הפקולטה למדעים מדויקים
ע״ש ריימונד ובברלי סאקלר
אוניברסיטת תל אביב

החוג למדעי המחשב (0368)
אלגוריתמים (2160)
(גרסה ארוכה)

מרצה: אורי צוויק וחיים קפלן
מתרגלים: דני דורפמן ועמרי פורת
תשפ"ג, סמסטר ב' (2023)
+ קורס בקמפוס (הפתוחה)

מסכם: רועי מעין

The Raymond and
Beverly Sackler Faculty
of Exact Sciences
Tel Aviv University

פרק 1 – אלגוריתמים בסיסיים

פרק 2 – מק"בים ותכנות דינמי

פרק 3 – זרימה ותכנות לינארי

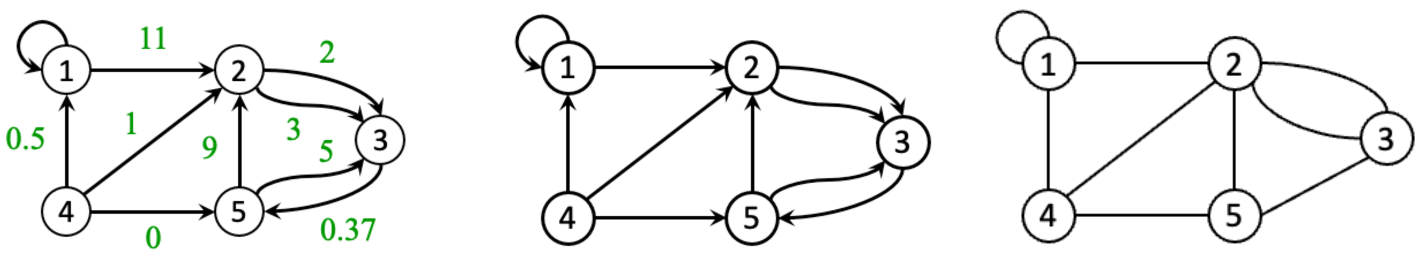
1 – אלגוריתמים בסיסיים

מבוא

הגדרת גרפים

גרף $G=(V,E)$ הוא אוסף של צמתים (V) וקשתות (E). נסמן לרוב $n=|V|, m=|E|$. סוגי גרפים:

- גרף לא מכוון – קשת (edge) היא זוג **לא סדור** של צמתים $\{u,v\}$. **אין כיוון לקשת**.
- גרף מכוון – קשת (arc) היא זוג **סדור** של צמתים, והיא הולכת מצומת מסוים לצומת אחר, **יש כיוון לקשת**. במקרה זה $(u,v)\neq(v,u)$.
- גרף ממושקל – נוסיף רכיב שלישי שהוא פונקציית משקל $w\colon E\to\mathbb{R}$ אשר מתאימה לכל קשת משקל ממשי. אפשר גם להסתכל על כל קשת בשלשה: $(u,v,w(e))$, כאשר עבור קשת e נקבל את המשקל שלה על ידי $w(e)$.



מונחים נוספים:

- לולאה – קשת עצמית מצומת לעצמו, למשל $\{1,1\}$. במציאת מסלולים קצרים ביותר לא נתייחס ללולאות.
- קשתות מקבילות – זוג קשתות בין אותם הצמתים, למשל $\{2,3\},\{2,3\}$. ברוב הבעיות שנדבר עליהן לא יהיה לזה משמעות. נוכל להחליף את הקשתות המקבילות בקשת אחת.
- גרף פשוט – **לא מכיל** לולאות וקשתות מקבילות.
- גרף לא פשוט – **מכיל** לולאות או קשתות מקבילות.
- קליקה (Clique): קבוצת צמתים בגרף לא מכוון כך שבין כל שניים מהם יש קשת.

ייצוג גרפים:

שיטה	יתרונות	חסרונות
רשימת שכנויות נשמור – (adjacency list) רשימה של צמתי הגרף, בה כל צומת u מצביע **לרשימה מקושרת** של השכנים שלו u.Adj (כל הצמתים שיש לו קשת אליהם). בגרף מכוון: נשמור רק את **הקשתות היוצאות** מכל צומת.	• סיבוכיות מקום: $O(m+n)$. • בהינתן צומת u ניתן בקלות לקבל את הקשתות שהוא נמצא בהן – בכמות הקשתות. • בהינתן קשת e ניתן בקלות למצוא את הקשת הקודמת או הבאה. • אפשר **להכניס ולמחוק קשתות** בקלות.	• בהינתן שני צמתים $u,v\in V$, אין דרך מהירה לבדוק האם קיימת קשת ביניהם, האם $(u,v)\in E$.
מטריצת שכנויות מטריצה – (adjacency matrix) בוליאנית בגודל $n\times n$. התא (i,j) הוא 1 אמ"מ $(i,j)\in E$.	• בהינתן שני צמתים $u,v\in V$ אפשר לבדוק ב-$O(1)$ האם קיימת קשת ביניהם. • אפשר להכניס ולמחוק קשתות בקלות. • בגרף לא מכוון: המטריצה סימטרית ואפשר לאחסן רק חצי ממנה.	• אם הגרף יחסית דליל: $m\ll n^2$ כלומר יש מעט קשתות, עדיין אנו צריכים $O(n^2)$ זיכרון. • בהינתן צומת u צריך $O(n)$ זמן כדי לעבור על הקשתות שלו, חייבים לעבור על כל התאים ולחפש היכן יש 1. • בהינתן קשת e, קשה למצוא קשת קודמת או עוקבת.
טבלת hash – נאחסן את הקשתות בטבלת hash (לצד רשימת הצמתים הרגילה).	• חזרנו לכמות זיכרון $O(m+n)$. • אפשר לבדוק האם קיימת קשת בזמן $O(1)$ בתוחלת.	• משתמשים ברנדומיזציה. • עדיין צריך $O(n)$ זמן כדי למצוא את כל השכנים של צומת u.
ייצוג היברידי – נשמור גם רשימת שכנויות וגם טבלת hash.	• משלב את היתרונות של שני הייצוגים.	• לרוב לא באמת זקוקים לזה.

מושגים בגרפים

מסלולים:

- מסלול (Path) – סדרה של קשתות: $(u_0,u_1),(u_1,u_2),\dots,(u_{k-1},u_k)$. המסלול הוא מ-$u_0$ ל-u_k ובעל k קשתות.
  - **פשוט:** אם כל הקשתות שונות זו מזו ולא מבקרים באותו צומת יותר מפעם אחת (אחרת, המסלול **לא פשוט**).
  - **מעגל פשוט:** אם כל הקשתות שונות פרט לכך שמתקיים $u_0=u_k$, המסלול מתחיל ונגמר באותו צומת. כלומר מדובר במסלול פשוט בתוספת קשת מהצומת האחרון לראשון.
- כאשר ניקח מטריצת שכנויות A ונעלה אותה בריבוע, נקבל שהמשמעות של $A_{i,j}^2$ הוא מספר המסלולים המכוונים מצומת i לצומת j, שאורכם 2. כך גם עבור A^3 נקבל את מספר המסלולים באורך 3.
- דרגה – עבור צומת v, הדרגה שלו היא מספר הקשתות המחוברות אליו. נסמן אותה ב-$\deg(v)$. **בגרף מכוון,** לכל קודקוד יש **דרגת כניסה** (מספר הקשתות הנכנסות) **ודרגת יציאה** (מספר הקשתות היוצאות).
- קשירות (Connectivity):
  - גרף **לא מכוון** הוא **קשיר** אם בין כל שני צמתים $u,v\in V$ בגרף יש מסלול בין u ל-v.
  - גרף **מכוון** הוא **קשיר חזק** אם בין כל שני צמתים $u,v\in V$ בגרף יש מסלול בין u ל-v **וגם** מסלול בין v ל-u.
- רכיב קשירות (Connected Component): תת-קבוצה מקסימלית של צמתים המקיימת:
  - תכונה פנימית – כל שני צמתים ברכיב קשורים זה לזה באמצעות מסלול.
  - תכונה חיצונית – אין מסלול מצומת בתוך הרכיב לצומת אחר מחוץ לרכיב.
- רכיב קשיר חזק (Strongly Connected Component): תת-קבוצה מקסימלית של צמתים קשירים בגרף מכוון.

עצים:

- עץ (Tree) – עץ הוא גרף **קשיר, לא מכוון, חסר מעגלים (פשוטים)**. נסתכל על השקילויות הבאות:
  - $T=(V,E_T)$ הוא עץ.
  - קיים מסלול פשוט יחיד בין כל זוג קודקודים.
  - T הוא **חסר מעגלים מקסימלי**, אבל כל קשת שנוסיף לו בהכרח תסגור מעגל.
  - T הוא **קשיר מינימלי**, כל קשת שנזרוק ממנו תהפוך אותו לגרף לא קשיר.
  - T חסר מעגלים, ומתקיים $|E_T|=|V|-1$.
  - T קשיר, ומתקיים $|E_T|=|V|-1$.
- תתי-גרפים (Subgraphs) – בהינתן גרף $G=(V,E)$, נאמר כי הגרף $H=(V',E')$ הוא תת-גרף של G: כאשר $V'\subseteq V, E'\subseteq E$. כלומר אנו זורקים צמתים ואת הקשתות שביניהם.

טענות:

- למה: בגרף (מכוון או לא מכוון) אם יש **מסלול** מ-u ל-v, יש גם **מסלול פשוט** מ-u ל-v.
  - הוכחה: במסלול לא פשוט אנו חוזרים לצומת שכבר היינו בו, אז כאשר נסגר מעגל כזה – נוריד אותו. אם מה שקיבלנו עדיין לא פשוט, נוכל לחזור על התהליך שוב.
- טענה: $\sum_{v\in V}\mathbf{deg}\,(v)=2|E|$. כל קשת (u,v) מחברת בין שני קודקודים, לכן היא תורמת 1 לדרגה של כל צומת.
  - בגרף לא יכול להיות שיש רק קודקוד אחד מדרגה אי-זוגית, יש לפחות 2. **סכום הדרגות חייב להיות זוגי.**
  - בגרף מכוון, נמדוד עבור דרגת כניסה ויציאה בנפרד ונקבל: $\sum_{v\in V}\mathbf{deg_{in}}\,(v)=\sum_{v\in V}\mathbf{deg_{out}}\,(v)=|E|$.
- אם גרף הוא קשיר אז מתקיים $|E|\geq|V|-1$.
- אם בגרף מתקיים $|E|\geq|V|$ אז קיים מעגל.

מעגלי ומסלולי אוילר (תרגול 1)

עבור גרף $G=(V,E)$:

- מסלול אוילר – מסלול בגרף אשר עובר על כל קשת בדיוק פעם אחת (לאו דווקא מסתיים בקודקוד שבו הוא התחיל).
- מעגל אוילר – מסלול אוילר שמתחיל ומסתיים באותו קודקוד.

מעגל אוילר: גרף (לאו דווקא פשוט) קשיר ולא מכוון $G=(V,E)$ מכיל מעגל אוילר $\Leftrightarrow$ לכל קודקוד $u\in V$ יש דרגה זוגית.

הוכחה:

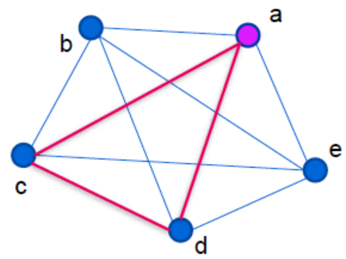
$\Leftarrow$: נניח שקיים מעגל אוילר ונראה שכל הדרגות בגרף זוגיות. נבחר מעגל אוילר כלשהו ונטייל לאורכו החל מקודקוד v. לכל קודקוד $u\in V$ שעברנו בו, נסמן את מספר הפעמים שעברנו בו במסלול ב-k_u. כל קודקוד שאינו קודקוד ההתחלה: $w\in V\backslash\{v\}$ בכל ביקור בו אנו נכנסים אליו דרך קשת אחת ויוצאים דרך אחרת. כלומר, $\deg(w)=2k_w$. עבור קודקוד ההתחלה נספור כמו קודם ונחסיר בהתאם 1 על כל קשת חסרה (בהתחלה יש רק קשת יוצאת, ובסוף רק קשת נכנסת): $\deg(v)=2k_v-1-1=2k_v-2$.

$\Rightarrow$: נניח שכל הדרגות זוגיות ונראה שקיים מעגל אוילר. נתאר אלגוריתם שתמיד מוצא מעגל אוילר בגרף מסוג זה. הטענה נובעת מנכונות האלגוריתם:

שלב ראשון:

- אלגוריתם: נבחר קודקוד כלשהו v ונתחיל ממנו טיול. בכל שלב, נבחר **קשת שעדיין לא עברנו עליה, נחצה אותה, ונזרוק אותה מהגרף**. נמשיך כך **עד שנחזור ל-v**.

$a \to c \to d \to a$



- נכונות: כל עוד לא חזרנו ל-v, תמיד נוכל להמשיך את הטיול.
  - נראה כי בכל שלב בריצת האלגוריתם מתקיים: אם הטיול נמצא בשלב מסוים בצומת u (השונה מ-v) אז ל-u דרגה אי-זוגית (מכאן נובעת הטענה כי זה אומר שיש לפחות קשת אחת להמשיך איתה את הטיול, הדרגה לא 0).
  - נוכיח משהו קצת יותר חזק: בכל שלב, **כל צומת u (השונה מ-v) שהטיול לא נמצא בו, הוא בעל דרגה זוגית**.
  - **נוכיח באינדוקציה על אורך המסלול שיש עד כה**.
  - בסיס: נצא מ-v ל-u, כיוון שכל הדרגות זוגיות, לאחר זריקת הקשת הדרגה של u הופכת להיות אי-זוגית (גם של v – לא מעניין אותנו). כל שאר הדרגות עדיין זוגיות.
  - צעד: נסתכל על הקשת מ-u_{k-1} מ-u_k (הצעד ה-k של המסלול). מהנחת האינדוקציה, u_{k-1} אי-זוגי, ו-u_k זוגי. כאשר נמחק את הקשת, הם יתהפכו. כיוון שבכל שלב הדרגה היא אי-זוגית, **לא ייתכן שנתקע בצומת מבלי אפשרות להמשיך**. נסיים את התהליך רק כאשר נגיע ל-v.

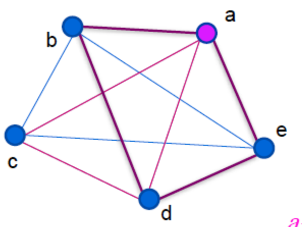
שלב שני:

- אלגוריתם: אם המעגל שמצאנו מכיל את כל הקשתות – סיימנו. אחרת, לפחות לאחד מהקודקודים שעברנו בהם **עדיין יש דרגה חיובית ונוכל להמשיך ממנו** (הגרף קשיר, ולכן יש מסלול בין כל שני קודקודים בגרף, יש עוד קשת לעבור בה). קיבלנו שני מעגלים זרים בקשתות. בנוסף, הקודקוד הראשון של המעגל שני מופיע גם במעגל הראשון. נאחד אותם למעגל יחיד (נדחוף במקום המתאים). **נחזור על שלב זה עד שהמעגל יכיל את כל הקשתות.**

:

$a \to c \to d \to a$

$d \to e \to a \to b \to d$



$a \to c \to d \to e \to a \to b \to d \to a$

- נכונות:
  - נכונות הטענה מהחלק הראשון נשמרת גם כאשר מבצעים אותו מספר פעמים – לא ייתכן שהאלגוריתם יתקע באמצע טיול על הגרף (הדרגות תמיד נשארות זוגיות).
  - בתחילת כל שלב, אם נותרו קשתות בגרף – קיים במעגל הנוכחי קודקוד עם דרגה חיובית (כיוון שהגרף קשיר).
  - בסוף כל שלב נקבל מעגל עם מספר גדול יותר של קשתות. האלגוריתם נמשך **כל עוד יש קשתות שלא נמצאות במעגל**, ולכן תמיד יתקבל לבסוף מעגל אוילר.

סיבוכיות:

- הגרף מיוצג על ידי **רשימת שכנויות**. המעגל שנבנה מוחזק **ברשימה מקושרת**. כאשר נזדקק לקשת שסמוכה לקודקוד כלשהו, נבחר את **הקשת הראשונה** ברשימת השכנויות שלו - $O(1)$. כיוון שאנו מוחקים כל קשת לאחר שעברנו עליה, סך זמן הפעולות על הקשתות הינו $O(|E|)$.
- נחזיק מצביע **לקודקוד הראשון במעגל הנוכחי שדרגתו עדיין חיובית** – נאתחל אותו ב-v (הקודקוד הראשון). אם הדרגה של הקודקוד הופכת ל-0, נקדם את המצביע קדימה לקודקוד הבא. סך עדכוני המצביע - $O(|E|)$. סה"כ זמן הריצה $O(|E|)$.

מסלול אוילר: גרף קשיר ולא מכוון $G = (V, E)$ מכיל מסלול אוילר (שאינו מעגל) $\Leftrightarrow$ הוא מכיל בדיוק שני צמתים מדרגה אי-זוגית.

$\Leftarrow$: נניח שקיים מסלול אוילר (שאינו מעגל) ונראה שבדיוק לשני קודקודים יש דרגה אי-זוגית. בדומה להוכחה הקודמת, נטייל לאורך המסלול. לקודקוד u (שאינו הראשון או האחרון במסלול) שעברנו דרכו k_u פעמים יש דרגה $2k_u$. לקודקוד v, אשר הוא הראשון או האחרון במסלול, יש דרגה $2k_v - 1$ (כאן מחסירים רק קשת אחת, כי אין קשת נכנסת חזרה ל-v אם הוא הראשון, ואין קשת יוצאת מ-v אם הוא האחרון).

$\Rightarrow$: נניח שבדיוק לשני קודקודים יש דרגה אי-זוגית ונראה שקיים מסלול אוילר. נוסיף קשת בין שני הקודקודים בעלי הדרגה האי-זוגית (יכול להיות שהיא קשת מקבילה). כל הקודקודים הם בדרגה זוגית, ולכן יש בו מעגל אוילר. **נסיר מהמעגל את הקשת שהוספנו לגרף ונקבל מסלול אוילר עבור הגרף המקורי.** מעגל האוילר שקיבלנו עובר בכל קשת כולל הקשת החדשה בדיוק פעם אחת, אז עכשיו המסלול שקיבלנו עובר בכל קשת פרט לקשת החדשה בדיוק פעם אחת, כלומר בכל קשת של הגרף המקורי – וזו ההגדרה של מסלול אוילר.

סריקה בגרפים

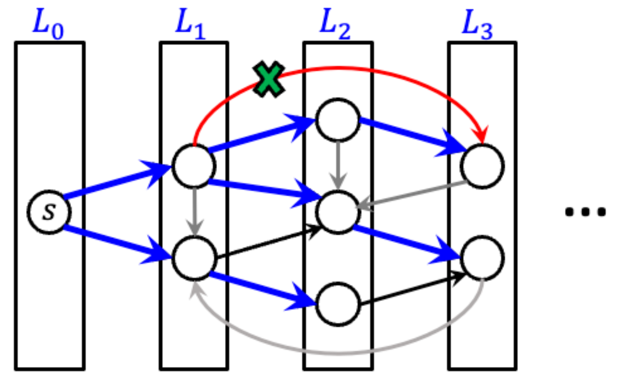
BFS (סריקה לרוחב)

<u>המטרה:</u> בהינתן גרף מכוון $G=(V,E)$ וצומת מקור $s\in V$, מי הם הצמתים הניגשים מ-s באמצעות מסלולים ישירים, ומה המרחקים והמסלולים הקצרים ביותר אליהם?

<u>הרעיון הבסיסי:</u> s הוא במרחק 0 מ-s. השכנים של s הם במרחק 1 מ-s. השכנים של השכנים של s הם במרחק 2 (לכל היותר) מ-s. נמשיך כך הלאה, ונעצור כאשר אנחנו מפסיקים לגלות צמתים חדשים. הסיבוביות תהיה $\boldsymbol{O(m+n)}$, כי אנחנו עוברים על כל צומת ועל כל קשת פעם אחת.

<u>שכבות ומסלולים קצרים ביותר:</u>

- נסמן ב-L_i עבור $i\geq 0$ את קבוצת הצמתים שבמרחק i מהצומת s. כמובן $L_0=\{s\}$, הקבוצה L_1 תכיל את השכנים של s. הקבוצה L_2 תכיל את השכנים של L_1 שאינם נמצאים ב-L_1.
- כל צומת פרט ל-s, כאשר נגלה אותו, נזכור **דרך איזה קשת הגענו אליו ונשמור אותה** (נצבע אותה בכחול). **הקשת הזאת היא האחרונה במסלול קצר ביותר לצומת הזה**. אם נסתכל על אוסף הקשתות הזה, נקבל **עץ של מסלולים קצרים ביותר**.
- נשים לב שלא יכולה להיות למשל קשת מצומת ב-L_1 לצומת ב-L_3, כי אז הצומת היה צריך להתגלות קודם, והוא היה בעצם ב-L_2. קשת אדומה כזאת כמו שסימנו, לא יכולה להתקיים.
- הערה: אם היינו שומרים לכל צומת את המסלול המלא, הוא יכול להיות מאוד ארוך, במקרה הגרוע כמעט n. לשמור את n המסלולים הקצרים ביותר היה עולה לנו $O(n^2)$ מקום. **לא היינו מקבלים אלגוריתם לינארי.**



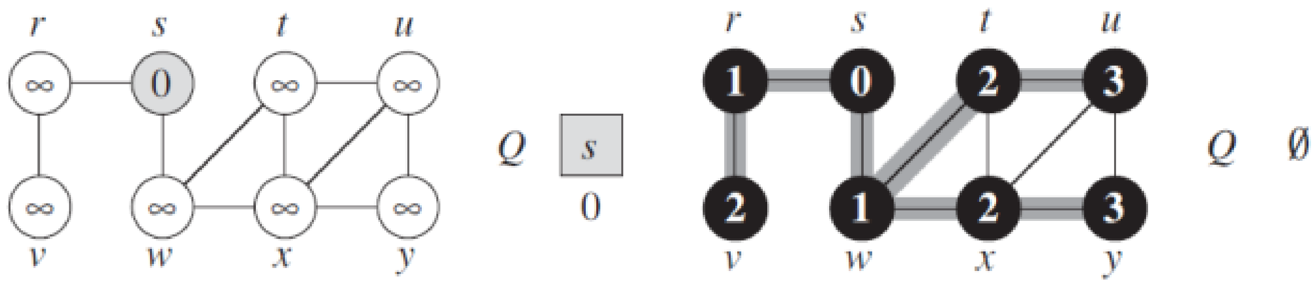
<u>האלגוריתם:</u>

- **לכל צומת v נשמור:**
  - <u>**color**</u> <u>(state)</u> – מצב נוכחי (WHITE – טרם גילינו אותו, GRAY – גילינו אבל לא סרקנו את הקשתות היוצאות שלו, BLACK – טופל. גילינו ובדקנו את הקשתות היוצאות שלו).
  - <u>**d**</u> <u>(distance)</u> – המרחק מהצומת s ל-v, אם כבר ידוע. תחילה נשים בערך זה אינסוף.
  - <u>**π**</u> <u>(prev)</u> – הצומת שממנו v התגלה, אם יש כזה. זה מלמד אותנו על הקשת הכחולה שנכנסת ל-v. כלומר, אם מ-u גילינו את v, אז $v.\pi=u$.
- כל הצמתים במצב GRAY מחכים בתור Q.
- (עקרונית, צריך למקם את שורה 18 אחרי שורה 11).

```
BFS(G, s)
1   for each vertex u ∈ G.V − {s}
2       u.color = WHITE
3       u.d = ∞
4       u.π = NIL
5   s.color = GRAY
6   s.d = 0
7   s.π = NIL
8   Q = ∅
9   ENQUEUE(Q, s)
10  while Q ≠ ∅
11      u = DEQUEUE(Q)
12      for each v ∈ G.Adj[u]
13          if v.color == WHITE
14              v.color = GRAY
15              v.d = u.d + 1
16              v.π = u
17              ENQUEUE(Q, v)
18      u.color = BLACK
```

<u>טענות:</u>

- <u>אינווריאנטה 1:</u> צומת v הוא GRAY ⇔ הוא ב-Q.
  - <u>הוכחה:</u> כאשר v מוכנס לתור צובעים אותו GRAY. כאשר מסירים אותו מהתור צובעים אותו BLACK.
- <u>אינווריאנטה 2:</u> הצמתים בתור Q הם תמיד צמתים משתי שכבות עוקבות. אם $v_1,\dots v_r\in Q$ אז מתקיים: $v_1.d\leq v_2.d\leq\cdots\leq v_r.d\leq v_1.d+1$
  - <u>הוכחה:</u> נוכיח באינדוקציה על צעדי האלגוריתם. בסיס הוא האתחול, הכנסנו את s ל-Q והתנאי מתקיים באופן ריק. נניח שהתנאי מתקיים. כאשר נוציא את v_1: אם הגענו לצומת u שלא ראינו קודם הוא צבוע WHITE, לכן נעדכן $u.d=v_1.d+1$.
- <u>מסקנה:</u> אם v_i נכנס ל-Q לפני v_j אז $v_i.d\leq v_j.d$.



נכונות:

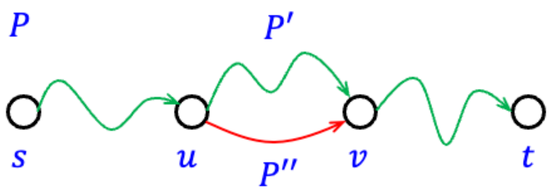
מרחק: עבור גרף $G = (V, E)$ וצמתים $u, v \in V$ נגדיר את המרחק ביניהם $\delta(u, v)$ להיות **מספר הקשתות של המסלול המינימלי** ביותר מ-u ל-v. אם אין מסלול ביניהם $\delta(u, v) = \infty$.

טענה: לאחר הרצת BFS מצומת s:

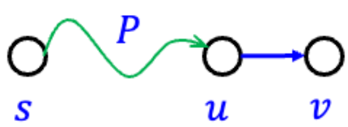
1. **לכל $v \in V$ מתקיים $v.d = \delta(s, v)$.**
2. **אם $\delta(s, v) < \infty$ אז יש מסלול קצר ביותר מ-s ל-v שמסתיים בקשת $(v.\pi, v)$.**

נגדיר $V_\pi = \{v | v \in V \ and \ v.\pi \neq NIL\} \cup \{s\}$ – כל הצמתים בעץ שבנינו, $E_\pi = \{(v.\pi, v) | v \in V \ and \ v.\pi \neq NIL\}$ – כל הקשתות בעץ שבנינו. נסיק, כי תת הגרף $G_\pi = (V_\pi, E_\pi)$ הוא **עץ של מסלולים קצרים ביותר מ-s לכל הצמתים הנגישים מ-s.**

- למה 1 (תת-מסלול של מק"ב הוא מק"ב): אם P מסלול קצר ביותר מ-s ל-t, ו-P' הוא החלק של P שמחבר בין u ל-v אז P' הוא מסלול קצר ביותר מ-u ל-v.
  - הוכחה: אם היה מסלול P' יותר קצר, אז יכולנו להחליף את המקורי בו, ונקבל מסלול קצר יותר מ-P בין s ל-t, וזו סתירה לכך ש-P הוא מסלול קצר ביותר..



- למה 2 (אי שוויון המשולש): עבור כל קשת $(u, v) \in E$ מתקיים $\delta(s, v) \leq \delta(s, u) + 1$.
  - הוכחה: אם P הוא מסלול קצר ביותר מ-s ל-u, קיים מסלול מ-s ל-v שאורכו $\ell(P) + 1$.



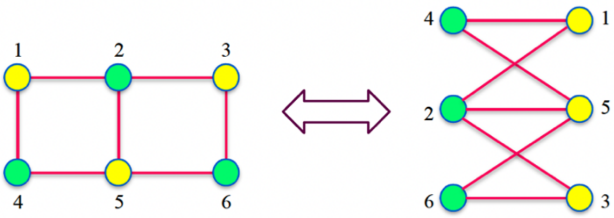
- למה 3: אם $(u, v) \in E$ הקשת האחרונה על מסלול קצר ביותר מ-s ל-v, אז מתקיים $\delta(s, v) = \delta(s, u) + 1$.
  - הוכחה: החלק של המסלול מ-s ל-u חייב להיות מסלול קצר ביותר, ומכאן נובע האורך של המסלול כולו.

