אלגוריתמים – ממ"ן 14

שם : איתי אירמאי

תז : 204078224

תאריך הגשה : 18/01/2020

מייל : I.tai307@gmail.com

תשובה טובה. כל הכבוד.
40\40

.1

נגדיר Opt(i,j) = מחיר מסלול מינימאלי לאיבר בשורה I ועמודה j מתוך איבר כלשהו בעמודה 1.

על פי הגדרת הבעיה (נוכיח בהמשך) :

① $opt(i,1) = c(i,1)$

② $opt(i,j) = \min(opt(i-1,j-1),opt(i,j-1),opt(i+1,j-1)) + c(i,j) \quad | 1 < j \le n$

③ $min\_path = \min(\{opt(i,n) | 1 \le I \le n\})$

<u>תיאור האלגוריתם</u> : נשמור במערך דו מימדי בגודל N x N את תוצאות חישוב המסלול המינימאלי לכל קודקוד, ונמלא אותו על פי שתי הנוסחאות הראשונות מעלה ; נחזיר מסלול לפי הנוסחא השלישית. כל עמודה תלוי במספר מוגבל של איברים בעמודה הקודמת בלבד, אז אפשר למלא לפי סדר עולה של העמודות.

<u>פירוט</u> :

קלט : רשימת קודקודים V ומחיריה c (כאשר $V[i,j] = c(i,j)$).

1) אתחל מערך דו מימדי M בגודל |V| עם ערכי אינסוף.

2) לכל $i = 1:n$ בצע : $M[i,1] = c[i,1]$

3) לכל $j = 1:n$ בצע :

3.1) לכל $I = 1:n$ בצע :

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

4) החזר $\min(M[i,n] | I = 1:n)$

<u>סיבוכיות</u> : **$O(n^2)$** פעולות אלמנטריות כנדרש :

- אתחול המערך M הינו N^2.

- חישוב עמודה 1 מורכב מפעולת העתקה בלבד.

- חישוב כל איבר בעמודות אחרות דורש 2 פעולות השוואה, פעולת חיבור (עם איבר שמור) והשמה שכולן אלמנטריות, ויש $(n-1)$ x n איברים כאלה ⇐ O(n^2).

- בחירת האיבר המינימאלי בעמודה אחת באמצעותN השוואות. [אפשר גם לאחד את חישוב המינימום תוך כדי בניית העמודה האחרונה ב-M.]

<u>הוכחה</u> : נתחיל מנוסחאות הנסיגה של opt.

① $opt(i,1) = c(i,1)$

כל מסלול רלוונטי לבעיה חייב להתחיל בקודקוד כלשהו בעמודה השמאלית. המסלול הקל ביותר המתחיל ומסתיים בקודקוד מסוים הינו הקודקוד עצמו (כאשר כל המחירים אי שליליים), ומחיר המסלול כמחיר הקודקוד.

② $opt(i,j) = \min(opt(i-1,j-1), opt(i,j-1), opt(i+1,j-1)) + c(i,j) \qquad | \ 1 < j \le n$

לכל קודקוד בעמודה k יש לכל היותר 3 קשתות נכנסות (או מעברים מותרים בשריג) בעמודה k-1 בלבד ⇐ כל מסלול אליו חייב להכיל אחד מהשכנים הצמודים הללו. נקרא לאחד מבין השלושה

$P = (v_{i1,1}, v_{i2,2}, \ldots, v_{i(k-1),k-1}, v_{i,k})$. נתבונן בתת המסלול P' מ-(i1,1) עד $(i_{k-1},k-1)$. הוא מקיים את הגדרת מסלול opt עבור $(i_{k-1},k-1)$, פרט למזעריות (בהכרח). נניח בשלילה שלא ⇐ קיים מסלול P* שמחירו קטן ממש מ-P'. נבנה את המסלול $Q = (P^*, v_{i,k})$ – מסלול זה מקיים את ההגדרה של opt על (I,k), וכן מחירו הכולל קטן מ-P:

$$c(Q) = c(P^*) + c(v_{i,k}) < c(P') + c(v_{i,k}) = c(P)$$

בסתירה להנחת המזעריות של P ⇐ P' בעל מחיר opt, ובסה״כ לכל השכנים נוסחת הנסיגה מתקיימת.

