<u>שאלה:</u>

הצע אלגוריתם המדפיס את כל צלעות הגרף $G=(V, E)$ בסדר כלשהו, בזמן $O(m + n)$.

<u>פתרון:</u>

<u>תאור האלגוריתם:</u>

נריץ על הגרף את אלגוריתם BFS^* המתואר להלן:

<u>BFS(G, s)</u>*

```
1.   for each vertex  u ∈ V
2.        color[u] ← WHITE
3.        d[u] ← ∞
4.        π[u] ← NIL
5.   color[s] ← GRAY
6.   d[s] ← 0
7.   Q ← {s}
8.   while  ( Q ≠ ϕ )
9.        u ← head[Q]
10.       for each v ∈ Adj[u]
11.            if  ( color[v] = WHITE )
12.                 color[v] ← GRAY
13.                 d[v] ← d[u] + 1
14.                 π[v] ← u
15.                 Enqueue(Q, v)
16.            if  ( color[v] ≠ BLACK )
17.                 PrintEdge (u, v)
18.       Dequeue(Q)
19.       color[u] ← BLACK
```

<u>הערה:</u>

נריץ את BFS^* על כל מרכיבי הקשירות.

<u>סיבוכיות:</u>

ברור שהסיבוכיות זהה לזו של BFS "הרגיל": $O(m + n)$.

<u>נכונות:</u>

סריקת BFS שעוברת על כל מרכיבי הקשירות, מטפלת בכל קדקודי הגרף. הכוונה ב-"טיפול" בקודקוד מסוים – מעבר על כל השכנים שלו.
אבל ההגעה לשכן מתבצעת דרך צלע, ולכן בסריקה כולה עוברים על כל הצלעות שקיימים בגרף.
לכן, אם נדאג להדפיס את הצלע ברגע שעוברים "דרכה" (כפי שאכן קורה בשורות 16-17) , אזי ברור שנדפיס את כל צלעות הגרף.

■

<u>שאלה:</u>

הצע אלגוריתם הבודק האם גרף $G=(V, E)$ הוא קשיר, בזמן $O(m + n)$.

<u>פתרון:</u>

<u>תאור האלגוריתם:</u>

1. נריץ על הגרף BFS מקודקוד אקראי.
2. נבדוק את קבוצת הקדקודים:

אם, לאחר ריצת BFS, קיימים קדקודים לבנים – אזי הגרף אינו קשיר.
אחרת – הגרף קשיר.

<u>סיבוכיות:</u>

1. BFS: $O(m + n)$.
2. בדיקת קבוצת הקדקודים: $O(n)$.

סה"כ: $O(m + n)$.

<u>נכונות:</u>

הנכונות נובעת ישירות מנכונות BFS.

■

שאלה:

הצע אלגוריתם הבודק האם גרף לא מכוון $G=(V, E)$ הוא יער, כלומר - חסר מעגלים, בזמן $O(n)$.

פתרון:

תאור האלגוריתם:

1. נריץ את BFS^*, המתואר להלן:

$BFS^*(G, s)$

```
for each vertex  u ∈ V
        color[u] ← WHITE
        d[u] ← ∞
        π[u] ← NIL
color[s] ← GRAY
d[s] ← 0
Q ← {s}
while  ( Q ≠ ϕ )
        u ← head[Q]
        for each v ∈ Adj[u]
                if  ( color[v] = WHITE )
                        color[v] ← GRAY
                        d[v] ← d[u] + 1
                        π[v] ← u
                        Enqueue(Q, v)
                else  if  π[u] ≠v
                        return FALSE
        Dequeue(Q)
        color[u] ← BLACK
return TRUE
```

2. אם לא קיבלנו תשובה שלילית, אזי נחזור לסעיף 1, על כל מרכיבי הקשירות האחרים, אם קיימים.

סיבוכיות:

<u>א.</u> ננתח עבור גרף לא שאינו יער:
אם הגרף אינו יער, אזי קיים בו מעגל. אם מצאנו מעגל – נעצור את האלגוריתם.
מעגל הוא מסלול שחוזר על אחד מקדקודיו יותר מפעם אחת. $\Leftarrow$ $|E| \leq |V|$ $\Leftarrow$ הסיבוכיות היא $O(n)$.

<u>ב.</u> ננתח עבור גרף שהוא יער:
יער הוא קבוצה של עצים. ידוע שבעץ מתקיים: $|E| = |V| - 1$. $\Leftarrow$ הסיבוכיות היא $O(n)$.

סה"כ: $O(n)$.

נכונות:

ברור שגילויו של קודקוד אפור מעיד על סגירתו של מעגל, ולכן ברור שהגרף אינו יער.
אם האלגוריתם סיים ללא תשובה שלילית, הרי שעבר על כל מרכיבי הקשירות, ולא מצא מעגל. כלומר – בסריקה של כל הקדקודים, דרך כל הקשתות, לא נמצא מעגל. אבל זהו בדיוק יער.

■

<u>שאלה:</u>

הצע אלגוריתם הבודק האם גרף לא מכוון $G=(V, E)$ הוא עץ, כלומר – קשיר וחסר מעגלים, בזמן $O(n)$.

<u>פתרון:</u>

<u>תאור האלגוריתם:</u>

1. נריץ את BFS^*, המתואר להלן:

<u>BFS(G, s)</u>*

```
1.   for each vertex  u ∈ V
2.       color[u] ← WHITE
3.       d[u] ← ∞
4.       π[u] ← NIL
5.   color[s] ← GRAY
6.   d[s] ← 0
7.   Q ← {s}
8.   while  ( Q ≠ ϕ )
9.       u ← head[Q]
10.      for each v ∈ Adj[u]
11.          if  ( color[v] = WHITE )
12.              color[v] ← GRAY
13.              d[v] ← d[u] + 1
14.              π[v] ← u
15.              Enqueue(Q, v)
16.          else  if  π[u] ≠v
17.              return FALSE
18.      Dequeue(Q)
19.      color[u] ← BLACK
20.  return TRUE
```

2. אם קיבלנו תשובה שלילית – הגרף אינו עץ, וסיימנו.
אחרת – נעבור על קבוצת הקדקודים:
אם קיימים קדקודים לבנים – אזי הגרף אינו עץ.
אחרת – הגרף הוא עץ.

<u>סיבוכיות:</u>

<u>א.</u> ננתח עבור גרף לא שאינו עץ:
אם הגרף אינו עץ, אזי קיים בו מעגל. אם מצאנו מעגל – נעצור את האלגוריתם.
מעגל הוא מסלול שחוזר על אחד מקדקודיו יותר מפעם אחת. $\Leftarrow$ $|E| \le |V|$ $\Leftarrow$ הסיבוכיות היא $O(n)$.

<u>ב.</u> ננתח עבור גרף שהוא עץ:
ידוע שבעץ מתקיים: $|E| = |V| - 1$. $\Leftarrow$ הסיבוכיות היא $O(n)$.

סה"כ: $O(n)$.

<u>נכונות:</u>

נוכיח שהאלגוריתם מחזיר תשובה חיובית אם"ם הגרף הוא קשיר וחסר מעגלים:

<u>קשיר:</u>

<u>טענה:</u>

אם הגרף אינו קשיר – אזי בסיום סריקת BFS יחידה יהיו קדקודים לבנים.

<u>הוכחה:</u>

נובעת ישירות מנכונות BFS.

<u>חסר מעגלים:</u>

<u>טענה:</u>

אם במהלך סריקת BFS מגלים קודקוד אפור – אזי קודקוד זה (עם הקשת שגילתה אותו) סוגר מעגל.

<u>הוכחה:</u>

נובעת ישירות מנכונות BFS.

לכן – אם בוצעה סריקת BFS, והגרף הנסרק התגלה כחסר מעגלים וכקשיר – אזי ברור שהוא עץ.
אם הגרף בעל מעגלים – אזי האלגוריתם מחזיר תשובה שלילית (שורות 17-16 ב-BFS).
אם הגרף אינו קשיר – אזי האלגוריתם מחזיר תשובה שלילית (סעיף 2 של האלגוריתם).

■

<u>שאלה:</u>

נתון גרף $G=(V, E)$ מכוון. הצע אלגוריתם הקובע האם הגרף הוא עץ מכוון עם שורש.

<u>פתרון:</u>

<u>תאור האלגוריתם:</u>

1. נבדוק את קבוצת הקדקודים, למציאת קדקודים בעלי דרגת כניסה (d_{in}) השווה ל-0:
   אם לא קיים קודקוד כזה – נודיע שהגרף אינו עץ כנדרש, ונסיים.
   אם קיימים יותר מקודקוד אחד – נודיע שהגרף אינו עץ כנדרש, ונסיים.
   אם קיים קודקוד יחיד כזה – נסמנו v.
2. נריץ על G, החל מקודקוד v, את אלגוריתם BFS^*, המתואר להלן:

<u>$BFS^*(G, s)$</u>

```
for each vertex  u∈V
      color[u] ← WHITE
      d[u] ← ∞
      π[u] ← NIL
color[s] ← GRAY
d[s] ← 0
Q ← {s}
while  ( Q ≠ ϕ )
      u ← head[Q]
      for each v∈Adj[u]
            if  ( color[v] = WHITE )
                  color[v] ← GRAY
                  d[v] ← d[u] + 1
                  π[v] ← u
                  Enqueue(Q, v)
            Else  if  π[u] ≠v
                  return FALSE
      Dequeue(Q)
      color[u] ← BLACK
return TRUE
```

   אם נקבל תשובה שלילית – נודיע שהגרף אינו עץ כנדרש, ונסיים.
3. נרוץ על קבוצת הקדקודים, ונחפש קדקודים לבנים.
   אם נמצא קדקודים לבנים – הגרף אינו עץ כנדרש.
   אחרת – הגרף הוא עץ מכוון עם שרש v.

<u>סיבוכיות:</u>

1. בדיקת דרגות הכניסה (d_{in}) של הקדקודים: $O(m + n)$.
2. BFS^*: $O(m + n)$.
3. מעבר על רשימת הקדקודים: $O(n)$.

סה"כ: $O(m + n)$.

<u>נכונות:</u>

<u>טענה:</u>

אם הגרף הוא עץ מכוון עם שרש, אזי יש קודקוד יחיד בעץ שדרגת הכניסה שלו שווה אפס, כלומר: $d_{in}=0$.

<u>הוכחה:</u>

ברור שעבור השרש מתקיים $d_{in}=0$.

לכל קודקוד u אחר בעץ קיים מסלול מהשורש אליו, ולכן קיימת קשת הנכנסת לקודקוד u, ולפיכך, לכל קודקוד u אחר, $d_{in}>0$.

<u>טענה:</u>

גרף מכוון שבו:

א. קיים קודקוד יחיד עם $d_{in}=0$,
ב. אין מעגלים,
ג. בין שני קדקודים קיים מסלול אחד לכל היותר,
ד. ניתן להגיע מ-v לשאר הקדקודים

הוא עץ מכוון עם שרש v.

<u>הוכחה:</u>

ישירות מהגדרה של עץ מכוון עם שורש.

■

<u>שאלה:</u>

<u>שאלה:</u>

נתון גרף $G=(V, E)$ מכוון חסר מעגלים. מצא את המסילה הארוכה ביותר בזמן $O(m + n)$.

<u>פתרון:</u>

<u>תאור האלגוריתם:</u>

1. נריץ TopSort על G. נסמן את הגרף הממוין 'G.
2. נריץ על 'G את השגרה SetLongestDistance, כמתואר להלן:

<u>SetLongestDistance(G')</u>

```
for i ← 1 to n
    δ[vi] ← 0
for i ← 1 to n
    for each vertex  u∈Adj[vi]
        if  ( δ[u] < δ[vi] + 1 )
            δ[u] ← δ[vi] + 1
            π[u] ← vi
```

3. נרוץ על קבוצת הקדקודים, ונמצא את הקדקוד עם ה-δ הגדול ביותר.
ה-δ שנמצא הוא האורך המקסימלי, ואת המסלול עצמו נשחזר על-ידי מעבר על שדה האב – ה-π.

<u>סיבוכיות:</u>

1. TopSort: $O(m + n)$.
2. SetLongestDistance: $O(m + n)$.
3. מציאת ה-δ המקסימלי: $O(n)$.
מציאת המסלול על-ידי שדה האב – ה-π: $O(m)$.

סה"כ: $O(m + n)$.

<u>נכונות:</u>

ניתן להפעיל TopSort, משום שהגרף חסר מעגלים (נתון).
לאחר המיון הטופולוגי – יהיו קשתות המכוונות לכיוון אחד בלבד.
לכן – לאחר שנסיים לטפל בקודקוד מסוים, לא נשוב אליו יותר.

על בסיס עובדה זו בנויה השגרה SetLongestDistance. אופן עדכון שדה ה-δ (שמציין את אורך המסילה), ושדה ה-π (שמציין את האב) ברור לחלוטין, ומכאן גם נובעת ישירות נכונותו.

■

שאלה:

נתון גרף לא מכוון $G=(V, E)$, סופי וקשיר. הצע אלגוריתם יעיל הבודק האם G הוא גרף דו-צדדי.
(לחלק את V ל-2 קבוצות V_1 ו-V_2 כך שאין קשת המחברת זוג קדקודים באותה הקבוצה)
(לצבוע את קדקודי הגרף ב-2 צבעים שונים, כך שאין קשת המחברת 2 קדקודים באותו הצבע)

פתרון:

תאור האלגוריתם:
נריץ את אלגוריתם Bi-Sided(G, s) המתואר להלן:

Bi-Sided(G, s)

```
1.    for each  v∈V
2.        color(u) ← WHITE
3.    color[s] ← BLUE
4.    Q ← {s}
5.    while  ( Q ≠ ϕ )
6.        u ← head[Q]
7.        for each v∈Adj[u]
8.            if  ( color[v] = color[u] )
9.                return  FALSE
10.           if  ( color[v] = WHITE )
11.               π[v] ← π[u]
12.               if  ( color[u] = BLUE )
13.                   color[v] ← RED
14.               else
15.                   color[v] ← BLUE
16.           Enqueue(Q, v)
17.       Dequeue(Q)
18.   return    TRUE
```

סיבוכיות:

2-1	$O(n)$.
4-3	$O(1)$.
17-5	$O(m)$.
18	$O(1)$.
סה"כ:	$O(m + n)$.

נכונות:
<u>טענה:</u>
גרף הוא דו-צדדי אס"ם ניתן לסמן את כל קדקודי הגרף בצבעים כחול ואדום, כאשר צבעו של כל קודקוד שונה משל שכניו.
<u>הוכחה:</u>

<u>כיוון $\Leftarrow$:</u> נחלק את קדקודי הגרף ל-2 קבוצות משלימות, על-פי הצבע.
כיוון שכל קשת בגרף מחברת בין 2 קדקודים, ועל-פי ההנחה, צבעם שונה – הרי שכל קשת תעבור בין 2 קבוצות הקדקודים.
וזהו גרף דו-צדדי.

<u>כיוון $\Rightarrow$:</u> נניח שהגענו לקודקוד u מקודקוד v, וצבעם זהה.
ברור שקודקוד u הוא צאצא של קודקוד v. $\Leftarrow$ קיים מעגל באורך אי-זוגי בגרף.
צבעי הקדקודים על המסלול בין u ו-v יהיו כחול-אדום, לסירוגין, עד שנגיע לקודקוד v, ששיך לקבוצה הכחולה.
מקודקוד v קיימת קשת לקודקוד u. אבל קודקוד u שייך גם הוא לקבוצה הכחולה.
לכן לא ייתכן שהגרף הוא דו-צדדי.

■

__שאלה:__

נתון גרף $G=(V, E)$, עם משקלים על הקדקודים, ושני קדקודים $s, t \in V$. תאר אלגוריתם המוצא את המסלול מ-s ל-t, כך שהקודקוד בעלת הערך המקסימלי, יהיה הקטן ביותר מכל הקדקודים המקסימליים במסלולים מ-s ל-t, בזמן $O(m \log n)$.

__פתרון:__

__תאור האלגוריתם:__

נריץ מעין Dijkstra על הקדקודים:

נגדיר שדה max שיכיל את הקודקוד המקסימלי במסלול עד לאותו הקודקוד.

ערך זה יעודכן פעם אחת בלבד. כלומר – כאשר נגיע לקודקוד v כלשהו, כאשר אביו הוא u – נבדוק האם max[v] עודכן. אם לא – נשווה בין max[v] לבין max[u]. נשמור את הגדול מביניהם ב-max[v]. כלומר: $max[v] = max\{ max[v], max[u] \}$, ויותר לא ישתנה.

בכל התקדמות נבחר את הקודקוד בעל הערך המקסימלי המינימלי.

עדכון האבות יהיה לפי הראשון שהגיע לקודקוד, כלומר אם הגענו בפעם הראשונה ל-u מ-v, אזי האב של u יהיה v.

__סיבוכיות:__

Dijkstra: $O(m \log n)$.

__נכונות:__

הנכונות נובעת ישירות מהוכחת הנכונות של Dijkstra.

■

<u>שאלה:</u>

נתון גרף לא מכוון קשיר $G=(V, E)$. תאר אלגוריתם המפחית את מספר הקשתות בגרף, תוך שמירה על קשירותו בסיבוכיות $O(m + n)$.

<u>פתרון:</u>

<u>תאור האלגוריתם:</u>

1. נבנה גרף G' באופן הבא:
   נריץ את אלגוריתם BFS מקודקוד כלשהו על הגרף, כאשר את הצלעות שנעבור דרכן – נוסיף לקבוצת הצלעות E' של G'.
2. הקשתות שניתן להוריד מהגרף G ללא פגיעה בקשירותו הם הקשתות המתקבלות על-ידי: $E \setminus E'$.

<u>סיבוכיות:</u>

1. BFS: $O(m + n)$.
2. מעבר על הקשתות: $O(m)$.

סה"כ: $O(m + n)$.

<u>נכונות:</u>

אם הגרף G היה קשיר, אז BFS הגיע לכל הקדקודים של הגרף.
על-פי ריצת ה-BFS, הגרף G' שנוצר הוא למעשה עץ.
לכן – אם נזרוק את כל הקשתות מהגרף G שאינן נמצאות בגרף G', נקבל גרף קשיר (ברור שעץ הוא קשיר...)

■

שאלה:

נתון גרף פשוט $G=(V, E)$, שאינו בהכרח קשיר. זוג צמתים $u, v\in V$, $u\neq v$, נקרא *קשיר* אם קיים מסלול ב-G בין u ל-v. תאר אלגוריתם יעיל ככל הניתן, המחשב את מספר הזוגות ה*קשירים* ב-G.

פתרון:

תאור האלגוריתם:

1. נריץ על G את אלגוריתם DFS, למציאת רכיבי קשירות. נסמנם: $T_1, T_2, \dots, T_k$.
2. עבור כל רכיב קשירות T_i נסמן את עוצמת קבוצת הקדקודים שלו (כלומר – את $|V_i|$), כ-n_i.
3. לכל רכיב קשירות T_i, נחשב חישוב קומבינטורי המתבסס על כך שכל קודקוד קשור לכל אחד אחר באותו רכיב. כלומר – נניח שברכיב הקשירות ה-i קיימים n_i קדקודים, אזי מספר הזוגות הקשירים ברכיב קשירות זה הוא: $\binom{n_i}{2}$
נסמן חישוב זה כ-c_i.
4. נסכום את כל ה-c_i-ים, $i=1, \dots, k$.

