אלגוריתמים 1

תוכן העניינים

פרק 1

אלגוריתמי BFS ו-DFS

1. BFS - Breadth First Search

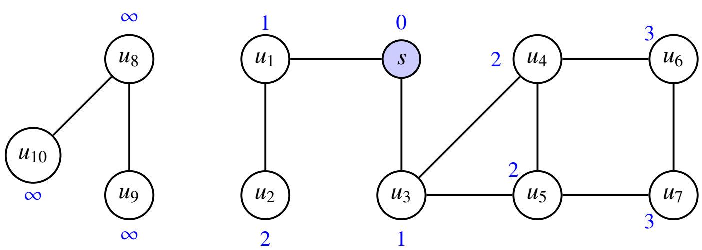
שאלה 1.1 כיצד לחשב מסלול קצר ביותר בין שני צמתים בגרף לא מכוון G?

1.1. הגדרת המרחק בגרף לא מכוון.

הגדרה 1.1 (המרחק בין צמתים u, v בגרף G)

בהינתן גרף לא מכוון $G = (V, E)$ ושתי צמתים $u, v \in V$,

המרחק בין u ו-v ב-G הוא האורך (מספר קשתות) של המסלול הקצר ביותר בין u ו-v ב-G. נסמן מרחק זה ב-$\delta_G(u, v)$ או ב-$\delta(u, v)$.



איור 1. המרחקים $\delta(s, u)$ מצומת s בגרף לא מכוון זה מסומנים בכחול ליד כל צומת $u \in V$.

טענה 1.1 (המקבילה לאי-שוויון המשולש)

יהי $G = (V, E)$ גרף לא מכוון, ויהי $s \in V$. לכל קשת $e = (u, v) \in E$ מתקיים:

$$\underbrace{\delta(s, v)}_{\text{המרחק בין } s \text{ ו-} v} \leq \underbrace{\delta(s, u)}_{\text{המרחק בין } s \text{ ו-} u} + \underbrace{1}_{\text{אורך הקשת } e}$$

הוכחת הטענה. אם אין מסלול בין u ו- s ב-G אז $\delta(s, u) = \infty$ והטענה מתקיימת.

אחרת, יהי P מסלול קצר ביותר בין s ו-u ב- G, כאשר אורכו שווה ל-$\delta(s, u)$.
נשרשר ל-P את הקשת e, וקיבלנו מסלול ב-G בין s ו-v, שאורכו שווה ל-$\delta(s,) + 1$.

$\delta(s, v)$ שווה לאורך המסלול הקצר ביותר בין s ו-v ב-G, ולכן $\delta(s, v) \leq \delta(s, u) + 1$. ■

1.2. מוטיבציה לאלגוריתם BFS.
נרצה לחשב את המרחק בין צומת s לכל צומת בגרף:

- קלט: גרף לא מכוון $G = (V, E)$ וצומת $s \in V$.
- מטרה: לחשב לכל $v \in V$ את $\delta_G(s, v)$.

האינטואיציה: להתחיל מהצומת היחיד שעבורו יודעים את $\delta(s, ?)$, וזה s עצמו.

1.3. אלגוריתם ה-BFS.

- <u>אתחול</u>: $\lambda(v) \leftarrow \begin{cases} 0 & v = s \\ \infty & v \neq s \end{cases}$, $T \leftarrow \{s\}$, $Q \leftarrow \{s\}$ תור.
- כל עוד $Q \neq \emptyset$:
  (1) יהי u הצומת בראש התור Q.
  (2) לכל קשת $e = (u, v) \in E$ כך ש-$v \notin T$:
    (א) $T \leftarrow T \cup \{v\}$.
    (ב) $\lambda(v) \leftarrow \lambda(u) + 1$.
    (ג) הכנס את v לסוף התור Q.
  (3) הוצא את u מהתור Q.

1.4. נכונות האלגוריתם.

הערה 1.1 (סימון מקובל בקורס) עבור גרף $G = (V, E)$, נסמן $|V| = n, \ |E| = m$.

שאלה 1.2

(1) מדוע האלגוריתם מחזיר תשובה נכונה?
(2) עד כמה האלגוריתם יעיל? (בד"כ יעילות תתייחס לזמן)

<u>נתחיל מ-(2).</u>

- האתחול: $O(n)$.
- האיטרציה בה u יוצא מ-Q: $O(\deg(u))$.

כמו כן,

- כל צומת נכנס ל-Q לכל היותר פעם אחת.
- כל צומת שנכנס ל-Q גם יוצא מ-Q.

סך הכל זמן ריצה:

$$\underbrace{O(n)}_{\text{אתחול}} + \underbrace{O\left(\sum_{u \in V} \deg(u)\right)}_{\text{חסם עליון על זמן הריצה של כל האיטרציות}} = O(n + m)$$

■

נתמקד בטענה (1), ונוכיח אותה תוך שימוש בטענות העזר הבאות:

למה 1.1 ("λ לא מפספס למטה") יהי $G = (V, E)$ גרף לא מכוון, ותהא צומת $s \in V$. יהיו $\forall v \in V, \ \lambda(v)$ הסימונים שהתקבלו מריצת BFS על G החל מ-s. אזי:

$$\lambda(v) \geq \delta(s, v), \ \forall v \in V$$

<u>הוכחה.</u> יהי $v \in V$.

אם v לא נכנס ל-Q, יתקיים $\lambda(v) = \infty$ והטענה נכונה.

אם v נכנס ל-Q (וזה קורה בדיוק פעם אחת), נוכיח את הטענה באינדוקציה על סדר כניסת הצמתים ל-Q:

- בסיס: s נכנס ראשון לתור (המקרה ש-$v = s$), ואז:

$$\lambda(s) = \underbrace{0}_{\text{הגדרת האלגוריתם}} = \delta(s, s)$$

- צעד: נניח נכונות עבור k הצמתים הראשונים שהוכנסו לתור, נניח כי v היא הצומת ה-$k+1$ שהוכנסה לתור.

ברגע ההכנסה של v ל-Q, נסמן ב-u את הצומת שבראש Q, ונקבל:

$$\lambda(v) \underbrace{=}_{\text{הגדרת האלגוריתם}} \lambda(u) + 1 \underbrace{\geq}_{\substack{\text{הנחת אינדוקציה} \\ \text{עבור } u}} \delta(s, u) + 1 \underbrace{\geq}_{\substack{\text{אי שוויון המשולש} \\ \text{עבור } s \text{ ו-} (u,v) \in E}} \delta(s, v)$$

■

למה 1.2 יהי $(v_1, v_2, \ldots, v_k)$ תוכן Q בשלב כלשהו של ריצת BFS על G החל מ-s. אזי:

(1) $\lambda(v_1) \leq \lambda(v_2) \leq \ldots \leq \lambda(v_k)$.

(2) $\lambda(v_k) \leq \lambda(v_1) + 1$.

הוכחה. נוכיח באינדוקציה על סדר הפעולות של הכנסה/הוצאה מ-Q:

- בסיס: האתחול הוא כש-Q מכיל רק את s. לכן (1) ו-(2) מתקיימים באופן ריק.
- צעד: נניח נכונות עבור r הפעולות הראשונות, ונוכיח עבור הפעולה ה-$r+1$.

אם הפעולה ה-$r+1$ הייתה הכנסה, נניח שהכנסנו את v ו-u בראש התור, אז:

$$\lambda(v) = \lambda(u) + 1$$

לפי הגדרת האלגוריתם.

בגלל שלפני הוספת v ל-Q (1) ו-(2) התקיימו, זה יתקיים גם לאחר הוספת v.

אם ההפעלה ה-$r+1$ הייתה הוצאה, אז ברור שמהנחת האינדוקציה (1) ו-(2) יתקיימו גם לאחריה.

■

משפט 1.1 (הוכחת נכונות אלגוריתם BFS)

יהי $G = (V, E)$ גרף לא מכוון ו-$s \in V$.

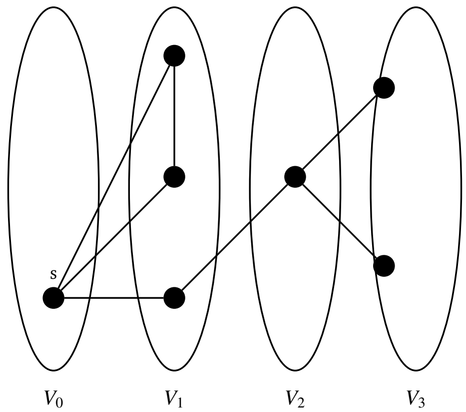
אז בסיום ריצת BFS על G החל מ-s מתקיים:

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

הוכחת המשפט משתמשת בטענות 1 ו-2.

רעיון ההוכחה: נסתכל על שכבות הגרף לפי מרחקן מ-s:

$$V_k \triangleq \{u \in V : \delta(s, u) = k\}$$



איור 2. שכבות של גרף לא מכוון לדוגמה עבור צומת s כלשהי

<u>הוכחת המשפט.</u> נניח שב-G אין מסלול בין s ו- v $\Longleftarrow$ $\delta(s,v)=\infty$.
לפי טענה 1 נקבל ש-$\lambda(v)\geq\infty$, כלומר $\lambda(v)=\infty$, והמשפט נכון.

נניח שב-G יש מסלול בין s ו-v ונסמן $\delta(s,v)=k$.
נוכיח את המשפט באינדוקציה על k:

- <u>בסיס</u>: $k=0$, אז $v=s$, והמשפט מתקיים מפני שבאתחול מוגדר $\lambda(s)=0$.
- <u>צעד</u>: נניח כי $v\in V_k$, ונסמן:

$$A\triangleq\{u\in V_{k-1}|(u,v)\in E\}$$

כאשר הגדרת A אינה תלויה באלגוריתם.
נסמן ב-u^* את הצומת ב-A שהיא <u>הראשונה</u> לצאת מהתור Q.

נשים לב ש-A אינה יכולה להיות ריקה, ולפי הנחת האינדוקציה,
בסיום ריצת האלגוריתם לכל הצמתים ב-A ישנו ערך λ השווה ל-$k-1$,
ולכן בהכרח כל אחד מהם הוכנס לתור Q.

נראה שבאיטרציה שבה u^* נמצא בראש התור Q, לצומת v מתקיים ש-$\lambda(v)=\infty$ (כלומר, v עדיין "לא התגלה").

נניח בשלילה שזה לא המצב, ולכן יש איטרציה קודמת לזו ש-u^* נמצא בה בראש Q, שבה v מוכנס לתור Q (ונניח ש-w נמצא בראש התור Q באיטרציה זו).

בגלל בחירת u^*, מתקיים ש-w הוא שכן של v בשכבה j, כך ש-$0\leq j\leq k-1$ (נובע מלמה 1.2).

לפי הנחת האינדוקציה $\lambda(w)<\lambda(u^*)$, וכעת:

$$\lambda(v)\underbrace{=}_{\text{הגדרת האלגוריתם}}\lambda(w)+1<\lambda(u^*)+1\underbrace{=}_{\substack{\text{הנחת האינדוקציה}\\ \text{עבור } u^*}}(k-1)+1=k=\delta(s,v)$$

סה"כ קיבלנו $\lambda(v)<\delta(s,v)$, וזו סתירה מלמה 1.1.

באיטרציה שבה u^* בראש התור Q, הצומת v מקיימת $\lambda(v) = \infty$, ולכן באיטרציה זו v יקבל סימון $\lambda(v) = k$ ויוכנס ל-Q.

■

2. DFS - Depth First Search

משימה: למצוא רכיבים קשירים היטב של גרף מכוון בזמן ליניארי.

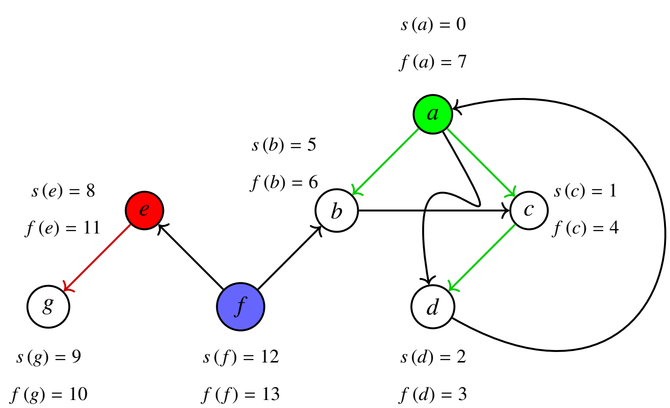
2.1. חותמות זמן: זמני גילוי וסיום של צומת (במהלך אלגוריתם סריקה).

הגדרה 1.2 $s(u)$ - זמן הגילוי של צומת u

הגדרה 1.3 $f(u)$ - זמן סיום של צומת u.

2.2. האלגוריתם.

- אתחול:
  (1) $\forall u \in V,\ \text{status}(u) \leftarrow \text{unvisited}$
  (2) $\forall u \in V,\ \begin{matrix} p(u) \leftarrow \text{NULL} \\ t \leftarrow 0 \end{matrix}$
- כל עוד יש צומת u כך ש-$\text{status}(u) = \text{unvisited}$: בצע $\text{visit}(u)$.
- $\text{visit}(u)$:
  (1) - $s(u) \leftarrow t$
  - $t \leftarrow t+1$
  - $\text{status}(u) \leftarrow \text{visited}$
  (2) לכל קשת $(u \to v) \in E$, אם $\text{status}(v) = \text{unvisited}$, אז $p(v) \leftarrow u$, וגם $\text{visit}(v)$.
  (3) $\begin{cases} f(u) \leftarrow t \\ t \leftarrow t+1 \end{cases}$



איור .3 דוגמת הרצה של אלגוריתם DFS.

מסקנה 1.1 בריצת DFS על גרף מכוון G, לכל צומת $u \in V$, $\text{visit}(u)$ יקרא בדיוק פעם אחת.

2.3. זמן ריצה.

- מה זמן הריצה של אלגוריתם ה-DFS?

- עבור צומת $u \in V$, כמה זמן לוקח לבצע $\text{visit}(u)$ ללא הקריאות הרקורסיביות (אם יש) שנובעות ממנו? - $O(1)+O(\deg_{\text{out}}(u))$.

$\Longleftarrow$ סה"כ $O(n+m)$ (ובפרט האלגוריתם עוצר).

הערה 1.2 לאלגוריתם ה-DFS דרגות חופש רבות.
חותמות הזמן s, f מהוות תיעוד של היסטוריית ריצת האלגוריתם.

הגדרה 1.4 (יער ה-DFS) נסתכל על הגרף $G_p=\left(V,E_p\right)$, כאשר:

$$E_p=\{(p(v)\to v)\in E : p(v)\neq \text{NULL}\}$$

נשים לב ש-G_p הוא תת-גרף של G.

משפט 1.2 (תרגיל) G_p הוא יער מכוון אשר פורש את כל צמתי V.

2.4. סוגי קשתות ביער ה-DFS.

שאלה 1.3 כיצד ניתן לסווג את קשתות G
בהינתן ריצה מסוימת של DFS?

הגדרה 1.5 (קשת עץ) $E \ni (u\to v)$ היא קשת עץ, אם $p(v)=u$.

הגדרה 1.6 (קשת קדמית) $(u\to v)\in E$ היא קשת קדמית, אם אינה קשת עץ,
ובנוסף u אב קדמון של v ביער ה-DFS.

הגדרה 1.7 (קשת אחורית) $(u\to v)\in E$ היא קשת אחורית, אם u צאצא של v ביער ה-DFS.

הגדרה 1.8 (קשת חוצה) כל שאר הקשתות מכונות קשתות חוצות.

הערה 1.3 כאשר מבצעים DFS על גרף לא מכוון, יווצרו רק קשתות עץ וקשתות אחוריות (ללא הוכחה).

2.5. אפיון יחסי אב-צאצא ביער ה-DFS.

למה 1.3 לכל גרף מכוון G, לכל ריצת DFS ולכל $u,v\in V$,
בדיוק אחד משלושת הבאים מתקיים:

(1) $[s(u),f(u)]$ ו-$[s(v),f(v)]$ זרים, ו-u אינו צאצא של v ו-v אינו צאצא של u.
(2) $s(v)<s(u)<f(u)<f(v)$, ו-u צאצא של v.
(3) $s(u)<s(v)<f(v)<f(u)$, ו-v צאצא של u.

הוכחה. נניח $s(u)<s(v)$ (המקרה ההפוך - סימטרי).

- מקרה ראשון: $\boxed{s(v)<f(u)}$

נרצה להראות שאנחנו במקרה ג'.
ברגע גילוי v, עדיין לא סיימנו את $\text{visit}(u)$ (בגלל ש-$s(v)<f(u)$).

$\Longleftarrow$ $\text{visit}(v)$ נקרא מתוך שרשרת קריאות רקורסיביות מתוך $\text{visit}(u)$.

$\Longleftarrow$ $\text{visit}(v)$ מסתיים לפני $\text{visit}(u)$.

$\Longleftarrow$ $f(v) < f(u)$

$\Longleftarrow$ $\boxed{s(u) < s(v) < f(v) < f(u)}$.

מדוע v הוא צאצא של u?

נוכיח באינדוקציה לפי מספר הקריאות של visit שבוצעו בין $\text{visit}(u)$ ל-$\text{visit}(v)$.

<u>בסיס</u>: $\text{visit}(v)$ בוצע ישירות מתוך $\text{visit}(u)$.

לפי הגדרת האלגוריתם, $p(v) = u$, ולכן v צאצא של u.

<u>צעד</u>: נניח כי $\text{visit}(v)$ נקרא מתוך $\text{visit}(w)$.

$\Longleftarrow$ $p(v) = w$, כלומר v הוא ילד ישיר (צאצא) של w.

לפי הנחת האינדוקציה, w הוא צאצא של u, ולכן v צאצא של u.

- <u>מקרה שני</u>: $\boxed{f(u) < s(v)}$

נרצה להראות שאנחנו במקרה א'.

חייב להתקיים:

$$s(u) < f(u) < s(v) < f(v)$$

מכיוון שלא ניתן לסיים צומת לפני שמגלים אותו.

נראה ש-v אינו צאצא של u (המקרה ההפוך - סימטרי):

אם נניח בשלילה ש-v הוא כן צאצא של u, אז צריך להתקיים ש-$\text{visit}(v)$ מתרחש בשרשרת קריאות רקורסיביות שמקורן ב-$\text{visit}(u)$, ובפרט $\text{visit}(v)$ מסתיים לפני סיום $\text{visit}(u)$, ז"א $f(u) > f(v)$ - בסתירה!

■

מסקנה 1.2 (מטענת העזר)

v צאצא של u ביער ה-DFS $\iff$ $s(u) < s(v) < f(v) < f(u)$.

משפט 1.3 (אפיון ליחסי אב-צאצא ביער ה-DFS) לכל גרף מכוון G ולכל ריצת DFS,
v צאצא של u ביער ה-DFS, **אם ורק אם** בזמן גילוי u, יש ב-G מסלול מ-u ל-v שכל הצמתים בו הן unvisited (פרט ל-u עצמו).

<u>הוכחה.</u>

<u>$\Leftarrow$</u>: נתון ש-v צאצא של u. נרצה להוכיח שברגע גילוי u
יש ב-G מסלול מ-u ל-v שמכיל רק צמתים שהם unvisited.

יהי P המסלול מ-u ל-v ביער ה-DFS (G_p).
נראה שברגע גילוי u, כל הצמתים ב-P הם unvisited.

יהי w צומת ב-P, לכן w צאצא של u ביער ה- DFS.
לפי המסקנה $s(u) < s(w)$,
ולכן ברגע גילוי u כל הצמתים ב-P הם unvisited.

<u>$\Longrightarrow$</u>: נתון שברגע גילוי u, קיים ב-G מסלול P מ-u ל-v
שכל הצמתים בו הם unvisited (באותו הרגע). נרצה להראות ש-v צאצא של u.

נניח בשלילה ש-v אינו צאצא של u, ויהי x הצומת הראשון במסלול שאינו צאצא של u (הערה: קיום x מובטח בגלל v, אחרת v צאצא של u).

יהי y הצומת הקודם ל-x במסלול (קיום y מובטח כי x בהכרח אינו הצומת הראשון ב-P). מתקיים:

$$s(u) \underbrace{<}_{\substack{\text{ברגע גילוי } u\text{, כל הצמתים} \\ \text{ב-}P\text{ הן unvisited.}}} s(x) \underbrace{<}_{\substack{\text{יש קשת מ-}y\text{ ל-}x\text{,} \\ \text{ולכן לא נסוגים מ-}y \\ \text{עד ש-}x\text{ מתגלה.}}} f(y) \underbrace{\le}_{\substack{y\text{ צאצא של }u\text{ (אולי } y = u\text{)} \\ \text{+ מהמסקנה.}}} f(u)$$

אבל לפי המסקנה x צאצא של u (שכן אינטרוולים לא יכולים להיחתך),
וזו סתירה להנחת השלילה.