③ $min\_path = \min(\{opt(i,n) \mid 1 \le I \le n\})$

המסלול הרצוי min_path הוא המזערי המתחיל בקודקוד כלשהו בעמודה 1 ומסתיים בקודקוד אחר בעמודה N על פי הגדרה. בקבוצת {opt(i,n)} מוכלים כל המסלולים המזעריים המתואריים כנ״ל (עם סיום בקודקוד I), ולכן בהכרח $min\_path \in \{opt\}$; בפרט מזערי ביניהם, אחרת מסלול מהרשימה בעל מחיר קטן יותר היה נבחר בתור min_path.

כפי שצוין מעלה, סדר הריצה באלגוריתם מבטיח בניית כל איבר חדש במטמון על בסיס איברים שכבר חושבו, אז הוא בהכרח מגיע למצב עצירה.

צ״ל כי $M[i,j] = opt(i,j)$ לכל האיברים, ואז מזהות נוסחת הנסיגה השלישית לשורה 4 באלגוריתם נסיק כי האלגוריתם מחזיר את מחיר המסלול המזערי הרצוי. זאת נוכיח באינדוקציה על j.

$j=1$: בעמודה הראשונה, $M[i,1] = c[i,1] = opt(i,1)$; בהתאם לשורה 2 באלג׳ ונוסחא ①.

$j = k$: נניח כי עבור כל $I = 1{:}n$ וכן $j = 1{:}k$, מתקיים $M[i,j] = opt(i,j)$.

$j = k+1$: נסתכל על נוסחא ②, שורת קוד 3.1.1, ונציב את $M = opt$ מההנחה מעלה.

$$opt(i,j) = \min(opt(i-1,j-1), opt(i,j-1), opt(i+1,j-1)) + c(i,j)$$

$$\mathbf{M[i,j]} = \min(M[i-1,j-1], M[i,j-1], M[i+1,j-1]) + c[i,j] =$$

$$= \min(opt(i-1,j-1), opt(i,j-1), opt(i+1,j-1)) + c(i,j) = \mathbf{opt(i,j)}$$

$$\Rightarrow \mathbf{return\_value} = \min(M[i,n] \mid I = 1{:}n) = \min(opt[i,n] \mid I = 1{:}n) = \mathbf{min\_path} \ \blacksquare$$

[הערה: אפשר לחלופין לשמור במקום M רק מערך חד מימדי + זיכרון שני איברים אחרונים עקב אי תלות ברוב העמודות בנוסחא, אבל אז המסלול אינו ניתן לשחזור בצורה יעילה.]

.2

[באלגוריתם מטה אולי הנחתי הנחה יותר רפויה מהנתון – אין שתי תיבות בעלות אותו אורך ורוחב ביחד; במקום ייחודיות כל אחד מהערכים בנפרד, אם זו הייתה הכוונה.]

נגדיר $Opt(l,w)$ = גובה המגדל היציב המקסימאלי שניתן לבנות על פני התיבה בעלת אורך = l ורוחב ≥ w.

נתייחס לגובה התיבות בנוטציה $h(l,w)$ – זו התאמה חד חד ערכית כי אין שתי תיבות בעלות אותו אורך ורוחב.

נמיין את התיבות מיון ראשי על פי רוחב ומיון משני על פי אורך. נקרא לכל קבוצת רוחב זהה w_z, כאשר w_1 הינו הרוחב המינימאלי ברשימה ו-w_m המקסימאלי.

