הרצאה 3 אלגוריתמים: $shorts\ path$

14 בנובמבר 2025

כיצד מודדים מהו המסלול הקצר ביותר?
אם הגרף אינו ממושקל: עלות המסלול היא מס' הקשתות במסלול= אורך המסלול.
אם הגרף כן ממושקל: עלות של מסלול = סכום משקלי הקשתות שעל המסלול.

הגדרה: עבור $u, v \in V$ נסמן את העלות המינימלית של מסלול מu לv ב$\delta(u, v)$.
יהיה $P = (u, v_1, ..., v_{k-2}, v)$ המסלול הקצר ביותר בין u לv (אם קיים). אזי,
אם G ממושקל:

$$\delta(u, v) = \sum_{v \in P} w(v)$$

אם G אינו ממושקל:

$$\delta(u, v) = |P| = k - 1$$

אם לא קיים מסלול בין u לv נגדיר:

$$\delta(u, v) = \infty$$

הגדרה: מסלול מu לv שעלותו היא $\delta(u, v)$ יקרא מסלול קצר ביותר.
הערה: יתכן שבין זוג קודקודים יש יותר ממסלול אחד קצר ביותר.

הערה: רוב האלגוריתמים שנראה בהרצאה יהיו עבור גרף מכוון. כיצד זה פותר את הבעיה עבור גרף שאינו מכוון? אם האלגוריתם יודע לפתור את הבעיה עבור גרף מכוון, נוכל להמיר כל גרף לא מכוון לפולטי גרף מכוון: כך שכל קשת $a \longleftrightarrow b$ תתורגם לשתי קשתות $a \to b, b \to a$.
הערה: ישנם מקרים בהם יש אלגוריתם יותר מהיר עבור גרף לא מכוון.

0.1 בעיית מציאת המסלול הקצר ביותר

לבעיה זו יש מס' גרסאות. נשים לב כי הן מדורגות כעת מהקלה לקשה.

1. **זוג קודקודים -**

קלט: $G=(V,E)$ וזוג קודקודים $v,u \in V$.
פלט: לחשב את $\delta(u,v)$ ואולי אף למצוא מסלול מu לv שהוא קצר ביותר.
2. **מקור יחיד** $(Singal\ Source\ Shorts\ Paths\ (SSSP))$ -
קלט: $G=(V,E)$ וקודקוד $s \in V$ שיקרא קודקוד מקור.
הפלט: לחשב עבור כל $v \in V$ את $\delta(s,v)$ ואולי גם למצוא מסלול קצר ביותר מs לכל $v \in V$.
3. **כל הזוגות** $(All\ Pairs\ Shorts\ Paths\ (APSP))$ -
קלט: $G=(V,E)$
פלט: לכל $u,v \in V$ להחזיר את $\delta(u,v)$ ואולי אף למצוא את המסלול הקצר ביותר לכל $u,v \in V$.

הערה: בעיה 1 מוכלת בתוך בעיה 2, ועם זאת כפי שנראה בהמשך לא קיים אלגוריתם שפותר אותה יותר טוב מאת בעיה 2. כלומר, לא מצאו עדיין אלגוריתם יעיל יותר בסיבוכיות עבור בעיה 1. כמו כן, בעיה 2 מוכלת בבעיה 3 - אך כן זמן הריצה של בעיה 2 טוב יותר משל 3.

0.1.1 איך נראה פתרון בכל אחד מהגרסאות כשמחפשים את המסלול?

זוג יחיד: מסלול $P=(u,v_0,.....,v_{k-1},v)$, שיעלה $O(|V|)$ זכרון.
מקור יחיד: נאיבית, אפשר להחזיר $|V|$ מסלולים שונים, אחד עבור כל קודקוד מטרה. כלומר:

$$p_1=(s,....,v_1)$$

$$p_2=(s,...,v_2)$$

..

$$p_n=(s,...,v_n)$$

מה עלות הזכרון בפתרון זה? $\sum_{i=1}^{n}|P_i| \le |V|\times max\{|P_i|\} \le |V|\times|V| = O(|V|^2)$
כעת נראה שישנה אפשרות להחזיר את הפתרון בצורה שתשתמש בפחות מקום. לשם כך נשתמש בלמה החשובה מאוד הבאה -

למה 1: תת מסלול של מסלול קצר ביותר הוא גם מסלול קצר ביותר. כלומר, יהיה מסלול $P=(v,w_1,..w_r,x,...,y,z_1,..,z_t,u)$ קצר ביותר. אזי, המסלול בין x לy שמוכל במסלול P הוא גם הקצר ביותר.

