<u>ממ"ן 12</u>

שאלה 1

הרעיון של האלגוריתם הוא ראשית לבדוק אם הקשת e שייכת בכלל ל T. אם לא - אז בהכרח העץ הפורש המינימלי T ב G הוא גם עץ פורש מינימלי של G' (כפי שיוכח בהמשך) ואין צורך בשום שינוי.

אם הקשת e שייכת ל T הרעיון הוא לבנות עץ פורש מינימלי של G' ע"י פירוק העץ המקורי T לשני תת עצים שמופרדים ע"י הקשת שהוסרה e וחיבורם מחדש ע"י מציאת הקשת הזולה ביותר שמחברת בין שני קודקודים כלשהם ששייכים לשני התת עצים.

האלגוריתם מחולק לשני חלקים. חלק ראשי ופרוצדורת MARK_TREE שמשמשת לסימון כל הקודקודים ששייכים לתת עץ של T שממנו התחלנו את הסריקה (שים לב שפרוצדורת MARK_TREE עוברת רק על הקשתות ששייכות ל T ולכן אינה מהווה סריקת DFS רגילה על הגרף G).

נסמן ב r את אחד הקודקודים של T, אותו נבחר באופן שרירותי מקבוצת הקודקודים V של G כשורש העץ T.

<u>MAIN ALGORITHM:</u>

Initialize an array VISITED whose size is the number of nodes in G. Set every item to FALSE.

Initialize a boolean variable E_FOUND to FALSE.

Call MARK_TREE(r, e) on the root r of T and specify the edge e = (u,v) (the edge that was deleted from G in order to get G’).

```
IF E_FOUND = FALSE THEN
        RETURN T
ELSE
        Initialize MIN to +∞
        Initialize d to NONE
        FOR every edge k = (x,y) in G’
                IF VISITED[x] = TRUE AND VISITED[y] = FALSE THEN
                        IF weight(k) < MIN THEN
                                MIN = weight(k)
                                d = k
                        END IF
                END IF
        END FOR
```

RETURN the tree T’ that is built by taking the union of 3 parts:

1. The subtree of T composed from all nodes in T that were marked as VISITED and all edges in T that connect these nodes

2. The subtree of T that is rooted by the node v where VISITED[v] = FALSE (remember that we marked the edge e that was removed from G as (u,v)) where all of the edges are from T and all of the nodes are from T where for every node c we have VISITED[c] = FALSE.
3. The edge d that was discovered in the last step and is used to stitch these two sub-trees together.

END IF

MARK_TREE(node n, edge e)

```
    VISITED[n] = TRUE
    FOR every edge k that connects node n to some node z in T
        IF k = e THEN
            E_FOUND = TRUE
        ELSE
            Call MARK_TREE(z, e)
        END IF
    END FOR

END MARK_TREE
```

כדי להוכיח את נכונות האלגוריתם נוכיח ראשית את הטענה הבא:

טענה: יהי גרף $G=(V,E)$ לא מכוון וקשיר עם משקלות חיוביים לקשתות ויהי T עץ פורש מינימלי של G. יהי G' הגרף המתקבל מ G ע"י הסרת קשת e מהגרף G. אם T לא כולל את e ו G' קשיר אז בהכרח T הוא עץ פורש מינימלי של G'.

הוכחה: נניח בשלילה ש T אינו עץ פורש מינימלי של G' ויהי T^* עץ אחר שהוא עץ פורש מינימלי של G'. כלומר סכום המשקלות של T^* קטן מזה של T אבל T^* פורש גם את G (כי G' קשיר וקבוצת הקודקודים שלו זהה לזו של G) וקיבלנו שיש עץ פורש מינימלי של G (העץ T^*) שמשקלו קטן יותר מזה של T וזו סתירה.