■

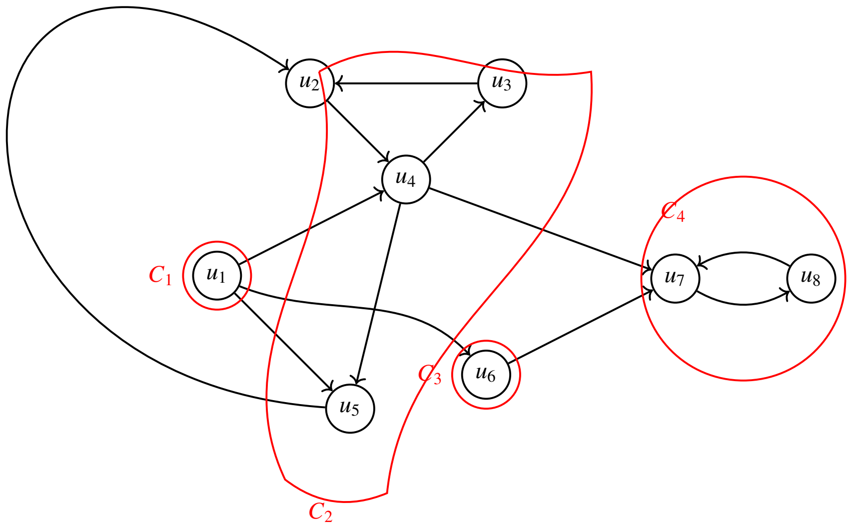
2.6. רכיבים קשירים היטב.

הגדרה 1.9 (רכיב קשיר היטב) נגדיר יחס (רלציה) על זוגות של צמתים באופן הבא:
u ו-v ביחס $\iff$

- ב- G יש מסלול מ-u ל-v.

- ב- G יש מסלול מ-v ל-u.

הרכיבים הקשירים היטב הם מחלקות השקילות של היחס הזה.

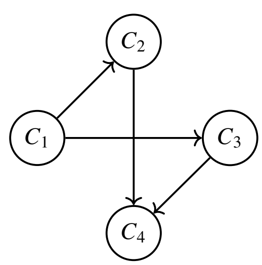


איור 4. רכיבים קשירים היטב עבור גרף לדוגמה

הגדרה 1.10 (גרף רכיבים קשירים היטב) לכל גרף מכוון G ניתן להגדיר את גרף הרכיבים הקשירים היטב שלו $\bar{G}\left(\bar{V},\bar{E}\right)$ להיות:

$$\bar{V} = \{C \mid C \text{ רכיב קשיר היטב של } G\}$$

$$\bar{E} = \left\{\left(C_i \to C_j\right) \middle| \begin{array}{l} \text{קיימים } u \in C_i \text{ ו- } v \in C_j \\ \text{כך ש-} \left(u_i \to u_j\right) \in E \end{array}\right\}$$



איור 5. גרף הרכיבים הקשירים היטב של הגרף מהאיור הקודם

דוגמה 1.1 *איור של גרף הרכיבים הקשירים היטב של הדוגמה הקודמת*

הערה 1.4 גרף רכיבים קשירים היטב הוא בהכרח חסר מעגלים מכוונים (גרף א-ציקלי),
ולכן ניתן לבצע עליו מיון טופולוגי.
באופן כללי, נוח לפתור בעיות על גרפים מסוג זה.

שאלה 1.4 (השאלה החישובית שתעניין אותנו) בהינתן גרף מכוון $G = (V, E)$,
כיצד נחשב את גרף הרכיבים הקשירים היטב שלו?

הערה 1.5 קל לפתור את הבעיה בזמן ריבועי, ע"י הרצת אלגוריתם סריקה (BFS, DFS) מכל צומת.
נרצה לפתור את הבעיה בזמן לינארי, בהתבסס על התכונות שמצאנו מקודם.

הערה 1.6 באופן כללי, מובטח שכל קשת אחורית "סוגרת מעגל".

נרצה לבחור נציג לכל רכיב קשיר היטב, שהוא:

- "קנוני".
- "הכי קדמון" $\Longleftarrow$ בעל זמן הנסיגה הגדול ביותר.

הגדרה 1.11 (הנציג של צומת u) בהינתן ריצת DFS נתונה, הנציג של צומת u זה הצומת v שישיג מ-u ב-G, בעל זמן הנסיגה $f(v)$ הגדול ביותר.
מסמנים $\varphi(u)$.

הערה 1.7 כל רכיב קשירות היטב מוכל בהכרח בעץ יחיד ביער ה-DFS (לפי המסקנה ממקודם),
אבל ההפך אינו בהכרח נכון.

למה 1.4 לכל ריצת DFS ולכל צומת u, מתקיים ש-u ו-$\varphi(u)$ באותו רכיב קשיר היטב.

הוכחה. ב-G יש מסלול מ-u ל-$\varphi(u)$ (מהגדרת נציג).
נשאר להראות שב-G יש מסלול מ-$\varphi(u)$ ל-u, ביחס לריצת DFS נתונה.

נראה קיום של מסלול שכזה על ידי כך שנוכיח ש-u הוא צאצא של $\varphi(u)$ ביער ה-DFS
(זוהי טענה חזקה יותר).

אם $u = \varphi(u)$ אז סיימנו, לכן נניח $u \neq \varphi(u)$.

מהנחה זו נובע כי $f(u) < f(\varphi(u))$.
לכן, בזמן גילוי u ע"י ה-DFS, לא יתכן שכבר נסוגנו מ-$\varphi(u)$.
לכאורה, יתכנו 2 אפשרויות:

(1) ברגע גילוי u, $\varphi(u)$ חדש (unvisited).
(2) ברגע גילוי u, $\varphi(u)$ אינו חדש, אבל עדיין לא נסוגנו מ-$\varphi(u)$.

נוכיח ש-(1) אינו אפשרי.

נניח בשלילה ש-(1) אפשרי, ויהי P המסלול לפי ההגדרה ש-$\varphi(u)$ נציג של u.

ברגע גילוי u לא יתכן שכל הצמתים ב-P חדשים
(אחרת, לפי משפט, $\varphi(u)$ צאצא של u, ולכן $f(\varphi(u)) < f(u)$, בסתירה להגדרת הנציג).

יהי v הצומת האחרון במסלול P שאינו חדש (visited) ברגע גילוי u ע"י DFS.
לכן, ברגע גילוי v, הסיפא של P מ-v ל-$\varphi(u)$ כולה חדשה (unvisited).

לכן, $\varphi(u)$ צאצא של v ביער ה-DFS, ואז:

$$f(\varphi(u)) < f(v)$$

וזו סתירה לכך ש-$\varphi(u)$ הוא הנציג של u.

לכן לפי משפט האינטרוולים, האינטגרל של u מוכל בזה של $\varphi(u)$, בפרט u צאצא שלו.

■

טענה 1.2 לכל גרף מכוון $G=(V,E)$ ולכל שני צמתים $u,v\in V$ ולכל ריצת DFS, מתקיים:

u,v באותו רכיב קשיר היטב $\iff$ $\varphi(u)=\varphi(v)$

הוכחה.

$\Leftarrow$: אוסף הצמתים שישיגים מ-u זהה לאוסף הצמתים שישיגים מ-v,
ולכן בהכרח יש ל-u,v אותו נציג.

$\Rightarrow$: לפי טענת העזר, $u,\varphi(u)$ באותו רכיב קשיר היטב.
באופן דומה, $v,\varphi(v)$ באותו רכיב קשיר היטב.
אבל $\varphi(u)=\varphi(v)$ מהנתון, ולכן u,v באותו רכיב קשיר היטב. ■

2.7. האלגוריתם למציאת רכיבים קשירים היטב.

(1) מריצים DFS על $G=(V,E)$ לקבלת זמני נסיגה $f(u)$ לכל צומת $u\in V$.
(2) נסמן ב-G^R את הגרף שמתקבל מ-G ע"י הפיכת כיווני הקשתות.
(3) מריצים DFS על G^R. כאשר מתחילים עץ חדש ביער ה-DFS, בוחרים את הצומת שנותר עם זמן הנסיגה הגדול ביותר משלב 1 של האלגוריתם.

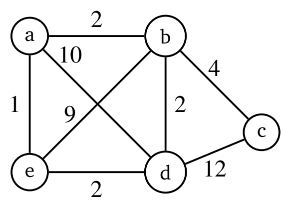
- הקלט: גרף $G=(V,E)$.
- הפלט: העצים שמתקבלים בריצת ה-DFS השנייה על G^R (שלב 5).
  הם הרכיבים הקשירים היטב של G.

פרק 2

עצים פורשים מינימליים

1. בעיות אופטימיזציה ברשתות

דוגמה 2.1 נתונה רשת התקשורת הבאה:



איור 1. על כל קשת מופיע מחיר השימוש בה.

נניח כי הצומת a מעוניין להפיץ הודעה לכל הצמתים ברשת.
יש למצוא תת קבוצת של קשתות ברשת, שעליהן ההודעה תעבור כך שתגיע לכל הצמתים.

שאלה 2.1 האם יכול להיות שבתת-הגרף שנבחר יהיו מעגלים?

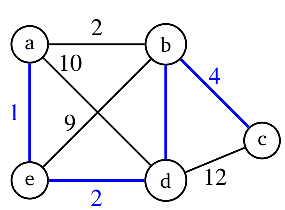
הערה 2.1 נשים לב כי היות שנרצה להשיג מחיר מינימלי, תת-הגרף שהתקבל מבחירת הקשתות הינו לבטח חסר מעגלים.

2. בעיית עץ פורש מינימום (עפ"מ)

הגדרה 2.1 (בעיית עץ פורש מינימלי) נתון גרף קשיר לא מכוון $G=(V,E)$, שבו לכל קשת (v,u) יש משקל $w(v,u)$.

יש למצוא עץ פורש של הגרף, שסך משקל הקשתות שלו מינימלי.

דוגמה 2.2 (דוגמה לעץ פורש של גרף משקלים נתון)



איור 2. עץ פורש של הדוגמה הנתונה.

- (a,e) הקשת הזולה ביותר שיוצאת מ-a, לכן נסמנה בכחול.
- נסמן את הקשת (e,d).

- נסמן את הקשת (b,d).
- נסמן את הקשת (b,d).
- נסמן את הקשת (b,c).

כאשר משקל העץ הפורש הינו 9.

ללא הוכחה, נציין שזהו גם למעשה עץ פורש מינימלי.

נבחין שזהו אינו העץ הפורש היחיד בעל משקל 9, שכן היה ניתן למשל להחליף את הקשת (e,d) בקשת (a,b) ולקבל תוצאה דומה.

3. אלגוריתמים לבעיית עץ פורש מינימום

נראה אלגוריתם גנרי, ובהמשך נציג אלגוריתמים שמתקבלים כמקרים פרטיים של אלגוריתם זה.

- הרעיון: נשתמש באלגוריתם חמדן שיבנה עפ"מ קשת אחר קשת, ע"י הוספת קשתות עם משקל נמוך והשמטת קשתות עם משקל גבוה.
- האלגוריתם יתקדם ע"י צביעת קשתות: קשתות שיצבעו בכחול יופיעו בעץ, וקשתות שיצבעו באדום יושמטו.
- האלגוריתם יקיים בכל שלב את שמורת הצבע.

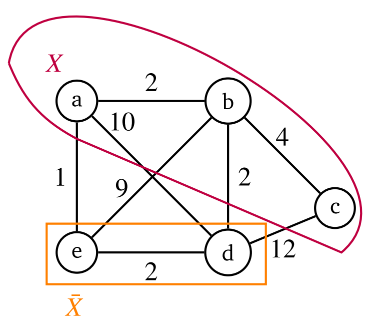
טענה 2.1 (שמורת הצבע) קיים עפ"מ שמכיל את כל הקשתות הכחולות ואף אחת מהקשתות האדומות.

מסקנה 2.1 משמורת הצבע נובע כי כאשר כל הקשתות ב-G נצבעו,
הקשתות הכחולות יוצרות עץ עפ"מ.

הגדרה 2.2 (חתך/cut) בגרף $G = (V, E)$ הוא חלוקה של קבוצת הצמתים V לשתי תתי קבוצות: X ו-$\bar{X} = V \setminus X$.

הגדרה 2.3 (חצייה של קשת את חתך הגרף) נאמר שקשת חוצה את החתך אם קצה אחד שלה ב-X והקצה האחר ב-$\bar{X}$.
לפעמים נגיד שקשת כזו תהיה קשת של החתך.

דוגמה 2.3 (דוגמה לחתך ולקשתות שחוצות אותו) נגדיר חתך ברשת: $X = \{a, b, c\}$
הקשתות שחוצות את החתך:
$\{(b,d), (a,e), (c,d), (a,d), (b,e)\}$



איור 3. החתך לדוגמה על גרף הרשת.

3.1. האלגוריתם הגנרי למציאת עפ"מ.

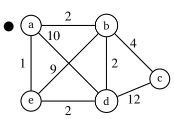
הגדרה 2.4 (הכלל הכחול) יהי $X \subseteq V$ כך שאין קשת כחולה שחוצה את $\left(X, \bar{X}\right)$.
אזי ניתן לצבוע בכחול את הקשת הקלה ביותר שאינה צבועה מבין אלו שחוצות את $\left(X, \bar{X}\right)$.

הגדרה 2.5 (הכלל האדום) יהי C מעגל שאין בו קשת אדומה.
אזי ניתן לצבוע באדום את הקשת הכבדה ביותר שאינה צבועה מבין קשתות המעגל C.

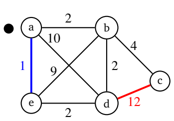
<u>האלגוריתם הגנרי:</u>

- אתחל את כל הקשתות ב-E ללא צבועות.
- כל עוד יש ב-E קשת שאינה צבועה, הפעל את הכלל הכחול או האדום לצביעת אחת מהקשתות.
- הקשתות הכחולות הן עפ"מ.

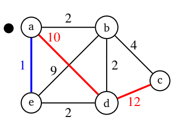
דוגמה 2.4 (דוגמת הרצה)
נחזור לדוגמת הרשת, ונבצע דוגמת הרצה של האלגוריתם החמדן:



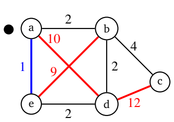
- הכלל הכחול: עם $X = \{a\}$.
  נבחר את הקשת הקלה ביותר שאינה צבועה היוצאת מ-a ונצבע אותה בכחול: זוהי (a, e).



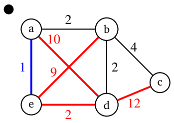
- הכלל האדום על $\{b, c, d\}$:
  נבצע באדום את הקשת הכבדה ביותר במעגל חסר הקשתות האדומות $d \to c \to b$, זוהי (d, c).



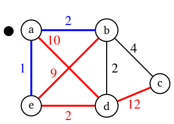
- הכלל האדום על $\{a, b, d\}$:
  נבצע באדום את הקשת הכבדה ביותר במעגל חסר הקשתות האדומות $d \to b \to a$, זוהי (a, d).



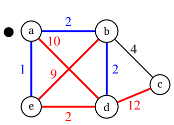
- הכלל האדום על $\{e, b, d\}$:
  נבצע באדום את הקשת הכבדה ביותר במעגל חסר הקשתות האדומות $d \to b \to e$, זוהי (e, b).



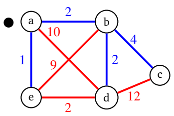
- הכלל האדום על $\{a, b, d, e\}$:
  נבצע באדום את הקשת הכבדה ביותר במעגל חסר הקשתות האדומות $e \to d \to b \to a$.
  ניתן לבחור כל אחת מהקשתות בעלות משקל 2, למשל (d, e).



- הכלל הכחול: עם $X = \{a, e\}$.
  נבחר את הקשת הקלה ביותר שאינה צבועה היוצאת מ-$\{a, e\}$ ונצבע אותה בכחול: זוהי (a, b).



- הכלל הכחול: עם $X = \{c, d\}$.
  נבחר את הקשת הקלה ביותר שאינה צבועה היוצאת מ-$\{c, d\}$ ונצבע אותה בכחול: זוהי (b, d).



- הכלל הכחול: עם $X = \{c\}$.
  נבחר את הקשת הקלה ביותר שאינה צבועה היוצאת מ-$\{c\}$ ונצבע אותה בכחול: זוהי (b, c).

ואכן, קיבלנו כי משקל העפ"מ הינו 9.

הערה 2.2 (ללא הוכחה) בכל שלב באלגוריתם הגנרי, הקשתות הכחולות יוצרות יער של עצים כחולים, שכן הקשתות הכחולות תמיד מוכלות באיזשהו עפ"מ של הגרף.

הגדרה 2.6 (אי שייכות צומת לעץ כחול) נאמר שצומת v אינו בעץ כחול, אם לא קיימת קשת שנוגעת בצומת v שצבועה בכחול.

3.1.1. נכונות האלגוריתם הגנרי.

שאלה 2.2 האם האלגוריתם תמיד מצליח לצבוע את כל הקשתות?

שאלה 2.3 האם מובטח שבסיום האלגוריתם הקשתות הכחולות יגדירו עפ"מ?

למה 2.1 (הבחנה על עצים פורשים) יהי T עץ פורש של G.
אם נוסיף ל-T קשת $e \notin T$, נקבל ב-T מעגל יחיד C.
אם נשמיט מ-C קשת, בוודאות נקבל שוב עץ פורש של G.

משפט 2.1 (נכונות האלגוריתם הגנרי) קיים עפ"מ T שמכיל את כל הקשתות הכחולות ואף אחת מהקשתות האדומות.

משפט 2.2 (כל הקשתות נצבעות + נכונות שמורת הצבע)
האלגוריתם הגנרי צובע את כל הקשתות של **גרף קשיר** G, ומקיים את "שמורת הצבע".

הוכחת המשפט.

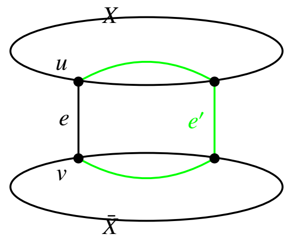
(1) הראינו כי האלגוריתם מקיים את שמורת הצבע אחרי הפעלה של הכלל הכחול.

■

הוכחה. נראה תחילה כי האלגוריתם מקיים את השמורה, באינדוקציה על מספר האיטרציות (הפעלות של הכלל האדום או הכחול):
בסיס האינדוקציה: בתחילה אף קשת לא צבועה, ולכן כל עפ"מ ב-G מקיים את השמורה (כלומר, הטענה נכונה באופן ריק).

צעד האינדוקציה: נטפל לחוד בשני מקרים:

(1) נניח כי השמורה מתקיימת לפני הפעלה של הכלל הכחול.
תהי e קשת שנצבעת כעת בכחול, ויהי T עפ"מ שקיים את השמורה לפני שהקשת e נצבעה.
אם $e \in T$, אזי T מקיים את השמורה אחרי שהקשת e נצבעה (סיימנו).
אם $e \notin T$, נסתכל על החתך $X, \bar{X}$ שעליו הפעלנו את הכלל הכחול.



יש מסלול בעץ T שמחבר בין הצמתים u, v בקצוות של הקשת e.
היות ש-e חוצה את החתך, קיימת על המסלול הנ"ל קשת אחרת e' שחוצה את החתך.

מהנחת האינדוקציה אין ב-T קשת אדומה, שכן הוא מקיים את שמורת הצבע.
מהכלל הכחול (בחרנו חתך ללא קשתות חוצות כחולות), נקבל גם כי e' לא צבועה בכחול.
לכן, e' אינה צבועה.

בנוסף, בהכרח $w(e') \geq w(e)$ (שכן e נבחרה להיות הקשת החוצה בעלת משקל מינימלי. למעשה, בהכרח יש שוויון).
לכן, נוכל להשמיט את הקשת e' מהעץ T ולהוסיף במקומה את e.

נשים לב כי אם המסלול היחיד בין שני צמתים ב-T עבר קודם דרך e', המסלול יעבור כעת דרך e. בנוסף, T נשאר עפ"מ, כי המשקל הכולל של הקשתות בעץ לא עלה.

אם נצבע כעת את e בכחול, נקבל כי השמורה מתקיימת עבור T החדש.

(2) נניח כי השמורה מתקיימת לפני הפעלה של הכלל האדום.
תהי E קשת שנצבעת כעת באדום, ויהי T עפ"מ שמקיים את השמורה לפני שהקשת e נצבעת.