על פי הגדרת הבעיה (נוכיח בהמשך), ניתן להגדיר נוסחת נסיגה בצורת מטריצה עם שורות ועמודות בעלות אינדקס l_x, w_z:

① $opt(l_x,w_1) = h(l_x,w_1) \quad | \quad \forall(l_x,w_1) \in B$

$opt(l_x,w_1) = 0 \quad | \quad \forall(l_x,w_1) \notin B \; ; \; \exists\,(l_x,w),(l,w_1) \in B$

② $opt(l_x,w_z) = \max(opt(l_x,w_{z-1}),h(l_x,w_z) + \max(\{opt(l_y,w_{z-1}) \mid y < x\})) \quad | \quad \forall(l_x,w_{z>1}) \in B$

$opt(l_x,w_z) = opt(l_x,w_{z-1}) \quad | \quad \forall(l_x,w_{z>1}) \notin B \; ; \; \exists\,(l_x,w),(l,w_z) \in B$

③ $max\_stack = \max(\{opt(l_x,w_m) \mid \forall l_x\})$

<u>תיאור האלגוריתם</u>: ניצור שתי פונקציות: אחת הממירה $w_z \rightarrow z$, והשנייה $l_x \rightarrow x$. נשמור במערך חד מימדי בגודל n [או כמספר האורכים השונים] את תוצאות חישוב המגדל המקסימאלי לכל גובה לפי קטגוריית רוחב נוכחי, כאשר רק הגבהים הרלוונטיים יעודכנו תוך כדי התקדמות ברוחב בהתאם לשתי הנוסחאות הראשונות מעלה; נחזיר גובה מגדל מקסימאלי לפי הנוסחא השלישית.

<u>פירוט</u>:

קלט: רשימת תיבות $B[l,w,h]$.

1) מיין את B לפי l וצור טבלת גיבוב $fl[l_x] \rightarrow x$, לפי סדר האיברים השונים המופיעים בסדרה הממוינת (כלומר הגובה הראשון יקבל ערך 1, הבא השונה ממנו ערך 2 וכו').

2) מיין את B לפי w וצור טבלת גיבוב $fw[w_z] \rightarrow z$; שמר את המיון המקורי לפי l כמשני.

3) אתחל מערך M בגודל n עם אפסים.

4) הגדר משתנה "רוחב_נוכחי" = 0; תור "עדכון_רוחב" ריק המכיל איברים (x,h).

5) לכל (l,w,h) ב-B הממוין בצע:

5.1) אם w שונה מרוחב_נוכחי:

5.1.1) רוחב_נוכחי =: w

5.1.2) עדכן את M לפי איברי עדכון_רוחב (ורוקן את התור).

5.2) אם $fw[w] = 1$:

5.2.1) הוסף את האיבר הבא לעדכון_רוחב: $(fl[l],h)$

5.3) אחרת:

5.3.1) הוסף את האיבר הבא לעדכון_רוחב:

$(fl[l] , \max(M[fl[l]] , h + \max(\{M[y] \mid y < fl[l]\})))$

6) עדכן את M לפי איברי עדכון_רוחב.

7) החזר את המקסימום של M.

סיבוכיות: $O(n^2)$ פעולות אלמנטריות כנדרש:

- מיון לפי $W + L = O(nlogn)$, יצירת טבלאות גיבוב $O(n)$, כנ"ל אתחול M.

- חישוב קבוצת w1 מורכבת משתי פעולות העתקה (לתור הזמני ומהתור ל-M), בהתאם למספר האיברים בקבוצת הרוחב.

- חישוב כל איבר בקבוצות אחרות דורש בנוסף איתור מקסימום בין 0 עד n-1 איברים, פעולת חיבור, פעולת השוואה; יש $O(n)$ איברים כאלה ⇐ $O(n^2)$. פונקציות ההמרה הנ"ל מונעות זמן פסודו פולינומי שבמטריצה דלילה.

- בחירת האיבר המינימאלי במערך באמצעות $O(n)$ השוואות.

<u>הוכחה</u>: נתחיל מנוסחאות הנסיגה של opt – צריך להראות כי הנוסחא מחזירה ערך נכון לכל תא במטריצה $[l_x, w_z]$. נוכיח אותן באינדוקציה על כל אורכי תיבות ברוחב i.