בעת נוכיח את הנכונות במספר טענות:

- טענה 1: **כאשר v הופך להיות GRAY מתקיים $v.d \geq \delta(s, v)$.**
  - הוכחה: נוכיח באינדוקציה על הסדר שבו צבענו את הצמתים באפור. הבסיס: הצומת הראשון שנצבע GREY הוא s, ועבורו $s.d = \delta(s, s) = 0$. נניח שעכשיו צובעים את v GREY והטענה מתקיימת עבור כל הצמתים הקודמים שצבענו GREY. נסתכל על הקשת (u, v) שדרכה גילינו את v. לכן u התגלה קודם והוא GREY לפני v. לפי הנחת האינדוקציה $u.d \geq \delta(s, u)$. לכן: $v.d = u.d + 1 \geq \delta(s, u) + 1 \geq \delta(s, u)$.
- טענה 2: **אם $\delta(s, v) < \infty$ אז v בסופו של דבר יצבע GREY, נציב $v.d = \delta(s, v)$. אם $s \neq v$ אז יש מסלול קצר ביותר מ-s ל-v שמסתיים עם הקשת $(v.\pi, v)$.**
  - הוכחה: נוכיח באינדוקציה על $\delta(s, v)$. הבסיס: s הוא הצומת היחיד שעבורו $\delta(s, v) = 0$ והטענה מתקיימת. יהי P מסלול קצר ביותר מ-s לצומת v שמסתיים בקשת (u, v). כעת $\boldsymbol{\delta(s, v) = \delta(s, u) + 1}$. לפי הנחת האינדוקציה מתקיים $\boldsymbol{u.d = \delta(s, u)}$.
  - אם v WHITE כאשר נבחן את (u, v) אז $v.d = u.d + 1 = \delta(s, v), v.\pi = u$ והטענה מתקיימת. אחרת, נסתכל על הקשת (u', v) שדרכה האלגוריתם גילה את v והפך אותו ל-GREY. כלומר מתקיים $v.d = u'.d + 1, v.\pi = u'$. גילינו את v מ-$u'$ ולא מ-u ולכן u' נכנס לתור Q לפני u ולכן נקבל מהאינווריאנטה 2 כי: $u'.d \leq u.d$.
  - חייב להתקיים כאן שוויון: נניח בשלילה כי $u'.d < u.d$ ואז $v.d = u'.d + 1 \leq u.d = \delta(s, u) < \delta(s, v)$, וזו סתירה לטענה 1. לכן: $u'.d = u.d$ ואז נקבל $v.d = \delta(s, v)$. כלומר: ההוספה של (u', v) למסלול הקצר ביותר מ-s ל-u' יוצרת מסלול קצר ביותר מ-s ל-v.

BFS (תרגול 2)

1) בעיית הצביעה: בהינתן גרף, אנו רוצים לדעת האם ניתן לצבוע אותו **בשני צבעים**: לצבוע את צמתי הגרף בשני צבעים כך שאין שני צמתים סמוכים בעלי אותו הצבע. כל קשת מחברת בין צמתים בצבעים שונים.



גרף דו-צדדי (Bipartite Graph): גרף לא מכוון נקרא דו-צדדי אמ"מ אפשר לצבוע את צמתיו בשני צבעים, כך שאף קשת לא מחברת בין שני צמתים בעלי אותו צבע. נוכל לחלק את הצמתים לשתי קבוצות, כאשר בכל קבוצה אין קשתות בין הצמתים באותה קבוצה, רק קשתות בין צמתים בקבוצות שונות. כך נצבע כל קבוצה בצבע אחד.

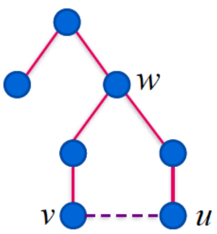
טענה: גרף $G = (V, E)$ הוא דו-צדדי ⇔ הוא אינו מכיל מעגל באורך אי-זוגי.

⇐: נניח ש-G הוא דו-צדדי, ונראה שכל מעגל בו הוא מעגל באורך זוגי. כל קשת ב-G עוברת מקבוצת צמתים אחת לשניה (מצד לצד). כדי לסגור מעגל מצומת מסוים בחזרה אליו, אנחנו יכולים לנוע רק לצד השני (קשת אחת), ובחזרה לצד הראשון (קשת שניה). כל מסלול שמתחיל ונגמר באותו צד – חייב להיות בעל מספר קשתות זוגי, ולכן **מעגל חייב להכיל מספר זוגי של קשתות**.

⇒: נניח שכל מעגל ה-G הוא באורך זוגי ונראה שהגרף דו-צדדי (הוא 2 צביע). מספיק להוכיח את הטענה לכל רכיב קשירות בנפרד (נצבע כל רכיב קשירות בנפרד, אין תלות בצבעים ביניהם). לכן, נניח בה"כ שהגרף קשיר, ונתאר **אלגוריתם שצובע את הגרף בשני צבעים** ונוכיח שהוא תמיד עובד.

אלגוריתם: נבחר צומת v באופן שרירותי ונריץ עליו BFS. את הצמתים ברמות זוגיות בעץ ה-BFS שנוצר נצבע בצהוב, ואת הצמתים ברמות האי-זוגיות נצבע בירוק.

נכונות: נראה שכל קשת מחברת בין שני צמתים בעלי צבעים שונים. מנכונות BFS: כל קשת בגרף מחברת בין צמתים **ברמות עוקבות של העץ**, או בין צמתים מאותה רמה (אין קשתות שמחברות בין רמות רחוקות).



- קשתות ברמות עוקבות: לפי האלגוריתם שלנו: קשת בין שתי רמות עוקבות תמיד תכיל צומת אחד צהוב וצומת אחד ירוק (כל שתי רמות עוקבות הן בצבעים שונים).
- קשתות באותה רמה: נראה שלא קיימות קשתות כאלו. נניח בשלילה שקיימת קשת המחברת בין הצמתים u, v אשר נמצאים באותה רמה בעץ. יהי w ה-lowest common ancestor (אב משותף ראשון) של u ו-v. נסמן ב-p את המסלול בין u ל-w בעץ, ונסמן את אורך המסלול ב-k. באופן דומה, נסמן ב-q את המסלול בין v ל-w בעץ, ונשים לב שאורכו אף הוא k. הקשת (v, u) **יוצרת מעגל באורך $2k + 1$** יחד עם p ו-q, וזו סתירה לכך שכל מעגל הוא באורך זוגי. לכן לא תיתכן קשת בין שני צמתים באותה רמה.

2) מסלולים קצרים ביותר:

מסלול קצר ביותר (מק"ב): עבור גרף $G = (V, E)$ וזוג צמתים $s, t \in V$ מק"ב מ-s ל-t הינו מסלול אשר מכיל מספר קשתות מינימלי מבין כל המסלולים מ-s ל-t (יכולים להיות כמה כאלו).

גרף המק"בים מ-s: הינו הגרף $G' = (V, E')$ כך שמתקיים $E' = \{e \in E : e \text{ is part of a shortest path from } s\}$.

תרגיל 1: נתון גרף לא מכוון וצומת $s \in V$. תארו אלגוריתם המחשב **את גרף המק"בים מ-s בזמן $O(|V| + |E|)$**.

פתרון פשוט שנכשל – נריץ BFS מ-s, נקבל עץ אשר מכיל מק"ב מ-s לכל צומת $v \in V$, אך יתכנו מק"בים שאינם מוכלים בעץ זה! יכול להיות **שחסרות קשתות**. אילו קשתות חסרות?

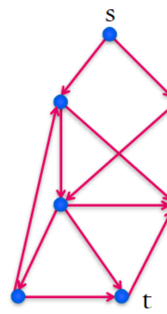
- קשתות בין **צמתים באותה הרמה – אף פעם לא על מק"ב**.
- קשתות בין **צמתים ברמות עוקבות – תמיד על מק"ב** (אבל לא כולן מופיעות בעץ BFS)

פתרון משופר – נריץ BFS מ-s. לאחר מכן, נעבור על כל הקשתות ונשמור את כל הקשתות בין רמות עוקבות.

טענה: קשת היא בין רמות עוקבות ⇔ היא ב-E' (כלומר על מק"ב מ-s).

⇐: כל קשת e בין צומת v ברמה $i - 1$ לבין צומת u ברמה i תמיד נמצאת במק"ב אל u. אורך המק"ב מ-s ל-u הוא i (כי u ברמה i). קיים מסלול באורך $i - 1$ מ-s ל-v. אם נוסיף את הקשת (v, u) למסלול, נקבל מסלול באורך i מ-s ל-u (והוא מק"ב).

⇒: באופן דומה נוכל להראות שקשת בין צמתים באותה רמה היא לא באף מק"ב (היא תמיד מאריכה את המסלול, וניתן לוותר עליה ולקבל מסלול קצר יותר).

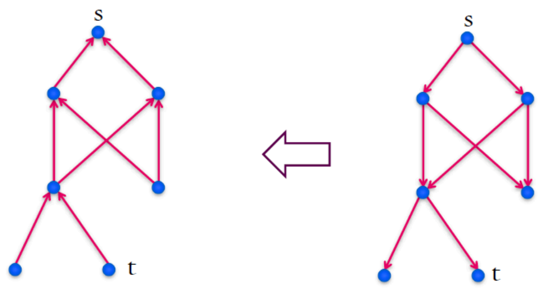


גרף המק"בים מ-s ל-t: עבור גרף $G=(V,E)$ וזוג צמתים $s,t\in V$, גרף המק"בים מ-s ל-t הינו הגרף $G'=(V,E')$ כך שמתקיים $E'=\{e\in E: e\ is\ part\ of\ a\ shortest\ path\ from\ \boldsymbol{s}\ \boldsymbol{to}\ \boldsymbol{t}\}$.

תרגיל 2: נתון גרף מכוון $G=(V,E)$ וזוג צמתים $s,t\in V$. תארו אלגוריתם המחשב את גרף המק"בים מ-s ל-t ב-$\boldsymbol{O}(|\boldsymbol{V}|+|\boldsymbol{E}|)$.

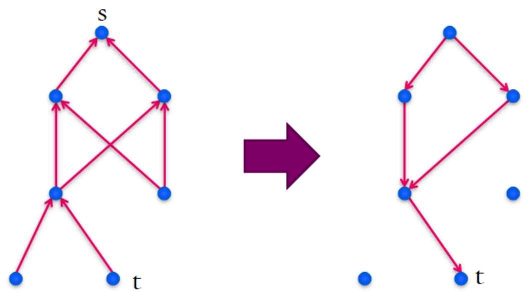
לאחר ריצת BFS על גרף מכוון, יש 3 סוגים של קשתות בגרף:

- קשתות בין צמתים באותה רמה.
- קשתות בין צמתים **ברמות עוקבות** (אלו יעניינו אותנו).
- קשתות מצומת ברמה i לצומת ברמה $j<i$ (קשת שלוקחת אחורה).



אלגוריתם (פתרון תשפ"א): נשתמש באלגוריתם שתיארנו קודם, הפעם עבור גרף מכוון (BFS + שמירת קשתות בין רמות עוקבות). נתעניין רק בצמתים שמהם אפשר להגיע ל-t.

- נהפוך את הכיוון של כל הקשתות שקיבלנו.
- נמצא אילו צמתים נגישים מ-t באמצעות BFS פעם שנייה. כל מסלול מ-v ל-t הופך למסלול מ-t ל-v בגרף ה"הפוך", לכן הצמתים שאפשר להגיע מ-t אליהם הם הצמתים שבמקור היה אפשר להגיע מהם אל t.
- נשמור כל קשת $u\to v$ עבור v שממנו t נגיש.
- נהפוך בחזרה את הכיוון של כל הקשתות בגרף שקיבלנו. זהו גרף המק"בים מ-s ל-t.



סיבוכיות: הרצת BFS פעם ראשונה: $O(|V|+|E|)$, שמירת קשתות בין רמות עוקבות: $O(|E|)$,
הפיכת קשתות הגרף: $O(|V|+|E|)$,
הרצת BFS פעם שנייה: $O(|V|+|E|)$, שמירת קשתות שבמסלול ל-t:
$O(|E|)$.

פתרון חלופי (תרגול 2 תשפ"ג):

אלגוריתם:

- לכל $v\in V$ נחשב את $\delta(s,v)$ ואת $\delta(v,t)$.
- קשת $e=(u,v)$ שייכת לגרף המסלולים הקצרים ביותר בין s ל-t אמ"מ

$$\delta(s,u)+1+\delta(v,t)=\delta(s,t)$$

נכונות:

- $\Rightarrow$ נניח ש-$e=(u,v)$ קשת המקיימת את השוויון $\delta(s,u)+1+\delta(v,t)=\delta(s,t)$
  - נסמן $s\to u_1\to\cdots\to u$ מק"ב בין s ל-u.
  - נסמן $v\to v_1\to\cdots\to t$ מק"ב בין v ל-t.
  - אזי המסלול $s\to u_1\to\cdots\to u\to v\to v_1\to\cdots\to t$ הוא מסלול שאורכו $\delta(s,u)+1+\delta(v,t)=\delta(s,t)$ ולכן הוא מסלול קצר ביותר בין s ל-t, ובפרט e קשת ששייכת למסלול קצר ביותר בין s ל-t.

- $\Leftarrow$ נניח ש-$e=(u,v)$ על מק"ב p בין s ל-t:
  $s\to u_1\to\cdots\to u\to v\to v_1\to\cdots\to t$
  - נסמן ב-p_1 את החלק במסלול מ-s ל-u.
  - נסמן ב-p_2 את החלק במסלול מ-v ל-t.
  - נוכיח ש-p_1 הוא מסלול קצר ביותר בין s ל-u (ובאופן סימטרי עבור p_2):
    נניח בשלילה שקיים ${p_1}'$ מסלול קצר יותר בין s ל-u:
    $s\to u_1'\to\cdots\to u$
    אזי נוכל לקבל מסלול קצר יותר בין s ל-t:
    $s\to u_1'\to\cdots\to u\to v\to v_1\to\cdots\to t$
    בסתירה למינימליות של p.
  - נובע ש-$\delta(s,t)=\delta(s,u)+1+\delta(v,t)$

סיבוכיות:

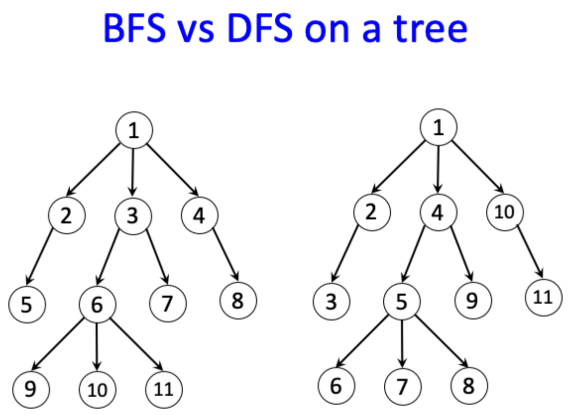
- חישוב $\delta(s,v)$ לכל $v\in V$, הרצת BFS מ-s – $\boldsymbol{O}(|\boldsymbol{V}|+|\boldsymbol{E}|)$
- איך מחשבים את $\delta(v,t)$??
- ~~הרצת BFS מכל $v\in V$ $\boldsymbol{O}\left((|V|+|E|)^2\right)$~~ ☹
- הרצת BFS מ-t על הגרף המשוחלף G^T – $\boldsymbol{O}(|\boldsymbol{V}|+|\boldsymbol{E}|)$ ☺
- בדיקת הקשתות שמקיימות את התנאי ושמירתן – $\boldsymbol{O}(|\boldsymbol{E}|)$
- סה"כ – $\boldsymbol{O}(|\boldsymbol{V}|+|\boldsymbol{E}|)$

DFS (סריקה לעומק)

<u>המטרה והרעיון הבסיסי:</u> נרצה לסרוק את הגרף ולגלות את המבנה שלו, אך הפעם אנחנו הולכים על קשתות הגרף, ואם מגיעים למקום שאנחנו לא מרוצים ממנו אנחנו חוזרים חזרה, מטיילים בתוך מבוך. כאן אנחנו עובדים על **גרף מכוון**. כשאנחנו נסוגים מקשת, זה נגד הכיוון שלה. הטיול בגרף מזכיר את Minotaur's Labyrinth, חיפוש באמצעות חוט.

<u>DFS מול BFS:</u>

- המסלולים ש-DFS מוצא הם לא מק"בים (עבור זה יש BFS).
- ב-BFS אנחנו מתקדמים לפי מציאת שכנים במרחק 1, ואז מציאת השכנים של השכנים וכך הלאה. אין דרך קצרה לעבור בין השכנים באותה רמה. אלגוריתם זה **מגלה קודם כל את הצמתים הסמוכים לצומת המקור, לפני שמעמיקים בחיפוש.**
- ב-DFS התחלנו מ-1 והגענו לשכן 2. לא מעניין אותנו שיש ל-1 עוד שכנים, אנו ממשיכים דרך 2 לשכן היחיד שלו 3. כעת אנו נסוגים אחורה ל-2, ומשם ל-1, וממשיכים לשכן אחר. אלגוריתם זה **מעדיף להמשיך ולהעמיק בגרף, תוך התרחקות מצומת המקור, לפני שהוא מחפש עוד צמתים בסביבה הקרובה של צומת המקור.**



זמן הריצה בדומה ל-BFS יהיה $O(m+n)$ כי אנו עוברים על כל קשת ועל כל צומת לכל היותר פעמיים (לפעמים נסוגים על קשת).

<u>האלגוריתם:</u>

- נתאר גרסה רקורסיבית בשם DFS-visit(G,u) אשר מתחילה לסרוק את הגרף מ-u, ומנסה למצוא את כל הצמתים שאפשר להגיע אליהם מ-u, שאי אפשר היה להגיע אליהם מצמתים שכבר סרקנו מהם קודם את הגרף.
- DFS עובר על כל הצמתים, ועבור כל צומת שהוא WHITE (יכול להיות שזה כבר התעדכן בטיולים קודמים), הוא קורא ל-DFS-Visit עליו.
- בדומה ל-BFS, לכל צומת v נשמור:
  - <u>**color** (state)</u> – מצב נוכחי (WHITE – טרם גילינו אותו, GRAY – גילינו אבל לא סרקנו, BLACK – טופל. גילינו ובדקנו).
  - <u>**d** (discovered)</u> – הזמן שבו גילינו את v.
  - <u>**f** (finished)</u> – הזמן שבו סיימנו לטפל ב-v.
  - <u>**π** (prev)</u> – הצומת שממנו v התגלה.

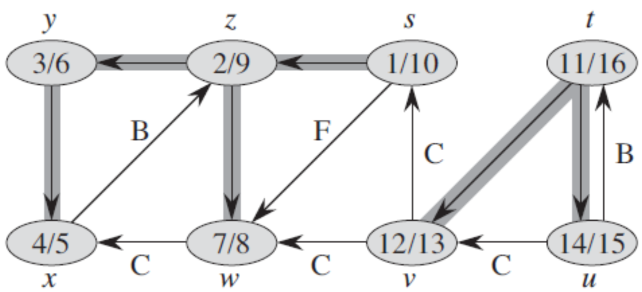
```
DFS(G)
1.  For each vertex u ∈ G.V
2.      u.color = WHITE
3.      u.π = NIL
4.  time = 0
5.  For each vertex u ∈ G.V
6.      if u.color == WHITE
7.          DFS-Visit(G, u)
```

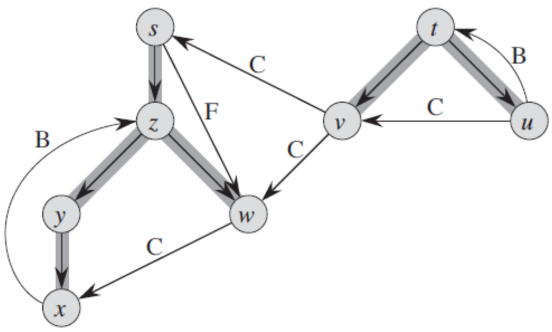
<u>DFS-Visit(G,u):</u>

- צובעים את u ב-GRAY.
- לכל קשת $(u,v) \in E$ נבדוק **האם v הוא WHITE** (לא ביקרנו בו עדיין), ואם כן מיד נתחיל **קריאה רקורסיבית** DFS-Visit(G,v). אם v לא WHITE לא נעשה כלום.
- צובעים את u ב-BLACK.

```
DFS-Visit(G, u)
1.  time = time + 1
2.  u.d = time
3.  u.color = GRAY
4.  For each vertex v ∈ G.Adj[u]
5.      if v.color == WHITE
6.          v.π = u
7.          DFS-Visit(G, v)
8.  u.color = BLACK
9.  time = time + 1
10. u.f = time
```

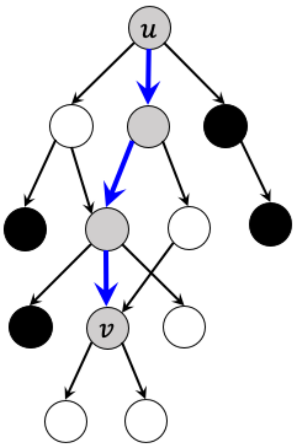
אנו מקבלים בסוף האלגוריתם **יער DFS**, שהוא תלוי בסדר שבו אנו מבצעים את ה-DFS. הסדר הראשון הוא איך אנחנו עוברים על הצמתים בשורה 5 של DFS, והסדר השני נקבע לפי סדר הקשתות של כל צומת מסוים בשורה 4 ב-DFS-Visit.





נכונות:

- עובדה 1: לכל צומת $u \in V$ מתקיים $1 \leq u.d < u.f \leq 2n$. n זה מספר הצמתים בגרף.
- עובדה 2: תת הגרף $G_\pi = (V, E_\pi)$ כאשר $E_\pi = \{(v.\pi, v) | v \in V \text{ and } v.\pi \neq NIL\}$ הוא יער, הנקרא **יער DFS**. זהו אוסף עצי הביקורים המתקבלים מהרצת DFS על הגרף – היער מכיל עץ ביקורים לכל רכיב קשירות בגרף. כל עץ כזה הוא חסר מעגלים כי כל קשת נוצרת כאשר אנו מגיעים לצומת שעדיין לא גילינו.

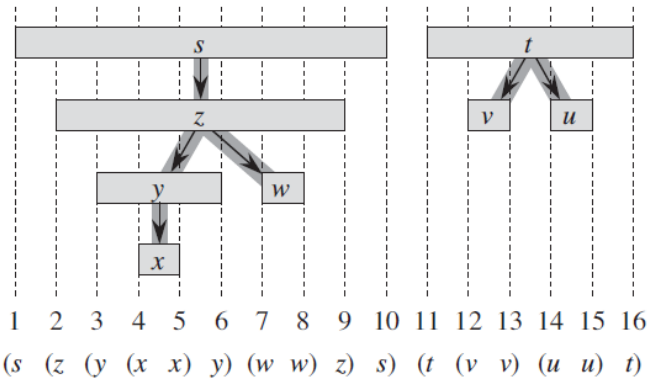


כעת נוכיח שלוש למות שונות הקשורות ל-DFS.

למה 1 (למת המסלול האפור): מסלול אפור הוא **מסלול שכל צמתיו אפורים**. למת המסלול האפור טוענת כי:

1. בכל שלב של ריצת DFS-Visit(G,u), הצמתים האפורים **תמיד מהווים מסלול אפור**.
2. צומת v הוא צאצא של צומת u ⟺ **ברגע ש-v נצבע באפור, יש מסלול אפור מ-u ל-v**.

הוכחה: נוכיח באינדוקציה על מספר הצעדים של האלגוריתם ש: אם הצמתים האפורים, בסדר שבו גילינו אותם, הם $u_1, \dots, u_k$, אז מתקיים כי $u_2.\pi = u_1, \dots u_k.\pi = u_{k-1}$ אז כל הקריאות $Visit(G, u_i)$ פעילות ועוד לא הסתיימו.
בסיס: כשאנחנו רק נכנסים לקריאה הראשונה יש צומת אפור אחד: $u_1 = u, k = 1$.
צעד: בשלב האחרון נסתכל על הקשת (u_k, v). אם v הוא לבן, אז $u_{k+1} = v$ הופך לאפור, והתנאי מתקיים. אחרי שכל הקשתות של u_k נסרקות, הוא הופך לשחור והקריאה מסתיימת. המסלול האפור הוא $u_1, \dots, u_{k-1}$. ברגע ש-v התגלה והפך לאפור, המסלול בין u ל-v הוא מסלול אפור.



למה 2 (למת הסוגריים): לכל שני צמתים $u \neq v \in V$:

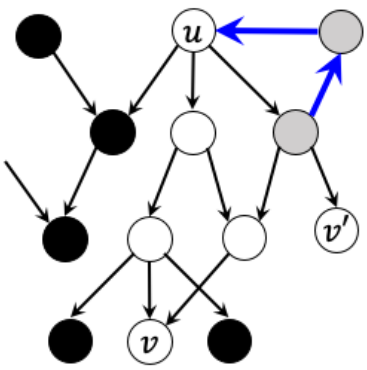
1. v צאצא של u ביער DFS ⟺ $[v.d, v.f] \subset [u.d, u.f]$.
2. u ו-v אינם קשורים ⟺ $[v.d, v.f] \cap [u.d, u.f] = \emptyset$.

כלומר, עבור שני צמתים, או שהקטע של אחד מוכל בשני (כאשר הראשון הוא צאצא של השני), או שהקטעים הם זרים (אין ביניהם שום קשר, אף אחד הוא לא אב קדמון של השני).

הוכחה: יהיו $u \neq v \in V$ כך ש-$u.d < v.d$ (בה"כ, גילינו קודם את u). נחלק למקרים:

- מתקיים $v.d < u.f$: כשגילינו את v, אז u עדיין אפור. לכן, לפי למת המסלול האפור, v הוא צאצא של u. כיוון שקראנו ל-DFS-Visit(G,v) כאשר DFS-Visit(G,u) עדיין פעילה, מתקיים $v.f < u.f$ ולכן $[v.d, v.f] \subset [u.d, u.f]$.
- מתקיים $u.f < v.d$: במקרה זה מתקיים $u.d < u.f < v.d < v.f$ ולכן $[v.d, v.f] \cap [u.d, u.f] = \emptyset$. לפי למת המסלול האפור, v אינו צאצא של u.

למה 3 (למת המסלול הלבן): מסלול לבן הוא **מסלול שכל צמתיו לבנים**. למת המסלול הלבן טוענת כי לכל שני צמתים $u, v \in V$: v הוא צאצא של u ביער DFS ⟺ יש מסלול לבן מ-u ל-v ב-G כאשר אנו מתחילים את הקריאה של DFS-Visit(G,u).



הוכחה:

⇐: נניח ש-v הוא צאצא של u. יהי $u = u_0, u_1, \dots, u_k = v$ המסלול מ-u ל-v. לפי למת המסלול האפור, כאשר v נצבע באפור, כל הצמתים האלו אפורים ולכן: $u_0.d < \dots < u_k.d$. לכן, כאשר נקרא ל-DFS-Visit(G,u) כל הצמתים על המסלול הם לבנים.

⇒: יהי $u = u_0, u_1, \dots, u_k = v$ מסלול לבן מ-u ל-v כאשר נקרא ל-DFS-Visit(G,u). נוכיח באינדוקציה על i ש-u_i הוא צאצא של u (כשנגיע ל-$i = k$ הטענה נובעת). במקרה הבסיס מתקיים $u_0 = u$. נניח שהיא מתקיימת עבור i. בשלב כלשהו נסרוק את הקשת (u_i, u_{i+1}):

- אם u_{i+1} לבן, אז הוא הופך להיות הילד של u_i.
- אם u_{i+1} אפור, אז הוא צאצא של u לפי למת המסלול האפור.
- אם u_{i+1} שחור, אז $u.d < u_{i+1}.d < u_{i+1}.f < u.f$ (כלומר מתקיים $[u_{i+1}.d, u_{i+1}.f] \subset [u.d, u.f]$) והוא צאצא של u לפי למת הסוגריים.