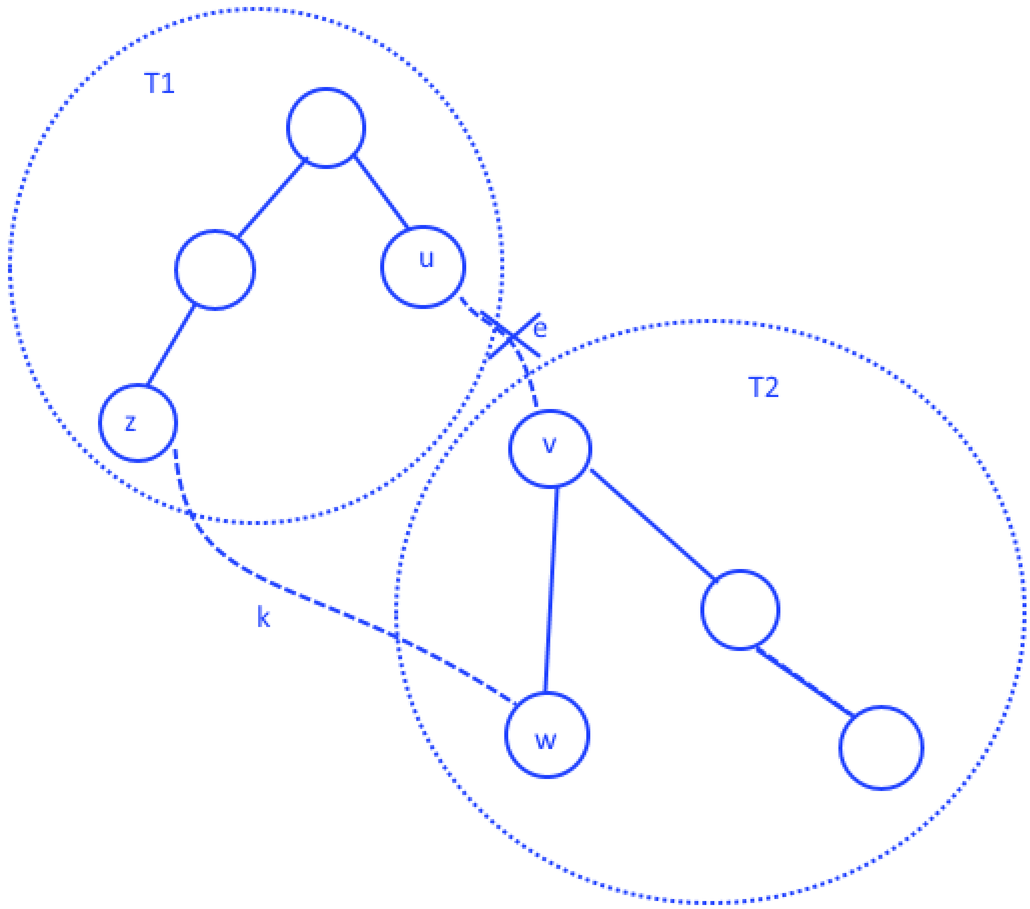
טענה זו מצדיקה את ההחזרה של T כעץ פורש מינימלי של G' במקרה שהקשת e אינה חלק מהעץ T.
האלגוריתם בודק את הערך E_FOUND ובמקרה שהוא FALSE הוא מחזיר את T כעץ פורש מינימלי של G'.

במקרה שהעץ T מכיל את הקשת e נסתמך על הטענה הבאה כדי להוכיח שהעץ שנבנה ע"י איחוד שני התת עצים באמצעות קשת שמחברת בין שני קודקודים כך שכל קודקוד שייך לתת עץ אחר שמשקלה הוא הקטן ביותר - הוא עץ פורש מינימלי של G':

טענה: יהי גרף $G=(V,E)$ לא מכוון וקשיר עם משקלות חיוביים לקשתות ויהי T עץ פורש מינימלי של G. אם נסיר באופן שרירותי קשת e שמחברת שני קודקודים u ו v בעץ T ונוסיף במקומה לעץ T קשת k שהיא הקשת הזולה ביותר שמחברת קודקוד w כלשהו מתת העץ אליו שייך הקודקוד v וקודקוד z כלשהו מתת העץ אליו

שייך הקודקוד u נקבל עץ פורש מינימלי T' של הגרף G' (שהוא הגרף שמתקבל מ G ע"י הסרת הקשת e מ G).

באיור להלן הקשת k היא הקשת הזולה ביותר ב G' שמחברת בין שני התת עצים שהתקבלו מהעץ T ע"י מחיקת הקשת e מ G:



<u>הוכחה:</u> ראשית נשים לב שהקשת k נבחרה להיות הקשת הזולה ביותר שמחברת את החתך של G' שנוצר ע"י חלוקת קודקודי G' לשני תת עצים T1 ו T2 לאחר הסרת הקשת e. לכן, עקב תכונת החתך (cut property) הקשת k חייבת להיות שייכת לעץ הפורש המינימלי של G'. כמו כן - חיבור שני תת עצים ע"י קשת יחידה מייצר עץ וקל לראות שעץ זה פורש את G'. נותר להוכיח שהעץ T' הוא עץ פורש מינימלי של G'.

נניח על דרך השלילה שהעץ T' אינו עץ פורש מינימלי של G'. יהי T^* עץ פורש מינימלי של G'.

T^* הוא עץ ולכן ניתוק הקשת k יגרום ליצירת שני תתי עצים C1 ו C2 כך שקבוצת הקודקודים ב C1 זהה לזו שב T1 (הקשתות שמחברות את קודקודי C1 אולי שונות מאלו שמחברות את קודקודי T1) וקבוצת הקודקודים ב C2 זהה לזו שב T2 (הקשתות שמחברות את קודקודי C2 אולי שונות מאלו שמחברות את קודקודי T2).