אם $e \notin T$, אזי T מקיים את השמורה גם אחרי שהקשת e נצבעת.

נניח ש-$e \in T$. אזי, השמטת e מ-T מחלקת את T לשני עצים ומגדירה חלוקה של הצמתים ב-G.

איור
המעגל שעליו הפעלנו את הכלל האדום מכיל מסלול נוסף בגרף G מ-u ל-v.

על המעגל יש קשת $e' = (x, y)$, שחוצה את החתך (T_1, T_2).

מהשמורה נובע ש-e' אינה כחולה כי $e' \notin T$, ומהכלל האדום נובע כי e' גם אינה אדומה.

בנוסף, מהכלל האדום נובע $w(e') \leq w(e)$.

הוספת e' ל-T והשמטת e יוצרת עץ פורש חדש (מבחנה 1, אם נוסיף... אם נשמיט...□).
בנוסף, לא הגדלנו את משקל העץ. לכן T החדש עפ"מ.

נראה כעת כי האלגוריתם צובע את כל הקשתות ב-G:
נניח בשלילה שיש קשת e לא צבועה, אבל אי אפשר להפעיל אף אחד מהכללים.
לפי הכלל הכחול, הקשתות הכחולות יוצרות יער של עצים כחולים, שכן הקשתות הכחולות תמיד מוכלות באיזשהו עפ"מ של הגרף.
נבחין בין 3 מקרים לגבי $e = (u, v)$:

(1) אם שני הקצוות של e באותו עץ כחול, אזי נקבל: *איור*
מצאנו בגרף G מעגל בלי קשתות אדומות, שמכיל את e ואת המסלול בעץ הכחול בין u ל-v.
לכן ניתן להפעיל את הכלל האדום.

(2) אם הקצוות של e בעצם כחולים שונים: *איור*
נסמן ב-X את קבוצת הצמתים ב-T_1, וב-$V \setminus X$ את שאר הצמתים.
קיבלנו חתך ב-G שאין בו קשתות כחולות, לכן נוכל להפעיל את הכלל הכחול.

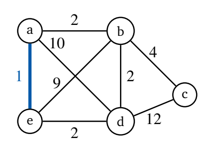
(3) לפחות קצה אחד של e אינו בעץ כחול, בה"כ נניח כי $e = (u, v)$ ו-v אינו בעץ כחול.
נגדיר $X = \{v\}$ ו-$\bar{X} = V \setminus X$.
מצאנו חתך ללא קשת כחולה, ולכן ניתן להפעיל את הכלל הכחול.

$\Longleftarrow$ כל עוד יש ב-G קשת לא צבועה, מובטח שנוכל להפעיל את אחד הכללים, ולכן האלגוריתם צובע את כל הקשתות.

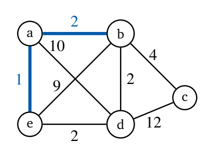
■

3.2. האלגוריתם של Prim. נתון גרף קשיר לא מכוון $G = (V, E)$.
(1) אתחול: כל הקשתות לא צבועות, נבחר $T \leftarrow \{r\}$ כאשר r צומת כלשהי.
(2) כל עוד $T \neq V$, בצע:
- תהי $e = (u, v)$ הקשת הקלה ביותר שחוצה את החתך $(T, V \setminus T)$, כך ש-$u \in T$.
- צבע את e בכחול ובצע $T \leftarrow T \cup \{v\}$.

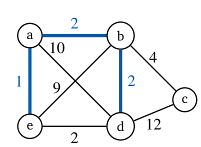
דוגמת הרצה:



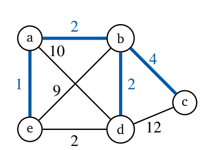
- נבחר $T = \{a\}$. הקשת הקלה ביותר שחוצה את החתך $(\{a\}, \{b, e, d, c\})$ היא $a \to e$,
נצבע אותה בכחול ונבצע $T \leftarrow \{a, e\}$.



- עבור $T = \{a, e\}$. קשת קלה ביותר שחוצה את החתך $(\{a, e\}, \{b, d, c\})$ היא $a \to b$,
נצבע אותה בכחול ונבצע $T \leftarrow \{a, b, e\}$.



- עבור $T = \{a, b, e\}$. קשת קלה ביותר שחוצה את החתך $(\{a, b, e\}, \{d, c\})$ היא $b \to d$,
נצבע אותה בכחול ונבצע $T \leftarrow \{a, b, d, e\}$.



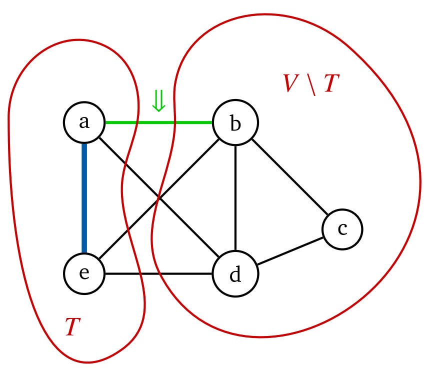
- עבור $T = \{a, b, d, e\}$. הקשת הקלה ביותר שחוצה את החתך $(\{a, b, d, e\}, \{c\})$ היא $b \to c$,
נצבע אותה בכחול ונבצע $T \leftarrow \{a, b, c, d, e\}$.

- מתקיים $T = V$, לכן הגענו לעצירה.

משפט 2.3 (נכונות אלגוריתם Prim) T הוא עפ"מ ב-G.

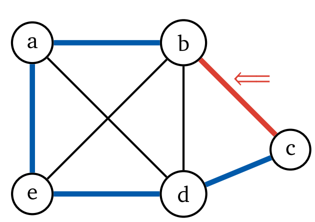
<u>הוכחה</u>. נראה כי האלגוריתם של Prim הוא מימוש של האלגוריתם הגנרי.
נתייחס לכל קשת שלא נוספה ל-T בסיום האלגוריתם כקשת שנצבעת באדום.

נסתכל על קשת $e = (u, v)$ שהאלגוריתם מוסיף ל-T באיטרציה כלשהי.
באיטרציה זו, אין קצת כחולה שחוצה את החתך $(T, V \setminus T)$.
בנוסף, הקשת e חוצה את החתך, והיא הקלה ביותר שחוצה את החתך הנ"ל ("בין אלו שאינן צבועות").
לכן צביעת e היא חוקית לפי הכלל הכחול.



איור 4. נבחר את הקשת הקלה ביותר שחוצה את החתך $(T, V \setminus T)$.

נבחן את הקשתות שאינן ב-T בסיום האלגוריתם.
כל קשת e <u>שלא נוספה ל-T</u> סוגרת מעגל ב-T. במעגל זה הינה הקשת היחידה שאינה צבועה, ושאר הקשתות בהכרח כחולות.
לכן צביעת e באדום היא הפעלה חוקית של הכלל האדום.



איור 5. כל קשת שלא נוספה ל-T סוגרת מעגל בעץ. ניתן להפעיל את הכלל האדום.

סה"כ קיבלנו שהאלגוריתם של Prim הוא מימוש ספציפי של האלגוריתם הגנרי.

■

3.2.1. סיבוכיות אלגוריתם Prim.

שאלה 2.4 מהי סיבוכיות האלגוריתם?
המפתח לשימוש יעיל של אלגוריתם Prim הוא לבחור בקלות קשת בעלת משקל מינימלי מבין אלו שחוצות את החתך.
נחזיק את כל הצמתים שאינן ב-T בתור עדיפויות Q.

לכל צומת v מוחזק מפתח $\text{key}(v)$ - המשקל המינימלי של איזושהי קשת שמחברת את v ל-T.
אם אין קשת כזאת, אז מסמנים $\text{key}(v)=\infty$. בסיום האלגוריתם התור ריק.

אם $v\in Q$ ו-$\text{key}(v)\neq\infty$, אז $\left(v,\underbrace{\pi(v)}_{\pi(v)\text{ היא צומת ב-}T}\right)$ היא הקשת הקלה בין v ל-T.

נממש את Q בעזרת ערימת מינימום (heap).

שאלה 2.5 מה הפעולות שנבצע על הערימה?

(1) <u>אתחול</u>: לכל צומת $v\in G$ מגדירים: $\begin{matrix}\text{key}(v)\leftarrow\infty\\ \pi(v)\leftarrow\text{NULL}\end{matrix}$
(2) מצא בערימה את המפתח המינימלי, נניח כי הינו שייך לצומת $u\in V\setminus T$.
(3) הוסף את u ל-T.
(4) לכל שכן v של u כך ש-$v\notin T$,
אם $w(u,v)<\text{key}(v)$, בצע פעולת Decrease Key, ועדכן $\pi(v)\leftarrow u$.

<u>סיבוכיות כל אחד מהשלבים:</u>

- צעד (1): מתבצע ב-$O(|V|)$.
- הוצאת המפתח המינימלי בצעד (2), ב-$O(\log|V|)$.
- תתבצע $|V|$ פעמים, סה"כ $O(|V|\log|V|)$.
- פעולת Decrease Key בצעד (4) תתבצע לכל היותר עבור כל השכנים של כל צומת, לכל היותר $|E|$ פעמים.

<u>סיבוכיות האלגוריתם:</u>

$$O(|V|\log|V|)+O(|E|\log|V|)=O(|E|\log|V|)$$

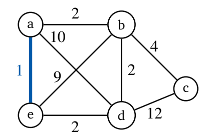
3.3. האלגוריתם של Kruskal.

(1) כל הקשתות לא צבועות, מיין את הקשתות בסדר לא יורד לפי משקלן, $F=\emptyset$.
(2) תהי e הקשת הבאה לפי סדר המיון.
אם e סוגרת מעגל בעץ כחול, צבע אותה באדום.
אחרת צבע את e בכחול, ובצע הוספה של e ל-F.
(3) החזר כפלט את אוסף הקשתות ב-F.

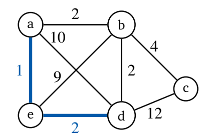
דוגמת הרצה:

נקבל מיון של הקשתות:

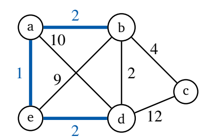
$$\{\underset{\substack{\uparrow\\ \text{מינימלית}}}{(a\to e)},(e\to d),(a\to b),(b\to d),(b\to c),(b\to e),(a\to d),(c\to d)\}$$



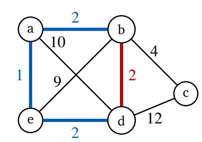
- נבחר בקשת $(a\to e)$. קשת זו לא סוגרת מעגל בעץ כחול, נבצע $F\leftarrow F\cup\{a\to e\}$.



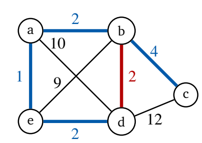
- נבחר בקשת $(e\to d)$. קשת זו לא סוגרת מעגל בעץ כחול, נבצע $F\leftarrow F\cup\{e\to d\}$.



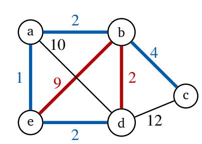
- נבחר בקשת $(a\to b)$. קשת זו לא סוגרת מעגל בעץ כחול, נבצע $F\leftarrow F\cup\{a\to b\}$.



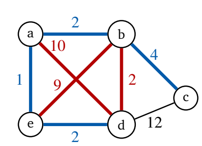
- נבחר בקשת $(b\to d)$. קשת זו סוגרת מעגל בעץ כחול, לכן לא נוסיף אותה ל-F.



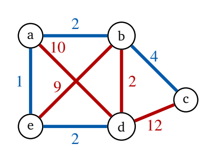
- נבחר בקשת $(b\to c)$. קשת זו לא סוגרת מעגל בעץ כחול, נבצע $F\leftarrow F\cup\{b\to c\}$.



- נבחר בקשת $(b\to e)$. קשת זו סוגרת מעגל בעץ כחול, לכן לא נוסיף אותה ל-F.



- נבחר בקשת $(a\to d)$. קשת זו סוגרת מעגל בעץ כחול, לכן לא נוסיף אותה ל-F.



- נבחר בקשת $(c\to d)$. קשת זו סוגרת מעגל בעץ כחול, לכן לא נוסיף אותה ל-F.

- סיימנו לעבור על קשתות G לפי סדר המיון.

נחזיר כפלט את $F=\{(a\to e),(e\to d),(a\to b),(b\to c)\}$.

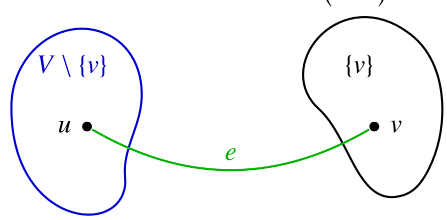
משפט 2.4 (נכונות) הגרף (V,F) שמורכב מכל הצמתים ב-G ומהקשתות F שאלגוריתם Kruskal החזיר, הוא עפ"מ ל-G.

הוכחה. נראה כי Kruskal מבצע הפעלה חוקית של הכלל הכחול או האדום.

אם e סוגרת מעגל בעץ כחול, אז מצאנו מעגל שאין בו קשתות אדומות.
היות ש-e היא היחידה שאינה צבועה, היא גם המקסימלית בין הקשתות שאינן צבועות על המעגל.
נפעיל את הכלל האדום.

אם $e=(u,v)$ לא סוגרת מעגל ב-F, אז נבחין בין שני מקרים:

(1) קצה אחד של e אינו בעץ כחול, בה"כ נניח כי זהו הצומת v.
נגדיר $X=\{v\}$ ואת $\bar{X}=V\setminus X$.
מצאנו חתך שלא חוצות אותו קשתות כחולות. נפעיל את הכלל הכחול.
מבין הקשתות הלא צבועות שחוצות את $\left(X,\bar{X}\right)$, היא בעלת משקל מינימלי (בגלל המיון).



איור 6. אם קצה אחד של e אינו בעץ כחול, ניתן להגדיר חתך ולהפעיל את הכלל הכחול על e.

(2) בשני הקצוות של e יש עצים כחולים.
ניקח עץ מאחד הקצוות, נסמנו ב-T_1, ונגדיר $X=T_1$, ואת $\bar{X}=V\setminus T_1$.

קיבלנו חתך שלא חוצות אותו קשתות כחולות.
הקשת e בעלת משקל מינימלי בין הקשתות שחוצות את החתך $\left(X,\bar{X}\right)$ ואינן צבועות, עקב המיון. נפעיל את הכלל הכחול.
לכן Kruskal הוא מימוש ספציפי של האלגוריתם הגנרי, ולכן T עפ"מ.

■

סיבוכיות האלגוריתם של Kruskal: נשתמש בקבוצות לייצוג עצים כחולים, נבצע על מבנה הנתונים את הפעולות הבאות:

- Make-Set(v) - יצירת קבוצה שמכילה רק את הצומת v.
- Find-Set(v) - מציאת הקבוצה שמכילה את הצומת v.
- Union(u,v) - איחוד הקבוצה של u עם הקבוצה של v.

(1) נסתכל על הקשת הבאה ברשימה לפי סדר המיון. נבצע Find-Set(u) ו-Find-Set(v).
(2) אם u ו-v באותה קבוצה, נבצע את e באדום. אחרת, נבצע Union(u,v) ונבצע את e בכחול.

- זמן המיון של הקשתות: $O\left(|E|\log|E|\right)\underbrace{=}_{|E|=O\left(|V|^2\right)}O\left(|E|\log|V|\right)$.
- ניתן לממש $O\left(|V|\right)$ פעולות Make-Set באתחול, ועוד $|E|$ פעולות Find-Set ו-Union בסיבוכיות:

$$O\left(\left(|V|+|E|\right)\log|V|\right)=O\left(|E|\log|V|\right)$$

סה"כ זמן הריצה הוא $O\left(|E|\log|V|\right)$.

פרק 3

מסלולים קלים ביותר

מוטיבציה: נדמיין גרף מכוון שיש בו "משקלים" על הקשתות. המשקלים יכולים לציין מרחק פיזי, זמן או עלות. בהקשר כזה, היינו מתעניינים במציאת מסלולים בין צמתים שהם בעלי סכום משקלים מינימלי.

1. מסלולים קלים ביותר בגרפים מכוונים ממושקלים

שאלה 3.1 מה הדרך הקצרה ביותר להגיע מצומת s לצומת t בגרף?

ראינו ש-BFS ימצא את המסלול הקצר כאשר אורך המסלול נמדד לפי מספר הקשתות בו.

נתון גרף מכוון $G = (V, E)$ ופונקציית משקל על הקשתות $w : E \to \mathbb{R}$, משקלים כלשהם על הקשתות ב-G.

שאלה 3.2 מה האורך של מסלול P בגרף מכוון ממשוקל?

הגדרה 3.1 (אורך של מסלול בגרף מכוון ממושקל) עבור גרף מכוון $G = (V, E)$, אורך המסלול P זהו סכום משקלי הקשתות על המסלול, דהיינו:

$$w(P) \triangleq \sum_{e \in P} w(e)$$

הגדרה 3.2 (מסלול קל ביותר בגרף מכוון ממושקל)

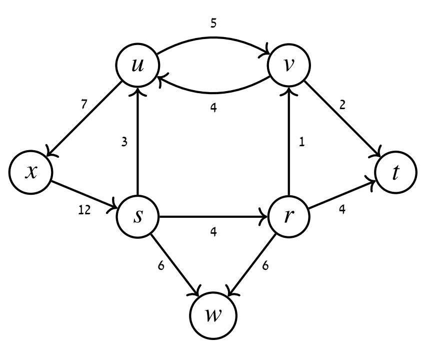
יהי $\delta(u, v)$ אורך מסלול קל ביותר ב-G מצומת u לצומת v, אזי:

$$\delta(u, v) = \begin{cases} \min\left\{ w(P) : \ u \xrightarrow{P} v \right\} & \text{אם } v \text{ ישיג מ-} u \text{ ב-} G \\ \infty & \text{אחרת} \end{cases}$$

♥ מסלולים קלים ביותר לא בהכרח מוגדרים כשיש מעגלים שליליים בגרף G!

נרצה לחשב מסלולים קלים ביותר בגרף מכוון נתון.

דוגמה 3.1 (דוגמה לסיבה שמסלול קל ביותר עלול להיות לא מוגדר היטב כאשר יש מעגלים שליליים)



איור 1. גרף מכוון ממושקל לדוגמה

אורך מסלול קל ביותר $s \to t$ הינו 7: $\boxed{s \to r \to v \to t}$

עתה, נניח שנשנה $w(u,v) \leftarrow (-5)$.

- נסתכל על המסלול:

$$s \xrightarrow{3} u \xrightarrow{-5} v \xrightarrow{2} t$$

אורך המסלול: 0.

- אם ניקח את המסלול:

$$s \xrightarrow{3} u \xrightarrow{-5} v \xrightarrow{4} u \xrightarrow{-5} v \xrightarrow{2} t$$

נקבל אורך מסלול (-1)!

- היות שהמשקל של המעגל $v \rightleftarrows u$ הוא (-1), אין מסלול קל ביותר מ-s ל-t, ולכן ניתן לעבור על המעגל השלילי מספר לא חסום של פעמים ולהגיע לאורך $(-\infty)$.

2. מבנה אופטימלי של מסלולים קלים

הגדרה 3.3 (תת-מסלול בגרף מכוון) יהי $P = \langle v_1, v_2, \ldots, v_k \rangle$ מסלול כלשהו בגרף מכוון G.
עבור $1 \le i < j \le k$, נגדיר את תת המסלול של P מ-v_i ל-v_j להיות:

$$P_{ij} = \left\langle v_i, v_{i+1}, \ldots, v_j \right\rangle$$

למה 3.1 (תכונת מסלול קל ביותר עוברת לתת-מסלול) יהי $P = \langle v_1, v_2, \ldots, v_k \rangle$ מסלול קל ביותר מצומת v_1 לצומת v_k, ויהי P_{ij} תת המסלול של P מ-v_i ל-v_j, עבור $1 \le i < j \le k$ כלשהם.
אזי P_{ij} הוא מסלול קל ביותר מ-v_i ל-v_j.