הוכחה: יהי המסלול הקצר ביותר בין הקודקודים v וu: $P=(v,..,x,p_1,...,p_k,y,..,u)$. נכ"ש כי המסלול בין x לy $P_{xy}=(x,p_1,...,p_k,u)$ אינו הקצר ביותר. כלומר, קיים מסלול $P_{xy2}=(x,u_1,....,u_m,u)$ בין x לy כך ש$m<k$, כלומר $|P_{xy}|<|P_{xy2}|$. אזי, נסתכל על המסלול $P'=(v,w_1,..,w_r,P_{xy2},z_1,..,z_t,u)$ קיבלנו מסלול בין v לu המקיים

$$|P'|=|P|-|P_{xy}|+|P_{xy2}|<|P|$$

בסתירה לכך שP הינו המסלול הקצר ביותר בין v לu.

כעת, נחזור לדון בשמירת הזכרון: בעת שמירת מסלול כלשהו, למשל $s \to v_4 = (s, v_2, v_{14}, v_3, v_{90}, v_4)$ אנחנו שומרים מסלולים קצרים ביותר נוספים: בין $v_2 \to v_{90}$ למשל, לפי הלמה שהוכחה לעיל גם הוא מסלול קצר ביותר.

כמו כן, נשים לב כי נוכל לקבל למשל שני מסלולים: $(s, v_{10}, v_5, v_6, v_2), (s, v_{10}, v_5, v_8, v_4, v_9)$ ולשרשר אותם למסלול יחיד כך:

$$(s, v_{10}, v_{5 \to v_6, v_2}^{\to v_8, v_4, v_9})$$

כלומר, ליצור צורה של עץ שs שורשו. סה"כ זו תהיה הטכניקה -

ניתן לייצג מסלולים קצרים ביותר מS לכל שאר הקודקודים בגרף באמצעות עץ מסלולים קצרים ביותר.

נשים לב - העץ לא מכיל את כל המסלולים הקצרים ביותר מs בגרף, כלומר: לכל $v \in V$ יהיה קיים מסלול קצר ביותר שייוצג בעץ מ$s \to v$, אך יתכן שקיימים שני מסלולים כאלו קצרים ביותר באותו משקל והעץ יבחר אחד מהם בדיוק שיופיע בו.

מסקנה: יתכן שישנם כמה עצי מסלולים קצרים ביותר.

לסיכום - בגרסת מקור יחיד אנחנו נחזיר **עץ מסלולים קצרים ביותר** שעלותו תהיה כגודל מס' הקודקודים בו, $O(|V|)$. נשים לב - קודקוד שהינו ראש העץ יהיה בלולאה עצמית עם עצמו, קודקודים שאין אליהם מסלול יסומנו אל $null$.

כל הזוגות $(APSP)$: במצב זה נרצה להחזיר מטריצה A בגודל $|V| \times |V|$, כשנרצה להחזיר את $\delta(v, u)$ אנחנו נייצג זאת במטריצה ע"י $A_{ij} = \delta(v_i, v_j)$, סה"כ יעלה $O(|V|^2)$ זכרון.

אם נהיה מעוניינים במסלולים - נרצה עץ מסלולים קצרים ביותר מכל קודקוד, כלומר בכל תא במטריצה נאחסן עץ שכזה - מה שיעלה $O(|V|^3)$ זכרון.

0.2 אלגוריתם $SSSP - BFS$ במקרה הלא ממושקל

BFS היא סריקה לרוחב של העץ לפי רמות, מבצעים אותה באמצעות תור קדימויות כפי שראינו בקורס מבני נתונים.

