גירסה 1.00 – 11.11.2002

אלגוריתמים בתורת הגרפים – חלק ראשון

מסמך זה הינו הראשון בסדרת מסמכים אודות תורת הגרפים, והוא חופף בחלקו לקורס "אלגוריתמים בתורת הגרפים" בטכניון (שאינו מועבר יותר).

ברצוני להודות תודה מיוחדת **לפרופסור שמעון אבן ז"ל**, על העזרה העצומה שעזר לי בהבנת החומר ובשאלות רבות עליהן ענה לי כאשר למדתי את הנושא. שמעון אבן היה מרצה שהערכתי מאוד ותרומתו למסמך זה היתה משמעותית.

מקורות:

- המסמך מבוסס במידה רבה על הרצאותיו של פרופסור שמעון אבן משנת 2002 וכן על שאלות רבות עליהן שמעון אבן ענה לי.
- ספרו של שמעון אבן מ-1979, Graph Algorithms
- פרקים מספרו החדש של שמעון אבן, שפורסמו בשנת 2002
- חלקים מהרצאות וידאו של **פרופסור ראובן בר יהודה** בנושא אלגוריתמים בתורת הגרפים.

המסמכים כוללים נקודות רבות ממקורות אלו, ובנוסף תובנות רבות, דוגמאות ורעיונות שאספתי במהלך לימוד הקורס.

ניר אדר

מושגי יסוד

הגדרה

גרף $G(V,E)$ הוא מבנה המכיל קבוצת צמתים $V=\{v_1,v_2,...\}$ וקבוצת קשתות $E=\{e_1,e_2,...\}$. לכל קשת יש שתי נקודות קצה, שאינן בהכרח שונות, וכן שם. אלא אם נציין אחרת, V, E הן קבוצות סופיות. במקרה זה נאמר גם כי הגרף G הוא סופי.

הגדרה

שתי קשתות שיש להן את אותם הקצוות יקראו **קשתות מקבילות**.

הגדרה

קשת שיש לה שני קצוות, שהם למעשה אותו קצה, תיקרא **חוג עצמי**.

הגדרה

הדרגה של צומת v, שתסומן $d(v)$, תוגדר בתור מספר הקשתות הפוגעות בצומת v.

הגדרה

אם דרגת צומת היא 0, נאמר כי **הצומת מבודדת**.

סימון

יהיו u, v קשתות ו-e צומת. נסמן: $u \xrightarrow{e} v$ אם יש צומת שקצוותיה הן u, v.
במקרה זה נאמר כי e מחבר בין u ל-v.

הגדרה

מסלול P הוא סידרה של צמתים $\{v_0,v_1,...\}$ וקשתות $\{e_1,e_2,...\}$, כך ש-P מתחילה בצומת כלשהו, נניח v_0, שאחריה מופיעה קשת, שתסומן e_1, שמחברת אותה אל v_1, וכך הלאה, המקיימת מספר תכונות. נסמן: $P: v_0 \xrightarrow{e_1} v_1 \xrightarrow{e_2} v_2 ...$.

תכונות המסלול:

1. ל-e_i ול-e_{i+1} יש צומת משותפת.
2. אם הקשת e_i איננה חוג עצמי, ואיננה בקצה המסלול, אז אחד מקצוותיה נמצא גם בקשת שלפניה והקצה השני שלה נמצא גם בקשת שאחריה.

הגדרה

יהי מסלול סופי P. נגדיר את **אורך המסלול** להיות מספר הקשתות במסלול.

הגדרה

מסלול ריק הוא מסלול בעל צומת אחד וללא קשתות.

הגדרה

מעגל הוא מסלול שמתחיל ומסתיים באותו הצומת.

הערות

- ניתן להבין מההגדרה כי אין מעגלים אינסופיים.
- מסלול ריק איננו מעגל.

הגדרה

מעגל פשוט יוגדר בצורה הבאה:

1. הליכת "הלוך וחזור" על קשת כלשהי איננה מעגל פשוט.
2. כל מעגל באורך 3 או יותר שאין בו חזרה על אף צומת פרט לקצוות הינו מעגל פשוט.
3. חוג עצמי הינו מעגל פשוט.

הגדרה

מסלול פשוט הוא מסלול שאף צומת אינה מופיעה בו פעמיים.

הגדרה

גרף פשוט הוא גרף בלי חוגים עצמיים ובלי קשתות מקבילות.

