הרצאה 1

אלגוריתמים לחיפוש בגרפים

דוגמה:

s$\rightarrow$ $\downarrow$	$\rightarrow$ $\downarrow$		
		■	
■			
			T

ההתקדמות בכל צעד: ימינה, שמאלה למטה

המטרה:

למצוא את המסלול הקצר ביותר מ-S ל-T.

חיפוש בגרף:

נתון גרף $G = (V, E)$ וצומת מקור $s \in V$.

1. יש לגלות את כל הצמתים ב-G שהם ברי-הגעה מ-s.
2. עבור צומת v שהוא בר-הגעה מ-s יש למצוא את המרחק הקצר ביותר בקשתות מ-s ל-v.

האלגוריתם BFS (Breath First Search)

עובר על כל הקשתות של G ו"מגלה" את קבוצת הצמתים שניתנים להגעה מ-s, $R \subseteq V$.

פלט האלגוריתם:

- ⋆ עץ ששורשו s המכיל את כל הצמתים ב-R.
- ⋆ המרחק, דהיינו מספר הקשתות המינימלי מ-s לכל צומת ב-R.
- ⋆ ניתן להפעיל את BFS גם על גרפים מכוונים והן על גרפים לא-מכוונים.

פסאודו קוד ל-BFS:

Input: A graph $G = (V, E)$, s$\in V$

Output: For any $v \in V$, $d(v)$ is the distance from s to v.

For any $v \in V$, do:

$d(v) \leftarrow \infty$

$d(s) \leftarrow 0$

$i \leftarrow 0$

While there is neighbor v of u with $d(v)$= ∞ do:

d(v)$\leftarrow i+1$

$i \leftarrow i+1$

מימוש BFS באמצעות תור:

בתחילה התור Q ריק.

צומת v ש"התגלה" נכנס ל-Q.

צומת w יוצא מראש התור לקראת סריקת שכניו.

<u>האב</u> של צומת v בעץ החיפוש הוא הצומת שגרם ל-v להתגלות.

האלגוריתם רץ כל עוד Q לא ריק.

סימונים:

* $d(v)$ - המרחק של v מ-s.
* $\pi(v)$ - האב של v בעץ החיפוש.
* $Adj(v)$ -קבוצת השכנים של v ב-G.

```
For any u ∈ V\{s} do: {d(u) ← ∞, π(u) ← NULL}
d(s) ← 0
Q ← ∅
Enqueue(Q, s)
While Q ≠ ∅ do :
  u ← dequeue(Q)
  Foreach v ∈ Adj(u) do:
    if d(v) = ∞ then:
      d(v) ← d(u) + 1
      π(v) ← u
      Enqueue(Q, v)
```

זמן הריצה של BFS:

אתחול: $O(|V|)$

לולאת ה-while:

עבור כל צומת v שיוצא מהתור עוברים על כל השכנים, דהיינו לכל היותר $O(|Adj(v)|)$ סה"כ נעבור על כל קשת בגרף לכל היותר פעמיים לכן זמן הריצה הוא $O(|E|)$.

הוכחת נכונות BFS:

- ⋆ נראה כי BFS מוצא את המסלול הקצר מ-s לכל צומת v אשר בר הגעה מ-s.
- ⋆ נסמן ב-$\delta(s,v)$ את המרחק הקצר ביותר (בקשתות) מ-s ל-v. אם אין מסלול מ-s ל-v אזי $\delta(s,v)=\infty$.

למה 1:

לכל קשת $(u,v)\in E$ **מתקיים** $\delta(s,v)\leq\delta(s,u)+1$

הוכחת למה 1:

1. אם u בר הגעה מ-s, אזי המסלול הקצר מ-s ל-v הוא לכל היותר המסלול הקצר מ-s ל-u + הקשת (u,v).
2. אחרת $\delta(s,u)=\infty$ ולכן אי-השוויון מתקיים.

למה 2:

בסיום האלגוריתם הערך $d(v)$ המחושב ע"י BFS מקיים $d(v)\geq\delta(s,v)$ לכל $v\in V$.

הוכחה:

נוכיח באינדוקציה על מספר צעד ה-Enqueue, שאחרי צעד כזה יתקיים $d(v) \geq \delta(s,v)$ לכל $v \in V$.

בסיס: אחרי פעולת $\text{Enqueue}(Q,s)$ הטענה מתקיימת כי $d(s) = 0 = \delta(s,s)$ ולכל $v \in V \backslash \{s\}$ מתקיים $d(v) = \infty$.

צעד האינדוקציה: נניח שצומת v "התגלה" תכאשר סרקנו את השכנים של u. מכאן, v נכנס לתור. מהנחת האינדוקציה, $d(u) \geq \delta(s,u)$. לכן:

$$d(v) \underbrace{=}_{\text{Algorithm}'} d(u) + 1 \underbrace{\geq}_{\text{Induction}} \delta(s,u) + 1 \underbrace{\geq}_{\text{Lema 1}} \delta(s,v)$$

היות ש-$d(v)$ לא ישתנה במהלך האלגוריתם, אי-השוויון שמצאנו לעיל מתקיים גם בסוף האלגוריתם.

למה 3:

נניח שבמהלך הביצוע של BFS התור Q מכיל את $\{v_1, v_2, \ldots, v_r\}$ אזי:

$$d(v_r) \leq d(v_1) + 1$$

$$\forall 1 \leq i \leq r, d(v_i) \leq d(v_{i+1})$$

מסקנה 4:

נניח ש-v_i נכנס ל-Q לפני ש-v_j נכנס לתור. אזי, $d(v_i) \leq d(v_j)$.

הוכחה:

נתבונן ב-v_i ו-v_j וסדרת הצמתים שהוכנסו ביניהם ל-Q: $\{v_i, v_{i+1}, v_{i+2}, v_j\}$
כל זוג צמתים עוקבים בסדרה v_k, v_{k+1} מקיימים את אחד מהתנאים הבאים:

1. v_k, v_{k+1} יחד בתור ולכן לפי למה 3 $d(v_k) \leq d(v_{k+1})$.
2. v_{k+1} נכנס לתור Q אחרי ש-v_k יוצא ממנו אזי מפני שאין צומת ביניהם בסדרה נובע כי v_kגורם ל-v_{k+1} להתגלות ולכן $d(v_{k+1}) = d(v_k) + 1$.

$\Leftarrow$

$$d(v_i) \leq d(v_2) \leq \cdots \leq d(v_l) \leq d(v_j)$$

$\blacksquare$.

משפט:

נניח שמריצים את BFS על G (מכוון/לא מכוון) עם צומת מקור s,
אזי BFS מגלה כל צומת v שהוא בר-השגה מ-s ובסיום האלג' $d(v) = \delta(s, v)$.
בנוסף, לכל צומת v בר-השגה מ-s אחד המסלולים הקצרים ל-v הוא מסלול קצר ביותר מ-s ל-$\pi(v)$, שאליו נוספת הקשת $(\pi(v), v)$.

הוכחה:

נניח בשלילה שקיים צומת v עבורו $d(v) \neq \delta(s, v)$ מלמה 2, מפני ש-$d(v) \geq \delta(s, v)$ אזי נקבל $d(v) > \delta(s, v)$. נקח את הצומת v עם $\delta(s, v)$ מינימלי שעבורו מתקיים אי-השוויון החזק.
נסתכל על המסלול הקצר ביותר מ-s ל-v. יהי u הצומת שמופיע מיד לפני v במסלול זה. אזי, $\delta(s, v) = \delta(s, u) + 1$.
מאופן הבחירה של v כצומת בעל $\delta(s, v)$ המינימלי שמקיים את אי-השוויון החזק, נקבל כי $d(v) = \delta(s, u)$.
מכאן:

$$d(v) > \delta(s, v) = \delta(s, u) + 1 = d(u) + 1 \quad (**)$$

נראה כי אי-השוויון (**) לא יתכן. נסתכל על הצעד בו u יוצא מהתור נבחין ב 3 מקרים:

1. v עוד לא "התגלה", לכן מהאלגוריתם נקבל $d(v) = d(u) + 1$ $\Leftarrow$ סתירה ל-(**).
2. v כבר לא היה בתור. לכן, v נכנס לתור לפני u (שכן התור FIFO) וממסקנה 4 $d(v) \leq d(u)$ $\Leftarrow$ סתירה ל-(**).
3. הצומת v עדיין בתור. לכן, מיד לפני ש-v יצא מהתור, u ראשון בתור ו-v "במקרה הגרוע" אחרון בתור. ולכן מלמה 3 $d(v) \leq d(u) + 1$ $\Leftarrow$ סתירה ל-(**).

מכאן נקבל כי לכל צומת v שהוא בר הגעה מ-s מתקיים $d(v) = \delta(s, v)$.
לסיום, נזכיר כי אם $\pi(v) = u$ אזי $d(v) = d(u) + 1 \Leftrightarrow \delta(s, v) = \delta(s, u) + 1$
לכן, ניתן לקבל מסלול קצר ביותר מ-s ל-v ע"י הוספת $(\pi(v), v)$ למסלול הקצר ביותר מ-s ל-$\pi(v)$.

הרצאה 2

אלגוריתם חיפוש לעומק

דוגמה:

$\overbrace{\downarrow}^{\text{Start}}$	☻	$\rightarrow$	
$\downarrow\rightarrow$	$\rightarrow$	$\uparrow$ $\downarrow$	☻
	☻	$\rightarrow$	$\downarrow$
☻		☻	

רובוט סורק אזור לצורך גילוי מוקשים, ההתקדמות בכל צעד: ימינה שמאלה למטה, למעלה.

אלגוריתם DFS (Depth First Search)

מנסה להתקדם כמה שיותר לעומק הגרף.
כאשר נבקשר בצומת v, אם יש קשת (u,v) לצומת u שעוד לא "התגלה", נחצה את הקשת ונמשיך את החיפוש מהצומת u.
המטרה: יש לגלות את כל הצמתים בגרף.

DFS על גרף מכוון:

קלט: גרף מכוון $G=(V,E)$, צומת s.
פלט: לכל $v \in V$, $d[v]$, זמן הגילוי של v.
סימונים:

- $\star$ $d[v]$ זמן גילוי של v.
- $\star$ $\pi[v]$ הצומת שגרם ל-v להתגלות.

```
DFS:
1.For all v ∈ V  d[v] ← 0,π[v] ← null
     mark all edges "unused"
     i ← 0,  v ← s
.2i ← i+1,  d[v] ← i
3.While there are unused out-edges from v,
     choose unused edges (v,u), mark (v,u) as used
     if d[u] = 0 :  {π[u] ← v, v ← u, i ← i+1, d[v] ← i}
4.If π[v] ≠ nullthen v ← π[v] and go to (3)
     else if there is u ∈ Vwith d[u] = 0
     then v ∈ u and go to (2).
5.stop
```

- ⋆ נשים לב כי בהרצות שונות של DFS נוכל לקבל פלטים שונים, אך הכולן נקבל "יער" שבו כל צומת מופיע מאיזשהו עץ מכוון.
- ⋆ בנוסף , DFS לא בהרכח מוצא מרחקים קצרים.
- ⋆ בסיום הרצת DFS נקבל predecessor subgraph. זהו תת-גרף שבו לכל צומת v מופיע קשת $(v,\pi(v))$ כפי שנמצא ע"י האלגוריתם, סימון: G_π.

For each $u \in V$ do:

    $\{$color$[u] \leftarrow$ white$, \pi[u] \leftarrow null\}$

For each $u \in V$ do :

    if color$[u]$ =white then DFS-VISIT(u(

DFS-VISIT(u):

    $color[u] \leftarrow gray$

    $i \leftarrow i+1$

    $d[u] \leftarrow i$

    For each $v \in Adj[i]$ do

        if color[v[=white then $\{\pi[v] \leftarrow u,$ DFS-VISIT$(v)\}$

    $i \leftarrow i+1$

    $f[u] \leftarrow i$

color$[u]$ - $\{white, gray\}$.

$f[u]$ - זמן היציאה האחרון מ-u.

$Adj[u]$ - אוסף השכנים של u.

זמן הריצה של DFS :

* ⋆ לולאת האתחול: $\theta(|V|)$.
* ⋆ נסמן ב-T(DFS-VISIT) את מספר הפעולות המבוצעות בקריאה ל-DFS-VISIT עבור הצומת .
נשים לב כי DFS-VISIT נקראת בדיוק פעם אחת עבור v כאשר v "לבן", ומיד בכניסה לפרוצדורה v הופך ל-"אפור". בנוסף, מספר הפעולות בלולאת ה-"For"
של DFS-VISIT הוא <u>לינארי</u> במספר השכנים של v. לכן סיבוכיות הקריאות ל-:DFS-VISIT

$$\sum_{v\in V}\text{T(DFS-VISIT)}=\sum_{v\in V}\theta(|Adj[v]|)=\theta(|E|)$$

⇐ סה"כ זמן הריצה של DFS: $\Theta(|E|+|V|)$.