① $opt(l_x,w_1) = h(l_x,w_1) \quad | \forall(l_x,w_1) \in B$

$opt(l_x,w_1) = 0 \quad | \forall(l_x,w_1) \notin B$

<u>$i = w_1$</u>: שקול לנוסחא 1. לפי ההגדרה, על התיבות ברוחב המינימאלי לא ניתן לערום דבר, מכיוון ש-wi <- w1. הגובה העצמי שלהן הוא אכן הגובה המקסימאלי למגדל בו הן הבסיס. אם לא קיימת תיבה באורך זה, מן הסתם אין בסיס ואורך מגדל ריק הוא 0.

<u>$I = w_k$</u>: נניח כי לכל עמודה במטריצה מ-w_1 עד w_k, התאים מכילים ערך opt נכון.

② $opt(l_x,w_z) = \max(opt(l_x,w_{z-1}),h(l_x,w_z) + \max(\{opt(l_y,w_{z-1}) \mid y < x\}) \quad | \forall(l_x,w_{z>1}) \in B$

$opt(l_x,w_z) = opt(l_x,w_{z-1}) \quad | \forall(l_x,w_{z>1}) \notin B$

<u>$I = w_{k+1}$</u>: נסתכל על כל תא $[l_x,i]$. נשים לב כי כל איברי המטריצה מאורך כלשהו עד w_k ממוינים בסדר מונוטוני לא יורד, עקב השימוש בפונקציית מקסימום הכוללת את האיבר באותו האורך והרוחב הקודם בשני המקרים – בעלת ערך גדול יותר מעמודות קודמות ויוצרת מגדל יציב.

אם לא קיימת תיבה מגודל זה, אז אין חלופה ברוחב I למגדל היציב המקסימאלי באורך l_x, והוא נותר

$opt(l_x,w_{z-1})$ על פי הנחת האינדוקציה והסדר המונוטוני, כמתואר במקרה השני של נוסחא 2.

אם קיימת תיבה כנ"ל, ננסה לבנות מגדל יציב מקסימאלי חלופי T. על הבסיס החדש ניתן לערום כל "תת מגדל" יציב שבסיסו ברוחב ואורך קטנים יותר = רשימת תיבות, כיוון שאין דרישת יחס בין שני איברים לא סמוכים במגדל. על פי הנחת האינדוקציה, תת המגדל T', המורכב מרשימת התיבות פרט לבסיס T, חייב להיות מוכל במטריצת opt, כי בסיסו מגודל $\geq w_k$: אם הוא אינו הגבוה ביותר (ועל כן לא נשמר במטריצה), נוכל להחליף אותו במגדל שמתאים לopt, למשל T*, ולקבל מגדל T2 באורך

$h(T2) = h(T^*) + h[l_x,i] > h(T') + h[l_x,i] = h(T)$

בסתירה למקסימאליות T.

נוסף על כך, באמצעות טיעון החלפה דומה לנ"ל, ניתן להראות כי תת מגדל T' בהכרח <u>מוכל בעמודה w_k</u> עד אורך l_x לא כולל. ביחד עם השוואה לחלופה של המגדל הקודם באורך l_x, נקבל נכונות נוסחא ②.

③ $max\_stack = \max(\{opt(l_x,w_m) \mid \forall l_x\})$

נוסחא זו נובעת ישירות מהגדרת opt – בחירת המקסימום בין מגדל עם בסיס בכל אורך (על פי הקבוצה) וכל רוחב (על פי הגדרה) בהכרח יחזיר את המגדל היציב המקסימאלי האפשרי.

האלגוריתם משטח את צורת המטריצה של opt למערך; תאים בהם $(l_x,w_z) \notin B$ אינם מעודכנים במקום להעתיקם מהעמודה הקודמת. צ"ל כי באיטרציה על תיבה (l_x,w_z) נכנס לתור ערך $opt(l_x,w_z)$, שבהמשך בהכרח יישפך ל-M; בסוף הריצה המערך מכיל opt שקול לעמודה w_m, ולפי נוסחא ③ / פקודה 7 נחזיר את הגובה המקסימאלי. נוכיח באינדוקציה על ערך הרוחב הנוכחי באיטרציית (l_x,w_z) עם l_x כלשהו.