מכך ש T' אינו עץ פורש מינימלי של G' נובע שלפחות אחת מהקשתות שמחברות קודקודים ב C1 או ב C2 שונה עבור תת העץ T1 או T2 המקביל (שים לב שהקשת k זהה בשני העצים T^* ו T' ולכן השוני חייב להיות באחד מתתי העצים שמחוברים ע"י k).

אבל אז יכולנו להחליף את תתי העצים T1 ו T2 בתתי העצים C1 ו C2 בעץ הפורש T ולחבר אותם באמצעות e ולקבל עץ פורש של G שסכום המשקלות על קשתותיו נמוך יותר מזה של T וזוהי סתירה מכיוון ש T הוא עץ פורש מינימלי של G.
הנחת השלילה הובילה לסתירה ולכן העץ T' הוא עץ פורש מינימלי של G' ומכאן נובעת הנכונות של האלגוריתם.

הסיבוכיות של האלגוריתם נגזרת מהפעולות הבאות:

1. אתחול מערך VISITED הוא מסדר של $O(n)$ כאשר n הוא מספר הקודקודים ב G
2. הרצת הפרוצדורה MARK_TREE היא בעצם מימוש חלקי של סריקת DFS ולכן היא מסדר גודל של $O(n+m)$ כאשר m הוא מספר הקשתות בגרף G (למעשה כמות הקשתות מוגבלת ע"י כמות הקשתות בעץ T אבל זה לא ישנה בחשבון הסופי).
3. מציאת הקשת הזולה ביותר שמחברת קודקוד שמסומן VISITED לקודקוד שמסומן NOT VISITED דורשת סריקת כל הקשתות בגרף G' וביצוע בדיקות קבועות בזמן על כל קשת ולכן היא מסדר גודל של $O(m)$ כאשר m הוא מספר הקשתות בגרף G.
4. חיבור שני תתי העצים ע"י הקשת שנמצאה בסריקה דורש זמן קבוע ולכן אינו נכלל בחשבון הסופי.

לכן בסה"כ נקבל שסיבוכיות האלגוריתם היא $O(n+m)$

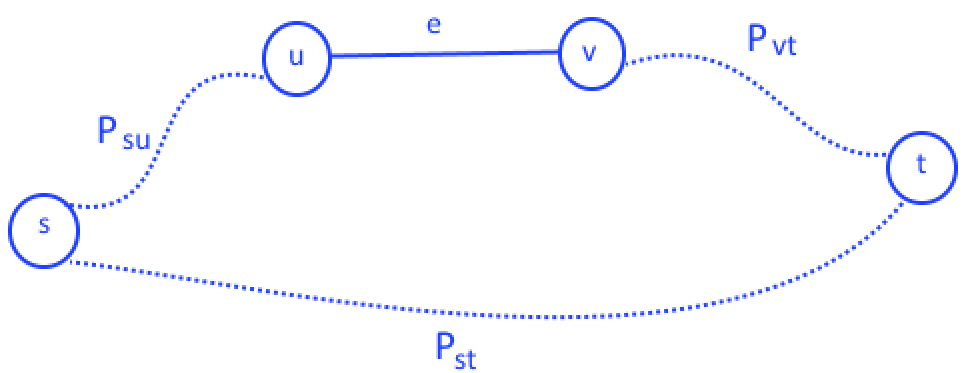
שאלה 2

הרעיון באלגוריתם הוא שיש רק שתי אפשרויות שבהן יכול מסלול קצר ביותר לעבור בין קודקוד s לקודקוד t:

1. המסלול הקצר ביותר עובר דרך הקשת e. במקרה זה נסמן את שני הקודקודים שאותם מחברת הקשת e כ u ו v. מכך ששאר הקשתות הן בעלי משקל לא שלילי נקבל שמסלול זה חייב להיות מורכב מהמסלול הקצר ביותר בין s ל u או v ועוד הקשת e ועוד המסלול הקצר ביותר בין t ל v או u.

2. המסלול הקצר ביותר לא עובר דרך הקשת e

ר' איור להדגמה:



אם המסלול הקצר ביותר עובר דרך הקשת e אז בהכרח הוא מורכב מהמסלול הקצר ביותר Psu בין s לבין u ועוד הקשת e ועוד המסלול הקצר ביותר בין v ל t (המסלול Pvt).
המסלול Pst מציין את המסלול הקצר ביותר בין s לבין t כאשר המסלול הקצר ביותר לא עובר דרך e.

נתון שאין בגרף מעגלים במשקל שלילי - דבר זה מונע את האפשרות לקבל מסלול במשקלים יותר ויותר קצרים (למעשה שליליים עד אינסוף שלילי) ע"י בניית מסלול שנכנס למעגל כזה ולעולם לא יוצא ממנו ולכן זו אופציה שלא מטופלת ע"י האלגוריתם.