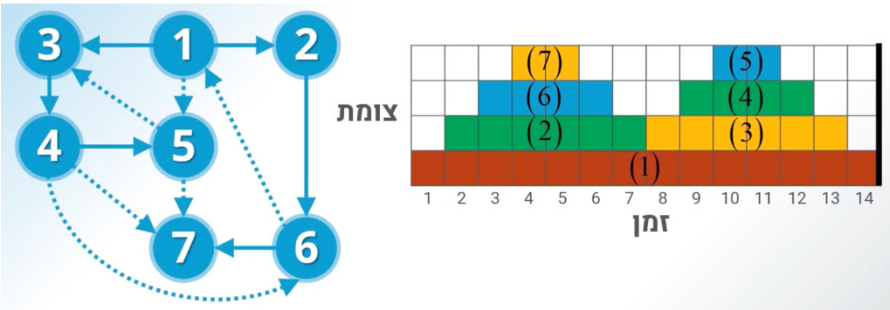
<u>סוגים שונים של קשתות:</u>

סוג	הסבר	עבור קשת $(u,v)\in E$	סיווג תוך כדי DFS
קשתות עץ (tree edges)	קשתות בגרף שהובילו במהלך הרצה של DFS **לגילוי צמתים חדשים**. אלו הן קשתות הכלולות באחד מהעצים ביער הביקורים.	$u=v.\pi$	אם v **לבן**
קשתות אחוריות (back edges) **כולל loop.**	קשתות בגרף שהובילו במהלך הרצה של DFS **לצמתים שכבר ביקרנו בהם**. אלו הן קשתות המובילות מצומת אחד לאחר שהוא **אב קדמון** שלו בעץ הביקורים.	u הוא צאצא של v	אם v **אפור** $(v.d<u.d)$ לפי למת המסלול האפור.
קשתות קדמיות (forward edges)	אלו הן קשתות המובילות מצומת אחד לאחר שהוא **צאצא לא ישיר** שלו בעץ הביקורים (צומת שכבר התגלה במהלך הסריקה דרך שכן אחר).	v הוא צאצא של u ו-$u\neq v.\pi$	אם v **שחור** וגם $v.d>u.d$ (כיוון ש-u אפור אז $v.f<u.f$ ומשם לפי למת הסוגריים)
קשתות חוצות (cross edges)	קשתות המובילות מצומת אחד לצומת אחר **שאיננו צאצא ואיננו אב קדמון שלו.**	אין קשר בין הצמתים	אם v **שחור** וגם $v.f<u.d$

נוכל בקלות לזהות את הקשתות לפי זמני הגילוי והעזיבה של הצמתים:

תהי (u,v) קשת בגרף:

- עבור קשת עץ/קדמית: $\boldsymbol{u}\to v\leftrightarrow v\leftarrow \boldsymbol{u}$
- עבור קשת חוצה: $v\leftrightarrow v\,\boldsymbol{u}\leftrightarrow \boldsymbol{u}$
- עבור קשת אחורית: $v\to \boldsymbol{u}\leftrightarrow \boldsymbol{u}\leftarrow v$
- קשת שאינה קיימת בגרף: $\boldsymbol{u}\leftrightarrow \boldsymbol{u}\,v\leftrightarrow v$
  - אם הקשת הייתה קיימת, אז v הוא אחד מהילדים של u (בהינתן זמני הגילוי והעזיבה), ואז צריך להתקיים כי $v.d<u.f$ בניגוד לנתון.



<u>DFS בגרף לא מכוון:</u> אפשר להריץ DFS גם על גרף לא מכוון. כל קשת זה זוג לא סדור של צמתים $\{u,v\}$, ולכן היא נבדקת פעמיים: פעם אחת מ-u ופעם אחת מ-v. הסיווג של הקשת **נקבע בפעם הראשונה שבדקנו אותה**. לכן, כל הקשתות הן קשתות עץ או קשתות אחוריות (אין כאן קשתות קדמיות וקשתות חוצות).

DAG

<u>DAG (Directed Acyclic Graph)</u> – גרף מכוון $G=(V,E)$ הוא חסר מעגלים (נקרא גם DAG) אם הוא **לא מכיל שום מעגלים מכוונים**. גרף כזה **מתאר קשרי תלות**.

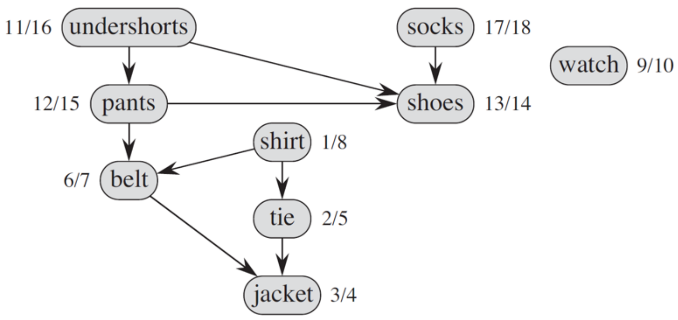
<u>מיון טופולוגי:</u> מיון טופולוגי של גרף מכוון $G=(V,E)$ הוא סידור $>$ של הצמתים כך ש-$(u,v)\in E$ גורר $u<v$. באופן שקול, זו פונקציה חד ערכית $\{1,2,\dots,|V|\}\to f:V$ כך ש-$(u,v)\in E$ גורר $f(u)<f(v)$. כלומר, זהו **סידור של הצמתים באופן כזה שכל קשת בגרף מכוונת מצומת אחד u לצומת אחר v המופיע מאוחר יותר במיון.**

<u>טענה 1:</u> גרף מכוון הוא DAG ⇔ יש לו מיון טופולוגי.

<u>⇒:</u> אם לגרף יש מיון טופולוגי, אז אין לו מעגלים ולכן הוא DAG.

<u>⇐:</u> נניח באינדוקציה על גודל G (מספר הצמתים בו), שאם הוא DAG אז יש לו מיון טופולוגי. אם G הוא DAG, אז חייב להיות צומת אחד s שאין לו קשתות נכנסות (אחרת יכולנו ללכת אחורה, והיה קיים מעגל). הגרף $G\backslash\{s\}$ המתקבל על ידי זריקת s וכל הקשתות שלו, הוא DAG כי אין לו מעגלים. באינדוקציה לגרף הנ"ל יש מיון טופולוגי. הוספת s כצומת הראשון נותן מיון טופולוגי של G.

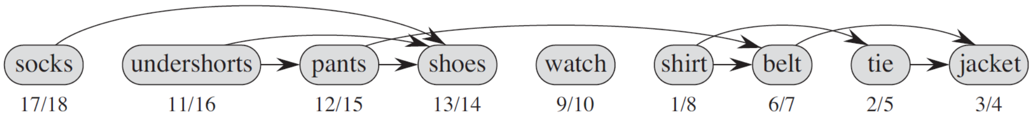
<u>האלגוריתם של Knuth:</u> ניתן לתרגם את ההוכחה לאלגוריתם בזמן $O(m+n)$ שבהינתן גרף מכוון **מוצא מעגל או מיון טופולוגי**. נתחזק את **דרגת הכניסה** של כל צומת (נעבור על כל הקשתות ונעדכן בהתאם כאשר קשת נכנסת לצומת כלשהו). נאתחל רשימה של הצמתים שדרגת הכניסה שלהם 0. אם הרשימה ריקה – יש **מעגל**. אחרת, ניקח צומת מהרשימה, נשים אותו כצומת ראשון במיון טופולוגי, נעבור על כל הקשתות שיוצאות ממנו ומפחיתים על כל קשת שיוצאת מפחיתים את דרגת הכניסה של הצמתים המתאימים. אם דרגת הכניסה ירדה ל-0, מוסיפים אותם לרשימה.



טענה 2 (DAG ומיון טופולוגי באמצעות DFS): גרף מכוון $G = (V, E)$ הוא DAG $\Leftrightarrow$ אין קשתות אחוריות במהלך ריצת DFS. אם G הוא DAG ומתקיים $u_1.f > \dots > u_n.f$ אז $u_1, \dots, u_n$ הוא מיון טופולוגי של G.

תזכורת: בכל המקרים של קשת עץ/קדמית/חוצה מתקיים $v.f < u.f$. רק במקרה של קשת אחורית מתקיים $u.f < v.f$.

הוכחה: אם קיימת קשת אחורית אז יש מעגל (נלך אחורה לפי המצביעים עד שנגיע ל-u) ולכן הגרף לא חסר מעגלים. בעת נראה כי אם אין קשתות אחוריות (הגרף הוא DAG), אז לכל קשת מתקיים $v.f < u.f$, ואם **נסדר את הצמתים בסדר הפוך לזמני הסיום שלהם, נקבל מיון טופולוגי** (הערה: מיון טופולוגי אינו יחיד!)



SCC (רק"חים)

SCC (רכיבי קשירות חזקה):

SCC (רק"ח): בהינתן גרף מכוון $G = (V, E)$, רק"ח הוא קבוצה **מקסימלית** של צמתים $C \subseteq V$ בגרף כך **שעבור כל שני צמתים $u, v \in C$ יש מסלול מכוון** מ-u ל-v (ומסלול מכוון מ-v ל-u).

למה: את הצמתים של גרף מכוון **ניתן לפרק לרק"חים זרים**. שני צמתים $u, v \in V$ הם באותו רק"ח (הם קשירים) $\Leftrightarrow$ יש מסלול מכוון מ-u ל-v ומסלול מכוון מ-v ל-u.

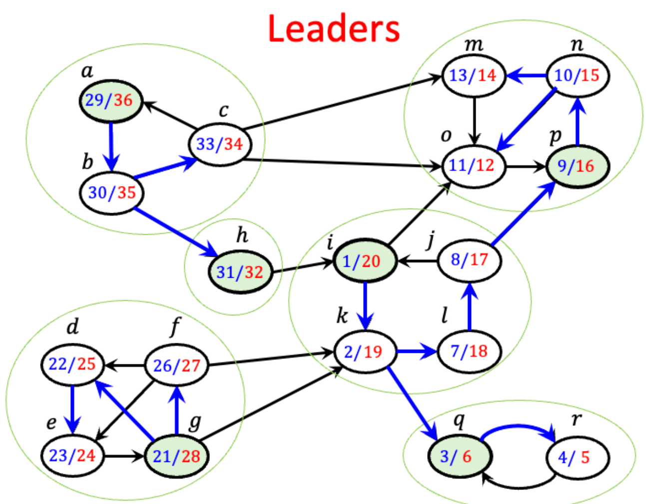
גרף רק"חים (גרף העל): נתייחס לכל רק"ח בתור צומת. **יש קשת בין שני רק"חים אם יש קשת בגרף המקורי מצומת כלשהו ברק"ח הראשון לצומת כלשהו ברק"ח השני.**

יהיו $C_1, \dots, C_k$ רק"חים של גרף G, נסמן את גרף הרק"חים באופן הבא: $\boldsymbol{G^{SCC} = (V^{SCC}, E^{SCC})}$. מתקיים כי $V^{SCC} = \{C_1, \dots, C_k\}$ וכי $(C, C') \in E^{SCC}$ אמ"מ $C \neq C'$ וקיימת קשת $(u, v) \in E$ כך ש-$u \in C \wedge v \in C'$.

למה: לכל גרף G, הגרף G^{SCC} הוא DAG.

הוכחה: אם היה מעגל זה אומר שזו לא חלוקה נכונה לרק"חים, אחרת הצמתים במעגל היו צריכים להיות באותו רק"ח.

פירוק גרף מכוון לרק"חים:



נרצה להיעזר ב-DFS למטרה זו. כאשר אנו מריצים DFS (ובוחרים קשתות בסדר שרירותי), אין סיבה שנישאר בהכרח בתהליך הגילוי שלנו באותו רכיב קשירות, לא עברנו רק על הצמתים ברק"ח של צומת המקור s שהתחלנו ממנו, וגם לא בסדר מסוים.

- מנהיג (Leader) – בכל רק"ח המנהיג הוא הצומת הראשון שאנחנו מגלים באותו רק"ח, כלומר עם $u.d$ מינימלי. **קשה לדעת תוך כדי הריצה מי הם המנהיגים**, כיוון שאנחנו לא יודעים מתי אנחנו ברק"ח חדש. קיבלנו יער DFS בסיום האלגוריתם, ואנחנו יודעים **שכל שורשי העצים הם מנהיגים, אבל יש מנהיגים שהם אינם שורשים.**
- איך מזהים מנהיג? צומת $u \in V$ עם $u.f$ מקסימלי ברק"ח. עבור רק"ח C נסמן $f(C) = \max_{v \in C} v.f = leader.f$.
- נשים לב כי בתוך כל רק"ח חייבת להיות **לפחות קשת אחורית אחת** (אם לא היו קשתות אחוריות אז לא היה כאן DAG, לא היו מעגלים ולא היה מסלול בין כל שני צמתים בתוך רכיב הקשירות).
- מונהג (Follower) – כל צומת שאינו מנהיג בתוך רק"ח הוא מונהג.

למה: המנהיג u ברק"ח C הוא אב קדמון של כל הצמתים ב-C. לכן, u הוא גם הצומת ב-C עם זמן העזיבה הגדול ביותר $u.f$. בנוסף, לכל צומת $v \in C$ כל הצמתים במסלול מ-u ל-v הם גם ב-C.

הוכחה: המנהיג הוא אב קדמון של כל שאר הצמתים, כיוון שכל שאר הצמתים הם לבנים, והם באותו רק"ח לכן לפי **למת המסלול הלבן** כל שאר הצמתים הופכים להיות צאצאים של המנהיג. כיוון שהמנהיג הוא אב קדמון, לפי **למת הסוגריים** הוא יהיה לא רק הצומת שמגלים ראשון אלא הצומת שגומרים לטפל בו אחרון.
בנוסף, כל הצמתים במסלול מהמנהיג לכל צומת אחר ברק"ח חייבים להיות ברק"ח. אם צומת w נמצא על המסלול מ-u ל-v (לאו דווקא מסלול בעץ), מכיוון ש-u ו-v באותו רק"ח, יש גם מסלול חזרה מ-v ל-u. אז כל צומת w נמצא בעצם במעגל עם u, ולכן הוא גם שייך לרק"ח של u.

למה: יהיו C_1, C_2 שני רק"חים זרים של G. אם קיימת קשת $(u,v) \in E$ כאשר $u \in C_1, v \in C_2$ אז $f(C_1) > f(C_2)$. כלומר, זמן העזיבה המקסימלי ב-C_1 גדול יותר מזמן העזיבה המקסימלי ב-C_2.

הוכחה: נסמן u', v' כמנהיגים של C_1, C_2. קיים מסלול $u' \to u \to v \to v'$ שנמצא בתוך האיחוד $C_1 \cup C_2$.

- מקרה ראשון – גילינו קודם את u'. לפי הגדרת המנהיגים ולמת המסלול הלבן u' (המנהיג של C_1) **הוא אב קדמון** של v' (המנהיג של C_2 – לפני שגילינו אותו כל הצמתים ברק"ח לבנים). לפי למת הסוגריים, $f(C_1) = u'.f > v'.f = f(C_2)$.
- מקרה שני – גילינו קודם את v'. u' הוא לא אב קדמון של v' כיוון שאנו מגלים את v' קודם. לא יכול להיות גם כי v' הוא לא אב קדמון של u' כי אם זה היה מתקיים – היה מסלול $v' \to u'$ וביחד עם המסלול הקודם הם היו באותו רק"ח, בסתירה. לכן הם **unrelated** ולפי למת הסוגריים כיוון ש-$u'.d > v'.d$ אז $f(C_1) = u'.f > v'.f = f(C_2)$.

מסקנה (מיון טופולוגי של רק"חים): אם $C_1, \ldots, C_k$ הם הרק"חים של G, **מסודרים בסדר יורד של זמני סיום** $f(C_1) > \cdots > f(C_k)$ אז $C_1, \ldots, C_k$ הם מיון טופולוגי של G^{SCC}.

הגרף ההופכי: עבור גרף $G = (V,E)$, ההופכי $G^R = (V, E^R)$ מתקבל ע"י הפיכת כל הקשתות: $E^R = \{(v,u) | (u,v) \in E\}$.
נשים לב כי **לשני הגרפים G, G^R יש את אותם רק"חים**.

פירוק גרף לרק"חים

הרעיון: איך נמצא רק"ח ראשון?

- הצומת $u \in V$ עם $u.f$ מקסימלי הוא המנהיג של הרק"ח הראשון שנסמנו C_1.
- איך נמצא את כל שאר הצמתים ב-C_1 (המונהגים)? עץ ה-DFS ששורשו הוא u יכול להכיל, כפי שראינו, צמתים מרק"חים אחרים. לפי הלמה הקודמת, C_1 הוא הרק"ח הראשון במיון טופולוגי על G^{SCC}.
- לכן, **הצמתים היחידים שאפשר להגיע מהם ל-u הם הצמתים של C_1** (במילים אחרות, C_1 הוא צומת בור – אין אף קשתות נכנסות אליו מרק"חים אחרים). למה לא צומת מקור? אם אף קשתות לא נכנסות אליו – הוא מקור!
- אם כך, כדי למצוא את כל הצמתים ברק"ח C_1 צריך להריץ פשוט DFS מ-u על הגרף ההפוך G^R.

כדי למצוא את הרק"ח השני – מבין כל הצמתים שהם לא ב-C_1 ניקח את הצומת שזמן העזיבה שלו הכי גדול ונחזור על התהליך.

אלגוריתם (Kosaraju-Sharir):

- נריץ DFS על הגרף G (נבחר צמתים בסדר שרירותי).
- נריץ DFS שני על הגרף G^R, כאשר נבחר להריץ DFS על הצמתים בסדר יורד לפי זמני העזיבה שלהם (הנתונים לנו מהרצת ה-DFS הראשון). כלומר, נתחיל מהצומת עם זמן העזיבה הגדול ביותר שהתקבל מ-DFS על G (מנהיג הרק"ח הראשון שהוא בור ב-G).
- כעת נוכל לחזור ליער ה-DFS שהתקבל מהגרף G, ולמחוק קשתות נכנסות למנהיגים שמצאנו. נקבל תת-יער שבו **כל עץ מתאים לרק"ח**.

זמן הריצה הוא כמובן $\boldsymbol{O(m+n)}$.

נכונות:

טענה: כל צומת u שעליו מבצעים $\text{DFS-Visit}(G^R, u)$ הוא מנהיג של הרק"ח C, ועץ ה-DFS המתקבל שהוא שורשו, מכיל בדיוק את הצמתים הנמצאים ב-C. בנוסף, הסדר שבו מגלים את הרק"חים הוא מיון טופולוגי של G^{SCC}.

הוכחה: יהיו $u_1, \ldots, u_k$ הצמתים שעליהם מתבצעת קריאת $\text{DFS-Visit}(G^R, u)$ בסדר הזה. יהיו $C_1, \ldots, C_k$ קבוצות הצמתים בעצי ה-DFS של הצמתים $u_1, \ldots, u_k$ (עצי DFS בגרף ההפוך). נוכיח באינדוקציה על i כי C_i הוא רק"ח שמנהיגו u_i.

- נניח שעבור $i = 1$ זה מתקיים וגם עבור $i - 1$ ונוכיח עבור i (נשלב את הבסיס ואת ה"א). נסמן $C_{1,i} = C_1 \cup \ldots \cup C_i$.
- כיוון שאנחנו עוברים על הצמתים בסדר יורד של זמני עזיבה, ולפי הנחת האינדוקציה הצמתים שאנחנו מגלים ב-$i-1$ הקריאות הראשונות הם כל הצמתים של $C_{1,i-1}$ אז u_i הוא הצומת ב-$V \setminus C_{1,i-1}$ עם זמן עזיבה מקסימלי.

- כיוון שהורדנו רק"חים שלמים, u_i חייב להיות מנהיג של רק"ח C_i. לפי הלמה שמתנה לנו את המיון הטופולוגי, אין קשתות מ-$V \setminus C_{1,i-1}$ ל-C_i, כי כל הצמתים ב-$V \setminus C_{1,i-1}$ הם רק"חים עם זמני עזיבה קטנים יותר.
- יכולות להיות קשתות (בגרף המקורי) מ-C_{i-1} ל-C_i, אבל כל הצמתים ב-$C_{1,,i-1}$ הם כבר שחורים, ולכן לא תהיה למעבר בקשתות אלו השפעה.
- כיוון ש-C_i הוא רק"ח יש מסלולים מכל הצמתים ב-C_i ל-u_i.
- לכן, DFS-Visit(G^R, u) מגלה בדיוק את כל הצמתים שנמצאים ב-C_i.

אלגוריתם Tarjan: מציאת רק"חים באמצעות ריצה אחת יחידה של DFS. האתגר העיקרי הוא, כאשר אנו עוזבים צומת לקבוע האם הוא מנהיג או מונהג.

- כדי להשיג את המטרה זו, נצטרך לתחזק גם **מחסנית** (נוסף למחסנית הרקורסיה של DFS).
- לכל צומת $u \in V$ נתחזק שדה נוסף $u.low$ שהוא זמן הגילוי המינימלי $v.d$ (באופן לא מדויק, זו האינטואיציה) של צומת v שאפשר להגיע אליו מ-u, **בלי לעזוב את הרק"ח** של u.
- צומת $u \in V$ הוא מנהיג אם כאשר נסיים איתו יתקיים $u.low = u.d$.
- מסיבה טכנית, כאשר מסיימים לטפל ברק"ח, אז נבצע $u.low = \infty$ לכל צומת u ברק"ח.

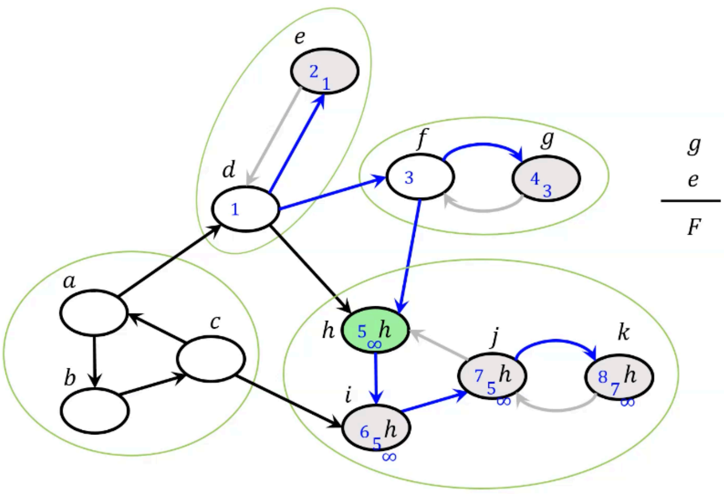
האלגוריתם:

- בפונקציה SCC ראשית נאתחל את כל השדות: שדה lead (**האם הצומת עדיין יכול להיות מנהיג**). האלגוריתם לא משתמש בזמני ההגעה והעזיבה לצמתים, נעבוד רק עם low. נאתחל את F להיות מחסנית ריקה של המונהגים.
- נעדכן את DFS-Visit: נציב את הזמן העדכני ב-u.low. אם $v.low == 0$ זה צומת לבן וחדש, ואנחנו קוראים ל-DFS.
- אחרי שהסתכלנו על קשת (u, v) – בדקנו את הקשת, או שנסוגנו עליה: כאן אנו **מגיעים לשורה 7 שהיא לב העניין**. אם $v.low$ נמוך, זה אומר שהצלחנו להגיע מ-v לצומת עם זמן גילוי נמוך (**צומת v הוא יותר טוב**). זה אומר **שנצליח גם מ-u ולכן הוא לא יהיה מנהיג – אלא מונהג**.
- Comp נקראת כשעומדים לסגת מ-u ואנחנו יודעים שהוא מנהיג, אז אנחנו רוצים לזהות את כל הצמתים ברק"ח שלו. כל הצמתים שמונהגים על ידי u נמצאים בראש המחסנית. אבל, אולי זה כולל צמתים גם של רק"חים אחרים. ניקח רק את הצמתים שה-low שלהם גדול או שווה ל-low של u.

```
SCC(G)
1.  For each u ∈ G.V
2.      u.low = 0
3.      u.lead = true
4.      u.π = NIL
5.  time = 0
6.  F = Stack()
7.  For each u ∈ G.V
8.      if u.low == 0
9.          DFS-Visit(G, u)

Comp(u)
1.  while F ≠ ∅
      and F.top().low ≥ u.low
2.      v = F.pop()
3.      v.ptr = u
4.      v.low = ∞
5.  u.ptr = u
6.  u.low = ∞
7.  u.π = NIL
```

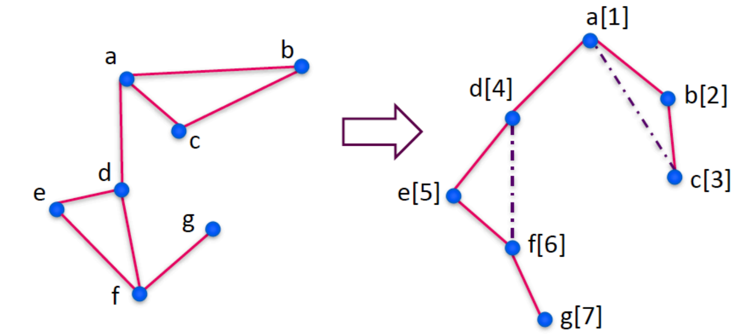
```
DFS-Visit(G, u)
1.  time = time + 1
2.  u.low = time
3.  For each v ∈ u.Adj
4.      if v.low == 0
5.          v.π = u
6.          DFS-Visit(G, v)
7.      if v.low < u.low
8.          u.low = v.low
9.          u.lead = false
10. if u.lead == true
11.     Comp(u)
12. else
13.     F.push(u)
```



סדר הגילוי של הרק"חים באלגוריתם זה הוא מנוגד לסדר הגילוי באלגוריתם Kosaraju-Sharir (סדר יורד של זמני סיום). כאן מדובר **בסדר עולה** של זמני סיום: $f(C_1) < \cdots < f(C_k)$.

DFS (תרגול 3)

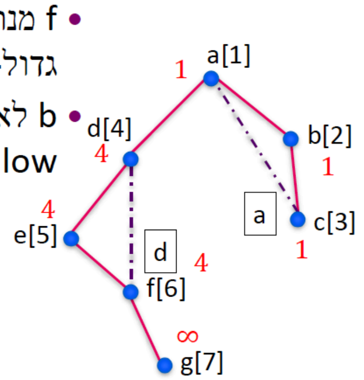
1) קודקודים מנתקים: קודקוד בגרף לא מכוון נקרא קודקוד מנתק אם הסרתו (יחד עם הקשתות הסמוכות אליו) מגדילה את מספר רכיבי הקשירות של הגרף. נתון גרף לא מכוון $G = (V, E)$. תארו אלגוריתם יעיל למציאת כל הקודקודים המנתקים שלו.



הרעיון:

- **אב חורג**: ביחס ליער DFS, אם יוצאת מקודקוד v קשת אחורית אל קודקוד w נגיד ש-w אב חורג של v.
- נריץ DFS על G לקבלת יער DFS, ונשמור את זמני הגילוי של כל צומת. כיוון שהגרף לא מכוון נקבל קשתות משני סוגים בלבד: קשתות עץ וקשתות אחוריות. נניח בה"כ שהגרף קשיר ולכן היער מכיל עץ יחיד.

אבחנות:



- f מנתק כי יש לו בן שערך ה-low שלו גדול-שווה מזמן הגילוי של f (6).
- b לא מנתק כי לכל בן שלו ערך ה-low שלו קטן מזמן הגילוי של b (2).

- עלים אף פעם לא מנתקים.
- שורש העץ לא מנתק ⇔ יש לו לכל היותר בן אחד.
- צומת u (לא שורש) לא מנתק ⇔ **לכל** בן שלו v, יש צאצא עם אב חורג w שמקיים $w.d < u.d$ (גילינו את w לפני u). איך נבדוק את קיום התנאי לכל צומת: נגדיר **ערך עזר low לכל צומת v, שהוא זמן הגילוי d של צומת w שהוא האב החורג הכי מוקדם של צאצא של v** (מי הקודקוד הכי גבוה בהיררכיה שאפשר להגיע אליו מתחת-העץ של v בעזרת קשת אחורית). אם אין אף w כזה נקבע $v.low = \infty$.
- באופן שקול: צומת u (לא שורש) לא מנתק ⇔ לכל בן שלו v מתקיים $v.low < u.d$.
- נחשב את הערך low לכל צומת באופן הבא:

$$low[v] = \min \begin{cases} \mathrm{d[w]} & ; \quad w \text{ אב חורג של } v \\ low[v'] & ; \quad v' \text{ בן של } v \end{cases}$$

- נשדרג את DFS ונוסיף לו את חישוב ה-low. נאתחל את הערך להיות ∞ בבולם, ולכל קודקוד נחשב את ה-low באמצעות המינימום שתואר.

אלגוריתם:

1. מריצים DFS משודרג כדי לקבל יער DFS, ולכל צומת את ערך ה-low שלו.
2. עוברים על כל צומת u ובודקים אם יש לו שכן v כך שהוא בן של u, וגם $v.low \geq u.d$. אם כן – מסמנים ש-u מנתק.
3. אם לשורש של עץ יש יותר מבן אחד מסמנים שהוא מנתק, אחרת שלא.

נכונות:

טענה: הסרת קודקוד u **לא מנתקת** בן שלו v (מהשורש) ⇔ יש ל-v צאצא עם אב חורג w כך שמתקיים: $\boldsymbol{w.d < u.d}$.

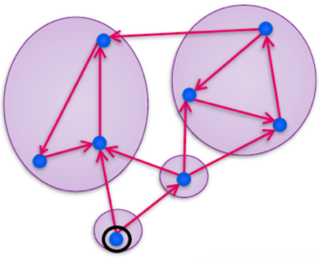
⇐: נניח של-v **אין** צאצא עם אב חורג כך שמתקיים $w.d < u.d$ (ונראה שהסרת קודקוד u **בן מנתקת**). נסתכל על קבוצת הצמתים T_v שמכילה את צמתי תת-העץ ששורשו v. לאחר מחיקת u, יוצאות מצמתי T_v קשתות עץ (שנכנסות לצומת ב-T_v) וקשתות אחוריות שיוצאות אל אב קדמון שלפי ההנחה הוא u או (נמחק) או צאצא של u (בהכרח נמצא גם ב-T_v). לכן כל הקשתות שיוצאות מ-T_v נכנסות ל-T_v ולכן הסרת הקודקוד u מנתקת את v מהשורש.