הגדרה

גרף קשיר הוא גרף בו קיים מסלול בין כל שני צמתים.

טענה

בגרף פשוט מתקיים: $|E| \le \binom{|V|}{2}$.

הגדרה + טענה

יהי G גרף פשוט. אם בין כל זוג צמתים קיימת קשת, נקרא ל-G **גרף מלא**.
במקרה זה, מתקיים כי $|E| = \binom{|V|}{2}$.

הגדרה

גרף מכוון $G(V,E)$ הוא מבנה המכיל קבוצת צמתים $V = \{v_1, v_2, ...\}$ וקבוצת קשתות $E = \{e_1, e_2, ...\}$. בגרף מכוון קשת הינה זוג סדור.

הגדרה

נגדיר **דרגות בגרף מכוון**:
d_{in} הינה מספר הקשתות הנכנסות אל הצומת.
d_{out} הינה מספר הקשתות היוצאות מן הצומת.

סימון

יהיו u, v קשתות ו-e צומת. נסמן: $u \xrightarrow{e} v$ אם יש צומת שקצוותיה הן u, v.
במקרה זה נאמר כי e מחבר בין u ל-v. u נקראת צומת ההתחלה ואילו v נקראת צומת הסיום.

טענה

בגרף מכוון וסופי, מתקיים כי $\sum_{v \in V} d_{in}(v) = \sum_{v \in V} d_{dout}(v) = |E|$.

טענה (חשובה)

בגרף לא מכוון וסופי, מתקיים כי $\sum_{v \in V} d(v) = 2 \cdot |E|$.

למה 1

בגרף לא מכוון וסופי יש מספר זוגי של צמתים שדרגתם אי זוגית.
הוכחה:
נשתמש בטענה, הטוענת כי מספר הקשתות בגרף לא מכוון הינו זוגי.
הזוגיות של מספר הקשתות נקבעת אך ורק על ידי הצמתים מהם יוצא מספר אי זוגי של קשתות. מכיוון שכך, ומכיוון שמספר הקשתות הינו זוגי, הלמה הוכחה.

ייצוג גרפים במחשב

על מנת שנוכל לנתח את סיבוכיות הזמן והמקום של אלגוריתמים בתורת הגרפים, נראה כיצד אנו מייצגים גרפים בזיכרון מחשב.

מטריצת שכנויות

כאשר נרצה לייצג גרפים פשוטים (מכוונים או לא מכוונים) במחשב, נוכל לייצגם בעזרת מטריצת שכנויות. במקום ה- (i, j) במטריצה יופיע 1 אם יש קשת בין הצמתים ה-i וה-j, אחרת יהיה במקום זה 0.
כאשר נייצג גרף לא מכוון, המטריצה תהיה מטריצה סימטרית.

הגדרה

גרף יקרא **דליל** אם $|E| << |V|^2$. גרף יקרא דליל אם $|E| \le |V|\log|V|$.

מטריצת שכנויות איננה ייצוג יעיל לגרפים, כאשר אנו עובדים עם גרפים דלילים.

ייצוג גרפים על ידי רשימות פגיעה

כאשר אנו משתמשים ברשימות פגיעה, אנו איננו מגבילים את עצמנו לגרפים פשוטים. כמו כן, סיבוכיות המקום הינה ליניארית בהתאם למספר הקשתות והצמתים, ואינה ריבועית כמו בייצוג בעזרת מטריצת שכנויות.
תיאור מבנה הנתונים:
אנו משתמשים במערך צמתים. לכל צומת v ישנו תא במערך. בכל תא ישנה רשימה של קשתות, שהן הקשתות הפוגעות בה. בסוף הרשימה ישנו NIL.
עבור כל אחת מהרשימות, קיים מצביע N(v) המצביע אל האיבר הנוכחי ברשימה.
בתחילה N(v) מצביע אל האיבר הראשון ברשימה או אל NIL אם הרשימה ריקה.

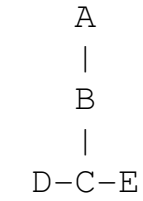
כמו כן, נשתמש במערך קשתות, שיכיל עבור כל צומת את שני הקצוות שלה (או את קצה ההתחלה וקצה הסיום, במקרה של גרף מכוון), וכן דגל שאומר האם הקשת בשימוש בגרף או לא. בתחילה כל הקשתות אינן משומשות.