תכונות של DFS:

1. <u>התכונה הבסיסית:</u> G_π הוא יער, שכן המבנה של עצי DFS מקשף את הקריאות הרקורסיביות ל-DFS-VISIT.
2. v הוא צאצא של u בעץ DFS אם v התגלה כאשר u היה אפור ולפני שנקבע ערך ל-$f[u]$.
3. <u>תכונת הסוגריים:</u> נייצג את הגילוי של צומת u ע"י סוגר שמאלי $'(u'$ ואת סיום הטיפול בו ע"י $'u)'$.
אזי, ההיסטוריה של "גילוי" ו"סיום הטיפול" מגדירה ביטוי שבו הסוגריים מקוננים היטב.

משפט 1 (הסוגריים):

בכל חיפוש לעומק של גרף מכוון/לא-מכוון G, לכל שני צמתים u ו-v מתקיים בדיוק אחד מהתנאים:

1. האנטרוולים $[d[u],f[u]]$ ו- $[d[v],f[v]]$ זרים לחלוטין ואין קשר של אב-קדמון/צאצא בין הצמתים.
2. האנטרוול $[d[u],f[u]]$ מוכל ממש בתוך $[d[v],f[v]]$ ו-u צאצא של v בעץ DFS.
3. $[d[v],f[v]]$ מוכל ממש ב-$[d[u],f[u]]$ ו-v צאצא של u בעץ DFS.
4. תכונה נוספת של צאצאים ביער במשפט הבא.

משפט 2 (<u>המסלול הלבן</u>):

ביער DFS של גרף (מכוון/לא-מכוון) צומת v הוא צאצא של צומת u אם"ם בזמן $d[u]$, הזמן בו u התגלה,ניתן להגיע ממנו ל-v ע"י מסלול המורכב כולו מצמתים לבנים.

הוכחה:

⇐

נניח ש-v צאצא של u. יהיה w צומת על המסלול בין u ו-v בעץ DFS כך ש-w צאצא של u.
ממשפט 1, $d[u]<d[w]$, לכן w היה לבן בזמן $d[u]$.

$\Rightarrow$

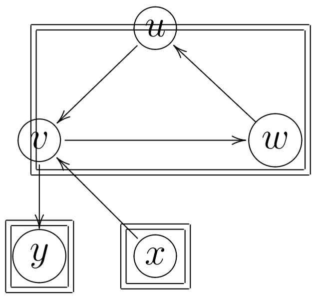
נניח בשלילה שיש מסלול לבן מ-u ל-v בזמן $d[u]$, אבל v לא נהיה צאצא של u בעץ DFS.
נניח ש-v הוא הצומת הראשון על המסלול הלבן שאנינו צאצא של u.
יהיה w הצומת לפני v על המסלול הלבן כך ש-w צאצא של u (או $w = u$).
אזי ממשפט 1 $f[w] \leq f[u]$.
נשים לב כי v מוכרח להתגלות אחרי u, אבל לפני שנצא בפעם האחרונה מ-w דהיינו:

$$d[u] \underbrace{<}_{\text{at d[u] there is a white path to v}} d[v] \underbrace{\leq}_{\text{we won't finish with w before we get to v}} f[u]$$

ממשפט 1 נקבל כי $[d[v], f[v]]$ חייב להיות מוכל ממש ב- $[d[u], f[u]]$
ולכן v יהיה צאצא של u בעץ DFS- סתירה $\blacksquare$.

אלגוריתמים - 3

DFS ושימושיו



הגדרות:

רכיב קשיר היטב(רק"ה): בגרף מכוון $G = (V, E)$ הוא קב' מקסימלית
של צמתים $C \subseteq V$. כך שלכל זוג צמתים $u, v \in C$
יש מסלול מכוון מ-v ל-u $(v \to u)$
ומ-u ל-v $(u \to v)$.
צומת בודד יכול להיות רק"ה.

דוגמה:

- הגרף ההופכי של גרף מכוון $G = (V, E)$ **הוא** $G^T = (V, E^T)$
$E^T = \{(u, v) : (v, u) \in E\}$

(דהיינו הופכים את כיווני הקשתות ב-G).
- נשים לב כי ב-G וב-G^T יש בדיוק אותם רק"ה. בפרטת אם u ו-v
ניתנים להגעה הדדית ב-G, אז גם ב-G^T.

אלגוריתם למציאת רכיבים קשירים היטב

Strongly Connected Comptonents (G)

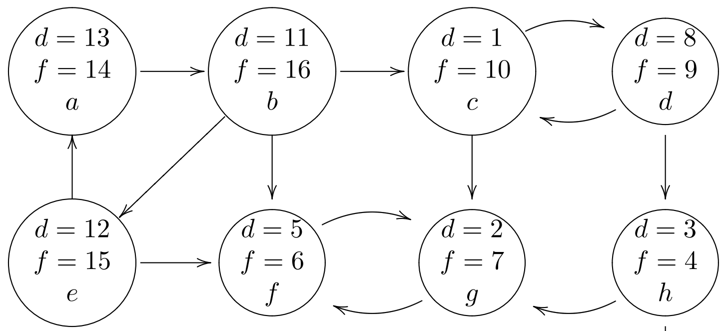
1.Call DFS to compute f[u] for all $u \in V$

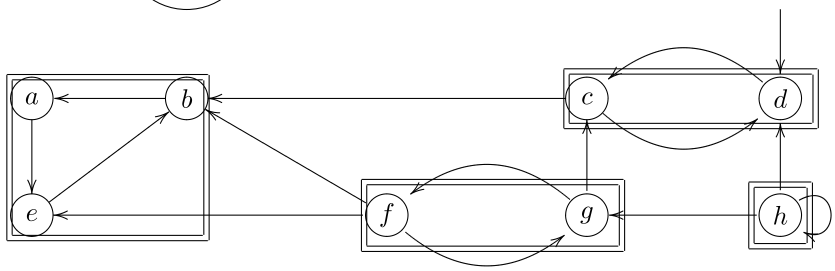
2.Compute G^T

3.Call DFS(G^T) but in the main loop consider the vertices
in decreasing order of $f[u]$ (as computed in 1.)

4.Output the vertices in each DFS tree generated in 3.
as a seperate component.

דוגמת הרצה:



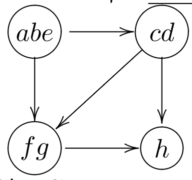


זמן הריצה של אלג' SCC.

זמן הריצה נקבע ע"י:

1. הרצה של DFS $\rightarrow \Theta(|V|+|E|) \Rightarrow$ (סה"כ זמן ריצה)
2. יצירת G^T : $O(|E|)$.
3. סריקת הצמתים בלולאה המרכזית בצעד 3. בסדר יורד לפי $f[u]$.
$\Leftarrow$ניתן לשמור את ערכי $f[u]$ בצעד 1. במחסנית.

לצורכך הוכחת הנכונות של SCC נגדיר את
גרף הרכיבים של G, $G^* = (V^*, E^*)$:
נניח כי הרק"ה ב-G הם $C_1, C_2, \dots, C_k$.
אזי $V^* = \{v_1, v_2, \dots, v_k\}$ והוא כולל צומת v_i לכל
רק"ה C_i ב-G.
תהיה קשת (v_i, v_j) ב-G^* אם G מכיל קשת מאיזשהו צומת
$x \in C_i$ ל- $y \in C_j$.
בדוגמה: גרף הרכיבים הוא:



למה 1: יהיו C ו-C' שני רק"ה בגרף מכוון $G = (V, E)$.
ויהיו $v, u \in C$ ו-$v', u' \in C'$ נניח שיש מסלול מכוון $u \to u'$ ב-G,
אזי לא ייתכן שיש מסלול מכוון מ-$v' \to v$.
(הוכחה בתרגול).
-
כאשר נשתמש ב- $d[u]$ ו- $f[u]$ בניתוח האלג',
נתייחס לערכים אלו כפי שנקבעו ל-u בקריאה הראשונה
ל-DFS (בצעד 1. של האלג').
-נרחיב את ההגדרות של זמן "גילוי" ו"נסיגה" לקבוצות
צמתים אם $U \subseteq V$ אזי $d(U) = \min_{u \in U}\{d[u]\}$
ו- $f(U) = \max_{u \in U}\{f[u]\}$
דהיינו $d(U)$ ו-$f(U)$ הם זמן הגילוי המוקדם
וזמן הנסיגה המאוחר בקבוצה U, בהתאמה.

למה 2:

יהיו C ו- C' שני רק"ה בגרף מכוון $G = (V, E)$.
נניח שיש קשת $(u, v) \in E$ כאשר $u \in C$
ו-$v \in C'$, אזי $f(C) > f(C')$.

הוכחה:

נטפל בשני מקרים כתלות ביחס
בין $d(C)$ ו- $d(C')$

1. $d(C) < d(C')$ ויהי x הצומת הראשון שהתגלה ב- C.
אזי בזמן $d[x]$ כל הצמתים ב-C ו-C' לבנים.
היות ו- $(u, v) \in E$ יש מסלול שמורכב רק מצמתים לבנים מ-x
לכל צומת ב-C וגם מ-x לכל צומת ב-C'.
ממשפט המסלול הלבן, כל הצמתים ב-C וב-C' יהפכו צאצאים של x בעץ DFS.
ממשפט הסוגריים, לכל צומת w ב-C או ב-C' מתקיים:
$d[x] < d[w] < f[w] < f[x]$ ולכן $f[x] = f[C] > f(C')$.

2. אם $d(C) > d(C')$, אזי יהי y הצומת הראשון שהתגלה ב-C' בזמן $d[y]$
יש מסלול לבן מ-y לכל צומת ב- C'. לכן, ממשפט המסלול הלבן כל הצמתים ב-C'
יהפכו לצאצאים של y בעץ DFS, וממפש הסוגריים $f[y] = f(C')$.
בזמן הגילוי של y כל הצמתים ב-C לבנים. היות ויש קשת (u, v) מ-C ל-C'.
מלמה 1 לא ייתכן שיש מסלול מ-C' ל-C, לכן לא ניתן להגיע מ-y לאף צומת ב-C.
לכן, בזמן $f[y]$ כל הצמתים ב-C עדיין לבנים.
מכאן נקבל כי לכל $w \in C$, $f[w] > f[y]$.
לכן $f(C) > f(C')$
■.

מסקנה 3:

יהיו C ו-C' רק"ה בגרף מכוון $G = (V, E)$
נניח שיש קשת $(u, v) \in E^T$, כאשר $u \in C$ ו- $v \in C'$
אזי בהרצה של DFS על G (בצעד 1 של SCC).
נקבל כי $f(C) < f(C')$.

הוכחה:

היות ו-$(u, v) \in E^T$, יש ב- G קשת $(v, u) \in E$ מלמה 2,
היות ויש קשת מ-C' ל-C ב-G.
נקבל כי $f(C) < f(C')$
■.

משפט 4:

האלגוריתם Strongly Connected Components מחשב נכון את הרק"ה בגרף מכוון G.

הוכחה:

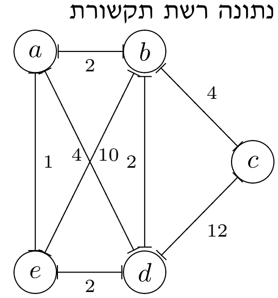
נוכיח באינדוקציה שכ"א מ-x העצים הראשונים שנמצאו בהרצת DFS על G^T (בצעד 3. של SCC) הוא רק"ה.
בסיס:
$k = 0$, נכון באופן ריק.
צעד האינדוקציה:
נניח שהטענה נכונה ל-k העצים הראשונים ונוכיח לעץ ה-$(k+1)$ נניח כי השורש של העץ הינו צומת u,
כאשר u שייך לרק"ה C ב-G.
- מאופן התקדמות האלג' בלולאה המרכזית של צעד 3. מתקיים:
$f[u] = f(C) > f(C')$
לכל רק"ה C' ב-G שעוד לא טופל (בצעד 3).
-מהנחת האינדוקציה, בזמן ש-DFS "מגלה" את u (בצעד 3.)
כל הצמתים ב-C לבנים ממשפט המסלול הלבן,
כל הצמתים ב-C יהפכו לצאצאים של u בעץ DFS.
בנוסף ממסקנה 3. כל קשת שיוצאת מ-C ב-G^T מובילה לרק"ה שכבר ביקרנו.
לכן הצאצאים של u בעץ DFS הם כל הצמתים ב-C ורק הצמתים ב-C.
■.