סיבוכיות:

1. DFS: $O(m + n)$.
2. ברור שסכום קבוצות הקדקודים של ה-T_i-ים, $i=1, \dots, k$, שווה ל-$|V|$, ולכן: $O(n)$.
3. לכל היותר קיימים $|V|$ רכיבים קשירים, והחישוב הקומבינטורי מתבצע ב-$O(1)$, ולכן: $O(n)$.
4. לכל היותר קיימים $|V|$ רכיבים קשירים, ולכן: $O(n)$.

סה"כ: $O(m + n)$.

נכונות:

סעיפים 1 ו-2 נובעים ישירות מנכונות DFS.
נכונות החישוב הקומבינטורי ברורה, ומושתתת על חוקים קומבינטוריים.
ומכאן נובעת נכונות האלגוריתם כולו.

■

<u>שאלה:</u>

גרף מכוון $G=(V, E)$ נקרא קשיר למחצה אם עבור כל זוגות הקדקודים $u, v\in V$ מתקיים: $u \rightsquigarrow v$, או $v \rightsquigarrow u$.

הצע אלגוריתם יעיל הקובע האם G קשיר למחצה.

<u>פתרון:</u>

<u>תאור האלגוריתם:</u>

1. נריץ SCC למציאת הרכיבים הקשירים היטב.
2. נצור גרף 'G של הרכיבים (ללא מעגלים), כך שכל רכיב ב-G הופך לקודקוד ב-'G.
3. נריץ TopSort על 'G, ונכניס את הקדקודים למחסנית לפי סדר עולה של f[u].
4. נתחיל מהקודקוד הראשון, ונבדוק האם קיימת קשת ממנו לקודקוד הבא.
   אם קיימת קשת כזו – נעבור אל הקודקוד הבא, ונמשיך באותו האופן עבור הקדקודים הבאים.
   אם קיימת קשת בין כל קודקוד i לקודקוד i+1, אזי הגרף קשיר למחצה.
   אחרת – הגרף אינו קשיר למחצה.

<u>סיבוכיות:</u>

1. SCC: $O(m + n)$.
2. בניית 'G: $O(m + n)$.
3. TopSort: $O(m + n)$.
4. בדיקת קשתות וקדקודים: $O(m + n)$.

סה"כ: $O(m + n)$.

<u>נכונות:</u>

בגרף 'G שבנינו בסעיף 2, כל קודקוד מהווה רכיב קשיר היטב, כי SCC מאחד קדקודים שיש ביניהם מסלול לרכיב קשיר היטב (בפרט – כל רכיב קשיר היטב הוא רכיב קשיר למחצה).

נותר לבדוק האם עבור כל זוג של רכיבי קשירות x, y מתקיים: $x \rightsquigarrow y$, או $y \rightsquigarrow x$.

אבל זה נובע ישירות TopSort וביצוע הבדיקות המתוארות בסעיף 4.

■

<u>שאלה:</u>

נתון גרף מכוון $G=(V, E)$ שאינו בהכרח קשיר. הצע אלגוריתם יעיל המדפיס את רשימת כל הצמתים השייכים למעגל כלשהו.

<u>פתרון:</u>

<u>תאור האלגוריתם:</u>

1. נריץ SCC על G.
2. נעבור על רכיבי הקשירות, בחפשנו אחר רכיב קשירות המונה יותר מאיבר יחיד (כלומר: $|c[i]| > 1$, עבור $1 \le i \le COMP$ כלשהו).
3. אם מצאנו רכיב קשירות כזה – נדפיס את כל איברי $c[i]$.
   אחרת – נודיע כי לא קיימים מעגלים בגרף.

<u>סיבוכיות:</u>

1. SCC: $O(m + n)$.
2. מקסימום רביכי קשירות: $O(n)$.
3. מקסימום הדפסת כל הקדקודים בגרף: $O(n)$.

סה"כ: $O(m + n)$.

<u>נכונות:</u>

הנכונות נובעת ישירות מנכונות SCC. (שכידוע מוצאת מעגלים...)

■

שאלה:

נתון גרף מכוון חסר מעגלים $G=(V, E)$, ושני קדקודים $s, t\in V$. תאר אלגוריתם לינארי המחשב את מספר המסילות השונות מ-s ל-t ב-G.

פתרון:

תאור האלגוריתם:

1. נריץ TopSort על G. נסמן את הגרף הממויין G'
2. נבנה גרף $G^T=(V, E^T)$ ל-G', כאשר:

$$E^T = \{ (u, v) \mid u, v\in V, (v, u)\in E) \}$$

3. נפעיל את השגרה CountPaths על G^T, מ-t, כמתואר להלן:

CountPaths(G^T, t, s)

```
1.   for i ← s to t
2.        n[vi] ← 0
3.   n[t] ← 1
4.   for i ← t downto s
5.        for each vertex  u∈Adj[vi]
6.             n[u] ← n[u] + n[vi]
```

4. מספר המסילות הזרות מ-s ל-t יהיה ב-n[s].

סיבוכיות:

1. TopSort: $O(m + n)$.
2. בניית G^T: $O(m + n)$.
3. CountPaths: $O(n)$.

סה"כ: $O(m + n)$.

נכונות:

באינדוקציה:

<u>בדיקה:</u> מקודקוד t לעצמו – קיימת אפשרות אחת בלבד (שורה 3).

<u>נניח:</u> עבור כל קודקוד $k < j$ מתקיים שהשדה n מעודכן בצורה נכונה, ונוכיח עבור j.

<u>הוכחה:</u> כשנגיע לקודקוד ה-j-י, כל קודמיו כבר "ביקרו" אצלו, ועידכנו את שדה ה-n שלו (על-פי תכונות המיון הטופולוגי, שמכוון את הגרף לכיוון יחיד).

מכיוון שעל-פי הנחת האינדוקציה כל הקודמים מעודכנים כראוי, הרי שהסכימה בקודקוד ה-j-י מניבה תוצאה נכונה.

אם, בשלילה, היה עוד מסלול מקודקוד כלשהו לקודקוד ה-j-י, קודקוד זה כבר טופל בעבר, ועל-כן שדה ה-n שלו כבר היה מתעדכן.

לכן – לא ייתכן שקיים עוד מסלול שלא נכלל בחישוב.

■

<u>שאלה:</u>

נתון אוסף של קטעים פתוחים $I_1=(s_1, t_1), I_2=(s_2, t_2), \ldots, I_n=(s_n, t_n)$. תאר אלגוריתם המכריע האם באוסף קיימים שני קטעים שאחד מהם מכיל את השני, בזמן $O(n \log n)$.

<u>פתרון:</u>

<u>תאור האלגוריתם:</u>

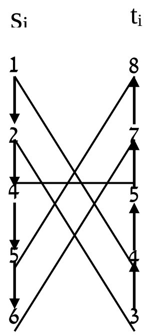
נמיין את s_i בסדר עולה, ו את t_i בסדר יורד.

נחבר את כל הקדקודים s_i בקשת מכוונת מהקטן ביותר, לגדול ביותר, ואת כל הקדקודים t_i בקשת מכוונת מהקטן ביותר, לגדול ביותר.

נחבר את s_i עם t_i בקשת ללא כיוון.

כעת נריץ BFS למציאת מעגלים (קודקוד שאינו לבן). כל מעגל יעיד עד כך שהקצוות המתחברות על-ידי הקשת העליונה ביותר מוכל בקט המכיל את הקצוות עם הקשת התחתונה ביותר.
אם קשתות נחתכות – הקטעים הקשורים אליהם לא מוכלים זה בזה.
למשל:

יהיו הקטעים: $(5, 8), (6, 7), (1, 4), (4, 5), (2, 3)$.
אזי הגרף יראה כך:



<u>סיבוכיות:</u>

מיון: $O(n \log n)$.
BFS: $(m + n)$.
סה"כ: $O(n \log n)$.

<u>נכונות:</u>

אם הגענו למעגל – בהכרח נקבל סידור כזה של קדקודים שמראים הכלה, כי עבור מעגל כלשהו קיימת הקשת (s_i, t_i), והקשת (s_j, t_j), כך ש-s_i נמצא מעל s_j ו-t_i נמצא מעל t_j, כלומר הקטעים לא נחתכים.
אם s_i נמצא מעל s_j, אזי ערכו (d) קטן משל s_j. אם t_i נמצא מעל t_j ערכו גדול יותר.
לכן נקבל – $(s_j, t_j) \subseteq (s_i, t_i)$.

■

<u>שאלה:</u>

נתון גרף G=(V, E) לא מכוון, וקבוצה $U \subseteq V$. הצע אלגוריתם המוצא בזמן O(m + n) את המרחקים הקצרים ביותר מ-U לכל אחד מקדקודי הגרף.

<u>פתרון:</u>

<u>תאור האלגוריתם:</u>

1. נריץ על G את אלגוריתם BFS^*, המתואר להלן:

<u>BFS(G, S)</u>*

```
for each vertex  u ∈ V
        color[u] ← WHITE
        d[u] ← ∞
        π[u] ← NIL
for each vertex  u ∈ S
        color[u] ← GRAY
        d[u] ← 0
        Q ← Q ∪ {u}
while  ( Q ≠ ϕ )
        u ← head[Q]
        for each v ∈ Adj[u]
                if  ( color[v] = WHITE )
                        color[v] ← GRAY
                        d[v] ← d[u] + 1
                        π[v] ← u
                        Enqueue(Q, v)
        Dequeue(Q)
        color[u] ← BLACK
```

<u>סיבוכיות:</u>

ברור שהסיבוכיות זהה לזו של אלגוריתם BFS "הרגיל": O(m + n).

<u>נכונות:</u>

נובעת ישירות מהוכחת הנכונות של BFS.

■

<u>שאלה:</u>

בהינתן גרף לא מכוון $G=(V, E)$, ושתי קבוצות קדקודים $S\subseteq V$, $T\subseteq V$, הצע אלגוריתם המוצא את המרחק הקצר ביותר ביניהן בסיבוכיות $O(m + n)$.

<u>פתרון:</u>

<u>תאור האלגוריתם:</u>

1. נריץ את אלגוריתם BFS^*, המתואר להלן:

<u>BFS(G, T, S)</u>*

```
1.   for each vertex u ∈ V
2.        color[u] ← WHITE
3.        d[u] ← ∞
4.        π[u] ← NIL
5.   for each vertex  u ∈ T
6.        color[u] ← RED
7.   for each vertex u ∈ S
8.        if  ( color[u] = RED)
9.             return 0
10.       color[u] ← GRAY
11.       d[u] ← 0
12.       Enqueue (Q, u)
13.  while  ( Q ≠ ϕ )
14.       u ← head[Q]
15.       for each v ∈ Adj[u]
16.            if  ( color[v] = RED )
17.                 return (d[v] ← d[u] + 1)
18.            if  ( color[v] = WHITE )
19.                 color[v] ← GRAY
20.                 d[v] ← d[u] + 1
21.                 π[v] ← u
22.                 Enqueue (Q, v)
23.       Dequeue(Q)
24.       color[u] ← BLACK
25.   return ∞                    // ∞ means there's no path connecting vertexes from S and from T
```

<u>סיבוכיות:</u>

6-5	מעבר על קבוצת הקדקודים:	$O(n)$.
12-7	מעבר על קבוצת הקדקודים:	$O(n)$.
17-16		$O(m)$.
25		$O(1)$.
השאר זהה לסיבוכיות של BFS "הרגיל":		$O(m + n)$.

סה"כ: $O(m + n)$.

<u>נכונות:</u>

עיקר הנכונות נובעת ישירות מהוכחת הנכונות של BFS "הרגיל". הותר להוכיח:

נניח שקודקוד v הוא הקודקוד האדום הראשון ש-BFS^* הגיע אליו. נניח שהמסלול הוא מקודקוד $u\in S$ לקודקוד $v\in T$. לפיכך נסמן את אורך המסלול ב- $d[v]$.

נניח שקיים מסלול אחר, מקודקוד $w\in S$ לקודקוד $z\in T$, שאורכו k. נניח בשלילה ש: $d[v] > k$.

נביט ב-BFS הרגע שהוא מטפל בקדקודים ברמה ה-k. אחד מקדקודים אלו הוא z. כלומר - האלגוריתם הגיע לקודקוד z לפני שהגיע לקודקוד v. $\Leftarrow$ v אינו הקודקוד האדום הראשון שהאלגוריתם הגיע אליו, **בסתירה להנחה.**

■

<u>שאלה:</u>

בהינתן גרף לא מכוון $G=(E, V)$, ושתי קבוצות קדקודים $S\subseteq V$, $T\subseteq V$, הצע אלגוריתם המוצא קבוצה Z – כל הקדקודים בגרף שנמצאים במרחק שווה מ-S ומ-T, בסיבוכיות $O(m + n)$.

<u>פתרון:</u>

<u>תאור האלגוריתם:</u>

1. נריץ את אלגוריתם BFS^*, המתואר להלן:

<u>BFS(G, T, S)</u>*

```
1.  for each vertex u ∈ V
2.        color[u] ← WHITE
3.        d[u] ← ∞
4.        π[u] ← NIL
5.  for each vertex  u ∈ S
6.        color[u] ← RED
7.        d[u] ← 0
8.        Enqueue (Q, u)
9.  for each vertex u ∈ S
10.       d[u] ← 0
11.       if  ( color[u] = RED)
12.             Z ← Z ∪ {u}
13.       else
14.             color[u] ← BLUE
15.             Enqueue (Q, u)
16. while  ( Q ≠ φ )
17.       u ← head[Q]
18.       for each v ∈ Adj[u]
19.             if  ( color[v] = WHITE )
20.                   color[v] ← color[u]
21.                   d[v] ← d[u] + 1
22.                   π[v] ← u
23.                   Enqueue (Q, v)
24.             if  ( color[u] = Opposite(color[v] ) )        //Opposite: if color is RED then BLUE,
                                                                if color is BLUE, than RED.
25.                   if  ( d[v] = d[u] + 1 )
26.                         Z ← Z ∪ {v}
27.       color[u] ← BLACK
28.       Dequeue(Q)
29. for each vertex u ∈ V
30.       if  ( color[u] = WHITE )
31.             Z ← Z ∪ {u}                              //this vertex has d = ∞ from both T and S.
```

<u>סיבוכיות:</u>

8-5 $O(n)$.

15-9 $O(n)$.

26-24 $O(m)$.

31-29 $O(n)$.

השאר זהה ל-BFS הרגיל.

סה"כ: $O(m + n)$.

<u>נכונות:</u>

רב הוכחת הנכונות נובעת ישירות מנכונות BFS, וביניהן 2 הטענות הבאות:

<u>טענה 1:</u>

לכל קודקוד v שנצבע באדום, d[v] מחזיק את המרחק המינימלי של v מהקבוצה S.

<u>טענה 2:</u>

לכל קודקוד v שנצבע בכחול, d[u] מחזיק את המרחק המינימלי של v מהקבוצה T.

מטענות אלו, ומכך שהאלגוריתם מכניס ל-Z רק קדקודים שהמרחק שלהם שווה (על-פי שורות 12-11, 26-25, 31-30).

■

<u>שאלה:</u>

נתון גרף קשיר ולא מכוון $G=(V, E)$. הצע אלגוריתם המוצא בסיבוכיות $O(m)$, בהינתן זוג צמתים בגרף s ו-t, את קבוצת כל הצמתים הנמצאים על כל מסלול קצר אפשרי בין s ל-t.

<u>פתרון:</u>

<u>תאור האלגוריתם:</u>

1. נתחזק רשימת שכנויות 'E עבור גרף $G'=(V, E')$.
2. נריץ את אלגוריתם BFS^* על G, מ-s ל-t, המתואר להלן:

<u>$BFS^*(G, s)$</u>

```
for each vertex  u∈V
        color[u] ← WHITE
        d[u] ← ∞
        π[u] ← NIL
        E'[u] ← ϕ
color[s] ← GRAY
d[s] ← 0
Q ← {s}
while  ( Q ≠ ϕ )
        u ← head[Q]
        for each v∈Adj[u]
                if  ( color[v] = WHITE )
                        d[v] = d[u] + 1
                        π[v] ← u
                        E'[v] ← u
                        Enqueue(Q, v)
                else
                        if  ( d[v] = d[u] + 1 )
                                E'[v] ← u
        Dequeue(Q)
        color[u] ← BLACK
```

3. נריץ BFS על גרף $G'=(V, E')$, מ-t ל-s.
   נדפיס את הקדקודים המתגלים במהלך ריצת ה-BFS.

<u>סיבוכיות:</u>

BFS^*: $O(m + n)$.
BFS: $O(m + n)$.
סה"כ: $O(m + n)$, ומכיוון שהגרף קשיר – $O(m)$.

<u>נכונות:</u>

נניח בשלילה כי קיים מסלול קצר כלשהו, שאחד מקדקודיו לא הודפס. נניח, בלי הגבלת הכלליות, כי קודקוד זה הוא v.
מכיוון שהמסלול הוא מסלול קצר ביותר, ערכי הקדקודים במסלול יהיו סדרה עולה מ-0 (עבור קודקוד המקור s), ועד d[t] (המרחק מ-s ל-t). לכן, ב-'G יהיה המסלול ההפוך – מ-t ל-s, העובר דרך אותם הקדקודים. לכן, בהכרח BFS מ-t ל-s על 'G ימצא את קודקוד v, בסתירה להנחה.

■

שאלה:

יהי $G=(V, E)$ גרף מכוון. תהי $u \in V$ צומת כלשהי בגרף. כתוב אלגוריתם המדפיס את אורך המעגל האי זוגי הקצר ביותר ש-u משתתף בו. אם לא קיים מעגל – הודע על כך.

פתרון:

תאור האלגוריתם:

1. נבנה גרף חדש $G'=(V', E')$ באופן הבא:

$$\forall u \in V,\ u', u \in V'$$
$$\forall (u, v) \in E,\ (u', v), (u, v') \in E'$$

2. נריץ על 'G את אלגוריתם BFS, החל מקודקוד u, כאשר אם נגיע ל-'u, אזי נחזיר את $d[u']$, ונסיים.
3. אם BFS הסתיים ולא הגענו ל-'u, נודיע שלא קיים מעגל כנדרש.

סיבוכיות:

1. בניית הגרף 'G: $O(m + n)$.
2. BFS: $O(m + n)$.
3. $O(1)$.

סה"כ: $O(m + n)$.

נכונות:

<u>טענה 1:</u>

בהינתן $u, v \in V$, קיים מסלול באורך אי-זוגי מ-u ל-v ב-G $\Leftrightarrow$ קיים מסלול מ-u ל-'v ב-'G.

<u>הוכחה:</u>

<u>כיוון $\Leftarrow$:</u> נניח ב-G קיים מסלול באורך אי-זוגי מ-u ל-v. נניח המסלול הוא: $u, v_1, v_2, \ldots, v_n, v$.
אזי, על-פי בנייה, ב-'G קיים המסלול: $u, v_1', v_2, v_3', \ldots, v_n, v'$.

