)p_{i,j}(x)-(x_i-x)p_{i+1,j+1}(x)}{x_{j+1}-x_i}$
3. החזר את OPT[1,n]

<u>נכונות האלגוריתם:</u>

1. <u>נכונות שורה 1</u> באלגוריתם נובעת ממה שהסברנו בסעיף א: מיהו $p_{i,j}$? $p_{i,j}(x_i) = y_i$ (על פי הגדרת אינטרפולציה שעבור אותם נקודות שיקבל, ישיג את אותה התוצאה) ←
   $p_{i,j+1}(x_{j+1}) = p_{i+1,j+1}(x_{j+1})$ && $p_{i,j+1}(x_i) = p_{i,j}(x_i)$
2. <u>נכונות שורה 2</u> נובעת ממה שהוכנו בסעיף א' על נכונות נוסחת הנסיגה.
3. <u>נכונות שורה 3</u> – זה הפיתרון המלא של האינטרפולציה.

<u>ניתוח זמן ריצה:</u>

שורה 1 תעבור איטרטיבית על כל האלכסון הראשי של המטריצה – ולכן O(n).
שורה 2 – מילוי המטריצה בעזרת נוסחת הנסיגה , נעבור לפחות פעם אחת על כל איברי המטריצה ועבור כל תא במטריצה נבצע זמן קבוע של פעולות ולכן - O(n^2).
שורה 3- החזרת ערך פולינום האינטרפולציה יקח לנו זמן – O(1).

שאלה 4

עקב העומס בחיי לצערי לא הספקתי לעשות שאלה זו.....!