$W_z = w_1$: מופעלת פקודה 5.2.1 בהתאמה לנוסחא ①.

$W_z = w_k$: נניח כי בזמן איטרציה (l_x,w_k), מערך M מכיל את ערכי $opt(l_x,w_{k-1})$ – טרם עדכון תור.

$W_z = w_{k+1}$: נסתכל על נוסחא ②, שורת קוד 5.3.1, ונציב את $M(lx) = opt(lx,wm)$ מההנחה מעלה.

$$opt(l_x,w_k) = \max(opt(l_x,wk_{-1}),h(l_x,w_k) + \max(\{opt(l_y,w_{k-1}) \mid y < x\})$$

$$\mathbf{Q[lx]} = \max(M[fl[lx]] , h + \max(\{M[y] \mid y < fl[lx]\})) =$$

$$= \max(opt(l_x,w_{k-1}),h(l_x,w_z) + \max(\{opt(l_y,w_{z-1}) \mid y < x\}) = \mathbf{opt(l_x,w_k)}$$

$$\Rightarrow \mathbf{return\_value} = \max(M) = \max(opt[l_x,w_m] \mid \forall l_x) = \mathbf{max\_stack} \blacksquare$$

[הערה : במקום טבלת גיבוב, שסיבוכיות המקום שלה מעט עמומה, אפשר לשמור מערכי המרה בכיוון ההפוך, למשל I: wi, לעדכן את האורך והרוחב לאינדקס, ובסוף האלגוריתם להמיר אותם חזרה ל-li / wi במקור.]

.3

א. נגדיר את הפולינומים בצורה יותר נוחה לעבודה:

$q(x) = ax + b, r(x) = cx + d, s(x) = ex + f$

על פי הגדרת פולינום אינטרפולציה, נדרוש כי הערכים יתאימו לנקודות הנתונות:

① $p_{i,j}(x_i) = p(x_i) = y_i \quad ; \; p_{i+1,j+1}(x_i) = y_i'$

② $p_{i,j}(x_k) = p_{i+1,j+1}(x_k) = p(x_k) = y_k \quad | I < k \le j$

③ $p_{i+1,j+1}(x_{j+1}) = p(x_{j+1}) = y_{j+1} \quad ; \; p_{i,j}(x_{j+1}) = y_{j+1}'$

המקדמים הנ״ל (כנראה) פורשים תת מרחב פתרונות אינסופי, אז נרצה לצמצם את בחירתם לערכים נוחים יותר לעיבוד. שימו לב כי k אינו בהכרח קיים, ואם $y_k = 0$ היחידי אז לא ניתן להסיק דבר ממשוואה ② לגבי המקדמים; אבל הוא עדיין יוצר סט משוואות הצריכות להתקיים.

① $yi = (q(xi) * yi - r(xi) * yi') / s(xi)$

נבצע צמצום ע״י בחירת $r(xi) = 0$, וכן $q(xi) / s(xi) = 1 \Leftarrow$ המשוואה מתקיימת, $yi = yi$.

$\Rightarrow \mathbf{d + c * xi = 0 \quad ; (a - e) * xi + (b - f) = 0}$

③ $y[j+1] = (q(x[j+1]) * y[j+1]' - r(x[j+1]) * y[j+1]) / s(x[j+1])$

בצורה דומה נצמצם $q(x[j+1]) = 0$ וכן $r(x[j+1]) / s(x[j+1]) = -1$, כדי לקבל $y[j+1] = -(-1 * y[j+1])$.

$\Rightarrow \mathbf{b + a * x[j+1] = 0 \quad ; (c + e) * x[j+1] + (d + f) = 0}$

כמו כן, נדרוש גם את קיום המשוואה השנייה על כל מקרה:

② $yk = (q(xk) * yk - r(xk) * yk) / s(xk)$

$\Rightarrow s(xk) = q(xk) - r(xk) \Rightarrow (c + e - a) * xk + (d + f - b) = 0$

כזכור, xk מייצג מספר ערכים שונים, אז במקרה הכללי נדרוש שהמקדמים יתאפסו בנפרד:

$\mathbf{c + e - a = 0 \; ; d + f - b = 0}$

יש לנו 6 משוואות עם 6 נעלמים, אך שתיים מהמשוואות מתקזזות (מטריצה לא הפיכה).

