פתרון ממן 13

הפתרון נכתב על-ידי שלומי וקנין

שאלה 1

א. נוכיח באינדוקציה על אורך מסלול:

a. מסלול באורך 1 : ברור, כי הקשת היחידה היא גם הקצרה ביותר
b. נניח שנכון לכל מסלול בעל אורך קטן מ n
c. עבור מסלול (קצר ביותר) מ s ל t באורך n, הקשת האחרונה מ x ל t תהיה קשת שימושית לפי הגדרת הקשת השימושית, ושארית המסלול מ s ל x מורכב רק מקשתות שימושיות לפי הנחת האינדוקציה (אורכו קטן מ n).

ב. יהי P מסלול מ s ל t המכיל קשת לא שימושית אחת, $e = (u, v)$
נניח בשלילה שהמסלול P הוא מסלול קצר ביותר.
נחלק את המסלול לשני חלקים $A = s \to v, B = v \to t$. מתקיים $A + B = P$.
הנתון כי e אינה קשת שימושית אומר כי לא קיים מסלול קצר ביותר $s \to v$ כך שהקשת האחרונה בו היא e, ולכן קיים מסלול אחר $A' = s \to v$, כך שהקשת האחרונה בו שימושית. נרכיב מסלול חדש: $A' + B = P'$, ומתקיים $|A'| < |A|$, כלומר $|P| > |P'| = |B| + |A'|$, בסתירה לנתון כי P מסלול קצר ביותר.

ג. יהי P מסלול מ s ל t, המכיל k קשתות שאינן שימושיות $E = \{e_1, \dots, e_k\}, k > 1$.
נניח בשלילה כי P מסלול שני קצר ביותר.
נבחר בקשת $e_i \in E$ הראשונה המופיעה במסלול P, ונסמן $e = (u, v)$. נחלק את המסלול לשני חלקים $A = s \to v, B = v \to t$. מתקיים $A + B = P$. מסעיף ב' ברור כי המסלול A אינו הקצר ביותר, ולכן נוכל להחליפו במסלול אחר $A' = s \to v$ קצר ביותר, ולכן מורכב רק מקשתות שימושיות.
קיבלנו מסלול $A' + B = P'$, ומתקיים $P' = |A'| + |B| < |A| + |B| = P$, אולם נשארו עדיין $k - 1 > 0$ קשתות שאינן שימושיות במסלול P', ולכן מסעיף ב' קיים מסלול קצר ביותר $s \to t$, שאורכו $\delta(t)$, וקיבלנו $\delta(t) < P' < P$, כלומר, לא יתכן כי P שני קצר ביותר, בסתירה.

כמו כן, אם נניח כי P מסלול שני קצר ביותר שאינו מכיל אף קשת לא שימושית, אז מסעיף א' נקבל סתירה.

ד. **טענה 1:** כל הקשתות המוחזרות מהרצת אלגוריתם דייקסטרה על צומת s הן קשתות שימושיות.
הוכחה: אלגוריתם דייקסטרה מחזיר את כל המסלולים הקצרים ביותר מ s לכל שאר הצמתים, ולפי סעיף א', כל מסלול כזה מורכב מקשתות שימושיות בלבד.

יהי G גרף מכוון. ניצור גרף מכוון 'G המתקבל מהיפוך הכיוון של כל הקשתות ב G
טענה 2: הרצת אלגוריתם דייקסטרה על צומת t ב 'G תחזיר את כל המסלולים הקצרים ביותר מכל הצמתים אל צומת t בגרף G.
הוכחה: נוכיח בשלילה: נניח כי הרצת דייקסטרה על צומת t ב 'G החזירה מסלול P שאינו קצר ביותר מצומת x ל t ב G. מההנחה ש P אינו המסלול הקצר ביותר, נקבל כי קיים מסלול קצר ביותר $P' = (a_1, \dots a_j)$ ב G, אולם מסלול זה יהיה המסלול $(a_j, \dots, a_1)$ ב G', והראנו כי דייקסטרה מצא מסלול שאינו הקצר ביותר ב G', בסתירה.