<u>כיוון $\Rightarrow$:</u> נניח ב-'G קיים מסלול מ-u ל-'v. נניח המסלול הוא: $u, v_1', v_2, v_3', \ldots, v_n, v'$.
אזי ברור, על פי בנייה, שהמסלול מ-u עד v_n הוא מסלול באורך זוגי, ולכן ברור שהמסלול מ-u ועד 'v הוא מסלול באורך אי זוגי.

<u>טענה 2:</u>

$u \in V$ משתתף במעגל אי זוגי ב-G $\Leftrightarrow$ ב-'G קיים מסלול מ-u ל-'u.

<u>הוכחה:</u>

על פי בנייה, קיים מעגל אי זוגי ב-G ש-u משתתף בו, והוא $u, v_1, v_2, \ldots, u$ $\Leftrightarrow$ קיים ב-'G המסלול: $u, v_1', v_2, \ldots, u'$.

<u>טענה 3:</u>

אם u משתתף במעגלים אי זוגיים ב-G, אזי האלגוריתם מוצא את אורך המעגל המינימלי.

<u>הוכחה:</u>

אם u משתתף במעגלים אי זוגיים ב-G, אז על פי טענות 1 ו-2, ב-'G קיימים מסלולים מ-u ל-'u.
המסלול מ-u ל-'u ב-'G הוא מינימלי, על פי נכונות ותכונות ה-BFS.

■

שאלה:

נתון גרף קשיר לא מכוון $G=(V, E)$, וקבוצת צמתים $A = \{a_1, a_2, ..., a_n\}$. תאר אלגוריתם הפועל בזמן $O(m)$, ומוצא לכל צומת $v \in V$ את אורכו של מסלול קצר ביותר מ-v לצומת קרוב ביותר מקבוצת צמתים A.

פתרון:

תאור האלגוריתם:

1. נוסיף לגרף G קודקוד s, וקשתות המחברות כל קודקוד מ-A עם s. נסמן גרף זה כ-'G.
2. נאתחל את השדה $d(s)$ ב- (1-), ונריץ BFS מ-s.

בסיום ה-BFS, כל קודקוד יחזיק את אורך המסלול המבוקש בשדה ה-d שלו.

סיבוכיות:

1. יצירת 'G: $O(n)$.
2. BFS: $O(m + n)$

סה"כ: $O(m + n)$, ומכיוון ש-G קשיר – $O(m)$.

נכונות:

טענה:

המסלול הקצר ביותר שמוצא האלגוריתם מ-s לכל $v \in V$ ב-'G מקיים את התכונות הבאות:

א. מסלול זה עובר דרך אחד הצמתים השייכים ל-A.

ב. בהינתן שהמסלול הנ"ל עובר דרך קודקוד $a_k \in A$, אזי אם לאחר הרצת BFS על 'G מ-s קיבלנו $d[v] = k$, אזי לאחר הרצת BFS על 'G מ-a_k נקבל $d[v] = k$.

ג. לא קיים $a_l \in A$, $a_l \neq a_k$, כך שלאחר הרצת BFS על 'G מ-a_l נקבל $d[v] < k$.

הוכחת הטענה:

א. לפי בניית 'G, לכל $v \in V$, כל קודקוד מ-s ל-v עובר דרך קודקוד a כלשהו השייך ל-A. מסלול זה עובר דרך קודקוד יחיד מ-A, משום שבונים את 'G כך שכל אברי A נמצאים ברמה R_1 של עץ ה-BFS שנוצר על-ידי הרצת BFS על 'G מ-s.

ב. אם נקבל $d[v] > k$, זו סתירה לנכונות אלגוריתם BFS על 'G מ-a_k; ואם נקבל $d[v] < k$, זו סתירה לנכונות אלגוריתם BFS על 'G מ-s.

ג. אם קיים קודקוד a_l כזה, נקבל סתירה לנכונות BFS על 'G מ-s.

מנכונות הטענה $\Leftarrow$ נכונות האלגוריתם.

■

שאלה:

נתון גרף לא מכוון, סופי וקשיר $G=(V, E)$ (משקלות הקשתות כולם 1), זוג צמתים s, t וקבוצה $A \subseteq V-\{s,t\}$. תאר אלגוריתם בסיבוכיות $O(|E|)$ שעונה על השאלה הבאה: "האם קיים צומת $v \in A$, כך שיש מסלול קל ביותר מ-s ל-t, העובר דרך v?"

פתרון:

תאור האלגוריתם:

1. נריץ את אלגוריתם BFS על G מקודקוד s.
2. לכל $v \in V$, נשמור את $d[v]$ כ-$l_{s,v}$.
3. נריץ את אלגוריתם BFS על G מקודקוד t.
4. לכל $v \in V$, נשמור את $d[v]$ כ-$l_{t,v}$.
5. לכל $v \in A$, נבדוק:

   אם מתקיים: $l_{s,v} + l_{t,v} = l_{s,t}$, אזי קודקוד v נמצא על מסלול קצר ביותר מ-s ל-t.

   אחרת – קודקוד v אינו נמצא על מסלול קצר ביותר מ-s ל-t.

סיבוכיות:

1. BFS: $O(m + n)$.
2. מעבר על כל קבוצת הקדקודים: $O(n)$.
3. BFS: $O(m + n)$.
4. מעבר על כל קבוצת הקדקודים: $O(n)$.
5. מעבר על כל קבוצת הקדקודים: $O(n)$.

סה"כ: $O(m + n)$, ומכיוון שמדובר בגרף קשיר, הלא שביטוי זה שווה ל-$\mathbf{O(m)}$.

נכונות:

מנכונות BFS נובע כי מסלול קצר ביותר מ-s ל-$v \in V$ כלשהו יוחזק ב-$l_{s,v}$.

באופן זהה, $l_{t,v}$ יחזיק מסלול קצר ביותר מ- t ל-$v \in V$ כלשהו.

לכן, מסלול קצר ביותר מ-s ל-t, העובר דרך קודקוד v הוא חיבורם של שני המסלולים הנ"ל, ואורכו הוא $l_{s,v} + l_{t,v}$.
נוכיח שאורך מסלול זה הינו קצר ביותר:

נניח בשלילה ש: $l_{s,v} + l_{t,v} > l_{s,t}$.
אזי קיימות האפשרויות:

א) קיים מסלול מ-s ל-v, שאורכו קטן מ-$l_{s,v}$ $\Leftarrow$ **סתירה לכך ש: $l_{s,v}$ מינימלי!**

ב) קיים מסלול מ-t ל-v, שאורכו קטן מ-$l_{t,v}$ $\Leftarrow$ **סתירה לכך ש: $l_{t,v}$ מינימלי!**

$\Leftarrow$ $l_{s,v} + l_{t,v} = l_{s,t}$ $\Leftarrow$ אורך המסלול שהרכבנו זהה למרחק מ-s ל-t, ולכן זהו מסלול קצר ביותר.

■

<u>שאלה:</u>

נתון גרף מכוון $G=(V, E)$, ממושקל, וקבוצת צמתים $U\subseteq V$, ושני צמתים s ו-t. תאר אלגוריתם המוצא את אורך המסלול הקל ביותר מ-s ל-t, העובר דרך צומת אחת לפחות ב-U.

<u>פתרון:</u>

<u>תאור האלגוריתם:</u>

1. נבדוק: אם $t\in U$ או $s\in U$, אז נפעיל Dijksta באופן הרגיל.
2. אחרת: ניצור גרף $G'=(V', E)$ מכוון וממושקל, כאשר $V'=V\setminus\{t\}$.
3. עבור כל קודקוד $v'\in U[G']$, נחבר בקשת (v', v) בעלת משקל 0, את קודקוד v' עם הקודקוד התואם שלו v בגרף G.
4. נריץ את האלגוריתם של Dijkstra על גרף G', מ-s' ל-t

<u>סיבוכיות:</u>

1. בדיקת קדקודי הקבוצה U: $O(n)$.
   אם נפעיל Dijkstra: $O(m \log n)$.
2. יצירת הגרף: $O(m + n)$.
3. מעבר על כל הקדקודים: $O(n)$.
4. האלגוריתם של Dijkstra: $O(m \log n)$.

סה"כ: $O(m \log n)$.

<u>נכונות:</u>

סעיף 1 – ברור.

לא ניתן להגיע לקודקוד t בגרף G', עליו אנו מריצים את Dijkstra. האפשרות היחידה להגיע ל-t, היא אם נעבור לגרף G. אנו עוברים מגרף G' לגרף G בפעם הראשונה שנפגש בקודקוד השייך לקבוצה U.

מכאן – הנכונות נובעת ישירות מהנכונות של Dijkstra. (נשים לב שה"מעבר" מגרף G' לגרף G, לא משנה את נכונות Dijkstra, משום ש"קשת המעבר" בעלת משקל מינימלי - 0).

■

<u>שאלה:</u>

נתון גרף $G=(V, E)$ לא מכוון וממושקל, ושתי תתי-קבוצות $V_1, V_2 \subseteq V$. תאר אלגוריתם המוצא את המסלול הקל ביותר בין V_1 ל-V_2 ב-$O(m \log n)$.

<u>פתרון:</u>

<u>תאור האלגוריתם:</u>

1. ראשית נבדוק: אם $V_1 \cap V_2 \neq \phi$ אזי נחזיר תשובה "0", ונסיים.
2. נריץ על G, החל מ-V_1 (ללא הגבלת הכלליות), את Dijkstra* , המתואר להלן:

<u>Dijkstra(G, w, V_1)</u>*

```
for each vertex  u ∈ V
      d[u] ← ∞
      π[u] ← NIL
for each vertex  u ∈ V1
      d[u] ← 0
S ← ϕ
Q ← V
while  ( Q ≠ ϕ )
      u ← ExtractMin(Q)
      S ← S ∪ {u}
      for each v∈Adj[u]
            if  ( d[v] > d[u] + w(u, v) )
                  d[v] ← d[u] + w(u, v)
                  π[v] ← u
```

3 נרוץ על הקבוצה V_2, ונחזיר את ערך ה-d המינימלי.

<u>סיבוכיות:</u>

1. בדיקה האם הקבוצות זרות: $O(n)$.
2. Dijkstra*: $O(m \log n)$.
3. ריצה על קדקודי הקבוצה V_2: $O(n)$.

סה"כ: $O(m \log n)$.

<u>נכונות:</u>

הנכונות של סעיף 1 – ברור(המרחק בין הצומת לעצמה).

אחרת:

הנכונות נובעת ישירות מנכונות של Dijkstra:

Dijkstra מחשב מרחקים מינימליים מקודקוד מקור לכל שאר קדקודי הגרף. את ערך ה-d של קודקוד המקור מאתחלים ב-0.

Dijkstra מוציא קודקוד זה ראשון מהתור Q, ומבצע הקלות לשכנים שלו.

באלגוריתם Dijkstra* שהוצע, יש כמה קדקודים המאותחלים בשדה ה-d שלהם ל-0, ולכן קדקודים אלו ישלפו ראשונים מהתור Q.

לכן, בתום סעיף 2 נקבל את רשימת המרחקים המינימליים של כל קודקוד v מקדקודי הקבוצה V_1, לרבות קדקודי הקבוצה V_2.

לכן – אם נמצא את הקודקוד בעל ערך ה-d המינימלי מבין קדקודי הקבוצה V_2, הרי שנמצא את המסלול המינימלי בין 2 הקבוצות V_1 ו-V_2. וזה בדיוק מה שמתבצע בסעיף 2.

ומכאן נובעת נכונות האלגוריתם.

■

<u>שאלה:</u>

נתון גרף $G=(V, E)$ מכוון וממושקל, ונתונים 2 קדקודים $s, t\in V$. נגדיר: *אורך מסלול* כמשקל הקשת הכבדה ביותר. הצע אלגוריתם המוצא מסלול כבד ביותר בין צומת s לצומת t.

<u>פתרון:</u>

<u>תאור האלגוריתם:</u>

1. נריץ על G את אלגוריתם $Kruskal^*$, המתואר להלן:

<u>$Kruskal^*(G, w)$</u>

```
A ← ϕ
for each vertex  v∈V
      MakeSet(v)
sort the edges of E by non-increasing weight w
for each edge  (u, v) ∈ E, in sorted order
      if  ( FindSet(u) ≠ FindSet(v) )
            A ← A ∪ {(u, v)}
            Union(u, v)
return A
```

2. נגדיר: $T = (V, A)$. (הערה: A היא קבוצת הקשתות עם משקליהן.)
נריץ על T את אלגוריתם BFS, מקודקוד s לקודקוד t.
אם BFS הסתיים ולא הגענו ל-t, אזי נחזיר תשובה: "לא קיים מסלול בין s ו-t כלל", ונסיים.
3. נרוץ לפי שדה ה-π, כאשר נתחיל בקודקוד t ועד ל-s, ונסכום את משקלי הקשתות עליהן אנו עוברים.

<u>סיבוכיות:</u>

1. $Kruskal^*$: $O(m \log n)$.
2. BFS על עץ: $O(m)$.
3. ריצה על הקשתות בעץ: $O(m)$.

סה"כ: $O(m \log n)$.

<u>נכונות:</u>

ברור ש-$Kruskal^*$ ימצא עץ פורש מקסימלי.
ברור שהקשת המקסימלית במסלול מ-s ל-t תהיה בעץ זה. (על-פי אופן ריצתו של $Kruskal^*$ - הקשתות הכבדות נכנסות קודם לקבוצה A).
ברור שעל-ידי הרצת BFS על עץ זה נמצא מסלול (יחיד) בין s ו-t. (מסלול יחיד כי מדובר בעץ).
לכן – ברור שסכימת משקלי הקשתות לאורך המסלול מ-s ל-t תניב לנו את אורך המסלול המקסימלי.

■

<u>שאלה:</u>

נתון גרף לא מכוון עם משקלים אי-שליליים על הקשתות, זוג צמתים s ו-t, וקשת e.

א. הצע אלגוריתם יעיל ככל שתוכל הבודק אם הקשת e נמצאת על כל המסלולים הקלים ביותר בין s ל-t. הוכח נכונות ונתח סיבוכיות.

ב. הצע אלגוריתם יעיל הבודק האם הקשת e נמצאת על מסלול קל ביותר כלשהו בין s ל-t.

<u>פתרון:</u>

<u>א.</u>

<u>תאור האלגוריתם:</u>

1. נריץ את אלגוריתם Dijkstra על גרף G מצומת s. נזכור $\lambda(t)$.
2. ניצור גרף $G'=G(V, E)\setminus\{e\}$.
3. נריץ את אלגוריתם Dijkstra על גרף G' מצומת s. נזכור $\lambda'(t)$.
4. אם $\lambda'(t)=\lambda(t)$ אזי ישנו מסלול קל שלא כולל את הקשת e.

<u>סיבוכיות:</u>

1. Dijkstra: $O(m \log n)$.
2. יצירת הגרף: $O(m+n)$.
3. Dijkstra: $O(m \log n)$.
4. בדיקה: $O(1)$.

סה"כ: $O(m \log n)$.

<u>נכונות:</u>

נראה שאם $\lambda(t) = \lambda'(t)$ אזי ישנו מסלול קצר שלא עובר דרך e:

לפי נכונות Dijkstra נמצא מסלול מ-s ל-t שמשקלו קטן ביותר, שהוא בהכרח לא עובר דרך הקשת e (שהרי ב-G' הקשת e לא נמצאת).

נראה שאם כל המסלולים הקלים ביותר עוברים דרך e, אז $\lambda'(t) \neq \lambda(t)$:

Dijkstra שהורץ על גרף G' לא היה מחזיר את אותו ערך של $\lambda(t)$, משום שללא e אין מסלול באורך $\lambda(t)$ מ-s ל-t, על-פי נכונות Dijkstra.

■

<u>ב.</u>

<u>תאור האלגוריתם:</u>

1. נמצא את קדקודי הקשת e, ונסמנם: $e=(u, v)$.
2. נריץ Dijkstra על גרף G מצומת s. נזכור $\lambda(t)$.
3. ניצור גרף $G'=G(V, E)\setminus\{e\}$.
4. נריץ את אלגוריתם Dijkstra על גרף G' מצומת s. נזכור $\lambda'(u)$.
5. נריץ את אלגוריתם Dijkstra על גרף G' מצומת t. נזכור $\lambda'(v)$.
6. אם $\lambda(t) = \lambda'(u) + w(e) + \lambda'(v)$, אזי הקשת e נמצאת על מסלול קל.

<u>סיבוכיות:</u>

1. מציאת קדקודי הקשת: $O(1)$.
2. Dijkstra: $O(m \log n)$.
3. יצירת הגרף: $O(m+n)$.
4. Dijkstra: $O(m \log n)$.
5. Dijkstra: $O(m \log n)$.
6. בדיקה: $O(1)$.

סה"כ: $O(m \log n)$.

<u>נכונות:</u>

נראה שאם הערך $\lambda(t) = \lambda'(u) + w(e) + \lambda'(v)$, אזי קיים מסלול קל ביותר העובר דרך קשת e:

על-פי Dijkstra – קיים מסלול באורך $\lambda'(u)$, מ-s עד u, ומסלול באורך $\lambda'(v)$, מ-t עד v. ולכן, קיים גם מסלול $s\rightarrow u\rightarrow v\rightarrow t$ (דרך e), ואורכו קל ביותר.
נראה שאם $\lambda(t) \neq \lambda'(u) + w(e) + \lambda'(v)$, אז לא קיים מסלול קל ביותר העובר דרך e :
הסכום : $\lambda = \lambda'(u) + w(e) + \lambda'(v)$, לא יכול להיות קטן מ-$\lambda(t)$, כי זה בסתירה ל-Dijkstra. $\Leftarrow$ המסלול הקל ביותר דרך e, $s\rightarrow u\rightarrow v\rightarrow t$, משקלו גדול/שווה מהמסלול הקל שלא עובר דרך e $\Leftarrow$ e לא נמצאת על מסלול קל ביותר.
לא יכול להיות מסלול דרך e, הקל יותר מ-λ, משום שמסלול דרך e חייב לעבור דרך $s\rightarrow u$, ודרך $v\rightarrow t$, וערכים אלו הם מינימליים (על-פי Dijkstra). $\Leftarrow$ λ הוא המסלול הקל ביותר האפשרי דרך e.

■

שאלה:

נתון גרף סופי לא מכוון $G=(V, E)$ עם משקלים אי-שליליים על הקשתות. הקשתות צבועות בשני צבעים: אדום ושחור. נתון קודקוד s.

א. הצע אלגוריתם בסיבוכיות $O(m \log n)$, שמוצא לכל קודקוד $v\in V$ את משקל מסלול קל ביותר (שסכום משקלי הקשתות מינימלי) בין המסלולים מ-s ל-v, המכילים מספר זוגי של קשתות אדומות.