מתקבלת התוצאה הבאה, כאשר $y = xi, z = x[j+1]$:

$a = e + ez / (y - z) + f / (x - z)$

$b = -e*y*z / (y - z) + f - y*f / (y - z)$

$c = e*z / (y - z) + f / (y - z)$

$d = -e*y*z / (y - z) - y*f / (y - z)$

עכשיו מותר לנו לבחור כמעט כל ערך ל-e, f (למעט 0,0 מן הסתם): נבחר $e = 0, f = (y-z)$, כי קל יותר לבצע חלוקה בקבוע מאשר בפולינום (קרי: אפשרי בכלל) ו-f מצטמצם בצורה נחמדה.

$a = 1 \; ; \; b = y - z - y = -z \; ; \; c = 1 \; ; \; d = -y \; ; \; e = 0 \; ; \; f = (y - z)$

בסה״כ, הפולינומים הבאים לדוגמא יקיימו את המשוואה המקורית:

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

$\mathbf{r(x) = x - x_i}$

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

ב. נשים לב, כי ניתן להשתמש במשוואה שנמצאה בסעיף הראשון לצורך אפיון נוסחת נסיגה, מכיוון שיש יחס ישיר בין האיבר הבא לשני איברים קודמים (הפער בין I ל-j מצטמצם בשני הכיוונים).

נגדיר opt(i,j) = פולינום אינטרפולציה על נקודות (xi,yi) .. (xj,yj).

① $opt(i,i) = (y_i) \quad | \quad 1 \le I \le n$

② $opt(i,j+1) = ((-x_{j+1} + x) * opt(i,j) + (x_i - x) * opt(i+1,j+1)) / (x_i - x_{j+1}) \quad | \quad j + 1 \ge I \ge 1$

③ interpolation_polynomial = opt(1,n)

<u>תיאור האלגוריתם</u>: נשמור במערך דו מימדי בגודל N x N את פולינומי האינטרפולציה על הנקודות מ-i עד j כאשר I = מספר שורה וj = מספר עמודה, בייצוג וקטור מקדמים. נמלא אותו על פי שתי הנוסחאות הראשונות מעלה, כאשר כפל בין פולינום OPT(ljl) לפולינום מדרגה 1 כנ"ל מתורגם ל<u>הזחת המקדמים וכפל + חיבור אלמנטריים</u>, החיבור הוא <u>אלמנטרי עבור כל מקדם</u>, <u>והחילוק הנדרש הוא אלמנטרי למקדם</u> בלבד – כלומר, כל פעולת וקטורים בסיבוכיות O(j). בשיטה האיטרטיבית, נתחיל מהאלכסון המרכזי ונבנה את המשולש העליון בהדרגה לפי התא משמאל + התא מתחת. לבסוף נחזיר את opt(1,n) כתוצאת האינטרפולציה.

<u>פירוט</u>:

קלט: רשימת נקודות במישור V = {(xi,yi) | 1 ≤ I ≤ n}.

1) אתחל מערך דו מימדי M בגודל n x n עם וקטורים ריקים.

2) לכל j = 1:n בצע: $M[j,j] = (y_j)$

3) לכל j = 1:n-1 בצע:

3.1) לכל I = 1:n-j בצע:

3.1.1) $p1 := multiply\_vectors(M[i,i+j-1],(-x_{j+1},1))$

3.1.2) $p2 := multiply\_vectors(M[i+1,i+j],(x_i,-1))$

3.1.3) $M[i,i+j] = (p1 + p2) / (x_i - x_{j+1})$

4) החזר M[1,n].

multiply_vectors:

1) צור וקטור_תוצאה ע"י הכפלת כל מקדם בפרמטר 1 במקדם 0 של פרמטר 2.