הגדרה

מסלול P יקרא **מסלול מקסימלי**, אם לא ניתן להרחיב אותו,כלומר אין קשתות יוצאות מצומת הסיום, בהן לא ביקרנו.
מסלול P יקרא **מסלול מקסימום**, אם לא ניתן למצוא מסלול ארוך ממנו בגרף הנתון.

<u>דוגמא</u>

נניח כי בידינו גרף הנראה כך:



במקרה זה, D–C–E הינו המסלול המקסימלי, מכיוון שאיננו מסוגלים להפוך אותו למסלול גדול יותר (לצמתים D ו-E, הקצוות, אין שכנים מלבד C).
ישנם שני מסלולי מקסימום: A–B–C–E ו-A–B–C–D. הם מקסימום מכיוון שאי מסלולים אחרים בגרף שארוכים מהם.

<u>TRACE</u>

נרצה כעת אלגוריתם TRACE(s, P), כך שבהינתן גרף סופי $G(V,E)$ וצומת התחלה $s \in V$, תאתר מסלול מקסימלי המתחיל ב-s ואינו עובר על צומת יותר מפעם אחת.
התוצאה תוחזר לתוך הרשימה או המערך P, שבתחילה הינו ריק.
סיבוכיות הזמן המבוקשת של TRACE הינה $O(|E|)$.

מימוש אפשרי:

```
Procedure TRACE(s, P)

v ← s
while N(v) points to an edge (and not to NIL) do
        if N(v) points to a used edge do
                change N(v) to point to the next item in the list
        else do
                e ← N(v)
                change the flag of e to used
                add e to the end of P
                use the edge table to find the second endpoint of e, say it is u
                v ← u
```

גרף אויילר

משפט אויילר

גרף בלתי מכוון, סופי וקשיר הוא גרף אויילרי אם ורק אם מתקיים אחד מהתנאים הבאים:

1. יש בדיוק שני צמתים שדרגתם אי זוגית.
2. כל הצמתים בעלי דרגה זוגית.

במקרה 1, כל מסלול אויילר הוא מסלול פתוח.
במקרה 2, כל מסלול אויילר הוא מעגל.

הגדרה

מסלול יקרא **מסלול אויילרי** אם זהו מסלול בו אנו עוברים על כל קשת פעם אחת.

הגדרה

מסלול יקרא **מסלול המילטוני** אם זהו מסלול בו אנו עוברים בכל צומת פעם אחת.

אלגוריתם למציאת מסלול אויילרי בין צומת a לצומת b

יהי G גרף סופי, בו יש בדיוק שני צמתים שדרגתם אי זוגית, שנסמנם a, b. נרצה למצוא מסלול אויילרי בין צמתים אלו.

ראשית, נמצא מסלול מקסימלי בעזרת TRACE בין a ל-b. מכיוון שהגרף הוא סופי, האלגוריתם ייעצר בסופו של דבר, ויחזיר לנו מסלול.
לאחר שמצאנו מסלול זה, והשתמשנו באחת מהקשתות שמחברת את a, דרגתו של a זוגית כעת. מכיוון שכך, a איננו יכול להיות סיום המסלול יותר. לפי אותו הגיון, מכיוון שכל דרגות הצמתים זוגיות מלבד b, המסלול אינו יכול להסתיים באף אחד מהצמתים בדרך, ולכן בהכרח, P מסתיים בצומת b.
אם P כולל את כל הצמתים ב-G, סיימנו, אחרת, מתקיימים ההיסקים הבאים:

1. הדרגה של כל צומת היא זוגית.
2. ישנן כמה קשתות הצמודות לצמתים, שלא נעשה בהן שימוש, הצמודות אל P.

כעת, נבצע את התהליך הבא: נמצא צומת כזו, ונבנה ממנה מסלול מקסימלי P', הכולל את הצמתים והקשתות השונים בהם איננו משתמשים ב-P, שמתחיל בצומת v.
מכיוון שדרגות כל הצמתים זוגיות, אזי זהו מעגל, ו-v הינה גם צומת הסיום.
נאחד את המעגל עם P וניצור מסלול גדול יותר.
ככה נמשיך עד שלא ישארו יותר קשתות אותן לא ביקרנו.

סדרות דה ברואין

<u>דוגמא</u>

נרצה לרשום מעגל של אפסים ואחדות, כל שכל המילים הבינריות באורך n יופיעו בו כרצף.