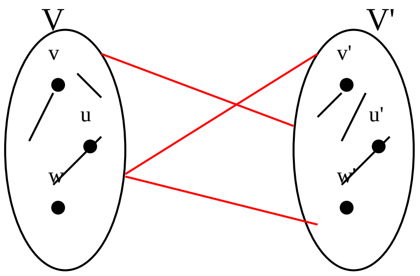
ב. הצע אלגוריתם בסיבוכיות $O(m \log n)$, שמוצא לכל קודקוד $v\in V$ את משקל מסלול קל ביותר (שסכום משקלי הקשתות מינימלי) בין המסלולים מ-s ל-v, המכילים מספר זוגי של קשתות אדומות ומספר אי-זוגי של קשתות שחורות.

פתרון:

א.

תאור האלגוריתם:

1. ניצור גרף חדש $G^*=(V^*, E^*)$, כאשר:
ניצור קבוצת קדקודים 'V, שהיא שכפול של קבוצת הקדקודים המקורית V. כלומר $u'\in V'$ אם $u\in V$.
נגדיר: $V^* = V \cup V'$
ניצור קבוצת קשתות E^*:
אם הקשת (u, v) ב-G אדומה: נוסיף לקבוצה E^* את הקשת (u, v').
אם הקשת (u, v) ב-G שחורה: נוסיף לקבוצה E^* את הקשת (u, v), ואת הקשת (u', v').



2. נריץ את האלגוריתם של Dijkstra מ-s ל-v.
Dijkstra ימצא את המשקל המינימלי.

סיבוכיות:

1. יצירת הגרף החדש: $O(m + n)$.
2. האלגוריתם של Dijkstra: $O(m \log n)$.

סה"כ: $O(m \log n)$.

נכונות:

למעשה – יצרנו ב-V^* **חתך** (V, V').
על-פי בניית הגרף G^*, הקשתות האדומות הן למעשה **גשרים**, והקשתות השחורות מקשרות בין קדקודים מאותה קבוצת קדקודים.
בכל פעם שנעבור בגשר, "נחליף" את קבוצת הקדקודים בה נמצא.
נניח בשלילה, שהגענו מ-s ל-v, במספר אי-זוגי של קשתות אדומות.
אזי, במסלול מ-s ל-v קיים מספר אי-זוגי של קשתות אדומות. ⇐ "החלפנו" קבוצת קדקודים מספר אי-זוגי של פעמים.
מכיוון שקיימות רק 2 קבוצות קדקודים, הרי שבסיום נמצא בקבוצת הקדקודים השונה מזו שהתחלנו בה. ⇐ הקדקודים s ו-v נמצאים בקבוצות קדקודים שונות. ⇐ **סתירה לבניית הגרף.**
מכאן – שאם הגענו מ-s ל-v, הרי שעברנו מספר זוגי של קשתות אדומות.
נכונות המשקל המינימלי נובעת ישירות מנכונות האלגוריתם של Dijkstra.

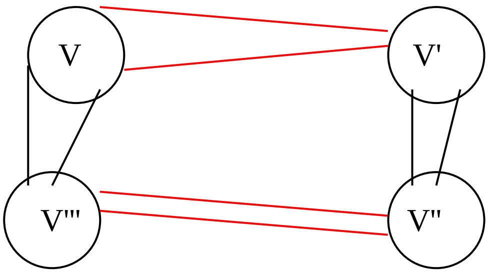
■

ב.

תאור האלגוריתם:

1. ניצור גרף חדש $G^*=(V^*, E^*)$, כאשר:
ניצור קבוצת קדקודים 'V, שהיא שכפול של קבוצת הקדקודים המקורית V. כלומר $u'\in V'$ אם $u\in V$.
ניצור באופן זהה קבוצת קדקודים "V ו-"'V.

נגדיר: $V^* = V \cup V' \cup V'' \cup V'''$
ניצור קבוצת קשתות E^*:
אם הקשת (u, v) ב-G אדומה: נוסיף לקבוצה E^* את הקשת (u, v'), ואת הקשת (u'', v''').
אם הקשת (u, v) ב-G שחורה: נוסיף לקבוצה E^* את הקשת (u, v''), ואת הקשת (u', v''').



2. נריץ את האלגוריתם של Dijkstra מ-s ל-v''.
Dijkstra ימצא את המשקל המינימלי.

סיבוכיות:
1. יצירת הגרף החדש: $O(m + n)$.
2. האלגוריתם של Dijkstra: $O(m \log n)$.
סה"כ: $O(m \log n)$.

נכונות:
בהנחה כי נתחיל מ-V, נטען את הטענות הבאות:
טענה I:
אם"ם עברנו מספר זוגי של קשתות אדומות, אזי נמצא ב-V או ב-V''.
הוכחת הטענה
בדומה לסעיף א' של השאלה.

טענה II:
אם"ם עברנו מספר אי זוגי של קשתות שחורות, אזי נמצא ב-V'' או ב-V'''.
הוכחת הטענה:
בדומה לסעיף א' של השאלה.

מטענה I ומטענה II נובע כי אם"ם עברנו מספר זוגי של קשתות אדומות ומספר אי זוגי של קשתות שחורות – נמצא ב-V''.
נכונות המשקל המינימלי נובעת ישירות מנכונות האלגוריתם של Dijkstra.

■

שאלה:

נתון גרף מכוון $G=(V, E)$, עם משקלים שלמים וגדולים מ-0 על הקשתות. כמו-כן נתון הפלט של הרצת אלגוריתם למציאת מסלולים קלים ביותר מ-s לכל קודקוד אחר בגרף: לכל קודקוד הגרף רשום משקל המסלול הקל ביותר שמצא האלגוריתם מ-s ועד אליו (האלגוריתם לא מוציא את המסלולים עצמם). נאמר לנו שכל המסלולים שמצא האלגוריתם הם גם הקצרים ביותר.

א. הצע אלגוריתם יעיל למציאת משקלי מסלולים קלים ביותר מ-s לשאר קדקודי הגרף, כאשר מגדילים את המשקל של כל קשת ב-1.

ב. הצע אלגוריתם יעיל למציאת משקלי מסלולים קלים ביותר מ-s לשאר קדקודי הגרף, כאשר מקטינים את המשקל של כל קשת ב-1.

<u>הערה:</u> מסלול קל ביותר הוא מינימלי מבחינת סכום משקלי קשתותיו. מסלול קצר ביותר הוא מינימלי מבחינת מספר הקשתות.

פתרון:

<u>א.</u>

<u>תאור האלגוריתם:</u>

נריץ BFS על G מ-s, ונמצא לכל v את אורך המסלול הקצר ביותר מ-s ועד אליו. נוסיף לערך זה למשקל שרשום ב-v (מהרצת האלגוריתם שתואר בגוף השאלה).

<u>סיבוכיות:</u>

ברור שהוספנו לאלגוריתם ה-BFS "הרגיל" פעולה שמתבצעת ב-$O(1)$, ולכן הסיבוכיות נשארת: $O(m + n)$.

<u>נכונות:</u>

יהי v קודקוד כלשהו בגרף.

כל מסלול מ-s ל-v מתארך בדיוק במספר הקשתות במסלול.

לכן, מסלול שהיה הקל ביותר, וגם הקצר ביותר מ-s ל-v, נשאר מסלול קל ביותר (כל מסלול אחר התארך לפחות באותה מידה).

■

<u>ב.</u>

<u>תאור האלגוריתם:</u>

נריץ Dijkstra על G מ-s, ונמצא לכל v את אורך המסלול הקל ביותר מ-s ועד אליו. נוסיף לערך זה למשקל שרשום ב-v(מהרצת האלגוריתם שתואר בגוף השאלה).

<u>סיבוכיות:</u>

ברור שהוספנו לאלגוריתם "הרגיל" של Dijkstra פעולה המתבצעת ב-$O(1)$, ולכן הסיבוכיות נשארת: $O(m \log n)$.

<u>נכונות:</u>

נשים לב שבניגוד לסעיף א', הפעם כל מסלול יורד במספר הקשתות שבו. לכן – מסלולים הקצרים ביותר לא יישארו בהכרח המסלולים הקלים ביותר.

נשים לב שנוכל לעשות זאת, משום שהמשקולות המקוריים היו גדולים מ-0. (כלומר, גדולים/שווים 1). לכן, ברור שהמשקולות החדשים ישארו אי-שליליים, ונוכל להריץ את האלגוריתם של Dijkstra.

■

שאלה:

תאר אלגוריתם יעיל, אשר בהינתן גרף לא מכוון עם משקלות חיוביים על הקשתות וקודקוד מקור s, מסמן כל קודקוד v בגרף באופן הבא: אם אורך מסלול קצר ביותר מ-s ל-v קטן ממש מאורך מסלול קצר ביותר מבין המסלולים הקלים ביותר מ-s ל-v, אזי יסומן בסימון "<". אחרת, יסומן בסימון "=". (אורך מסלול הוא מספר הקשתות בו. משקל מסלול הוא סכום משקלי הקשתות בו).

פתרון:

תאור האלגוריתם:

1. נריץ את אלגוריתם BFS הרגיל על G. נסמן את הגרף <u>המכוון</u> שיתקבל לאחר הריצה כ-'G.
2. נריץ על 'G את אלגוריתם *Dijkstra, המתואר להלן:

<u>Dijkstra(G, w, s)</u>*

```
for each  v∈V
      d[v] ← ∞
      π[v] ← NIL
      λ[v] ← ∞
d[s] ← 0
λ[s] ← 0
S ← ϕ
Q ← V
while  ( Q ≠ ϕ )
      u ← ExtMin(Q)
      S ← S ∪ {u}
      for each v∈Adj[u]
            if  ( d[v] > d[u] + w(u, v) )
                  d[v] ← d[u] + w(u, v)
                  π[v] ← u
                  λ[v] ← λ[u] + 1
            if  ( ( d[v] = d[u] + w(u, v) )  and  ( λ[u] + 1 < λ[v] ) )
                  λ[v] ← λ[u] + 1
```

השדה λ יחזיק עבור כל קודקוד v את אורך המסלול (מספר הקשתות) המינימלי מבין המסלולים הקלים ביותר מ-s ועד לקודקוד v.

3. נריץ על V את אלגוריתם MarkVertexes(V), המתואר להלן:

<u>MarkVertexes(V)</u>

```
for each v∈V
      if  ( d[v] < λ[v] )
            mark[v] ← ">"
      else
            mark[v] ← "="
```

<u>סיבוכיות:</u>

1. BFS: O(m + n).
2. ברור שהסיבוכיות של *Dijkstra זהה לסיבוכיות של Dijkstra ה"רגיל": O(m log n).
3. MarkVertexes: O(n).

סה"כ: O(n + m log n). (אם הגרף היה קשיר – O(m log n)).

<u>נכונות:</u>

נריץ את אלגוריתם BFS על G בכדי לקבל את המסלול הקצר ביותר בין s לקודקוד v כלשהו. כמו-כן, BFS יכוון לנו את הגרף, וכך נוכל להפעיל עליו וריאציה של האלגוריתם של Dijkstra. נכונות הטענות שעלו כאן נובעת ישירות מנכונות BFS.

נכונות אלגוריתם *Dijkstra שהוצע נובעת ישירות מנכונות האלגוריתם של Dijkstra ה"רגיל". קל לראות שהוספת השדה λ, ואופן תחזוקו מביאים לכך ששדה זה מחזיק בסיום ריצת האלגוריתם את אורך המסלול הקצר ביותר מבין המסלולים הקלים ביותר מ-s לכל קודקוד v.

ברור שעם סיום ריצת 2 האלגוריתמים הללו (BFS ו-*Dijkstra), יהיה נתון לנו d[v] ו-λ[v] מעודכנים ונכונים עבור כל קודקוד v. לכן, אלגוריתם MarkVertexes הוא מימוש טכני, שלא באמת דורש הוכחת נכונות...

■

שאלה:

נתון גרף מכוון וסופי $G=(V, E)$. לקשת e יש אורך ממשי $l(e)$, לאו דווקא אי-שלילי. יתכנו ב-G מעגלים שליליים.
בהינתן שני צמתים $i, j \in V$ ומספר טבעי k $(|V| > k)$, תאר אלגוריתם המחזיר משקל מסלול מינימום מבין כל המסלולים המכוונים (לאו דווקא פשוטים) שבהם k קשתות בדיוק.

פתרון:

תאור האלגוריתם:

1. נפעיל את האלגוריתם Find-k-LongPaths המתואר להלן:

Find-k-LongPaths(G, k)

```
1.   for each vertex  v∈V
2.        λ[v] ← ∞
3.   λ[i] ← 0
4.   T ← {i}
5.   P ← ϕ
6.   t ← 1
7.   while  ( t ≤ k )
8.        for each vertex  v∈V[T]
9.             T ← T \ {v}
10.            for each edge  (v, u) ∈ E
11.                 if  ( u ∉ P )
12.                      λ[u] ← λ[v] + l[(v, u)]
13.                      P ← P ∪ {u}
14.                 else
15.                      λ[u] ← min { λ[u],  λ[v] + l[(v, u)] }
16.       T ← P
17.       P ← ϕ
18.       t ← t + 1
```

<u>הסבר:</u>

T – בסוף האיטרציה ה-t, T יכיל את כל הצמתים שיש אליהם מסלול בן k קשתות, החל מצומת i.
P – קבוצת צמתים זמנית שבמהלך האיטרציה "צוברת" את הצמתים בעלי מסלול של k קשתות מצומת i.
בסוף כל איטרציה נעתיק את P ל-T.
החיוניות של P היא למנוע מצב שהאלגוריתם לא יעצר בגלל לולאות אין-סופיות (על-ידי הוצאת צומת v מ-T בשורה 9, והכנסתה ל-T חזרה בשורה 13).
סה"כ נעבור על כל המסלולים הקיימים בני k קשתות שמתחילים ב-I, ונעדכן את משקל המסלול באחת מהאפשרויות:

1. כשנמצא לראשונה מסלול בן k קשתות מצומת i לצומת הנבדק כעת (שורות 11-13).
2. כשנמצא מסלול קל יותר לצומת הנבדק כעת (שורות 14-15)

סיבוכיות:

ישנן k איטרציות, כאשר במקרה הגרוע ביותר – נעבור על כל הקשתות בכל איטרציה: $O(k \cdot m)$.

נכונות:

<u>טענה:</u>

בסוף האלגוריתם אם צומת כלשהו $j \in T$, אזי קיים מסלול בן k קשתות מ-i ל-j.

<u>הוכחה:</u>

נוכיח באינדוקציה על סדר כניסת הצמתים ל-T:

<u>בסיס:</u> הצומת היחיד שנמצא במרחק 0 קשתות מ-i הוא i עצמו. נכון לפי שורה 3.

<u>נניח:</u> נכונות לכל הצמתים שנמצאים במרחק קטן מ-k קשתות מ-i.

<u>נוכיח:</u> עבור k קשתות:
נניח קיים צומת j שנמצא במרחק בדיוק k קשתות מ-i. נניח j' נצאת לפני j במסלול מ-i ל-j.
על-פי הנחת האינדוקציה, $j' \in T$. לכן – במהלך האיטרציה ה-k-ית נעבור על כל הקשתות היוצאות מצומת j', ובפרט על הקשת (j', j), שתוביל אותנו לצומת j.
כעת – אם זוהי הפעם הראשונה שהגענו ל-j במסלול בן k קשתות – נכניס את j ל-P.
אם הגענו אליו קודם – אז הוא כבר ב-P.
בכל מקרה – בסוף האיטרציה, נעתיק את P ל-T, ובפרט הצומת $j \in T$ בסוף האיטרציה ה-k-ית.

<u>טענה:</u>

אם קיים מסלול מ-i ל-j בן k קשתות, אזי בסוף האלגוריתם $\lambda[j]$ יכיל את המסלול הקל ביותר מ-i ל-j בן k קשתות.

<u>הוכחה:</u>

קיים מסלול בן k קשתות מ-i ל-j.

נניח בשלילה שבסוף האלגוריתם $\lambda[j]$ אינו מכיל את המסלול הקל ביותר בן k קשתות מ-i ל-j.

כלומר – קיים מסלול נוסף בן k קשתות בין i ל-j, בעל אורך קצר יותר.

נניח הצומת לפני j במסלול זה הינה v, אזי בין i ל-v קיים מסלול בן k-1 קשתות.

הראינו שאם קיים מסלול בן k-1 קשתות מ-i ל-v, אזי בסוף האיטרציה ה-k-1, $v \in T$.

כלומר באיטרציה ה-k-ית, נבדוק את כל הקשתות היוצאות מ-v, ובפרט את הקשת (v, j), שתוביל אותנו ל-j.
כעת יתכנו שני מקרים :

1. $j \notin P$

   ואז $\lambda[j] \leftarrow \lambda[v] + l[(v, j)]$.

   במקרה זה עבור כל מסלול נוסף בן k קשתות, שנמצא מ-i ל-j, נכנס לשורה ה-j, ומכיוון שהמסלול הקל ביותר כבר נתגלה - $\lambda[j]$ לא ישתנה.

2. $j \in P$

   אבל מכיוון שאנו רצים על המסלול הקל ביותר, ברור ש : $\lambda[j] > \lambda[v] + l[(v, j)]$.

   ולכן – נעדכן את $\lambda[j]$, כך שיכיל את אורך המסלול הקל ביותר.

<u>מסקנה :</u>

בסוף האלגוריתם $\lambda[j]$ הוא ארך המסלול הקל ביותר בן k קשתות מ-i ל-j.

■

<u>שאלה:</u>

יהי $T=(V, E)$ עץ עם משקלות אי-שליליים על הקשתות. הקוטר של T מוגדר כמרחק המקסימלי בין שני צמתים ב-T. הצע אלגוריתם המוצא את הקוטר (מסלול כבד ביותר בין 2 קדקודים) של T בזמן $O(n)$.

<u>פתרון:</u>

<u>תאור האלגוריתם:</u>

1. נריץ מקודקוד $s\in V$ כלשהו את אלגוריתם BFS^*, המתואר להלן:

<u>$BFS^*(G, s)$</u>

```
for each vertex  u∈V
        color[u] ← WHITE
        d[u] ← ∞
        π[u] ← NIL
color[s] ← GRAY
d[s] ← 0
Q ← {s}
while  ( Q ≠ ϕ )
        u ← head[Q]
        for each  v∈Adj[u]
                if  ( color[v] = WHITE )
                        d[v] ← d[u] + w(u, v)
                        π[v] ← u
                        Enqueue (Q, v)
        Dequeue (Q)
        color[u] ← BLACK
```

2. נמצא את הקודקוד u, כך שהמרחק מ-s ל-u מקסימלי.
1. נריץ BFS^* מ-u, ונמצא קודקוד v, כך שהמרחק מ-u ל-v הוא מקסימלי.
2. מרחק זה יהיה הקוטר והמסלול הוא המסלול היחידי בעץ מ-u ל-v.

<u>סיבוכיות:</u>

1. ברור שסיבוכיות BFS^* זהה לזו של BFS "הרגיל": $O(m + n)$.
2. ריצה על כל הקדקודים, ומציאת המינימום: $O(n)$.
3. שוב BFS^*, ומציאת מינימום: $O(m + n)$.

סה"כ: $O(m + n)$, ומכיוון שהגרף הוא למעשה עץ קשיר: $O(n)$.