הרצאה 4 אלגוריתמים

בעיות אופטימיזציה ברשתות

דוגמה:

נתונה רשת תקשורת



- ⋆ על כל קשת מופיע מחיר השימוש בה
- ⋆ נניח כי הצומת a מעוניין להפיץ הודעה לכל הצמתים במחיר מינימלי

$\Leftarrow$ יש למצא תת-קב' קשתות ברשת שעליהן ההודעה תעבור כך שתגיע לכל הצמתים

- ⋆ נשים לב כי היות ונרצה להשיג מחיר מינימלי תת-הגרף שהתקבל מבחירת הקשתות יהיה חסר מעגלים

מכאן נקבל את בעיית עץ פורש מינימום:

- ⋆ נתון גרף קשיר לא מכוון $G = (V, E)$, שבו לכל קשת (v, u) יש משקל $w(v, u)$ יש למצוא
  עץ פורש של הגרף שסה"כ משקל הקשתות בו מינימלי.

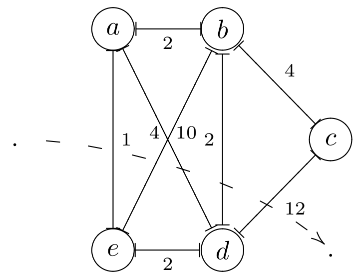
אלגוריתמים לבעיות עץ חיפוש מינימום('עפ"מ')

- ⋆ נראה תחילה אלג' גנרי שבונה עפ"מ קשת אחר קשת. על-ידי הוספת קשתות עם מקשל נמוך והשמטת קשתות עם משקל גבוה.
- ⋆ האלגוריתם יתקדם על-ידי צביעת קשתות. קשתות שיצבעו בכחול יופיעו בעץ, וקשתות שיצבעו בצבע אדום יושמטו.
- ⋆ האלג' יקיים בכל שלב את:
  "שמורת הצבע" - קיים עפ"מ שמכיל את כל הקשתות הכחולות ואף אחת מהקשתות האדומות.

⋆ משמורת הצבע נובע כי כאשר כל הקשתות ב-G נצבעו.
הקשתות הכחולות יוצרות עפ"מ.

חתך(cut) בגרף $G=(V,E)$ הוא חלוקה של קב' הצמתים v לשתי תת-קב' x ו-$\overline{x}=v\backslash x$ קשת חוצה את החתך אם קצה אחד שלה ב-x והקצה האחר ב-$\overline{x}$.

⋆ לעיתים נתייחס לחתך כאל "קבוצת הקשתות החוצות".



אלגוריתם גנרי למציאת עפ"מ

הכלל הכחול: מצא חתך שאין בו קשת כחולה. צבע בכחול את את הקשת הקלה ביותר שאינה צבועה.

הכלל האדום: מצא מעגל שאין בו קשת אדומה. צבע באדום את הקשת הכבדה ביותר שאינה צבועה.

אלגוריתם חמדן: הפעל את הכללים לעיל עד שכל הקשתות צבועות.
הקשתות הכחולות הן עץ פורש מינימלי.

נוכיח את נכונות האלג' הגנרי

משפט 1: האלגוריתם הגנרי צובע את כל הקשתות של גרף קשיר G ומקיים את שמורת הצבע.

הוכחה: נראה תחילה כי האלג' מקיים את השמורה,
באינדוקציה על מס' האיטרציות
(דהיינו הפעלות של הכלל הכחול או האדום).

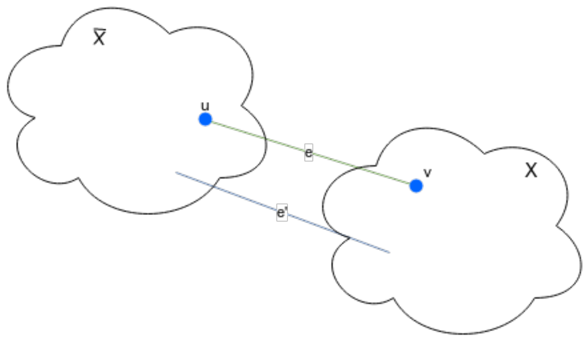
⋆ בסיס: בתחלה כל הקשתות אינן צבועות ולכן כל עפ"מ ל-G מקיים את השמורה (באופן ריק).

⋆ צעד האינדוקצייה: נטפל לחוד בשני מקרים:

1. נניח כי השמורה מתקיימת לפני הפעלה של הכלל הכחול.
תהיי e קשת שנצבית כעת בכחול, ויהיה T עפ"מ שמקיים את השמורה לפני שהקשת e נצבעה.

(א) אם $e\in T$ אזי T מקיים את השמורה אחריי שהקשת e נצבעה (סיימנו).

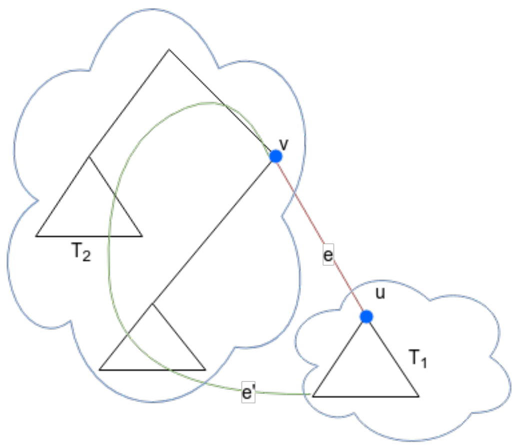
(ב) אם $e \notin T$ אזי נסתכל על החתך $x, \overline{x}$ שעליו הפעלנו את הכלל הכחול.
יש מסלול ב-T שמחבר את הצמתים u, v בקצוות של e.



- היות והקשת e חוצה את החתך, קיימת על המסלול הנ"ל
  קשת אחרת e', שחוצה את החתך ו-$e' \in T$.
- הקשת e' לבטח לא צבועה בכחול (וגם אינה צבועה באדום מהנחת האינדוקציה).
- בנוסף $w(e') \geq w(e)$
- $\Leftarrow$ ניתן להשמיט את e' מ-T ולהוסיף במקומה את e.
  נשים לב , כי אם המסלול היחדי בין שני צמתי ב-T עבר דרך e', המסלול יעבור כעת דרך e.
  בנוסף, T נשאר עפ"מ, כי המשקל הכולל לא יכול לעלות.
  - אם נצבע כעת את e בכחול, נקבל כי השמורה מתקיימת כבור העץ T (החדש).

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

- אם $e \in T$ אזי T מקיים את השמורה גם אחרי שהקשת e נצבעת.
- נניח כי $e \in T$ אזי השמטת e מ- T מחלקת את T לשני עצים
  וגם מגדירה חלוקה של הצמתים ב-G



•

⋆ לקשת e יש קצה אחד ב-T_1 וקצה אחר ב-T_2, נסמנם ב-u ו-v בהצאמה המעגל שעליו שפעלנו את הכלל האדום מכיל
מסלול נוסף מ-u ל-v על המעגל יש קשת e' שקצה אחד שלה ב-T_1 והקצה האחר ב-T_2.

- מהשמורה נובע כי e' לא כחולה (כי $e' \notin T$), ומהכלל האדום e' לא צבוע באדום.
- בסוף, $w(e') \leq w(e)$.

⋆ ⇐ הוספת הקשת e' לעץ T והשמטת הקשת e יוצרת עץ חדש מקיים את השמורה אחרי שהקשת e נצבעת
באדום. בנוסף לא הגדלנו את משקל העץ, לכן T (החדש) הינו עפ"מ.

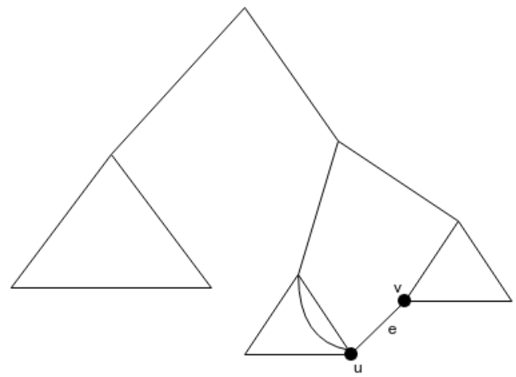
חלק שני של ההוכחה:

נראה כעת כי האלג, הגנרי יצבע את כל הקשתות בגרף. נניח בשלילה כי קיימת קשת e לא צבועה, אך לא ניתן
להפעיל אף אחד מהכללים.

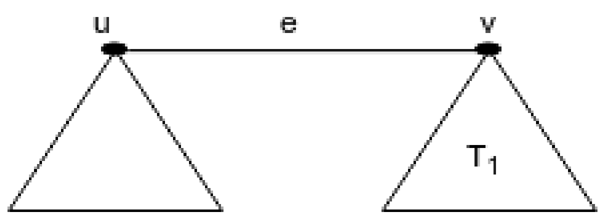
⋆ נשים לב כי מהכלל הכחול, הקשתות הכחולות יוצרות יער של עצים כחולים.

נבחין בין שני מקרים:

1. שני הקצוות של e באותו עץ כחול, אז נקבל:
כלומר, מצאנו ב-G מעגל שאין בו קשתות אדומות
לכן ניתן להפעיל על e את הכלל האדום.



2. אם הקצוות של e בעצים כחולים שונים, נסמן ב-X את אוסף הצמתים ב-T_2 וב-$\overline{X}$ את שאר הצמתים בגרף. קיבלנו חתך שאין בו קשתות כחולות $\Leftarrow$ ניתן להפעיל את הכלל הכחול על e.



מכאן נקבל שכל עוד יש ב-G קשת לא צבועה מובטח שניתן להפעיל את אחד הכללים, לכן האלג' הגנרי צובע את כל הקשתות.

אלגוריתמים קלאסיים למציאת עפ"מ

אלגוריתם Prim: נפעיל את הכלל הכחול על החתך שמוגדר ע"י העץ שהולך ונבנה, כלומר בין צמתי העץ ושאר צמתי הגרף.

1.Init: All the edges are non-colored, $T := \{r\}$
while $T \neq V$ do:
  $e = (u, v)$ minimal edge in the cut $(T, V \setminus T) : u \in T$
  $e \leftarrow$ Blue
  $T := T \cup \{v\}$

זמן הריצה של Prim:

<u>מימוש ע"י מערכים:</u> יהיה v צומת שגובל בעץ כחול T. דהיינו, יש קשת (v, u) כך ש-$u \in T$.

⋆ עבור כל צומת v שגובל ב-T נגדיר קשת בצבע <u>תכלת</u>.
זו היא הקשת "הקלה" ביותר מבין הקשתות שמחברות את v ל-T

(הקשתות בצבע תכלת "מועמדות להיות כחולות).
כאשר נפעיל את הכלל הכחול נבחר קשת בצבע תכלת ונהפוך אותה לכחולה.

⋆ נניח כי הצומת v נוסף לעץ T. נסתכל על כל הקשתות (שאינן צבועות) (v,x) כך ש-$x \notin T$. אם אין ל-x קשת תכלת, נצבע את (v,x) בתכלת. אחרת, קיימת ל-x קשת תכלת ונסמנה (x,y) $(y \in T)$ ו- $w(v,x) < w(x,y)$, אזי נהפוך את (v,x) לתכלת במקום הקשת (x,y).

סיבוכיות:
בכל איטרציה של הכלל הכחול הסיבוכיות היא $O(|V|)$ וכיוון שנעשה $|V|-1$ איטרציות נקבל $O(|V^2|)$ האלגוריתם יתחזק מערך בגודל $|V|$ של קשתות בצבע תכלת.

הרצאה 5

אלגוריתם Prim המשך:

תזכורת - האלגוריתם:

$1.T \leftarrow \emptyset,\ S \leftarrow \{s\}$

$2.$**while**$S \neq V$:

$2.1\ e=(u,v)$ **e is the lightest edge crossing** $(S,\overline{S})$ **assuming** $u \in S, v \notin S$

$2.2\ T \leftarrow T \cup \{e\}, \quad S \leftarrow S \cup \{v\}$

$3.$**Return** T

כיצד נממש את האלגוריתם Prim?

נשמור את צמתי S בערימת מינימום. הערך והמפתח של צומת v בערימה יהיה משקל הקשת הקלה ביותר שנוגעת בו ומחברת אותו ל-S (אם לא קיימת קשת כזו ערך המפתח יהיה ∞).

אתחול:

המפתח של S יהיה O.