ועכשיו, לאלגוריתם:

1. הרץ דייקסטרה מצומת s, וסמן את כל הקשתות שדייקסטרה עבר בהן (השימושיות, לפי טענה 1)
2. הרץ דייקסטרה מצומת t, עבור קשתות הפוכות (ייתן מכל צומת את מרחקה ל t, לפי טענה 2)
3. $min \leftarrow \infty$
4. עבור כל צומת $e(u,v)$ שאינה שימושית,
5. $min \leftarrow Min\{min, w(e) + dist(s,u) + dist(v,t)\}$ (שהתקבלו משורות 1 ו 2)
6. החזר את min

נכונות: שורות 1 ו 2 מוצאות את המסלולים הקצרים ביותר מ s לכל צומת, ומכל צומת אל t. מטענות א' ו ב', מסלולים אלו מורכבים מקשתות שימושיות בלבד. לפי סעיף ג', המסלול השני הקצר ביותר יהיה מורכב מקשת שאינה שימושית אחת, ולכן נמצא את הקשת הלא שימושית הנותנת את הסכום המינימאלי, וזה יהיה המסלול השני הקצר ביותר.

סיבוכיות: האלגוריתם מריץ דייקסטרה פעמיים, הופך פעם אחת את כל הקשתות ורץ על כל קשת לא שימושית בזמן $O(1)$. הפיכת הקשתות וריצה על כל הקשת הלא שימושיות לוקחות $O(|E|)$, זמן זניח ביחס לדייקסטרה, ומכאן שהאלגוריתם רץ בזמן ריצה של דייקסטרה.

<u>שאלה 2</u>

א. יהי T עץ. נבחר את שורשו להיות s, ונוריד אותו. נקבל $m = rank(s)$ רכיבי שקילות, בכל אחד יש s_i צמתים. נסמן $a = \max_{i\in[1..m]} s_i$ נבחר את c להיות $c = \frac{a}{a+1}$
ברור כי $s_i \le a = \frac{an}{n}$, אולם $n \ge a + 1$ בגלל שתמיד נוריד לפחות צומת אחת (השורש), ולכן קיבלנו כי $s_i \le \frac{an}{n} \le \frac{a}{a+1} n = cn$, מש"ל.

ב. **טענה 1**: מרחק קצר ביותר בין u ל v הוא גם המרחק בין v ל u
הוכחה: הגרף אינו מכוון, ולכן לו היה קיים מסלול p בעל מרחק בין u ל v קצר יותר מהמרחק הקצר ביותר בין v ל u, הרי שהיינו הופכים את מסלול p, ומוצאים מרחק קצר יותר מהמרחק הקצר ביותר שמצאנו, בסתירה.

טענה 2: מרחק של צומת מעצמו הוא 0
הוכחה: טריוויאלית – עלינו לעבור 0 קשתות כדי להגיע מצומת אל עצמה

טענה 3: מרחק מצומת x לצומת y, כך ש x <u>אינו</u> אב קדמון שלה הוא $1 + \mathrm{Length}(\mathrm{parent}(\mathrm{x}), \mathrm{y})$
הוכחה: מאחר ואנחנו רצים על עץ, הרי שאין מעגלים, ולכן יש רק מסלול אחד (ובהכרח הקצר ביותר) בין כל שני צמתים. מהעובדה כי x אינו אב קדמון של y, על מנת להגיע אל x מ y עלינו לעבור דרך אביו של x. לכן, המרחק בין אביו של x ל y יחד עם אורך הקשת מ x לאביו, יתנו את המרחק המבוקש.

אלגוריתם:

```
mod_dfs(Lengths, Explored, s)
  if s = nil then return
  for each node e in Explored
    if e = s then Length[s,s] = 0                // טענה 2
    else Length[s,e] <- 1 + Length[parent[s],e] // טענה 3
         Length[e,s] <- Length[s,e]              // טענה 1
  Explored[s] = true
  for each u <- child of s
     mod_dfs(Length, Explored, u)
```

נכונות:
1. האלגוריתם מסתיים, כי תמיד מורידים לפחות צומת אחת
2. נראה שבתנאי היציאה הערך נכון – עץ ריק לא משנה דבר בטבלת המרחקים
3. נראה שהקריאה הרקורסיבית עושה את הדבר הנכון – כאשר סיימנו לחשב את המרחקים עבור צומת בא אנו נמצאים, נמשיך בחישוב עבור כל הילדים, ונשתמש בטענה 3 כדי לחשב את המרחקים.

זמן ריצה: האלגוריתם דומה מאוד ל DFS, ולכן זמן ריצתו ליניארי.