<u>נכונות:</u>

<u>טענה:</u>

לכל קודקוד v שניקח ולכל קודקוד u שנמצא הכי רחוק ממנו, u נמצא על איזשהו מסלול שהוא קוטר.

<u>הוכחה:</u>

ישירות מגובה של תתי העצים.

קיימים 2 סוגי קדקודים:

1. קדקודים שנמצאים על המסלול – ואז – ברור.
2. קדקודים שלא נמצאים על המסלול – ואז ברור שהם חייבים לעבור דרך קודקוד שנמצא על המסלול הארוך ביותר.

ברור שאם אחד הקדקודים של המסלול P מ-v ל-u נפגש עם אחד הקדקודים על מסלול באורך הקוטר – אזי u חייב להיות קצה של מסלול באורך הקוטר.

■

שאלה:

נתונים גרף מכוון $G=(V, E)$ עם משקלים חיוביים על הקשתות. נתונים שלושה קדקודים $x, y, z \in V$. מצא בזמן $O(m \log n)$ מסלול קצר ביותר מ-x ל-y, העובר דרך v מספר זוגי של פעמים.

פתרון:

תאור האלגוריתם:

1. נריץ Dijkstra על G מקודקוד x.
2. אם המסלול מ-x ל-y לא עובר דרך v – נדפיס את המסלול ונסיים.
3. למציאת מסלול קל ביותר שלא עובר דרך קודקוד v:
   א. ניצור גרף $G'=(V', E')$, כאשר:

$$V' = V \setminus \{v\}$$
$$E' = E \setminus \{ (v, u) \mid u \in V, (v, u)\in E \} \setminus \{ (u, v) \mid u\in V, (v, u)\in E \}$$

   ב. נריץ Dijkstra על G' מקודקוד x.
   ג. נשמור את משקל המסלול הקל ביותר מ-x ל-y ב-d_1 (נשמור גם את המסלול עצמו).
4. למציאת מסלול קל ביותר שעובר דרך קודקוד v פעמיים:
   א. נריץ Dijkstra על G מקודקוד v.
   ב. נמצא קודקוד $v \neq t \in V$ עבורו מתקיים שהסכום: $\delta(v, t) + w(t, v)$ הוא מינימלי.
   הערות:
      1. אם לא קיימת הקשת (t, v), אזי משקלה הוא ∞.
      2. $\delta(v, t)$ הוא משקל המסלול הקל ביותר מ-v ל-t.
   ג. נשמור את משקל המסלול הקל ביותר (נשמור גם את המסלול עצמו) ב-d_2:

$$d_2 = \delta(x, v) + \delta(v, t) + \delta(v, y) + w(t, v)$$

5. נבדוק:
   א. אם $d_1 < d_2$, אז המסלול המבוקש הוא המסלול שנמצא על-ידי סעיף 3.
   ב. אם $d_1 \geq d_2 \neq \infty$, אז המסלול המבוקש הוא המסלול הנמצא על-ידי סעיף 4.
   ג. אם $d_1 = d_2 = \infty$, אז אין מסלול כנדרש.

סיבוכיות:

1.	Dijkstra:	$O(m \log n)$.
2.	בדיקת המסלול:	$O(n)$.
3.	א. בניית G':	$O(m + n)$.
	ב. Dijkstra:	$O(m \log n)$.
	ג. שמירת המשקל הקל ביותר והמסלול:	$O(m)$.
4.	א. Dijkstra:	$O(m \log n)$.
	ב. מעבר על הקשתות והקדקודים:	$O(m + n)$.
	ג. שמירת המשקל הקל ביותר והמסלול:	$O(m)$.
5.	א. בדיקה:	$O(1)$.
	ב. בדיקה:	$O(1)$.
	ג. בדיקה:	$O(1)$.
סה"כ:		$O(m \log n)$.

נכונות:

נבחין כי מאחר ומשקלי קשתות הגרף G הם חיוביים, הרי מסלול שעובר דרך vמספר זוגי של פעמים עובר דרכו אפס פעמים או פעמיים בדיוק.

על-פי סעיף 1 אנו מוצאים את המסלול הקל ביותר מ-x ל-y. נכונות הדבר נובעת מנכונות האלגוריתם של Dijkstra.

על-פי סעיף 2 אם המסלול הקל ביותר מ-x ל-y (שנמצא בסעיף 1) לא עובר כלל דרך קודקוד v, הרי שמצאנו מסלול קל ביותר העובר דרך v מספר זוגי (אפס) של פעמים. לפיכך - אנו מדפיסים מסלול זה, ומסיימים.

אם המסלול הקל ביותר עובר דרך קודקוד v פעם אחת בגרף G, אזי המסלול הקל ביותר שעובר דרך קודקוד v מספר זוגי של פעמים הוא המסלול המינימלי מבין המסלול העובר דרך קודקוד v אפס פעמים לבין המסלול העובר דרך קודקוד v פעמיים.

סעיף 3 מוצא מסלול קל ביותר מ-x ל-y שלא עובר כלל דרך קודקוד v (כלומר – אפס פעמים – זוגי).

סעיף 4 מוצא מסלול קל ביותר שעובר דרך קודקוד v פעמיים באופן המתואר להלן:

הוא מחבר את המסלול הקל ביותר שעובר דרך v פעם אחת, למעגל הקל ביותר המתחיל ומסתיים בקודקוד v.

נכונות עניין זה נובעת מנכונות Dijkstra (כלומר – על-פי אותו העיקרון של המסלול הלבן).

■

<u>שאלה:</u>

נתון גרף $G=(V, E)$ פשוט, לא מכוון, וממושקל במשקלים אי-שליליים, ונתונים שני קדקודים $u, v\in V$. תאר אלגוריתם המוצא את המסלול הזוגי הקל ביותר בין u ל-v.

<u>פתרון:</u>

<u>תאור האלגוריתם:</u>

1. נריץ על G, החל מקודקוד u, את $Dijkstra^*$, המתואר להלן:

<u>Dijkstra(G, w, s)</u>*

```
for each vertex v∈V
    odd[v] ← ∞
    even[v] ← ∞
    π-odd[v] ← NIL
    π-even[v] ← NIL
even[v] ← 0
S ← ϕ
Q ← V
while  ( Q ≠ ϕ )
    u ← ExtractMin(Q)
    S ← S ∪ {u}
    for each  v∈Adj[u]
        if  ( odd[v] > even[u] + w(u, v) )
            odd[v] ← even[u] + w(u, v)
            π-odd[v] ← u
        if  ( even[v] > odd[u] + w(u, v) )
            even[v] ← odd[u] + w(u, v)
            π-even[v] ← u
```

2. למציאת המסלול עצמו –

נתחיל מקודקוד v מה-π הזוגי (π-even), ונרוץ על שדות ה-π, זוגי - אי-זוגי, לסירוגין, כלומר: π-odd - π-even.

<u>סיבוכיות:</u>

1. ברור שהסיבוכיות של $Dijkstra^*$ זהה לזו של Dijkstra "הרגיל": $O(m \log n)$.
2. ריצה על האבות: $O(m)$.

סה"כ: $O(m \log n)$.

<u>נכונות:</u>

עיקר הנכונות, ובפרט נכונות המסלול הקל ביותר נובעת ישירות מנכונות Dijkstra.

אשר לזוגיות – הדבר ברור על-פי אופן עדכון ותחזוק השדות הזוגיים (π-even, even), והשדות האי-זוגיים (π-odd, odd), שכן בכל פעם אנו מעדכנים את המשתנים ההפכיים לאלה שעבדנו אתם – כלומר – אם עבדנו אם ה-"זוגיים" – נעדכן את ה-"אי-זוגיים", ולהפך.

■

<u>שאלה:</u>

נתון גרף לא מכוון ממושקל $G=(V, E)$, שני צמתים $u, v\in V$, וקשת $e\in E$. תאר אלגוריתם המוצא מסלול קל ביותר מ-u ל-v, המשתמש ב-e.

<u>פתרון:</u>

<u>תאור האלגוריתם:</u>

נניח שאין מעגלים שליליים בגרף G.

נסמן, מטעמי נוחות: $e=(w_1, w_2)$.

1. נריץ את האלגוריתם של Dijkstra על G, מקודקוד מקור u.
2. נשמור את $d[w_1]$ כ-$d[u, w_1]$, ואת $d[w_2]$ כ-$d[u, w_2]$.
3. נריץ את האלגוריתם של Dijkstra על G, מקודקוד מקור v.
4. נשמור את $d[w_1]$ כ-$d[v, w_1]$, ואת $d[w_2]$ כ-$d[v, w_2]$.
5. נסכום: $d_1 = d[u, w_1] + w(e) + d[v, w_2]$
$d_2 = d[u, w_2] + w(e) + d[v, w_1]$
6. נבדוק -

<u>אם $d_1 < d_2$:</u> המסלול הנדרש הוא המסלול שנמצא על-פי Dijkstra בסעיף 1 מקודקוד u לקודקוד w_1, הקשת e, והמסלול שנמצא על-פי Dijkstra בסעיף 3 מקודקוד w_2 לקודקוד u. (למעשה מצאנו את המסלול מ-u ל-w_2, אך הגרף לא מכוון).

<u>אחרת:</u> המסלול הנדרש הוא המסלול שנמצא על-פי Dijkstra בסעיף 1 מקודקוד u לקודקוד w_2, הקשת e, והמסלול שנמצא על-פי Dijkstra בסעיף 3 מקודקוד w_1 לקודקוד u. (למעשה מצאנו את המסלול מ-u ל-w_1, אך הגרף לא מכוון).

<u>סיבוכיות:</u>

1. Dijkstra: $O(m \log n)$.
2. הצבה: $O(1)$.
3. Dijkstra: $O(m \log n)$.
4. הצבה: $O(1)$.
5. פעולה אריתמטית: $O(1)$.
6. השוואה: $O(1)$.

<u>נכונות:</u>

האלגוריתם של Dijkstra מוצא מסלול קל ביותר מקודקוד מקור יחיד לכל שאר הקדקודים בגרף G.

כשנריץ Dijkstra מקודקוד מקור u, נמצא, בין היתר, את המסלול הקל ביותר מ-u ל-w_1, ומ-u ל-w_2.

כשנריץ Dijkstra מקודקוד מקור v, נמצא, בין היתר, את המסלול הקל ביותר מ-v ל-w_1, ומ-u ל-w_2.

כעת יש לבדוק שתי אפשרויות, כך שמעבר על המסלול יהיה המינימלי.

נניח, בלי הגבלת הכלליות, שהמסלול $u \rightsquigarrow w_2 \rightsquigarrow w_1 \rightsquigarrow v$ הוא המסלול הקל ביותר מ-u ל-v, המשתמש ב-e.

נניח בשלילה, שקיים מסלול אחר, קצר יותר, העובר דרך e.

כלומר – נקבל שזהו המסלול הקל ביותר מ-u ל-w_1 והמסלול הקל ביותר מ-w_2 ל-v.

אבל זוהי **סתירה למינימליות של Dijkstra**.

■

שאלה:

נתון גרף $G=(V, E)$ מכוון, **וחסר מעגלים** עם משקלים על הקשתות. עוד נתונים שני קדקודים $s, t \in V$. תאר אלגוריתם לינארי באורך הקלט, המוצא את המסילה הקצרה ביותר מ-s ל-t.

פתרון:

תאור האלגוריתם:

1. נריץ TopSort על G. נסמן את הגרף הממוין $G'=(V, E')$.
   הערה: ה-DFS המופעל ב-TopSort יופעל מקודקוד s.
2. נבנה גרף הפוך $G^T=(V, E^T)$ ביחס ל-'G, כאשר:

$$E^T = \{ (u, v) \mid (v, u) \in E[G'] \}$$

3. נריץ על G^T את השגרה CalculateWeight, החל מקודקוד t, כמתואר להלן:

```
CalculateWeight (G^T, t)
for i ← 1 to t
     μ[v_i] ← ∞
     β[v_i] ← NIL
μ[t] ← 0
for i ← t downto 1
     for each vertex u∈Adj[v_i]
          if ( μ[u] > μ[v_i] + w(v_i, u) )
               μ[u] ← μ[v_i] + w(v_i, u)
               β[u] ← v_i
```

4. משקל המסלול המבוקש הוא הערך שנמצא ב-$\mu[s]$.
   המסלול עצמו ימצא על-ידי מעבר מקודקוד s, לפי שדה הבנים - β.

סיבוכיות:

1. TopSort: $O(m + n)$.
2. בניית G^T: $O(m + n)$.
3. CalculateWeight: $O(m + n)$.
4. מציאת המסלול: $O(m)$.

סה"כ: $O(m + n)$.

נכונות:

מכיוון שהגרף חסר מעגלים – ניתן להפעיל את אלגוריתם TopSort.
מתכונות ומנכונות TopSort אנו יודעים שלכשנסיים טיפול בקודקוד מסוים – לא נחזור אליו יותר.
מכיוון שעל בסיס עובדה זו כתובה השגרה CalculateWeight – ניתן ישירות לראות בנכונותה, שכן אופן עדכון ותחזוק שדה המשקל - μ - ברור מאליו – אם קיים משקל מצטבר קטן יותר מהנוכחי – מתבצע עדכון.
כך גם ברור אופן תחזוק שדה הבנים - β (שהוא בהפוך לשדה האב - π, המוכר לנו מאלגוריתמים רבים אחרים).

■

שאלה:

נתונים גרף $G=(V, E)$ לא מכוון עם משקלים חיוביים על הקשתות, ונתונה תת קבוצה של צמתים $U\subseteq V$, ושני צמתים $u, v\in V$

מצא מסלול P מ-u ל-v כך ש:

א. P משתמש לפחות בצומת אחת מ-U.

ב. המשקל של P הוא מינימלי הין כל המסלולים המקיימים את סעיף א'.

פתרון:

תאור האלגוריתם:

1. נריץ את האלגוריתם של Dijkstra מקודקוד u, ונסמן את המרחקים המינימליים שנמצאו מ-u לכל $w\in V$ ב: $d_1[w]$.
2. נריץ את האלגוריתם של Dijkstra מקודקוד v, ונסמן את המרחקים המינימליים שנמצאו מ-v לכל $w\in V$ ב: $d_2[w]$.
3. נבחר $u'\in U$ המקיים:

$$d_1[u'] + d_2[u'] = \min_{u\in U} \{ d_1[u'] + d_2[u'] \}$$

4. המסלול המבוקש הוא: $u \rightsquigarrow u' \rightsquigarrow v$.

סיבוכיות:

1. Dijkstra: $O(m \log n)$.
2. Dijkstra: $O(m \log n)$.
3. מציאת המינימום: $O(n)$.

סה"כ: $O(n + m \log n)$, ואם G קשיר אז: $O(m \log n)$.

נכונות:

נניח בשלילה כי קיים $u'\neq w'\in U$ המקיים:

$$\delta(u, w') + \delta(w', v) < \delta(u, u') + \delta(u', v)$$

אזי מנכונות Dijkstra אנו מקבלים ישר סתירה לסעיף 3, שבו בחרנו את קודקוד u' ולא את w'.

■

<u>שאלה:</u>

נתונה רשת כבישים המתוארת על-ידי גרף G=(V, E) מכוון, עם משקלים אי-שליליים על הקשתות. משקל הקשת (u, v) מציין את המרחק מעיר u לעיר v.

משאית עם מטען נוסעת בכבישים באילוצים הבאים: גובה משאית ללא מטען הוא 2 מטר, וגובה משאית עם מטען הוא 4 מטר.

הצמתים V מסוגים לשני סוגים: צמתים שאם משאית עם מטען עוברת בהם, היא מורידה בהם את כל המטען, וצמתים רגילים, שאינם משנים את המטען במשאית.

הקשתות E מסווגות אף הן לשני סוגים: קשתות שיש בהן גשר שגובהו 3 מטר, וקשתות שאין בהם גשר (משאית עם מטען לא יכולה לעבור בקשר בה יש גשר).

הצע אלגוריתם, שבהינתן צמתים s, t∈V צמתים רגילים, מוצא את משקל המסלול הקצר ביותר (הקל ביותר) מ-s ל-t, כאשר המשאית יוצאת לדרכה מ-s.

<u>פתרון:</u>

<u>תאור האלגוריתם:</u>

נגדיר: 'E – קבוצת הקשתות ללא הגשרים.

נגדיר: 'V – קבוצת הקדקודים בהם המשאית פורקת את המטען.

1. ניצור גרף G'=(V, E').
2. לכל קודקוד v∈V', נחבר את קודקוד v מ-'G לקודקוד v שב-G, בקשת שמשקלה 0.
3. נריץ את האלגוריתם של Dijkstra מ-s על גרף 'G.

<u>סיבוכיות:</u>

1. יצירת הגרף: O(m + n).
2. מעבר על הקודקודים: O(n).
3. האלגוריתם של Dijkstra: O(m log n).

סה"כ: O(m log n).

<u>נכונות:</u>

משאית תוכל לנוע בכל הקשתות ובכל הצמתים אם היא ללא מטען. על-פי בנייה – גרף G מכיל את כל הצמתים ואת כל הקשתות.

משאית עם מטען יכולה לנוע רק בקשתות ללא גשרים, עד שתגיע לקודקוד בו היא תפרוק את מטענה. על פי בנייה – גרף 'G מכיל רק את הקשתות ללא הגשרים.

כמו-כן, לכל קודקוד ב-'G, שבו המשאית פורקת את מטענה, המשאית "מועברת" ישירות לגרף G (בקשת בעלת משקל 0).

האלגוריתם של Dijkstra ימצא לנו מסלול מינימלי, תוך כדי שהוא מתעלם מהקשת המחברת את גרף 'G ו-G (כי היא בעלת משקל 0).

■

שאלה:

פרופסור זיגזג נסע בתום סמסטר החורף לחופשת סקי. הוא רצה להחליק במורד מסלול סללום. המסלול מתחיל בנקודה המסומנת s, מסתיים בנקודה המסומנת t, ובמורד המסלול יש שורות של דגלים לרוחבו. מחליקים באופן הבא: בכל שורה עוקפים את אחד הדגלים, ומשנים את כיוון ההחלקה. בתחילת החופשה, פרופסור זיגזג מעוניין בנתיב החלקה מתון ככל האפשר, כלומר – בנתיב ארוך ככל האפשר (אורך הנתיב = סכום מרחקי הקטעים בין השורות). בסוף החופשה, הוא מעוניין במסלול תלול ככל האפשר, כלומר – בנתיב קצר ככל האפשר. תאר אלגוריתם יעיל לחישוב הנתיבים הללו.

פתרון:

תאור האלגוריתם:

נניח כי מיקומי הדגלים נתונים על-ידי קואורדינטות (x, y).

נשים לב, אם כן, שלדגלים באותה השורה יש את אותה קואורדינטה y.

כמו כן, באופן זה קל להבחין איזה דגל נמצא מימין לדגל מסוים, או משמאלו של דגל אחר.

1. נבנה גרף G=(V, E), כאשר:

   קבוצת הקדקודים: לכל שורה נבנה קדקודים כמספר הדגלים בשורה כפול 2. 2 קדקודים אלו עבור כל דגל יציינו האם הגענו לדגל זה מצד ימין או מצד שמאל.
   כמו כן, יהיו s ו-t קדקודים בודדים.