<u>האלגוריתם:</u>

בנה גרף G' ע"י הסרת הקשת e מהגרף G.

נסמן כ u ו v את הקודקודים אותם מחברת הקשת e.

הרץ אלגוריתם דיקסטרה לחישוב מסלולים קצרים ביותר בגרף G' מהקודקוד s. עצור את האלגוריתם בנקודה שבה חושבו המרחקים $d(s,v),\ d(s,u),\ d(s,t)$ והמסלולים $p(s,v),\ p(s,u),\ p(s,t)$ מ s אל v, u ו t או עצור בנקודה שבה אין אפשרות להמשיך.

הערה: קל לתחזק במהלך ביצוע אלגוריתם דיקסטרה מערך עזר שמחזיק עבור כל קודקוד את הקודקוד ממנו הגענו אליו ובאמצעות מערך זה לשחזר את המסלולים הקצרים ביותר עבור כל קודקוד אליו הגיע האלגוריתם. אני מניח באלגוריתם שמערך כזה אכן מנוהל כחלק מהרצת אלגוריתם דיקסטרה ולכן יש לנו כתוצאת ההרצה לא רק את המרחקים הקצרים ביותר אלא גם את את המסלולים הקצרים ביותר.

אם $d(s,t) = \infty$ המשמעות היא שהגרף G' אינו קשיר עקב הסרת הקשת e ומכאן שהמסלול הקצר ביותר ב G בין s לבין t חייב לעבור דרך e.

נסמן באות z את **הקודקוד היחיד** מבין u או v כך ש $d(s,z) = \infty$, נסמן באות w את הקודקוד השני מבין u או v שעבורו מתקיים $d(s,w) < \infty$.
כעת נריץ אלגוריתם דיקסטרה לחישוב מסלולים קצרים ביותר בגרף G' מהקודקוד z ונעצור את האלגוריתם כאשר בידינו תוצאת החישוב $d(z,t)$.
המסלול הקצר ביותר הוא איחוד המסלול הקצר ביותר בין s לבין w ביחד עם הקשת e ביחד עם המסלול הקצר ביותר בין z לבין t.

אם $d(s,t) < \infty$ יש שתי אפשרויות:

1. אין מסלול מ s ל t שעובר דרך e
2. יש מסלול מ s ל t שעובר דרך e וגם מסלול מ s ל t שלא עובר דרך e

אם אלגוריתם דיקסטרה חישב שאחד מהמרחקים $d(s,u) = \infty$ או $d(s,v) = \infty$ אז האלגוריתם מחזיר את המסלול $p(s,t)$ שחושב ע"י אלגוריתם דיקסטרה.

אחרת האלגוריתם מחשב מסלול ב G מ s ל t שעובר דרך e ומחזיר את המסלול שעלותו היא הקטנה ביותר בין המסלול שעובר דרך e לבין המסלול שלא עובר דרך e שחושב ע"י אלגוריתם דיקסטרה.

כדי לחשב את המסלול שעובר דרך e האלגוריתם מבצע:

1. מריצים אלגוריתם דייקסטרה על G' כדי לחשב מרחקים/מסלולים קצרים ביותר מ t ל v ול u.
2. נסמן: $M = d(s,u),\ N = d(s,v),\ P = d(t,u),\ Q = d(t,v)$ (שים לב שהמרחקים בין s לבין u ו v חושבו בהרצת דיקסטרה הראשונה בעוד שהמרחקים בין t לבין u ו v חושבו בהרצת דיקסטרה השניה).
3. יש שני מסלולים אפשריים דרך הקשת e:
   a. $s \to u \to v \to t$ - למסלול זה מתאים המשקל: $M + weight(e) + Q$
   b. $s \to v \to u \to t$ - למסלול זה מתאים המשקל: $N + weight(e) + P$
4. נבחר את המסלול שמשקלו הנמוך יותר מבין שניהם.

<u>הוכחת נכונות</u>

במקרה שבו $d(s,t) \ = \ \infty$ מתקיים בהכרח שהמסלול בין s ל t עובר דרך e לכן המסלול הקצר ביותר בין s ל t חייב להיות מורכב מהמסלול הקצר ביותר בין s לבין הקודקוד של e שהוא נגיש ל s, הקשת e והמסלול הקצר ביותר בין הקודקוד של e שאינו נגיש (reachable) ל s ובין הקודקוד t.