המפתח של כל צומת אחר יהיה ∞.

צעד:

ברגע שמוציאים את צומת u מהערימה, מעדכנים את המפתחות של השכנים שלו שעדיין בערימה.

סיבוכיות:

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

האלגוריתם של Kruskal:

1. מיין את הקשתות: $w(e_1) \leq w(e_2) \leq \cdots \leq w(e_m)$
   $T \leftarrow \emptyset$
2. מ-$i=1$ עד m:
   אם $T \cup \{e_i\}$ לא מכיל מעגלים אזי $T \leftarrow T \cup \{e_i\}$.
3. החזר את T.

משפט:

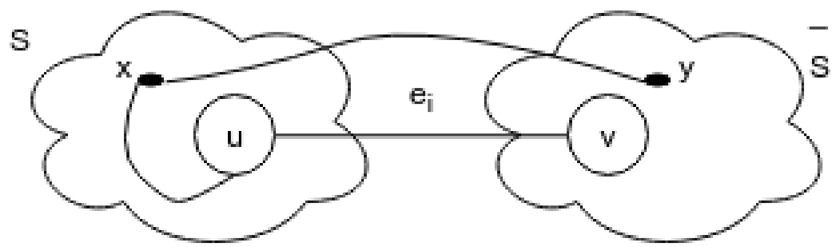
האלגוריתם של Kruskal מחזיר עץ פורש מינימום.

הוכחה:

נראה שהאלגוריתם של Kruskal הוא מימוש מסוים של השיטה הכללית. <u>כלומר</u> אם נצבע בכחול כל קשת שהאלגוריתם מוסיף ל-T ונצבע באדום כל קשת שהאלגוריתם לא מוסיף ל-T, נקבל הפעלה חוקית של הכללים.
ממחלק לשני מקרים:

1. ברגע שהאלגוריתם בוחן את $T \cup \{e_i\}$ מתקיים: $T \cup \{e_i\}$ מכיל מעגל C. נשים לב שבמעגל C כל הקשתות שייכות ל-T (פרט ל-e_i), כלומר כולן צבועות בכחול.
לפי הכלל האדום, ניתן לצבוע באדום את הקשת הכבדה ביותר ב-C מבין הקשתות הלא-צבועות. e_i היא הקשת היחידה ב-C שאינה צבועה ולכן צביעתה באדום היא הפעלה חוקים של הכלל האדום.

2. ברגע שהאלגוריתם בוחן את הקשת e_i מתקיים: ב-$T \cup \{e_i\}$ לא נסגרת מעגל



נתבונן בתחום: $S = \{x | \text{there is a path from } x \text{ to } u \text{ in } T\}$

(א) $u \in S$ (מההגדרה).

(ב) $v \notin S$ נראה זאת: נניח השלילה כי $v \in S$
$\Leftarrow$ יש ב-T מסלול (כחול) בין u ו-v (הגדרת S)
$\Leftarrow$ $T \cup \{e_i\}$ מכיל מעגל וזו סתירה.

(ג) למה אין קשת כחולה שחוצה את S?
נניח בשילה שיש קשת כחולה (x,y) כך שמתקיים $x \in S, y \notin S$
$x \in S \Leftarrow$ יש מסלול כחול בין u ו-x. נשרשר למסלול זה את הקשת (x,y) וקבלנו מסלול כחול בין u ו-y וזו סתירה לכך ש-$y \notin S$.
<u>שאלה:</u> מדוע אין אף קשת e שחוצה את S, אינה צבועה וגם $w(e) < w(e_i)$?
אם זה קורה, eהייתה צריכה להיצבע באיטרציה קודמת וזו סתירה.
$\Leftarrow$ זו הפעלה חוקית של הכלל הכחול. ■

כיצד ניתן לממש את האלגוריתם ?

הרעיון: הכל איטרציה נרצה לשמור את רכיבי הקשירות של (V,T).
נשתמש ב-union find.
הפעולות שצריך בכל איטרציה:

1. לבדוק האם שני רכיבי קשירות באותה קבוצה.

2. אולי לאחד שני רכיבי קשירות.

$\Leftarrow$ סה"כ סיבוכיות: $O(|E| \log |V|)$.

מסלולים קלים ביותר

נתון:

גרף מכוון $G=(V,E)$, פונקציית משקל $W : E \to \mathbb{R}$

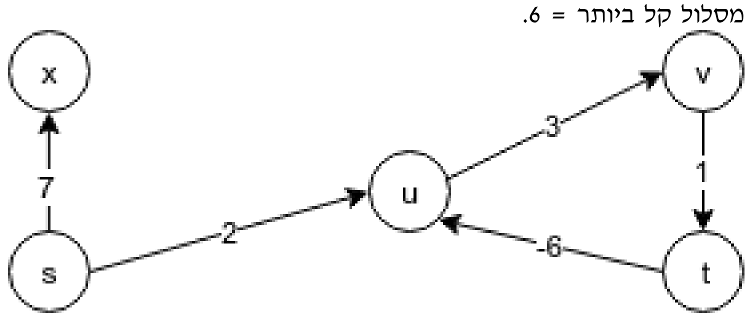
<u>משקל</u> של מסלול p מוגדר להיות סך משקלי הקשתות ב-p: $[w(p) \triangleq \sum\limits_{e\in p} w(e)]$

מטרה:

בהינתן שני צמתים s ו-t, מהו המסלול <u>הקל ביותר</u> מ-s ל-t?

דוגמה:

מסלול קל ביותר = 6.



⋆ אם יש מעגל שלילי בגרף (מעגל שסכום משקלי הקשתות בו קטן ממש מאפס, אזי מרחקים קלים ביותר לא בהכרח מוגדרים).

⋆ <u>הערה:</u> המקרה ש- $w(e)=1 \qquad \forall e \in E$ נפתר ע"י BFS.

הבחנה:

אם p מסלול קל ביותר מ-u ל-v, כל תת-מסלול של p גם הוא קל ביותר.

הוכחה:

נסתכל על p:

$$p:\ u=u_0 \xrightarrow{e_1} u_1 \xrightarrow{e_2} u_2 \to \cdots \to u_{k-1} \to u_k = v$$

נתבונן בתת-המסלול מ-u_i ל-u_j $(i<j)$.

מתקיים: $u \xrightarrow{p'} u_i \xrightarrow{p_{ij}} u_j \xrightarrow{p''} v$

$$w(p)=w(p')+w(p_{ij})+w(p'')$$

נניח בשלילה ש-p_{ij}אינו קל ביותר.

$\Leftarrow$ יש מסלול q מ-u_i ל-u_j כך ש-$w(q)<w(p_{ij})$

נבנה מסלול חדש מ-u ל-v באופן הבא: $u \xrightarrow{p'} u_i \xrightarrow{q} u_j \xrightarrow{p''} v$

שאורכו: $w(p)=w(p')+w(q)+w(p'')$

וזו סתירה ■

נסמן ב- $\delta(u,v)$ את אורך המסלול הקל ביותר מ-u ל-v:

$$\delta(u,v)=\begin{cases} \infty & v \text{ is not reachable from } u \\ -\infty & \text{"negative" circle reachable from } u \\ & \text{and v reachable from the circle} \\ \min\{w(p) : p\text{=path from } u \text{ to } v\} & \text{otherwise} \end{cases}$$

הרצאה 6 אלגוריתמים

תזכורת: מסלולים קלים ביותר

<u>נתון:</u>

- $\star$ גרף מכוון $G=(V,E)$
- $\star$ פונ' משקל $W:E\to\mathbb{R}$:
- $\star$ צומת $s\in V$

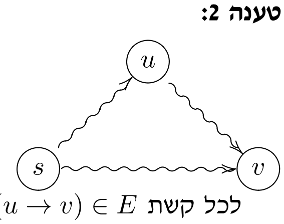
<u>מטרה:</u>
למצוא לכל $v\in V$ מסלול קל ביותר מ-s ל-v.
"אורך" נמדד ע"י סכום משקלי הקשתות שבמסלול.
מסמנים ב-$\delta(u,v)$ את אורך המסלול הקשל ביותר ב-G מ-u ל-v.

$$\delta(u,v)=\begin{cases}\infty & v\text{-non reachable from } u\\ -\infty & \text{there is "negative circle"} \\ & \text{reachable from u and} \\ & \text{v is reachable from the circle} \\ min\{w(p):\text{p-path from } u \text{to } v\} & \text{otherwise}\end{cases}$$

טענה 1:

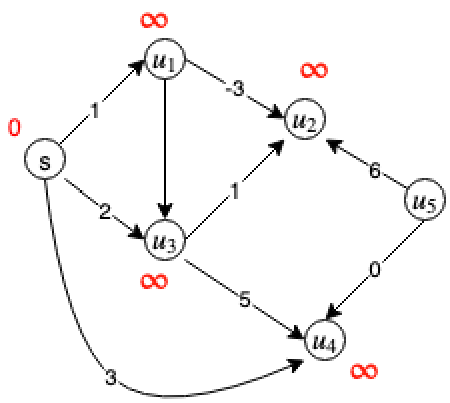
אם p מסלול קל ביותר מ-u ל-v, אז כל תת-מסלול שלו הוא קל ביותר.

טענה 2:



לכל קשת $(u\to v)\in E$ מתקיים:
$\delta(s,v)\le\delta(s,u)+w(u\to v)$

הוכחה:



אם אין מסלול מ-s ל-u אז $\delta(s,u)=\infty$ וסיימנו.
אחרת, נסתכל על המסלול קל ביותר מ-s ל-u, ונשרשר לו את הקשת $(u \to v)$.
$\Leftarrow$ אחרת המסלול החדש הוא $\delta(s,u)+w(u \to v)$ ולכן אי-השוויון מתקיים.■

השיטה הגנרית:

1. אתחול: $d(s) \leftarrow 0$, ולכל $u \neq s$: $d(u) \leftarrow \infty$.
2. כל עוד קיימת קשת $(u \to v) \in E$ כך ש:
$d(v) > d(u) + w(u \to v)$:
$d(v) \leftarrow d(u) + w(u \to v)$.

משפט:

אם אין מעגלים שליליים בגרף, אז:

1. לכל צומת $v \in V$ ולכל שלב בריצת השיטה הגנרית : $d(v) \geq \delta(s,v)$
2. כשה השיטה הגנרית עוצרת, אז:
$d(v) = \delta(s,v) \quad \forall v \in V$

הוכחה:

1. נוכיח באינדוקציה על הצעדים של השיטה הגנרית.

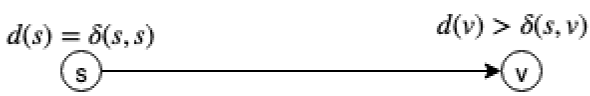
⋆ בסיס:
באתחול $d(u)$ לכל $u \neq s$ שווה ל-∞ ועבור s מתקיים $\delta(s,s) \underbrace{=}_{\text{there is no negative circles in G}} d(s)=0$

⋆ צעד:
בהינתן קשת $(u \to v) \in E$ שהפרה את אי-שוויון המשלולש ביחס לסימונים d,

ביצענו: $d(v) \leftarrow d(u) + w(u \to v)$.

$$
\begin{aligned}
& d(v) = d(u) + w(u \to v) \\
& \underbrace{\geq}_{\text{induction on d(u)}} \delta(s,u) + w(u \to v) \underbrace{\geq}_{\text{triangle inequality}} \delta(s,v)
\end{aligned}
$$

2. מ-א, מספיק להראות שהשיטה הגנרית עוצרת שלכל $v \in V$:
$d(v) \leq \delta(s,v)$



נניח בשלילה שקיים צומת v עבורו:

$$d(v) > \delta(s,v)$$

$\Leftarrow$ $\delta(s,v)$ אינו יכול להיות ∞, ובגלל שאין מעגלים שליליים, $\delta(s,v)$ סופי.
$\Leftarrow$יהיה p מסלול קל ביותר כלשהו מ-s ל-v

$$p = v_1 \to v_2 \to \cdots \to v_{k-1} \to v_k = v$$

עבור s: $d(s) = \delta(s,s)$ (כי אין מעגלים שליליי).
עבור v: $d(v) > \delta(s,v)$
ולא קיים אף צומת v_iבמסלול כך ש-$d(v_i) < \delta(s,v_i)$ (בגלל א).
$\Leftarrow$קיימת קשת ב-p $(v_i \to v_{i+1})$ כך ש:

$$\begin{cases} d(v_i) = \delta(s,v_i) \\ d(v_{i+1}) > \delta(s,v_{i+1}) \end{cases}$$

(וזו הקשת הראשונה שבה זה קורה).

$$
\begin{aligned}
& d(v_{i+1}) > \delta(s,v_{i+1}) \underbrace{=}_{\text{1.sub-path of lightest path is lightest.}} \delta(s,v_i) + w(v_i \to v_{i+1}) \\
& = d(v_i) + w(v_i \to v_{i+1})
\end{aligned}
$$

$\Leftarrow$הקשת $(v_i \to v_{i+1})$ מפרה את אי-שוויון המשולש ביחס לסימונים d.
$\Leftarrow$השיטה הגנרית לא היתה צריכה לעצורה.
$\blacksquare$.

$$\text{(סכום משקלי הקשתות הריש של } \delta \text{ מ-} s \text{ ל-} v_{i+1}) \underbrace{=}_{(1)} \delta(s, v_{i+1})$$
$$= w(v_0 \to v_1) + w(v_1 \to v_2) + \cdots + w(v_{i-1} \to v_i) + w(v_i + v_{i+1})$$
$$= \text{סכום משקלי הקשתות ברישא של } p \text{ מ-} s \text{ ל-} v_i + w(v_i \to v_{i+1})$$
$$\underbrace{=}_{(1)} \delta(s, v_i) + w(v_i, v_{i+1})$$

שאלה:

כיצד ניתן לשחזר איזשהו מסלול קל ביותר ?