   קבוצת הקשתות: מ-s לכל הדגלים בשורה הראשונה. נשים לב שעבור כל דגל נשלח קשת רק לקודקוד המתאים לכיוון ההגעה הנכון, כלומר – מימין או משמאל. (קל לעשות זאת, בהסתמך על הקואורדינטות (x, y)).
   מכל קודקוד (המציין דגל, וכיוון ההגעה אליו), נחבר קשתות לקודקודים בשורה ה-i+1, כך שנגיע רק לקדקודים שניתן להגיע אליהם (בכיוון ההפוך מהקודקוד שבשורה i).
   מקדקודים בשורה האחרונה, אשר t נמצא בכיוון ההגעה של אותם קדקודים.
   כל קשת e=(u, v), תהיה ממושקלת, כך ש: המרחק בין קודקוד u לקודקוד v = w(e).
2. למציאת המסלול התלול ביותר (הקצר ביותר), נריץ את אלגוריתם Dijkstra על G, מ-s ל-t.
3. למציאת המסלול המתון ביותר (הארוך ביותר), נריץ את אלגוריתם Dijkstra*, המתואר להלן, על G מ-s ל-t:

Dijkstra(G, w, s)*

1. for each $v \in V$
2. $\quad d[v] \leftarrow -\infty$
3. $\quad \pi[v] \leftarrow NIL$
4. $d[s] \leftarrow 0$
5. $S \leftarrow \phi$
6. $Q \leftarrow V$
7. while ($Q \neq \phi$)
8. $\quad u \leftarrow ExtMax(Q)$
9. $\quad S \leftarrow S \cup \{u\}$
10. $\quad$ for each $v \in Adj[u]$
11. $\quad\quad$ if ($d[v] < d[u] + w(u, v)$)
12. $\quad\quad\quad d[v] \leftarrow d[u] + w(u, v)$
13. $\quad\quad\quad \pi[v] \leftarrow u$

סיבוכיות:

הקלט הנו, למעשה, מספר הדגלים והשורות. אנו ננתח סיבוכיות לפי m ו-n, האופן המוכר, כאשר נחשב תחילה את m ו-n:

n: נניח שקיימות i שורות של דגלים, ובכל שורה k_j דגלים. לכן:

$$n = 2 * \sum_{j=1}^{i} k_j + 2$$

m: מדגל מסוים בשורה ה-j, ניתן להחליק רק לחלק מהדגלים בשורה ה-j+1. ולכן:

$$m = k_1 + k_i + \sum_{j=1}^{i-1} k_j \cdot k_{j+1}$$

כעת ננתח:

1. בניית הגרף G: O(m + n).
2. Dijkstra O(m log n).
3. Dijkstra*: O(m log n).

סה"כ: O(m log n).

נכונות:

קל לראות שהגרף נבנה באופן שכל מסלול בו מ-s ל-t הינו מסלול החלקה חוקי, בהתאם להגבלות ולאופן ההחלקה שתואר בשאלה. נשים לב שהמשקלים הם חיוביים (מרחקים), ולכן שאר הוכחת הנכונות נובעת ישירות מנכונות Dijkstra.

■

שאלה:

פרופ׳ שושני נוסע לחופשה באילת בתום תקופת הבחינות. למכונית שלו מיכל דלק זעיר, המאפשר נסיעה למרחק 30 קילומטרים בלבד. בידיו מפה מדויקת של כבישים ותחנות דלק. המפה היא בצורת גרף מכוון וממושקל, כאשר הקשתות הן כבישים, המשקלות הם מרחקים, ועל גבי חלק מהצמתים יש תחנות דלק. עליך לספק לפרופ׳ שושני (שאינו בקיא באלגוריתמים בתורת הגרפים) אלגוריתם יעיל למציאת מסלול נסיעה באורך מינימלי מחיפה לאילת (שני צמתים בגרף), באופן שהמסלול לא יעבור יותר מ-30 קילומטרים בין תחנת דלק אחת לבאה. (ניתן להניח שפרופ׳ שושני מתחיל את מסעו עם מיכל מלא, ומוכן להגיע ליעדו עם מיכל ריק

פתרון:

תאור האלגוריתם:

1. נסמן את מפתו של פרופ׳ שושני כ-G, ונריץ על G את האלגוריתם Dijkstra*(G, w, s), המתואר להלן, מכל קודקוד $v \in V$:

Dijkstra(G, w, s)*

```
a.    for each  v∈V
b.       d[s, v] ← ∞
c.       π[s, v] ← NIL
d.    d[s, s] ← 0
e.    S ← ϕ
f.    Q ← V
g.    while  ( Q ≠ ϕ )
h.       u ← ExtractMin(Q)
i.       S ← S ∪ {u}
j.       for each  v∈Adj[u]
k.          if  ( d[s, v] > d[s, u] + w(u, v) )
l.               d[s, v] ← d[s, u] + w(u, v)
m.               π[s, v] ← u
```

2. נבנה גרף $G' = (V', E')$, כאשר:
   - V' - הצמתים המייצגים את חיפה, אילת, ואת כל תחנות הדלק.
   - E' - בין כל זוג צמתים תהיה קשת ממושקלת ומכוונת (u, v), אם״ם המרחק של המסלול מ-u ל-v הוא לכל היותר 30 קילומטרים.

   על-פי הגדרה זו בין כל זוג צמתים קיימות קשתות בשני הכיוונים - כלומר הקשת (u, v), והקשת (v, u). נוציא מכלל זה את צומת המקור (חיפה), אשר לה יהיו קשתות יוצאות בלבד, ואת צומת היעד (אילת), אשר לה יהיו קשתות נכנסות בלבד.

   משקל הקשתות ייצג את המרחק מ-u ל-v.
3. נריץ את האלגוריתם של Dijkstra על G'.

סיבוכיות:

1. ברור שסיבוכיות Dijkstra* זהה לזו של Dijkstra, ואנו מבצעים זאת על כל קודקוד: $O(n \cdot m \log n)$.
2. בניית G': $O(m + n)$.
3. Dijkstra: $O(m \log n)$.

סה״כ: מכיוון שבין כל זוג צמתים יהיו לכל היותר 2 קשתות (אחת בכל כיוון), הרי שסה״כ הסיבוכיות היא: $O(n \cdot m \log n)$.

נכונות:

מסלול טוב עובר דרך סדרה של תחנות דלק. המרחק שעובר פרופ׳ שושני בין זוג צמתים ״קריטיים״ עוקבים (מקור, יעד, או צומת שהיא תחנת דלק) ≤ 30 ק״מ.

בין כל זוג צמתים שכזה יש למצוא את המסלול הקל ביותר. לשם כך נריץ את אלגוריתם Dijkstra*, מכל קודקוד ב-V. הנכונות נובעת ישירות מנכונותו של האלגוריתם של Dijkstra ״הרגיל״.

בגרף שניצור G' המרחק בין כל זוג צמתים הוא לכל היותר 30, על-פי בניה.
כל שנותר הוא למצוא את המסלול הקל ביותר מצומת המקור (חיפה) לצומת היעד (אילת).

האלגוריתם של Dijkstra מבצע משימה זו בנאמנות. הנכונות נובעת ישירות מנכונותו של האלגוריתם של Dijkstra.

■

<u>שאלה:</u>

נתון גרף $G = (V, E)$ קשיר, לא מכוון, ממושקל. נתונה קבוצת קדקודים $U \subseteq V$. מצא בזמן $O(m \log n)$ עץ פורש מינימלי כך שקדקודי U יהיו בעלים של עץ זה.

<u>פתרון:</u>

<u>תאור האלגוריתם:</u>

1. ניצור גרף חדש $G' = (V', E')$, כאשר: $V' = V \setminus U$, ו- $E' = \{(u, v) \mid u \notin U \wedge v \notin U\}$.
2. נבנה עץ פורש מינימלי $T' = (V', E'_T)$ עבור הגרף G', תוך כדי שימוש באלגוריתם של Prim.
3. אם $|E'_T| \neq |V'|-1$, אזי אין עץ פורש מינימלי כנדרש.
4. לכל $u \in U$ תהי קשת $e=(u, v)$ המקיימת: $w(e)=\min_{t \notin U} w(u, t)$. נוסיף אותה ל-T'.

   אם קיים $u \in U$ עבורו לא קיימת אף קשת המחברת את u לקודקוד השייך ל-$V\setminus U$ – אזי אין עץ פורש מינימלי כנדרש.
5. העץ הנדרש הוא T'.

<u>סיבוכיות:</u>

1. בניית G': $O(m)$.
2. הפעלת האלגוריתם של Prim: $O(m \log n)$.
3. בדיקת תנאי: $O(1)$.
4. מעבר על הקשתות: $O(m)$.

סה"כ: $O(m \log n)$.

<u>נכונות:</u>

מנכונות האלגוריתם של Prim נובע כי T' הוא עץ פורש מינימלי של G'.

על-פי משפט, בעץ T' מתקיים: $|E'_T| = |V'|-1$. לכן – אם $|E'_T| \neq |V'|-1$, אזי T' אינו קשיר, ולכן אין עץ פורש מינימלי כנדרש.

נגדיר חתך $(U, V\setminus U)$. הקשתות E'_T שומרות על חוקיות החתך.

סעיף 4 באלגוריתם שקול לתהליכים הבאים:

א. הוספת קשת קלה $e=(u, v)$, $u \in U$, $v \in V\setminus U$, החוצה את החתך. הקשת e בטוחה עבור T (על-פי משפט).

ב. הגדרת חתך חדש: $(U\setminus\{u\}, (V\setminus U)\cup\{u\})$. הקשתות $E_t \cup \{e\}$ שומרות על חוקיות חתך זה.

ג. הוספת קשת קלה $e'=(u', v')$, $u' \in U\setminus\{u\}$, $v' \in (V\setminus U)\cup\{u\}$, החוצה את החתך. הקשת e בטוחה עבור T (על-פי משפט).

ד. הגדרת חתכים והוספת קשתות באופן המתואר לעיל.

משקילות של סעיף 4 לתהליך שתואר נובעת נכונות האלגוריתם.

■

שאלה:

נתונים גרף לא מכוון, קשיר וממושקל G, ועץ פורש מינימלי T בתוכו. מורידים מ-G קשת e=(u, v). מצא עץ פורש מינימלי בגרף G' = G \ e, בזמן O(m).

פתרון:

תאור האלגוריתם:

1. אם $e \notin T$ אזי T הוא גם העץ הפורש המינימלי של G'.
   אחרת:
2. T \ {e} הוא יער המורכב מ-2 עצים.
   נסמן: T_v את העץ ששורשו הוא v.
   T_u את העץ ששורשו הוא u.
   נפעיל BFS מקודקוד v על T \ {e}.
   נפעיל BFS^* מקודקוד u על T \ {e}.
   BFS^* הנו אלגוריתם הזהה לחלוטין לאלגוריתם ה-BFS "הרגיל", פרט לכך שהוא צובע את הקדקודים באדום במקום בשחור.
3. נעבור על כל הקשתות ב-G', ונחפש קשת e' = (i, j), שמשקלה מינימלי, וכן שקדקודיה, i ו-j צבועים בצבעים שונים (האחד אדום והשני שחור).
   אם לא נמצאת קשת כזו – לא קיים עץ פורש ב-G'.
   אם קיימת קשת כזו – אזי העץ הפורש המינימלי ב-G' הוא: $T' = T_v \cup T_u \cup \{e'\}$.

סיבוכיות:

1. בדיקת כל קשתות העץ: O(m).
2. הרצת BFS: O(m + n), ומכיוון שהגרף קשיר – O(m).
   הרצת BFS^*: O(m + n), ומכיוון שהגרף קשיר – O(m).
3. בדיקת כל הקשתות ב-G': O(m).

סה"כ: O(m).

נכונות:

ברור שאם $e \notin T$ אזי T הוא גם העץ הפורש המינימלי של G'.
הוצאת הקשת e=(u, v) מגדירה את החתך: (קדקודי T_u, קדקודי T_v) = (S, V\S).
נגדיר: A קבוצת הקשתות השייכות ל-T_u או ל-T_v. כלומר: $A = \{ e \in E(T) \mid (e \in T_v) \vee (e \in T_u) \}$.
ברור ש-A היא קבוצת קשתות השייכות לעץ פורש מינימלי כלשהו של G'.
על פי משפט (משפט 24.1 בעמוד 110 בספר "מבוא לאלגוריתמים") – A היא קבוצת קשתות השייכות לעץ פורש מינימלי כלשהו של G', ומכבדת את החתך (S, V\S) ⇐ הקשת הקלה e' = (i, j), החוצה חתך זה היא קשת בטוחה עבור A.
הוספת הקשת הבטוחה e' ל-A מחברת בין רכיבים זרים (שקדקודיהן צבועים בצבעים שונים – שחור או אדום) ⇐ A (לאחר הוספת e') מכילה n-1 קשתות, והגרף $G_A = (V, A)$ הנו חסר מעגלים.
⇐ הקשתות שב-A מהוות את העץ הפורש המינימלי של G': $T' = T_v \cup T_u \cup \{e'\}$.

■

שאלה:

נתון גרף קשיר ולא מכוון $G=(V, E)$ בעל משקולות חיוביים על הקשתות. מצא בזמן $O(m \log n)$ עץ פורש T של G, כזה שמכפלת משקולות הקשתות ב-T תהייה מקסימלית.

פתרון:

תאור האלגוריתם:

נפעיל את אלגוריתם $Prim^*$, הזהה לאלגוריתם של Prim (המתואר בעמוד 115, בספר "מבוא לאלגוריתמים"), פרט לכך שבמקום לבחור את המינימלי, נבחר את המקסימלי:

```
Prim*(G, w, r)
    Q ← V[G]
    for each vertex  u ∈ Q
          key[u] ← - ∞
    key[r] ← 0
    π[r] ← NIL
    while  Q ≠ ϕ
          u ← ExtractMax(Q)
          for each v ∈ Adj[u]
                if  (  ( v ∈ Q )  and  ( w(u, v)  >  key[v] )  )
                      π[v] ← u
                      key [v] ← w(u, v)
```

סיבוכיות:

ברור שהסיבוכיות זהה לסיבוכיות של Prim: $O(m \log n)$.

נכונות:

ברור שהאלגוריתם $Prim^*$ בונה עץ פורש מקסימלי, כלומר – ש**סכום** משקלי הקשתות הוא מקסימלי.

טענה:

הקשתות $e_1, e_2, \dots, e_n$ הן קשתותיו של עץ פורש של G כך ש: $w(e_1) + w(e_2) + \dots + w(e_n)$ הוא סכום מקסימלי של משקלי הקשתות אס"ם $w(e_1) \cdot w(e_2) \cdot \dots \cdot w(e_n)$ הוא מכפלה מקסימלית של משקלי קשתות עץ פורש ב-G.

הוכחת הטענה:

נגדיר פונקציית משקל חדשה 'w: $w'(e) = \log [w(e)]$. ניתן להגדיר פונקציה כזו, משום שנתון כי: $\forall e \in E,\ w(e) > 0$.

ברור כי $w(e_1) > w(e_2) \Leftrightarrow w'(e_1) > w'(e_2)$.

$\Leftarrow$ תהליך מיון הקשתות על פי פונקצית המשקל w, זהה לתהליך מיון הקשתות על פי פונקצית המשקל 'w.

$\Leftarrow$ על ידי שימוש באלגוריתם $Prim^*$, נקבל את אותו העץ הפורש מקסימלי, בין אם נשתמש בפונקצית המשקל 'w, ובין אם נשתמש בפונקצית המשקל w.

העץ הנפרש על-ידי שימוש באלגוריתם $Prim^*$ מקיים: $w(e_1) + w(e_2) + \dots + w(e_n)$ הוא סכום מקסימלי של משקלי קשתות עץ פורש ב-G, על פי פונקצית המשקל w. $\Leftarrow$ $w'(e_1) + w'(e_2) + \dots + w'(e_n)$ הוא סכום מקסימלי של משקלי קשתות עץ פורש ב-G, על פי פונקצית המשקל 'w.

נתבונן בסכום המשקלים הנ"ל:

$$\begin{aligned} w'(e_1) + w'(e_2) + \dots + w'(e_n) &= \log [w(e_1)] + \log [w(e_2)] + \dots + \log [w(e_n)] = \\ &= \log [w(e_1) \cdot w(e_2) \cdot \dots \cdot w(e_n)] \end{aligned}$$

כלומר - ברור שהסכום $w(e_1) + w(e_2) + \dots + w(e_n)$ הוא סכום מקסימלי של משקלי קשתות עץ פורש ב-G, על פי פונקצית המשקל w $\Leftarrow$ הסכום $w'(e_1) + w'(e_2) + \dots + w'(e_n)$ הוא סכום מקסימלי של משקלי קשתות עץ פורש ב-G, על פי פונקצית המשקל 'w $\Leftarrow$ $\log [w(e_1) \cdot w(e_2) \cdot \dots \cdot w(e_n)]$ הוא log של מכפלה מקסימלית של משקלי קשתות עץ פורש ב-G $\Leftarrow$ המכפלה $w(e_1) \cdot w(e_2) \cdot \dots \cdot w(e_n)$ היא מכפלה מקסימלית של משקלי קשתות עץ פורש ב-G.

■

שאלה:

נתון גרף $G=(V, E)$ לא מכוון, קשיר וממושקל. מצא עץ פורש בעל משקל מינימלי שני הכי טוב ב-$O(n^2)$.

פתרון:

תאור האלגוריתם:

1. נבנה עץ פורש מינימלי T על-ידי האלגוריתם של Kruskal.
2. נבנה טבלה M בגודל $n \times n$, כך שלכל זוג קדקודים $u, v \in V$, התא $M[u, v]$ מכיל את הקשת בעלת משקל מקסימלי במסלול מ-u ל-v בעץ הפורש מינימלי T.
את הטבלה M נבנה כך:
לכל $s \in V$ נריץ BFS^* על העץ T, מקודקוד מקור s.
BFS^* הוא אלגוריתם דומה לאלגוריתם BFS "הרגיל", עם השינוי הבא:
בנוסף לשדות ה-π וה-d, נוסיף שדה max, כך שלכל שכן לבן v של u נגדיר:
$$max(v) = \begin{cases} (u, v) & w(u, v) > w(max(u)) \\ max(u) & \text{אחרת} \end{cases}$$
בסיום BFS^* נקבל לכל קודקוד $v \in T$ את הקשת המקסימלית במסלול מ-s ל-v.
3. נבנה טבלה A בגודל $n \times n$, כך שלכל קשת $(u, v) \notin T$ נציב ב-$A[u, v]$ את ההפרש: $w(u, v) - w(M[u, v])$.
אם לא קיימת קשת שכזו – לא קיים עץ פורש שני, כנדרש.
4. נעבור על A, ונחפש (u, v) כך ש-$A[u, v]$ חיובי מינימלי.
אם לא קיים $A[u, v]$ חיובי – לא קיים עץ פורש שני, כנדרש.
5. נחליף את הקשת (u, v) עם הקשת $M[u, v]$.