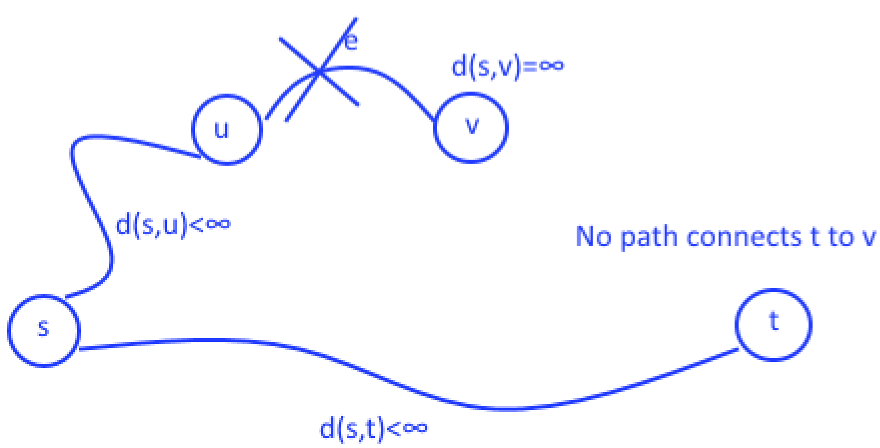
אין אפשרות ששני הקודקודים של e אינם נגישים ל s ב G' מכיוון ששניהם היו נגישים ל s ב G ורק הקשת שמחברת אותם הוסרה.

כמו כן - אין אפשרות ששני הקודקודים נגישים ל s ב G' מכיוון שב G יש מסלול בין t לבין אחד מהקודקודים של e ואז אפשר לחבר את t ל s ב G' דרך קודקוד זה (כי הוא נגיש ל s) בסתירה לכך ש t אינו נגיש מ s ב G'.

מכאן שאנו יכולים להיות בטוחים שבדיוק אחד מהקודקודים של e הוא נגיש ל s ובדיוק אחד מהקודקודים של e אינו נגיש ל s ולכן חישוב המסלול הוא נכון במקרה זה.

במקרה שבו $d(s,t) < \infty$ אם אלגוריתם דיקסטרה חישב שאחד מהמרחקים $d(s,u) \ = \ \infty$ או $d(s,v) \ = \infty$ המשמעות היא שאין מסלול מ s ל t שעובר דרך e ב G'. הסיבה לכך היא שאם היה כזה מסלול אז אפשר היה להגיע מ s ל u או v (הקודקוד שאליו חישבנו מרחק אינסופי) דרך t באמצעות המסלול שקיים ב G בין t לבין u או v בסתירה לכך שהמרחק שחושב הוא אינסופי (כלומר יש נתק בגרף שלא מאפשר מסלול)

האיור הבא מדגים:

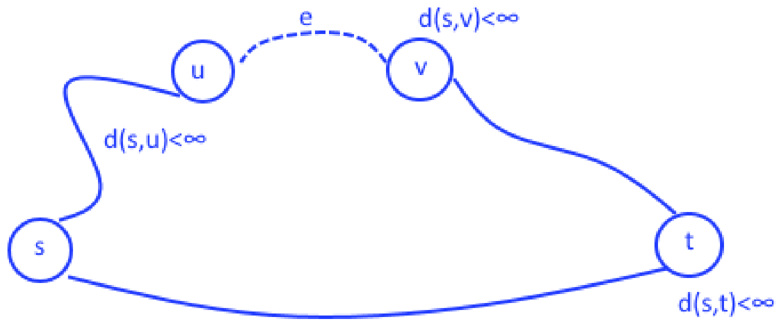


הערה: אין אפשרות ששני המרחקים $d(s,u)$ ו $d(s,v)$ יהיו שווים לאינסוף מכיוון שאז המשמעות היא ששני הקודקודים של e מנותקים מקודקוד s בסתירה לכך שרק הקשת שמחברת אותם הוסרה והגרף היה קשיר לפני כן.

לכן - במקרה זה האלגוריתם מחזיר את המסלול $p(s,t)$ שחושב ע"י אלגוריתם דיקסטרה מכיוון שזהו המסלול הקצר ביותר האפשרי בין s ל t ב G.

אם אלגוריתם דיקסטרה חישב ששני המרחקים $d(s,u) < \infty$ וגם $d(s,v) < \infty$ המשמעות היא שיש מסלול מ s ל t שעובר דרך e (כי G קשיר) וגם מסלול מ s ל t שלא עובר דרך e (כי יש מסלול מ s ל t שלא עובר דרך e ב G' ולכן גם ב G).

האיור הבא מדגים:



במקרה זה חייב להיות מסלול ב G' בין v ל t כי יש מסלול בין t לבין s ויש מסלול בין s לבין v. קל לראות שהמסלול הקצר ביותר ב G בין s לבין t חייב להיות הקצר מבין שני המסלולים האפשריים האלה (מסלול דרך e ומסלול שלא עובר דרך e).