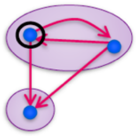
⇒: נניח של-v יש צאצא עם אב חורג w כך שמתקיים $w.d < u.d$. אז יש מסלול $s \to \cdots \to w \to v' \to \cdots \to v$ (v' זה צאצא של v, ו-s זה השורש) שלא עובר דרך u ולכן v לא מתנתק. מתקיים $u < v < v'$ ולכן המסלול לא עובר ב-u. הקשת $\{v', w\}$ נשארת גם לאחר מחיקת הצומת u.

סיבוכיות:

הרצת DFS משודרג עולה $O(|V| + |E|)$. מעבר על כל הצמתים לסימון מנתקים עולה $O(|V| + |E|)$. סה"כ $O(|V| + |E|)$.

2) קבוצת מוצא: קבוצת קודקודים S בגרף מכוון $G = (V, E)$ נקראת קבוצת מוצא אם **יש ממנה מסלול לכל קודקוד בגרף**. כלומר מתקיים לכל צומת $v \in V$ קיים $s \in S$ כך שיש מסלול $s \to \cdots \to v$ (כל קודקוד s בפרט מגיע לעצמו). נתון גרף מכוון $G = (V, E)$. תארו אלגוריתם למציאת קבוצת מוצא **מינימלית**.





אלגוריתם:

1. נחשב רק"חים (כפי שראינו בשיעור – DFS, הפיכת קשתות לגרף ההופכי, ושוב DFS), וניצור את גרף העל.
2. נבחר קודקוד אחד מכל רק"ח (אפשר באמצעותו להגיע לכל קודקוד אחר ברק"ח בשל הקשירות), שאין אף קשת שנכנסת אליו, נכנה כזה רק"ח **מקור**.

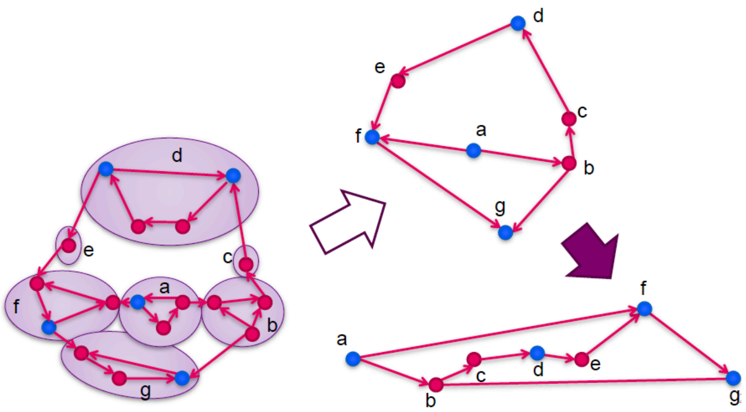
נכונות:

- נניח שמצאנו m מקורות כאלו. כל קבוצת מוצא תכיל לפחות צומת אחד מכל מקור (אחרת לא היינו יכולים להגיע לרק"ח הזה שהגדרנו כמקור).
- מצד שני, לכל רק"ח אחר, **בהכרח ניתן להגיע מרק"ח ללא קשתות נכנסות**. לכן, אין טעם להוסיף שום קודקוד אחר.
- כלומר כל קבוצת מוצא היא בגודל לפחות m ומצאנו קבוצה בגודל בדיוק m ולכן זו **קבוצה מינימלית**.

סיבוכיות:

חישוב רק"חים, בניית גרף העל עולים $O(|V| + |E|)$. בחירת הצמתים לקבוצת המוצא היא $O(|V|)$. סה"כ $O(|V| + |E|)$.

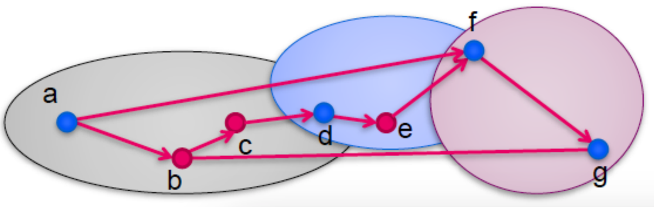
3) תחנות במסלול: נתונים גרף מכוון $G = (V, E)$ וקבוצה $S \subseteq V$. תארו אלגוריתם אשר בודק האם קיים בגרף מסלול העובר דרך כל קודקודי S. המסלול יכול לעבור בקודקודים נוספים ואף לחצות את אותה קשת מספר פעמים.



אלגוריתם:

1. נחשב רק"חים.
2. נחפש בגרף העל מסלול אשר עובר דרך כל הרק"חים המכילים לפחות איבר אחד מ-S (כחולים). נמיין את הגרף **טופולוגית**.
3. נתקדם על פי הסדר הטופולוגי: בכל פעם שנגיע לאיבר המייצג רק"ח רלוונטי v (כחול) נבדוק האם קיים מסלול ממנו אל הרק"ח הרלוונטי u (הכחול) הבא – **בסדר הטופולוגי**. חיפוש מסלול זה יתבצע באמצעות BFS מ-v אל u על תת-הגרף שמושרה בין v ל-u בסדר הטופולוגי.

נכונות: קיים הילוך ⇔ קיים מסלול כנדרש בגרף הרק"חים.



⇐: אם יש הילוך שעובר דרך כל קודקודי S אז קיים "מסלול מושרה" בגרף העל (שמשתמש רק בקשתות שיוצרות קשתות בין רק"חים).

⇒: אם קיים מסלול כנדרש בגרף העל, וכיוון שניתן להגיע מכל קודקוד לכל קודקוד אחר ברק"ח, ניתן להרחיב אותו להילוך מתאים שעובר דרך כל קודקודי S (ניתן לחזור על קשתות).

סיבוכיות:

מציאת רק"חים ומיון טופולוגי, ומציאת רק"חים רלוונטיים (כחולים) עולה $O(|V| + |E|)$.

כיוון שתמיד אנו עושים BFS על תתי-הגרפים, זמן הריצה הכולל יהיה $O(|V| + |E|)$. יש קודקודים שנחתכים בין הרצות שכנות של BFS, ולכן נקבל את ה-2 בחישוב:

- כל קודקוד משתתף לכל היותר בשתי ריצות BFS.
- כל קשת משתתפת לכל היותר בריצת BFS אחת.
- סך זמן הריצה של כל פעולות ה-BFS – $O(|V|+|E|)$.

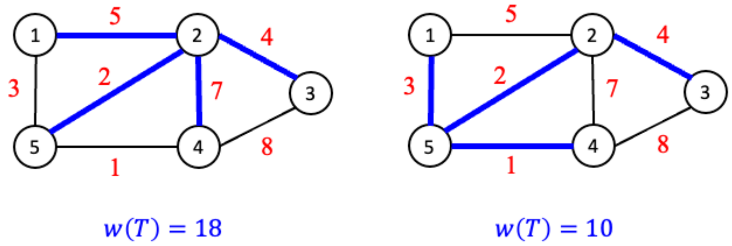
$$(|V_1|+|E_1|)+(|V_2|+|E_2|)+\ldots \le 2(|V|+|E|)$$

$$O(|V_1|+|E_1|)+O(|V_2|+|E_2|)+\ldots = O(|V|+|E|)$$

עצים פורשים מינימליים

MST (עפ"מים)

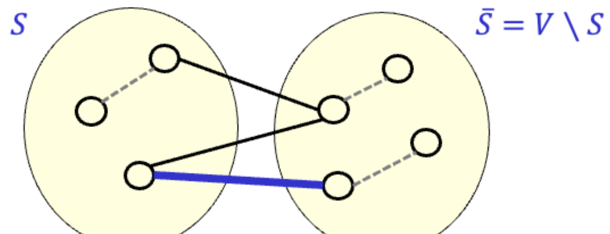
MST (עצים פורשים מינימליים):



עץ פורש: עץ פורש של גרף לא מכוון $G = (V, E)$ הוא תת-גרף $T = (V, E_T)$ של G שהוא עץ. העץ **פורש את כל צמתי הגרף**.

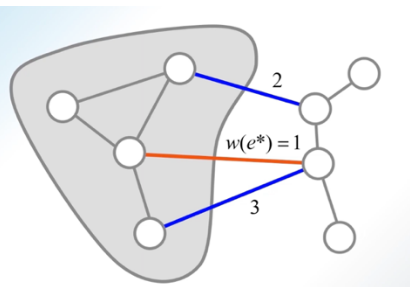
MST (עפ"מ): **עץ פורש מינימלי** של גרף קשיר, ממושקל, ולא מכוון $G = (V, E, w)$ הוא תת-גרף $T = (V, E_T)$ של G שהוא עץ כך **שסכום המשקלים של הקשתות בו הוא מינימלי**: $w(T) = \sum_{e \in E_T} w(e)$.

המסגרת הכללית לאלגוריתם: נניח שמצאנו קבוצת קשתות $A \subseteq E$ שמוכלת באיזשהו עפ"מ. תחילה $A = \emptyset$. כל עוד יש פחות מ-$n - 1$ קשתות (זה לא עץ פורש), נמצא קשת $e \in E$ כך שאם נוסיף אותה לקבוצה, גם הקבוצה החדשה תהיה מוכלת בעפ"מ. כאשר האלגוריתם יסתיים נקבל קבוצה עם $n - 1$ קשתות, שמוכלת בעפ"מ והיא בעצמה עץ פורש – לכן קיבלנו עפ"מ. **איך נמצא את הקשת שעלינו להוסיף לקבוצה?**



חתך: חתך הינו חלוקה של קבוצת הקודקודים V של גרף $G = (V, E)$ לשתי קבוצות $(S, V \backslash S)$ כאשר $\emptyset \neq S \subset V$. חלוקה זו מגדירה **קשתות אשר חוצות** את החתך: $E(S, S^C) = \{\{u, v\} \in E | u \in S, v \in V \backslash S\}$.

משפט החתך: תהי $E \supseteq A$ קבוצת קשתות המוכלת בעפ"מ כלשהו. נניח כי (S, S^C) חתך שאף אחד מהקשתות שחוצות אותו לא ב-A. תהי $e \in E(S, S^C)$ **קשת בעלת משקל מינימלי**. אזי, גם הקבוצה $A \cup \{e\}$ **מוכלת בעפ"מ כלשהו.**



כלומר, בכל גרף לא מכוון ממושקל, לכל צלע e מתקיים:

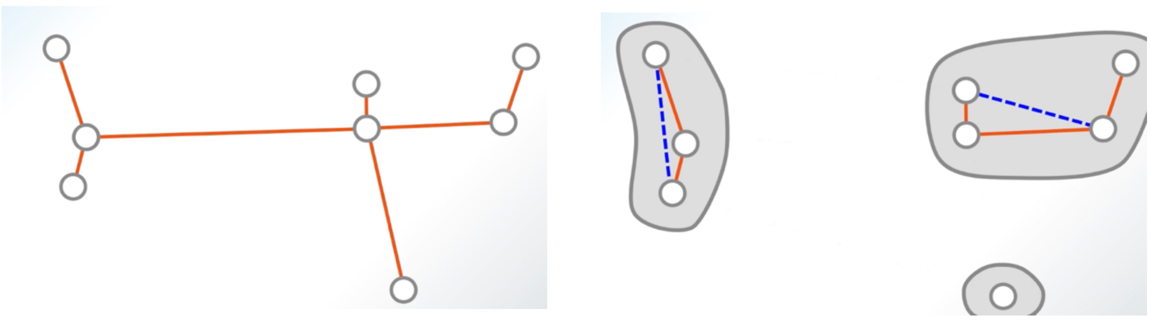
- אם e היא **צלע מזערית**, שחוצה חתך מסוים, אז יש **עפ"מ אחד לפחות** שכולל את e.
- אם e היא **צלע מזערית יחידה**, שחוצה חתך מסוים, אז **כל עפ"מ** כולל את e.

אבחנה (הוספת קשת לעפ"מ): כאשר יש לנו עפ"מ ואנחנו מוסיפים לו קשת שאינה בעץ – נסגור מעגל. נוכל לבחור קשת שרירותית כלשהי בעץ, להסיר אותה ונקבל עפ"מ חדש.

הוכחה: יהי T עפ"מ שמכיל את הקבוצה A. נתונה לנו קשת e בעלת משקל מינימלי. אם $e \in T$ סיימנו. אחרת, כאשר נוסיף אותה לעפ"מ נסגור מעגל. המעגל חייב להכיל קשת אחרת $f \in E(S, S^C) \cap T$ שחוצה את החתך ובעץ. אנו יודעים כי $w(e) \leq w(f)$ כי היא בעלת משקל מינימלי. לכן $w(T \cup \{e\} \backslash \{f\}) \leq w(T)$, כלומר אם נוסיף את e ונסיר את f אז נקבל עפ"מ שמכיל את $A \cup \{e\}$.

אלגוריתם Kruskal

אלגוריתם: נמיין את הקשתות בסדר משקלים עולה: $w(e_1) \leq \cdots \leq w(e_m)$. נאתחל את $T = \emptyset$ (עתידה להיות עפ"מ). נעבור על כל הקשתות לפי הסדר: אם ההוספה של הקשת $T \cup \{e_i\}$ לא סוגרת מעגל, אז נבצע $T = T \cup \{e_i\}$.



אינטואיציה: האלגוריתם מאתחל **יער**, שבו כל קודקוד מהווה עץ נפרד. בכל איטרציה, מוסיפים ליער צלע בעלת משקל מזערי **שמחברת בין שני עצים (אינה סוגרת מעגל)**.

*נשים לב שתמיד יש לנו **יער (חסר מעגלים), פורש**: באתחול היה לנו יער שבו כל קודקוד היה בעץ נפרד – אז ברור שאנחנו פורשים וגם אין מעגלים (אין בכלל צלעות). בכל צעד אנו מוסיפים צלע (מזערית), אז אם מראש לא היו מעגלים כאשר נחבר בין שני עצים שונים עדיין לא יוצרים מעגלים, ותמיד שומרים על הפרישה (לא זורקים קודקודים). מספר רכיבי הקשירות יורד בכל איטרציה (שני עצים מתלכדים לאחד), ובתנאי העצירה נקבל רכיב קשירות יחיד, חסר מעגלים, ופורש. קיבלנו עץ פורש.*

```
Kruskal(G = (V,E,w)):
sort(E,w)
T ← ∅
for v ∈ V:
    Makeset(v)
for i ← 1 to m:
    (u,v) ← e_i
    if Find(u) ≠ Find(v):
        T ← T ∪ {e_i}
        Union(u,v)
return T
```

מימוש: נמיין את הקשתות לפי המשקלים. ניעזר במבנה Union-Find. לכל קודקוד נבצע MakeSet. נעבור על כל הקשתות לפי הסדר: אם הקשת מחברת בין קודקודים בקבוצות שונות נבצע Union ונאחד בין הקבוצות.

סיבוכיות: עלות המיון היא $O(mlogn)$ כי אנחנו מניחים שאין קשתות מקבילות ($m \le n^2$). עלות כל הפעולות על מבנה הנתונים הן $O(m\alpha(m,n))$. סה"כ $O(mlogn)$.

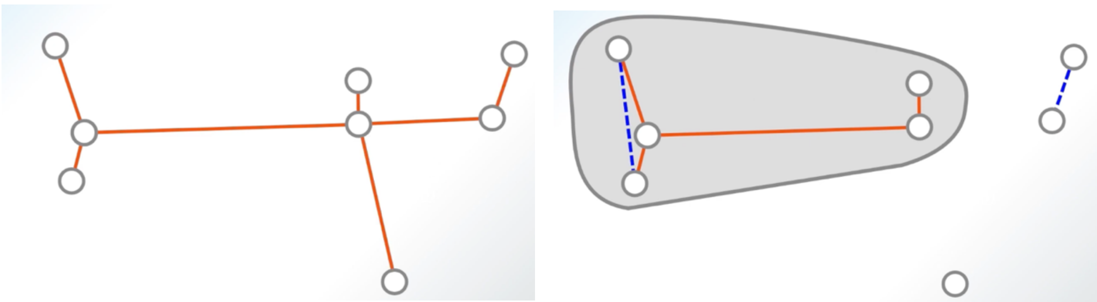
נכונות: כל קשת שאנחנו מוסיפים ל-T, היא קשת במשקל מינימלי בחתך המוגדר על ידי אחד מהעצים ביער T:

- אם e_i מחברת שני עצים שונים, היא קשת במשקל מינימלי החוצה את החתכים המוגדרים על ידי כל אחד מהעצים.
- אם e_i מחברת שני קודקודים בתוך אותו העץ, היא הקשת הכבדה ביותר על מעגל.

לפי משפט החתך: עבור משקלים ייחודיים (כל הקשתות בעלות משקל שונה) – נסמן את הפלט שלנו בכל שלב בתור T, כאשר העפ"מ הסופי הוא TM. בכל איטרציה אנחנו מוסיפים ל-T קשת השייכת לכל TM. גם T וגם TM פורשים, ולכן בכל אחד מהם יש $|V|-1$ קשתות. זה אומר שב-TM יש את כל הקשתות של T, ואסור לו לכלול צלעות נוספות, לכן הם שווים.

אלגוריתם Prim

אלגוריתם: יהי r קודקוד שרירותי, נאתחל $S = \{r\}, T = \emptyset$. כל עוד $S \neq V$ (לא כוללת את כל הצמתים), נמצא קשת $e = (u,v)$ בעלת משקל מינימלי בחתך המוגדר על ידי S, כלומר $e \in E(S, S^C)$. נבצע $S = S \cup \{v\}, T = T \cup \{e\}$.



אינטואיציה: האלגוריתם מאתחל **עץ**, עם קודקוד שרירותי. בכל איטרציה, מוסיפים לעץ צלע בעלת משקל מזערי **שיוצאת מהעץ**.

מוצאים כל פעם את הקשת המינימלית בחתך שמשרה העץ עד כה. בעצם לוקחים שורש ומגדילים אותו לעץ עד שמקבלים עץ שפורש את כל צמתי הגרף (ממשקל מינימלי).

נכונות: נובעת ישירות ממשפט החתך.

```
Prim(G = (V,E,w),s):
Init()
while S ≠ V:
  u ← ExtractMin(P)
  if u ≠ s: T ← T ∪ { (u.π,u) }
  S ← S ∪ {u}
  for (u,v) ∈ E:
    if v.d = ∞:
      v.π ← u ;  v.d ← w(u,v)
      Insert(P,v,v.d)
    else if v ∉ S and w(u,v) < v.d:
      v.π ← u ;  v.d ← w(u,v)
      DecreaseKey(P,v,v.d)
return T
```

```
Init():
P ← Heap()
for v ∈ V:
  v.π ← null
  v.d ← ∞
s.d ← 0
Insert(P,s,s.d)
S ← ∅
T ← ∅
```

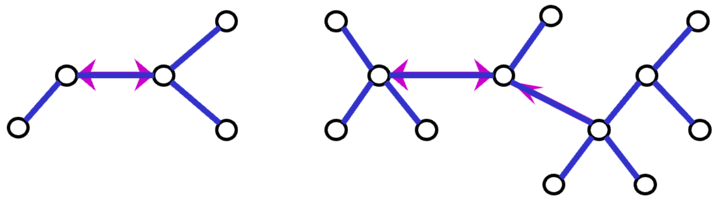
If $u \in P$, then $(u.\pi,u)$ is an edge of minimum cost connecting S and u.
Its weight is $u.d$

מימוש: ניעזר בתור עדיפויות (ערימה) P – לצמתים שאינם בעץ הנוכחי אבל מחוברים אליו באמצעות קשתות החתך. לכל צומת v נשמור שדה d – המשקל המינימלי של קשת מהעץ ל-v (זה יהיה גם המפתח ב-P). הקשת $(v.\pi, v)$ היא בעלת משקל מינימלי מהעץ לצומת v.

נמצא את הצומת u עם מפתח מינימלי בערימה P. נסיר אותו מהערימה ונוסיף אותו ל-S. נוסיף את $(u.\pi, u)$ ל-T. נבחן גם את הקשתות של u (יכולות להניב חיבורים נוספים בעלי משקל מינימלי).

סיבוכיות: זמן ריצה $O(m + nlogn)$. במימוש עם ערימת פיבונאצ'י אנחנו עושים decrease-key בזמן $O(1)$. יש לנו $O(m)$ פעולות כאלו, ועוד $O(n)$ פעולות של מציאת מינימום ב-$O(logn)$. לעומת זאת, **במימוש עם ערימה בינארית – נקבל $O(mlogn)$**.

אלגוריתם Borovska



אלגוריתם: בתחילת כל שלב יש לנו יער של עצים כחולים (כמו אצל קרוסקל), כאשר מובטח לנו שיש עפ"מ שמכיל את כל הקשתות הכחולות שבחרנו. מתחילים עם הקבוצה הריקה – היער T מכיל עצים של קודקודים בודדים. כל עוד הקבוצה T אינה עפ"מ, כל עץ ב-T בוחר קשת מינימלית שיוצאת ממנו ומוסיפים אותה ל-T. אנחנו מניחים **משקלים ייחודיים** (כולם שונים). יכול להיות ששני עצים שונים בחרו את אותה הקשת.

ניתוח: בכל איטרציה מספר העצים קטן לפחות ב-2, ולכן לכל היותר $logn$ איטרציות. כל איטרציה בעלות $O(m)$. האלגוריתם הזה פחות יעיל מ-Prim, אבל הוא לא משתמש במבני נתונים מתוחכמים. בנוסף, זה אלגוריתם מקבילי – כל העצים יכולים לבחור את הקשתות המינימליות שיוצאות מהם במקביל (בניגוד לקרוסקל שצריך למיין את המשקלים). סיבוכיות סופית היא $O(mlogn)$.

נכונות:

- הוכחנו את הנכונות של משפט החתך באופן סדרתי, אך במקרה הזה הוא מופעל באופן מקבילי. כל עץ בוחר קשת אחרת מכלל החתך, ויכולים להיות עצים שבוחרים את אותה הקשת במקביל.
- לכן נוכיח את הטענה הבאה: **אם כל המשקלים ייחודיים, אז העפ"מ הוא יחיד**. נניח שכל המשקלים שונים זה מזה, ונניח בשלילה שיש שני עצים שונים שהם עפ"מים $T_1 \neq T_2$. נסמן ב-e_1 את הקשת המינימלית ב-$T_1 \backslash T_2$, ובאופן דומה את הקשת e_2 ב-$T_2 \backslash T_1$. נניח בה"כ כי $w(e_1) < w(e_2)$. כאשר נוסיף את e_1 ל- T_2 אנחנו סוגרים מעגל. במעגל הזה חייבת להיות קשת שהיא לא ב-T_1 (אחרת היינו מקבלים מעגל ב-T_1 אבל הוא עץ). כלומר, קיימת קשת $f \in T_2 \backslash T_1$, ולכן מתקיים $w(e_1) < w(e_2) \leq w(f)$, בסתירה לכך ש-$T_2$ עפ"מ, כי נוכל לבנות את $T_2 \cup \{e_1\} \backslash \{f\}$.
- נקבל כי לפי כלל החתך, קשת מינימלית שנבחרת חייבת להיות בעפ"מ כלשהו, אבל אם **יש רק אחד כזה**, היא חייבת להיות בעפ"מ היחיד. לכן אנחנו יכולים **לבחור את כל הקשתות במקביל**.

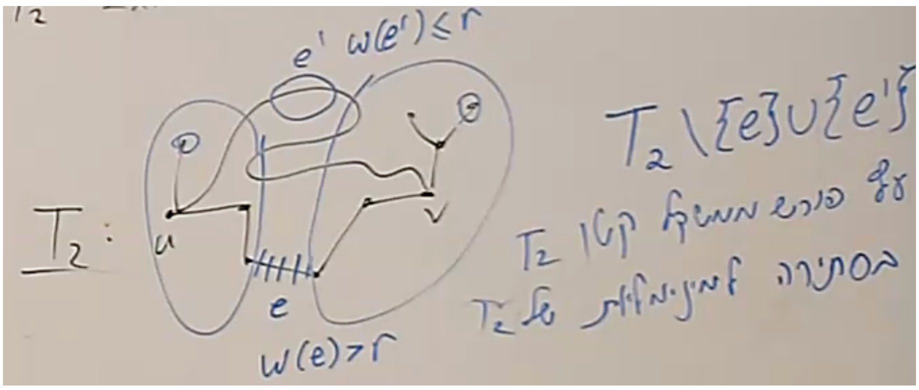
כלל המעגל: תהי $A \subseteq E$ קבוצת קשתות שמוכלת בעפ"מ כלשהו. יהי C מעגל, ו-e קשת בעלת משקל **מקסימלי** ב-C. לכן, יש עפ"מ כלשהו שמכיל את A אבל לא את הקשת e. (במילים אחרות, אפשר לזרוק את e מהגרף ועדיין מובטח לנו שיש עפ"מ שמכיל את A).

עפ"מים (תרגול 4)

1) תכונות של עפ"מים: נתונים גרף לא מכוון וקשיר $G = (V, E)$, פונקציית משקל $w: E \rightarrow \mathbb{R}$, ושני עפ"מים T_1, T_2. עבור כל מספר ממשי r, תהי $E_{1,r}$ קבוצת הקשתות של T_1 שמשקלן לכל היותר r. הקבוצה $E_{2,r}$ מוגדרת באופן דומה.
הגרפים $G_1 = \left(V, E_{1,r}\right), G_2 = (V, E_{2,r})$ מחזיקים בדיוק באותם רכיבי קשירות (לאו דווקא אותן קשתות, אותן קבוצות של קודקודים).

הוכחה: נניח בשלילה שקיימים T_1, T_2, r כך שרכיבי הקשירות של הגרפים G_1, G_2 שונים. לכן, קיימים שני קודקודים u, v שנמצאים באותו רכיב קשירות בגרף אחד וברכיבים שונים בגרף השני – נניח בה"כ שהם באותו רכיב קשירות ב-G_1 וברכיבים שונים ב-G_2.

- כל הקשתות במסלול P בין u ל-v ב-G_1 הן במשקל $w(e) \leq r$.
- לעומת זאת, ב-T_2 במסלול מ-u ל-v בהכרח יש קשת במשקל $w(f) > r$. אם **נסיר** את הקשת f נקבל **שני רכיבי קשירות**, אחד של u ואחד של v.
- אנו יודעים כי ישנו המסלול P ועליו **קשת המחברת בין רכיבי הקשירות האלו**, ממשקל $w(e) \leq r$.
- נחליף את הקשת f בקשת e ונקבל עץ פורש ממשקל קטן יותר. בסתירה לכך ש-T_2 עפ"מ.



2) רשימות משקלים: נתונים גרף לא מכוון וקשיר $G=(V,E)$, פונקציית משקל $w:E\to\mathbb{R}$, ושני עפ"מים T_1,T_2. תהי $\alpha_1\le\cdots\le\alpha_{n-1}$ רשימה ממוינת של משקלי קשתות T_1 (בהתאמה עם β עבור T_2). הוכיחו שמתקיים $\forall i:\alpha_i=\beta_i$.

כלומר, אם יש לנו שני עפ"מים לאותו גרף, תהיה להם **בדיוק אותה רשימה ממוינת של משקלים**. כל עפ"מ מורכב בדיוק מאותו צירוף של משקלים.

אבחנה: בהינתן יער F: אם יש רכיב קשירות 1 יש $n-1$ קשתות, אם יש 2 רכיבי קשירות יש $n-2$ קשתות. עבור n רכיבי קשירות יש 0 קשתות. אם יש k רכיבי קשירות אז $|F|=\sum_{i=1}^{k}|E_i|=\sum_{i=1}^{k}|V_i|-1=n-k$.

הוכחה: תהי $W=\{w_1,\dots,w_{n-1}\}$ רשימה ממוינת של המשקלים בגרף. נוכיח את הטענה $(\#_{T_1}w_i=\#_{T_2}w_i)$, כלומר **כמות הקשתות ממשקל w_i זהה בשני העפ"מים**. נוכיח עבור משקלים שהם לכל היותר w_i: ניעזר בשאלה הקודמת עם $r=w_i$, נקבל שני גרפים בעלי אותה כמות של רכיבי קשירות, ומהאבחנה יש להם אותה כמות של קשתות. אפשר להפעיל בדיוק את אותה הטענה על $r=w_{i-1}$. ואז נקבל: $\#w_i=\#w_{\le i}-\#w_{\le i-1}$.

3) קשתות צבועות: נתונים גרף לא מכוון וקשיר $G=(V,E)$, פונקציית משקל $w:E\to\mathbb{R}$. בנוסף, כל קשת צבועה בכחול או באדום. תארו אלגוריתם שמוצא עפ"מ המכיל מספר מקסימאלי של קשתות אדומות (מבין העפ"מים).