עבור n=3:

ניתן לפרוש את המעגל לאיברים השונים:

000
001
011
111
110
101
010
100

עבור n=2:

עבור n=1:

0

1

<u>הגדרה</u>

יהי $\Sigma = \{0,1,...\sigma - 1\}$ אלף בית של σ אותיות. מספר המילים באורך n מעל Σ יסומן L, וערכו: $L = \sigma^n$. **סדרת דה ברואין** היא רצף (מעגלי) $a_0 a_1 ... a_{L-1}$ מעל Σ, כך שעבור כל מילה w בעלת n אותיות מעל Σ, קיים $0 \le j < \sigma$ כך שמתקיים: $a_j a_{j+1} ... a_{j+n-1} = w$ (חישובי האינדקסים נעשים מודולו L).

הגדרה

יהיו $n,\sigma \in \mathbb{N}^+$.

גרף מכוון דה ברואין $G_{\sigma,n}(V,E)$ מוגדר כך:

1. $V = \Sigma^{n-1}$, כלומר קבוצת כל σ^{n-1} המילים באורך $n-1$ מעל Σ.
2. $E = \Sigma^n$.
3. הקשת המכוונת $b_1b_2...b_n$ מתחילה בצומת $b_1b_2...b_{n-1}$ ומסתיימת בצומת $b_2b_3...b_n$.

בקשתות נמצאות המילים השונות של השפה, בעוד שבצמתים נמצאת "התוצאה" אחרי shift של התו האחרון של המילה, שבעזרתה נוכל להרכיב את המילים הבאות בשפה.
איך נמצא בגרף כזה סדרת דה ברואין? נמצא מסלול אויילרי על כל הקשתות.

טענה

לכל $n,\sigma \in \mathbb{N}^+$, מתקיים:

א. $G_{\sigma,n}(V,E)$ קשיר היטב.

ב. $d_{in}(v) = d_{out}(v) = \sigma, \forall v \in V$.

ג. $G_{\sigma,n}(V,E)$ הינו גרף אוליירי מעגלי.

הוכחת ג': גרף התשתית קשיר (לפי א'), ודרגות הכניסה והיציאה שוות (לפי ב'), ולכן הגרף אויילרי מעגלי.

מסקנה

לכל $n,\sigma \in \mathbb{N}^+$, יש סדרת דה ברואין באורך σ^n.

אלגוריתם למציאת המסלול הקצר ביותר בגרפים

<u>הקדמה</u>

ישנן מספר סוגי בעיות כאשר אנו נדרשים למצוא את המסלול הקצר ביותר בין גרפים. נברר את הפרטים הבאים, לפני שנחליט על האלגוריתם בו נשתמש:

- האם הגרף הוא גרף מכוון או גרף לא מכוון?
- אורכי הקשתות – שוות באורכן או שונות? האם מותרות קשתות "באורך" שלילי?
- מי המקור והיעד? האם נרצה את המסלול הקצר ביותר מצומת $s \to t$ מצומת s אל כל שאר הצמתים? מכל צומת אל כל צומת?
- האם נרצה למצוא מסלול אחד? את כל המסלולים האפשריים?

BFS

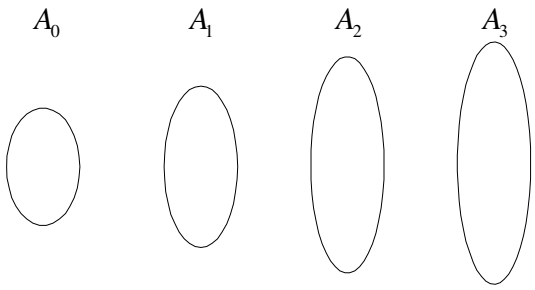
אלגוריתם המוצא מסלול קצר ביותר בגרף לא מכוון G(V, E) מצומת $s \to t$, כאשר אורכי כל הקשתות שווים ל-1.

האלגוריתם:

סמן את s ב-0.
$i \leftarrow 0$
חזור:
כל צומת לא מסומנת השכנה לצומת שמסומנת ב-i, תסומן ב-$i+1$.
הגדל את i ב-1.
כל עוד t לא סומן (או כל עוד יש צומת לא מסומנת, אם נתעניין במרחקים בין s לכל שאר הצמתים).