2) הוסף כל מקדם בפרמטר 2 * מקדם 1 של פרמטר 2 במקום הבא אחרי המקדם לוקטור_תוצאה.

3) החזר וקטור_תוצאה.

<u>סיבוכיות</u>: $\mathbf{O(n^3)}$ פעולות אלמנטריות:

- אתחול המערך M ב-O(n^2) – אולי 'בחינם' כי משתמשים בערכים ריקים, אבל ההבדל לא משמעותי.

- מספר הפעולות עבור חישוב תא מסוים באלכסון j הינה c * j עקב כל פעולות הוקטורים כמתואר, וקיימים n – j תאים כאלה ⇐ sum(c * j * (n-j) | j = 1:n-1) = c / 6 * (n-1) * n * (n+1) = O(n^3)

[ודרך אגב, את פולינום הסכום הנ"ל אפשר לחשב באמצעות האלגוריתם...]

- החזרת איבר טריוויאלית.

<u>הוכחה</u>: את נכונות נוסחת הנסיגה הוכחנו בסעיף א; כאן ניתן לה ביטוי בצורה מטריצה.

נוכיח כי M[i,i + j] = opt(i,i + j) באינדוקציה על ערכי I,j.

J = 0: פקודה 2 תואמת לנוסחא ① .

J = k: נניח כי M[i,i + k] = opt(i,i + k) לכל I.

J = k+1: נסתכל על איחוד פקודות 3.1. ונוסחא ③ לכל i.

$opt(i,k+1) = ((-x_{k+1} + x) * opt(i,k) + (x_i - x) * opt(i+1,k+1)) / (x_i - x_{k+1})$

M[i,k+1] $= ((-x_{k+1} + x) * M[i,j] + (x_i - x) * opt(i+1,k+1)) / (x_i - x_{k+1}) = ((-x_{k+1} + x) * opt(i,k) + (x_i - x) * opt(i+1,k+1)) / (x_i - x_{k+1}) =$ **opt(i,k+1)**

ועל פי נוסחא 3 :

⇒ **return_value** = M[1,n] = **interpoly**

ג. p(x) = (0,1,2,3,4)

נחשב כמה נקודות פשוטות (שלמים וערכים לא גדולים) :

(0,0), (1,10), (2,98), (-1,2), (-2,46)

$opt(i,j+1) = ((-x_{j+1} + x) * opt(i,j) + (x_i - x) * opt(i+1,j+1)) / (x_i - x_{j+1})$

M[1,1] = (y1) = (0)

M[2,2] = (10)

M[3,3] = (98)

M[4,4] = (2)

M[5,5] = (46)

M[1,2] = ((-1 + x) * 0 + (0 – x) * 10) / (0 – 1) = (0,10)

M[2,3] = ((-2 + x) * 10 + (1 – x) * 98) / (1 – 2) = (-78,88)

M[3,4] = ((1 + x) * 98 + (2 – x) * 2) / (2 + 1) = (34,32)

M[4,5] = ((2 + x) * 2 + (-1 – x) * 46) / (-1 + 2) = (-42,-44)

M[1,3] = ((-2 + x) * (0,10) + (0 – x) * (-78,88)) / (0 – 2) = (0,-29,39)

M[2,4] = ((1 + x) * (-78,88) + (1 – x) * (34,32)) / (1 + 1) = (-22,4,28)

M[3,5] = ((2 + x) * (34,32) + (2 – x) * (-42,-44)) / (2 + 2) = (-4,13,19)

M[1,4] = ((-1 + x) * (0,-29,39) + (0 – x) * (-22,4,28)) / (0 + 1) = (0,-7,6,11)

M[2,5] = ((1 + x) * (-22,4,28) + (2 – x) * (-4,13,19)) / (1 + 2) = (-16,1,22,3)

M[1,5] = ((2 + x) * (0,-7,6,11) + (0 – x) * (-16,1,22,3)) / (0 + 2) = **(0,1,2,3,4)** = **P(x)** ▌

15/20

.4

a -5

א. האלגוריתם הנתון מחזיר את אותו מיפוי כמו A^*: המשקל של מסלול מזערי מקודקוד r לכל קודקוד אחר בגרף G (או אינסוף אם לא קיים מסלול עקב אי קשירות). בסריקה בלתי יעילה למדי.