- נרצה לעדיף קשתות אדומות, לכן נגדיר קבוע קטן ε ונתייחס למשקל של קשת כחולה e כאל $w(e)+\varepsilon$. כלומר נגדיר פונקציית משקל חדשה $w':E\to\mathbb{R}$ כך שלכל קשת אדומה $w'(e)=w(e)$ ולכל קשת כחולה $w'(e)=w(e)+\varepsilon$.
- נריץ את האלגוריתם של Kruskal על הגרף עם פונקציית המשקל החדשה, ונמצא עפ"מ T.
- נשים לב כי מתקיים $w'(T)=w(T)+\varepsilon\cdot\#blue$. לפי טענה (שנוכיח בתרגיל בית) עפ"מ ב-G עם w' הוא עפ"מ ב-G גם עם w. לכן $w(T)$ הוא קבוע וגם ε. האלגוריתם מביא את $w'(T)$ למינימום **ולכן נקבל גם את המינימום של הקשתות הכחולות**.
- נגדיר $\varepsilon<\Delta_{min}$ כאשר זהו ההפרש הקטן ביותר בין משקלי 2 קשתות בגרף.
- כל עוד יחס הסדר בין המשקלים נשמר, Kruskal ו-Prim ירוצו באותה הצורה (הם רק משווים בין משקלים).

מסקנה שימושית: אם נרצה שעפ"מ מסוים יבחר, תמיד נוכל לסדר את הקשתות כך שקרוסקל יבחר אותו (נסדר אותן בסדר הנכון). למשל, **אם נרצה להוכיח שעץ הוא לא עפ"מ, אפשר להראות שהוא לא יכול להתקבל בשום הרצה של קרוסקל**.

הוכחה: יהי T עפ"מ. נסמן את הקשתות של T באדום, וכל הקשתות האחרות $E\backslash E_T$ יהיו כחולות. האלגוריתם מהתרגיל יעשה הרצה של קרוסקל שתחזיר את T (מקסימום קשתות אדומות).

4) עלים מנדטוריים: נתונים גרף לא מכוון וקשיר $G=(V,E)$, פונקציית משקל $w:E\to\mathbb{R}$ ותת קבוצה של קודקודים $L\subseteq V$. תארו אלגוריתם המוצא עץ במשקל מינימלי מבין העצים הפורשים שכל איברי L הם **עלים (דרגה 1) שלהם** (או מודיע שלא קיים כזה).

אלגוריתם: נגדיר גרף $G'=(V',E')$ כך ש-$V'=V\backslash L$ ו-E' היא קבוצת הקשתות ששני הקודקודים שלהם ב-V'. נבנה את הגרף הזה ונבדוק האם הוא קשיר. אם לא, נודיע שלא קיים עץ מתאים. אחרת, נמצא עפ"מ T' בגרף G'. עבור כל קודקוד ב-L, נמצא את הקשת הקלה ביותר שמחברת בינו לבין T' ונוסיף אותה לעץ – **כך נחבר את העפ"מ לעלים**. (אם לא קיימת קשת כזו, נודיע שלא קיים עץ מתאים).

נכונות: (במצגת תרגול 4).

סיבוכיות: בניית הגרף בעלות $O(|V|+|E|)$. נמצא עפ"מ ב-G' בסיבוכיות של Kruskal/Prim. מציאת הקשת הקלה ביותר וחיבורה לעץ בעלות $O(|E|)$.

2 – מק"בים ותכנות דינמי

SSSP (Single Source)

מסלולים קצרים ביותר

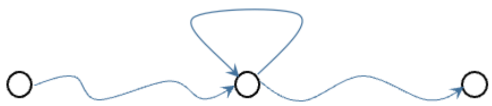
מסלולים, מעגלים ואורכים:

יהי $G = (V, E, \ell)$ גרף ממושקל, כאשר $\ell: E \to \mathbb{R}$.

- אורך: $\ell(u, v)$ הוא האורך של הקשת (u, v). האורך הוא מספר ממשי (יכול להיות גם שלילי). אם מדברים על מחירים, יש משמעות למחיר שלילי.
- path - אם $P(u_0, \dots, u_k)$ הוא מסלול, אז אורך המסלול הוא $\ell(P) = \sum_{i=0}^{k-1}(u_i, u_{i+1})$: סכום אורכי הקשתות במסלול. לא מניחים שהמסלול P הוא פשוט (הוא הילוך).
- cycle - זה בפרט כולל גם מעגל (סכום אורכי הקשתות על המעגל). מעגל הוא שלילי אם **אורכו שלילי**.
- אבחנה: אם מסלול לא פשוט הוא חייב לכלול מעגל (כלומר יש שני צמתים שהם אותו צומת, ונקבל מעגל).

מסלולים קצרים ביותר (מק"בים):

- מק"ב: מסלול P מ-s ל-t בגרף G הוא מק"ב מ-s ל-t אם $\ell(P) \le \ell(P')$ לכל מסלול אחר P'.
- למה: אם יש מסלול מ-s ל-t אבל אין אף מסלול מ-s ל-t **שעובר דרך מעגל שלילי**, אז יש **מק"ב**. בנוסף, יש **מק"ב פשוט**.



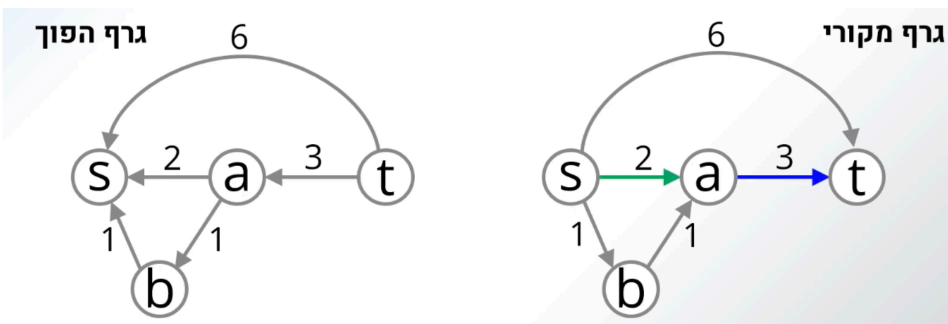
הוכחה: אם P מסלול לא פשוט מ-s ל-t אז הוא מכיל מעגל. הנחנו שאין מסלול שכולל מעגלים שליליים, לכן המעגל הזה הוא לא שלילי. אם נוריד את המעגל מהמסלול P, נקבל מסלול P' שהוא לא יקר יותר מ-P: $\ell(P') \le \ell(P)$. נחזור על התהליך ונקבל מסלול **פשוט** P' כזה.

- מרחק: המרחק בין s ל-t בגרף G: $\delta(s, t)$ הוא אורך המק"ב $\ell(P)$, ואם אין כזה נגדיר אותו להיות ∞.
- למה (אי שוויון המשולש): לכל גרף ולכל שלושה צמתים $u, v, w \in V$ מתקיים $\delta(u, v) \le \delta(u, w) + \delta(w, v)$.
- למה (תת-מסלול): אם P מק"ב, אז כל תת-מסלול של P הוא מק"ב.

מציאת מק"בים:

(1) מוצא מסוים ← יעד מסוים	
(2) מוצא מסוים ← לכל היעדים	(3) כל נק' המוצא ← ליעד מסוים
(4) מכל נק' המוצא ← לכל היעדים	

(2) ו-(3) הן בעיות דומות, בזכות השימוש בגרף הפוך:



(1) מול (2), לכאורה (1) הרבה יותר פשוטה, אבל הרבה פעמים על הדרך נמצא מסלולים ליעדים אחרים. לכן לא נפתח אלגוריתם ספיציפי לווריאציה (1), אלא ל-(2).

- וריאציות (1), (2), (3) – אלגוריתם Dijkstra, אלגוריתם Bellman-Ford
- וריאציה (4) – אלגוריתם Floyd-Warshall

בעיית SSSP: בהינתן גרף $G = (V, E, \ell)$ וצומת מקור $s \in V$, נרצה לדעת **האם יש מעגל שלילי** שאפשר להגיע אליו מ-s (אין עץ מק"בים, לחלק מהצמתים יכול להיות מרחק $-\infty$), אחרת נרצה **עץ מק"בים** מ-s לכל הצמתים הנגישים מ-s (עץ שבו לכל קודקוד v **המסלול היחיד בעץ מ-s ל-v הוא מק"ב ב-G**).

אבחנה: אם **אין מעגל שלילי** שאפשר להגיע אליו מ-s, **תמיד קיים עץ מק"בים מ-s לכל הצמתים הנגישים מ-s**.

רלקסציה (Relax):

לכל צומת $u \in V$ נשמור $\boldsymbol{u.d}$ שהוא **חסם עליון** על המרחק $\delta(s,u)$, כלומר $\delta(s,u) \le u.d$. אם $(u,v) \in E$ אז מתקיים כי $u.d + \ell(u,v)$ הוא גם חסם עליון על $\delta(s,v)$, כלומר $\delta(s,v) \le u.d + \ell(u,v)$ (נובע מאי שוויון המשולש).

Relax(u,v):
 if $v.d > u.d + \ell(u,v)$
  $v.d = u.d + \ell(u,v)$
  $v.\pi = u$
  $change = 1$

האלגוריתמים שלנו ישתמשו בפעולה Relax על קשת (u,v). האם אנחנו יכולים לשפר את $v.d$? כלומר את המסלול הקצר ביותר ל-v?

נשווה את המרחק הנוכחי של v (estimation) לאופציה החדשה (להגיע דרך u ואז עם הקשת מ-u ל-v). אם נגלה מסלול טוב יותר, נעדכן את $v.d$.
אם $v.d > u.d + \ell(u,v)$ אז נעדכן את $v.d = u.d + \ell(u,v)$ ונזכור את הקשת הקודמת $v.\pi = u$. בפועל מתקיים: $Relax(u,v): v.d \leftarrow \min\{v.d, u.d + \ell(u,v)\}$. פעולה זו מעדכנת את החסם העליון להיות ההדוק ביותר.

- למה U: אם בהתחלה $\ell(s,u) \le u.d$ לכל צומת $u \in V$ אז אחרי סדרת פעולות Relax זה עדיין יתקיים. כלומר, פעולה אחת משמרת את העובדה **ששדות ה-d הם חסמים עליונים על המרחק**.
  - מסקנה 1: אם אין מסלול מ-s ל-v ובהתחלה $v.d = \infty$ לאחר כל סדרת פעולות Relax זה ישאר כך.
  - מסקנה 2: אם בשלב כלשהו $v.d = \delta(s,v)$ אז תכונה זו לא יכולה להשתנות (אין מרחק קטן יותר).
- למה L: אם $s.d = 0$ ו-P זה מסלול כלשהו מ-s ל-t, אז אחרי כל סדרת פעולות Relax שכוללת את $Relax(u_0,u_1)$ עד הפעולה $Relax(u_{k-1},u_k)$, מתקיים $t.d \le \ell(P)$.

תזכורת (עץ פורש מכוון, BFS, DFS - קמפוס): שני אלגוריתמים מפורסמים במיוחד למציאת עץ פורש מכוון הינם BFS ו-DFS. שניהם מקבלים בקלט גרף מכוון G עם קודקוד מוצא s ושניהם מניבים כפלט איזשהו **"עץ פורש מכוון T שמושרש במוצא s"**. כשברור מההקשר מיהו קודקוד המוצא, אז עץ T כזה נקרא בדרך כלל בפשטות **עץ פורש**. פורמלית, T הינו תת-גרף של G שמקיים עבור כל קודקוד v ב-G את התכונה הבאה: אם יש ב-G מסלול (אחד או יותר) מ-s ל-v, אז יש ב-T מסלול אחד ויחיד מ-s ל-v.

אלגוריתם Bellman-Ford

אלגוריתם:

- קוראים ל-Initialize: מתאחלים את שדות ה-d בחסמים העליונים הכי טריוויאליים (∞), ואת שדות ה-π ל-NIL. מאתחלים את $s.d = 0$.
- יש לנו לולאה חיצונית של n מעברים (מספר הצמתים):
  - מאפסים את change.
  - עבור כל קשת בגרף מבצעים relax.
  - האלגוריתם נכון לא משנה באיזה סדר נעבור על הקשתות.
  - בודקים אם change עדיין אפס. אם כן, אין טעם לבצע איטרציה נוספת ונצא.
- בסופו של דבר נחזיר את change:
  - אם מוחזר 1 – נמצא מעגל שלילי מ-s.
  - אם מוחזר 0 – אין מעגל שלילי מ-s, וחושבו כל המק"בים מ-s.

BF$(G = (V,E,\ell), s)$
Initialize(G, s)
for $i = 1$ to $|V|$
 $change = 0$
 for $(u,v) \in E$
  relax(u,v)
 if $change == 0$
  break
return $change$

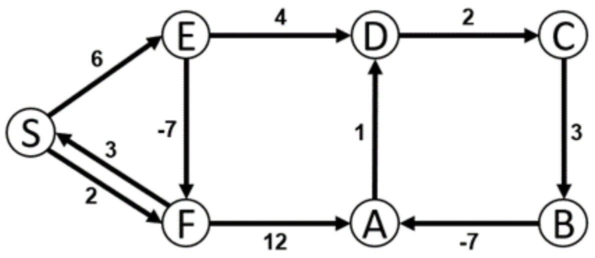
Initialize$(G = (V,E,\ell), s)$
for $u \in V$
 $u.d = \infty$
 $u.\pi$ = NIL
$s.d = 0$

relax(u,v)
if $v.d > u.d + \ell(u,v)$
 $v.d = u.d + \ell(u,v)$
 $v.\pi = u$
 $change = 1$

מה BF עושה (קמפוס): בדומה ל-Dijkstra, גם BF הוא אלגוריתם שמרן, שמבוסס על שיפור הדרגתי ע"י הוספת **צלע בודדת בקצה של מסלול** שכבר מצאנו בעבר. מכיוון שתכונת "הסדר הטוב" מופרת בנוכחותם של משקלים שליליים, אז לא ברור כבר באיזה סדר כדאי לאלגוריתם לסרוק את הצלעות. כפיצוי על כך, **BF פשוט סורק את כל הצלעות פעמים רבות** ($|V| - 1$ פעמים), מה שגורר יותר כוח חישוב.

קל לחשוב על ריצת האלגוריתם באיטרציות, ולנתח את זה בתוך טבלה. לראות איך ה-d מתעדכן עבור כל קודקוד.

איטרציה	$d(A)$	$d(B)$	$d(C)$	$d(D)$	$d(E)$	$d(F)$	$d(S)$
1	∞	∞	∞	∞	6	2	0
2	11	∞	∞	10	"	-1	"
3	11	∞	12	10	"	"	"
4	11	15	12	10	"	"	"
5	8	15	12	10	"	"	"
6	8	15	11	9	"	"	"
7	8	14	11	9	"	"	"



נכונות (מרחקים):

למה 1 (נכונות BF): אם אין מעגלים שליליים נגישים מ-s, אז האלגוריתם מסיים לאחר לכל היותר $n-1$ מעברים, מחזיר 0, ושדות ה-d מייצגים את המרחקים הקצרים ביותר $u.d=\delta(s,u)$ לכל $u\in V$.

- תחילה $s.d=0, \forall u\in V: u.d=\infty$. לפי למה U, בכל שלב יש לנו חסם עליון על המרחקים $\delta(s,u)\leq u.d$. כיוון שאין מעגלים שליליים נגישים מ-s, לכל צומת u נגיש מ-s יש מק"ב P_u שכולל לכל היותר $n-1$ קשתות.
- אחרי המעבר הראשון, עשינו relax לקשת הראשונה. אחרי המעבר השני, עשינו relax לקשת השנייה אחרי שכבר עשינו relax לקשת הראשונה, וכך הלאה. לכן, נפעיל את למה L על המסלול P_u: לכל היותר אחרי $n-1$ מעברים מתקיים $u.d\leq \ell(P_u)=\delta(s,u)$.
- אנחנו עוצרים אחרי שלא קרה שום שינוי ($change=0$) כיוון שלא יקרו שינויים נוספים בכל האיטרציות הבאות.
- שילוב שני החסמים מלמעלה ומלמטה, מניב לנו את העובדה ש-$u.d=\delta(s,u)$.

למה 2 (מעגלים שליליים): אם יש מעגל שלילי $C=(u_0,\dots,u_k=u_0)$ נגיש מ-s, אז חייב להיות לפחות שינוי אחד במעבר ה-n-י ולכן האלגוריתם יחזיר 1.

- המעגל נגיש מ-s ולכן בפרט כל קודקוד u_i נגיש מ-s. אחרי לכל היותר $n-1$ מעברים $u_i.d<\infty$.
- נניח בשלילה שלא קרה שום שינוי במעבר ה-n. כלומר, $u_i.d\leq u_{i-1}.d+\ell(u_{i-1},u_i)$.
- נסכום את אי-השוויונות ונקבל שאורך המעגל הוא חיובי, בסתירה לכך שהמעגל שלילי.

נכונות (מק"בים):

נגדיר $G_\pi=(V_\pi,E_\pi)$ כאשר $E_\pi=\{(v.\pi,v)|v.\pi\neq NIL\}$, $V_\pi=\{v\in V|v.\pi\neq NIL\}\cup\{s\}$.

למה 1: אם בזמן כלשהו, $C=(u_0,\dots,u_k=u_0)$ הוא מעגל ב-G_π (כלומר $u_i.\pi=u_{i-1}$) אז C מעגל **שלילי**.

הוכחה: נניח שיש מעגל C. נסתכל על פעולות ה-relax האחרונות (לפני רגע עצירת האלגוריתם) על הקשת (u_{i-1},u_i) שעשתה שינוי בערכים $.d,\pi$ מיד אחריה, מתקיים $u_i.d=u_{i-1}.d+\ell(u_{i-1},u_i)$. נשים לב כי $u_i.d$ לא יכול להשתנות עוד. אמנם $u_{i-1}.d$ כן יכול לקטון. לכן $u_i.d\geq u_{i-1}.d+\ell(u_{i-1},u_i)$. עכשיו נסכום על פני כל הקודקודים, ונקבל כי $\ell(C)\leq 0$.

כדי לתקן את זה סופית, מבין כל k פעולות ה-relax האלו נסתכל על האחרונה על קשת (u_{j-1},u_j). נסתכל על ערכי d רגע לפני הפעולה הזו: $u_j.d>u_{j-1}.d+\ell(u_{j-1},u_j)$. עבור כל $j\neq i$ מתקיים $u_i.d\geq u_{i-1}.d+\ell(u_{i-1},u_i)$ כי עשינו את פעולת ה-relax האחרונה. לכן נסכום שוב פעם, רק שיש לנו אי שוויון חזק, נקבל $\ell(C)<0$.

למה 2 (עץ מק"בים): אם אין בגרף מעגלים שליליים נגישים מ-s, אז בסיום האלגוריתם G_π הוא עץ מק"בים לכל הצמתים הנגישים מ-s.

הוכחה: לפי למה 1, לא יכול להיות שב-G_π יש מעגל, והוא חייב להיות עץ. לכל $t\in V$ נגיש מ-s ידוע כי $t.d=\delta(s,t)<\infty$, ויש מסלול מ-s ל-t ב-G_π. לכל קשת $(u_{i-1},u_i)\in E_\pi$ מתקיים $u_i.d\geq u_{i-1}.d+\ell(u_{i-1},u_i)$. נסכום ונקבל: $\delta(s,t)=t.d=u_k.d\geq \ell(P_t)$. לכן המסלול הוא קצר ביותר ומדובר בעץ מק"בים.

למה 3: אם יש מעגל שלילי נגיש מ-s, אז בסיום האלגוריתם G_π חייב לכלול מעגל שלילי (כל המעגלים ב-G_π הם שליליים).

הוכחה: הוכחנו שחייב להיות שינוי כלשהו במעבר ה-n-י. נסמן $u\in V$ עבור הקודקוד שה-d שלו השתנה במעבר ה-n-י. עבור כל מסלול P מ-s ל-u שמכיל לכל היותר $n-1$ קשתות מתקיים $u.d<\ell(P)$ כי אנחנו יודעים שהיה שינוי בשלב ה-n-י. נניח בשלילה כי G_π מכיל מסלול P_u מ-s ל-u אז ראינו לפי הלמה הקודמת $\ell(P_u)\leq u.d$. זו סתירה כי P_u מכיל לכל היותר $n-1$ קשתות. לכן, u חייב להיות על מעגל, ולפי למה 1 מעגל זה הוא שלילי.

נכונות של BF (קמפוס): הנכונות מבוססת על טענת האינדוקציה הבאה: בתום איטרציה l, האלגוריתם מצא עבור כל קודקוד v, איזשהו מסלול מהמוצא אל v, שמשקלו קטן או שווה לזה של המסלול הטוב ביותר, שמורכב מ-l צלעות ומטה. בשום מקום במסגרת

הוכחת טענת האינדוקציה, לא משתמשים בעובדה שמדובר במסלולים שמספר הצלעות שבהם l מקיים $l \le |V| - 1$. הטענה תקפה גם עבור $l \ge |V|$ (מסלולים שבוודאות כוללים מעגלים).

סיבוכיות: לולאה חיצונית לכל n הקודקודים, ולולאה פנימית לכל m הקשתות ולכן הסיבוכיות היא $\boldsymbol{O(mn)}$.

מעגלים שליליים (קמפוס): בגרף יש **מעגלים שליליים** (שנגישים מהמוצא) אמ"מ כשממשיכים להריץ את BF **לאיטרציה אחת נוספת**, אז עדיין **מצליחים לשפר מסלולים** עבור אחד מהקודקודים לפחות. נראה שפה אנחנו מתחזקים את change והוא מצביע לנו על הימצאות מעגל שלילי, ערך החזרה כבר מספר לנו את זה.

מק"בים ב-DAG

ב-DAG אפשר לפתור את הבעיה בזמן הרבה יותר מהיר. אנחנו יודעים שגרף הוא DAG ⟺ יש לו מיון טופולוגי. אפשר למצוא את המיון הזה בזמן לינארי בעזרת DFS. הבעיה באלגוריתם של BF הייתה שלא היה ברור באיזה סדר לעבור על הקשתות, ולכן עברנו עליהן n פעמים. ב-DAG קל לעשות את זה. **נעבור על הקשתות בסדר של המיון הטופולוגי: כל הצלעות נמתחות אך ורק משמאל לימין.**

אלגוריתם:

- נמצא את המיון הטופולוגי של הגרף $(u_1, \dots, u_n)$.
- נאתחל באופן דומה ל-BF.
- נבצע לולאה חיצונית על הצמתים:
  - עבור כל צומת u_i נבצע relax לכל הקשתות שיוצאות ממנו.

```
SSSP-DAG(G = (V,E,ℓ), s):
⟨u_1, u_2, …, u_n⟩ ← Topological-Sort(G)
Initialize(G, s) // As in Bellman-Ford
for i ← 1 to n
    for (u_i, v) ∈ u_i.Adj
        Relax(u_i, v)
```

נכונות: עבור כל מסלול מכוון בגרף $P = (s = v_0, \dots, v_k)$ אנחנו מבצעים relax על (v_i, v_{i+1}) בסדר הנכון (טופולוגית).

סיבוכיות: מציאת מיון טופולוגי, והמעבר על כל הקשתות הוא בזמן לינארי, כלומר $\boldsymbol{O(m+n)}$.

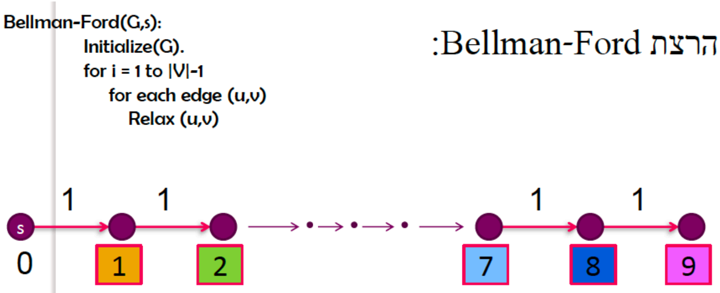
BF בגרף טופולוגי (קמפוס): ניתן לבצע לפני הרצה של BF בדיקה פשוטה, האם לגרף יש מיון טופולוגי. אם **הגרף אכן טופולוגי**, אז כל המסלולים המזעריים מתגלים כבר **באיטרציה הראשונה** (כל עוד נקפיד לסרוק את הצלעות בהתאם למיון הטופולוגי של הקודקודים). גם עבור גרפים לא טופלוגיים מובטח, שאם באיטרציה שלמה של BF **לא עודכן שום מסלול, אז אין צורך באיטרציות נוספות**, כי מובטח שמצאנו כבר את כל המסלולים המזעריים.

SSSP (תרגול 5)

1) הדיקות מספר האיטרציות: הראו שיתכן והאלגוריתם של בלמן-פורד זקוק ל-$n-1$ איטרציות בשביל לחשב מק"בים.

פתרון: אם עוברים על הקשתות מימין לשמאל עבור הגרף הבא, בכל איטרציה נעדכן בפעולת relax רק צומת אחד. רק אחרי $n-1$ איטרציות נסיים לעדכן את כל הצמתים.

הרצת Bellman-Ford:

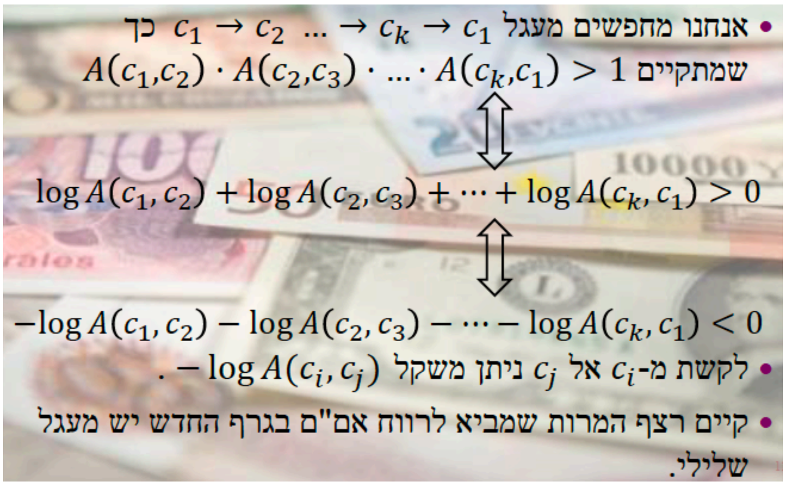


2) גרף ללא משקלים חיוביים: נתונים גרף מכוון $G = (V, E)$, קודקוד $s \in V$ ופונקציית משקל אי-חיובית $w: E \to (-\infty, 0]$. בנוסף, ידוע שאין מעגלים שליליים בגרף. תארו אלגוריתם למציאת משקלי מסלולים קלים ביותר מ-s אל שאר קודקודי הגרף.

- אם יש מעגל כלשהו בגרף – כל המשקלים על קשתות המעגל חייבים להיות 0.
- אם נמצא את גרף הרק"חים, מספיק לנו למצוא מק"ב מ-s לצומת כלשהו ברק"ח, כי אפשר לעבור בתוך הרק"ח בין כל הקודקודים במחיר 0.
- קשתות בין רק"חים הן בעלות משקל שלילי או 0, אבל בתוך רק"ח רק 0.

אלגוריתם: נבנה את גרף העל. נמצא מק"בים בגרף העל (חסר מעגלים).

סיבוכיות: מציאת **מק"בים ב-DAG** בזמן $\boldsymbol{O(|E|+|V|)}$.



3) ארביטראג': נתונים n סוגי מטבעות וטבלת מחירי המרה ביניהם. תארו אלגוריתם הבודק האם ניתן להתחיל עם מטבע אחד מסוג i ולבצע סדרת המרות שבסופה נקבל יותר ממטבע אחד מסוג i.

- רעיון – נבנה גרף מכוון, כאשר קשת שיוצאת מקודקוד שמתאים למטבע a לעבר קודקוד שמתאים למטבע b היא במשקל ההמרה $A(a,b)$. אנחנו רוצים מעגל שבו מכפלת המשקלים גדולה מ-1.
- כדי לעבור ממכפלה לסכום, נוציא log. כיוון שאלגוריתם בלמן-פורד מוצא מעגל שלילי.
- הסיבוכיות היא $O(|V| \cdot |E|) = O(n^3)$ כיוון שמתקיים $m = n^2$.
- הגרף קשיר חזק ולכן נוכל להריץ בלמן-פורד מכל קודקוד מקור שהוא.

4) המסלול הקל לקודקוד: נתונים גרף מכוון $G = (V,E)$ ופונקציית משקל $w: E \to \mathbb{R}$. בנוסף, ידוע **שאין מעגלים שליליים בגרף**. עבור כל קודקוד $v \in V$ נגדיר $\delta^*(v) = \min_{u \in V} \delta(u,v)$: משקל המסלול הקל ביותר מכל המסלולים שמסתיימים ב-v. תארו אלגוריתם אשר מחשב את כל ערכי $\delta^*(v)$.

אלגוריתם:

טריק: נשים לב כי לכל צומת מתקיים $\delta^*(v) \le 0$. בהינתן שני מסלולים לצומת t, ניתן להוסיף קודקוד חדש s ולמתוח ממנו קשתות לשני הצמתים המובילים ל-t עם משקל 0. אחרי הרצת בלמן-פורד מהקודקוד s נקבל את המסלול הקל ביותר בגרף המקורי (לאחר הורדת הקשת החדשה).