כדי לבחור את המסלול הקצר ביותר - האלגוריתם משווה את המסלול שחושב כבר בין s לבין t שלא עובר דרך e למסלול בין s לבין t שעובר דרך e. האופן שבו האלגוריתם מחשב את המסלול שעובר דרך e מבטיח קבלת המסלול הקצר ביותר בין s ל t שעובר דרך e.

<u>סיבוכיות האלגוריתם:</u>

סיבוכיות האלגוריתם נגזרת מהצורך להריץ את אלגוריתם דיקסטרה פעמיים בעוד כל שאר הפעולות הם בעלי סיבוכיות קבועה, לכן בחשבון הסופי נקבל שסיבוכיות האלגוריתם היא $O(m \cdot logn)$.

שאלה 3

הרעיון של האלגוריתם הוא להמיר את הבעיה של מציאת מסלול עם סכום מינימלי של דרגות קודקודים בין s ל t בגרף G לבעיה של מציאת מסלול בגרף G' שלו יש אותה קבוצת קודקודים כמו ב G ועם סכום משקלי קשתות מינימלי אותה ניתן לפתור באמצעות אלגוריתם דיקסטרה.

הגרף G' ייבנה באופן כזה שמשקל הקשתות לאורך מסלול כלשהו ב G' יהיה מותאם לסכום דרגות הקודקודים באותו מסלול על גרף G. כלומר - אם $p1(s,t)$ ו $p2(s,t)$ הם שני מסלולים שונים בין קודקוד s לקודקוד t ב G ומתקיים $cost(p1) > cost(p2)$ כאשר המחיר של המסלול נקבע לפי סכום דרגות הקודקודים במסלול, אז בגרף G' נקבל שהמחיר של אותם מסלולים לפי סכום משקלי הקשתות בין הקודקודים במסלול שומר על האי-שיוויון, כלומר: $weight(p1) > weight(p2)$.

באופן הזה נוכל למצוא מסלול בין s לבין t שמשקלו קטן ביותר בגרף G' ומהעובדה שהאי-שיוויון נשמר יהיה ברור שזהו המסלול עם סכום דרגות הקודקודים הקטן ביותר בין s לבין t. את המסלול בין s לבין t בגרף G' נמצא באמצעות אלגוריתם דיקסטרה כמובן.

<u>אלגוריתם</u>

נבנה גרף G' שלו אותה קבוצת קודקודים ואותה קבוצת קשתות כמו הגרף G. עבור כל קשת בגרף G' נקבע משקל שהוא סכום הדרגות של שני הקודקודים אותם מחברת הקשת.

נריץ אלגוריתם דיקסטרה על הגרף G' והקודקוד s בתור קודקוד ההתחלה. עבור כל קודקוד אליו מגיע האלגוריתם נשמור הן את המרחק שלו מ s והן את הקודקוד ממנו הגענו אליו. באופן זה נוכל תמיד לשחזר את המסלול הקצר ביותר בין t ל s כאשר האלגוריתם מחשב את המרחק $d(s,t)$ בין s לבין t.

נעצור את האלגוריתם כאשר חושב המרחק $d(s,t)$. המסלול בין s לבין t כך שסכום הדרגות של הקודקודים במסלול הוא הקטן ביותר הוא המסלול הקצר ביותר שחושב ע"י אלגוריתם דיקסטרה.

<u>הוכחה</u>

כדי להוכיח את נכונות האלגוריתם מספיק להוכיח את נכונות הטענה הבאה:
טענה: נבחר באופן שרירותי שני קודקודים שונים u ו v ב G ונסמן ב $p1(u,v)$ ו $p2(u,v)$ שני מסלולים בין הקודקודים u ו v.
נסמן ב $cost(p)$ את סכום דרגות הקודקודים במסלול כלשהו p ב G. נסמן ב $weight(p)$ את סכום המשקלות של הקשתות באותו מסלול p על הגרף G'.
אז בהכרח מתקיים ש $cost(p1) > cost(p2)$ אם ורק אם מתקיים $weight(p1) > weight(p2)$.

הוכחה: נתבונן במסלול כלשהו p על הגרף G' ובאותו מסלול p על הגרף G.