<u>סימונים</u>

נסמן ב-$\lambda(v)$ את הסימון של צומת v.
נסמן ב-$\delta(v)$ את המרחק המינימום בין s אל v.
נסמן ב-A_i את הקבוצה: $\{v \mid \lambda(v) = i\}$.



<u>תכונות</u>

- אין קשתות המחברות קבוצות A_i מרוחקות.
- כל מסלול מ-s אל A_i, עובר דרך A_{i-1}.
- אם לצומת v סימון $\lambda(v)$, אזי קיים מסלול באורך $\lambda(v)$ אליו.

<u>משפט</u>

$\forall v \in V, \lambda(v) = \delta(v)$

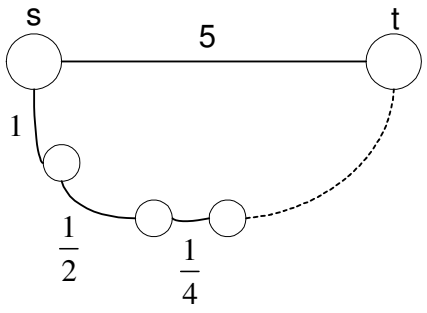
<u>הערות</u>

- האלגוריתם תקף גם עבור גרפים מכוונים.
- ניתן לממש את האלגוריתם גם באמצעות תור.
- אם נחפש את המרחקים בין s אל כל הצמתים, נדרוש שהגרף יהיה סופי. אם נחפש רק את המרחק בין s אל t, מספיק לדרוש רק כי דרגות היציאה מכל הצמתים הינן סופיות. טענה זו תורחב מייד.
- סיבוכיות האלגוריתם: אם מחפשים מסלול בין s אל t, או בין s אל כל שאר הצמתים, סיבוכיות האלגוריתם הינה $O(|E|)$. אחרת, אם מחפשים את המרחקים הכי קצרים בין כל צומת אל כל צומת אחרת, סיבוכיות האלגוריתם הינה $O(|E| \cdot |V|)$.

<u>בעיה</u>

אחת ההערות שהרגע הצגנו, היא כי כדי למצוא מסלול מ-s אל t, מספיק לדרוש כי דרגות היציאה יהיו סופיות.
דרישה זו איננה נכונה במלואה.

נביט בדוגמא הבאה:



מסלול אחד באורך סופי והוא 5, ומסלול שני באורך אינסופי, אך מתכנס לפחות מ-5.
אם זאת, האלגוריתם לא יהיה מסוגל לאתר את המסלול השני.

לפיכך, נתקן את הערה ונדרוש גם כי עבור כל קשת e, יתקיים כי $l(e) > \varepsilon > 0$, כאשר ε קבוע.

Diskjstra's Algorithm

האלגוריתם שנראה כעת מטפל אף הוא בגרפים מכוונים.
יהי גרף G(V, E), ותהי צומת התחלה s. נדרוש כי לכל קשת e השייכת לגרף, אורך $l(e) \geq 0$. המשימה שלנו היא למצוא את $\delta(v)$ עבור כל צומת בגרף.

אם ידוע שכל אורכי הקשתות הוא שלמים חיוביים, ניתן להחליף כל קשת e על ידי מסלול שעובר דרך $l(e)$ צמתים באורך 1, ולהשתמש באלגוריתם הישן. אם זאת, פתרון מעין זה הופך את האלגוריתם לבלתי סביר מבחינת סיבוכיות.
האלגוריתם שנראה כעת הוא מעין שכלול של BFS, והוא מאפשר לעשות את החיפוש בזמן ליניארי.

```
Procedure DIJKSTRA(G, s,λ)

λ(s) ← 0
T ← {s}
P ← φ
while T ≠ φ do
        choose a vertax v ∈ T for which λ(v) is minimum
        T ← T \ {v}
        P ← P ∪ {v}
        for every v —e→ u do
                if u ∈ T then do
                        λ(u) ← min{λ(u), λ(v) + l(e)}
                else, if u ∉ P then do
                        λ(u) ← λ(v)+l(e)
                        T ← T ∪ {u}
```

T זוהי קבוצה זמנית של צמתים מסומנים, כלומר, צמתים עבור λ נקבע, אולם הערך עוד יכול להשתנות.
P היא קבוצה של צמתים מסומנים באופן קבוע.
צמתים יכולים להיות ב-P, ב-T או עדיין לא מסומנים.

<u>טענה</u>

כאשר מפעילים אלגוריתם זה על כל גרף סופי, הוא נעצר בסופו של דבר.