נוסיף קודקוד חדש s ונוציא ממנו קשתות במשקל 0 לכל שאר קודקודי הגרף. נריץ Bellman-Ford ועבור כל קודקוד $v \in V$ נציב $\delta^*(v) = \delta(s,v)$.

נכונות: ניזכר כי המסלול הקצר ביותר $\delta(s,v)$ הוא רקורסיבית לבחור את הקשת הראשונה ואת שאר המסלול. כיוון שהגדרנו את הקשת הראשונה בתור קשת במשקל 0, נקבל:

$$\boldsymbol{\delta(s,v) = \min_{u}\big(w(s,u) + \delta(u,v)\big) = \min_{u}\big(\delta(u,v)\big) = \delta^*(v)}$$

סיבוכיות: הוספת הקודקוד לגרף בעלות $O(|V|)$, הרצת בלמן-פורד $O(|V| \cdot |E|)$ וסה"כ $O(|V| \cdot |E|)$.

5) מעגלים במשקל 0: נתונים גרף מכוון $G = (V,E)$ ופונקציית משקל $w: E \to \mathbb{R}$. בנוסף, ידוע **שאין מעגלים שליליים בגרף**. תארו אלגוריתם שבודק האם קיים מעגל במשקל 0 בגרף.

אלגוריתם:

- נבדוק אם יש מעגל ב-G? לא נוכל לעבוד עם רק"חים כמו בתרגיל 2 כיוון שיכול להיות מעגל ממשקל חיובי.
- נבדוק אם יש מעגל ב-G_s^*! נסתכל על גרף מק"בים מ-s (בנינו אותו בתרגול 2, נוסיף s כזה שמחובר עם קשתות 0 לכל שאר הקודקודים).
  - מעגל **במשקל 0 תמיד חלק ממק"ב** מ-s.
  - מעגל **במשקל חיובי לא חלק מאף מק"ב** מ-s (נוכל להוריד אותו ולקבל מסלול קצר יותר).
- איך נבנה את גרף המק"בים? בתרגול 2 הרצנו BFS והוספנו קשתות בין רמות עוקבות: $\delta(s,v) = \delta(s,u) + \mathbf{1}$. כעת אנחנו עובדים עם משקלים ולכן נשמור רק קשתות המקיימות: $\delta(s,v) = \delta(s,u) + \boldsymbol{w(u,v)}$. אנחנו יכולים להשתמש בגרף מק"בים כדי להפריד בין המקרים האלה: זה גרף ששומר רק על מעגלי 0 ומעיף את כל המעגלים החיוביים.
- נבדוק האם קיים מעגל בתת-הגרף שקיבלנו (למשל על ידי DFS).

נכונות: נראה ישירות שכל מעגל בגרף החדש הוא במשקל 0, ולהפך. המעבר משמר רק מעגלי 0 (**הוכחה במצגת תרגול 5**).

- אם קיים בגרף החדש מעגל, אז בהכרח משקלו 0 – הדלתות מתבטלות ונקבל 0.
- מעגל שמשקלו 0 ב-G יהיה קיים גם בגרף החדש.

סיבוכיות: הוספת הקודקוד לגרף בעלות $O(|V|)$, הרצת בלמן-פורד $O(|V| \cdot |E|)$, וזריקת הקשתות ובדיקת מעגל בזמן ליניארי. סה"כ נישאר עם הסיבוכיות של בלמן-פורד $O(|V| \cdot |E|)$.

אלגוריתם Dijkstra

כעת נפתור את בעיית SSSP כאשר המשקלים הם **אי-שליליים**. בפתרון הבעיה הזאת, אנחנו יכולים לבצע פעולת relax על כל קשת רק פעם אחת: יש סדר נכון לביצוע פעולות relax, ונמצא אותו תוך כדי ריצת האלגוריתם. הטריק: הצומת הבא שכדאי לעשות relax לקשתות שיוצאות ממנו, הוא יהיה הצומת **שעדיין לא הוספנו והמרחק אליו הוא הכי קטן**.

אלגוריתם:

- נאתחל ערימת מינימום ריקה, נגדיר $s.d=0$ ונכנס את s לערימה (נשמור את כל הצמתים שעדיין לא עשינו להם relax בערימת מינימום).
- נגדיר את הקבוצה S – הצמתים שהוצאנו כבר מהערימה, אנחנו כבר יודעים את המרחקים אליהם.
- כל עוד הערימה לא ריקה:
  - נוציא מהערימה איבר u עם מפתח מינימלי ומוסיפים ל-S.
  - עוברים על כל הקשתות שיוצאות מ-u ומבצעים פעולות relax. אם $v.d=\infty$ פעולת ה-relax תצליח בוודאות ונכניס לערימה את v. אחרת, נבצע decrease-key על v (המרחק אליו קטן).
- האלגוריתם זהה מאוד ל-Prim: ההבדל בקוד הוא בבדיקה של $u.d+len$ כאשר ב-Prim לא התחשבנו ב-$u.d$.

$Init()$:
$Q \leftarrow Heap()$
for $v \in V$:
  $v.\pi \leftarrow null$
  $v.d \leftarrow \infty$
$s.d \leftarrow 0$
$Insert(Q,s,s.d)$
$S \leftarrow \emptyset$

Almost identical to the Jarnik-Prim-Dijkstra algorithm for MST

The $u.d$'s are only changed using relax operations.

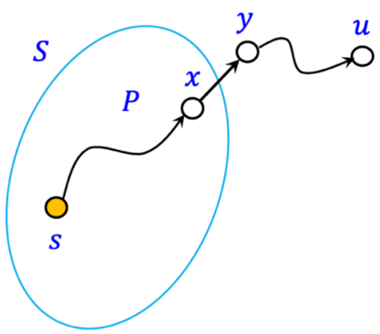
Dijkstra$(G=(V,E,\ell),s)$:
$Init()$
while $Q \neq \emptyset$:
  $u \leftarrow ExtractMin(Q)$
  $S \leftarrow S \cup \{u\}$
  for $(u,v) \in u.Adj$:
    if $v.d=\infty$:
      $v.\pi \leftarrow u$; $v.d \leftarrow u.d+\ell(u,v)$
      $Insert(Q,v,v.d)$
    else if $u.d+\ell(u,v)<v.d$:
      $v.\pi \leftarrow u$; $v.d \leftarrow u.d+\ell(u,v)$
      $DecreaseKey(Q,v,v.d)$

ביצד Dijkstra פועל (קמפוס): האלגוריתם פועל באיטרציות (שיפור-הדרגתי). באתחול, העץ T כולל רק את s. בכל איטרציה נבצע:
(א) **נוסיף ל-T צלע** $v \to u$ שיוצאת מ-T, הצלע שמובילה **לקודקוד v מחוץ ל-T הקרוב ביותר** להערכתנו ל-s. כלומר v שממזער את $v.d$. ליתר דיוק, נוסיף את הצלע $v \to u$ שיוצאת מ-T וממזערת את $u.d+w(u\to v)$.
(ב) **ננסה לשפר מסלולים** לכל הקודקודים z שמחוץ ל-T על ידי הוספת צלע ישירה $z \to v$ בקצה של המק"ב, שמצאנו הרגע ל-v.
האלגוריתם הוא "נגעת-נסעת" – ברגע ש-v מתווסף לעץ T, בחרנו סופית את המסלול מ-s ל-v, לא נעדכן יותר את $v.d$.

נכונות:

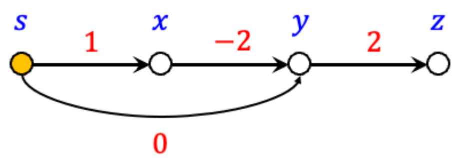
למה: אם צומת $u \in S$ (הוצא מהערימה) אז $u.d=\delta(s,u)$.

הוכחה: נוכיח באינדוקציה על הסדר שבו יוצאים הצמתים מהערימה Q. בסיס האינדוקציה הוא s, מתקיים $s.d=0$ וזה בדיוק המרחק. נניח שמוציאים צומת u והטענה נכונה לכל הצמתים הקודמים שהוצאנו מהערימה Q. נסתכל על מק"ב P מ-s ל-u. $s \in S$, וכרגע $u \notin S$, ולכן חייב להיות **צומת אחרון על המסלול** שעבורו $x \in S$.



- מתקיים $y.d \le x.d+\ell(x,y)=\delta(s,x)+\ell(x,y)=\delta(s,y)$. זאת כיוון שהוצאנו את x מהערימה לכן עשינו relax לקשת (x,y). לפי הנחת האינדוקציה $x.d$ לא יכול לקטון. השוויון השני נובע מכך שתת-מסלול של מק"ב (P) הוא גם מק"ב (מ-s ל-y). לכן בסופו של דבר נקבל $y.d=\delta(s,y)$.
- מתקיים $u.d \le y.d=\delta(s,y) \le \delta(s,u)$. זאת כיוון שאנחנו הולכים להוציא את u מהערימה והוא עם המפתח הכי קטן בערימה (אולי הם שווים). השוויון השני נובע מכך שתת-מסלול של מק"ב הוא גם מק"ב וגם אורך המסלול מ-y ל-u הוא אי-שלילי כי כל המשקלים אי-שליליים. לכן $u.d=\delta(s,u)$.

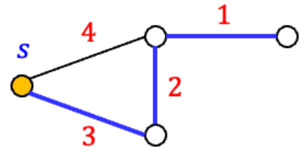
לאחר שהאלגוריתם מסיים, מתקיים $u.d=\delta(s,u)$ לכל $u \in V$ ו-G_π **הוא עץ מק"בים מ-s** לכל הצמתים הנגישים מ-s.



הרצת Dijkstra עם משקלים שליליים: כאשר חלק מהקשתות בעלות משקל שלילי, האלגוריתם **יכול להניב תוצאות לא נכונות**! האלגוריתם יעשה relax לקשתות $(s,x),(s,y)$. עכשיו הוא יקפוץ ל-y (יותר קרוב) ויעשה relax לקשת (y,z). כך נקבל מרחק 2 ל-z. עכשיו הוא יעבור ל-x (מה שנשאר) ויבין שהמרחק ל-y הוא -1. עכשיו לא יסתכל על y שוב **והמרחק ל-z לא יהיה נכון**.

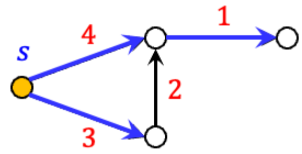
סיבוכיות: בדיוק כמו Prim, הסיבוכיות בעזרת ערימת פיבונאצ'י היא $\boldsymbol{O(m+n\log n)}$. לכל היותר n פעולות insert, extract-min, ולכל היותר m פעולות decrease-key.

Prim: $v.d \leftarrow \min\{\, v.d\,,\ w(u,v)\,\}$



Keys extracted:
3, 2, 1

Dijkstra: $v.d \leftarrow \min\{\, v.d\,,\ u.d + \ell(u,v)\,\}$



Keys extracted:
3, 4, 5

<u>השוואה בין Dijkstra ל-Prim:</u>

- באלגוריתם של Dijkstra, אם x יוצא מהערימה לפני y אז מתקיים $x.d \le y.d$. סדר המפתחות הוא **לא יורד**.
- זה לא בהכרח קורה ב-Prim. זה עלול לרדת, מה שלא יכול לקרות ב-Dijkstra.

<u>הערות אחרונות:</u>

- <u>ערימה מונוטונית</u> – ערימה שאמורה לתמוך בצורה נכונה בפעולות, רק אם היא מקיימת את התכונה שראינו קודם: סדרת המפתחות של האיברים שאנחנו מקבלים ב-extract-min היא לא יורדת. כדי לממש את האלגוריתם של Dijkstra לא צריכים לממש ערימה כללית, מספיק לממש ערימה מונוטונית.
- <u>אלגוריתם Dial</u> – D הוא חסם עליון על כל המרחקים מ-s. נוכל לקבל אלגוריתם יותר יעיל מ-Dijkstra: $O(m+D)$. נשתמש בערימה פשוטה שמבוססת על buckets.

המשך SSSP (תרגול 6)

0) עמק"ב = עפ"מ?: הוכיחו או הפריכו: העמק"ב המתקבל מהרצת Dijkstra על גרף קשיר ולא מכוון G מצומת s, הוא עפ"מ של G.

<u>השוואה Dijkstra מול Prim:</u> האלגוריתמים מאוד דומים. ההבדל היחידי זה האופן שבו אנחנו עושים את פעולת ה-relax. ב-Dijkstra אנחנו בודקים האם אפשר למצוא מסלול קצר יותר לצומת באמצעות הקשת שכרגע מסתכלים עליה, וב-Prim הבדיקה השקולה בודקת האם הקשת היא מינימלית באיזשהו חתך. פרט לאופן שבו בוחרים את הערכים ששמים בתוך ערימת המינימום, המבנה הכללי מאוד דומה. לכן זמן הריצה הוא אותו דבר.

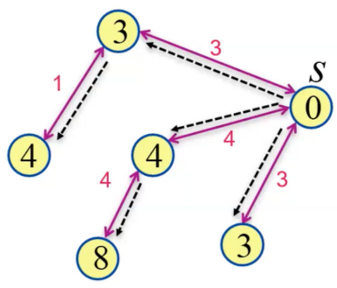
Dijkstra בקצרה:

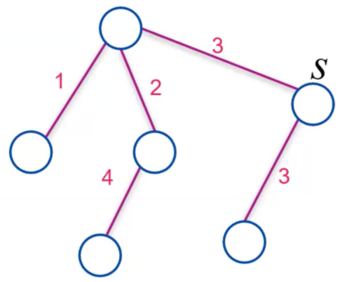
- נבחר קודקוד הכי קרוב ל- s שעוד לא בחרנו.
- לכל קשת לשכן שלו, נעשה relax.

<u>הערה:</u> Dijkstra רץ על גרף מכוון, בשביל להריץ על גרף לא מכוון, נהפוך כל קשת לא מכוונת (u,v) לשתי קשתות מכוונות $(u,v),(v,u)$. כיוון שהגרף הוא בעל משקלים אי-שליליים אנחנו לא ניצור מעגלים שליליים בטרנספורמציה הזו.

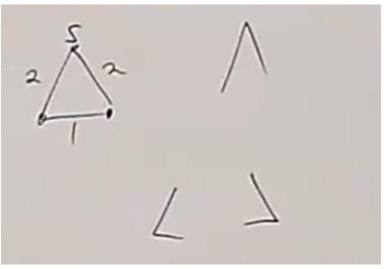
<u>פתרון:</u> כאשר הוספנו צומת, **מובטח לנו שהמרחק של נקבע ולא ישתנה עוד** (אין משקלים שליליים בגרף ולכן אין אופציה לקבל מסלול קצר יותר אליו). העמק"ב והעפ"מ לא אותו דבר:

- נשווה עם עפ"מ:



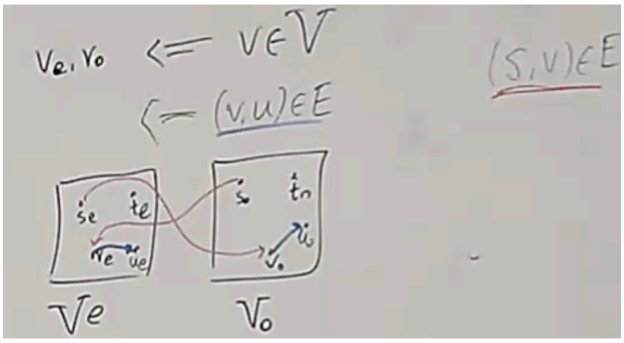


דוגמה יותר פשוטה (עמק"ב למעלה, עפ"מ למטה).



1) מספר זוגי של קשתות אדומות: נתונים גרף מכוון $G=(V,E)$, זוג צמתים $s,t\in V$ ופונקציית משקל אי-שלילית $w: E \to [0,\infty)$. בנוסף, כל קשת בגרף צבועה באדום או בכחול. תארו אלגוריתם למציאת המק"ב מ-s ל-t מבין המסלולים המכילים מספר זוגי של קשתות אדומות (המסלול ראשי לעבור דרך אותה קשת יותר מפעם אחת).

אלגוריתם: נשכפל את הגרף:



- לכל צומת $v\in V$ ניצור שני צמתים v_e, v_o.
- אם יש לנו קשת $(v,u)\in E$ בגרף המקורי, נמיר אותה לשתי קשתות. אם הקשת כחולה נמתח קשתות בין קודקודים באותו גרף: (v_e,u_e), (v_o,u_o). אם הקשת $(s,v)\in E$ אדומה, נמתח קשתות שמעבירות בין הגרפים: $(s_e,v_o),(s_o,v_e)$.
- אם נחפש מסלול בגרף המקורי שעובר במספר זוגי של קשתות אדומות, אפשר להתאים אותו למסלול בגרף החדש **שמתחיל ונגמר באותו הצד** – סימן שעברנו **במספר זוגי של קשתות אדומות, כי הן אלה שמעבירות צד**. הדרך היחידה לעבור צד היא דרך קשת אדומה.

נכונות: יש התאמה (בשני הכיוונים) בין מסלולים $p: s \to \cdots \to t$ ב-G עם מספר זוגי של קשתות אדומות ובין מסלולים דומים p' בגרף G'. התאמה זו משמרת את המשקל, כלומר $w(p) = w(p')$. בהינתן מסלול p' ניתן להחזיר את המתאים לו p בזמן לינארי.

הנכונות נובעת: מספיק למצוא מק"ב מ-s_e ל-t_e ב-G' כדי למצוא מק"ב עם מספר זוגי של קשתות אדומות מ-s ל-t ב-G.

כיוון 1: יהי מסלול $s_e \to v_?^1 \to \cdots \to v_?^k \to t_e$ בגרף G'. כל מסלול כזה שמתחיל ונגמר באותו צד מכיל מספר זוגי של קשתות שמעבירות צד. מהבנייה, קשת ב-G' $u_? \to v_?$ מעבירה צד אם $u \to v$ קשת אדומה ב-G.
לכן המסלול המתאים $s \to v^1 \to \cdots \to v^k \to t$ הוא מסלול ב-G שמכיל מספר זוגי של קשתות אדומות, ובאותו משקל.

סיבוכיות: בגרף החדש שכפלנו את מספר הקודקודים ואת הקשתות. סיבוכיות Dijkstra נשארת $O(|V|\log|V| + |E|)$.

2) וידוא פונקציית מרחק: נתון גרף מכוון וממושקל $G = (V, E, w)$ **ללא מעגלים שליליים**. נתונים בנוסף $s \in V$ ופונקציה $f: V \to \mathbb{R}$. הציעו אלגוריתם יעיל שבודק האם $\forall v \in V. f(v) = \delta(s, v)$.

פונקציית פוטנציאל: פונקציה שנותנת לכל קודקוד ערך ממשי. בעזרתו נגדיר משקל משוחלף/מתוקן, משקל חדש שמוגדר באמצעות פונקציית הפוטנציאל. $w_f(u, v) = w(u, v) + f(u) - f(v)$.
למה זה שימושי? לכל מסלול $p: s \to \cdots \to t$ יתקיים כי $w_f(p) = w(p) + f(s) - f(t)$. מסלול מינימלי בגרף המקורי יביא גם את המסלול המינימלי עם המשקל המשוחלף. כלומר **w_f משמרת מק"בים!**

אבחנה: נבחר $f(u) = \delta(s, u)$. מתקיימים כמה דברים:

1. $f(s) = 0$.
2. $w_f(u, v) = w(u, v) + f(u) - f(v) = w(u, v) + \delta(s, u) - \delta(s, v) \geq 0$. זה נובע מאי שוויון המשולש כיוון שמתקיים $w(u, v) + \delta(s, u) \geq \delta(s, v)$.
3. אפשר להגיע מ-s לכל צומת $u \in V$ במסלול ממשקל משוחלף 0. בפרט אם נסתכל על מק"ב $p: s \rightsquigarrow t$ מתקיים $w_f(p) = w(p) + f(s) - f(t) = \delta(s, t) + 0 - \delta(s, v) = 0$.

פתרון: נבדוק את כל הקריטריונים הללו, אלו תנאים מספיקים והכרחיים ולכן $f(v) = \delta(s, v)$.

נכונות: נוכיח את הכיוון שבו נניח את קיום הקריטריונים.

- מקריטריון 2 נקבל שכל המרחקים המשוחלפים הם אי שליליים, לכל $v \in V$ מתקיים $\delta_f(s, v) \geq 0$.
- לכן, מקריטריון 3 לכל $v \in V$ מתקיים $\delta_f(s, v) = 0$.
- לכן $0 = \delta_f(s, v) = \delta(s, v) + f(s) - f(v)$ ומקריטריון 1 נקבל $f(v) = \delta(s, v)$.

סיבוכיות: כל קריטריון ניתן לבדיקה בזמן לינארי. 1 ו-2, זה פשוט חישוב הפונקציה f. לגבי תנאי 3 – אם המסלול הוא ממשקל 0 בהכרח כל הקשתות הן ממשקל 0. נמחק את כל הקשתות שהן ממשקל שונה מ-0, ולהריץ BFS מ-s ולראות שאפשר להגיע לכל הצמתים.

APSP (All Pairs)

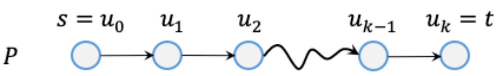
פונקציית פוטנציאל

בעיית APSP: נרצה למצוא מק"בים עבור **כל זוג צמתים בגרף**, לא נתחיל ממוצא s ספציפי. הקלט הוא רק הגרף $G = (V, E, \ell)$.

- הפלט של המרחקים יהיה בצורת מטריצה $n \times n$ בשם D, כאשר $D_{uv} = \delta(u, v)$. נניח $V = \{1,2, \ldots, n\}$.
- החלק השני של הפלט יהיה מק"בים, נייצג אותם באמצעות מטריצה $n \times n$ בשם Π, כאשר אם $\Pi_{uv} = w$ אז (w, v) נמצאת במק"ב שמקורו ב-u, כלומר (w, v) היא הקשת האחרונה על המק"ב מ-u ל-v.

פונקציית פוטנציאל: פונקציית פוטנציאל $p: V \to \mathbb{R}$ היא פונקציה שנותנת לכל מספר את הפוטנציאל שלו. אם ℓ זו פונקציית אורך, נוכל לכתוב פונקציה ℓ_p שזה האורך מתוקן לפי p: $\boldsymbol{\ell_p(u, v) = \ell(u, v) + p(u) - p(v)}$.

- לא משנה איזה פונקציה p כזו נבחר, אם נסתכל על מחירים מתוקנים לפי הפונקציה p, **המק"בים לא ישתנו**. זאת כיוון שלכל מסלול אם מסתכלים על האורך המקורי לעומת האורך המתוקן, קל לראות שהאורך המתוקן הוא האורך המקורי ועוד פוטנציאל צומת ההתחלה פחות פוטנציאל צומת הסיום (טור טלסקופי): $\ell_p(P) = \ell(P) + p(s) - p(t)$.



$$\ell_p(P) = \sum_{i=0}^{k-1} \ell_p(u_i, u_{i+1}) =$$

$$\boxed{\begin{array}{c} \ell(u_0,u_1) + p(u_0) - p(u_1) + \\ \ell(u_1,u_2) + p(u_1) - p(u_2) + \\ \vdots \\ \ell(u_{k-1},u_k) + p(u_{k-1}) - p(u_k) \end{array}}$$

$$= \sum_{i=0}^{k-1} \ell(u_i, u_{i+1}) + p(u_0) - p(u_k)$$
$$= \ell(P) + p(s) - p(t)$$

- אנחנו רוצים להסתכל על כל המסלולים בין s ל-t ולמצוא את המסלול עם האורך הכי קצר. האורך משתנה, אבל היחס בין האורכים של שני מסלולים שונים לא משתנה. אם לכל שני מסלולים הוספנו בדיוק את אותו דבר $p(s) - p(t)$.
- המשימה שלנו היא לבחור פונקציית פוטנציאל feasible (שהרי המק"בים לא ישתנו) כך שהמחירים המתוקנים יהיו אי-שליליים: כלומר לכל $(u,v) \in E$ יתקיים $\boldsymbol{\ell_p(u,v) \geq 0}$. אם זה יתקיים נוכל להריץ את Dijkstra.

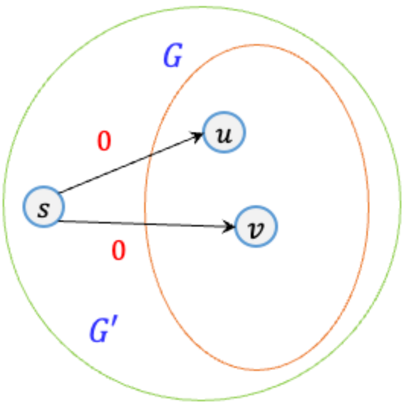
טענה: היא **שאם P מסלול מ-s ל-t מתקיים** $\boldsymbol{\ell_p(P) = \ell(P) + p(s) - p(t)}$.

- מסקנה 1: P הוא מק"ב מ-s ל-t ביחס ל-ℓ ⇔ P הוא מק"ב מ-s ל-t ביחס ל-ℓ_p.
  - לכל שני מסלולים, הם עברו בדיוק את אותו שינוי, הוספנו להם את אותו קבוע. אם אחד היה יותר קצר מהשני ביחס לפונקציית האורך המקורית, הוא יהיה קצר יותר מהשני ביחס לפונקציית האורך המתוקנת.
- מסקנה 2: אם P הוא מעגל, אז $\ell_p(P) = \ell(P)$ כי $s = t$.
- מסקנה 3: נוכל לחשב $\delta_\ell(s,t) = \delta_{\ell_p}(s,t) - p(s) + p(t)$.

אלגוריתם Johnson

טענה: יש פונקציית פוטנציאל פיזיבילית (עבור כל קשת $\ell_p(e) \geq 0$) ⇔ אין בגרף מעגל שלילי. ניתן למצוא פונקציה כזו על ידי הרצת אלגוריתם SSSP כמו BF.

שימוש ב-Bellman-Ford למציאת פונקציית פוטנציאל פיזיבילית:



- נבחר צומת שרירותי $s \in V$. נריץ BF מהצומת s, וכך נמצא את המרחקים ממנו לכל הצמתים בגרף. נגדיר את הפוטנציאל של צומת v להיות המרחק בין s ל-v: $p(v) = \delta(s,v)$. נשים לב כי לפי אי שוויון המשולש: $\delta(s,v) \leq \delta(s,u) + \ell(u,v)$ ⇔ $\ell(u,v) + \delta(s,u) - \delta(s,v) \geq 0$ לכן קיבלנו כי: $\ell(u,v) + p(u) - p(v) \geq 0$. כלומר $\ell_p(u,v) \geq 0$ קיבלנו את מה שרצינו. האורך המתוקן של הקשת הוא חיובי.
- בעיה: אם אפשר מ-s להגיע לכל הצמתים הכול בסדר. **אולי יש צמתים שאינם נגישים מ-s** ונקבל $\delta(s,u) = \infty$, פוטנציאל צריך להיות סופי.
- אפשר לדאוג שבגרף יהיה צומת כזה, **נוסיף צומת חדש**, ונוסיף קשתות ממנו לכל הצמתים בגרף. המשקלים של הקשתות האלה נבחרים על ידינו, אפשר לשים בכולם אורך 0.
- אם G זה הגרף המקורי ונקרא לחדש G': ב-G' יש מעגל שלילי ⇔ ב-G יש מעגל שלילי. ההוספה של הצומת לא הוסיפה מעגלים לגרף. לכן אם נריץ את BF על הגרף החדש, הפוטנציאלים יהיו המרחקים מ-s.

אלגוריתם:

1. מריצים את BF על גרף עזר שמוסיפים לו את המקור s. מוצאים את פונקציית הפוטנציאל p או מעגל שלילי.
2. נגדיר את האורכים המתוקנים ביחס ל-p על ידי $\ell_p(u,v) = \ell(u,v) + p(u) - p(v)$.
3. מריצים Dijkstra מכל צומת עם פונקציית האורך המתוקנת.
4. מתקנים את המרחקים: $\delta_\ell(u,v) = \delta_{\ell_p}(u,v) - p(u) + p(v)$.

נכונות: המק"בים שמצאנו הם גם המק"בים ביחס לפונקציית האורך המקורית.

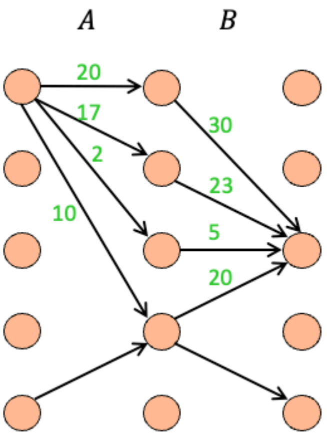
סיבוכיות: BF פעם אחת: $O(mn)$, ובנוסף הרצת n פעמים Dijkstra כלומר $O(n(m + nlogn))$. העלות של BF נבלעת. נקבל סה"כ סיבוכיות של $\boldsymbol{O(mn + n^2 logn)}$.