אם נסמן את הקודקודים במסלול p כ $v_1, v_2, \dots v_k$ ואת הדרגה של כל קודקוד v נסמן כ $deg(v)$ אז נקבל שמתקיים:

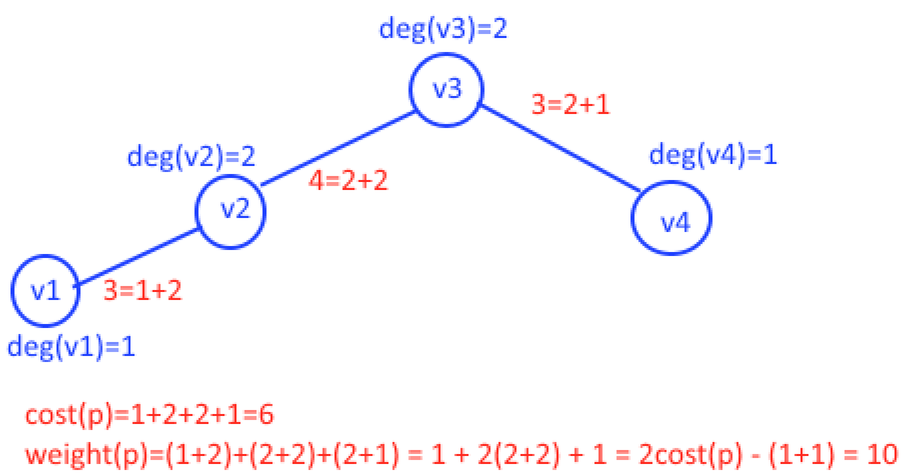
$$cost(p) = \sum_{j=1}^{k} deg(v_j)$$

בגרף G

וגם מתקיים:

$$weight(p) = deg(v_1) + \left(2\sum_{j=2}^{k-1} deg(v_j)\right) + deg(v_k)$$

האיור הבא מדגים מדוע מתקיים השיוויון האחרון (בדוגמא המסלול מחושב בין v1 לבין v4):



קל לראות שבכל מסלול ב G' סכום משקלי הקשתות כולל את משקל כל קודקוד פנימי במסלול פעמיים ומכאן נובעת נוסחת המשקל.

ולכן:

$$weight(p) = cost(p) + \sum_{j=2}^{k-1} deg(v_j) \;=\; 2cost(p) \;-\; deg(v_1) \;-\; deg(v_k)$$

בגרף G'

מכך שמתקיים $cost(p1) > cost(p2)$ נובע שמתקיים:

$$2cost(p1) - deg(v) - deg(u) \;>\; 2cost(p2) - deg(v) - deg(u)$$

(שים לב ששני המסלולים מחברים את אותם קודקודים ולכן אפשר להחסיר משני האגפים את סכום דרגות קודקודי הקצה)

ולכן בהכרח נובע ש:

$$weight(p1) > weight(p2)$$

הכיוון השני נובע בדיוק באותו האופן רק בכיוון ההפוך.

מכך נובעת נכונות האלגוריתם מכיוון שאלגוריתם דיקסטרה ימצא עבורנו את המסלול p בין s ל t ב G' כך ש $weight(p)$ הוא מינימלי ולכן לא קיים מסלול אחר ב G שעבורו סכום הדרגות הוא קטן יותר מזה של המסלול שחושב. אם היה קיים מסלול כזה b אז היה מתקיים $cost(b) < cost(p)$ ומכאן בהכרח היה מתקיים לפי הטענה לעיל ש $weight(b) < weight(p)$ בסתירה לכך שאלגוריתם דיקסטרה מצא את המסלול בעל המשקל המינימלי.

<u>סיבוכיות האלגוריתם:</u>

סיבוכיות האלגוריתם נגזרת מהסיבוכיות של שני חלקיו (m הוא מספר הקשתות ב G ו n הוא מספר הקודקודים ב G):

1. יצירת הגרף G' דורשת $O(m+n)$ פעולות, למעשה אפשר להסתפק ב $O(m)$ פעולות אם מבצעים את השינוי על הגרף G עצמו אבל בחשבון הסופי זה לא משנה בגלל הסיבוכיות של אלגוריתם דייקסטרה.
2. הרצת אלגוריתם דייקסטרה על G' דורשת $O(m \cdot logn)$ פעולות.

לכן הסיבוכיות של האלגוריתם היא $O(m \cdot logn)$

שאלה 4

א.

מכך שקשת $e=(u,v)$ נכללת בגרף H רק אם היא יוצרת מסלול בין u ל v שלא היה קיים קודם או אם מתקיים $3l_e < d_{uv}$ נובע שאם הקשת e לא נכללת בגרף H אז קיים מסלול ב H שאורכו לכל היותר $3l_e$ שמחבר את u ל v.

כעת נתבונן במסלול הקצר ביותר בין שני קודקודים u ו v בגרף G. בין כל שני קודקודים עוקבים במסלול קיימת קשת ב G שקיימת ב H או אינה קיימת ב H. אם הקשת אינה קיימת ב H אז אפשר ל "תקן" את המסלול ב H כך ששני הקודקודים יחוברו באמצעות מסלול ב H שאורכו לכל היותר 3 פעמים המשקל של הקשת בין שני הקודקודים האלו ב G. אם הקשת אכן קיימת ב H אז אין צורך לתקן ואפשר להמשיך לקשת הבאה במסלול. בסיום התהליך נקבל מסלול ב H בין u ל v שאורכו חסום מלמעלה ע"י 3 פעמים סכום משקלי הקשתות במסלול המקורי ב G והוא מחבר את u ו v.