סיבוכיות:

1. Kruskal: $O(m \log n)$.
2. בניית הטבלה M: $O(n^2)$.
3. בניית הטבלה A: $O(n^2)$.
4. מעבר על הטבלה: $O(n^2)$.
5. החלפת הקשת: $O(1)$.

סה"כ: $O(m + m \log n + n^2) = O(n^2)$ (כי הגרף קשיר).

נכונות:

$T = (V, E')$ הוא עץ פורש מינימלי של גרף G.
ברור שכדי לקבל עץ פורש T' של G המקיים: $w(T') = D + w(T)$.
כאשר $D > 0$, יש להחליף קשתות מ-E' עם קשתות מ-E\E', כך ש-T' ישאר עץ פורש על G.
בכדי לקבל T' עץ פורש מינימלי שני הכי טוב צריך למצוא את ההחלפה שתתן D חיובי ומינימלי.
ככל שנבצע יותר החלפות כאלה, המשקל הכללי של העץ החדש T' רק יעלה (החלפות מתבצעות עבור D <u>חיובי</u>), ולכן החלפה יחידה תניב את העץ הפורש המינימלי <u>השני</u> הכי טוב.
אם כן, מחפשים $e \in E'$, ו-$e' \in E \setminus E'$, כך שאם נחליף את e ב-e', נקבל עץ פורש T' שיתן D מינימלי וחיובי.
לכל $e' \in E \setminus E'$ מחפשים קשת $e \in E'$, כך ש:

1. החלפת e' עם e תיצור עץ פורש T' של G, ולכן e חייבת להיות על מעגל ש-e' סוגרת ב-T.
2. הקשת e צריכה להיות כזו שתתן הפרש מינימלי בין כל הקשתות שניתן להחליפן עם e', ולכן בוחרים את ה-e המקסימלית במעגל שהקשת e' סוגרת ב-T.

סעיפים 1 ו-2 של האלגוריתם מתבצעים לכל קשת השייכת ל-E\E', וכך מוצא את הקשתות $e' \in E \setminus E'$ ואת $e \in E'$, כך שהחלפת e ב-e' תיתן עץ פורש T' שיתן D מינימלי חיובי.
ולכן - T' הנוצר כתוצאה מהחלפה זו הוא עץ פורש מינימלי שני הכי טוב.

■

<u>שאלה:</u>

נתון גרף $G=(V, E)$ לא מכוון, קשיר וממושקל. מצא עץ פורש בעל משקל מינימלי שני הכי טוב ב-$O(m \log n)$.

<u>פתרון:</u>

<u>תאור האלגוריתם:</u>

1. נמצא עץ פורש מינימלי T, על-ידי הפעלת האלגוריתם של Kruskal.
2. נריץ BFS על T, החל מקודקוד $s \in V$ כלשהו.
3. לכל קשת $(u, v) \notin T$, נחפש את האב הקדמון המשותף של u ו-v ב-T. נשמור בשדה $w[v]$ את משקל הקשת המקסימלית במסלול מ-v לאב הקדמון המשותף של v ו-u.

   לכל קשת $(u, v) \notin T$ נשמור את תוספת המשקל שיתקבל כשנוסיף את (u, v) ל-T, ונוציא את הקשת בעלת המשקל המקסימלי במעגל שייסגר על-ידי הוספת (u, v).
4. נבחר את הקשת כך שהתוספת שתתקבל היא חיובית קטנה ביותר.

<u>סיבוכיות:</u>

1. Kruskal: $O(m \log n)$.
2. BFS על עץ: $O(m)$.
3. בדיקת הקשתות ושמירת המקסימום: $O(m)$.
4. בחירת קשת: $O(m)$.

סה"כ: $O(m \log n)$.

<u>נכונות:</u>

Kruskal מוצא עץ פורש מינימלי – ידוע.

ברור כי עץ פורש מינימלי שני הכי טוב הוא עץ $\{(x, y)\} \cup (T \setminus \{(u, v)\})$, המתקבל על-ידי שני הקשתות: $(u, v) \in T$, ו-$(x, y) \notin T$, כך שתוספת המשקל המתקבלת על-ידי ההפרש: $w[u, v]–w[x, y]$ היא חיובית ומינימלית.

נניח בשלילה כי קיימות קשתות $e, e' \in T$, $f, f' \notin T$, כך שהאלגוריתם מציע את העץ $(T \setminus e) \cup f$.

אבל גם $(T \setminus e') \cup f'$ הוא עץ פורש כך שמתקיים:

$$w(T) < w[(T \setminus e') \cup f'] < w[(T \setminus e) \cup f]$$

כלומר – $(T \setminus e') \cup f'$ עץ פורש שני הכי טוב, שלא נמצא על-ידי האלגוריתם.

אזי:

$$w(T) < w(T) – w(e') + w(f') < w(T) – w(e) + w(f)$$
$$\Rightarrow \; 0 < w(f') – w(e') < w(f) – w(e)$$

בסתירה למינימליות של תוספת המשקל $w(f) - w(e)$.

■

<u>שאלה:</u>

נתון גרף קשיר לא מכוון $G=(V, E)$ ממושקל, ועץ פורש מינימלי T בו. תן אלגוריתם המוצא עץ פורש מינימום 'T ב-G, כך שמספר הקשתות המשותפות ל-T ול-'T יהיה מינימלי, בזמן $O(m \log m)$.

<u>פתרון:</u>

<u>תאור האלגוריתם:</u>

n. נמיין את הקשתות בסדר עולה על פי משקלן. אם $w(e_i) = w(e_k)$, אזי נקבע כי אם $e_i \in T$, ו-$e_k \notin T$, אזי e_k יופיעו לפני e_i (כלומר – אם קיימות קשתות שוות משקל, אזי הקשתות השייכות ל-T יופיעו במיון אחרי קשתות שלא שייכות ל-T).

o. נריץ את אלגוריתם Kruskal על הקשתות הממוינות (על פי סעיף 1), ונקבל עץ פורש מינימלי חדש 'T.

<u>סיבוכיות:</u>

1. מיון (QuickSort או MergeSort): $O(m \log m)$.
2. האלגוריתם של Kruskal: $O(m \log m)$.

סה"כ: $O(m \log m)$.

<u>נכונות:</u>

נראה כי לא קיים עץ פורש מינימלי "T המתקבל מ-'T על ידי החלפת קשת $(e \in T' \wedge e \in T)$, בקשת $((e' \notin T \vee e' \notin T') \wedge e' \in T'')$ (כלומר – על-ידי החלפת קשת e השייכת ל-T ול-'T בקשת 'e השייכת ל-"T ולא שייכת ל-'T או ל-T).

נניח בשלילה שקיימת $e \neq e'$ כזו.

<u>טענת עזר 1:</u>

$w(e) = w(e')$

<u>הוכחה:</u>

אם $w(e') > w(e)$, זו סתירה לכך ש-"T הוא עץ פורש מינימלי.

אם $w(e') < w(e)$, זו סתירה לכך ש-T ו-'T הם עצים פורשים מינימליים.

<u>טענת עזר 2:</u>

קשת 'e סוגרת מעגל עם e ב-'T.

<u>הוכחה:</u>

נתבונן באופן בו האלגוריתם בונה את 'T. במיון הקשתות, הקשת 'e מופיעה לפני הקשת e.

כאשר האלגוריתם בוחן את 'e, מתקיים כי 'e <u>אינה</u> סוגרת מעגל (כי e סוגרת מעגל עם e ב-'T, אבל e עוד לא נבחרה).

$\Leftarrow$ 'e תבחר ל-'T.

סתירה להנחה ש-'e לא שייכת ל-'T.

■

שאלה:

נתון גרף קשיר לא מכוון $G=(V, E)$ ממושקל, ועץ פורש מינימלי T בתוכו. מוסיפים קשת (u, v) חדשה ל-G. מצא עץ פורש מינימלי 'T בגרף $G' = G \cup \{(u, v)\}$, בזמן $O(n)$.

פתרון:

תאור האלגוריתם:

1. נריץ את אלגוריתם DFS^* מ-u על $T \cup \{(u, v)\}$.
   DFS^* דומה לאלגוריתם DFS "הרגיל", פרט לשינויים הבאים:

$DFS^*(G, s)$

```
1.  for each vertex w∈V[G]
2.        color[w] ← WHITE
3.        π[w] ← NIL
4.  time  ← 0
5.  DFS-VISIT*(s)
```

$DFS\text{-}VISIT^*(w)$

```
1.  color[w] ← GRAY
2.  d[w] ← time ← time + 1
3.  for each z∈Adj[w]
4.        if  ( z = s )
5.              stop
6.        if  ( color[z] = WHITE )
7.              π[z] ← w
8.              DFS-VISIT*(z)
9.  color[w] ← BLACK
10. f[w] ← time ← time + 1
```

2. נריץ את אלגוריתם FindMax(w): (w הוא הקודקוד ש"גילה מחדש" את u)

FindMax(s)

```
1.  max ← 0
2.  while (s ≠ u)
3.        if (max < w(s, π[s])
4.              max ← w(s, π[s])
5.        s ← π[s]
```

3. אם $max \le w(u, v)$, אזי 'T הוא T.
   אם $max > w(u, v)$, אזי נחליף את הקשת בעלת המשקל max עם הקשת (u, v) ב-T, ונקבל את 'T.

סיבוכיות:

1. ברור שהסיבוכיות זהה לסיבוכיות של DFS "הרגיל": $O(m + n)$.
2. ריצה על הקדקודים על-פי האב: $O(n)$.
3. בדיקה: $O(1)$.

סה"כ: $O(m + n)$.

נכונות:

ברור שהקשת (u, v) סוגרת מעגל ב-T $\Leftarrow$ אם קיימת קשת e במעגל זה בעלת משקל הגדול מ-$w(u, v)$, האלגוריתם יחליף בין e ו-(u, v). כך יתקבל עץ 'T שמשקלו קטן ממשקל T.

נותר להוכיח שאין קשת השייכת ל-G\T' הניתנת להחלפה בקשת השייכת ל-'T.

נניח שאכן בוצעה ההחלפה בסעיף 3. אזי: $G\backslash T' = (G\backslash T) \cup \{e\}$. $\Leftarrow$ די להוכיח שלא קיימת קשת השייכת ל-G\T אשר אם נחליפה בקשת השייכת ל-'T, נקבל עץ "T בעל משקל קטן יותר.

כל קשת השייכת ל-G\T סוגרת מעגל ב-'T (וכן סוגרת מעגל ב-T). תהי קשת $e=(i, j)$ השייכת ל-G\T. אם e סוגרת מעגל ב-'T שאינו מכיל את הקשת (u, v), אזי ברור שלא ניתן להחליף את e עם קשת השייכת ל-'T (מכיוון ש-T הוא עץ פורש מינימלי).

אם e סוגרת מעגל ב-'T המכיל את הקשת (u, v), אזי e סוגרת מעגל ב-T, המכיל את המסלול מ-u ל-v.

$\Leftarrow$ $w(e) \ge$ ממשקל כל קשת במסלול מ-i ל-j ב-T. (אחרת e הייתה שייכת ל-T). כמו כן, $w(u, v) >$ מהמשקל המקסימלי של הקשתות במסלול "העוקף" מ-u ל-v ב-T (על פי סעיף 3).

$\Leftarrow$ $w(e) \ge$ ממשקל של כל קשת במסלול מ-i ל-j ב-'T.

$\Leftarrow$ אם נחליף את e בקשת אחרת השייכת למסלול מ-i ל-j, ***לא*** נפחית ממשקלו של 'T.

■

שאלה:

נגדיר עדיפות של גרף $G=(V, E)$ באופן הבא: $pri(G) = \prod_{e \in E} w(e)$

בהינתן גרף סופי, לא מכוון עם משקלים על הקשתות, כך שלכל קשת e מתקיים ש: $0 < w(e) < 1$, תן אלגוריתם שמחזיר עץ פורש בעל עדיפות מקסימום.

פתרון:

תאור האלגוריתם:

נריץ את אלגוריתם $Prim^*(G, w, r)$, המתואר כלהלן:

Prim(G, w,r)*

```
Q ← V[G]
for each  u ∈ Q
      key[u] ← -∞
key[r] ← 0
π[r] ← NIL
while  Q ≠ 0
      u ← ExtractMax(Q)
      for each  v ∈ Adj[u]
            if  ( ( v ∈ Q )  and  ( w(u, v) > key[v] ) )
                  π[v] ← u
                  key[v] ← w(u, v)
```

סיבוכיות:

ברור שהסיבוכיות זהה לסיבוכיות של האלגוריתם של Prim "הרגיל": $O(m \log n)$.

נכונות:

ברור שהאלגוריתם $Prim^*$ בונה עץ פורש מקסימלי, כלומר – ש**סכום** משקלי הקשתות הוא מקסימלי.

טענה:

הקשתות $e_1, e_2, \ldots, e_n$ הן קשתותיו של עץ פורש של G כך ש: $w(e_1) + w(e_2) + \ldots + w(e_n)$ הוא סכום מקסימלי של משקלי הקשתות אם"ם $w(e_1) \cdot w(e_2) \cdot \ldots \cdot w(e_n)$ הוא מכפלה מקסימלית של משקלי קשתות עץ פורש ב-G.

הוכחת הטענה:

נגדיר פונקצית משקל חדשה w': $w'(e) = \log [w(e)]$. ניתן להגדיר פונקציה כזו, משום שנתון כי: $\forall e \in E,\ w(e) > 0$.

ברור כי $w(e_1) > w(e_2) \Leftrightarrow w'(e_1) > w'(e_2)$.

$\Leftarrow$ תהליך מיון הקשתות על פי פונקצית המשקל w, זהה לתהליך מיון הקשתות על פי פונקצית המשקל w'.

$\Leftarrow$ על ידי שימוש באלגוריתם $Prim^*$, נקבל את אותו העץ הפורש מקסימלי, בין אם נשתמש בפונקצית המשקל w', ובין אם נשתמש בפונקצית המשקל w.

העץ הנפרש על-ידי שימוש באלגוריתם $Prim^*$ מקיים: $w(e_1) + w(e_2) + \ldots + w(e_n)$ הוא סכום מקסימלי של משקלי קשתות עץ פורש ב-G, על פי פונקצית המשקל w. $\Leftarrow$ $w'(e_1) + w'(e_2) + \ldots + w'(e_n)$ הוא סכום מקסימלי של משקלי קשתות עץ פורש ב-G, על פי פונקצית המשקל w'.

נתבונן בסכום המשקלים הנ"ל:

$$w'(e_1) + w'(e_2) + \ldots + w'(e_n) = \log [w(e_1)] + \log [w(e_2)] + \ldots + \log [w(e_n)] = \log [w(e_1) \cdot w(e_2) \cdot \ldots \cdot w(e_n)]$$

כלומר - ברור שהסכום $w(e_1) + w(e_2) + \ldots + w(e_n)$ הוא סכום מקסימלי של משקלי קשתות עץ פורש ב-G, על פי פונקצית המשקל w $\Leftarrow$ הסכום $w'(e_1) + w'(e_2) + \ldots + w'(e_n)$ הוא סכום מקסימלי של משקלי קשתות עץ פורש ב-G, על פי פונקצית המשקל w' $\Leftarrow$ $\log [w(e_1) \cdot w(e_2) \cdot \ldots \cdot w(e_n)]$ הוא log של מכפלה מקסימלית של משקלי קשתות עץ פורש ב-G $\Leftarrow$ המכפלה $w(e_1) \cdot w(e_2) \cdot \ldots \cdot w(e_n)$ היא מכפלה מקסימלית של משקלי קשתות עץ פורש ב-G.

■

<u>שאלה:</u>

נתון גרף G=(V, E) לא מכוון, פשוט וקשיר, עם משקולות אי-שליליים. נתונה קשת $e \in E$. כתוב אלגוריתם המוצא מבין העצים המכילים את הקשת e את העץ הפורש המינימלי (כלומר – למצוא עץ פורש מינימלי המכיל את הקשת e).

<u>פתרון:</u>

<u>תאור האלגוריתם:</u>

נריץ על G את אלגוריתם $Kruskal^*$, המתואר להלן:

<u>Kruskal(G, w)</u>*

1. $A \leftarrow \phi$
2. for each vertex $v \in V$
3. MakeSet(v)
4. sort the edges of E by nondecreasing weight w, *<u>BUT</u>* place the edge e first.
5. for each edge $(u, v) \in E$
6. if (FindSet(u) $\neq$ FindSet(v))
7. $A \leftarrow A \cup \{ (u, v) \}$
8. return A

<u>סיבוכיות:</u>

ברור שהסיבוכיות זהה לזו של אלגוריתם Kruskal "הרגיל": $O(m \log n)$.

<u>נכונות:</u>

נובעת ישירות מנכונות Kruskal.

■

<u>שאלה:</u>

נתון $G=(V, E)$ גרף לא מכוון, ממושקל, ועץ פורש מינימלי T בתוכו. מוסיפים ל-G קודקוד חדש $w\in V$, ושתי קשתות: (w, u), (w, v). כאשר $u, v\in V$. הצע אלגוריתם המוצא עץ פורש מינימלי בגרף החדש ב-$O(m)$.

<u>פתרון:</u>

<u>תאור האלגוריתם:</u>

1. נריץ BFS על העץ T (לפני הוספת הקודקוד והקשתות), מ-u ועד v. במהלך ריצת ה-BFS, נזכור את הקשת הכבדה ביותר במסלול היחיד בין u ל-v, ונסמנה (x, y).
2. נוסיף את הקודקוד והקשתות, ונשווה בין משקלי הקשתות (x, y), (w, v), (w, u):
   א. אם הקשת (x, y) אינה הכבדה ביותר – נוסיף ל-T את הקשת הקלה מבין (w, v) ו-(w, u).
   ב. אם הקשת (x, y) היא הכבדה ביותר – נוציא אותה מהעץ T, ונוסיף את שתי הקשתות (w, v) ו-(w, u).
   (כל שאר הקשתות של T יהיו בעץ הפורש החדש).

<u>סיבוכיות:</u>

1. BFS שמתבצע על עץ: $O(m)$.
2. השוואה: $O(1)$.
   א. השוואה בין הקשתות, והוספת הקשת הקלה: $O(1)$
   ב. הוצאת קשת מהעץ, והוספת שתי הקשתות: $O(1)$.

סה"כ: $O(m)$.

<u>נכונות:</u>

<u>טענה 1:</u>

הגרף T החדש המתקבל בסיום האלגוריתם הוא עץ.

<u>הוכחה:</u>

ברור שבסיום האלגוריתם T יהיה עץ, כי $T \cup \{(w, u), (w, v)\}$ מכיל מעגל אחד בלבד, אבל אחת מקשתותיו של מעגל זה לא ימצאו בעץ החדש (על-פי סעיף 2).

לכן – T החדש יהיה גרף חסר מעגלים. אבל – T החדש גם יכיל את כל קדקודי G, ולכן יהיה קשיר.

$\Leftarrow$ T החדש הוא אכן עץ.

<u>טענה 2:</u>

העץ T החדש המתקבל בסיום האלגוריתם הוא עץ פורש מינימלי.