<u>שאלה 3</u>

מהנתון כי $DFT_n\big(p(x)\big) = (v_1, \dots v_n)$ אנו יודעים כי $p(w_n^k) = p\left(e^{\frac{2\pi i}{n}k}\right) = v_k$ עבור $0 \le k \le n-1$

החישוב של $DFT_{2n}\big(p(x^2)\big)$, ידרוש וקטור תשובה בגודל $2n$,
ולכן בחישוב זה $0 \le k \le 2n-1$

נציב את w_{2n}^k בפונקציה x^2, ונקבל כי

$$(w_{2n}^k)^2 = w_{2n}^{2k} = e^{\frac{2\pi i}{2n}2k} = e^{\frac{2\pi i}{n}k}$$

קיבלנו כי בתחום $0 \le k \le n-1$ השורשים חייבים להיות זהים,

נראה מה קורה בתחום $n \le k` \le 2n-1$:

ברור כי $k + n = k`$ ולכן נוכל לרשום את התחום הנ"ל כך : $n \le k+n \le 2n-1$, כאשר $0 \le k \le n-1$

$$(w_{2n}^{k`})^2 = (w_{2n}^{k+n})^2 = w_{2n}^{2(k+n)} = e^{\frac{2\pi i}{2n}2(k+n)} = e^{\frac{2\pi i}{n}(k+n)} = e^{\frac{2\pi i}{n}k+\frac{2\pi i}{n}n} = e^{\frac{2\pi i}{n}k+2\pi i}$$

אולם הוספה של $2\pi i$ בחזקה היא סיבוב של 360 מעלות בדיוק, ולכן נקבל כי גם בתחום $n \le k` \le 2n-1$ אנו מקבלים:

$$(w_{2n}^{k`})^2 = e^{\frac{2\pi i}{n}k}$$

ומכאן, שלכל התחום $0 \le k \le 2n-1$:

$$DFT_{2n}\big(p(x^2)\big) = (v_1, \dots v_n, v_1, \dots, v_n)$$

כנדרש.

<u>שאלה 4</u>

א. נראה 5 מכפלות אשר יתנו ריבוע של מטריצה *2x2*:

$$A = \begin{pmatrix} a & b \\ c & d \end{pmatrix} => A^2 = \begin{pmatrix} a & b \\ c & d \end{pmatrix}\begin{pmatrix} a & b \\ c & d \end{pmatrix} = \begin{pmatrix} a^2 + bc & ab + bd \\ ca + dc & cb + d^2 \end{pmatrix}$$

נגדיר את חמשת המכפלות הבאות:

1. a^2
2. d^2
3. $bc = cb$
4. $b(a + d)$
5. $c(a + d)$

נראה כי בעזרת חמשת המכפלות הנ"ל ניתן לבנות את כל ארבעת המחוברים, מעל הממשיים :

1. $a^2 + bc = (1) + (3)$
2. $ab + bd = b(a + d) = (4)$
3. $ac + cd = c(a + d) = (5)$
4. $cb + d^2 = (3) + (2)$

ומכאן שכל מטריצה בגודל 2x2 ניתן להעלות בריבוע בעזרת 5 פעולות כפל של מספרים ממשים.

ב. האלגוריתם המוצע אינו פותר את הבעיה:
האלגוריתם מנסה לפתור את הבעיה בצורה רקורסיבית, לפי סעיף א' :

$$M_{nxn} = \begin{pmatrix} A_{\frac{n}{2}x\frac{n}{2}} & B_{\frac{n}{2}x\frac{n}{2}} \\ C_{\frac{n}{2}x\frac{n}{2}} & D_{\frac{n}{2}x\frac{n}{2}} \end{pmatrix} = \begin{pmatrix} A^2 + BC & AB + BD \\ CA + DC & CB + D\text{^}2 \end{pmatrix}$$

כלומר חמשת תת הבעיות הן:

1. A^2
2. D^2
3. BC
4. $B(A + D)$
5. $C(A + D)$

אולם, במרחב המטריצות כפל אינו חילופי, ולכן :

$BC \neq CB$
$B(A + D) \neq$AB+BD
$C(A + D) \neq CA + DC$

ומכאן שהאלגוריתם לא יכול להשתמש בחמשת המכפלות הללו, ולכן הוא שגוי.
בנוסף האלגוריתם מכפיל מטריצות זהות זו לזו (מעלה בריבוע) ואילו הקריאות הרקורסיביות מתבקשות להכפיל שתי מטריצות שאינן בהכרח זהות.

שאלה 5

תהי M_{nxn} מטריצה

טענה 1: לכל מטריצה M יש מינימום מקומי
הוכחה: נבחר את המספר הקטן ביותר במטריצה. מכך שהוא מינימאלי הוא קטן מכל שאר המספרים המטריצה, ובפרט מהאיברים משני צידיו, מעליו ומתחתיו.