<u>הוכחת הלמה</u>. אם נפרק את המסלול P לתתי-מסלולים:

$$v_1 \overset{P_{1i}}{\rightsquigarrow} v_i \overset{P_{ij}}{\rightsquigarrow} v_j \overset{P_{jk}}{\rightsquigarrow} v_k$$

אזי:

$$w(P) = w(P_{1i}) + w\left(P_{1j}\right) + w\left(P_{jk}\right)$$

נניח בשלילה שקיים מסלול P'_{ij} מ-v_i ל-v_j, כך ש-$w\left(P'_{ij}\right) < w\left(P_{ij}\right)$.
אזי קיים מסלול מ-v_1 ל-v_k:

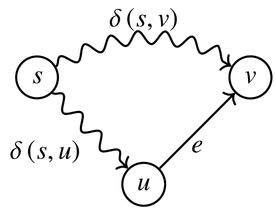
$$v_1 \overset{P_{1i}}{\rightsquigarrow} v_i \overset{P'_{ij}}{\rightsquigarrow} v_j \overset{P_{jk}}{\rightsquigarrow} v_k$$

קיבלנו שאורכו קטן מ-$w(P)$, בסתירה למינימליות של P.

$\blacksquare$

טענה 3.1 (אי שוויון המשולש עבור מסלולים קלים ביותר) יהיה $G = (V, E)$ גרף מכוון ו-$w : E \to \mathbb{R}$, ונניח שב-G אין מעגלים שליליים.
אזי לכל קשת $e = (u \to v)$:

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



איור 2. אי שוויון המשולש בטרמינולוגיה של מסלולים קלים ביותר.

<u>הוכחה</u>. אם לא קיים מסלול מ-s ל-u ב-G, אז $\delta(s,u)=\infty$. לא משנה מה ערך w_e, אי השוויון יתקיים.

אחרת יש ב- G מסלול מ-s ל- u, ומכיוון שאין ב-G מעגלים שליליים, אז $\delta(s,u)$ סופי.

יהי P מסלול קל ביותר ב-G מ-s ל-u, כלומר סך משקלי הקשתות ב-P שווה ל-$\delta(s,u)$.
נסתכל על המסלול הבא מ-s ל-v:

$$s \overset{P}{\rightsquigarrow} u \xrightarrow{e} v$$

שמשקלו $\underbrace{w_e}_{e \text{ משקל}} + \underbrace{\delta(s,u)}_{P \text{ משקל}}$. מהגדרה, גודל זה חוסם ממעלה את $\delta(s,v)$, אורך המסלול הקל ביותר מסוג זה (בין s ל-v ב-G).

■

3. פתרון הבעיה האלגוריתמית בגרף ללא מעגלים שליליים

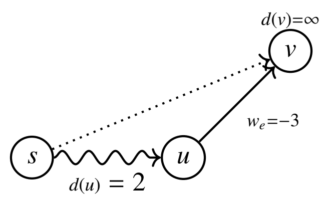
הערה 3.1 (הבעיה האלגוריתמית כפי שהגדיר אותה רועי)

- נתון: $\left.\begin{array}{l} \text{גרף מכוון } G=(V,E) \\ \text{צומת התחלה } s \in V \end{array}\right\}$
- מטרה: למצוא אורך מסלול קל ביותר $\delta(s,u)$ לכל צומת $u \in V$ בגרף.

שאלה 3.3 (כיצד נפתור את הבעיה האלגוריתמית?) אם אין מעגלים שליליים בגרף, האם יש צומת u עבורו יודעים את $\delta(s,u)$?
צומת שכזה הוא s עצמו, והוא מקיים: $\delta(s,s)=0$.

מכיוון שהמרחקים (δ-ות) מקיימים את אי-שוויון המשולש, נדע שאין לנו את התשובה הנכונה,
אם יש קשת מכוונת שמפרה את אי-שוויון המשולש.

("אורך מסלול קל ביותר" נוכחי באלגוריתם)



שלב ביניים אפשרי באלגוריתם שנרצה לממש.
מתקיים $\infty = d(v) \nleq d(u)+w_e$, לכן יש כאן הפרה של אש"מ.

אלגוריתם (הכללי למציאת מסלולים קלים ביותר מ-s, ע"י בדיקת הפרות של אי-שוויון המשולש):

- אתחול: $d(u) \leftarrow \begin{cases} \infty & u \neq s \\ 0 & u = s \end{cases}$, לכל $u \in V$.
- כל עוד יש קשת $e=(u \to v) \in E$ כך שמתקיים: $\overbrace{d(v) > d(u)+w_e}^{\text{הפרה של אי שוויון המשולש עבור הסימונים } d \text{ של האלגוריתם והקשת } e}$, בצע $d(v) \longleftarrow d(u)+w_e$.

משפט 3.1 (נכונות האלגוריתם למציאת מסלולים קלים ביותר)
יהי $G=(V,E)$ גרף מכוון, $w: E \to \mathbb{R}$ פונקציית משקל על הקשתות, ויהי $s \in V$.
אם ב-G אין מעגלים שליליים, אז אם באיזשהו שלב של ריצת האלגוריתם הכללי, לכל קשת $(u \to v) \in E$ מתקיים:

$$d(v) \leq d(u) + w_{(u \to v)}$$

אז $d(v)=\delta(s,v)$ לכל $v \in V$.

הוכחה. בשלב ראשוני של ההוכחה, אנחנו נראה שבכל רגע של ריצת האלגוריתם הכללי, מתקיים:

$$\forall v \in V, \quad d(v) \geq \delta(s,v) \qquad \left(\begin{array}{c}\text{האלגוריתם אף פעם} \\ \text{לא "מפספס"} \\ \text{כלפי מטה.}\end{array}\right)$$

את זה נוכיח באינדוקציה על סדר פעולות האלגוריתם הכללי.

- בסיס: אתחול האלגוריתם.

עבור s, מתקיים $\delta(s,s)=0$ (בגלל שאין מעגלים שליליים ב-G) ואכן $d(s)=0$.

לכל $v \neq s$, מתקיים: $d(v) = \infty$ ולכן הטענה מתקיימת.

- צעד: נניח שהטענה נכונה עד תחילת האיטרציה הנוכחית, ובאיטרציה זו מבצעים עדכון של $d(v)$ בגלל הקשת $e = (u \to v)$.
בסיום האיטרציה, עבור כל צומת שאינו v הטענה מתקיימת עבורו.
עבור v עצמו:

$$d(v) \underbrace{=}_{\text{הגדרת האלגוריתם}} d(u) + w_e \underbrace{\geq}_{\text{הנחת האינדוקציה}} \delta(s,u) + w_e \underbrace{\geq}_{\text{אש"מ}} \delta(s,v)$$

לכן, בכל רגע של ריצת האלגוריתם הכללי, מתקיים $\forall v \in V \;\; d(v) \geq \delta(s,v)$.

בשביל לסיים את הוכחת המשפט, מספיק להראות שאם תנאי העצירה מתקיים $\forall e = (u \to v) \in E, \;\; d(v) \leq d(u) + w_e$, אזי:

$$\forall v \in V, \;\; d(v) \leq \delta(s,v)$$

נניח בשלילה שקיים צומת v, עבורו מתקיים $d(v) > \delta(s,v)$ (וכמובן תנאי העצירה מתקיים).
מכיוון שמתקיים $\delta(s,v) < \infty$ (לפי הנחת השלילה), וגם $\delta(s,v) \neq -\infty$ (כפי ב-G אין מעגלים שליליים), אז $\delta(s,v)$ סופי.
יהי $s \overset{P}{\rightsquigarrow} v \subseteq E$ מסלול קל ביותר כלשהו ב-G מ-s ל-v:

$$P \;\; = \;\; \underset{\underset{s}{\shortparallel}}{v_0} \to v_1 \longrightarrow v_2 \ldots \longrightarrow v_{k-1} \longrightarrow \underset{\underset{v}{\shortparallel}}{v_k}$$

- עבור s מתקיים $d(s) = \delta(s,s) = 0$.
- עבור v מתקיים $d(v) > \delta(s,v)$ (הנחת השלילה).

לכן ניתן להסיק מהחלק הראשון של ההוכחה ("אין פספוס כלפי מעלה"), שיש קשת $e = (v_i \to v_{i+1})$,
עבורה $d(v_i) = \delta(s,v_i)$, וגם $d(v_{i+1}) > \delta(s,v_{i+1})$.

מתקיים:

$$d(v_{i+1}) \underbrace{>}_{\substack{\text{הנחת}\\\text{השלילה}}} \delta(s,v_{i+1}) \underbrace{=}_{\substack{P \text{ כל תת מסלול של}\\\text{גם הוא קל ביותר}}} \delta(s,v_i) + w_e = d(v_i) + w_e \underbrace{\geq}_{\substack{\text{תנאי העצירה}\\ e \text{ עבור קשת}}} d(v_{i+1})$$

וזו סתירה להנחת השלילה.

■

4. עץ מסלולים קלים ביותר

שאלה 3.4 אם האלגוריתם הכללי עצר,
כיצד ניתן לשחזר איזשהו מסלול קל ביותר מ-s לצומת v, בהנחה ש-$\delta(s,v)$ סופי?

הגדרה 3.4 (עץ מסלולים קלים ביותר) בהינתן גרף מכוון $G=(V,E)$, $w:E\to\mathbb{R}$ וצומת $s\in V$, בהנחה שאין מעגלים שליליים ב-G,
נגדיר עץ מסלולים קלים ביותר להיות תת גרף $G'=(V',E')$ של G כך שמתקיים:

(1) V' הוא אוסף הצמתים הישיגים מ-s ב-G.
(2) G' הוא עץ מכוון ששורשו s.
(3) לכל $u\in V'$, המסלול היחידי מ-s ל-u ב-G' הוא מסלול קל ביותר מ-s ל-u ב-G.

שאלה 3.5 מה נוסיף לאלגוריתם הכללי בשביל שנקבל עץ מסלולים קל ביותר?

אלגוריתם (התוספת לאלגוריתם הכללי לצורך מציאת עץ המסלולים):

- אתחול: $\forall v\in V,\ \ \pi(v)\leftarrow \text{NULL}$
- באיטרציות: אם עִדְכַּנּוּ את $d(v)$ בגלל קשת $e=(u\to v)$, אזי $\pi(v)\leftarrow u$.

טענה 3.2 (נכונות ; ללא הוכחה) אם ב-G אין מעגלים שליליים, אז ברגע שהאלגוריתם הכללי עוצר, מתקיים:

$$V_\pi \triangleq \{u\in V \ :\ \pi(u)\neq \text{NULL}\}\cup\{s\}$$
$$E_\pi \triangleq \{(\pi(v)\to v)\ :\ v\in V_\pi\setminus\{s\}\}$$

הגרף $G_\pi=(V_\pi,E_\pi)$ הוא עץ מסלולים קלים ביותר של G מ-s.

5. אלגוריתמים למסלולים קלים ביותר ממקור יחיד

שאלה 3.6 באיזה סדר כדאי לעבור על הקשתות?

שאלה 3.7 האם האלגוריתם הכללי תמיד עוצר?

שאלה 3.8 איך יודעים אם ב-G יש מעגל שלילי?

התשובה לשאלות הללו תלויה במה שידוע על המשקלים w.

- המקרה הראשון (והחשוב) שנטפל בו הוא כאשר $\forall e\in E,\ \ w_e\geq 0$ (משקלים אי שליליים).
- המקרה השני יהיה המקרה הכללי (אין מגבלה על המשקלים).

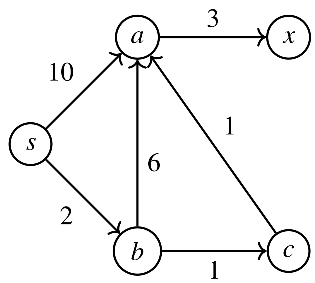
באופן כללי, נראה עבור שני המקרים מימוש מסוים (ומהיר) של האלגוריתם הכללי.

הערה 3.2 המקרה הראשון חשוב ביותר כי חלק גדול מהאפליקציות בעולם האמיתי מתייחסות למקרה הזה:
יתכן שנייחס למשקלים משמעות של מרחק פיזי, זמן או delay (למשל ברשתות תקשורת), כאשר כל אלה יחידות מידה אי-שליליות.

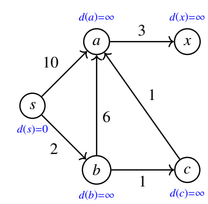
5.1. אינטואיציה למשקלים אי שליליים. מקרה ראשון: ($w_e\geq 0$ לכל קשת $e\in E$)

ברור שאין מעגלים שליליים בגרף.

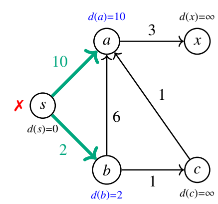
דוגמה 3.2 (פיתוח אינטואיציה לאלגוריתם הכללי עם משקלים אי שליליים)



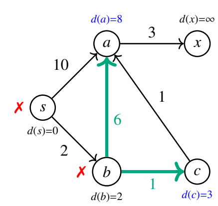
איור 3. גרף מכוון עם משקלים אי-שליליים עליו נבדוק את האלגוריתם הכללי.



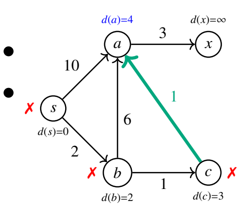
- בהתחלה $d(s) = 0$,
ועבור כל שאר הצמתים $d(a) = d(b) = d(c) = d(x) = \infty$.



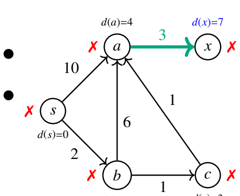
- הקשת $(c \to a)$ לא מפרה את אש"מ, כי $\infty = d(a) \le d(c) + w(c \to a) = \infty + 1 = \infty$.
- מאותה סיבה, הקשת $(b \to a)$ גם היא לא מפרה את אש"מ.
- למעשה, הקשתות היחידות שמפרות את אש"מ הן $(s \to b), (s \to a)$,
ניתן לעדכן $d(a) \leftarrow 10,\ d(b) \leftarrow 2$ (אינטואיטיבית, היינו רוצים לעדכן את הערך 2 לפני הערך 10).



- כעת יש 3 קשתות שמפרות את אש"ם: $\{(b \to c), (b \to a), (a \to x)\}$.
- אינטואיטיבית, נסתכל על הקשתות שיוצאות מ-b (כי המסלול $s \overset{2}{\rightsquigarrow} b$ הוא הכי קטן, ובהכרח נכון!)
כמו כן, נרצה לעדכן את הקשת $(b \to c)$ בגלל משקלה הנמוך.
- עדכון של c יהיה: $d(c) = d(b) + w(b \to c) = 2 + 1 = 3$.
עדכון של a יהיה: $d(a) = d(b) + w(b \to a) = 2 + 6 = 8$.



- נעבור שוב (אינטואיטיבית) לצומת c, שכן $(b \to c)$ היא מינימלית שיוצאת מ-b.
- ישנה הפרה של אש"מ מהצורה $8 = d(a) \nleq d(c) + w(c \to a) = 3 + 1 = 4$,
לכן נעדכן $d(a) \leftarrow d(c) + w(c \to a) = 4$.



- נעבור שוב (אינטואיטיבית) לצומת a, שכן $(c \to a)$ מינימלית (היחידה שיוצאת מ-c).
- ישנה הפרה של אש"מ מהצורה $\infty = d(x) \nleq d(a) + w(a \to x) = 4 + 3 = 7$,
לכן נעדכן $d(x) \leftarrow 7$.

- אין עוד קשתות שמפרות את אש"מ, לכן האלגוריתם הגיע לסיום.

ובעצם, מה האינטואיציה לזה שהמרחק $d(b) = 2$ הוא נכון?
התשובה היא: כי אין משקלים שליליים! לכן כל מסלול אחר בהכרח "יצבור" משקלים נוספים ויהיה לכל הפחות באותו גודל.

האינטואיציה הזאת מניבה לנו אלגוריתם עבור עצים מכוונים עם משקלים אי שליליים:

5.2. האלגוריתם של Dijkstra.

אלגוריתם (Dijkstra, 1959)**:**

- אתחול:

$$Q \leftarrow V \ , d(u) \leftarrow \begin{cases} 0 & u = s \\ \infty & u \neq s \end{cases}$$

כאשר Q מייצג את אוסף הצמתים שעדיין לא "טיפלנו" בקשתות היוצאות מהן.

- כל עוד $Q \neq \emptyset$:
  (א) יהי $u \in Q$ הצומת עם ערך $d(u)$ קטן ביותר.
  (ב) לכל קשת $(u \to v)$,
  אם $d(v) > d(u) + w_e$, אז: $d(v) \leftarrow d(u) + w_e$.
  (ג) הוצא את u מ- Q.

שאלה 3.9 מה זמן הריצה של האלגוריתם של Dijkstra?
אם נממש ב-heap למימוש Q, נקבל זמן ריצה של:

- אתחול הערימה: לינארי במספר הצמתים: $O(|V|)$.
- בכל איטרציה מוציאים צומת אחת u מ-Q, וממשיכים כל עוד Q לא ריק, סה"כ:

$$\underbrace{O(\log|V|)}_{\text{הוצאת } u \text{ מהערימה}} + \underbrace{O(d_{\text{out}}(u)\log|V|)}_{\text{במקרה הגרוע, נצטרך לבצע עדכון לכל הצמתים ש-}u\text{ נכנסת אליהן.}} + \underbrace{O(1)}_{\text{הוצאת הצומת עם } d \text{ מינימלי מהערימה}}$$

סה"כ, באיטרציה שבה u יוצא מ-Q, סיבוכיות $O((d_{\text{out}} + 1)\log|V|)$.

- סה"כ על פני כל האיטרציות: $O(|V| + |E|\log|V|) \equiv O(n + m\cdot\log n)$.

מסקנה 3.1 זמן הריצה של האלגוריתם הוא $O(|E|\log|V|) \equiv O(m\cdot\log n)$, כאשר $(m = |E|, n = |V|)$.

5.2.1. הוכחת נכונות.

טענה 3.3 (אבחנה: יחס סדר בין צמתים במובן של d בסיום איטרציית היציאה מ-Q)
נניח כי v יצא מ-Q באיטרציה עוקבת (המיידית אחרי) לזו ש-u יצא מ-Q. אזי:

$$\left(\substack{\text{בסיום האיטרציה בה}\\ v \text{ יצא מ-}Q}\right) \quad d(u) \leq d(v) \quad \left(\substack{\text{בסיום האיטרציה בה}\\ u \text{ יצא מ-}Q}\right)$$

הוכחה. בתחילת האיטרציה בה u יצא מ-Q: $d(u) \leq d(v)$ (בגלל אופן בחירת הצומת מ-Q בכל איטרציה):

- אם אין קשת $e = (u \to v) \in E$ שמפרה את אש"מ, אז $d(v)$ לא ישתנה במהלך האיטרציה ולכן הטענה נכונה.
- אם יש קשת $e = (u \to v) \in E$ והייתה הפרה של אש"מ במהלך האיטרציה ביצענו עדכון: $d(v) \leftarrow d(u) + w_e$
  אבל משקלי הקשתות אי-שליליים, ולכן גם בסיום איטרציה זו, עדיין מתקיים: $d(u) \leq d(v)$.

ולכן האבחנה נכונה. ■

טענה 3.4 (מסקנות מהאבחנה)

- האבחנה נכונה גם עבור שני צמתים שיצאו מ-Q באיטרציות שאינן עוקבות.
- לכל צומת u מתקיים ש-$d(u)$ לא מתעדכן אחרי ש-u יצא מ-Q.

הוכחת המסקנה. נניח בשלילה שהאבחנה השנייה לא נכונה.

נסתכל על הפעם הראשונה שבה $d(u)$ התעדכן באיטרציה מאוחרת מזו ש-u יצא מ-Q, ונסמן ב-v את הצומת שיצאה מ-Q באיטרציה זו.

באיטרציה בה v יצא מ-Q, התקיים עבור קשת $e = (v \to u)$:

$$\underbrace{d(u)}_{\substack{\text{שווה לערך } d(u) \\ \text{בסיום האיטרציה בה} \\ u \text{ יצא מ-} Q}} > \underbrace{d(v)}_{\substack{\text{שווה לערך } d(v) \\ \text{בסיום האיטרציה בה} \\ v \text{ יצא מ-} Q}} + w_e$$

בגלל ש-$w_e \geq 0$, קיבלנו סתירה לאבחנה (u יוצא מ-Q באיטרציה שקודמת לאיטרציה שבה v יצא מ-Q). ■

משפט 3.2 (נכונות האלגוריתם של Dijkstra)

בסיום ריצת האלגוריתם של Dijkstra, מתקיים שלכל קשת $e = (u \to v)$, מתקיים אי שוויון המשולש ($d(v) \leq d(u) + w_e$).
לכן האלגוריתם הוא מימוש מסוים של השיטה הכללית.

הוכחה.