האלגוריתם סורק את הצמתים בסדר שנקבע על פי מרחקם מהצומת ההתחלתי. **אלגוריתם זה מטפל ב$SSSP$ במקרה הלא ממושקל.** וכן, אורך המסלול נספר לפי מס' הקשתות.

האלגוריתם מאחסן 3 סוגי מידע לכל קודקוד:

1. $d[u]$ - אומדן לגבי $\delta(s, u)$. בסיום הריצה יתקיים $d[u] = \delta(s, u)$
2. $\pi[u]$ - כלי עזר לחישוב המסלולים הקצרים ביותר מs. בסיום הריצה הוא מצביע לאבא של u בעץ המסלולים הקצרים.
3. $Color[u]$- מעין מזהה:

א. $w(hite)$= האלגוריתם עוד לא ביקר בu.

ב. $g(ray)$= האלגוריתם ביקר בu ולא טיפל בו.

ג. $b(lack)$= האלגוריתם סיים לטפל בu.

0.2.1 האלגוריתם $BFS = (G = (V, E), s)$

```
BFS(G = (V, E), s)
1  for each u ∈ V
2      d[u] ← ∞
3      π[u] ← NULL
4      color[u] ← w
5  d[s] ← 0
6  color[s] ← g
7  Q.Enqueue(s)
8  while Q ≠ ∅
9      u ← Q.Dequeue()
10     for each v ∈ ADJ[u]
11         if color[v] = w
12             color[v] ← g
13             d[v] ← d[u] + 1
14             π[v] ← u
15             Q.Enqueue(v)
16     color[u] ← b
```

האלגוריתם מאתחל בתחילה את $d[u]$ להיות אנסוף, את π להיות $null$ ואת כל הצבעים להיות w ־ לא ביקר. לאחר מכן נאתחל את s. נשים לב כי $d[s] = 0$ כיוון שאורך המסלול הקצר ביותר מs לעצמו הוא אפס. תהליך האתחול מתרחש עד לשלב 7.

אנחנו משתמשים בתור $FIFO$, ומכניסים אליו את s. כעת כל עוד התור לא ריק אנחנו מבצעים: מוציאים מהתור את האיבר הראשון, עוברים על כל השכנים של הקודקוד שהוצאנו, אם הצבע שלהם לבן משמע לא ביקרנו אותו עוד, נסמן את הקודקודים באפור, נגדיר את הd שלהם להיות הd של הקודקוד שהוצאנו (שהיה השכן שלהם) ועוד אחד ־ כי יש קשת שנוספה למסלול,וכן נגדיר את אבא של כל הקודקודים האלו להיות u (הקודקוד שהוצאנו), לבסוף נכניס את כל השכנים הללו לתור.

בסיום, נגדיר את הצבע של הקודקוד שהוצאנו כb, סיימנו לטפל בו. וכעת, נעבור לטפל בקודקוד הבא בתור. כך ־ עד שהתור יתרוקן.

זמן הריצה: האתחול עולה $O(|V|)$ זמן, לאחר מכן מבצעים לולאה ־ נשים לב כי במהלך הלולאה אף קודקוד לא נצבע בלבן, ולכן כל קודקוד נכנס ויוצא מהתור לכל היותר פעם אחת. ולכן הפעולות $dequeue, enqueue$ מתבצעות פעם אחת לכל קודקוד. ומכאן, שכל הלולאה של ה$while$ מתבצעת לכל היותר $O(|V|)$ פעמים. באשר לעלות לולאת for על קודקוד u היא $O(deg(u))$, ולכן סה"כ זמן הריצה יהיה

$$|V| + \sum_{u \in U} deg(u) = |V| + 2|E| = O(|V| + |E|)$$

וקיבלנו זמן לינארי. כמו כן נשים לב כי אסור להניח $|V| \leq |E|$, אנחנו מדברים על גרף כללי G.