ייצוג על ידי מטריצות

נגדיר את הבעיה:

- קלט: נניח כי $V = \{1, \dots, n\}$, והקלט הוא מטריצה בגודל בגודל $n \times n$ שבה $L_{uv} = \ell(u,v)$ לכל קשת ב-E, כאשר לכל קשת שלא קיימת נשים ∞ ומצומת לעצמו נרשום 0.
- פלט: מטריצה בגודל $n \times n$ שבה $D_{uv} = \delta(u,v)$.

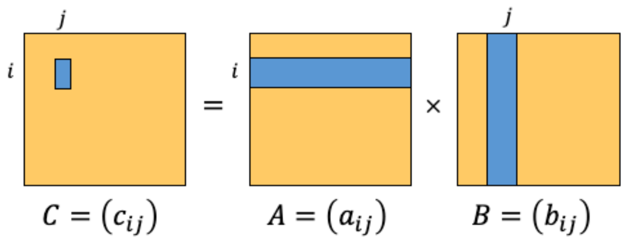
Min-Plus Product: נוסחה דומה לכפל מטריצות, כאשר את הסכום נחליף במינימום, ואת הכפל נחליף ב+. זהו מקרה פרטי של בעיית APSP. **בהינתן שתי מטריצות A ו-B, נוכל לקודד את הבעיה בגרף שמורכב מ-3 שכבות.** יש לנו קשת בין הצומת ה-i בשכבה הראשונה לבין הצומת ה-j בשכבה השנייה, אמ"מ האיבר $a_{ij} < \infty$ ונשים על הקשת את המשקל. אותו דבר נעשה למטריצה B. התוצאה של חישוב ה-Min-plus product: זה יהיה המק"ב מהצומת ה-i בשכבה הראשונה לצומת ה-j בשכבה השלישית.



$$C = A * B$$
$$c_{ij} = \min_k \ a_{ik} + b_{kj}$$

Min-Plus product

Algebraic Product	Min-Plus Product
$C = A \cdot B$	$C = A * B$
$c_{ij} = \sum_k a_{ik} b_{kj}$	$c_{ij} = \min_k \ a_{ik} + b_{kj}$



על פי ההגדרה חישוב יעלה $O(n^3)$. נשים לב כי Min-plus הוא **אסוציאטיבי**: $(A \times B) \times C = A \times (B \times C)$.

וריאנט של BF: נניח שבגרף אין מעגלים שליליים. נסמן את המק"ב מ-s ל-v **שמשתמש בלכל היותר k קשתות**: $d_v^k = \delta_k(s,v)$. עבור $k = 1$ נקבל $d_v^1 = \ell(s,v)$.

- מק"ב כזה יכול להיות אחד מן השניים: לכל היותר $k-1$ קשתות, או k קשתות: $d_v^k = \min_{u \in V}[d_v^{k-1}, d_u^{k-1} + \ell(u,v)]$.
  נניח כי $\ell(v,v) = 0$ כלומר תמיד מתקיים לכל $v \in V$: $d_v^k \le d_v^{k-1}$. **זה שאנחנו מרשים יותר קשתות, לא נותן למרחקים לגדול.** המרחקים לא עולים. לכן נוכל לרשום פשוט $d_v^k = \min_{u \in V}[d_u^{k-1} + \ell(u,v)]$.
- נשים לב כי המק"ב בין s ל-v מכיל לכל היותר $n-1$ קשתות, ולכן $d_v^{n-1} = \delta(s,v)$.
- בכתיב מטריציוני אפשר לכתוב $d^k = d^{k-1} \times L$ כאשר הכפל הוא Min-plus.

<u>איך נפתור את APSP על ידי כפל מטריצות:</u>

- נגדיר מטריצה $D_{uv}^{(k)} = \delta_k(u,v)$: אורך המק"ב מ-u ל-v בעל לכל היותר k קשתות.
- אם אין מעגלים שליליים – אז $D^{(n-1)} = D^{(n)}$ וזה הפלט שאנחנו רוצים לקבל.
- אם יש מעגלים שליליים – אז אם u מופיע במעגל שלילי נקבל כי $D_{uu}^{(n)} < 0$.
- כלומר, **אנחנו צריכים לקחת את המטריצה L ולהעלות אותה בחזקת n ביחס לכפל min-plus.**

$D_{uv}^{(k)} = \delta_k(u,v)$ - length of the shortest path from u to v that uses *at most* k edges.

$$D_{uv}^{(1)} = \ell(u,v)$$
$$D_{uv}^{(k)} = \min_{w \in V} D_{uw}^{(k-1)} + \ell(w,v)$$

Equivalent to running BF from all vertices

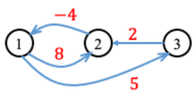
We assume that $\ell(v,v) = 0$, so $D_{uv}^{(k)} \le D_{uv}^{(k-1)}$.

$$D^{(k)} = D^{(k-1)} * L = L^k$$

<u>repeated squaring:</u>

- חישוב רגיל עולה $O(n^4)$.
- פתרון מהיר יותר הוא לחשב ... $L, L^2, L^4 = (L^2)^2$ עד שנגיע לחזקה $2^k \ge n$. כך נקבל סיבוכיות של $O(n^3 logn)$.
- אפשר לחשב גם אם החזקה היא לא של 2: נסתכל על הייצוג הבינארי של n: אם הביט ה-i הוא 1 אפשר להכפיל את כל החזקות האלה.
- אפשר לשנות את האלגוריתם כך שנקבל גם עץ של מסלולים קצרים ביותר ולא רק את המרחקים.

Example I – no negative cycles



$$L = \begin{pmatrix} 0 & 8 & 5 \\ -4 & 0 & \infty \\ \infty & 2 & 0 \end{pmatrix}$$

$$L^2 = \begin{pmatrix} 0 & 8 & 5 \\ -4 & 0 & \infty \\ \infty & 2 & 0 \end{pmatrix} * \begin{pmatrix} 0 & 8 & 5 \\ -4 & 0 & \infty \\ \infty & 2 & 0 \end{pmatrix} = \begin{pmatrix} 0 & 7 & 5 \\ -4 & 0 & 1 \\ -2 & 2 & 0 \end{pmatrix}$$

$$L^3 = \begin{pmatrix} 0 & 8 & 5 \\ -4 & 0 & \infty \\ \infty & 2 & 0 \end{pmatrix} * \begin{pmatrix} 0 & 7 & 5 \\ -4 & 0 & 1 \\ -2 & 2 & 0 \end{pmatrix} = \begin{pmatrix} 0 & 7 & 5 \\ -4 & 0 & 1 \\ -2 & 2 & 0 \end{pmatrix}$$

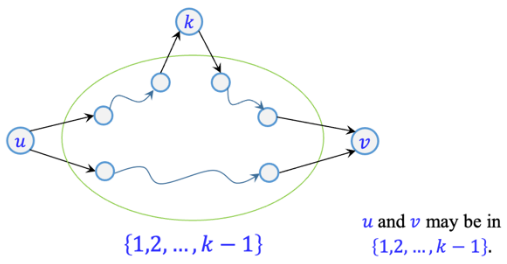
אלגוריתם Floyd-Warshall

נגדיר $F_{uv}^{(k)}$ להיות האורך של המק"ב מ-u ל-v **שצמתי הביניים** שלו הם בטווח $\{1,\dots,k\}$. המטריצה L היא $F^{(0)}$.

אלגוריתם:

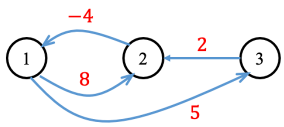
1. נקבל את נוסחת הנסיגה הבאה: נניח שכבר חישבנו את המטריצה $F^{(k-1)}$. לכל $k=1,\dots,n$ נחשב באופן הבא: $F_{uv}^{(k)}=\min\{F_{uv}^{(k-1)},F_{uk}^{(k-1)}+F_{kv}^{(k-1)}\}$ לכל $u,v\in V$.
2. נחזיר את $D=F^{(n)}$.

$$F_{uv}^{(k)} \leftarrow \min\left\{F_{uv}^{(k-1)}, F_{uk}^{(k-1)}+F_{kv}^{(k-1)}\right\}\ ,\ \text{for every } u,v\in V.$$



נכונות: אנו מבצעים מינימום בין האפשרות לא לעבור ב-k, לבין האפשרות לעבור ב-k: נגיע ל-k ונמשיך ממנו עם מסלולים שכל צמתי הביניים שלהם הם מהקבוצה $\{1,\dots,k-1\}$. אנחנו מניחים ש-k מופיע פעם אחת (במק"בים אפשר להניח שהמסלול הוא פשוט, אם הוא היה מופיע פעמיים, מההנחה שאין מעגלים שליליים, אם נוריד את המעגל נקבל מסלול קצר יותר).

סיבוכיות: זמן הריצה הוא $O(n^3)$, אין כאן repeated squaring. כדי לחשב את האיבר uv אנחנו עושים מספר קבוע של פעולות. לכן יש לנו n פעמים n^2. זה משתווה לזמן הריצה של Johnson אם $m=\Theta(n^2)$. בגרפים דלילים ($m \ll n^2$) הוא פחות יעיל מ-Johnson.



$$F^{(0)}=L=\begin{pmatrix}0&8&5\\-4&0&\infty\\\infty&2&0\end{pmatrix}\qquad F^{(1)}=\begin{pmatrix}0&8&5\\-4&0&-1\\\infty&2&0\end{pmatrix}$$

$$F^{(2)}=\begin{pmatrix}0&8&5\\-4&0&-1\\-2&2&0\end{pmatrix}\qquad F^{(3)}=\begin{pmatrix}0&7&5\\-4&0&-1\\-2&2&0\end{pmatrix}$$

APSP (תרגול 7)

1) מספר זוגי של קשתות אדומות: נתונים גרף מכוון $G=(V,E)$, ופונקציית משקל $w:E\to\mathbb{R}$. בנוסף, כל קשת בגרף צבועה באדום או בכחול. ידוע שאין מעגלים שליליים בגרף. תארו אלגוריתם למציאת המק"בים בין כל זוג קודקודים בגרף, מבין המסלולים המכילים מספר זוגי של קשתות אדומות.

פתרון 1: בדומה לפתרון הבעיה בתרגול 6, נשכפל את הגרף, קשתות אדומות עוברות צד, וקשתות כחולות נשארות באותו צד. על הגרף החדש נפעיל את האלגוריתם של Johnson: נקבל את כל המק"בים אבל כל הקודקודים בזמן $O(|V||E|+|V|^2\log|V|)$.

פתרון 2: ניעזר באלגוריתם Floyd-Warshall: באלגוריתם בונים סדרה של מטריצות כך שהמטריצה $D^{(k)}$ מתאימה למשקלי מק"בים מבין המסלולים שעוברים רק דרך קודקודים פנימיים מהקבוצה $\{1,\dots,k\}$. איבר נבנה באופן הבא: או שהאיבר k מופיע (נפצל למסלול בין i ל-k, ומסלול בין k ל-j) או שהאיבר k לא מופיע. אנחנו מניחים ש-k מופיע פעם אחת (מק"ב הוא מסלול פשוט כיוון שאין מעגלים שליליים, אם היה מעגל היינו יכולים להוריד אותו ולקבל מסלול קצר ביותר. **לכן אנחנו יודעים שקיים מסלול קצר ביותר ש-k מופיע בו רק פעם אחת**).

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

שימוש ב- k אינו עוזר לנו. | קיים מסלול קל יותר שעובר דרך k.

אלגוריתם:

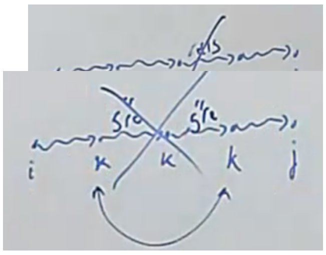
נשנה את האלגוריתם של FW כך שיבנה סדרה של מטריצות E בעלי מספר קשתות אדומות זוגי (מתחילים ומסיימים באותו צד), וסדרה של מטריצות O בעלי מספר קשתות אדומות אי-זוגי. אנו זקוקים לשני הסוגים כדי לחשב: נחשוב על כל הדרכים השונות לקבל סכום זוגי של קשתות:

- $E^{(0)},E^{(1)},...,E^{(n)}$, עבור משקלי מק"בים מבין המסלולים המכילים מספר זוגי של קשתות אדומות.
- $O^{(0)},O^{(1)},...,O^{(n)}$, עבור משקלי מק"בים מבין המסלולים המכילים מספר אי-זוגי של קשתות אדומות.

$$e_{i,j}^{(k)}=\min\begin{cases}e_{i,j}^{(k-1)}\\ e_{i,k}^{(k-1)}+e_{k,j}^{(k-1)}\\ o_{i,k}^{(k-1)}+o_{k,j}^{(k-1)}\end{cases}\qquad o_{i,j}^{(k)}=\min\begin{cases}o_{i,j}^{(k-1)}\\ o_{i,k}^{(k-1)}+e_{k,j}^{(k-1)}\\ e_{i,k}^{(k-1)}+o_{k,j}^{(k-1)}\end{cases}$$

נשים לב שכאן **המק"ב לא בהכרח מסלול פשוט**, ההנחה מקודם לא נכונה כיוון שאנחנו עובדים עם FW ולא עם Johnson. לכן צריך לטפל במקרה הזה שהמק"ב הוא לא פשוט.

נכונות: אם קיים מסלול מ-i ל-j עם מספר זוגי של קשתות אדומות כך ש-k מופיע 3 פעמים או יותר, ניתן להסיר את אחד המופעים ולקבל מסלול לא יותר ארוך.



- אם אחד המעגלים הוא בעל **מספר זוגי של קשתות אדומות**, נוכל **להסיר** אותו ונקבל עדיין מסלול עם מספר זוגי של קשתות אדומות, קצר יותר.
- אחרת, אם שני המעגלים הם בעלי מספר **אי-זוגי של קשתות**, נוכל **להסיר את שניהם** כי המעגל הארוך יותר הוא בעל מספר זוגי של קשתות.

מסקנה: אפשר להניח בה"כ שהמקרים היחידים שמעניינים הם 0, 1 או 2 מופעים.

סיבוכיות: חישוב איבר בודד ב-$O(1)$, חישוב מטריצה שלמה $O(|V|^2)$, ויש לנו V מטריצות לחשב. סה"כ $\boldsymbol{O(|V|^3)}$.

2) מסלול באורך m:

רקע: סגור טרנזיטיבי זה ייצוג של ההגעתיות של הגרף (reachability): מטריצה A כך שקיים מסלול מ-i ל-j בגרף $a_{i,j}=1 \Leftrightarrow$. האלגוריתם המקורי של Warshall מוגדר על גרף לא ממושקל והמטרה שלו היה לחשב סגור טרנזיטיבי.

Floyd לקח את האלגוריתם ושינה אותו קצת כדי לחשב מק"בים: **הוא הגדיר תת-בעיה אחרת, במקום מינימום יש או, ובמקום + יש וגם.** זו הגדרה רקורסיבית להאם אפשר להגיע מ-i ל-j עם קודקודים $\{1,...,k\}$.

"במקום" לחשב (על גרף ממושקל) את

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

הוא מחשב (על גרף בוליאני) את

$$d_{i,j}^{(k)}=\vee\left\{d_{i,j}^{(k-1)},d_{i,k}^{(k-1)}\wedge d_{k,j}^{(k-1)}\right\}$$

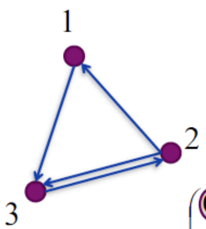
תרגיל: נתון גרף מכוון $G=(V,E)$ ומספר שלם $m\in\{1,...,|V|-1\}$. תארו אלגוריתם שבודק לכל זוג קודקודים $u,v\in V$ האם קיים מסלול באורך m מ-u ל-v. המסלול לא חייב להיות פשוט ויכול לעבור דרך קשת יותר מפעם אחת.

אלגוריתם:

- עבור $m=1$ נבדוק האם יש קשת (u,v). נעבוד עם מטריצת שכנויות של הגרף ונסתכל בתא המתאים אם יש קשת.
- נגדיר את תת-הבעיה: נסמן מטריצה $A^{(*t)}$ שמופיע בה 1 אם יש מסלול באורך t ו-0 אחרת. ראינו כי $A^{(*1)}=A$ (אנחנו רק מחפשים קשתות ישירות בין הצמתים). נשים לב שמכפלה בוליאנית בודקת לנו את כל האפשרויות למסלולים על ידי הרכבה של קשת אחת על השנייה. בכל מכפלה כזו אנו "עוברים" על 3 אופציות למסלולים בעלי 2 קשתות בין צמתים.

נחשב את A^m ביחס לכפל בוליאני של מטריצות. מכפלה בוליאנית של מטריצות $A\cdot B=C$ מוגדרת כך:

$$c_{i,j}=\bigvee_{k=1}^{n}\left(a_{i,k}\wedge b_{k,j}\right)$$



$$\begin{pmatrix}0&0&1\\1&0&1\\0&1&0\end{pmatrix}\begin{pmatrix}0&0&1\\1&0&1\\0&1&0\end{pmatrix}=\begin{pmatrix}0&1&0\\0&1&1\\1&0&1\end{pmatrix}$$

נכונות: טענה: $A^m = A^{(*m)}$. נוכיח באינדוקציה על m.

- בסיס: אם $m = 1$ ברור כי $A^{(*1)} = A = A^1$. אכן, קיים מסלול $\Leftrightarrow$ קיימת קשת בין שני קודקודים.
- צעד: עבור $m > 1$ נניח כי $A^{m-1} = A^{(*m-1)}$. נראה כי $A^{(*m)} = A^m$. נרצה להוכיח שקיים מסלול באורך m מקודקוד i לקודקוד j $\Leftrightarrow$ $a_{i,j}^m = 1$.
  - $\Leftarrow$ נניח שקיים הילוך באורך m. נסמן את הקודקוד האחד לפני אחרון במסלול ב-ℓ. מההגדרה יש מסלול באורך $m-1$ בין i ל-ℓ ולפי הנחת האינדוקציה מתקיים $a_{i,\ell}^{m-1} = 1$. אנחנו יודעים כי לפי הקשת האחרונה מתקיים $a_{\ell,j} = 1$. בפרט עבור $k = \ell$ יתקיים כי $a_{i,k}^{m-1} \wedge a_{k,j} = 1$ וכך נקבל 1 בתא המתאים.
  - $\Rightarrow$ נניח שכתוב במטריצה 1. זה אומר קיים k שעבורו תוצאת ה"וגם" היא 1. כלומר יש קשת בסוף, ומהנחת האינדוקציה יש הילוך באורך $m-1$ עד תחילת הקשת הזו. אם נחבר את הקשת ואת ההילוך הנ"ל נקבל הילוך באורך m.

סיבוכיות: מחשבים $m-1$ מטריצות עם $O(|V|^2)$ תאים. חישוב ערך בודד לוקח $O(|V|)$ זמן כי אנחנו עוברים על n ביטויים. סה"כ נקבל כי הסיבוכיות היא $O(|V|^3 m)$.

איך נשפר? במקום להכפיל את כל המטריצות, נבצע repeated squaring. אחרי השיפור נבצע $O(logn)$ פעולות כפל. זמן הריצה יורד ל-$O(|V|^3 logn)$.

<u>וריאנט של השאלה:</u> אם נשתמש באלגוריתם מהשאלה הקודמת עם כפל סטנדרטי של מטריצות, נקבל את מספר ההילוכים באורך m מ-i ל-j. אפשר להוכיח באינדוקציה על m. בפועל אנחנו מחלקים למקרים לפי הקשת האחרונה, אם היא קיימת (1) אנחנו סוכמים את כל המסלולים עד לקשת זו באורך $m-1$.

<u>תכנות דינמי</u>

מבוא

תכנות דינמי היא שיטה לפתרון בעיות אופטימיזציה.

- נראה שניתן לפרק את הבעיה לאוסף של תתי-בעיות יותר קטנות. נשים לב, כי בתכנות דינמי **הבעיות הקטנות יכולות להיות חופפות** (זאת לעומת שיטה אחרת, "הפרד ומשול" בה תתי-הבעיות לרוב זרות, כמו merge sort, ממיינים כל חצי מהמערך בנפרד וצריך רק למזג).
- נקבל **נוסחת נסיגה** שמאפשרת לחשב את הערכים האופטימליים.
- נפתור את נוסחת הנסיגה, ונשתמש בממואיזציה או ישירות במערך/וקטור/מטריצה כך שלא נצטרך לחשב ערכים יותר מפעם אחת.

קבוצה בלתי-תלויה מקסימלית: עבור גרף $G = (V, E)$, קבוצה בלתי תלויה $I \subseteq V$ היא קבוצה שבה בין כל שני צמתים אין קשת.

בעיית מציאת קבוצה בלתי תלויה מקסימלית היא NP-hard (אם הופכים קשת ללא-קשת, הגרף המשלים, זה שקול למציאת קליקה). נראה שאם הגרף עליו אנחנו מדברים הוא **מסלול (מערך של מספרים), הבעיה היא קלה**. נניח שלכל **צומת יש משקל**. נרצה למצוא קבוצה בלתי-תלויה שסכום המשקלים שלה הוא מקסימלי.

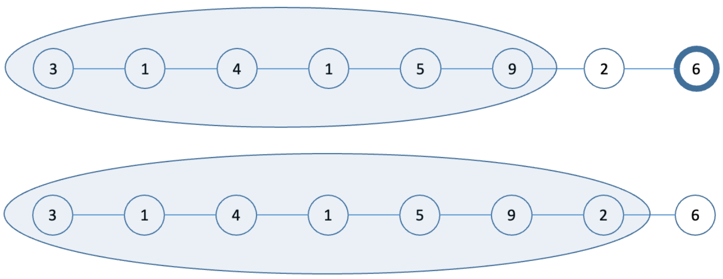
האם אנחנו רוצים להוסיף לקבוצה את הצומת האחרון או לא? אם **כן**, נרצה להסתכל על הפתרון האופטימלי **עבור $n-2$ הצמתים שנותרו** (אסור לבחור את הצומת שצמוד לצומת שבבחרנו). אחרת, אם **לא** בוחרים אותו, אנחנו רוצים את הפתרון האופטימלי **עבור $n-1$ הצמתים**. פירקנו לשתי תתי-בעיות שיש ביניהן חפיפה גדולה.

נגדיר opt(i) בתור הקבוצה הבלתי-תלויה המקסימלית שהיא תת-קבוצה של i הצמתים הראשונים.

Let $w_i \geq 0$ be the weight of the i-th vertex on the path.

Let $opt(i)$ be the maximum weight of an independent set $I \subseteq \{1,2,\dots,i\}$

$$opt(0) = 0 \ , \ \ opt(1) = w_1$$
$$opt(i) = \max\{ w_i + opt(i-2) \ , \ opt(i-1)\} \ , \ \ 2 \leq i \leq n$$



LCS

LCS (Longest Common Subsequence): נתונות שתי סדרות $X = x_1 \dots x_m, Y = y_1 \dots y_n$, נרצה למצוא תת-סדרה ארוכה ביותר Z שהיא תת-סדרה גם של X וגם של Y.

- נסמן ב-$X_k = x_1 \dots x_k$ את הרישא בגודל k של X, וב-$Y_j = y_1 \dots y_j$ את הרישא בגודל j של Y.
- נסמן ב-$lcs(k,j)$ את אורך ה-LCS של X_k, Y_j. אנחנו מעוניינים במציאת $lcs(m,n)$.
- אם האיברים הבאים שווים אז נוסיף 1, ואם לא, ננסה להתקדם ב-X בנפרד וב-Y בנפרד ולקחת את המקסימלי.

$$lcs(k,\ell) = \begin{cases} 0 & k=0 \text{ or } \ell=0 \\ lcs(k-1,\ell-1)+1 & k,\ell>0\,,\, x_k = y_\ell \\ \max\{\, lcs(k-1,\ell)\,, lcs(k,\ell-1)\,\} & k,\ell>0\,,\, x_k \neq y_\ell \end{cases}$$

If $x_k = y_\ell$ then $LCS(X_k, Y_\ell)$ definitely includes the common last item.
If $x_k \neq y_\ell$ then it may include one of x_k and y_ℓ but not both.

נמלא את המטריצה שורה אחר שורה. כשמסתכלים על ההגדרה, כדי למלא את התא (k,j) נצטרך להסתכל על האלכסון שמאלה אם $x_k = y_j$ או שנסתכל תא אחד למעלה או שמאלה. למשל בשורה 1 באיבר 4 זו הפעם הראשונה שהאיברים שווים ולכן נסתכל על האלכסון שמאלה.

ה-LCS הוא באורך 4. ניעזר בחיצים ונלך איתם. כל פעם שיש חץ שמאלה למעלה זה אומר שלקחנו את האיבר המתאים ב-X וב-Y. לפי הסדר נקבל את A, ואז B, ואז C, ולבסוף B. לכן הסדרה היא $BCBA$.

כיוון שיש לנו חופש בחיצים אם היינו בוחרים אחרת היינו מקבלים LCS אחר.

סיבוכיות: $O(n^2)$.

	ℓ	0	1	2	3	4	5	6
k		ϵ	B	D	C	A	B	A
0	ϵ	0	0	0	0	0	0	0
1	A	0	0	0	0	1	1	1
2	B	0	1	1	1	1	2	2
3	C	0	1	1	2	2	2	2
4	B	0	1	1	2	2	3	3
5	D	0	1	2	2	2	3	3
6	A	0	1	2	2	3	3	4
7	B	0	1	2	2	3	4	4

LIS

LIS (Longest Increasing Subsequence): נתונה סדרה אחת $X = x_1 \dots x_n$, נרצה למצוא תת-סדרה $X' = x_{i_1} \dots x_{i_k}$ **עולה ממש**, כלומר $x_{i_1} < \cdots < x_{i_k}$.

אבחנה: LIS של X הוא LCS של X ו-sort(X) (אם ניקח תת-סדרה של המיון של X זה מחייב אותה להיות עולה). זה בזמן $O(n^2)$. אפשר למצוא טריק ולהשתמש בחיפוש בינארי כדי להוריד את זמן הריצה ל-$O(nlogn)$.

פתרון 1:

- תת-בעיה: נסתכל על הרישא באורך k שנסמן ב-X_k, ונשאל מה ה-LIS של X_k. זה לא בדיוק עובד. לחילופין נגדיר את $len(k)$ להיות האורך של **ה-LIS של X_k שמסתיים ב-x_k**. אנחנו מחפשים את $lis(n) = \max\{len(1), \dots, len(n)\}$.
- נוסחת נסיגה: $\boldsymbol{len(1) = 1, len(k) = \max\{len(i) | 1 \le i < k | x_i < x_k\} + 1,\ \ 1 < k < n}$. תת-הסדרה הכי ארוכה של X_k שנגמרת ב-x_k: חייבים לקחת את האיבר האחרון, כל האיברים הקודמים של תת-הסדרה חייבים להיות קטנים מ-x_k (אחרת זה לא יהיה עולה). ניקח את ה-len המקסימלי ונוסיף 1.
- דוגמה: למשל $len(3)$ צריכים להסתכל על כל האיברים $x_i < x_3 = 3$, המקסימלי מבניהם הוא $len(1)$ ובתוספת 1 נקבל 2. יש שתי אפשרויות לאיבר האחרון ב-LIS (9 או 8), ו-5 מתקבל בשניהם לכן יש שתי אפשרויות. גם 4 מופיע פעמיים.
- כדי לקבל את ה-LIS עצמו צריך לחשב ערכי π (מיהו ה-i שהשיג את המקסימום):
- סיבוכיות: עדיין $O(n^2)$.

k	1	2	3	4	5	6	7	8	9
X	2	4	3	5	1	7	6	9	8
len	1	2	2	3	1	4	4	5	5

Let $\pi(k) = \text{argmax}\{\, len(i) \mid 1 \le i < k\,,\ x_i < x_k \,\}$.

k	1	2	3	4	5	6	7	8	9
X	2	4	3	5	1	7	6	9	8
len	1	2	2	3	1	4	4	5	5
π	-	1	1	2	-	4	4	6	6

The LIS of X_k ending with x_k is $\dots x_{\pi(\pi(k))} x_{\pi(k)} x_k$

<u>פתרון 2:</u>

- <u>תת-הבעיה:</u> במקום $len(k)$ נגדיר (ℓ) **min: האיבר הכי קטן x_k כך ש-$len(k) = \ell$, או ∞.** כלומר, נניח שאנחנו רוצים LIS באורך ℓ, מה האיבר הכי קטן שיכול להיות האיבר האחרון בסדרה הזאת.
- דוגמה:
  - $\min(1) = 1$: נחפש את ה-x_k הקטן ביותר שמסיים סדרה באורך 1.
  - $\min(2) = 3$: מיהו ה-x_k הקטן ביותר שמסיים סדרה באורך 2. יש הרבה תתי-סדרות עולות באורך 2, האיבר הכי קטן הוא 3: יש לנו את תת-הסדרה $3 \to 2$.
- כאן הסדרה min **היא כן עולה ממש**.