- אם v יצא מ-Q אחרי u, באיטרציה בה u יצא מ-Q, בדקנו האם $d(v) > d(u) + w_e$,
  ולכן בכל מקרה, בסיום האיטרציה בה u יצא מ-Q, התקיים $d(v) \leq d(u) + w_e$.
  לפי מסקנה 2 של טענה 3.4, $d(u)$ לא ישתנה מרגע זה ועד סיום ריצת האלגוריתם של Dijkstra.
  $d(v)$ יכול רק לרדת, ולכן אי-השוויון מתקיים גם בסיום ריצת האלגוריתם.

- אם u יצא מ-Q אחרי v, לפי מסקנות 1 ו-2 של טענה 3.4, בסיום ריצת האלגוריתם יתקיים $d(v) \leq d(u)$,
  ולכן בגלל ש-$w_e \geq 0$, בסיום ריצת האלגוריתם יתקיים $d(v) \leq d(u) + w_e$.

■

הערה 3.3 האלגוריתם של Dijkstra דומה מאוד לאלגוריתם של Prim,
ונבדל בעיקר בכלל המשמש להתניית הכניסה לרכיב הפלט:

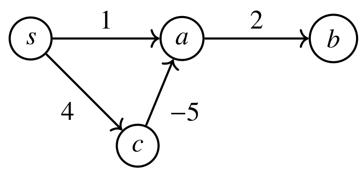
- ב-Prim - הקשת הקלה שחוצה את השפה של הרכיב.
- ב-Dijkstra (הגדרה שקולה למה שלמדנו) - הקשת שיוצאת מהרכיב u
  בעלת v עם $d(u) + w(u \to v)$ מינימלי.

הערה 3.4 גם האלגוריתם של Prim וגם האלגוריתם של Dijkstra הם דוגמאות לאלגוריתמים חמדנים. נפרט על אלגוריתמים חמדנים בהמשך.

שאלה 3.10 האם האלגוריתם של Dijkstra אכן נכשל אם בגרף יש קשתות שליליות, אבל אין מעגלים שליליים? - **כן!**

- אם מבצעים עדכון של התנאי לעדכון, סיבוכיות האלגוריתם כבר לא תהיה פולינומיאלית.
- אם משתמשים באלגוריתם כמו שהוא, אז לא ניתן להבטיח שיחזיר תשובה נכונה.

דוגמה 3.3 (האלגוריתם של Dijkstra נכשל בגרף עם קשתות שליליות)



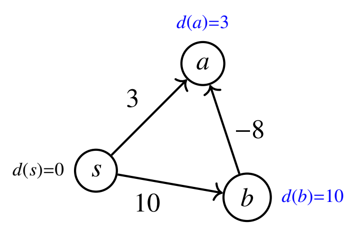
איור 4. גרף מכוון שמכיל משקלים שליליים אך לא מעגלים שליליים.

בתור תרגיל, הראו שבמקרה זה Dijkstra מחזיר תוצאה שגויה עבור הצומת b.

5.3. משקלים כלליים.

5.3.1. אינטואיציה לפעילות אלגוריתם נכון עבור משקלים כלליים.

עבור הגרף הבא, נבצע פאזה שבה נבדוק הפרות של אש"מ בכל רחבי הגרף. נקבל את העדכון הבא:



- נבחין שבמקרה זה, עבור b מופיעה התשובה הנכונה.
  אבחנה זו תהיה נכונה גם אם נוסיף מסלולים לא קלים יותר מהצורה $s \rightsquigarrow b$,
  כלומר בכל מצב שבו מבין המסלולים הקלים ביותר $s \rightsquigarrow b$, קיים אחד שמכיל רק קשת אחת.
- לאחר פאזה נוספת תתעדכן גם התשובה הנכונה עבור a.
- נבחין שבמסלול קל ביותר בגרף, יהיו לכל היותר $|V| - 1$ קשתות.

סיכום הרעיון:

- נבצע פאזות, ובכל פאזה נעבור על כל הקשתות של E בסדר שרירותי, ונבדוק הפרה של אי שוויון המשולש.
- נבצע $n - 1$ פאזות, ומכיוון שנוכיח שאחרי k פאזות מובטח שיש לנו את התשובה הנכונה עבור כל הצמתים u עבורם קיים מסלול קל ביותר $s \rightsquigarrow u$, המכיל לכל היותר k קשתות, נקבל את התשובה הנכונה לכל הצמתים בגרף.

5.4. האלגוריתם של Bellman-Ford.

אלגוריתם (Bellman-Ford):

- אתחול: $d(u) \leftarrow \begin{cases} 0 & u = s \\ \infty & u \neq s \end{cases}$, לכל $u \in V$.
- עבור $i = 1$ עד $n - 1$, בצע לכל קשת $e = (u \to v) \in E$:
  אם $d(v) > d(u) + w_e$, אז: $d(v) \leftarrow d(u) + w_e$.

שאלה 3.11 מה זמן הריצה של האלגוריתם? - $O(|E|\,|V|) = O(n \cdot m)$.

טענה 3.5 יהא G גרף חסר מעגלים שליליים.
אזי לכל צומת v, אם קיים מסלול קל ביותר $s \rightsquigarrow v$ המכיל k קשתות, אז בסיום הפאזה ה-k מתקיים $d(v) = \delta(s, v)$.

הוכחה. באינדוקציה על k.

- בסיס: $k = 0$, ואכן בכלל שאין מעגלים שליליים, s הוא הצומת היחיד עבורו יש מסלול קל ביותר מ-s המכיל אפס קשתות, ועבור s מתקיים אחרי האתחול כי $d(s) = 0 = \delta(s, s)$.
- צעד: יהי v צומת ו-P מסלול קל ביותר המכיל $k + 1$ קשתות.

$$P \;=\; \underset{s}{\overset{}{\underset{\parallel}{v_0}}} \to v_1 \longrightarrow v_2 \ldots \longrightarrow v_k \to \underset{v}{\underset{\parallel}{v_{k+1}}}$$

מכיוון ש-P קל ביותר, גם הרישא שלו $v_k \rightsquigarrow s$ היא מסלול קל ביותר מ-s ל-v_k,
ולכן, לפי הנחת האינדוקציה, בסיום הפאזה ה-k התקיים $d(v_k) = \delta(s, v_k)$.
מנכונות השיטה הכללית, אנחנו יודעים שבכל רגע של הריצה לא יכול להתקיים $d(v_k) < \delta(s, v_k)$.
לכן, גם במהלך הפאזה ה-$k + 1$, בהכרח מתקיים $d(v_k) = \delta(s, v_k)$.

במהלך הפאזה ה-$k + 1$, בהכרח בדקנו את הקשת $v_k \to \overset{v}{\overset{\parallel}{v_k}}$, ולכן בסיום הפאזה ה-$k + 1$ מתקיים:

$$d(v_{k+1}) \leq d(v_k) + w_{(v_k \to v_{k+1})} \underbrace{=}_{\substack{\text{לאחר הפאזה ה-}k\text{,} \\ \text{תמיד מתקיים:} \\ d(v_k)=\delta(s,v_k)}} \delta(s, v_k) + w_{v_k \to v_{k+1}} = \underbrace{=}_{\text{אורך המסלול } P} \delta(s, v_{k+1})$$

וקיבלנו שבסיום הפאזה ה-$k + 1$, $d(v_{k+1}) \leq \delta(s, v_{k+1})$.
כאמור, מהוכחת האלגוריתם הכללי לא יתכן שמתקיים $d(v_{k+1}) < \delta(s, v_{k+1})$, ולכן $d(v_{k+1}) = \delta(s, v_{k+1})$.

■

פרק 4

אלגוריתמים חמדניים

גישה לפתרון בעיות אלגוריתמיות, שבה האלגוריתם בוחר בכל צעד את האפשרות הטובה ביותר כרגע.

1. שיבוץ משימות על מכונה אחת

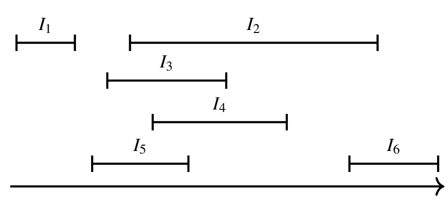
נתונות n משימות, כך שכל משימה i מיוצגת ע"י זמן התחלה s_i וזמן סיום f_i.

יש מכונה בודדת, שיכולה בכל רגע נתון לבצע לכל היותר משימה אחת, ונניח שרשימת הבקשות למשימות, כולל זמני ההתחלה והסיום, ידועה מראש.

מטרה: מה המספר הכי גדול של משימות שניתן לבצע?

1.1. תיאור הבעיה ע"י אינטרוולים.

דוגמה 4.1 נתבונן בדוגמה הבאה, כאשר נניח כי הקצה השמאלי מסמן s_i והימני f_i:



איור 1. בעיית שיבוץ משימות בייצוג של אינטרוולים

פתרון אופטימלי גודלו 3:

$\{I_2, I_5, I_6\}$

$\{I_2, I_3, I_6\}$

$\{I_2, I_4, I_6\}$

תיאור אלטרנטיבי של הבעיה: רוצים לבחור תת-קבוצה גדולה ביותר של אינטרוולים כך שכל שניים לא נחתכים.

שאלה 4.1 מה המשמעות של הגישה החמדנית עבור בעיית השיבוץ שלנו?

נחליט על איזשהו סדר על האינטרוולים, ובאופן "חמדני" נעבור על האינטרוולים לפי סדר זה, ונוסיף לפתרון אינטרוול אם הוא לא נחתך עם האינטרוולים שנבחרו עד עכשיו.

1.2. הסדר החמדן.

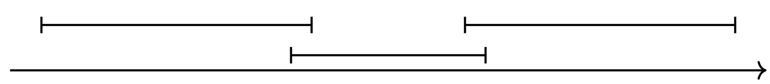
שאלה 4.2 מהו הסדר החמדן בו נעבור על האינטרוולים? מספר אפשרויות:

(1) ? לפי זמן סיום מהמוקדם למאוחר (או בסדר הפוך).
(2) ✗ לפי אורך האינטרוול, מהקצר לארוך.
(3) ✗ לכל אינטרוול נספור עם כמה אינטרוולים אחרים הוא נחתך, ונעבור מהמספר הקטן לגדול.
(4) ✗ לפי זמן התחלה מהמוקדם למאוחר (או בסדר הפוך).

נשים לב שחלק מהסדרים הללו לא יחזירו את התשובה הנכונה.
הנה מספר דוגמאות נגדיות לחלק מההצעות לעיל לסדרים חמדניים:

1.3. סדר חמדן לא נכון עלול להוביל לתוצאה שגויה.

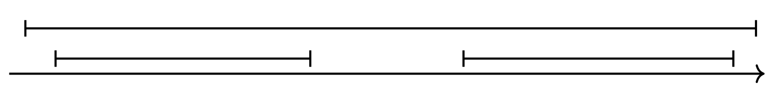
עבור הצעה מספר (2):



איור 2. דוגמה נגדית להצעה מספר (2).

הפתרון האופטימלי הוא 2, אך לפי סדר (2) יוחזר 1.

עבור הצעה מספר (4):



איור 3. דוגמה נגדית להצעה מספר (4).

הפתרון האופטימלי הוא 2, אך לפי סדר (4) יוחזר 1.

תרגיל: הוכיחו שגם הצעה (3) איננה נכונה.

ואמנם, הסדר החמדן (1) תמיד נותן פתרון אופטימלי (נראה זאת בהמשך).

1.4. האלגוריתם החמדן, הוכחת נכונות.

אלגוריתם:

(1) מיין את האינטרוולים לפי זמן סיום לא יורד ($f_1 \leq f_2 \leq f_3 \leq \ldots \leq f_n$).
(2) הגדר $X \leftarrow \emptyset$.
(3) עבור $1 \leq j \leq n$:
אם I_j לא נחתך עם אף אינטרוול ב-X, בצע $X \leftarrow X \cup \left\{I_j\right\}$.
(4) הפלט זה X.

סיבוכיות זמן ריצה: $O\left(n \log n\right)$.

- מיון האינטרוולים לפי זמני הסיום: מתבצע ב-$O\left(n \log n\right)$.
- אתחול X: מתבצע ב-$O\left(1\right)$.
- מעבר על האינטרוולים ובניית X: מתבצע ב-$O\left(n\right)$.

(הסבר: כדי לבדוק האם אינטרוול נחתך עם X, מספיק לבדוק האם נחתך עם האינטרוול המסתיים מאוחר ביותר ב-X. באמצעות מידע נוסף זה, ניתן לממש את המעבר על האינטרוולים בזמן לינארי).

משפט 4.1 (הזדהות של האלגוריתם החמדן עם אופטימום כלשהו בכל איטרציה)
בסיום איטרציה k, קיים פתרון אופטימלי X^* כך שמתקיים:

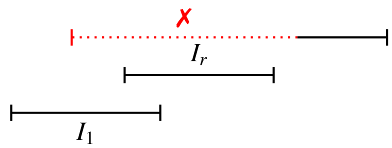
$$I_j \in X^* \iff I_J \in X$$

מסקנה 4.1 בסיום האיטרציה ה-n, קיבלנו ש-X שווה לאיזשהו פתרון אופטימלי X^*.

הוכחת המשפט. באינדוקציה על k.

- בסיס: $k = 1$.
בסיום האיטרציה הראשונה, $x = \{I_1\}$.
יהי X^* איזשהו פתרון אופטימלי לבעיה.
  - אם $I_1 \in X^*$, סיימנו.
  - אחרת $I_1 \notin X^*$, ויהי $I_r \in X^*$ האינטרוול בעל זמן הסיום המוקדם ביותר.
נסתכל על $X^* \setminus \{I_r\} \cup \{I_1\}$, ונטען שאוסף אינטרוולים זה הוא פיזיבילי (חוקי),
ומכיוון שגודלו זהה לגודל X^* הוא גם אופטימלי.
מספיק שנראה ש-I_1 לא נחתך עם אף אינטרוול ב-$X^* \setminus \{I_r\}$.

כל אינטרוול ב-$X^* \setminus \{I_r\}$ מתחיל אחרי סיום I_r, ו-I_r מסתיים לא אחרי זמן הסיום של I_r,
ולכן I_1 לא נחתך עם אף אינטרוול ב-$X^* \setminus \{I_r\}$.



איור 4. כל אינטרוול שאינו I_r ב-X^* בהכרח לא נחתך עם I_r, לכן גם לא יחתך עם I_1.

- <u>צעד האינדוקציה:</u> נניח נכונות עד סיום האיטרציה ה-k,
ויהי X^* פתרון אופטימלי שקיומו מובטח מהנחת האינדוקציה, כלומר: $I_j \in X,\ \forall j = 1,\dots,k \iff I_j \in X^*$.
ישנם שני מקרים:
  (1) האלגוריתם בחר את I_{k+1}, כלומר: $I_{k+1} \in X$.
    - אם $I_{k+1} \in X^*$, סיימנו את הצעד.

    - אחרת, $I_{k+1} \notin X^*$, ואז נבחר $I_r \in X^*$ עם $r > k+1$ בעל זמן סיום קטן ביותר
(קיום I_r שכזה מובטח כי אחרת $|X| > |X^*|$ בסיום האיטרציה ה-$k+1$, וזוהי סתירה לאופטימליות X^*).

    נסתכל על: $X^* \setminus \{I_r\} \cup \{I_{k+1}\}$.
    כל מה שנשאר להראות זה ש-$X^* \setminus \{I_r\} \cup I_{k+1}$ פתרון פיזיבילי:
    נראה ש-I_{k+1} לא נחתך עם אף אינטרוול ב-$X^* \setminus \{I_r\}$.

    <u>I_{k+1} לא יכול להיחתך עם אינטרוולים עם אינדקס $\{1,\dots,k\}$ ב-$X^* \setminus \{I_r\}$.</u>
    זאת משום ש-I_{k+1} נבחר ע"י האלגוריתם, לכן לכל $1 \le j \le k$, כך ש-$I_j \in X$,
    I_{k+1} לא נחתך אתו. מהנחת האינדוקציה $I_j \in X^*$, לכן הטענה נובעת.

    <u>כמו כן, I_{k+1} לא יכול להיחתך עם אינטרוולים עם אינדקס $\{k+1,\dots,n\}$ ב-$X^* \setminus \{I_r\}$:</u>
    באופן דומה לבסיס האינדוקציה (איור 4), לכל $I_j \in X^* \setminus \{I_r\}$:

$$s_{k+1} \le f_{k+1} \le f_{k+2} \le \dots \le f_r \underbrace{<}_{\substack{\text{אינטרוולים ב-}X^* \setminus \{I_r\} \\ \text{עם אינדקס } r+1 \text{ ומעלה} \\ \text{לא נחתכים עם } I_r}} s_j \le f_j$$

    שהרי מהמינימליות של I_r במובן של זמן סיום קטן ביותר כך שאינו I_{k+1},
    ב- $X^* \setminus \{I_r\}$ אין אינטרוולים עם אינדקס $k+1, k+2, k+3, \dots, r$.

    ולכן, $X^* \setminus \{I_r\} \cup \{I_{k+1}\}$ הוא פתרון פיזיבילי ואופטימלי שמקיים:

$$I_j \in X \iff I_j \in X^* \setminus \{I_r\} \cup \{I_{k+1}\},\ \ \forall j = 1,\dots,k+1$$

  (2) האלגוריתם לא בחר את I_{k+1}, כלומר $I_{k+1} \notin X$.
  נטען גם ש-X^* שקיומו מובטח מהנחת האינדוקציה מקיים את מה שצריך: $I_{k+1} \notin X^*$.
  האלגוריתם לא בחר את I_{k+1} כי הוא נחתך עם אינטרוולים ב-X,
  ולכן I_{k+1}, לפי הנחת האינדוקציה, נחתך גם עם אינטרוולים ב-X^*, ולכן $I_{k+1} \notin X^*$.

■

שאלה 4.3 (פתרון חמדני בתוספת משקלים אי שליליים)
נניח שלאינטרוול I_j יש רווח $P_j \geq 0$.
כיצד נמצא אוסף אינטרוולים כך שכל שניים באוסף לא נחתכים, שממקסם את סך הרווחים?

לפי רועי, קשה (עד בלתי אפשרי) למצוא פתרון חמדני שיגיע לתשובה הנכונה במקרה זה (משקלים אי-שליליים).
נתייחס לבעיה זו בהמשך הקורס.

בכל אופן, נשים לב ששינוי קטן מאוד בניסוח הבעיה עלול להערים קשיים רבים על הגישה החמדנית.

2. קידודים

המטרה היא לקודד קובץ המורכב מתווים (למשל, בשפה האנגלית) בעזרת $\{0,1\}$, כך שאורך הקובץ המקודד יהיה כמה שיותר קטן.

שאלה 4.4 כיצד מקודדים? - כל תו נתון יועתק למילה מעל הא"ב $\{0,1\}$.

הגדרה 4.1 (מילת קוד) מעל א"ב $\{0,1\}$, מילת קוד היא רצף של אפסים ואחדים:

$$w = a_1 a_2 \dots a_\ell \qquad a_i \in \{0,1\} \;\; \forall i = 1,\dots,\ell$$

הגדרה 4.2 (אורך של מילת קוד) נסמן ב-$\ell(w)$ את האורך של מילת הקוד w.

הגדרה 4.3 (קוד) אוסף של מילות קוד שונות יקרא קוד.

הערה 4.1 (סימון) עבור קוד $C = \{c_1, c_2, c_3\}$ ותווים $\{x_1, x_2, x_3\}$,
נסמן את כלל ההתאמה בין כל תו למילת קוד באופן הבא: $C = \left\{\overset{x_1}{c_1}, \overset{x_2}{c_2}, \overset{x_3}{c_3}\right\}$.

דוגמה 4.2 נתון הקוד: $C = \left\{\overset{x_1}{c_1}, \overset{x_2}{c_2}, \overset{x_3}{c_3}\right\}$, ע"י: $c_1 = 00,\; c_2 = 01,\; c_3 = 011$
הקידוד של שרשור שלושת התווים $x_1 x_2 x_3$ יתקבל בתור:

$$\underbrace{00}_{c_1}\underbrace{01}_{c_2}\underbrace{001}_{c_3}$$

הגדרה 4.4 (קוד חד-פענח) קוד יקרא חד-פענח, אם הקידוד של כל רצף תווים ניתן לפענוח באופן יחיד.

דוגמה 4.3 (דוגמה לקוד לא חד-פענח) נתבונן בקוד $C = \left\{\overset{x_1}{c_1}, \overset{x_2}{c_2}, \overset{x_3}{c_3}\right\}$, עם: $c_1 = 01,\; c_2 = 0,\; c_3 = 10$.
ניתן לפענח את הקידוד 010 בשני אופנים:

(1) $c_2 c_3$, מתאים ל-$x_2 x_3$.

(2) $c_1 c_2$, מתאים ל-$x_1 x_2$.