לכל צומת $v \in V$ מסמן ב-$\pi(v)$ את הצומת שגרם לעדכון האחרון של $d(v)$.

- ⋆ אתחול: $\forall v \in V \ \pi(V) \leftarrow NULL$
- ⋆ בעדכון: שהקשת $(u \to v)$ מבצעת עדכון ל-$d(v)$ נצבע: $\pi(v) \leftarrow u$

הגדרה:

<u>עץ מסלולים קלים ביותר</u> של G ו-s הוא תת-גרף $G' = (V', E')$ של G כך ש:

1. V' זה אוסף הצמתים הישיגיים מ-s.
2. G' הוא מכוון ששורשו s.
3. לכל $v \in V'$ המסלול היחיד ב-G' מ-s ל-v הוא מסלול קל ביותר מ-s ל-v ב-G.

טענה:

אם אין מעגלים שליליים, אז שהשיטה הגנרית עוצרת, נסתכל על הגרף הבא:

$$G' = (V', E')$$
$$V' = \{v : \pi(v) \neq NULL\} \cup \{s\}$$
$$E' = \{(\pi(v) \to v) : \pi(v) \neq NULL)\}$$

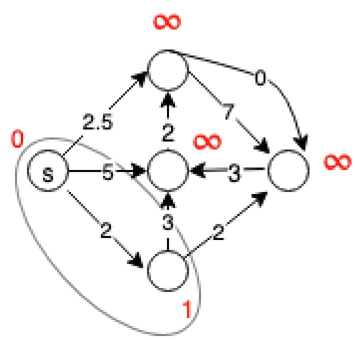
אז G' הוא עץ מסלולים קלים ביותר של G מ-s.

"הוכחה":

<u>שלבים להוכחה(לא נוכיח):</u>

- ⋆ בכל שלב בריצה של השיטה הגנרית, G' חסר מעגלים מכוונים.
- ⋆ בסיום הריצה, V' זה אוסף הצמתים הישיגים מ-s.
- ⋆ בסיום הריצה, המסלול היחיד ב-G' מ-s ל-v הוא מסלול קל ביותר ב-G מ-s ל-v.

נתמקד במקרה שמשקלים הם אי-שליליים, כלומר $w(u \to v) \geq 0 \quad \forall (u \to v) \in E$.
דוגמה:



האלג' של Dijkestra (1959)

1. אתחול:
$d(s) = 0$, ולכל $u \neq s$ $d(u) \leq \infty$, $Q \leftarrow V$.

2. כל עוד $Q \neq \emptyset$:

   (א) יהי u הצומת בעל d הקטן ביותר ב-Q.

   (ב) לכל קשת $(u \to v) \in E$, אם $d(v) > d(u) + w(u \to v)$ אז:
   $d(v) \leftarrow d(u) + w(u \to v)$

   (ג) הוצא את u מ-Q.

זמן ריצה:

למשל, אם מממשים את Q בעזרת ערימת מינימום זמן הריצה הכולל: $O(|E| \log |V|)$

נכונות:

מנכונות השיטה הגנרית, מספיק שנראה שהאלג' של Dijkestra עוצר
וכל הקשתות $(u \to v) \in E$ מקיימות:

$$d(v) \leq d(u) + w(u \to v)$$

טענה 1:

נניח באיטרציה מסויימת u יצא מ-Q, ובאיטרציה העוקבת לה v יצא מ-Q. אז:

- $\star$ $d(u)$ ברגע ההוצאת u מ-Q.
- $\star$ $d(v)$ ברגע ההוצאת v מ-Q.

ומתקיים:

$$d(u) \leq d(v)$$

הוכחה:

ברגע הוצאת u מ-Q מתקיים: $d(u) \leq d(v)$ (מאופן בחירת הצומת שיוצא מ-Q).

- $\star$ אם $d(v)$ לא התעדכן במהלך האיטרציה ש-u יצא מ-Q סיימנו.
- $\star$ אחרת יש קשת $(u \to v) \in E$ וביצענו עדכון $d(v) \leftarrow d(u) + \underbrace{w(u \to v)}_{\geq 0}$ הטענה נובעת מאי-שליליות המשקלים.

$\blacksquare$.

מסקנה 1:

טענה 1 נכונה גם אם v יצא אחרי u אבל לא בהכרח באיטרציה העוקבת.

מסקנה 2:

לאחר הוצאת v מ-Q, $d(v)$ לא מתעדכן.

הוכחה:

נניח בשלילה שקיים צומת v עבורו $d(v)$ מתעדכן אחרי ש-v יצא מ-Q.
נסתכל על הפעם הראשונה שזה קרה.
$\Leftarrow$ קיים צומת u, שכרגע יוצא מ-Q
וקשת $(u \to v) \in E$ כך ש-$d(v) > \underbrace{d(u)}_{\text{when we get it from Q}} + w(u \to v)$
(ואז ברגע הוצאת v מ-Q בגלל שזו הפעם <u>הראשונה</u>).
מאי-שליליות המשקים קיבלנו ש-$d(v)$ ברגע הוצאת v מ-Q גדול ממש מ-$d(u)$
ברגע הוצאת u מ-Q וזו <u>סתירה</u> למסקנה הראשונה.$\blacksquare$

הרצאה 7 אלגוריתמים

שאלה:

מה יהיה אלו יהיו משקלים שליליים בגרף?
האלגוריתם של Dijkstra יכול להכשל גם אם תהיה אפילו קשת אחת שלילת בגרף ואין מעגלים שליליים.

הרעיון:

האלגוריתם יתקדם בפאזות. בכל פאזה, נעבור על כל קשתות הגרף בסדר כלשהו, ונבדוק הפרה של אי-שוויון המשולש ביחס ל-d.

אלגוריתם Bellman Ford:

1. אתחול $d(s) \leftarrow 0$ ולכל $v \neq s$ $\quad d(v) \leftarrow \infty$.
2. מבצעים $n-1$* פעמים: $(|V| = n)^*$
   (א) עוברים על כל הקשתות פעם אחת ולכל קשת $(u \to v) \in E$, אם $d(v) > d(u) + w(u \to v)$
   אז $d(v) \leftarrow d(u) + w(u \to v)$

זמן ריצה: $O(|V| \cdot |E|)$

טענה:

אם קיים מסלול קל ביותר מ-s ל-v שמכיל k קשתות, אז בסיום הפאזה ה-k: $d(v) = \delta(s, v)$.

הערה:

נכונות האלגוריתם נובעת מהטענה ומהנכונות של השיטה הגנרית שאומרת שבכל שלב של ריצת השיטה הגנרית אין פספו כלפי מטה, כלומר $d(v) \geq \delta(s, v)$.

הוכחת הטענה:

נוכיח באינדוקציה על k.

* בסיס: $k = 0$
  רק עבור s יש מסלול קל ביותר מ-s ל-s שמכיל אפס קשתות (כשה אין מעגלים שליליים) ולכן $d(s) = \delta(s, s) = 0$.

* צעד:
נניח ש ל-v יש מסלול קל ביותר מ-s ל-v המכיל $k+1$ קשתות. נסמנו:

$$p = \underbrace{v_0}_{s} \rightarrow v_1 \rightarrow \ldots \rightarrow v_k \rightarrow \underbrace{v_{k+1}}_{v}$$

⇐ הרישא של p מ-s ל-v_k מסלול קל ביותר המכיל k קשתות.
⇐ לפי הנ"א על v_k, בסיום הפאזה על v_Kמובטח ש-$d(v_k) = \delta(s, v_k)$
במהלך הפאזה ה-$k+1$ בוחנים את הקשת $(v_k \rightarrow v_{k+1})$ ואז מובטח שבסיום הפאזה ה-$k+1$:

$$d(v_{k+1}) \leq \underbrace{d(v_k) + w(v_k \rightarrow v_{k+1})}_{\text{p-length}} = \delta(s, v_{k+1})$$

לפי התכונה של השיטה הגנרית, נקבל בהכרח ש- $d(v_{k+1}) \leq \delta(s, v_{k+1})$ בסיום הפאזה ה-$k+1$.

הגישה החמדנית

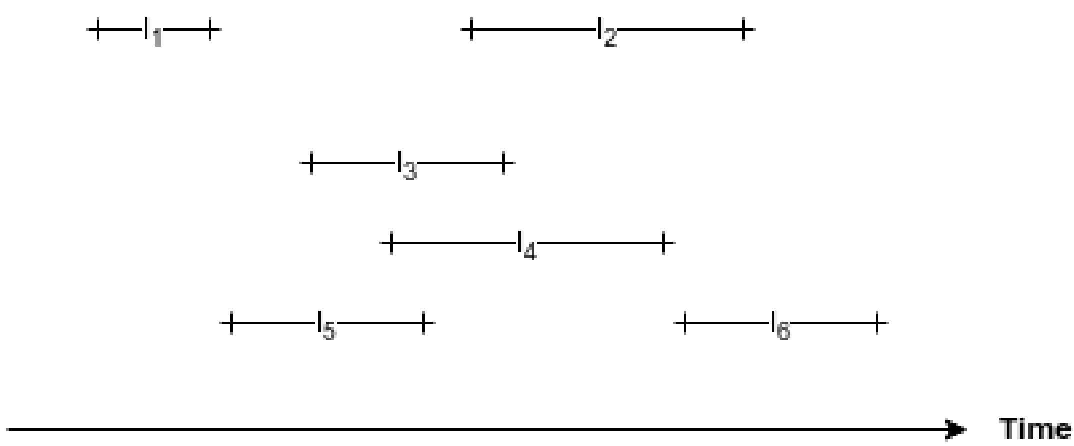
הרעיון:

הצגת גישה לפתרון בעיות בה האלגוריתם בוחר את האפשרות הטובה ביותר כרגע.

דוגמה:

נתונות n משימות, וכל משימה i מיוצגת ע"י זמן התחלה s_i וזמן סיום f_i.
יש מכונה בודדת שיכולה להריץ את המשמות. המטרה היא לבחור אוסף משימות גדול ביותר כך ש
כל שתי משימות שנבחרו לא נחתכות.

לדוגמה:



פתרון אופטימלי: $\{I_1, I_4, I_6\}, \{I_1, I_3, I_6\}, \{I_1, I_5, I_6\}$.

* ניתן לנסח את הכיה ע"י גרפים. גרף אינטרוולים הוא גרף שבו כל צומת מייצג אינטרוול, יש קשת בין שני צמתים $\Leftrightarrow$ האינטרוולים נחתכים.
המטרה היא למצוא קבוצה בלתי-תלויה גדולה ביותר (תת-קבוצה של הצמתים כך שבין כל שתיים בתת-הקבוצה אין קשת).

האלגוריתם:

1. ממיינים את האינטרוולים לפי זמני סיום $f_1 \leq f_2 \leq \ldots \leq f_n$, $X \leftarrow \emptyset$
2. עבור $j = 1$ עד n.
   (א) אם I_jלא נחתך עם אף אינטרוולי ב-X, בצע $X \leftarrow \{I_j\} \bigcup X$.

סיבוכיות: $O(nlog(n))$

טענה:

לכל $k = 0$ עד n, בסיום איטרציה k קיים פתרון אופטימלי X^* כך ש:
$(\forall 1 \leq j \leq k) \quad I_J \in X \Leftrightarrow I_j \in X^*$.