טענה 2: בכל מטריצה M, המינימום המקומי ימצא או על שפת המטריצה, או בפנים המטריצה
הוכחה: מטענה 1, לכל מטריצה יש מינימום מקומי, אם הוא אינו על השפה, אזי נוכל לקצץ מהמטריצה את שפתה, ולקבל מטריצה חדשה, קטנה יותר, אשר גם עליה חלה טענה 1.

יהיו $i = 1,2,3,4$, $M^i_{\frac{n}{4}x\frac{n}{4}}$ תתי המטריצות המתקבלות מחלוקת מטריצה M לארבע חלקים.
כמו כן, תהיה S^i קבוצת כל האיברים שעל שפת M^i, ונסמן $s_m = Min_{i\in[1..4];k\in[1..\frac{n}{4}]}S^i_k$, כלומר האיבר הקטן ביותר על כל השפות S^i.

טענה 3: אם s_m הפינה, אז הוא המינימום.
אחרת, מספיקה השוואה אחת בלבד על מנת לדעת האם s_m הוא מינימום מקומי או לא.
הוכחה: s_m נמצא על שפה של תת מטריצה M^i כלשהי. מכך שהשפה סגורה, ברור כי לכל תא בשפה ישנם שני שכנים באותה השפה.
אם s_m נמצא בפינה, או שהפינה היא הפינה של המטריצה המלאה M, ואז s_m בוודאות הוא המינימום המקומי (כי שני התאים הנוספים שיש להשוות מולן נמצאים על השפה, ובפרט גדולים מהמינימום), או ש s_m צמוד לשפות של שתי תתי מטריצות, ובפרט קטן יותר מכל האיברים על שפותיהן, וגם כאן הוא המינימום המקומי.
אחרת, לכל צלע בשפה, או שהצלע היא גם השפה של M המקורית, או שהצלע צמודה לצלע שפה של M^k אחר $(k \neq i)$. במקרה הראשון, השפה היא השפה של המטריצה המלאה, ולכן לא ניתן להשוות לאיבר שמחוץ למטריצה, ומתוך ארבעת האיברים להשוואה נותר איבר בודד, ובמקרה השני s_m הוגדר להיות המינימום על כל השפות, ולכן קטן מאיבר זה, וגם כאן, מתוך ארבעת האיברים להשוואה הראנו כי s_m תמיד קטן משלושה איברים, ולכן נשאר להשוות רק עוד איבר נוסף בודד.

תיאור האלגוריתם:

1. חלק את המטריצה ל 4 חלקים
2. מצא את המינימום משפות כל ארבעת המטריצות שנתקבלו
3. אם המינימום נמצא בפינה, אז מצאנו מינימום מקומי, סיים. (טענה 3)
4. אחרת בדוק את ערך התא שעוד לא נבדק, השכן של המינימום (טענה 3)
5. אם ערך התא גדול מהמינימום, אזי מצאנו מינימום מקומי, סיים (הגדרת המינימום המקומי)
6. אחרת קרא ברקורסיה על תת המטריצה המכילה את המינימום

הוכחת נכונות:
1. נראה שהאלגוריתם מסתיים: בכל קריאה רקורסיבית אנו מקטינים את גודל המטריצה, ולכן במקרה הגרוע ביותר, נגיע למטריצה בגודל 2x2, בא כל התאים הם פינות, ומטענה 3 המינימום יהיה אחד מהם.
2. נראה שכאשר הוא מסתיים, תשובתו נכונה: אם האלגוריתם יסתיים בשורה 3 אז הוא נכון לפי טענה 3, ואם בשורה 5 ואז הוא נכון לפי טענה 3 והגדרת המינימום המקומי.

3. נראה שהקריאה הרקורסיבית עושה את הדבר הנכון: מטענה 2, או שהמינימום על השפה, או שהוא בפנים המטריצה. מכך שהוא לא על השפה, אנו קוראים ברקורסיה לפנים המטריצה, ולכן הקריאה נכונה.

משלושת אלו, נובע כי האלגוריתם נכון.

נחשב זמן ריצה:

1. מציאת מינימום – במטריצה בגודל nxn אשר חולקה ל 4 מטריצות בגודל $\frac{n}{4}x\frac{n}{4}$, יש $\frac{16n}{4}=4n$ איברים, ומציאת מינימום ב 4n איברים לוקחת $O(n)$
2. בדיקה האם המינימום הוא גם מינימום מקומי – מטענה 3 לוקחת השוואה אחת, ולכן $O(1)$
3. וקיבלנו את נוסחת הנסיגה: $T(n)=T\left(\frac{n}{2}\right)+O(n)$, ולפי משפט האב מקרה 3 נקבל O(n),

כנדרש