נוכיח זאת באינדוקציה על אורך המסלול המזערי (כן, מספר הקודקודים בו) של כל קודקוד v בגרף: כלומר, אם $length(P(v)) = k$, אז אחרי k איטרציות לכל היותר נקבל $A(v) = w(P)$. נכלול את האתחול כאיטרציה לצורך הנוחות.

$\underline{n = 1}$: בעת האתחול, r הוא היחיד מקבל את המשקל 0, התואם למסלול המכיל רק אותו.

$\underline{n = k}$: נניח כי אחרי k איטרציות, A קיבל את משקלי המסלולים המזעריים עבור כל הקודקודים במרחק $\leq k$.

$\underline{n = k + 1}$: נתבונן על קודקוד v, המקיים $P(v) = (r,v_1,v_2..v_k=v)$ מסלול המזערי (ללא הנחה על סדר מסוים). כפי שניתן לראות, $len(P(v)) = k+1$.

יהא $P'(v) = (r,v_1,v_2,..v_{k-1})$ תת המסלול של P לשכן של v. על פי משפט, P' הינו מסלול מזערי גם כן. נוסף על כך, $len(P'(v)) = k$, ולכן על פי ההנחה $A[v_{k-1}] = w(P'(v))$. במידה ו $A[v] > w(P(v))$ בתחילת איטרציה k, אז כאשר הלולאה תסרוק את הקשת (v_k,v), בלי קשר לסדר, היא תגיע למסקנה כי יש לעדכן את $A[v]$: ומה קורה במידה ו$A[v]<w(P(v))$? צריך להראות שזה לא יכול לקרות.

$\mathbf{A[v]} := A[v_{k-1}] + c(v_{k-1},v) = w(P'(v)) + c(v_{k-1},v) = \mathbf{w(P(v))}$, כנדרש.

התכנסות: לא ניתן למצוא מסלול בעל משקל קטן מהמזערי על פי הגדרה, לכן בהכרח אחרי n איטרציות חיצוניות (כולל האתחול) האלגוריתם חייב למצוא את כל המסלולים המזעריים, ולסיים באיטרציה שאחרי.

ב. יהא $\mathbf{M(n) = (\{v_i\},\{(v_i,v_{i-1}) \mid 1 < I \leq n\}}$, וכן $\mathbf{r = v_n}$ – קרי רשימה בסדר לקסיקוגרפי יורד.

עקב הסריקה הלקסיקוגרפית, בכל איטרציה נעדכן רק את v_{n-i}, כי בקשתות $(v_{n-i..2}, v_{n-i-1..1})$ הנסרקות ראשונות $A[v_{n-i..1}] = \infty$ (מיני אינדוקציה), ולכן מתעדכנת קשת אחת בדיוק בכל איטרציה, וקיימות n-1 קשתות; לבסוף נדרשת איטרציה נוספת לצורך ווידוא סיום.

$\Leftarrow \mathbf{B(n) = n}$, לא כולל אתחול.

ג. יהא $\mathbf{M'(n) = (\{v_i\},\{(v_{i-1},v_i) \mid 1 < I \leq n\}}$, וכן $\mathbf{r = v_1}$ – קרי רשימה בסדר עולה. כפי שניתן לראות, $|E(M'(n))| = |E(M(n))|$ – מספר הקשתות הוא n-1.

עקב הסריקה הלקסיקוגרפית, באיטרציה הראשונה יעודכן כל A למשקל המסלול המזערי (והיחיד) – תחילה נבדקת הקשת (v1,v2) ומתעדכן A[v2], לאחר מכן (v2,v3); מכיוון ש A[v2] כבר חושב, ניתן להמשיך ל A[v3] וכו'. באיטרציה השנייה אין עדכונים והאלגוריתם יסיים. ∎

ב ו-ג מעולה.