נשים לב שבקוד זה, מילת הקוד $c_2 = 0$ הינה רישא של מילת הקוד $c_1 = 01$.

הגדרה 4.5 (קוד חסר רישאות) קוד יקרא חסר רישאות אם אין בו מילת קוד שהיא רישא של מילת קוד אחרת.

דוגמה 4.4 הקוד מדוגמה 4.2 הוא לא חסר רישאות.

אנחנו נתמקד בקודים מסוג חד-פענח, שהם חסרי רישאות.

2.1. פיענוח של קידודים בעזרת קודים חסרי רישאות.

שאלה 4.5 כיצד מפענים קידוד בעזרת קוד חסר רישאות?

נשים לב שהיות שהקוד הוא חסר רישאות, ישנה דרך יחידה לפענח כל מילת קוד שמופיעה בקידוד.

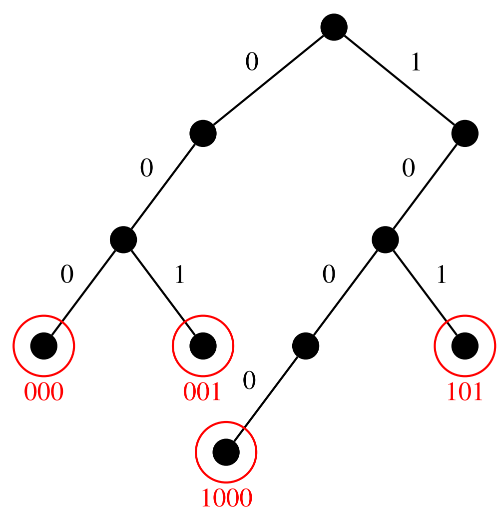
סורקים את הקידוד, וברגע שמזהים מילת קוד, מפענחים אותה וממשיכים.

הערה 4.2 (קוד חסר רישאות כעץ בינארי) קוד חסר רישאות ניתן לייצוג בעזרת עץ בינארי.

כל אחד מהילדים הישירים של צומת פנימי יותאם ל-0 או 1, ומילות הקוד יהיו העלים בעץ.

דוגמה 4.5 (עץ בינארי שמתאר קוד חסר רישאות)

$$\mathbf{C = \{000, 001, 1000, 101\}}$$



איור 5. עץ בינארי של קוד לדוגמה.

2.2. תיאור הבעיה.

- נתון: נתונים n תווים $x_1,\dots,x_n$, ולכל תו x_i מספר מופעים f_i.
- מטרה: למצוא קוד חד-פענח עם n מילות קוד $C=\{c_1,c_2,\dots,c_n\}$ שממזער את הביטוי $\sum_{i=1}^{n} f_i\cdot\ell(c_i)$ (כלומר, ממזער את אורך הקידוד).

טענה 4.1 (ללא הוכחה) מבין כל הפתרונות האופטימליים, קיים לפחות אחד שהוא קוד חסר רישאות.

מסקנה 4.2 (בקוד חסר רישאות שהוא פתרון אופטימלי, אורך מילת קוד הוא עומק העלה בעץ)
רוצים לחפש עץ בינארי T עם n עלים שממזער את $\sum_{i=1}^{n} f_i\cdot \underbrace{d_T(c_i)}_{\text{עומק העלה ה-}i\text{ בעץ }T}$.

טענה 4.2 (אבחנה) עץ בינארי המייצג קוד אופטימלי הוא שלם (לכל צומת פנימי שאינו עלה יש שני ילדים ישירים).

הוכחה. נניח בשלילה שיש בעץ צומת פנימי לו ילד ישיר בודד, אז נמחק צומת זה ונחבר את הילד הישיר שלו להורה של הצומת. קיבלנו עץ בינארי חדש שאורך הקידוד שהוא מגדיר לא יותר ארוך מקידוד העץ המקורי. ■

באופן "איכותי", נרצה שלתווים נפוצים יתאים קידוד קצר ולתווים נדירים קידוד ארוך.

2.3. האלגוריתם של Huffman.

אלגוריתם (של Huffman):

- נתון: אוסף משקלים ממוינים $f_1\ge f_2\ge\dots\ge f_k$.
- תנאי עצירה: $k=1$, ואז נחזיר עץ שהוא צומת בודד ללא קשתות.
- רקורסיה:
  - נאחד את התווים ה-k וה-$k-1$ לתו חדש שמשקלו $f'=f_{k-1}+f_k$.
  - נפעיל את האלגוריתם רקורסיבית על $k-1$ המשקלים החדשים (המשקלים $f_1,\dots,f_{k-2}$ ו-f'), ומקבלים עץ T.
  - ניקח את T, ולעלה שמייצג את איחוד התווים ה-k וה-$k-1$ נוסיף שני ילדים ישירים: אחד מהם מייצג את התו ה-k והשני את התו ה-$k-1$.
  - מחזירים את העץ משלב (3).

דוגמה 4.6 (דוגמת הרצה) עבור המשקלים והתווים הבאים, נציג ריצה של האלגוריתם:

משקל	תו	ערך
f_1	x_1	5
f_2	x_2	4
f_3	x_3	3
f_4	x_4	2
f_5	x_5	1

ראשית, נציג את הצעדים הרקורסיביים:

(1)

משקל	תו	ערך
f_1	x_1	5
f_2	x_2	4
f_3	x_3	3
f_4	x_4	2
f_5	x_5	1

$\Longleftarrow$ (2)

משקל	תו	ערך
f_1	x_1	5
f_2	x_2	4
f_3	x_3	3
f'	$\{x_4, x_5\}$	3

$\Longleftarrow$ (3)

משקל	תו	ערך
f_1	x_1	5
f_2	x_2	4
f''	$\{x_3, x_4, x_5\}$	6

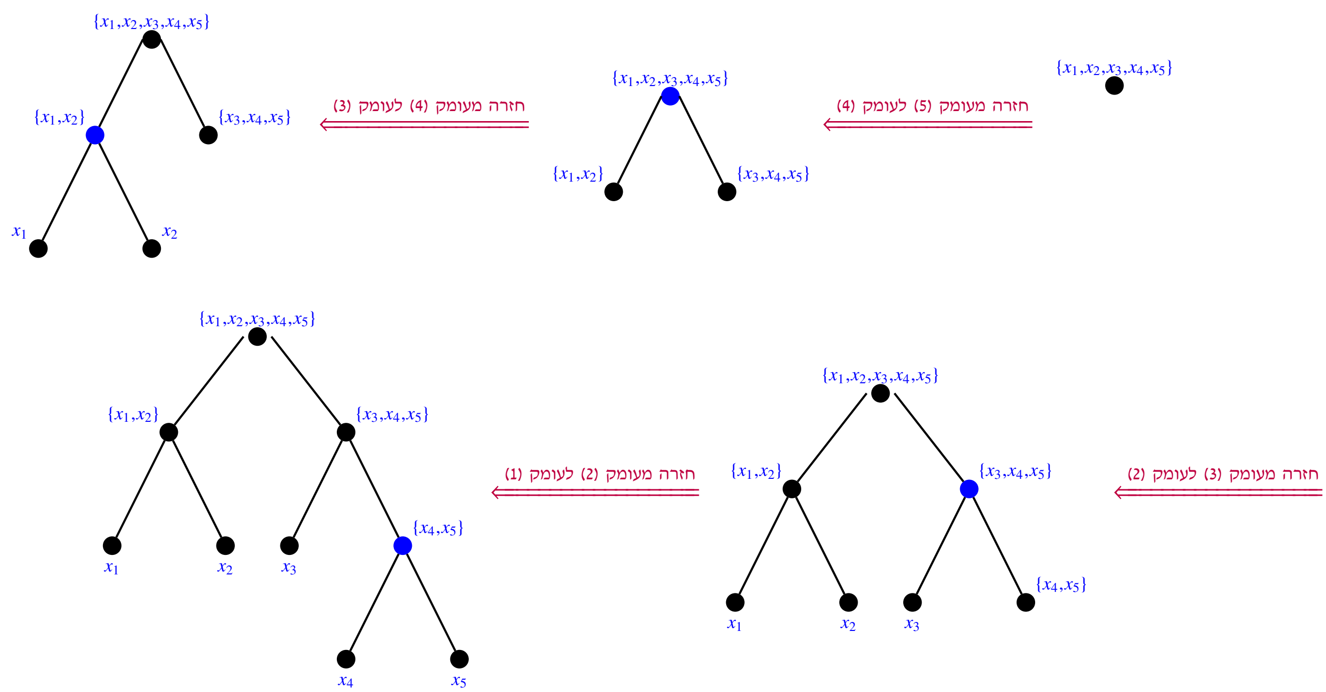
$\Longleftarrow$ (4)

משקל	תו	ערך
f'''	$\{x_1, x_2\}$	9
f''	$\{x_3, x_4, x_5\}$	6

$\Longleftarrow$ (5)

משקל	תו	ערך
f''''	$\{x_1, x_2, x_3, x_4, x_5\}$	15

כעת, נציג את החזרות מהרקורסיה (פתיחות של תווים):



2.3.1. זמן הריצה של האלגוריתם של Hoffman.

- מיון ראשוני: $O(n \log n)$.
- כל צעד רקורסיבי: $O(\log n)$ (למשל ע"י חיפוש בינארי).
- כל חזרה מרקורסיה: $O(1)$ (ע"י כך שנזכור מי שני התווים האחרונים שאיחדנו).

$\Longleftarrow$ סה"כ: $O(n \log n)$.

טענה 4.3 בהינתן משקלים ל-n תווים: $f_1 \geq f_2 \geq \ldots \geq f_n$, קיים עץ אופטימלי T בו התו ה-n והתו ה-n-1 הם עלים אחים עמוקים ביותר ב-T.

<u>הוכחה</u>. נניח בשלילה שאין עץ T שכזה, ויהי T^* עץ אופטימלי.
האבחנה גוררת שב-T^* יש שני עלים אחים עמוקים ביותר.

לפחות אחד משני התווים ה-n וה-n-1 אינו מיוצג ע"י שני עלים אלו.
ניקח את העלה שמייצג את התו שחסר מבין ה-n וה-n-1, ונחליף אותו עם העלה מבין השניים האחים העמוקים ביותר שאינו ה-n וה-n-1.

נבחין כי בהכרח אורך הקידוד של העץ החדש יכול רק לקטון, כי משקל כל עלה שאינו ה-n וה-n-1 הוא לפחות $f_{n-1} \geq f_n$.

- אם הערך (אורך הקידוד) קטן, זו סתירה לאופטימליות של T^*.
- אחרת, אם התו ה-n וה-n-1 בעץ החדש שקיבלנו אחים (עמוקים ביותר), קיבלנו סתירה וסיימנו.
- אם לא, נבצע שוב החלפה שכזו עבור התו השני שחסר מבין ה-n וה-n-1.

■

משפט 4.2 יהיו n תווים עם משקלים $f_1 \geq f_2 \geq \ldots \geq f_n$,
ויהי T' עץ אופטימלי עבור הבעיה המצומצמת עם n-1 תווים שמשקליהם: $f_1, f_2, \ldots, f_{n-2}$ ו-$f' = f_{n-1} + f_n$.

יהי T העץ עבור n התווים המתקבל מ-T' באופן הבא:

(1) לוקחים את T'.

(2) לעלה שמשקלו f', מוסיפים שני ילדים ישירים שמייצגים את התווים ה-n וה-n-1.

אזי T עץ אופטימלי עבור המשקלים: $f_1, \ldots, f_n$.

<u>הוכחה</u>. נסמן עבור T את אורך הקידוד שהוא משרה ע"י $\operatorname{cost}(T)$.
נניח בשלילה שקיים עץ T'' עבור n התווים כך שמתקיים $\operatorname{cost}(T'') \leq \operatorname{cost}(T)$.

בה"כ, לפי הטענה, התווים ה-n וה-n-1 הם אחים עמוקים ביותר ב-T''.
מתקיים:

$$\operatorname{cost}(T) = \operatorname{cost}(T') + f_{n-1} + f_n$$

ניצור בעזרת T'' עץ חדש T''' עבור n-1 התווים (התווים 1 עד n-2 ותו חדש שמייצג את האיחוד של התווים ה-n וה-n-1),
ע"י זה שנמחק את שני העלים האחים (עמוקים ביותר) ב-T'' שמייצגים את התווים ה-n וה-n-1,
ונכריז על ההורה הישיר שלהם כעלה שמייצג את התו שהוא האיחוד שלהם:

$$\underbrace{\operatorname{cost}(T''')}_{\text{עבור } n\text{-1 התווים}} = \operatorname{cost}(T'') - f_{n-1} - f_n \underbrace{<}_{\text{הנחת השלילה}} \operatorname{cost}(T) - f_{n-1} - f_n = \operatorname{cost}(T')$$

וזאת סתירה לאופטימליות של T'.

■

פרק 5

תכנון דינאמי

הגדרה 5.1 (שיטת התכנון הדינאמי) היא פרדיגמה שימושית לתכנון אלגוריתמים.
שיטה זו מתאימה לבעיות אופטימיזציה אותן ניתן לבטא באמצעות בעיות קטנות יותר מאותו הסוג (כלומר, עבור בעיות שיש להן מבנה רקורסיבי).

דבר המתאפשר כחלק מטכניקה זו הוא לשמור בזכרון תשובות לקראיות רקורסיביות, ובכך לחסוך קריאות חוזרות ולהביא לשיפור בסיבוכיות.

1. חישוב יעיל של כפל מטריצות

1.1. תיאור הבעיה, כשלון הפתרון הנאיבי.

- נתון: $A_1 \cdot A_2 \cdot \ldots \cdot A_n$, כאשר A_i זו מטריצה במימדים $n_i \times n_{i+1}$.
- מטרה: למצוא דרך לחישוב המכפלה, שממזערת את מספר הכפלים הסקלאריים שמתבצעים.

אם כופלים $A \cdot B$ ו-A במימדים $p \times q$ ו-B במימדים $q \times r$, מספרים הכפלים הסקלאריים שמתבצעים הינו $p \cdot q \cdot r$.

דוגמה 5.1 $A_1 \cdot A_2 \cdot A_3$, והמימדים $A_1 = 10 \times 100,\ A_2 = 100 \times 5,\ A_3 = 5 \times 50$

מספר המכפלות יהיה אחד משניים:

- $(A_1 \cdot A_2) \cdot A_3$: $10 \cdot 100 \cdot 5 + 10 \cdot 5 \cdot 50 = 7500$
- $A_1 \cdot (A_2 \cdot A_3)$: $100 \cdot 5 \cdot 50 + 10 \cdot 100 \cdot 50 = 75000$

שאלה 5.1 מדוע לא ניתן לעבור על כל האפשרויות?
נקבל שמספר האפשרויות לביצוע המכפלה עם n מטריצות הינו:

$$p(n) = \begin{cases} 1 & n = 1 \\ \sum_{k=1}^{n-1} p(k)\, p(n-k) & n > 1 \end{cases}$$

הפתרון הוא מספרי קטלן ($p(n) = C(n-1)$), וניזכר שמתקיים:

$$C(n) = \frac{1}{n+1}\binom{2n}{n} = \Omega\left(\frac{4^n}{n^{\frac{3}{2}}}\right)$$

$\Longleftarrow$ לא נרצה לבצע חיפוש ממצה על פני כל האפשרויות לחישוב המכפלה באורך n
(מספר האפשרויות הוא אקספוננציאלי באורך n)!

באופן אינטואיטיבי, נרצה לייצג את הבעיה באופן רקורסיבי.

1.2. יצוג הבעיה באופן רקורסיבי ע"י תתי בעיות.

1.2.1. הגזרת תת-הבעיה. לכל תת-תרגיל שמתחיל ב-A_i ומסתיים ב- A_j ($i \le j$), נגדיר תת-בעיה.

הגדרה 5.2 (תת-בעיה) $M(i,j)$ של תת-תרגיל A_k ($\prod_{k=i}^{j}$) תוגדר להיות מספר הכפלים הסקלאריים הקטן ביותר שצריך בשביל לפתור את תת-התרגיל $A_i \cdot A_{i+1} \cdot \ldots \cdot A_j$.

הערה 5.1 נשים לב שחישוב הערך $M(1,n)$ יוביל למעשה לפתרון הבעיה הראשית.

1.2.2. חישוב תת-תרגיל ככפל בין שני תתי-תרגילים. כשרוצים לחשב את תת-התרגיל: $A_i \cdot A_{i+1} \cdot \ldots \cdot A_j$, ניתן ליצור חלוקה של כל האפשרויות השונות לחישוב ע"י פירוק למכפלה בין שני תתי-תרגילים, כפונקציה של המיקום שבו מתבצע הכפל האחרון. אם הכפל האחרון הוא בין A_k ל-A_{k+1}, אז לפנינו:

$$(A_i \cdot \ldots A_k) \cdot \left(A_{k+1} \ldots A_j\right)$$

אינטואיטיבית, נבחין שמתקיים הקשר:

$$\underbrace{M(i,j)}_{\text{מספר הכפלים הסקלאריים המינימלי הדרוש לפתרון תת-התרגי}} = \underbrace{\min_{k=i,\ldots,j\text{-}1}}_{\text{מאחר שזו חלוקה של כל האפשרויות ניקח את זו שמניבה מספר כפלים סקלאריים מינימלי}} \left\{ \underbrace{M(i,k) + M(k+1,j)}_{\text{מספר הכפלים המינימלי לחישוב כל אחד מתתי התרגילים בנפרד}} + \underbrace{n_i \cdot n_{k+1} \cdot n_{j+1}}_{\text{עלות המכפלה בין תוצאות תתי-התרגילים}} \right\}$$

הגדרה 5.3 (הסדרה הרקורסיבית $A(i,j)$) נגדיר את הסדרה הרקורסיבית הבאה $A(i,j)$ לכל $1 \le i \le j \le n$:

$$A(i,j) \triangleq \begin{cases} 0 & i = j \text{ (תנאי העצירה)} \\ \min\limits_{k=i,\ldots,j-1} \left\{ A(i,k) + A(k+1,j) + n_i \cdot n_{k+1} \cdot n_{j+1} \right\} & i < j \text{ (קשר רקורסיבי)} \end{cases}$$

<u>טענה</u>: לכל $1 \le i \le j \le n$, $M(i,j) = A(i,j)$

טענה 5.1 (שקילות) $\forall 1 \le i \le j \le n, \quad \boxed{M(i,j) = A(i,j)}$

<u>הוכחה</u>. תרגיל (באינדוקציה דו מימדית) ■

הערה 5.2 במקום לחשב את $M(1,n)$, מספיק שנחשב את $A(1,n)$.

1.3. חישוב איטרטיבי של תת-הבעיה הרקורסיבית ע"י שימוש בזכרון.

שאלה 5.2 כיצד ניתן לייתר את הצורך ברקורסיות מיותרות ולייעל את הסיבוכיות?

במקום להשתמש בקשר הרקורסיבי באופן ישיר (דבר שעלול לגרום לכך שנחשב $A(i_0, j_0)$, כלשהו הרבה פעמים), נזכור תוצאות של תתי בעיות כך שכל תת בעיה נצטרך לחשב רק פעם אחת.

$$A = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ \\ n \end{array} \begin{array}{c} \begin{array}{ccccc} 1 & 2 & 3 & & n \end{array} \\ \begin{pmatrix} 0 & & & & \blacksquare \\ & 0 & & & \\ & & 0 & & \\ & & & \ddots & \\ & & & & 0 \end{pmatrix} \end{array} \longleftarrow \text{התוצאה שמעוניינים בה}$$

- נשים לב שלפי הגדרת A, תא $A(i, j)$ תלוי בתאים שמשמאלו ומתחתיו בלבד.
- כמו כן, כל ערכי האלכסון הם אפסים (לפי הגדרת A).
- נרצה למצוא סדר חישוב כך שברגע שרוצים לחשב את $A(i, j)$, כל ערכי A ש-$A(i, j)$ תלויים בו כבר חושבו.

שאלה 5.3 אילו סדרים טובים?

- אלכסונים החל מהאלכסון הראשי כלפי מעלה (בתוך כל אלכסון הסדר לא משנה).
- עמודות משמאל לימין, וכל עמודה מלמטה (מהאלכסון) כלפי מעלה.
- שורות מלמטה למעלה, וכל שורה משמאל (מהאלכסון) ימינה.

שאלה 5.4 מה זמן הריצה של האלגוריתם?