מסקנה:

אם נבחר $k = n$ נקבל שיש פתרון אופטימלי שזהה לפלט האלגוריתם.

הוכחה:

באינקוציה עלk.

* בסיס:
$k = 1$ נשים לב שתמיד $I_1 \in X^*$ (האלג' בוחר את I_1). נבחר X^* להיות פתרון אופטימלי כלשהוא. אם $I_1 \in X^*$ סיימנו. אחרת, $I \notin X^* \Leftarrow$ יהי $I_r \in X^*$ האינטרוול בעל זמן הסיום הקטן ביותר בנחתך עם I_1
(אם אין I_r כזה אזי $X^* \bigcup \{I_1\}$ פתרון חוקי וזו סתירה לאופטימליות של X^*).
נטען ש-$X^* \backslash \{I_r\} \bigcup \{I_1\}$ פתרון חוקי (אם זה נכון סיימנו שכן פתרון זה גם הוא אופטימלי).
מפני ש-f_1 זמן הסיום הקטן ביותר של כל האינטרוולים אזי I_r הוא יחיד
(כלומר אין עוד אינטרוולים ב-X^* הנחתך עם I_1, אחרת X^* לא פתרון חוקי).
$\Leftarrow X^* \backslash \{I_r\} \bigcup \{I_1\}$ פתרון חוקי.

הרצאה 8 אלגוריתמים

המשך הוכחה מהרצאה 7:

<u>**צעד:**</u> הנחת האינדוקצייה היא שיש פתרון אופטימלי כך שלכל $1 \leq j \leq k$, $I_J \in X \Leftrightarrow I_j \in X^*$

נתבונן באינטרוול $k+1$. נחלק לשני מקרים לפי בחירת האלגוריתם ב- I_{k+1}:

1. $I_{k+1} \in X$ אם $I_{k+1} \in X^*$ סיימנו, אחרת $I_{k+1} \notin X^*$ ובהכרח קיים אינטרוול ב-X^*שנחתך עם I_{K+1}.
נבחר את האינטרוול I_n בעל האינדקס המינימלי מ-X^* שנחתך עם I_{k+1}. נשים לב ש-$r \geq k+2$
שכן אלו $r \leq k$ מפני ש-$I_r \in X^*$ נקבל ש-$I_r \in X^*$ (מהנ"א).
זוהי <u>סתירה</u> לכך שהאלגוריתם בחר את I_{k+1}. נשים לב ש-I_n הוא היחיד ב-X^* שנחתך עם I_{k+1}
(אחרת X^* לא אופטימלי).
נסתכל על $X^* \backslash \{I_r\} \cup \{I_{k+1}\}$ ונוכיח שהוא פתרון חוקי, והטענה תנבע מכך ש-$|X^*| = |X^* \backslash \{I_r\} \cup \{I_{k+1}\}|$.
נראה ש-I_{k+1} לא נחתך עם אף אינטרוול ב-$X^* \backslash \{I_r\}$.
   (א) האם I_{k+1} נחתך עם משימות עם אינדקסים $r+1, r+2, \ldots, n$? לא.
   (ב) האם I_{k+1} נחתך עם משימות עם אינדקסים $k+1, k+2, \ldots, r-1$? לא.
   (ג) האם I_{k+1} נחתך עם משימות עם אינדקסים $1, 2, \ldots, k$? לא.
   $\Leftarrow$ $X^* \backslash \{I_r\} \cup \{I_{k+1}\}$ פתרון חוקי.

2. $I_{k+1} \notin X$ לפי הנ"א, X^* לא יכול להכיל את I_{k+1} שכן במקרה כזה ב-X^* היו שני אינטרוולים שנחתכים.

שאלה:

לכלכ אינטרוול I_j נתון רווח $p_j \geq 0$. כיצד ממקסמים את סך הרווחים מהבקשות שמספקים?

קידוד Huffman

בגדול, בהנתן קובץ המורכב מאוסף תווים, רוצים למצוא כיצד "לקודד" אותו כך שאורך הקובץ המקודד
יהיה קצר ככל הניתן. בקידוד כל תו בקובץ יהפוך למחרוזות בינארית (באורך כלשהו).
מילת קוד היא מחרוזת בינארית $w = w_1 w_2 \ldots w_n$ $w_i \in \{0,1\}$
האורך של מילת הקוד w יסומן ע"י $l(w)$.
קוד הוא אוסף של מילות קוד.

דוגמה:

$C = \{ \underset{(a_1)}{C_1}, \underset{(a_2)}{C_2}, \underset{(a_3)}{C_3} \}$ כאשר $C_1 = 01,\ C_2 = 0,\ C_3 = 10$

פעולת הקידוד נעשית ע"י החלפת כל תו במילת הקוד המתאימה לו.

כלומר $a_1a_2a_3 \underbrace{\rightarrowtail}_{\text{(code)}} 01010$

כיצד מפענחים קידוד ?

$a_2a_3 \rightarrow 010 \leftarrow a_1a_2$, $\underbrace{01}_{c_1}\underbrace{0}_{c_2}$ או $\underbrace{0}_{c_2}\underbrace{10}_{c_3}$?

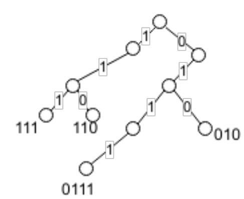
אנו נרצה להגביל את עצמינו לקידוד שיש רק דרך אחת לפענח אותו.
נתמקד בקודים חסרי רישאות, שבהם אין מילת קוד שהיא רישא של מילת קוד אחרת.
עבור קודים חסרי רישאות, הפיענוח פשוט ונעשה ע"י קריאה של הקובץ המקודד.
נתון: n תווים כך שלתו ה-i נתונה תדירות f_i.
מטרה: למצוא קוד שבו התו ה-i מותאם למילת קוד c_i, שיש דרך אחת לפענח אותו
המביאה למינימום $\sum_{i=1}^{n} l(c_i) \cdot f_i$.

טענה (לא להוכחה):

מבין הפתרונות האופטימליים קיים אחד לפחות שהוא קוד חסר רישאויות.

שאלה:

האם קיימת דרך נוחה לייצג קוד חסר רשאיות?
ניתן להציג קוד חסר רישאות ע"י עץ בינארי
(למשל שמאלה זה 1 וימינה 0) כאשר העלים הם מילות הקוד.



טענה 1

קוד אופטימלי מיוצג ע"י עץ מלא (לכל צומת שאינו עלה יש שני ילדים ישירים).

הוכחה בתרגול.

הרעיון:

שני העלים עם התדירות הנמוכה ביותר יהיו עלים אחים עמוקים ביותר בעץ.

האלגוריתם של Huffman (1952)

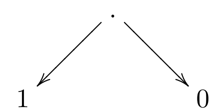
1. נמיין את התווים הנתונים לפי התדירויות: $f_1 \geq f_2 \geq \cdots \geq f_n$

2. נוציא את התווים ה-n וה-$n-1$, ובמקומם נכניס תו "מלאכותי" בעל תדירות $f_{n-1}+f_n$.
נפתור רקורסיבית את הבעיה עבור הקלט המצומצם ונקבל עץ T'.

3. ב-T' נוסיף לעלה שמייצג את התו המלאכותי שני לדים ישירים
כאשר האחד מייצג את התו ה-$n-1$ והשני את התו ה-n.
נסמן ב-T את התו המתקבל.

4. החזרת את T.

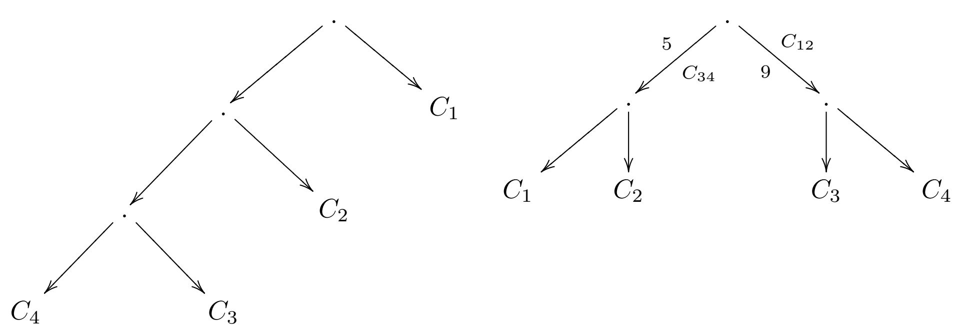
מה תנאי העצירה של האלגוריתם הרקורסיבי? למשל אם $n=2$ מחזירים עץ:



דוגמה:

$$f_1=1, f_2=2, f_3=3, f_4=4, f_5=5$$

$$\{C_1,C_2,C_3,C_4\}$$
$$\{C_1,C_2,C_{34}\}$$
$$\{C_{12},C_{34}\}$$

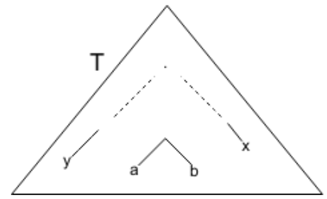


טענה 2

יהיו x ו-y שתי מילות הקוד בעלות התדרים הנמוכים ביותר.
אזי קיים פתרון אופטימלי T שבו x ו-y עלים אחים נמוכים ביותר.

הוכחה:

יהי T עץ אופטימלי: בה"כ, $f_x \le f_y$ וגם $f_a \le f_b$.



ניצור עץ חדש T' בו נחליף בין x ו-a ובין y ו-b.
מהו השינוי בערך של T?
$l_T(x) \cdot f_x + l_r(y) \cdot f_y + l_T(a) \cdot f_a + l_T(b) \cdot f_b$ (תרומת התווים a, b, x, y ל-T.
$l_T(x) \leq l_T(a)$ וגם $f_x \leq f_a$. בופן דומה, $l_t(y) \leq l_t(b)$ וגם $f_y \leq f_b$.
$\Leftarrow$ אם נבצע את ההחלפה, ערכו של T לא יכול לגדול.

מסקנה:

נסמן ב-x וב-y את שני התווים בעלי תדירויות הנמוכות ביותר ונסמן ב-z את התו המלאכותי שהוספנו במקום שניהם.
נסמן ב-$cost(T)$ את ערך העץ T.
נסמן ב-T' את העץ המתקבל מהקריאה הרקורסיבית, אזי:

$$cost(T) = cost(T') - l_T(z) \cdot f_z + (l_{T'}(Z) + 1)(f_x + f_y) = cost(T') + f_x + f_y$$

נכונות האלגוריתם נובעת מטענה 2 והמסקנה (ניתן להוכיח בשלילה).

אלגוריתמים הרצאה 9

תכנון דינאמי

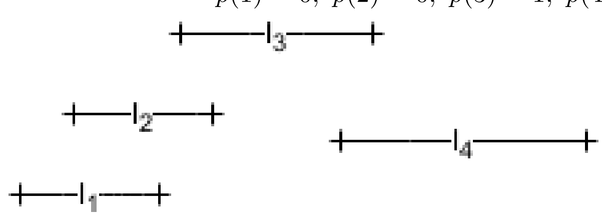
טכניקה המאפשרת לפרק בעיה באופן רקורסיבי לתתי-בעיות קטנות מאותו סוג.

דוגמה 1:

שיבוץ משימות ממושקלות.
נתון: n אנטרוולים, כאשר כל אינטרוול נתון ע"י התחלה s_i וזמן סיום f_i. לאינטרוול ה-i נתון רווח w_i.
מטרה: למצוא אוסף אינטרוולים S כך שכל שני אינטרוולים ב-S לא נחתכים שממקסם $\sum_{j \in S} w_j$.
נזכר שמיינו את האינטרוולים: $f_1 \leq f_2 \leq \cdots \leq f_n$.
עבור אינטרוול I_j, נסמן ב-$p(j)$ את האינדקס הכי גדול כך ש-$p(j) \leq s_j$
(כלומר , אינטרוול $I_{p(j)}$ לא נחתך עם אינטרוול I_j).
אם לא קיים אינטרוול כזה, p_j יוגדר להיות 0.

דוגמה:

$p(1) = 0,\ p(2) = 0,\ p(3) = 1,\ p(4) = 2$



שאלה:

מה הן תתי הבעיות שנוצרו ?
תתי-הביות שיעניינו אותנו יהיו רישאות, כלומר לכל $0 \leq j \leq n$ הראישר ה-j תכיל $\{I_1, I_2, \ldots, I_j\}$

הגדרה:

נסמן ב-$A(j)$ את ערך הפתרון האופטימלי עבור תת-הבעיה ה-j.