נשים לב: π מגדיר את עץ המסלולים הקצרים, כלומר ריצת BFS יכולה גם להחזיר לנו את עץ המסלולים הקצרים ביותר. באמצעות ערך π ניתן לבנות את עץ זה.

הערה: $O(|V| + |E|)$ הוא חסם תחתון לגודל הקלט ולכן אלגוריתם BFS הוא האופטימלי לפתרון הבעיה.

0.2.2 נכונות של BFS

המטרה היא להוכיח שבסוף ריצת $BFS(G,s)$ מתקיים כי $\forall u \in V : d[u]=\delta(s,u)$

למה 2: למת אי שוויון המשולש. יהי $G=(V,E)$ לא ממושקל, ויהי $s \in V$, ויהי $(u,v) \in E$, אזי

$$\delta(s,v) \le \delta(s,u)+1$$

(כלומר, בהינתן המסלול $s \to ... \to u \to v$, המסלול הקצר ביותר לעבור מ-s אל v חסום במסלול הקצר ביותר לעבור מ-s אל u ועוד מעבר על הקשת $e=(u,v)$, נשים לב שזה אפשרות למסלול ויתכן שיש מסלול טוב יותר קצר יותר. מדברים על חסם בלבד!).

הוכחה: נחלק למקרים.

א. אם אין מסלול בין s ל-u: אזי בהכרח $\delta(s,v) \le \infty+1$, כנדרש.

ב. אם יש מסלול בין s ל-u: זה גורר שמתקיים מסלול בין s ל-v: המסלול בין s ל-u בתוספת הקשת (u,v). במקרה זה, עלות של המסלול הקצר ביותר מ-s ל-v לא יכול להיות גדול יותר מעלות המסלול מ-s ל-u בתוספת העלות של הקשת (u,v) כיוון שאחד המסלולים האפשריים מ-s ל-v הוא המסלול שבינינו כנ"ל, ולכן המסלול הקצר ביותר בוודאות יהיה או זה, או באורך קטן מזה. מסקנה $\Longleftarrow$ $\delta(s,v) \le \delta(s,u)+1$.

למה 3: לאחר הרצת $BFS(G,s)$ לכל $v \in V$ מתקיים $d[v] \ge \delta(s,v)$.

הוכחה: נבצע אינדוקציה על מס' פעולות $enqueue$ שיתבצעו באלגוריתם. נסמנם n.

בסיס: $n=1$, עבור s יתקיים $d[s]=0$ ועבור שאר הקודקודים u יתקיים $d[u]=\infty$ ואכן מתקיים התנאי.

צעד: נניח שהטענה נכונה עבור $n-1$ פעולות הכנסה. נוכיח ל-n.

שינוי הערך $d[u]$ יכול להתבצע רק במהלך מעבר על השכנים של u שצבעם כעת לבן. אם כן, יספיק להוכיח שעבור כל שכן של u, v שצבעו לבן יתקיים $d[v] \ge \delta(s,v)$

יהי v קודקוד כנ"ל. לאחר העדכון יתקיים -

$$d[v]=d[u]+1 \ge \delta(s,u)+1 \ge (*)\delta(s,v)$$

כאשר $(*)$ נכון לפי למת אי שיוויון המשולש והנחת האינדוקציה.

למה 4: בכל זמן באלגוריתם אם $Q=(v_1,...,v_r)$ מתקיימות שתי תכונות:

א. $d[v_1] \le d[v_2] \le ... \le d[v_r]$

ב. $d[v_r] \le 1+d[v_1]$ (כלומר לא יתכן שבתור בו זמנית ישנם יותר משתי שכבות במקביל).

הוכחה:

נוכיח את הטענה באינדוקציה על סדרת פעולות $enqueue, dequeue$

בסיס: תור מכיל רק את s ולכן בצומת אחד אכן מתקיימות שתי הלמות.

צעד:

1. $dequeue$: כעת התור נראה כך $Q=(v_2,...,v_r)$, אכן א' מתקיים כי אי השוויון בפרט יכול להתחיל מ-$d[v_2]$.