חישוב $A(i, j)$ יקח זמן של $O(j - i)$, ולכן סה"כ נקבל זמן ריצה של $O\left(n^3\right)$:

$$\sum_{i=1}^{n}\sum_{j=1}^{n}(j-i) = \sum_{i=1}^{n}\left(\frac{n^2+n}{2}\right) - \sum_{i=1}^{n} i = O\left(n^3\right) - O\left(n^2\right) = O\left(n^3\right)$$

שאלה 5.5 כמה זכרון מצריך האלגוריתם? - $O\left(n^2\right)$.

2. All Pairs Shortest Path

2.1. הגדרת הבעיה.

- נתון: $G=(V,E)$ גרף מכוון, $w:E\to\mathbb{R}$ (מניחים שאין מעגלים שליליים).
- מטרה: לכל $s,t\in V$, רוצים לחשב את $\delta(s,t)$, כלומר אורך מסלול קל ביותר ב-G מ-s ל-t.
- פתרונות אפשריים (שראינו):
  - להריץ Bellman-Ford מכל צומת.
  - להריץ Johnson.

שאלה 5.6 האם ניתן להשתמש בגישה של תכנון דינאמי בשביל לפתור את הבעיה?

שאלה 5.7 מה יהיו תתי-הבעיות שיעניינו אותנו?

הגדרה 5.4 (צמתי ביניים של מסלול $v_i\rightsquigarrow v_j$) נניח שיש לנו סדר שרירותי על הצמתים: $V=\{v_1,v_2,\ldots,v_n\}$.
נגדיר צמתי ביניים של מסלול v_i ו-v_j להיות אוסף הצמתים במסלול פרט לצומת הראשון (v_i) ולצומת האחרון (v_j).

2.2. הגדרת תתי-בעיות.

2.2.1. הרעיון. נסתכל על מסלולים קלים ביותר $v_i\rightsquigarrow v_j$ המוגבלים לקבוצות צמתי ביניים מסוימות.
אלו יהיו תתי הבעיות שבהן נשתמש.

הגדרה 5.5 (תת בעיה עם אילוץ $0\le k\le n$ על קבוצת ביניים) תת-בעיה תוגדר לכל זוג (סדור) של צמתים v_i ו-v_j,
ולכל $k=0,\ldots,n$, שמציין שצמתי הביניים לקוחים מתוך תת הקבוצה $\begin{cases}\{v_k\}_{k=1}^{n} & 1\le k\le n\\ \emptyset & k=0\end{cases}$.

הערה 5.3 (אבחנה) נשים לב שתחת המגבלה שקבוצת צמתי הביניים האפשרית היא $\emptyset$,
יודעים מה המסלול הקל ביותר מ-v_i ל-v_j:

- אם $i=j$, מסלול זה ערכו אפס.
- אחרת, אם יש קשת $\left(v_i\to v_j\in E\right)$, אז ערך מסלול זה שווה למשקל הקשת w_{ij}.
- אחרת, אין מסלול מ-v_i ל-v_j תחת מגבלה זו על צמתי הביניים.

הערה 5.4 נשים לב שמספר תתי הבעיות הוא $O\left(n^3\right)$.

נגדיר את הגודל הבא לכל תת-בעיה:

הגדרה 5.6 (משקל $v_i\rightsquigarrow v_j$ קל ביותר תחת האילוץ k על צמתי ביניים)
$\delta_k(i,j)$ יוגדר להיות משקל מסלול קל ביותר ב-G מ-v_i ל-v_j, תחת האילוץ שצמתי הביניים לקוחים מתוך $\begin{cases}\{v_k\}_{k=1}^{n} & 1\le k\le n\\ \emptyset & k=0\end{cases}$.

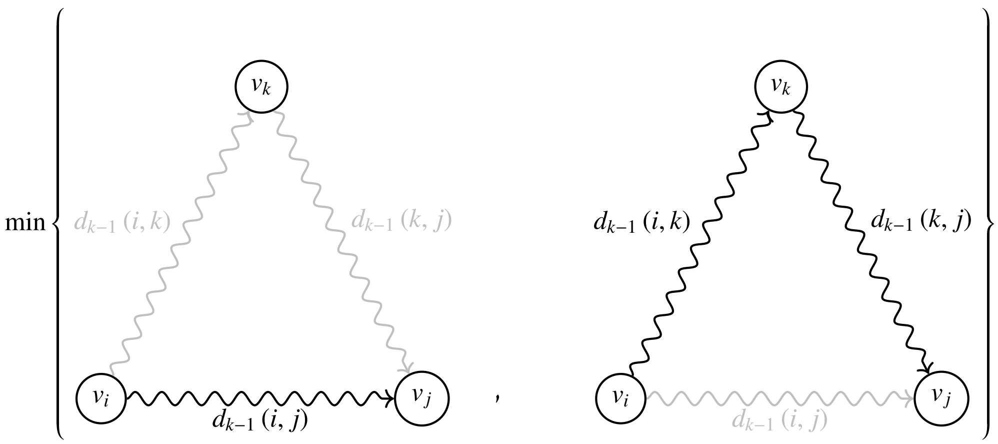
נגדיר סדרה רב-מימדית (רקורסיבית) באופן הבא:

- תנאי העצירה:

$$d_0(i,j) \triangleq \begin{cases} 0 & i=j \\ w_{ij} & i \neq j, \left(v_i \rightarrow v_j\right) \in E \\ \infty & i \neq j, \left(v_i \rightarrow v_j\right) \notin E \end{cases}$$

- הקשר הרקורסיבי:

$$d_k(i,j) = \min\left\{d_{k-1}(i,j), d_{k-1}(i,k) + d_{k-1}(k,j)\right\}$$



איור 1. הקשר הרקורסיבי בסדרה הרב מימדית $\{d_k(i,j)\}_{k,i,j}$.

טענה 5.2 לכל $1 \leq i,j \leq n$ ולכל $k = 0,1,\ldots,n$ מתקיים: $\delta_k(i,j) = d_k(i,j)$

רעיון ההוכחה של הטענה:. הרעיון הוא שניתן להוכיח את הטענה לפי סדר החישוב של האלגוריתם, כלומר באינדוקציה על k (החל מאפס ועד n).

נוכיח שלכל $1 \leq i,j \leq n$ מתקיים השוויון $\delta_k(i,k) = d_k(i,j)$.

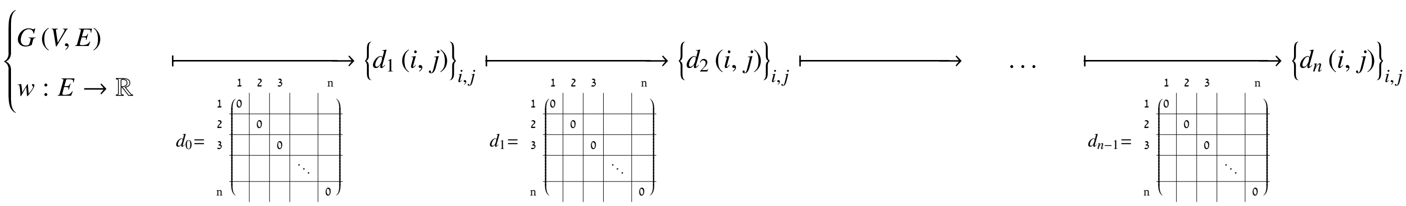
■

2.3. אלגוריתם Floyd-Warshall.

שאלה 5.8 בהינתן נכונות הטענה, כיצד נפתור את הבעיה?

2.3.1. הרעיון.

- אנחנו מעוניינים ב-$\delta_n(i,j)$ לכל $1 \le i,j \le n$.
- לפי הטענה, מספיק שנחשב את $d_n(i,j)$ לכל $1 \le i,j \le n$:
- לכל k, ניתן להסתכל על $\left\{d_k(i,j)\right\}_{1\le i,j\le n}$ כמטריצה / טבלה.
- תנאי העצירה של d, שמתאים ל-$k=0$, ממלא את הטבלה השמאלית.
- ערך $d_k(i,j)$ תלוי רק בשלושה ערכים, שנמצאים בטבלה עם ערך k-1 (הקודמת).
- באופן זה, ניתן למלא את כל הטבלאות ולהגיע לתשובה הרצויה:



-

אלגוריתם (קווי מתאר, Floyd-Warshall)**:**

- מתחילים עם הטבלה ש-d שמתאימה ל-$k=0$ (תנאי העצירה).
- לכל $k \in \{1,\ldots,n\}$:
  - נחשב (בסדר כלשהו) את כל התאים של הטבלה $\left\{d_k(i,j)\right\}_{1\le i,j\le n}$
- נחזיר את הפלט $\left\{d_n(i,j)\right\}_{1\le i,j\le n}$.

- <u>זמן הריצה:</u> $O\left(n^3\right)$.
- <u>זכרון:</u> $O\left(n^2\right)$ (כי כשמחשבים את הטבלה ה-k של d, מספיק לזכור רק את הטבלה ה-k-1 של d).

הערה 5.5 את בעיית שיבוץ המשימות כשיש רווח (אי-שלילי) לכל משימה, ניתן לפתור בעזרת תכנון דינאמי.

פרק 6

רשתות זרימה וזרימת מקסימום

לעתים קרובות, נוכל להשתמש בגרפים למידול רשתות תעבורה, שבהן הקשתות נושאות איזושהי "סחורה":
חבילות ברשת תקשורת, תחבורה של כלי רכב וכד'.
ברשתות כאלה, הצמתים משמשים כ"מתגים" לחיבור הקשתות.
נסתכל על רשתות כאלו ונתייחס לתעבורה בהן ע"י שימוש בזרימה.

1. זרימה ברשתות

הגדרה 6.1 (רשת זרימה) רשת זרימה הינה אוסף $N = (G, s, t, c)$, כאשר:

- $G = (V, E)$ גרף מכוון נתון. לצורך הפשטות, נניח ש-G פשוט:
  אינו מכיל לולאות עצמיות או קשתות מקבילות.
- לכל קשת $e \in E$ מוגדרת קיבולת $c(e) \geq 0$ (כלומר, $c : E \to \mathbb{R}^+$ הינה פונקציית הקיבולת).
- הצומת $s \in V$ הוא "מקור".
- הצומת $t \in V$ הוא "בור" (יתכן כי יש קשתות יוצאות מ-t ב-G).

הגדרה 6.2 (זרימה על קשת מכוונת) עבור רשת זרימה $N = (G, s, t, c)$, נסמן ב-$f(e)$ את הזרימה על הקשת המכוונת $e \in E$.

הגדרה 6.3 (זרימה חוקית) עבור רשת זרימה $N = (G, s, t, c)$ וקשת מכוונת $e \in E$,
נאמר ש-f היא זרימה חוקית אם היא מקיימת את התכונות הבאות:

(1) <u>אילוץ הקיבול</u>: הזרימה דרך קשת e אינה יכולה לחרוג מקיבולת הקשת: $\boxed{0 \leq f(e) \leq c(e)}$

(2) <u>שימור הזרימה</u>: לכל הצמתים $v \in V \setminus \{s, t\}$,
סך כל הזרימה הנכנסת ל-v שווה לסך כל הזרימה היוצאת מ-v, כלומר:

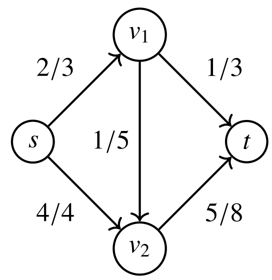
$$\sum_{e \in \text{in}(v)} f(e) = \sum_{e \in \text{out}(v)} f(e)$$

הערה 6.1 (סימון קשתות ברשת זרימה) בדרך כלל ברשתות זרימה, תופיע על כל קשת תווית מהצורה $e = u \xrightarrow{f(e)\ /\ c(e)} v$.

הגדרה 6.4 (ערך של זרימה) בהינתן פונקציית זרימה חוקית f, הערך של f הוא סך כל הזרימה מהמקור לבור, כלומר:

$$|f| = \sum_{e \in \text{in}(t)} f(e) - \sum_{e \in \text{out}(t)} f(e)$$

דוגמה 6.1 (רשת לדוגמה) נביט על הרשת הבאה לדוגמה, ונבדוק האם עומדת בתנאים לזרימה חוקית.



- נשמרים אילוצי הקיבול.
- מתקיים שימור הזרימה, שכן לכל צומת פנימית $v \in V \setminus \{s,t\}$, מתקיים שימור הזרימה:
  (1) עבור v_1: $f(s \to v_1) = 2 = 1 + 1 = f(v_1 \to s_2) + f(v_1 \to t)$
  (2) עבור v_2: $f(s \to v_2) + f(v_1 \to v_2) = 4 + 1 = 5 = f(v_2 \to t)$

- לכן פונקציית הזרימה חוקית.
- לכן מוגדרת הזרימה של f. במקרה זה:

$$|f| = \left(f(v_1 \to t) + f(v_2 \to t)\right) - \ 0 \ = 1 + 5 = 6$$

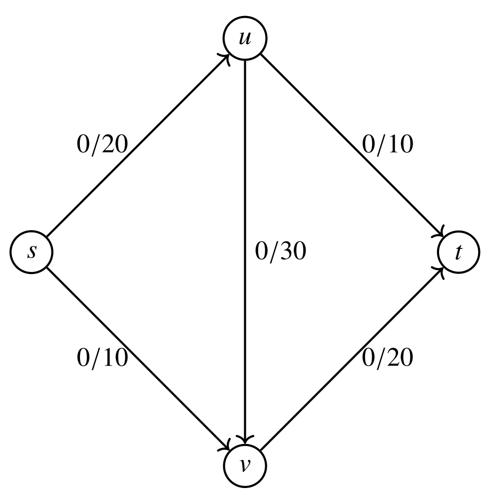
2. בעיית זרימת מקסימום

שאלה 6.1 מהי הזרימה המקסימלית שניתן להעביר מ-s ל-t, תוך שמירה על אילוצי הקיבולת של כל הקשתות?

2.1. אינטואיציה לגישה החמדנית. נראה מתאים לנסות את הגישה החמדנית.

דוגמה 6.2

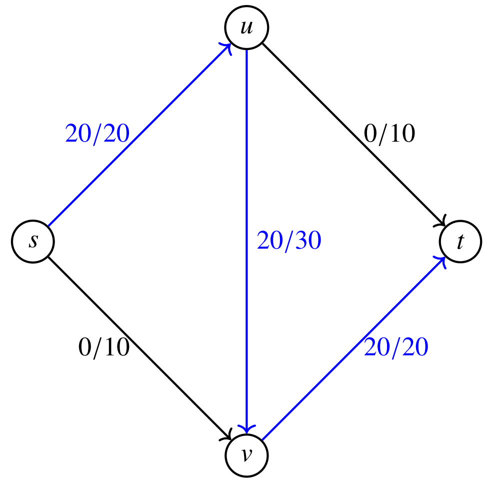
- נתונה רשת עם זרימה 0 על כל הקשתות:



איור 1. רשת לדוגמה.

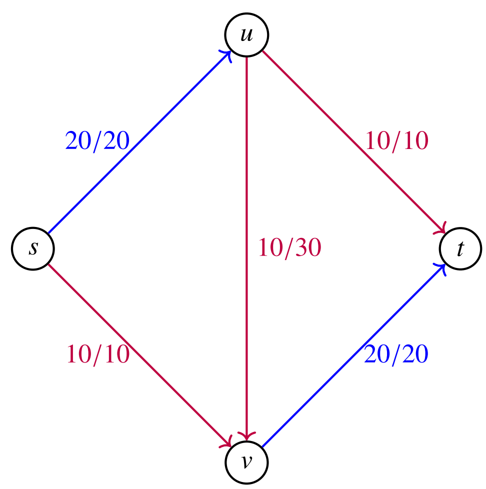
- ברשת זו, אפשר להעביר כמות מקסימלית של 30 יחידות זרימה (כ-$\sum_{e \in \text{out}(s)} c(e)$).

- באופן חמדני (בחירת קשתות בעלות קיבולת גדולה ביותר, יאפשרו לנו "להעביר יותר זרימה"), נוסיף 20 יחידות זרימה על המסלול $s \to u \to v \to t$:



איור 2. ניסיון ראשוני להגדיר זרימה ברשת הנתונה.

- נבחין שזרימה זו היא חוקית.
- הבעיה: לא ניתן להעביר זרימה נוספת על מסלול אחר מ-s ל-t!
- היינו רוצים:
  - להוציא 10 יחידות זרימה על הקשת $s \to v$.
  - "להחזיר" 10 יחידות זרימה על הקשת $u \to v$.
  - להעביר 10 יחידות זרימה על $u \to t$.



איור 3. רשת הזרימה לאחר העברת הזרימה בין הקשתות.

- קיבלנו שאנחנו מעבירים 10 יחידות זרימה על הקשתות הסגולות. כעת, ערך הזרימה ברשת הינו $|f| = 30$.
- זהו גם ערך הזרימה המקסימלי האפשרי, שכן בהכרח $|f| \leq \sum_{e \in \mathrm{in}(t)} c(e)$.
- העקרון החמדני שילווה אותנו: נבצע איטרציות, כאשר בכל איטרציה ננסה באופן חמדני להעביר כמה שיותר זרימה.

דוגמה זו ממחישה את האופן שבו בכל איטרציה, ניתן לא רק "להוסיף" זרימה על איזשהו מסלול $s \rightsquigarrow t$, אלא גם "להחסיר" זרימה מקשתות מסוימות, כדי שנוכל לנתב אותה לקשתות אחרות.

2.2. הרשת השיורית.

מוטיבציה: נגדיר גישה כללית לשיפור הזרימה ברשת:

- נוכל לדחוף זרימה "קדימה" על קשתות שאינן רוויות (דהיינו, לא השתמשנו בכל הקיבול שלהן).
- נוכל לדחוף זרימה "אחורה" על קשתות שיש עליהן זרימה, כדי לנתב את הזרימה עליהן לקשתות אחרות.

נממש את הגישה ע"י הגדרת רשת שיורית.

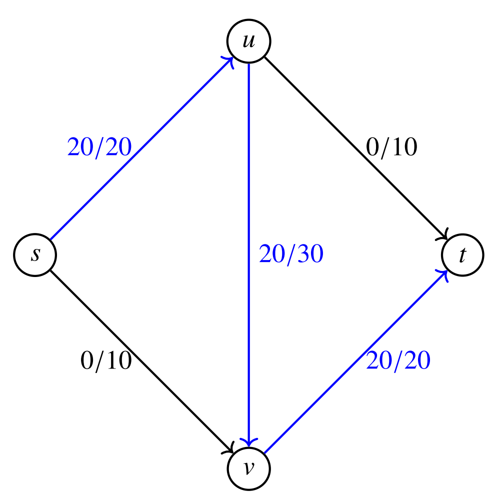
הגדרה 6.5 (רשת שיורית)
בהינתן רשת זרימה $N = \left((V,E), s, t, c\right)$ עם זרימה f, נגדיר את הרשת השיורית N_f ביחס ל-f באופן הבא:

- צמתים: אוסף הצמתים ב-N_f הוא כמו ב-N (כך גם הצמתים s, t).
- קשתות קדמיות: לכל קשת $e = u \to v$ ב-N עם $f(e) < c(e)$, יש $c(e) - f(e)$ יחידות זרימה נוספות, שמאפשרות לדחוף "קדימה" עוד זרימה.
  נוסיף ל-N_f את הקשת $e = u \to v$ עם קיבולת שיורית: $c(e) - f(e)$.
  נקרא לקשת כזו קשת קדמית.
- קשתות אחוריות: לכל קשת $e = (u, v)$ ב-N עבורה $f(e) > 0$, ניתן לבטל $f(e)$ יחידות זרימה, ע"י דחיפת זרימה "אחורה".
  נוסיף ל-N_f את הקשת $e' = v \to u$, עם קיבולת שיורית $f(e)$.
  נקרא לקשת כזו קשת אחורית.

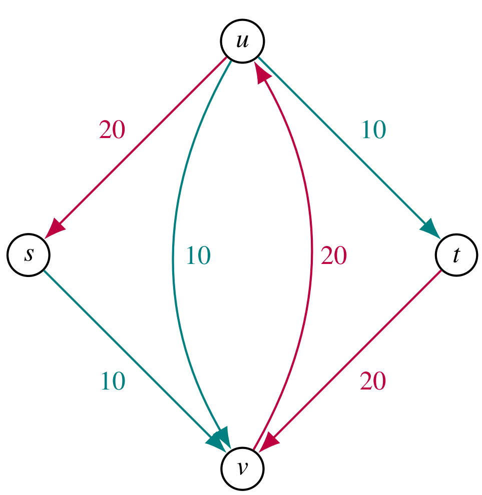
סה"כ יהיו ב-N_f לכל היותר $2|E|$ קשתות, כאשר הקיבולת השיורית מושרה ע"י הקשתות הקדמיות והאחוריות.