שאלה:

האם יש קשר רקורסיבי בין $A(0), A(1), \dots, A(n)$?

$$A(j) = \begin{cases} 0 & j=0 \\ max\{A(j-1), w_j + A(p_j) & 1 \le j \le n \end{cases}$$

טענה:

$A(0), \dots, A(n)$ מקיימות את הנוסחה הרקורסיבית לעיל.

הוכחה:

אם $j=0$ אזי $A(0)$ הוא ערך הפתרון האופטימלי עבור קלט ריק, כלומר $A(0)=0$, כפי שאכן מגדירה הנוסחה הרקורסיבית.
אחרת $1 \le j \le n$. נסמן ב-S_j^* פתרון אופטימלי כלשהו לתת-הבעיה $\{I_1, I_2, \dots I_j\}$.

- $\star$ אם $I_J \notin S_j^*$ אזי S_j^* פתרון אופטימלי לבעיה ה-$j-1$ (שכן אחרת S_j^* לא היה פתרון אופטימלי לתת-הבעיה ה-j).
$\Leftarrow$ $\sum\limits_{i \in S_j^*} w_i = A(j-1)$ (מההגדרה)
$\Leftarrow A(j) = A(j-1)$

האם יתכן כי $\sum\limits_{i \in S_j^*} w_i < w_j + A(p(j))$?
<u>לא</u>, מפני שבמקרה זה היה פתרון עדיף מ-s_j ל-$\{I_1, I_2, \dots, I_j\}$.

- $\star$ אם $I_j \in S_j^*$ אזי $S_j^* \backslash \{I_j\}$ הוא פתרון אופטימלי לתת-הבעיה $p(j)$ (שכן אחרת S_j^* לא היה פתרון אפשרי עבור תת-הבעיה ה-j).
$\Leftarrow$ $\sum\limits_{i \in S_j^*} w_i = w_j + A(p(j))$
$\Leftarrow A_j = w_j + A(p(j))$

האם יתכן כי $A(j-1) > \underbrace{w_j + A(p(j))}_{A_J}$? <u>לא!</u>

$\Leftarrow$ $A(0), \dots, A(n)$ מקיימת את הנוסחה הרקורסיבית.

שאלה:

האם ניתן לחשב את $A(n)$ במהירות בעזרת הנוסחה הרקורסיבית?
<u>הרעיון:</u> בדומה לסדרת פיבונאצ'י, ניתן לשלם בזכרון ולחשב את $A(n)$ בזמן לינארי.

$A =$ (סדר חישוב $\to$)

0			...		
0	1	2	...	$n-1$	n

האלגוריתם:

1. חשב את $p(1),\dots,p(n)$
2. עבור $j-0$ עד n בצע:
   (א) חשב את $A(j)$ לפי הנוסחה הרקורסיבית.
3. החזר את $A(n)$

<u>**(ללא המיון $f_1 \le \dots \le f_n$ זמן הריצה הוא $O(n)$)**</u>

דוגמה 2:

סדר ביצוע כפל מטריצות
<u>נתון:</u> תרגיל כפל מטריצות $A_1,\dots,A_n$ כאשר A_i במימדים $p_{i-1\times p_i}$.
<u>מטרה:</u> למצוא סדר לביצוע התרגיל הממזער את מספר כפלים האריתמטיים.
(הערה: כפל של $A\cdot B$ כשא A במימדים $p\times q$ ו-B במימדים $q\times r$ דורש pqr מכפלות).

דוגמה:

$$A_1(10\times 100), A_2(100\times 5), A_3(5\times 50)$$

$$A_1\cdot A_2\cdot A_3$$

$$(A_1\cdot A_2)\cdot A_3 \qquad\qquad A_1\cdot(A_2\cdot A_3)$$

$$\Downarrow \qquad\qquad \Downarrow$$

$$10*100*5+10*5*50 \qquad\qquad 100*5*50+10*100*50$$

$$\Downarrow \qquad\qquad \Downarrow$$

$$=7500 \qquad\qquad =75000$$

שאלה:

כמה אפשרויות יש לביצוע תרגיל באורך n?
נסמן ב-$p(n)$ את מס' האפשרויות למקם סוגריים בתרגיל באורך n.

$$p(n) \triangleq \begin{cases} 1 & n=1 \\ \sum_{k=1}^{n-1} p(k)\cdot p(k-1) & n\neq 1 \end{cases}$$

הפתרון הוא $p(n)=\underbrace{C_{n-1}}_{\text{Cathalan's number}}$

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

$\Leftarrow$ לא ניתן לעבור באופן יעיל על כל האפשרויות.

נגדיר תת-בעיה לכל תת-סדרה של התרגיל, כלומר לכל $1 \leq i \leq j \leq n$: $A_i, A_{i+1}, \ldots, A_j$

הגדרה

נסמן ב-$M(i,j)$ את המספר הכפלים האריתמטיים ברטן ביותר שצריך על מנת לפתור את $A_i \cdot A_{i+1} \cdots A_j$.

$$M\left(i,j\right) \triangleq \begin{cases} 0 & i=j \\ \underbrace{\min}_{i \leq k \leq j} \{ \underbrace{M(i,j)}_{\text{dimesnsions } p_{i-1} \times p_k} + \underbrace{M(k+1,j)}_{\text{dimesnsions} p_k \times p_j} + p_{i-1} \cdot p_k \cdot p_j \} & i<j \end{cases}$$

טענה

$M(i,j)$ מקיים את נוסחת הרקורסיה.

שאלה:

כיצד לחשב במהירות את נוסחת הרקורסיה ?

$M =$

	1	2	3	$\ldots$	$n-1$	n
1	0					(פלט)?
2		0				
3			0			
$\vdots$				$\ddots$		
$n-1$					0	
n						0

נשים לב שאם נייצג את M כמטריצה מתקיים:

- $\star$ תנאי העצירה נותן את ערכי האלכסון.
- $\star$ מעוניינים רק במשולש העליון של המטריצה.
- $\star$ לפי נוסחת הרקורסיה $M(i,j)$ תלוי בתאי המטריצה משמאלו ומתחתיו.

$\Leftarrow$

קיימים הרבה סדרים שבהם ניתן לחשב את איברי המטריצה M
כך שברגע שמחשבים את $M(i,j)$ כל התאים במטריצה שהוא תלוי בהם כבר חושבו.
בכל סדר חישוב שכזה, חישוב תא $M(i,j)$ לוקח זמן של $O(j-i)$.

$\Leftarrow$ במקרה הגרוע זה $O(n)$.

$\Leftarrow$ זמן הריצה הוא $O(n^3)$.

אלגוריתמים הרצאה 10

תכנון דינאמי - המשך

דוגמה 3:

מסלולים קלים ביותר בין כל זוגות הצמתים בגרף
<u>נתון:</u> גרף מכוון $G=(V,E)$
פונקציית משקל $w:E\to\mathbb{R}$ (נניח שאין מעגלים שליליים)
<u>מטרה:</u> לכל שני צמתים u ו־v, נרצה לחשב את אורך המסלול הקל ביותר ב־G מ־u ל־v.
<u>פתרון פשוט:</u> להריץ Bellman Ford מכל צומת n.
<u>סיבוכיות:</u> $O(|V|^2\cdot|E|)$.

שאלה:

כיצד להגדיר תתי־בעיות?
צמתי הבנים של מסלול p יהיו כל הצמתים ב־p פרט לצומת ההתחלה וצומת הסיום.
נניח מטעני נוחות, שצמתי הגרף הם $\{v_1,\dots,v_n\}$.
נגדיר תת־בעיה: לכל זוג צמתים u ו־v ולכל $k=0,1,\cdots,n$ ונסמן $\delta_k(i,j)$ להיות משקל מסלול
ב־G מ־v_iל־v_J מבין כל המסלולים שצמתי הביניים שלהם לקוחים מתוך $\{v_1,\dots,v_k\}$.

$$D_k(i,j)=\begin{cases}0 & k=0,i=j\\ w_{i,j} & k=0,i\neq j,(v_i,v_j)\in E\\ \infty & k=0,i\neq j,(v_i,v_j)\notin E\\ min\{D_{k-1}(i,j),D_{k-1}(i,k)+D_{k-1}(k,j) & \text{otherwise}\end{cases}$$

טענה:

$D_k(i,j)=\delta_k(i,j)$ לכל $1\le i,j\le n$ ולכל $k\in\{0,1,\dots,n\}$

- ⋆ **שאלה:** כיצד לחשבת את ה־D־ים ? (באופן מהיר)
  חישוב כל מטריצה לוקח $O(n^2)$ זמן
  $\Leftarrow$ סה"כ $O(n^3)$

$$\underbrace{\boxed{\begin{array}{c}D_0(i,j)\\ (nxn)\end{array}}}_{\underbrace{k=0}_{\text{(stop cond.)}}} \Rightarrow \underbrace{\boxed{\begin{array}{c}D_1(i,j)\\ (nxn)\end{array}}}_{k=1} \Rightarrow \cdots \Rightarrow \underbrace{\boxed{\begin{array}{c}D_n(i,j)\\ (nxn)\end{array}}}_{k=n}$$

⋆ **הערה:** האלגוריתם צורך $O(n^2)$ במקום.

אלגוריתם זה נקרא Floyd Warshall.

הוכחה:

באינדוקצייה על k.
<u>בסיס:</u> $k=0$ נובע מידי מהגדרת $\delta_0(i,j)$ ו־$D_0(i,j)$
<u>צעד:</u> נקבע שני צמתים v_i ו־v_j ונזכר ש־$\delta_k(i,j)$ הוא אורך המסלול הקל ביותר מ־v_i ל־v_j ב־G שצמתי הביניים שלו לקוחים מתוך $\{v_1,\dots,v_k\}$. יהי p מסלול כזה.
נחלק לשני מקרים:

1. נניח כי v_k נמצא ב־p, נשים לב ש־v_K מופיע פעם אחת $v_i \longrightarrow v_k \longrightarrow v_j$.
ב־p כי אין מעגלים שליליים שמסלול קל ביותר הוא פשוט.
$=\delta_k(i,j)$
סך משקלי הקשתות ב־p=
סך משקלי הקשתות בסיפא מ־v_k ל־ v_j+סך משקל הקשתות ברישא מ־v_iעל־v_k
$\delta_k(i,j)=\delta_{k-1}(i,k)+\delta_{k-1}(k,j)$.
נשים לב שסך משקלי הקשתות ברישא של p מ־v_i ל־v_k שווה ל־$\delta_{k-1}(i,k)$
שכן אחרת $\delta_{k-1}(i,k)$ קטן יותר וניתן להחליף את הרישא במסלול טוב יותר מ־v_iל־v_k
שצמתי הביניים שלו מתוך $\{v_i,\dots,v_{k-1}\}$ בסתירה לאופטימליות של k).
באופן דומה, סך משקלי הקשתות של הסיפא של p מ־v_kל־v_J שווה ל־ $\delta_{k-1}(k,j)$.

$$\delta_k(i,j)=\delta_{k-1}(i,j)+\delta_{k-1}(i,j)\underbrace{=}_{\text{induction}}D_{k-1}(i,k)+D_{k-1}(k,j)$$

נראה ש־ $D_{k-1}(i,j)\geq D_{k-1}(i,k)+D_{k-1}(k,j)$

$$D_{k-1}(i,j)\underbrace{=}_{\text{induction}}\delta_{k-1}(i,j)\geq\delta_k(i,j)\underbrace{=}_{\text{proved}}D_{k-1}(i,k)+D_{k-1}(k,j)$$

$\Longleftarrow\delta_k(i,j)=D_k(i,j)$.

2. נניח כי v_K לא נמצא ב־p.

$$\Rightarrow\delta_k(i,j)=\Sigma(\text{weights in p})\underbrace{=}_{v_k\text{not in p}}\delta_{k-1}(i,j)\underbrace{=}_{\text{induction}}D_{k-1}(i,j)$$

נראה ש־$D_{k-1}(i,j)\leq D_{k-1}(i,k)+D_{k-1}(k,j)$

$$D_{k-1}(i,j)+D_{k-1}(k,j)\underbrace{=}_{\text{induction}}\delta_{k-1}(i,k)+\delta_{k-1}(k,j)\underbrace{\geq}_{\text{definition}}\delta_k(i,j)=D_{k-1}(i,j)$$

$$\delta_k(i,j)=D_k(i,j)\Longleftarrow$$

המקרה הנותר הוא שאין מסלול מ־v_iל־v_j שצמתי הביניים שלו לקוחים מתוך $\{v_1,\dots v_k\}$
, כלומר $\delta_k(i,j)=\infty$.
נראה שמתקיים $D_k(i,j)=\infty$.
נניח בשלילה שלא, ולכן לפי הגדרת D מתקיים:
$D_k(i,j)$ סופי או $D_{k-1}(i,k)+D_{k-1}(k,j)$ סופי.
לפי הנחת האינדוקצייה, אם זה המצב אזי יש מסלול מv_i ל־v_j שצמתי הביניים שלו לא מתוך $\{v_1,\dots,v_k\}$ בסתירה להנחת השלילה.