הוכחה: כל צומת נכנס ל-T לכל היותר פעם אחת, וכל צומת שנבחרת בשורה 5 באלגוריתם עוזבת את T, ולכן הביצוע של שורה 5 לוקח לכל היותר $O(|V|)$.
אחרי ביצוע שורה 5 $|V|$ פעמים לכל היותר, T חייבת להיות ריקה, והאלגוריתם עוצר.

<u>טענה</u>

במהלך ביצוע האלגוריתם, כל צומת שניתן להגיע אליה מ-s מסומנת.

<u>טענה</u>

עבור כל צומת v שמסומנת במהלך האלגוריתם ב- $\lambda(v)$, קיים מסלול באורך $\lambda(v)$ מהצומת s אל הצומת v.

<u>מסקנה</u>

בכל זמן במהלך ריצת האלגוריתם, $\lambda(v) \geq \delta(v)$.

<u>משפט</u>

כאשר האלגוריתם עוצר, עבור כל צומת v שניתן להגיע אליה מהצומת s, מתקיים כי $\lambda(v) = \delta(v)$.

<u>הערות</u>

- האלגוריתם עובד על גרפים מכוונים וגם על גרפים לא מכוונים.
- אם נחפש את המרחקים בין s אל כל הצמתים, נדרוש שהגרף יהיה סופי. אם נחפש רק את המרחק בין s אל t, מספיק לדרוש רק כי דרגות היציאה מכל הצמתים הינן סופיות.
- האלגוריתם עובד עבור אורכי מסלול כלשהם (L) ולא רק עבור $L > 0$.
- הגרף יכול להיות פשוט או לא פשוט.
- סיבוכיות האלגוריתם: $O\left(|V|^2 + |E|\right)$

The Ford Algorithm

אלגוריתם זה מטפל בגרפים מכוונים.
יהי גרף סופי G(V, E), ותהי צומת התחלה s. לכל קשת e השייכת לגרף, אורך $l(e)$ כלשהו. המשימה שלנו היא למצוא את $\delta(v)$ עבור כל צומת בגרף.

הגדרה

מעגל ייקרא **מעגל שלילי** אם סכום ארכי הקשתות עליו הוא שלילי.

הערה

נשים לב שאם קיים מסלול בין צומת s בגרף אל מעגל שלילי C, הרי שהמרחק מ-s אל המעגל אינו מוגדר. עבור כל אורך מסלול אפשרי אל C, נוכל לעשות סיבוב נוסף על המעגל, ולקבל מסלול באורך קצר יותר.

האלגוריתם

Procedure gen-FORD(G, s, l, λ)

for every $v \in V$ do
$\lambda(v) \leftarrow \infty$
$\lambda(s) \leftarrow 0$
while there is an edge $u \xrightarrow{e} v$ such that $\lambda(u)$ is finite and $\lambda(v) > \lambda(u) + l(e)$ do
$\lambda(v) \leftarrow \lambda(u) + l(e)$

הערה

באלגוריתם זה אף צומת איננה מקבלת קביעות עד לסיום האלגוריתם.

<u>טענה</u>

במהלך ריצת האלגוריתם של Ford, אם קיימת צומת v, עבורה מתקיים כי $\lambda(v)$ הינו סופי, אזי בהכרח קיים מסלול ישיר בין s אל v, שאורכו $\lambda(v)$.

הוכחה:
תהא $x \xrightarrow{e} y$ הקשת האחרונה ששופרה, כך שאחריה $\lambda(y)$ שונתה - $\lambda(y) \leftarrow \lambda(x) + l(e)$. על פי ההנחה, יש מסלול באורך $\lambda(x)$ אל x. אם נחבר אליו את e, נגיע אל y, ולכן יש מסלול באורך $\lambda(v)$ אל y.

<u>טענה</u>

אם לגרף אין מעגלים שליליים נגישים מ-s, ואם במהלך ריצת האלגוריתם של Ford, קיימת צומת v, עבורה מתקיים כי $\lambda(v)$ הינו סופי, אזי קיים מסלול מכוון פשוט, בין s אל v, שאורכו $\lambda(v)$.

<u>טענה</u>

אם לגרף אין מעגלים שליליים נגישים מ-s, אזי בסוף ריצת האלגוריתם של Ford, מתקיים עבור כל צומת כי $\lambda(v) = \delta(v)$.