מכיוון שאורכו של המסלול הקצר ביותר ב H קטן או שווה לזה של המסלול ה "מתוקן" שיצרנו הרי שגם אורכו של המסלול הקצר ביותר ב H חסום מלמעלה ע"י 3 פעמים סכום משקלי הקשתות במסלול המקורי הקצר ביותר ב G.

מ.ש.ל.

ב.

לצערי לא הצלחתי לפתור את החלק הזה של השאלה.
רק כדי לקבל חוות דעת על הכיוון - במהלך הניסיונות לפתור את השאלה גיליתי שב H אין מעגל עם פחות מ 5 קשתות וניסיתי להשתמש בעובדה הזאת כדי לתת חסם מקסימלי על דרגת כל קודקוד ב H ואז להשתמש בנוסחה שסכום דרגות הקודקודים מחולק בשניים נותן את מספר הקשתות כדי לחסום את מספר הקשתות ב H כך שהגבול ישאף לאפס (ברור שמספר הקשתות צריך להיות או לינארי במספר הקודקודים ב H או בין לינארי לריבועי לא כולל). נתקעתי בשלב של ניסיון למצוא ולחשב חסם מקסימלי על דרגות הקודקודים ב H..

שאלה 5

יהי עץ בינרי לחלוטין T בעל n עלים. נבנה סדרת שכיחויות $f_1, f_2, \ldots, f_n$ כך שעץ הופמן של סדרה זו הוא T.

נסמן ב d את העומק של העץ T (כלומר - אורכו של המסלול הארוך ביותר בין שורש העץ לעלה כלשהו ב T).

עבור כל העלים שנמצאים ברמת עומק p ניתן שכיחות $\frac{1}{2^p}$

הערה: אלגוריתם הופמן לא מתייחס לערכים האבסולוטיים של השכיחויות - רק ליחס ביניהם ולכן אין בעיה לבחור סדרת שכיחויות שהסכום שלה אינו 1. תמיד אפשר לנרמל את סדרת השכיחויות בשלב מאוחר יותר אבל לצורך ההוכחה נוח לעבוד עם הסדרה לעיל.

טענה: יהי עץ T וסדרת שכיחויות F שחושבה לפי הכלל לעיל עבור T. אז עץ הופמן H שמחושב לפי סדרת השכיחויות לעיל זהה לעץ T.

הוכחה: נוכיח באינדוקציה על עומק העץ T.

עבור עומק d=1 נקבל בקלות שהעץ H זהה ל T (לשניהם יש שני עלים בדיוק)

נניח שהטענה נכונה עבור כל העצים בעומק d ונוכיח שהיא נכונה עבור כל העצים בעומק d+1.

יהי T עץ בינארי לחלוטין בעומק d+1. כעת נתבונן ב k העלים שנמצאים בעומק d+1 ב T. עלים אלו ממופים ל k איברים זהים בסדרה F שהם בעלי הערך הנמוך ביותר ב F. בנוסף - k הוא מספר זוגי כי זוהי הרמה העמוקה ביותר של T.

נתבונן בתוצאה של הרצת אלגוריתם הופמן על סדרת השכיחויות F לאחר $\frac{k}{2}$המיזוגים הראשונים. נקבל סדרת שכיחויות *F שבה k השכיחויות הנמוכות ביותר ב F הוחלפו ב $\frac{k}{2}$שכיחויות ב *F שכל אחת מהן היא כפולה ב 2 של השכיחות הנמוכה ביותר ב F.

כעת נתבונן בעץ *T שמתקבל מהעץ T ע"י מיזוג כל זוג עלים בעומק d+1 לעלה אחד בעומק d. לפי אופן בניית סדרת השכיחויות עבור *T נקבל שסדרת השכיחויות שמתאימה ל *T זהה לסדרת השכיחויות *F. ברור גם שעומקו של העץ *T הוא d כי אין יותר עלים בעומק d+1.

לכן לפי הנחת האינדוקציה נקבל לאחר הרצת אלגוריתם הופמן על *F (שנוצרה מהעץ *T בעומק d) עץ הופמן *H שהוא זהה לעץ *T. אבל הרצת אלגוריתם הופמן על *F היא בדיוק המשך ביצוע אלגוריתם הופמן על F לאחר ביצוע $\frac{k}{2}$המיזוגים הראשונים ולכן לאחר קבלת *H (שהוא שווה ל *T) האלגוריתם ממשיך לפצל $\frac{k}{2}$עלים בעומק d ב *T שמוזגו במקור מעלים בעומק d+1 ב T בחזרה לעלים בעומק d+1 ומקבלים בדיוק את T.