הערה 6.2 (סימון קשתות ברשת השיורית) בדרך כלל ברשתות שיוריות, תופיע על כל קשת תווית מהצורה $e = u \xrightarrow{c_{N_f}(e)} v$; כלומר הקיבולת השיורית מופיעה על הקשת.

דוגמה 6.3 (דוגמה לרשת שיורית)



איור 4. רשת הזרימה N.



איור 5. הרשת השיורית N_f.

בטורקיז: קשתות קדמיות.
בסגול: קשתות אחוריות.

2.3. מסלולי שיפור.

שאלה 6.2 כיצד נשפר את הזרימה בעזרת הרשת השיורית?

אלגוריתם (שיפור הזרימה על P ב-N_f)**:** יהי $P = s \rightsquigarrow t$ מסלול פשוט ב-N_f.

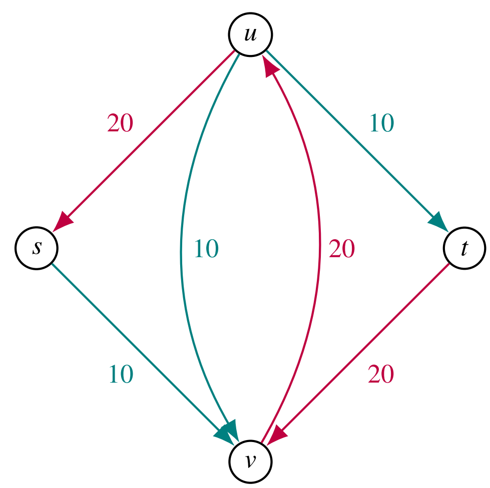
(1) נסמן ב-Δ את הקיבולת השיורית המינימלית על איזושהי קשת ב-P.

(2) לכל קשת $(u \to v) \in P$:

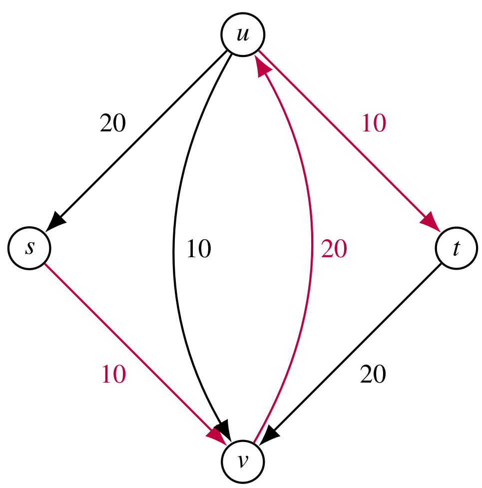
- אם $e = (u \to v)$ היא קשת קדמית, אזי נגדיל $f(e) \leftarrow f(e) + \Delta$ (ברשת הזרימה N).
- אחרת, תהי $e = (v \to u)$ נקטין $f(e) \leftarrow f(e) - \Delta$ (ברשת הזרימה N).

הגדרה 6.6 (מסלול שיפור) מסלול $s \rightsquigarrow t$ ב-N_f נקרא מסלול שיפור.

דוגמה 6.4 (שיפור זרימה לדוגמה ברשת שיורית) נתבונן ברשת השיורית N_f עבור רשת הזרימה מהסעיף הקודם.
יהי $P = s \to v \to u \to t$ המסלול הפשוט הבא ב-N_f:



איור 6. הרשת השיורית N_f.



איור 7. המסלול P ב-N_f.

- מתקיים כי $\Delta = 10$.
- $s \to v$ היא קשת קדמית, לכן נגדיל $f(s \to v) \leftarrow f(s \to v) + \Delta = 10 + 10 = 20$.
- $v \to u$ היא קשת אחורית, לכן נקטין $f(u \to v) \leftarrow f(u \to v) - \Delta = 20 - 10 = 10$.
- $u \to t$ היא קשת קדמית, לכן נגדיל $f(u \to t) \leftarrow f(u \to t) + \Delta = 0 + 10 = 10$.

שאלה 6.3 האם אחרי פעולת השיפור נקבל זרימה חוקית ב-N?
ואם כן, בכמה הגדלנו את ערך הזרימה ברשת?

למה 6.1 (למה 1) פעולת שיפור הזרימה על מסלול P ב-N_f עם קיבולת שיורית מינימלית Δ,
תגדיר ב-N פונקציית זרימה חוקית f', כך שמתקיים:

$$\left|f'\right| = |f| + \Delta$$

הוכחה. לא הוכח בסמסטר זה. ■

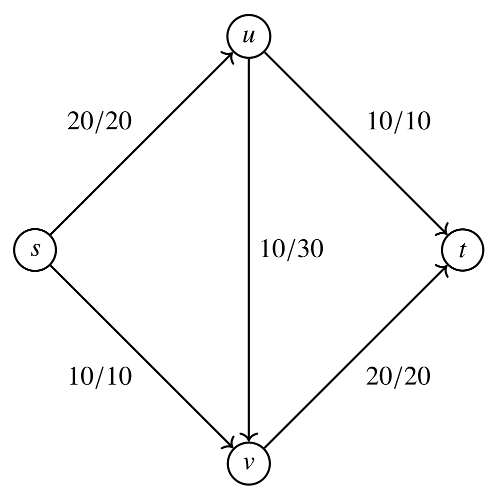
נראה אלגוריתם שמוצא זרימת מקסימום בעזרת רשת שיורית ומסלולי שיפור.

2.4. שיטת Ford-Fulkerson למציאת זרימת מקסימום.

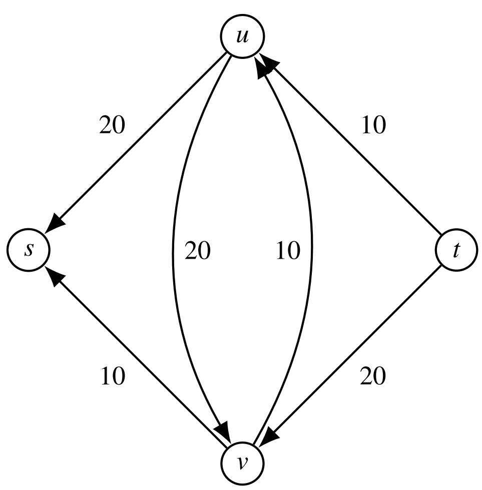
אלגוריתם (גנרי, Ford-Fulkerson):

- אתחול: זרימה 0 בכל הקשתות.
- איטרציות: כל עוד קיים מסלול שיפור $P = s \rightsquigarrow t$ ב-N_f, שפר עליו את הזרימה.

דוגמה 6.5 נחזור לדוגמה הקודמת. נוכל לראות את צעדי שיפור הזרימה ברשת N כאיטרציות של FF.



איור 8. רשת הזרימה אחרי השיפור
(דוגמה קודמת)



איור 9. הרשת השיורית $N_{f'}$.

במקרה זה, מכיוון שאין מסלול שיפור $s \rightsquigarrow t$ ב-$N_{f'}$, אלגוריתם Ford-Fulkerson מגיע לעצירה.

2.4.1. זמן הריצה של Ford-Fulkerson.

- אם קיבולי כל הקשתות שלמים, מס' האיטרציות של FF הוא לכל היותר $|f^*|$, כאשר f^* היא זרימת מקסימום (כי בכל איטרציה, נגדיל את הזרימה לפחות ב-1).
- ניתן למצוא מסלול שיפור ב-N_f ב-$O(|E|)$ ע"י הרצת BFS/DFS.
- סך הכל, זמן הריצה: $O(|E| \cdot |f^*|)$

את נכונות האלגוריתם נראה לאחר שנחקור את הנושא של חתכים ברשת זרימה.

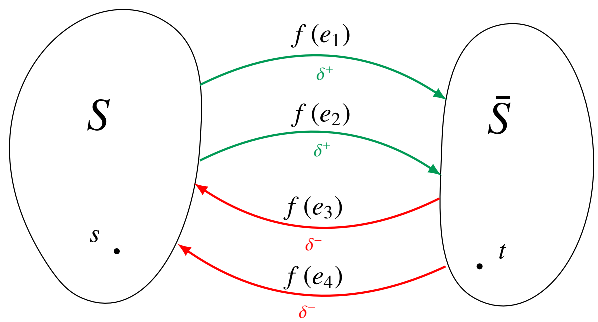
2.5. חתכים ברשת זרימה.

הגדרה 6.7 (חתך s-t ברשת זרימה) עבור $S \subset V, \ \bar{S} := V \setminus S$, $\left(S, \bar{S}\right)$ נקרא <u>חתך s-t</u> אם $s \in S$, אבל $t \in \bar{S}$.

הגדרה 6.8 (קשתות חוצות חתך s-t) נציג את הסימונים הבאים לקשתות שחוצות את החתך מ-S ל-$\bar{S}$ (+) ולהיפך (-):

$$\delta^{+}(s) = \left\{e = (u,v) \in E \mid u \in S, v \in \bar{S}\right\}$$
$$\delta^{-}(s) = \left\{e = (u,v) \in E \mid u \in \bar{S}, v \in S\right\}$$

דוגמה 6.6 (קשתות חוצות בחתך s-t של רשת זרימה)

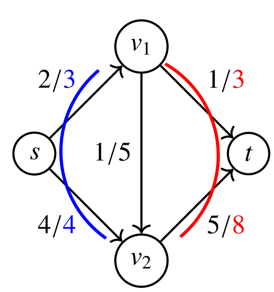


איור 10. קשתות בכיוון החתך וקשתות נגד כיוון החתך.

הגדרה 6.9 (קיבול של חתך) קיבול של חתך $\left(S, \bar{S}\right)$ מוגדר על ידי:

$$c(S) = \sum_{e \in \delta^{+}(S)} c(e)$$

דוגמה 6.7 נתונה הרשת:



- ניקח את החתך $\left(S = \{s, v_1, v_2\}, \ \bar{S} = \{t\}\right)$. מתקיים כי $c(S) = 11$.
- אם ניקח $\left(S = \{s\}, \ \bar{S} = \{v_1, v_2, t\}\right)$, נקבל $c(S) = 7$.

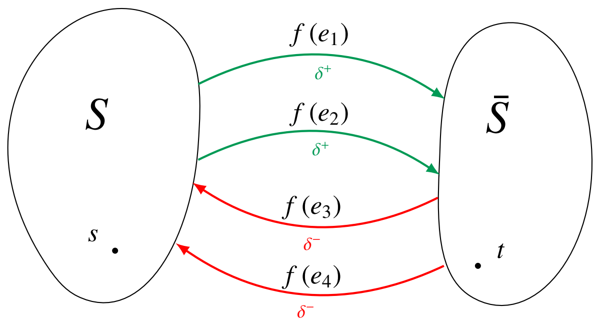
שאלה 6.4 איך נחשב את ערך הזרימה ברשת ע"י שימוש בחתכים?

למה 6.2 (למה 2) לכל חתך s-t $\left(S,\bar{S}\right)$ ולכל זרימה f מתקיים:

$$|f| = \sum_{e\in\delta^{+}(S)} f(e) - \sum_{e\in\delta^{-}(S)} f(e)$$

דוגמה 6.8 (דוגמה לחישוב ערך הזרימה ע"י חתך s-t)

נתבונן בדוגמה שראינו לחתך s-t:



מתקיים:

$$\boxed{|f| = \left(f(e_1)+f(e_2)\right) - \left(f(e_3)+f(e_4)\right)}$$

<u>הוכחה.</u> נסתכל על הסכום הבא:

$$\sum_{v\in\bar{S}}\left(\sum_{e\in \text{in}(v)} f(e) - \sum_{e\in \text{out}(v)} f(e)\right) \underbrace{=}_{(1)} \sum_{e\in \text{in}(t)} f(e) - \sum_{e\in \text{out}(t)} f(e) \underbrace{=}_{|f| \text{ מהגדרת}} |f|$$

$$\sum_{v\in\bar{S}}\left(\sum_{e\in \text{in}(v)} f(e) - \sum_{e\in \text{out}(v)} f(e)\right) \underbrace{=}_{(2)} \sum_{e\in\delta^{+}(S)} f(e) - \sum_{e\in\delta^{-}(S)} f(e)$$

(1) נשים לב כי משימור הזרימה, נובע שלכל צומת $v\in\bar{S}\setminus\{t\}$ מתקיים:

$$\sum_{e\in \text{in}(v)} f(e) - \sum_{e\in \text{out}(v)} f(e) = 0$$

(2) מצד שני, לכל קשת $e=(u,v)$ כך ש-$u,v\in\bar{S}$, נקבל בסכום את $f(e)$ פעמיים:
פעם אחת בסימן + (עבור הצומת v), ופעם אחת בסימן − (עבור הצומת u).
לכן, איברים אלו יתקזזו בסכום, וישארו רק האיברים המתייחסים לקשתות ב-$\delta^{+}(S)$ או ב-$\delta^{-}(S)$.

מ-(1) ו-(2) קיבלנו ש-$|f| = \sum_{e\in\delta^{+}(S)} f(e) - \sum_{e\in\delta^{-}(S)} f(e)$.

■

הערה 6.3 מלמה 2, נוכל לחשב את ערך פונקציית הזרימה $|f|$ ע"י הסתכלות על חתך s-t כלשהו $\left(S,\bar{S}\right)$.

שאלה 6.5 מה בין ערך פונקציית הזרימה לקיבולי החתכים ברשת?

למה 6.3 (למה 3) לכל חתך $\left(S,\bar{S}\right)$ ולכל פונקציית זרימה f, מתקיים:

$$|f| \le c(S)$$

<u>הוכחה</u>. בהינתן חתך $\left(S,\bar{S}\right)$, נקבל מהלמה הקודמת כי מתקיים:

$$|f| = \sum_{e\in\delta^+(S)} f(e) - \sum_{e\in\delta^-(S)} f(e) \underbrace{\le}_{f\ge 0} \sum_{e\in\delta^+(S)} f(e) \underbrace{\le}_{\text{את אילוץ הקיבול } \text{מקיימת } f} \sum_{e\in\delta^+(S)} c(e) = c(S)$$

■

מסקנה 6.1 (ישירה מהלמה) אם קיים חתך s-t $\left(S,\bar{S}\right)$ כך ש-$|f| = c(S)$, אזי f זרימת מקסימום.

משפט 6.1 (משפט חתך-מינימום זרימת מקסימום / Min-Cut Max-Flow)
תהי f פונקציית זרימה ברשת זרימה $N = (G, s, t, c)$.
אזי, הטענות הבאות שקולות:

(1) f היא זרימת מקסימום.
(2) אין מסלול שיפור מ-s ל-t ברשת השיורית N_f.
(3) קיים חתך s-t $\left(S,\bar{S}\right)$ עבורו $|f| = c(S)$.

הגדרה 6.10 (חתך מינימום) חתך ה-s-t $\left(S,\bar{S}\right)$ שעבורו $|f| = c(S)$ מכונה חתך מינימום.

<u>הוכחת משפט Min-Cut Max-Flow</u>. די להראות כי $(1) \to (2) \to (3) \to (1)$:

- <u>$(1) \to (2)$</u>: אם f זרימת מקסימום אז לא יתכן שיש מסלול שיפור $s \rightsquigarrow t$ ב-N_f. אחרת, לפי למה 1, אפשר היה להגדיל את $|f|$.

- <u>$(2) \to (3)$</u>: נגדיר:

$$S = \left\{v \mid \text{יש מסלול מכוון ב-} N_f \text{ מ-} s \text{ ל-} v\right\}$$

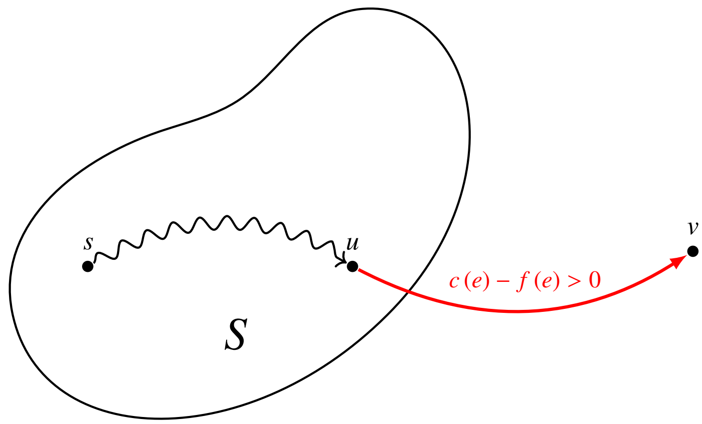
ברור כי $s \in S$ ו-$t \notin S$, אחרת היה מסלול שיפור מ-s ל-t ב-N_f. יהי $\bar{S} = V \setminus S$. קיבלנו כי $\left(S,\bar{S}\right)$ הוא חתך s-t ברשת. מלמה 2:

$$|f| = \sum_{e\in\delta^+(S)} f(e) - \sum_{e\in\delta^-(S)} f(e)$$

נחשב את $|f|$ בעזרת הטענות הבאות:

(טענה) אם $e \in \delta^+(S)$, אזי $f(e) = c(e)$, כלומר הקשת רוויה.

הוכחה: תהי $e = (u,v)$, כאשר $u \in S$ ו-$v \in \bar{S}$. נניח בשלילה ש-$f(e) < c(e)$,
אזי ב-N_f תהיה קשת $e = u \to v$ עם קיבולת שיורית $c(e) - f(e)$.
לכן אפשר להגיע ל-v ב-N_f, כלומר $v \in S$, סתירה להגדרת הקבוצה S.



איור 11. נקבל סתירה להגדרת S.

(טענה) אם $e \in \delta^-(S)$, אזי $f(e) = 0$.

אם נסמן $e = (u,v)$ כך ש-$u \in \bar{S}$ ו-$v \in S$.
נניח בשלילה כי $f(e) > 0$.
אזי ב-N_f יש קשת אחורית מ-v ל-u, עם קיבולת שיורית $f(e) > 0$.
לכן יש מסלול מכוון $s \rightsquigarrow u$ ב-N_f, בסתירה לכך ש-$u \notin S$.

מטענות 1 ו-2 נקבל:

$$|f| \underbrace{=}_{\text{מלמה 2}} \sum_{e\in\delta^+(S)} f(e) - \sum_{e\in\delta^-(S)} f(e) \underbrace{=}_{\text{מהטענות}} \sum_{e\in\delta^+(S)} f(e) - \sum_{e\in\delta^-(S)} 0 = c(S)$$

- $(3) \to (1)$: נובע ישירות מסמקנה מלמה 2.

■

למה 6.4 (למה 4) תהי f זרימת מקסימום ברשת $N = (G,s,t,c)$, ותהי $S = \{v \mid \text{יש מסלול מכוון ב-} N_f \text{ מ-} s \text{ ל-} v\}$. אזי:

$$c(S) = |f|$$

הוכחה. מלמה 1 נובע כי אם f זרימת מקסימום אזי אין מסלול שיפור ב-N_f. לכן $(S,\bar{S})$ הוא חתך s-t.
כעת, מטענות 1 ו-2 בהוכחת משפט ה-Max-Flow Min-Cut, נובע כי $c(S) = |f|$. ■

2.6. נכונות Ford-Fulkerson.

משפט 6.2 (נכונות האלגוריתם) לכל רשת זרימה $N=(G,s,t,c)$, האלגוריתם של Ford-Fulkerson מוצא זרימת מקסימום f^* ב-N.

<u>הוכחה</u>. ממשפט Min-Cut Max-Flow נובע כי אם אין מסלול שיפור ב-N_f, אזי f זרימת מקסימום.
לכן האלגוריתם של Ford-Fulkerson עוצר כאשר נמצאה זרימת מקסימום f^* ברשת. ■

למה 6.5 (למה 5) אם כל הקיבולים ברשת שלמים, אז Ford-Fulkerson מוצא זרימת מקסימום בשלמים.

<u>הוכחה</u>. נראה כי בכל איטרציה של Ford-Fulkerson, לכל $e\in E$ מתקיים כי $f(e)$ והקיבולת השיורית ב-N_f הם מספרים שלמים.

- ברור כי הטענה נכונה לפני איטרציה 1, כי הזרימה על כל הקשתות שווה ל-0.
- נניח כי הטענה נכונה אחרי איטרציה j.
- עבור האיטרציה ה-$j+1$, היות שכל הקיבולות השיוריות ב-N_f הן ערכים שלמים, גם הקיבולת השיורית המינימלית במסלול השיפור באיטרציה זו הוא מספר שלם, נסמנו Δ_{j+1}.

אחרי השיפור נקבל כי לכל קשת $e\in E$, $f'(e)$ הוא מספר שלם, וכך גם יהיו הקיבולות ברשת השיורית החדשה $N_{f'}$.

■