<u>הערה</u>

סיבוכיות אלגוריתם זה ידועה. כיוון שאנו יודעים את הסיבוכיות של אלגוריתם זה, נוכל לכתוב אלגוריתם לאיתור מעגלים שליליים. ניתן לאלגוריתם זמן לרוץ בהתאם לסיבוכיות שלו. אם האלגוריתם עוצר בסוף זמן זה, הרי שאין מעגלים שליליים בגרף, אחרת ישנם מעגלים שליליים בגרף.

<u>מימוש וסיבוכיות</u>

מימוש אפשרי לאלגוריתם:

```
E = {e_1, e_2, ..., e_|E|}
for |v| times
        for i = 1 to |E|
                update any node that need update
```

סיבוכיות האלגוריתם הינה $O\left(|V|\cdot|E|\right)$.
נשים לב כי זהו אלגוריתם סופי, אפילו אם בגרף מצויים מעגלים שליליים.
אם נחפש את המסלולים הכי קצרים מכל צומת אל כל צומת, הרי שהסיבוכיות תהיה $O\left(|V|^2\cdot|E|\right)$.

Floyd Algorithm

אלגוריתם זה מטפל בגרפים מכוונים.
יהי גרף סופי G(V, E), ותהי צומת התחלה s. לכל קשת e השייכת לגרף, אורך $l(e)$ כלשהו. המשימה שלנו היא למצוא את $\delta(u,v)$ עבור כל צומת בגרף, כלומר למצוא את המרחק בין הצומת u אל הצומת v, עבור כל זוג סדור של צמתים בגרף.
כמו כן, אנו מניחים כי אין מעגלים שליליים בגרף, שאין בו חוגים עצמיים, וכן שאין בו קשתות מקבילות. אם בין שתי צמתים יש יותר מקשת אחת, הרי שאנו מסוגלים להתעלם מכל הצמתים, מלבד מהקצרה ביותר, ולהפעיל את האלגוריתם עליה.
האלגוריתם של Floyd מבצע את המשימה של מציאת המרחקים בין כל צומת אל כל צומת אחרת, בסיבוכיות של $O\left(|V|^3\right)$.

מבנה הנתונים והנחות:
נניח כי אוסף הצמתים V נתון והוא: $V=\{1,2,...,n\}$.
נגדיר מטריצה בגודל $n\times n$ בשם δ^k, ונניח כי עבור כל $0\le k\le n$ מתקיים כי $\delta^k(i,j)$ היא למעשה המסלול הקצר ביותר בין צומת i אל צומת j, אשר לא עובר בין הצמתים $k+1,k+2,...,n$.

דוגמא

$$\lambda^0(i,j)=\begin{cases} l(e) & i\xrightarrow{e} j \ exists \\ \infty & else \end{cases}$$

$$\lambda^1(i,j)=\min\left(\left[\lambda^0(i,j)\right],\left[\lambda^0(i,1)+\lambda^0(1,\lambda)\right]\right)$$

$$\lambda^2(i,j)=\min\left(\left[\lambda^1(i,j)\right],\left[\lambda^1(i,2)+\lambda^1(2,\lambda)\right]\right)$$

הכללת הדוגמא

$$\lambda^0(i,j)=\begin{cases} l(e) & i \xrightarrow{e} j \ exists \\ \infty & else \end{cases}$$

$$\lambda^k(i,j)=\min\left(\left[\lambda^{k-1}(i,j)\right],\left[\lambda^{k-1}(i,k)+\lambda^{k-1}(k,\lambda)\right]\right)$$

האלגוריתם

Procedure FLOYD(G(V,E),l,δ^n)

for every $1 \le i \le n$ do
  if there is a self loop $i \xrightarrow{e} i$ and $l(e) < 0$ then do
    $\delta^0(i,i) \leftarrow l(e)$
  else $\delta^0(i,i) \leftarrow 0$
for every $1 \le i, j \le n$ such that $i \ne j$ do
  if there is an edge $i \xrightarrow{e} j$ then do
    $\delta^0(i,j) \leftarrow l(e)$
  else $\delta^0(i,j) \leftarrow \infty$
for every k, starting with $k=1$ and ending with $k=n$ do
  for every $1 \le i, j \le n$ do
    $\delta^k(i,j) \leftarrow \min\left\{\delta^{k-1}(i,j), \delta^{k-1}(i,k)+\delta^{k-1}(k,j)\right\}$

סיבוכיות זיכרון של האלגוריתם: $O\left(|V|^2\right)$ - גודל המטריצה.