דוגמה 4 Sequence Alignment

⋆ דוגמה: הוקלדה המחרוזת "ocurrance" ומוצע למשתמש התיקון ל־"ocurrence".

כיצד נתאים בין שתי המחרוזות ?

<table>
<tr><td>o</td><td>c</td><td>$\boxed{\text{-}}$</td><td>u</td><td>r</td><td>r</td><td>$\boxed{\text{a}}$</td><td>$\boxed{\text{-}}$</td><td>n</td><td>c</td><td>e</td></tr>
<tr><td>$\vert$</td><td>$\vert$</td><td>$\vert$</td><td>$\vert$</td><td>$\vert$</td><td>$\vert$</td><td>$\vert$</td><td>$\vert$</td><td>$\vert$</td><td>$\vert$</td><td>$\vert$</td></tr>
<tr><td>o</td><td>c</td><td>c</td><td>u</td><td>r</td><td>r</td><td>$\boxed{\text{e}}$</td><td>$\boxed{\text{e}}$</td><td>n</td><td>c</td><td>e</td></tr>
<tr><td colspan="11">שלושה תוים שלא התאימו</td></tr>
<tr><td colspan="11"></td></tr>
<tr><td>o</td><td>c</td><td>-</td><td>u</td><td>r</td><td>r</td><td>$\boxed{\text{a}}$</td><td>n</td><td>c</td><td>e</td><td rowspan="4"></td></tr>
<tr><td>$\vert$</td><td>$\vert$</td><td>$\vert$</td><td>$\vert$</td><td>$\vert$</td><td>$\vert$</td><td>$\vert$</td><td>$\vert$</td><td>$\vert$</td><td>$\vert$</td></tr>
<tr><td>o</td><td>c</td><td>c</td><td>u</td><td>r</td><td>r</td><td>e</td><td>n</td><td>c</td><td>e</td></tr>
<tr><td colspan="10">תו אחד שלא התאים + התאמה שאינה בין אותו תו</td></tr>
</table>

<u>נתון:</u> שתי מחרוזות $\begin{cases} x=x_1x_2\dots x_n\\ y=y_1y_2\dots y_m\end{cases}$

שידוך M בין $\{1,\dots,n\}$ ו־$\{1,\dots,m\}$ יהיה חוקי אם:

1. כל אינדקס מופיע לכל היותר פעם אחת ב־M.
2. $(i,j),(i',j')\in M$ ו־$i<i'$ $\Leftarrow$ $j<j'$.

שאלה:

כיצד נכמת ערך שידוך חוקי?
אילו יש מיקום שאינו מותאם, אזי נשלם עליו $\delta>0$.
$(i,j)\in M\Leftarrow$ קיימת טבלה שמציימת מהי עלות השידוך של $\alpha(x_i,y_j)$.
<u>מטרה:</u> למצוא שידוך חוקי זול ביותר בין x ל־y.
<u>הבחנה:</u> נתונות $x=x_1\dots,x_n$ ו־$y=y_1\dots y_m$ אזי:
$(n,m)\in M$ או שלפחות אחד מ־x_n ו־y_nלא משודך (הוכחה, כיוון שאין הצטלבויות).
תתי הבעיות שעניינו אותנו הן רשאיות(תאופיין ע"י i ו־j)

$$\begin{cases} x_1\dots x_i\\ y_1\dots y_j\end{cases}$$

נגדיר את $A(i,j)$ להיות עלות שידוך חוקי זול ביותר בין $x_1 \dots x_i$ ו־$y_1 \dots y_j$.

$$B(i,j) = \begin{cases} \delta \cdot j & i = 0 \\ \delta \cdot i & j = 0 \\ min\{\alpha(x_i, y_j) + B(i-1, j-1), \delta + B(i-1, j), \delta + B(i, j-1)\} & \text{otherwise} \end{cases}$$

לכל $1 \leq i \leq n$ ו־$1 \leq j \leq m$: $A(i,j) = B(i,j)$.

שאלה:

כיצד לחשב את B?

	1	2	3	$\cdots$	m
1	0	δ	2δ		$m\delta$
2	δ				
3	2δ				
$\vdots$					
n	$n\delta$				$\underbrace{?}_{\text{output}}$

קיימים מספר סדרים עבורם זמן הריצה הוא $O(n \cdot m)$.
סיבוכיות זכרון $O(min\{n,m\})$.

אלגוריתמים הרצאה 11

שאלה:

מתןמה רשת תקשורת (גרף מכוון) ונתון צומת שולח וצומת מקבל.
לכל קשת נתון הקצב בו ניתן לשלוח מידע עליה.

- ⋆ מהו הקצב הגדול ביותר בו ניתן לשלוח מהצומת השולח לצומת המקבל?
- ⋆ מה הניתוב שמשיג את הקצב הגדול ביותר?

זרימה

<u>הגדרה:</u> רשת זרימה היא רביעייה (G, s, t, c) כאשר:

- ⋆ $G = (V, E)$ גרף מכוון.
- ⋆ $s, t \in V$ שני צמתים בגרף (s נקרא מקור ו־t נקרא בור).
- ⋆ $c : E \to \mathbb{R}_+$ פונקציית קיבול על הקשתות.

<u>הגדרה:</u> בהנתן קשת זרימה (G, s, t, c), **פונקציית זרימה** היא $f : E \to \mathbb{R}_+$ שמקיימת:

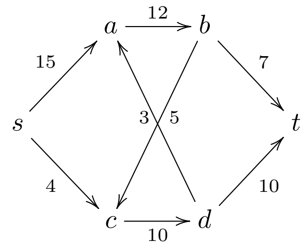
- ⋆ $\forall e \in E \; 0 \leq f(e) \leq c(e)$ (אילוצי קיבול).
- ⋆ $\forall u \in V \backslash \{s, t\} \sum\limits_{e \in \delta^+(u)} f(e) = \sum\limits_{e \in \delta^-(u)} f(e)$ (אילוצי שימור).

כאשר: $\delta^+(u)$ מהווה את אוסף הקשתות היוצאות מ־u.
$\delta^-(u)$ מהווה את אוסף הקשתות הנכנסות מ־u.
<u>הגדרה:</u> בהינתן רשת זרימה (G, s, t, c) ופונקציית זרימה f ברשת, **הערך** של f יסומן ויוגדר:

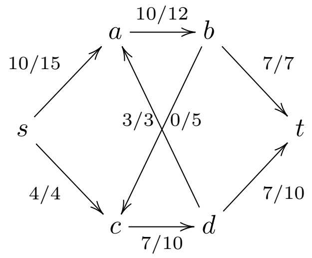
$$|f| \triangleq \sum_{e \in \delta^+(s)} f(e) - \sum_{e \in \delta^-(s)} f(e)$$

כלומר, "נטו" הזרימה היוצאת מ־s.
<u>הבעיה:</u>
בהינתן רשת זרימה (G, s, t, c), מהי פונקציית הזרימה f בעלת ערך $|f|$ הקטן ביותר?

דוגמה:



הפונקצייה המזרימה O על כל קשת היא חוקית ועותקים עבורה $|f| = 0$.



הפונקצייה (הערך השמאלי בגרף) היא פונקציית זרימה חוקית וערכה $|f| = 14$.

שאלה:

כיצד נוכיח שאין פונקציית ערימה שערכה גדול ממש מ־14 ?

הגדרה:

בהינתן רשת זרימה (G, s, t, c), <u>חתך $s - t$</u> הוא $S \subseteq V$ כך ש־$s \in S$ ו־$t \notin S$.
<u>קיבול</u> החתך מוגדר להיות סך הקיבולים של הקשתות הקדמיות בחתך, כלומר

$$c(S) = \sum_{\substack{e=(u \to v) \in E: \\ u \in S, v \notin S}} c(e)$$

טענה 1

בהינתן רשת זרימה (G, s, t, c), לכל פונקציית זרימה חוקית f ברשת ולכל חתך $s - t$ ו־S:

$$|f| < c(S)$$

טענת עזר:

בהינתן רשת זרימה (G, s, t, c), וחתך $s - t$ S ופונקציית זרימה f, מתקיים:

$$\sum_{\substack{e=(u\to v)\in E:\\ u\in S, v\notin S}} f(e) \quad - \sum_{\substack{e=(u\to v)\in E:\\ u\notin S, v\in S}} f(e) \quad = |f|$$

הוכחת טענת העזר:

$$|f| \underbrace{=}_{\text{def.}} \sum_{e\in\delta^+(s)} f(e) - \sum_{e\in\delta^-(s)} f(e) \underbrace{=}_{\text{current save.}} \sum_{u\in S}\left[\sum_{e\in\delta^+(s)} f(e) - \sum_{e\in\delta^-(s)} f(e)\right]$$
$$\underbrace{=}_{\text{sum order change}} \sum_{\substack{e=(u\to v)\in E:\\ u\in S, v\notin S}} f(e) \quad - \sum_{\substack{e=(u\to v)\in E:\\ u\notin S, v\in S}} f(e) \quad = |f|$$

הוכחת טענה 1:

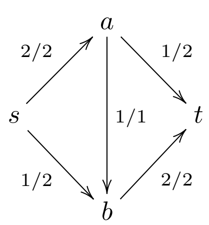
$$|f| \underbrace{=}_{\text{Aux. claim}} \sum_{\substack{e=(u\to v)\in E:\\ u\in S, v\notin S}} f(e) \quad - \sum_{\substack{e=(u\to v)\in E:\\ u\notin S, v\in S}} f(e) \quad \le \sum_{\substack{e=(u\to v)\in E:\\ u\in S, v\notin S}} c(e) \quad - 0 = c(S)$$

מסקנה:

אילו מצאנו פונקציות זרימה וחתך $s-t$ S כך ש: $|f| = c(S)$, מובטח שיש לנו זרימה אופטימלית.

הרעיון באופן כללי:

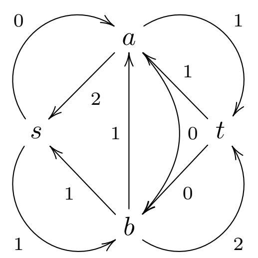
נתחיל מזרימה חוקית (למשל, זרימת האפס) וננסה לשפר אותה עד שנתקע.
צעד השיפור יעשה על ידי הזרימה על מסלול מ־s ל־t.
<u>דוגמה:</u>



הגדרה:

נתונה רשת זרימה (G,s,t,c) ופונקציית זרימה חוקית f ברשת.
השרת השיורית היא (G_f,s,t,c_p) כאשר:

- $\star$ $G_f=(V,E_f)$
- $\star$ $E_f = E\bigcup\{\overline{e}|e\in E$ הקשת ההפוכה לקשת $\overline{e}\}$.
- $\star$ $\forall e\in E,\ C_f(e)\triangleq c(e)-f(e)$.
- $\star$ $C_f(\overline{e})=f(e)$



הגדרת פעולת חיבור זרימות

נתונה רשת זרימה (G,s,t,c) ופונקציית זרימה f חוקית בה.
f' פונקציית זרימה חוקית ברשת השיורית (G_f,s,t,c_f).
נגדיר פונקציית זרימה שנסמן ב־$f+f'$ באופן הבא:

$$\forall e\in E,\ (f+f')(e)\triangleq f(e)+f'(e)-f'(\overline{e})$$

טענה:

בהינתן רשת זרימה (G,s,t,c), פונקציית זרימה f החוקית בה, ו־f' פונק' זרימה חוקית ב־(G_f,s,t,c_f), אזי: $f+f'$ זרימה חוקית ב־(G,s,t,c) וערכה: $|f+f'|=|f|+|f'|$

הגדרה:

מסלול שיפור הוא מסלול מ־s ל־t ברשת השיורית שכל הקיבולים השיוריים בו חיוביים ממש.

אלגוריתם Ford Fulkerson

1. מתחילים עם זרימה 0 $\forall e\in E\ f(e)\equiv 0$
2. כל עוד ב־G_f יש מסלול שיפור p,

(א) מגדירים f' ברשת השיורית (G_f, s, t, c_f):

$$e \notin P \Rightarrow f'(e) = 0 \, , e \in P \Rightarrow f'(e) = \min_{e \in P}\{c_f(e)\}$$

(ב) $f \leftarrow f + f'$

3. הפלט הוא f.