<u>הוכחה:</u>

לאחר הוספת הקודקוד והקשתות – נוצר מעגל. מתוך המעגל הוצאנו את הקשת הכבדה ביותר, ולכן ברור שסכום הקשתות באופן זה יהיה המינימלי.

מטענות 1 ו-2 נובע כי הגרף T החדש שמתקבל בסיום האלגוריתם הוא עץ פורש מינימלי של הגרף G עם איחוד הקודקוד w והקשתות (w, v) ו-(w, u).

■

שאלה:

נתון גרף סופי G=(V, E) לא מכוון, קשיר, וממושקל במשקלים אי-שליליים.

נגדיר: *עץ קל ביותר ביחס ל-s* כעץ פורש של G בו המסלול מ-s לכל צומת v הינו בעל משקל מינימום מבין כל המסלולים מ-s ל-v ב-G.

הצע אלגוריתם המוצא, בהינתן G ו-s עץ קל ביותר ביחס ל-s.

פתרון:

תאור האלגוריתם:

נריץ על הגרף, החל מ-s, את אלגוריתם $Prim^*$ המתואר להלן:

$Prim^*(G, w, s)$

```
1.  Q ← V
2.  for each  u∈Q
3.      λ[u] ← ∞
4.  λ[s] ← 0
5.  π[s] ← NIL
6.  while  ( Q ≠ φ )
7.      v ← ExtractMin(Q)
8.      for each  u∈Adj[v]
9.          if   ( ( u ∈ Q )  and  ( λ[u] > λ[v] + w[(v, u)] ) )
10.             λ[u] ← λ[v] + w[(v, u)]
11.             π[u] ← v
```

הרעיון:

$\lambda[v]$ יחזיק את במסלול הקל ביותר מ-s ל-v, כך שבחירת הצמתים שעדיין לא נכנסו לעץ הפורש תהיה על-פי מרחקם מ-s.

סיבוכיות:

ברור שהסיבוכיות זהה לזו של Prim "הרגיל": $O(m \log n)$.

נכונות:

טענה:

בכל פעם שמוסיפים צומת חדשה לעץ הנפרש, מרחקה של צומת זו מ-s הוא מינימלי מבין הצמתים שעדיין לא נבחרו.

הוכחה:

באינדוקציה על סדר כניסת הצמתים:

בסיס: הצומת הראשונה שנכנסת לעץ הנפרש היא s, ואכן מרחקה מעצמה – 0, הוא מינימלי.

נניח: נכונות הטענה עבור כל הצמתים שהוכנסו לפני הצומת ה-n-ית.

צעד: נראה שהצומת ה-n-ית בסדר הכניסה לעץ הנפרש היא בעלת המסלול הקל ביותר מבין כל הצמתים שעדיין לא נכנסו לעץ הנפרש.

נסמן צמת זו כ-v.

נניח בשלילה שקיימת צומת u שמרחקה מ-s קטן יותר, והיא לא נבחרה לפני v.

תהי 'u הצומת הקודמת ל-u במסלול הקל ביותר.

מכיוון שמסלול זה קצר יותר מהמסלול מ-s ל-v, בהכרח שמתקיים: $\lambda[u'] < \lambda[v]$.

כלומר - 'u כבר נכנס לעץ הנפרש, ולפי הנחת האינדוקציה הוא היה בעל ערך λ מינימלי מבין כל הצמתים באותו הרגע, וגם קבע את הסימון: $\lambda[u] \leftarrow \lambda[u'] + w[(u', u)]$.

כלומר - $\lambda[u]$ נכנסה לעץ הנפרש בטרם טיפלנו בצומת v, **בסתירה להנחה!**

■

<u>שאלה:</u>

יהי G גרף לא מכוון, קשיר וממושקל. נתון: $\left| |E| - |V| \right| \le 100$.
תאר אלגוריתם לינארי ב-$|V|$, המוצא עץ פורש מינימלי ב-G.

<u>פתרון:</u>

<u>תאור האלגוריתם:</u>

נריץ DFS על G, עם השינויים הבאים:
תוך כדי הסריקה, נזכור ב-max_edge את הקשת בעלת המשקל המקסימלי (המשתנה יהיה קיים לכל דרגה ברקורסיה).
אם נגיע לקודקוד אפור - נוריד את הקשת המקסימלית שזכרנו ב-max_edge.
כך נבצע 101 פעמים לכל היותר.

<u>סיבוכיות:</u>

$101 \cdot O(n) = O(n)$

<u>נכונות:</u>

נריץ 101 סריקות DFS, כי קיימים 100 מעגלים, ועוד מסלול פשוט).
לאחר כל סריקות ה-DFS – בעצם הורדנו את כל המעגלים שבגרף (100).
אם נבצע 101 הורדות, נקבל $|E| = |V| - 1$, וזה הרי מתקיים עבור עץ.
נותר להראות שזהו עץ פורש מינימלי:
במקום לנקוט באופן "הסטנדרטי" של בניית העץ הפורש על-ידי הוספת קשתות מינימליות, אנו פשוט מורידים קשתות מקסימליות.
בכל פעם נוריד קשת יחידה, ולכן יש לבצע 101 סריקות DFS.

■

<u>שאלה:</u>

נתון גרף לא מכוון, קשיר וממושקל $G=(V, E)$. כל קשת צבועה בכחול או בלבן. תאר אלגוריתם הבודק האם קיים עץ פורש מינימלי ללא הקשתות הכחולות והלבנות. כל הקשתות ממשקל 10 או פחות הן כחולות, וכל הקשתות ממשקל 20 ויותר הן לבנות. אין משמעות לשאר הצבעים של שאר הקשתות.

<u>פתרון:</u>

<u>תאור האלגוריתם:</u>

1. נריץ על G את האלגוריתם של Kruskal, למציאת עץ פורש מינימלי T.
   נסמן: $w(T)$ – המשקל של העץ הפורש מינימלי.
2. נסיר מ-G את הקשתות הלבנות שמשקלן קטן/שווה ל-10, ואת הקשתות הכחולות שמשקלן גדול/שווה ל-20.
   נסמן גרף זה כ-'G.
3. נריץ על 'G את האלגוריתם של Kruskal.
   אם לא נצליח לבנות עץ פורש – אזי לא קיים עץ כנדרש.
   אחרת - נסמן את העץ 'T.
4. אם $w(T') = w(T)$, אזי 'T הוא העץ כנדרש.
   אחרת – לא קיים עץ כנדרש.

<u>סיבוכיות:</u>

1. Kruskal: $O(m \log n)$.
2. בניית 'G: $O(m + n)$.
3. Kruskal: $O(m \log n)$.
4. בדיקה: $O(1)$.

סה"כ: $O(m \log n)$.

<u>נכונות:</u>

הנכונות נובעת ישירות מנכונות Kruskal.

■

שאלה.
במסיבה כלשהי יש n אנשים; חלקם לחץ ידיים זה עם זה. אנא הוכח שמספר האנשים שלחצו ידיים עם מספר אי-זוגי של אנשים - הוא זוגי.

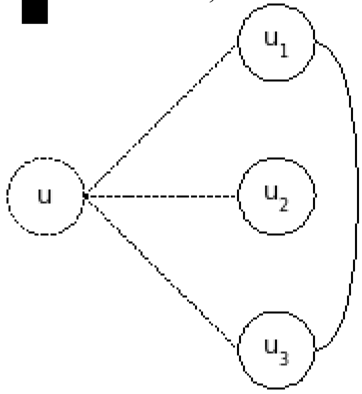
דוגמה:
נניח שבמסיבה יש שלשה אנשים: 1, 2, ו3, ורק 1 ו2 לחצו ידיים זה עם זה. יש שני אנשים שלחצו ידיים עם אדם אחד; אכן יש מספר זוגי (שניים) של אנשים שלחצו ידיים עם מספר אי-זוגי (אחת) של אנשים.

רמז
צייר גרף שבו כל אדם הוא צומת. מה יקרה אם תמחק צומת כלשהו והקשתות היוצאות ממנו?

תשובה
1. נוכיח את הטענה באינדוקציה על n, מספר הצמתים.
(בסיס האינדוקציה) אם בגרף יש צומת יחיד, אין קשתות. אם בגרף יש שני צמתים, אז או שיש קשת אחת, או שאין אף קשת. בכל אחד ממקרים אלה, בדיקה פשוטה מראה שהטענה מתקיימת. לכן הטענה נכונה עבור n=1,2 .
(מעבר האינדוקציה) נניח שהטענה נכונה עבור כל גרף בעל n-1 צמתים, ונראה שהטענה נכונה עבור כל גרף בעל n צמתים. ניקח גרף בעל n צמתים, ונחפש צומת u שמספר שכניו אי-זוגי. (אם אין צומת כזה, הטענה

בהכרח נכונה לגרף זה.) נניח ששכניו של u הם $u_1, u_2,\ldots,u_i$. נשים לב ש i בהכרח אי-זוגי. אם נמחק את uוהקשתות שיוצאות ממנו, הטענה נכונה מהנחת האינדוקציה. לאחר הוספת u מחדש, כמה צמתים בעלי דרגה אי-זוגית נוצרו ? u עצמו תורם 1. כל שכן של u בעל דרגה זוגית הופך להיות בעל דרגה אי-זוגית, וכל שכן של u בעל דרגה אי-זוגית הופך להיות בעל דרגה זוגית. היות שi אי-זוגי, תרומת שכניו היא מספר אי-זוגי (חיובי או שלילי). לכן נוצר עוד מספר זוגי (חיובי או שלילי) של צמתים בעלי דרגה זוגית.

דוגמה:
בתרשים הבא, יש ל u 3 שכנים: 2 בעלי דרגה אי-זוגית (בלעדיו), ו1 בעל דרגה זוגית (בלעדיו).



לאחר הוספת u ,יש לu 2 שכנים בעלי דרגה אי-זוגית (יחד אתו), ושכן יחיד בעל דרגה זוגית (יחד אתו). הוספת u ייצרה עוד 0 צמתים בעלי דרגה אי-זוגית : u עצמו בעל דרגה אי-זוגית , u1 הפך להיות בעל דרגה זוגית , u2 הפך להיות בעל דרגה אי-זוגית וu3 הפך להיות בעל דרגה זוגית.

. במסיבה כלשהי יש n אנשים; חלקם לחץ ידיים זה עם זה. אנא הוכח שמספר האנשים שלחצו ידיים עם מספר אי-זוגי של אנשים - הוא זוגי.

דוגמה:
נניח שבמסיבה יש שלשה אנשים: 1, 2, ו3, ורק 1 ו2 לחצו ידיים זה עם זה. יש שני אנשים שלחצו ידיים עם אדם

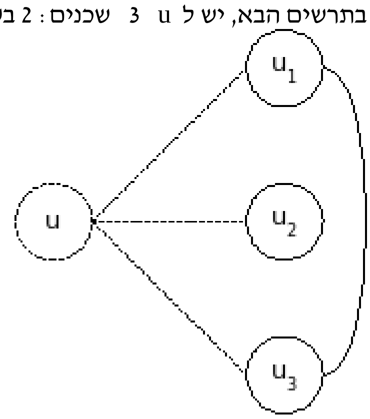
אחד ; אכן יש מספר זוגי (שניים) של אנשים שלחצו ידיים עם מספר אי-זוגי (אחת) של אנשים.

רמז
צייר גרף שבו כל אדם הוא צומת. מה יקרה אם תמחק צומת כלשהו והקשתות היוצאות ממנו?

תשובות
1. נוכיח את הטענה באינדוקציה על n, מספר הצמתים.
(בסיס האינדוקציה) אם בגרף יש צומת יחיד, אין קשתות. אם בגרף יש שני צמתים, אז או שיש קשת אחת, או שאין אף קשת. בכל אחד ממקרים אלה, בדיקה פשוטה מראה שהטענה מתקיימת. לכן הטענה נכונה עבור $n=1,2$.
(מעבר האינדוקציה) נניח שהטענה נכונה עבור כל גרף בעל $n-1$ צמתים, ונראה שהטענה נכונה עבור כל גרף בעל n צמתים. ניקח גרף בעל n צמתים, ונחפש צומת u שמספר שכניו אי-זוגי. (אם אין צומת כזה, הטענה בהכרח נכונה לגרף זה.) נניח

ששכניו של u הם $u_1, u_2,\ldots,u_i$. נשים לב ש i בהכרח אי-זוגי. אם נמחק את u והקשתות שיוצאות ממנו, הטענה נכונה מהנחת האינדוקציה. לאחר הוספת u מחדש, כמה צמתים בעלי דרגה אי-זוגית נוצרו ? u עצמו תורם 1. כל שכן של u בעל דרגה זוגית הופך להיות בעל דרגה אי-זוגית, וכל שכן של u בעל דרגה אי-זוגית הופך להיות בעל דרגה זוגית. היות שi אי-זוגי, תרומת שכניו היא מספר אי-זוגי (חיובי או שלילי). לכן נוצר עוד מספר זוגי (חיובי או שלילי) של צמתים בעלי דרגה זוגית.

דוגמה:
בתרשים הבא, יש ל u 3 שכנים : 2 בעלי דרגה אי-זוגית (בלעדיו), ו1 בעל דרגה זוגית (בלעדיו).



לאחר הוספת u ,יש לו 2 שכנים בעלי דרגה אי-זוגית (יחד אתו), ושכן יחיד בעל דרגה זוגית (יחד אתו). הוספת u ייצרה עוד 0 צמתים בעלי דרגה אי-זוגית : u עצמו בעל דרגה אי-זוגית , $u1$ הפך להיות בעל דרגה זוגית , $u2$ הפך להיות בעל דרגה אי-זוגית ו$u3$ הפך להיות בעל דרגה זוגית.

תרגיל:
נתון גרף לא מכוון וקשיר $G = (V, E)$ פונקציית משקל על הקשתות $\boldsymbol{w}: \boldsymbol{E} \to \mathbb{N}$. כל קשת צבועה בצהוב או בשחור. הציעו אלגוריתם המוצא את העפ"מ הצהוב ביותר (כלומר עם מספר הקשתות הצהובות הגדול ביותר).

פיתרון:
נרצה לתת לקשתות צהובות עדיפות על פני קשתות שחורות מאותו משקל.
$n = |V|$
נגדיר w' חדשה:

$$w'(e) = \begin{cases} w(e) - \frac{1}{n} & e \text{ is yellow} \\ w(e) & otherwise \end{cases}$$

נמצא עפ"מ T לפי w' באמצעות *Prim*.

נוכיח ש-T העפ"מ הצהוב ביותר לפי w אם"ם T הוא עפ"מ לפי w':
נסמן ב-$y(T)$ את מספר הקשתות הצהובות בעפ"מ T. מתקיים ש-

$$w'(T) = w(T) - \frac{y(T)}{n}$$

- נשים לב ש-w תמיד מחזיריה ערכים שלמים לכן $w(T)$ שלם לכל T.

ראשית ניקח 2 עצים $T1,T2$ כך ש- $w(T_1) > w(T_2)$, אז:

$$w'(T_1) - w'(T_2) = \underbrace{w(T_1) - w(T_2)}_{\geq 1} - \frac{1}{n}\underbrace{(y(T_1) - y(T_2))}_{<1} > 0$$

ולכן:

$$w(T_1) > w(T_2) \implies w'(T_1) > w'(T_2)$$

כלומר המונוטוניות של משקלי העצים נשמרת ללא קשר לצבע הקשתות. עץ שלא היה עפ"מ לפי w' לא יכול להיות עפ"מ לפי w רק בגלל שיש לו הרבה קשתות צהובות, כי למרות עדיפותן על השחורות, השפעתן לא מספיקה, בגלל הפקטור הנמוך שהוספנו.

שנית נבחין שאם $w(T_1) = w(T_2)$, אז $w'(T_1) < w'(T_2)$ אם"ם $y(T_1) > y(T_2)$
כלומר כאשר המשקל זהה, העפ"מ הצהוב ביותר יהיה בעל מספר הקשתות הצהובות הגדול ביותר.

← מכאן שקבוצת העפ"מים של G לפי w' היא תת-קבוצה של העפ"מים לפי w עם מספר מקסימלי של קשתות צהובות.

שאלה 51 (**מרצים: פרופ' יוסי עזר , פרופ' רון שמיר , מתרגלים: ידעאל ולדמן , אדם שפר**)
נתון גרף קשיר, לא מכוון וממושקל $G=(V,E)$. תארו אלגוריתם יעיל אשר בודק האם ל- G יש עפ"מ יחיד. מספיק למצוא אלגוריתם שסיבוכיות הזמן שלו הינה $O(|E|\log|V|)$.

יעילות: של האלגוריתם של פרים

אלגוריתם והסבר:
נמצא עץ פורש מינימלי.
כעת נמצא שוב עץ פורש מינימלי בעזרת האלגוריתם של פרים, כאשר כרגיל נתן עדיפות לכל קשת על פני קשתות כבדות ממנה, אך נתן גם עדיפות לכל קשת שלא היתה בעץ הראשון על פני קשת בעלת משקל שווה לה שהיתה בעץ הראשון. אם"ם באיזשהו שלב תבחר לעץ קשת שלא היתה בעץ הראשון אז נדע שיש יותר מעץ פורש מינימלי יחיד. על-פי הנכונות של האלגוריתם של פרים, מותר היה לבחור בכל קשת כזאת. אם לא נבחרה אף קשת כזאת אז כל קשתות העץ המקורי מפרידות בין חלקי גרף שכל הקשתות שבינהם הן כבדות יותר.

הערה:

בגרף שבו משקלי הקשתות שונים כולם אחד מהשני יהיה עץ פורש מינימלי יחיד.

ובצורה כללית יותר: כאשר יש לגרף כמה עצים פורשים מינימליים, מלבד המשקל הכולל השווה בכולם - גם משקלי הקשתות של העצים פורשים המינימליים הינו זהה.

לדוגמא בגרף הבא – יתכנו 2 עצים פורשים מינימליים אך בכולם משקלי הקשתות הם: {1,2,4,4,5}

הוכחה:

- נניח בסתירה שקיימים 2 עצים פורשים מינימליים עם משקלי קשתות שונים לגרף
- נסמן את משקלי הקשתות בכל אחד מהם ב- A, B
- מכיוון ששניהם שונים – יש בכל אחד מהם לפחות קשת אחת במשקל שונה.
- נסמן את הקשת בעלת המשקל הנמוך ביותר מבין כל הקשתות השונות ב- e1
- נניח ש- e1 נמצאת ב- A
- נוסיף את e1 ל B – כעת, מאחר ו- B הוא עץ פורש, אז הוספה של קשת אליו סוגרת מעגל, נסמן את המעגל ב- C
- מאחר וגם A הוא עץ פורש אז במעגל C יש קשת שאינה קיימת ב- A, נסמן אותה ב- e2
- כעת, אם נוריד את e2 מ- B נקבל עץ פורש (כי בחרנו קשת אחרת במעגל) אך משקל העץ הפורש בוודאות קטן יותר ממשקל B המקורי, כי e1<e2, ואם כן סתירה לכך ש- B הוא עץ פורש מינימלי!