סיבוכיות הזמן של האלגוריתם: $O\left(|V|^3\right)$.

נסכם את האלגוריתמים השונים:

שם האלגוריתם	BFS	Diskjstra	Ford	Floyd
אורך כל קשת	$l = 1$	$l >= 0$	any l	any l
מה אנחנו מוצאים	$s \to t$ *or* $s \to all$	$s \to t$ *or* $s \to all$	$s \to all$ *or* $all \to all$	$all \to all$
סיבוכיות זמן	$O(\|E\|)$	$O(\|V\|^2 + \|E\|)$ *or* $O(\|V\|^3)$	$O(\|V\| \cdot \|E\|)$ *or* $O(\|V\|^2 \cdot \|E\|)$	$O(\|V\|^3)$

סגור טרנזיטיבי של גרפים

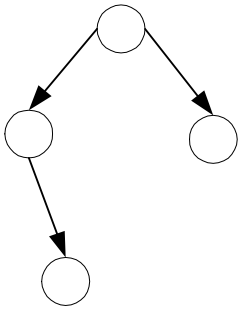
<u>הגדרה</u>

סגור טרנזיטיבי של גרף מכוון G=(V,E) הוא גרף מכוון $T = (V, E_T)$ כך ש:
$e = u \to v \in E_T$ אם ורק אם יש מסלול מכוון לא ריק מ-u ל-v ב-G.

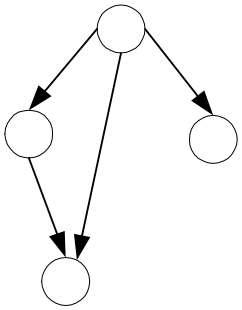
כלומר, סגור טרנזיטיבי הוא גרף בו קיימת קשת המחברת בין כל שתי צמתים בהם קיים מסלול בגרף המקורי.

דוגמא

נניח כי נתון הגרף הבא:



הסגור הטרנזיטיבי של הגרף הנ"ל הינו:



האלגוריתם של Warshall לחישוב סגור טרנזיטיבי

נניח כי הצמתים ממוספרים מ–1 ועד n, כלומר: $V = \{1,2,...,n\}$.
להלן האלגוריתם של Warshall לחישוב סגור טרנזיטיבי של גרף מכוון:

(1) for every $1 \le i, j \le n$, do:
(2) if $e = i \to j \in E$, then $T(i, j) \leftarrow 1$
(3) else, $T(i, j) \leftarrow 0$
(4) for every k, starting with k=1 and ending with k=n, do:
(5) for every $1 \le i, j \le n$, do:
(6) $T(i, j) \leftarrow \max\{T(i, j), T(i, k) \cdot T(k, j)\}$

<u>טענה</u>

בסיום האיטרציה ה-k של הלולאה שבשורות (6)-(4) מתקיים כי $T(i,j)=1$ אם ורק אם יש מסלול מכוון ב-G מ-i ל-j שצמתי הביניים בו הם מהקבוצה $\{1,2,...,k\}$.

<u>משפט</u>

האלגוריתם של Warshall מחשב את הסגור הטרנזיטיבי של G, כלומר בסיום האלגוריתם מתקיים לכל תא במטריצה T:

$$T(i,j)=\begin{cases}1 & there\ is\ a\ nonempty\ directed\ path\ in\ G\ from\ i\ to\ j\\ 0 & otherwise\end{cases}$$

<u>קשר לאלגוריתם של Floyd:</u>

לפי נכונות האלגוריתם של Floyd מתקיים כי לכל $i\neq j$, $\delta(i,j)=\infty$ אם ורק אם לא קיים מסלול מכוון ב-G מ-i ל-j. לפי המשפט לגבי האלגוריתם של Warshall, אנו נקבל כי לכל $i\neq j$: $\delta(i,j)$ סופי אם ורק אם $T(i,j)=1$.

סיבוכיות:

שורות (3)-(1) לוקחות $O(|V|^2)$ זמן.

הלולאה שבשורות (6)-(4) מתבצעת |V| פעמים, כאשר כל איטרציה לוקחת $O(|V|^2)$ זמן.

לכן הסיבוכיות הכוללת של האלגוריתם היא $O(|V|^3)$.

EOF