אם כן,

$$d[v_r] \le_{(*)} d[v_1]+1 \le_{(**)} d[v_2]+1$$

כאשר $(*)$ מהנחת האינדוקציה ו-$(**)$ מא'.

ג. $enqueue$: כעת התור נראה $Q = (v_1, .., v_r, v_{r+1})$. נסמן v_0 את הצומת שיצא מהתור ובגינו נכנס v_{r+1}. מכאן $d[v_{r+1}] = d[v_0] + 1$, נפצל למקרים:
א. אם v_1 היה בתור בעת שv_0 יצא אזי בהכרח לפי א' $d[v_0] \leq d[v_1]$ ומכאן נקבל כי $d[v_{r+1}] = d[v_0] + 1 \leq d[v_1] + 1$ ומכאן שתכונה 2 מתקיימת.
- אם v_r היה בתור כשv_0 יצא אזי $d[v_r] \leq d[v_0] + 1 = d[v_{r+1}]$ ואז גם תכונה 1 מתקיימת.
- אם v_r לא היה בתור כשv_0 יצא אזי אזי v_r נכנס בגין v_0 ולכן $d[v_0] + 1 = d[v_r] = d[v_{r+1}]$ ו1 מתקיים גם כאן.
ב. אם v_1 לא היה בתור בעת שv_0 יצא, אזי כל הצמתים נכנסו בגין v_0 והערך $d[v_i]$ שלהם הוא $d[v_0] + 1$ ולכן גם 1 וגם 2 מתקיימים.
כנדרש.

מסקנה 5: אם $u \in V$ יוצא מQ לפני $v \in V$ תוך כדי ריצת $BFS(G, s)$ אזי $d[u] \leq d[v]$. (כלומר, ערכי הd של הקודקודים יכולים רק לעלות לאורך ההרצה).
הוכחה: נובע ישירות מלמה 4.

למה 6 (**הוכחת נכונות** BFS) **:** לאחר הרצת אלגוריתם BFS על גרף מכוון $G = (V, E)$ וקודקוד $s \in V$, מתקיים:

$$\forall u \in V : d[u] = \delta(s, u)$$

הוכחה: נניח בשלילה כי קיים צומת אחד לפחות עבורו אין שוויון. נסמן u הצומת עם $\delta(s, u)$ הכי קטן עבורו זה מתקיים. כלומר $d[u] \neq \delta(s, u)$. עבור v הקודם לu במק"ב מs לu בהכרח קיים שוויון $d[v] = \delta(s, v)$ (בהכרח $u \neq s$ כי $d[s] = 0$ לפי האלגוריתם ואכן $\delta(s, s) = 0$).
כאשר v יצא מהתור, הוא עובר על כל שכניו. אם u היה לבן, הוא היה מקבל את הערך הנכון של $d[u] = \delta(s, v) + 1$, ולכן u אינו לבן כלומר u קדם לv בתור. ולפי מסקנה 5: $d[u] \leq d[v] = \delta(s, v) < \delta(s, u)$, קיבלנו $d[u] < \delta(s, u)$ בסתירה ללמה 3.

0.3 אלגוריתם סריקת DFS

בהינתן גרף, סריקת DFS על הגרף היא סריקה לעומק.
הסריקה עוברת על כל הקודקודים של הגרף באופן הבא: כל עוד יש קודקוד שלא ביקרנו בו - נבקר בו. כאשר מבקרים בקודקוד, בודקים אם יש מישהו משכניו שטרם ביקרו בו - ואם כן מבקרים בו בקריאה רקורסיבית.
בהינתן גרף מכוון $G = (V, E)$ האלגוריתם DFS סורק את כל הקודקודים.
בדומה לBFS, האלגוריתם משייך לכל קודקוד צבע שמסמל את מצב הקודקוד:
b- שחור, ביקרנו וסיימנו לטפל בקודקוד.
w- לבן, טרם ביקרנו בקודקוד
g- אפור, ביקרנו אך טרם סיימנו לטפל בקודקוד.
בנוסף לכל קודקוד $u \in V$ **שומר האלגוריתם שלושה ערכים:**
א. $d(u)$ - זמן הגעה (צביעה באפור)
ב. $f(u)$- זמן עזיבה (צביעה בשחור)
ג. $\pi(u)$- קודקוד קודם. השדה π מגדיר לכל קודקוד קודקוד קודם, אשר אם נסתכל עליו כ"אבא" של הקודקוד הראשון נקבל אוסף של עצים, המכונה גם יער העומק. נשים לב שיער העומק תלוי בריצה ספציפית של האלגוריתם הDFS ולגרף נתון יכולים להיות מס' יערי עומק שונים.

להלן האלגוריתם:

DFS(G)

1. for $u \in V$
   (א) $color[u] \leftarrow w, \pi[u] \leftarrow$ NULL
2. $t \leftarrow 0$
3. for $u \in V$
   (א) if $color[u] = w$
   (ב) then DFS-Visit(u)

DFS-Visit(u)

1. $color[u] \leftarrow g$
2. $d[u] \leftarrow t \leftarrow t+1$
3. for $v \in adj(u)$
   (א) if $color[v] = w$
   (ב) then DFS-Visit(v)
      i. $\pi[v] \leftarrow u$
4. $color[u] \leftarrow b$
5. $f[u] \leftarrow t \leftarrow t+1$

האלגוריתם ישמור כמשתנה גלובלי את t: שיאותחל בהתחלה לאפס ויגדל בהתאם לאלגוריתם. t ישמש לציין את זמן הכניסה הנוכחי. האלגוריתם הוא אלגוריתם רקורסיבי שעובר לעומק על כל הגרף.

הגדרה: יער העומק מוחזר אלינו כמערך π. נשים לב כי $null$ במערך מציין שורש של עץ. יער העומק מכיל עצים (מסלולים ארוכים) שונים.

סיבוכיות זמן ריצה: נראה כי כל צומת נקראת עם $DPS - Visit(u)$ פעם אחת בדיוק. בתחילה, האתחול עולה $O(|V|)$ וסה"כ מבצעים לולאה של $O(|V|)$ כפול $1 + deg(u)$ וסה"כ נקבל $\sum_{v\in V}(1+deg(u)) = O(|V|+|E|)$ זמן הריצה של האלגוריתם. זמן הריצה לינארי.

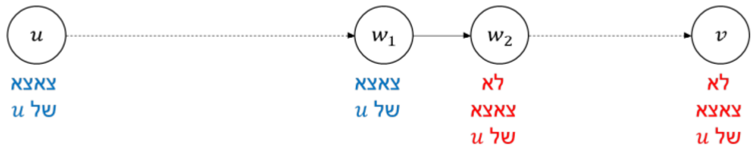
הערה חשובה: לא מתואר כיצד אנחנו רצים על האלגוריתם, ולכן כל סדר של הקודקודים (כל עוד אנחנו יודעים אותם מראש) הינו חוקי. יתכנו $|V|!$ פרמוטציות אפשריות לריצת DFS.

משפט הסוגריים: בכל DFS על גרף $G=(V,E)$ מכוון או שלא מכוון,
עבור כל זוג קודקודים $v, u \in V$ מתקיים אחד מהשלושה הבאים (עבור האינטרוול של זמן ההתחלה עד זמן הסיום):

1. $[d[u], f[u]] \cap [d[v], f[v]] = \emptyset$
2. $[d[u], f[u]] \subseteq [d[v], f[v]]$
3. $[d[u], f[u]] \supseteq [d[v], f[v]]$

הערה. אם $u \neq v$ אזי בסעיפים 2 ו3 מדובר על הכלה ממש (לא שוויון)
מסקנה: קודקוד v הוא צאצא של קודקוד u ביער העומק אמ"מ מקרה 3 מתקיים.

משפט המסלול הלבן: $v \in V$ הוא צאצא של $u \in V$ ביער העומק אמ"מ בזמן $d[u]-1$ קיים בG מסלול מu לv שכל קודקודיו צבועים בלבן.
הוכחה:
$\Longleftarrow$: אם v צאצא של u ביער העומק, הרי קיים מסלול מv לu ביער העומק ובפרט בגרף G. כל הקודקודים על המסלול הזה הם צאצאים של u, ולכן ע"פ משפט לכל הקודקודים x הללו מתקיים $d[u] < d[x]$, כולם צבועים בלבן, לכן הראינו מסלול לבן מu לv. (נשים לב ־ המסלול הוא לבן וניתן לראות זאת באלגוריתם שיש קריאה של DFS אפ"ם הקודקוד בלבן).
$\Longrightarrow$: נניח כי בזמן $d[u]-1$ קיים בG מסלול מu לv שכל קודקודיו צבועים בלבן. נניח בשלילה כי v אינו צאצא של u ביער העומק בזמן $d[u]-1$. נסתכל על המסלול הנ"ל מu לv ויהי w_1 הקודקוד הכי קרוב לu על המסלול שהוא צאצא של u וw_2 הקודקוד העוקב במסלול (שבהכרח אינו צאצא של u־ ראה באיור), כיוון שw_1 צאצא של u מתקיים $d[u] \leq d[w_1], f[w_1] \leq f[u]$ בזמן שביקרנו בw_1 בוודאות האלגוריתם בודק את צבעו של w_2.
מכיוון שw_2 לבן (כי אחרת זה אומר שביקרנו בו במהלך הסריקה מ־u ־ והוא צאצא שלו, שהרי אנחנו עדיין בתוך סריקה זו) אזי בהכרח נבקר בw_2 ומפילא $d[u] \leq d[w_1] < d[w_2] < f[w_2] <$



איור 1: המחשת ההוכחה

$f[u] \leq f[w_1]$ בסתירה להנחתנו. מכאן בהכרח v צאצא של u ביער העומק.

0.3.1 סיווג קשתות

ניתן לסווג את הקשתות בגרף בהתאם לריצת ה־DFS כך שכל קשת מסווגת לפי אחד מהסוגים הבאים, כך שאין קשת שנמצאת בשתי קבוצות :

1. קשת עץ: קשת מהצורה $(\pi(u), u)$ עבור $u \in V$ כלשהו. יש בדיוק $|v| - k$ קשתות כאלו באשר k הוא מס' העצים.
2. קשת אחורה: קשת מ־v לאב קדמון של v שנקרא לו u.
צריך להתקיים $d[u] < d[v] < f[v] < f[u]$
3. קשת קדימה: קשת מ־v לצאצא לא ישיר של v, נקרא לו u.
צריך להתקיים $d[v] < d[u] < f[u] < f[v]$
4. קשת חוצה: קשת מ־u לקודקוד שאינו צאצא ואינו אב קדמון של u. זה כל שאר הקודקודים, הDF של צמתי הקשת זרים.

0.4 גרף מכוון חסר מעגלים (DAG)

גרף שכזה נקרא DAG, כיצד נזהה האם גרף G הוא DAG?

טענה: גרף G הוא DAG $\iff$ בהרצת DFS אין קשתות אחוריות.

הוכחה: נוכיח קונטרה פוסיטיב.

נניח שיש מעגל בG. $\Longleftarrow$ נסמן u צומת ראשון שנקרא עם $DFS - visit$ במעגל. לכן יש מסלול לבן לשאר הצמתים במעגל. $\Longleftarrow$לפי משפט המסלול הלבן, צמתי המעגל הם צאצאים של u ביער העומק ויש לפחות קשת אחת מהם שחוזרת לu (מהגדרת מעגל).

נניח שיש קשת אחורית מv לu. בשילוב עם קשתות העץ מu לv (שקיימות כי v צאצא של u לפי הגדרת קשת אחורית) ונקבל כי יש מעגל בגרף המקורי. כנדרש.

מסקנה: כעת בהינתן הרצת DFS, נוכל לבדוק בקלות האם יש קשתות אחוריות (נעבור על כל הקשתות), ואם אין, משמעות הדבר שאין מעגלים. כלומר אלגוריתם לבדיקה האם יש בG מעגל בעלות $O(|E| + |V|)$.