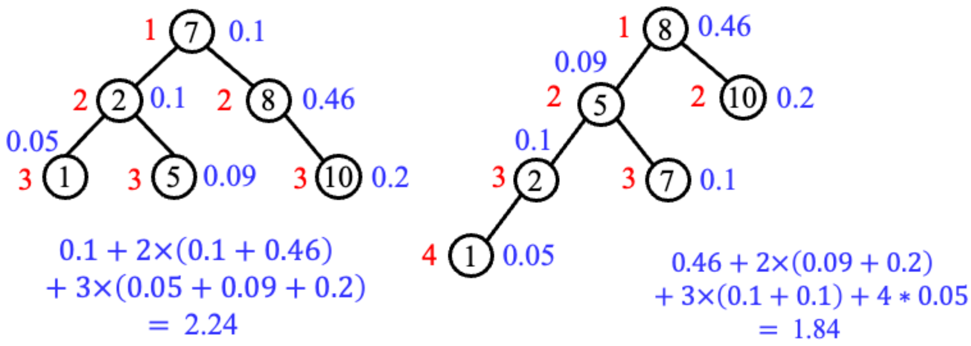
k	1	2	3	4	5	6	7	8	9
X	2	4	3	5	1	7	6	9	8
len	1	2	2	3	1	4	4	5	5
min	**1**	**3**	**5**	**6**	**8**	∞	∞	∞	∞

- <u>נוסחת נסיגה:</u> נניח שחישבנו את המערך min עבור הסדרה X_{k-1}. בעזרת חיפוש בינארי (לא צריך לעבור איבר-איבר), נחפש את האינדקס i עבור $\mathbf{min}(\boldsymbol{i}-\mathbf{1}) \leq \boldsymbol{x_k} < \mathbf{min}\ (\boldsymbol{i})$. נמצא איפה ביחס לסדרת ה-min נופל האיבר החדש x_k. אחרי שמצאנו את האינדקס היחיד הזה נגדיר $\min(i) = x_k$. זמן הריצה הוא $\boldsymbol{O(nlogn)}$.
- דוגמה: השורות זה המצב של המערך min בשלבים שונים של הריצה. אחרי כל איבר שאנחנו קוראים אנחנו רואים איך צריך לעדכן את המערך min. מאתחלים את כל הערכים ל-∞.

k	X	min(1)	min(2)	min(3)	min(4)	min(5)	min(6)	,,,
1	2	2						
2	4	2	4					
3	3	2	3					
4	5	2	3	5				
5	1	1	3	5				
6	7	1	3	5	7			
7	6	1	3	5	6			
8	9	1	3	5	6	9		
9	8	1	3	5	6	8		

Optimal Static BST

<u>Optimal Static BST:</u> יש הרבה BST-ים שמכילים איברים $x_1 < \cdots < x_n$. אם נסתכל על איבר מסוים x_i, חיפוש שלו הוא פרופורציונלי לעומק שלו. נגדיר את העומק של השורש ב-1, וכל רמה מתחת עם עומק שהולך וגדל. נניח שנותנים לנו סדרה של הסתברויות $p_1, \dots, p_n$ שמעידה על ההסתברות שמחפשים כל איבר. תוחלת זמן החיפוש ב-BST הוא $\sum_{i=1}^{n} p_i \cdot depth(x_i)$. נרצה למצוא את **העץ בעל תוחלת זמן החיפוש המינימלית**. אנחנו מסתכלים כאן על **הבעיה הסטטית** – אין הכנסות ואין הוצאות, אנחנו יודעים מי האיברים שנמצאים במילון.



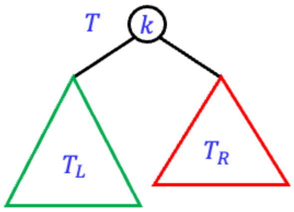
<u>תתי הבעיות:</u> הקלט הוא סדרה של n מספרים, והפלט צריך להיות עץ שתוחלת זמן החיפוש בו היא קטנה ביותר. נניח שניקח את x_k בתור שורש. האיברים שנמצאים בתת-העץ השמאלי $x_1, \dots, x_{k-1}$ והימני $x_{k+1}, \dots, x_n$ נקבעים לפיו. עכשיו עלינו לפתור את הבעיה על כל תת-עץ בנפרד. ניקח את תוחלת זמן החיפוש ונפריד לשלושה מקרים:

- מחפשים את k: המחיר הוא 1 וזה קורה בהסתברות p_k.
- חיפוש בתת-העץ השמאלי.
- חיפוש בתת-העץ הימני.

$$cost(T) = \sum_{j=1}^{n} p_j\, depth_T(x_j)$$

$$= p_k + \sum_{j=1}^{k-1} p_j\left(depth_{T_L}(x_j)+1\right) + \sum_{j=k+1}^{n} p_j \cdot \left(depth_{T_R}(x_j)+1\right)$$

$$= \left(\sum_{j=1}^{n} p_j\right) + cost(T_L) + cost(T_R)$$



Thus, T_L is an optimal BST for $x_1, x_2, \dots, x_{k-1}$,
and T_R is an optimal BST for $x_{k+1}, x_{k+2}, \dots, x_n$.

We do not assume now that $\sum_{j=1}^{n} p_j = 1$.

כלומר כדי למצוא את העץ T האופטימלי, צריך למצוא את תתי-העצים האופטימליים. זה מריח כמו תכנות דינמי. בהינתן ההחלטה מי יהיה בשורש, הבעיה מתפרקת לשתי תתי-בעיות מאותו סוג.

<u>נוסחת נסיגה:</u> נגדיר את $p(i,j) = \sum_{k=i}^{j} p_k$ הסכום של כל ההסתברויות בין $p_i, \dots, p_j$. נגדיר את $C(i,j)$ כמחיר של העץ האופטימלי עבור $x_i, \dots, x_j$. נקבל את נוסחת הנסיגה הבאה: העץ האופטימלי על האיברים מ-1 עד n, זה הסכום של כל ה-p, ועוד המינימום על כל ה-k: הסכום של הפיצול לשתי תתי-העצים.

$$C(1,n) = p(1,n) + \min_{1 \le k \le n} C(1,k-1) + C(k+1,n)$$

$$C(i,j) = p(i,j) + \min_{i \le k \le j} C(i,k-1) + C(k+1,j) \quad , \quad i < j$$

$$C(i,i-1) = 0 \quad , \quad C(i,i) = p_i$$

נוח להוסיף עמודה 0 למטריצה. האלכסון $C(i,i-1) = 0$, ו-$C(i,i) = p_i$. נניח שאנחנו רוצים את $C(3,5)$. אנחנו צריכים לחשב את תתי-הבעיות של $C(3,2 \dots 5)$ ואת $C(3 \dots 6,5)$. נחשב אלכסון-אלכסון. כדי למצוא את העץ האופטימלי עצמו, נשמור גם מטריצה $K(i,j)$ עבור ה-k שהשיג את המינימום בחישוב של $C(i,j)$.

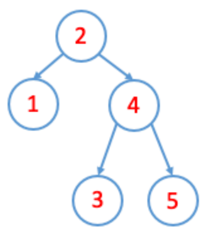
1	2	3	4	5
0.4	0.15	0.10	0.15	0.2

$p(i,j)$

	1	2	3	4	5
1	0.4	0.55	0.65	0.8	1
2		0.15	0.25	0.4	0.6
3			0.10	0.25	0.45
4				0.15	0.35
5					0.2

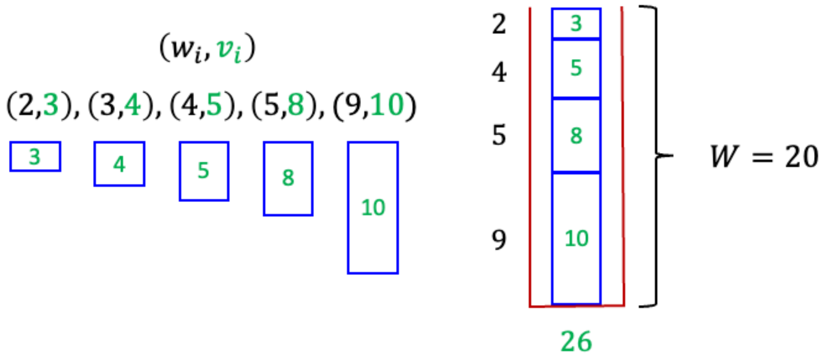
$C(i,j)$

	0	1	2	3	4	5
1	0	0.4	0.70 ^{1}	1.0 ^{1}	1.5 ^{1}	2.15 ^{2}
2		0	0.15	0.35 ^{2}	0.70 ^{3}	1.15 ^{4}
3			0	0.10	0.35 ^{4}	0.75 ^{4}
4				0	0.15	0.50 ^{5}
5					0	0.2
6						0



זמן הריצה יהיה $O(n^3)$. אפשר לשפר ל-$O(n^2)$.

Knapsack



:Knapsack יש לנו תרמיל שמסוגל לשאת עד W קילוגרמים. יש n פריטים שנרצה לקחת. לפריט i יש משקל w_i וערך v_i. נרצה לקחת את תת-קבוצה S **שהמשקל הכולל שלה הוא לכל היותר W והערך שלה הוא הכי גדול**: $max\sum_{i\in S} v_i$, $\sum_{i\in S} w_i \leq W$.

בצורה חמדנית נוכל להסתכל על כל איבר מבחינת ערך ליחידת משקל, וניקח את האיברים עם הערכים הכי גדולים הללו. זה יוביל לפתרון לא אופטימלי. אם הבעיה הייתה fractional אז זה היה עובד.

תתי-בעיות: נקטין את מספר הפריטים – k הראשונים, ונגדיל את גודל ה-Knapsack. נסמן ב-$B(k, W')$ להיות הפתרון האופטימלי עבור $(w_1, v_1), \ldots, (w_k, v_k), W'$.

נוסחת הנסיגה: אם $w_k > W$ לא נוכל לקחת אותו. אחרת, נשקול האם אנחנו רוצים לקחת אותו או לא (נחשב מקסימום).

$$B(k,w) = \begin{cases} B(k-1,w) & w_k > w \\ \max\{ B(k-1,w) \ , \ v_k + B(k-1, w-w_k) \} & w_k \leq w \end{cases}$$

$$B(0,w) = 0 \ , \ \ 0 \leq w \leq W$$

$$B(i,0) = 0 \ \ , \ \ 1 \leq i \leq n$$

Can compute $B(k,w)$ row by row.

The complexity is $O(nW)$.

Not polynomial! Said to be pseudo-polynomial.

תכנות דינמי (תרגול 8)

הפתרון לשאלות תכנות דינמי מורכב ממספר שלבים קבועים:

1. הגדרת תתי בעיות (לפעמים זה טריוויאלי, ולפעמים זה מורכב).
2. מציאת הקשר בין תתי הבעיות שהגדרנו לפתרון השאלה.
3. חישוב תתי הבעיות הקטנות ביותר ("תנאי עצירה").
4. חישוב תתי בעיות באמצעות תתי בעיות קטנות מהן ("צעד", שלב מאתגר). **נרצה לבצע הנחה מקלה ואז להיפטר ממנה.**
5. ניתוח זמן הריצה.

לא מוכיחים נכונות!

תרגיל 1 (יופי של עודף): נתונים n מטבעות מסוג שקלים מאוד חדשים מאוד (שמ"ח) בשווי $1 = a_1 < a_2 < a_3 < \cdots < a_n$. אתם מוכרים בלון ב-4 שמ"ח, וקיבלתם שטר של $x \leq 100n^3$ שמ"ח. תארו אלגוריתם שיחשב את מספר המטבעות המינימאלי שתצטרכו בשביל להחזיר עודף מדויק. דוגמה: המטבעות הם 1,5,7 ו-$x = 14$. צריך להחזיר 10 שמ"ח עודף. האפשרויות הן 7,1,1,1 ו-5,5. נבחר את המינימלית מביניהן, מספר המטבעות הוא 2.

שלב	פתרון
1	טריוויאלי: נגדיר את $c[k]$ להיות מספר המטבעות המינימאלי עבור עודף ששווה k.
2	הפתרון הוא $c[x-4]$.
3	נגדיר $c[0] = 0$.
4	נדמיין את הפתרון האופטימלי עבור $c[k]$. נרצה ללמוד משהו על המבנה של הפתרון. נעשה הנחה מקלה – **נניח שיודעים ש-a_i הוא מטבע שהוחזר בפתרון האופטימלי.** אז $c[k] = c[k-a_i] + 1$ כי בלעדיו הפתרון צריך להיות גם אופטימלי. אנחנו לא יודעים מי ה-a_i הזה, אז נעבור על כל אחד מהם וניקח את הכי טוב מביניהם (המינימלי): $c[k] = \min_{1\leq i\leq n} c[k-a_i] + 1$ יכול להיות לנו ערך שלילי. לכן נצטרך להוסיף $\forall x < 0, c[x] = \infty$.
5	חישוב של $c[k] = \min_{1\leq i\leq n} c[k-a_i] + 1$ לוקח $O(n)$. נחשב $O(n^3)$ ערכים כי $x \leq 100n^3$ ולכן סה"כ $\boldsymbol{O(n^4)}$.

תרגיל 2 (בעיית המנורות הכללית): אנחנו רוצים לדעת מאיזו קומה נפתחת מנורה כשזורקים אותה מבניין בגובה n קומות. לרשותנו m מנורות. אם זורקים מנורה מהקומה הנכונה, היא נפתחת. אם זורקים מקומה נמוכה מידי, לא קורה לה כלום ואפשר לזרוק אותה שוב. אם זורקים מקומה גבוהה מידי, המנורה נשברת. כמה זריקות נצטרך לכל היותר בכדי לדעת באיזו קומה מנורה נפתחת?

שלב	פתרון
1	טריוויאלי: נגדיר את $c(k,j)$ להיות מספר הזריקות שנצטרך עבור k מנורות ו-j קומות.
2	הפתרון הוא $c(m,n)$.
3	אם יש לנו מנורה אחת, אנחנו צריכים לעבור קומה-קומה: $\forall j: c(1,j) = j$. אם יש לנו רק קומה אחת, נזרוק מנורה וסיימנו: $\forall k: c(k,1) = 1$.
4	נדמיין את הפתרון האופטימלי עבור j קומות ו-k מנורות. נעשה הנחה מקלה - נניח שאנחנו יודעים מה הצעד הבא של הפתרון – **נניח שיודעים שהפתרון האופטימלי זורק מקומה i**. אם הוא זרק מ-i: יש 2 אופציות, נשבר או לא נשבר.  • אם לא נשבר – בוודאות הג'יני בין i ל-j: $c(k,j) = c(k,j-i)$. • אם נשבר – בוודאות הג'יני מתחת לקומה i ומנורה אחת פחות: $c(k,j) = c(k-1,i-1)$. על שני הערכים האלה נרצה לקחת מקסימום – המטרה שלנו זה ה-WC. מבין כל ה-WC האלה אנחנו רוצים את הכי טוב, כלומר את המינימום על כל $1 \le i \le j$. נקבל: $c(k,j) = \min_i\{\max[c(k,j-i), c(k-1,i-1)]\} + 1$
5	המקסימום בין שני ערכים הוא $O(1)$, אבל המינימום הוא על n איברים: לכן $O(n)$ לכל חישוב. אנחנו מחשבים $c(k,j)$ לכל k,j לכן סה"כ mn חישובים. סה"כ נקבל $\boldsymbol{O(mn^2)}$.

תרגיל 3 (יופי של עודף): ועדת קישוט של חברה מארגנת מסיבה, ומנסה להחליט את מי להזמין. בידי הועדה מאגר שמחזיק לכל עובד את רמת הכיפיות שלו. ידוע שעובד אינו כיפי כאשר הבוס הישיר שלו נמצא במסיבה, ולכן הוחלט לא להזמין שני אנשים שאחד מהם הוא הבוס הישיר של השני. לצורך כך, הועדה מחזיקה בעץ המתאר את יחסי העובד-בוס בחברה. **תארו אלגוריתם למציאת תת-קבוצת עובדים שממקסמת את רמת הכיפיות במסיבה.**

נתחיל מתיאור אלגוריתם שמוצא מהי **רמת** הכיפיות המקסימלית. מערך הפתרון אפשר להסיק את קבוצת העובדים.

שלב	פתרון
1	עבור צומת u בעץ, נגדיר את $C(u)$ להיות רמת הכיפיות המקסימאלית במסיבה שבה תת-העץ ששורשו u הוא עץ המוזמנים האפשריים.
2	נסמן את שורש העץ ב-r. אנחנו רוצים למצוא את $C(r)$.
3	נסמן את רמת הכיפיות של עובד u בתור $fun(u)$. אם u עלה בעץ אז $C(u) = fun(u)$.
4	נסמן ב-$child(u)$ את קבוצת הבנים (הישירים) של u בעץ. אם u איננו עלה בעץ. נסתכל על המסיבה האופטימלית. יש 2 מקרים: • u מוזמן (אף אחד מהילדים שלו לא מוזמן): $C(u) = fun(u) + \sum_{w \in grandson(u)} c(w)$. • u לא מוזמן: $C(u) = \sum_{v \in child(u)} c(v)$. ניקח מקסימום בין שני הערכים הללו.
5	נסמן את מספר העובדים בחברה בתור n. נסתכל על u ספציפי: עוברים על כל הבנים שלו ועל כל הנכדים שלו. נסמן $A(u) \subseteq V$ – הבנים שלו, $B(u) \subseteq V$ – הנכדים שלו. עכשיו נקבל כי $T(u) = \|A(u)\| + \|B(u)\| = O(n)$. זה לא הדוק. לא יכול להיות שיש צומת אחד עם שני אבות שונים. עבור שני צמתים הקבוצות A ו-B זרות: $A(u) \cap A(v) = \emptyset = B(u) \cap B(v)$. לכן אם נסכום את זמן הריצה על פני כל ה-$T(u)$, נסכום קבוצות זרות שבסך הכול יש בהם בדיוק n איברים. נקבל $O(n)$.

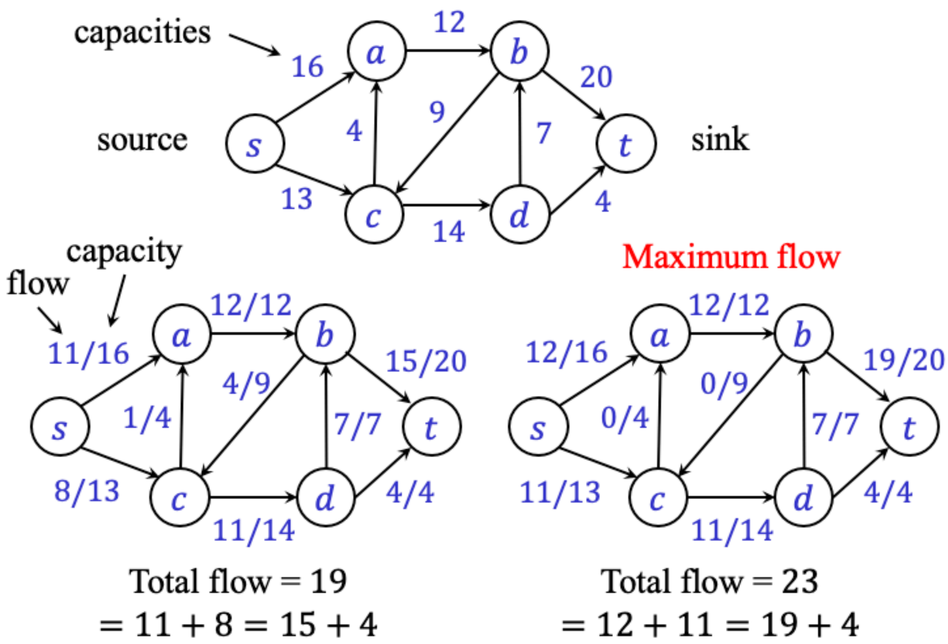
3 – זרימה ותכנות ליניארי

זרימה I

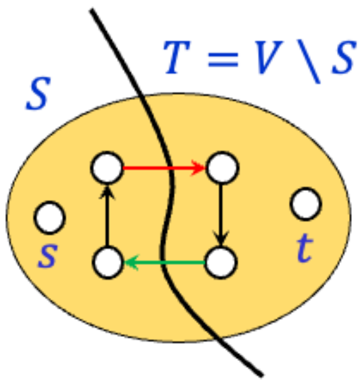
רקע והגדרות

רשת זרימה:

- רשת זרימה: נתונים לנו גרף מכוון $G = (V, E)$, כאשר $c: E \to \mathbb{R}^+$ היא פונקציית קיבול (לכל קשת יש קיבול), צומת מקור $s \in V$, וצומת בור (יעד) $t \in V$.
- קשתות נכנסות ויוצאות: נסמן $in(v)$ עבור קבוצת הקשתות שנכנסות לצומת v, ו-$out(v)$ עבור הקשתות היוצאות ממנו.
- זרימה: פונקציה $f: E \to \mathbb{R}^{\geq 0}$ המתאימה לכל קשת ערך של זרימה, **ומקיימת שני אילוצים:**
  - **אילוצי קיבול/חוק הקשת (capacity constraints)** – הזרימה בכל קשת אינה עולה על קיבול הקשת. כלומר, לכל קשת $e \in E$ מתקיים $0 \leq f(e) \leq c(e)$.
  - **אילוצי שימור זרימה/חוק הצומת (flow conservation constraints)** – בכל צומת הזרימה היוצאת שווה לזרימה הנכנסת (inflow=outflow). כלומר, לכל צומת $v \in V$ שאינו המקור או הבור צריך להתקיים $f(in(v)) = f(out(v))$.
- קשת רוויה (saturated edge): קשת שלא ניתן להגדיל את הזרימה דרכה, כלומר מתקיים $f(e) = c(e)$.
- ערך הזרימה: לכל זרימה f נגדיר $|f| = f(out(s)) - f(in(s))$.
- בעיית הזרימה המקסימלית: נרצה למצוא פונקציית זרימה f חוקית שמביאה למקסימום את הערך $|f|$.



חתכים וזרימה:

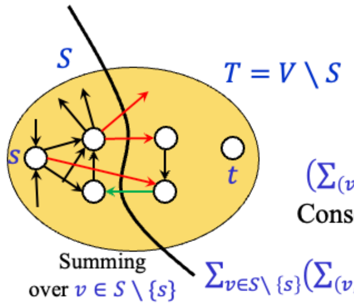


- חתך s-t: חלוקה של G לזוג (S, T) שתי קבוצות צמתים זרות זו לזו, כאשר $s \in S, t \in T$. יעניינו אותנו חתכים שבהם המקור והבור הם בשני צדדים שונים, לצד של המקור נקרא S ולצד של הבור נקרא T. נקבל חלוקה של קשתות הגרף לסוגים שונים: קשתות $E(S,T)$ **חוצות** (crossing) מ-S ל-T, וקשתות $E(T,S)$ **הפוכות** (reverse) מ-T ל-S.
- קיבול של חתך: סכום הקיבולים על **הקשתות החוצות ממנו**: $c(S,T) = \sum_{e \in E(S,T)} c(e)$.
- זרימה בחתך: נסכם את מה שזורם החוצה, ונחסיר את מה שנכנס: $f(S,T) = \sum_{e \in E(S,T)} f(e) - \sum_{e \in E(T,S)} f(e)$. נשים לב כי נוכל גם לרשום את ערך הזרימה בגרף באמצעות החתכים הבאים: $|\boldsymbol{f}| = \boldsymbol{f}(\{\boldsymbol{s}\}, \boldsymbol{V} \backslash \{\boldsymbol{s}\})$. אנו אכן מקבלים את כל מה שיוצא מ-s, ומחסירים את כל מה שנכנס ל-s (מכל שאר הצמתים בגרף).

למה 1 (ערך הזרימה זהה בכל החתכים): אם f היא זרימה מ-s ל-t ברשת זרימה G, ו-(S,T) הוא s-t cut של G, אז מתקיים $|f| = f(S,T)$. כלומר, לא משנה איזה חתך נבחר, ערך הזרימה בחתך הזה יהיה שווה לערך הזרימה בכל הרשת.

הוכחה:

- ניקח את הצמתים בצד שמאל שאינם המקור – הזרימה עליו מקיימת שימור inflow=outflow.
- הפרש ערכי הזרימה הוא 0 בכל צומת. נסכום עבור כל הצמתים שאינם המקור ונקבל 0. נוסיף את ההפרש עבור s (שזו הגדרת $|f|$). נקבל סה"כ את $|f|$.
- מה ההבדל בין מה שקיבלנו לבין חישוב הזרימה בחתך? כל האדומות (חוצות) נמצאות בצד השמאלי, וכל הירוקות (ההפוכות) נמצאות בצד הימני. בעצם, בתוך הסכום שלנו קיבלנו את האדומות פחות הירוקות. כל השחורות (קשתות מסוג אחר) התבטלו.



Proof:

$$\left(\sum_{(v,w)\in E} f(v,w) - \sum_{(u,v)\in E} f(u,v)\right) = 0$$

Conservation constraint, assuming $v \neq s,t$.

Summing over $v \in S\setminus\{s\}$

$$\sum_{v\in S\setminus\{s\}}\left(\sum_{(v,w)\in E} f(v,w) - \sum_{(u,v)\in E} f(u,v)\right) = 0$$

Adding the definition of $|f|$

$$|f| \equiv \sum_{(s,w)\in E} f(s,w) - \sum_{(u,s)\in E} f(u,s) \quad +$$

$$\sum_{v\in S}\left(\sum_{(v,w)\in E} f(v,w) - \sum_{(u,v)\in E} f(u,v)\right) = |f|$$

$$\sum_{v\in S}\left(\sum_{(v,w)\in E} f(v,w) - \sum_{(u,v)\in E} f(u,v)\right) = |f|$$

$$\|$$

$$\sum_{(u,v)\in E(S,T)} f(u,v) - \sum_{(u,v)\in E(T,S)} f(u,v) = f(S,T)$$

מסקנה 1: אפשר להגדיר את ערך הזרימה גם על צומת הבור (היעד), $|f| = f(\{s\}, V\setminus\{s\}) = \boldsymbol{f(V\setminus\{t\},\{t\})}$.

למה 2 (ערך הזרימה לא גדול מהקיבול): אם f זרימה מ-s ל-t, ו-(S,T) הוא s-t cut של G, אז $|f| = f(S,T) \le c(S,T)$.

הוכחה: נגדיל את האדומות עד הקיבול המקסימלי, ואת הירוקות נקטין ל-0. נקבל:

$$\text{Proof: } f(S,T) = \sum_{(u,v)\in E(S,T)} f(u,v) - \sum_{(u,v)\in E(T,S)} f(u,v)$$
$$\le \sum_{(u,v)\in E(S,T)} c(u,v) = c(S,T)\,. \quad \blacksquare$$

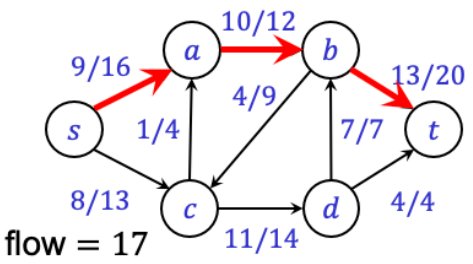
מסקנה 2: אם $|f| = f(S,T) = c(S,T)$ אז f **זרימה מקסימלית**, (S,T) **חתך מינימלי**. לכן על בסיס למה 2, הקיבול של החתך חוסם מלמעלה את הזרימה, אז בפרט **החתך המינימלי יהיה החסם העליון הכי נמוך לזרימה**. כלומר, החתך המינימלי (בעל הקיבול הכי קטן) חוסם את הזרימה המקסימלית (חוסם את הגודל של כל זרימה), שום זרימה לא יכולה לעבור אותו. נניח בשלילה שיש חתך קטן יותר (ואז נסתור את למה 2), ונניח בשלילה שיש זרימה יותר גדולה (וגם אז נסתור את למה 2). *בקרוב נראה את הכיוון ההפוך – נראה שזה לא רק חסם, אלא שיש זרימה שהערך שלה הוא בדיוק הערך של הקיבול המינימלי.*

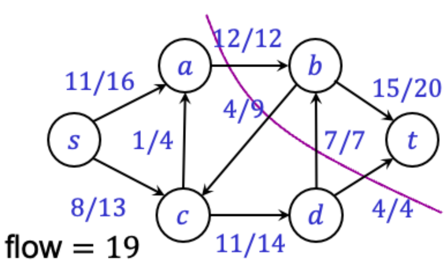
דחיפת זרימה

מסלול מכון (incrementing path): סוג מיוחד של מסלול תשתית (augmenting path).
אם P הוא s-t path (מסלול מהמקור ליעד) כך שלכל קשת במסלול מתקיים $f(e) < c(e)$ (**אין קשת רוויה במסלול**) אז אפשר לדחוף זרימה ב-f:

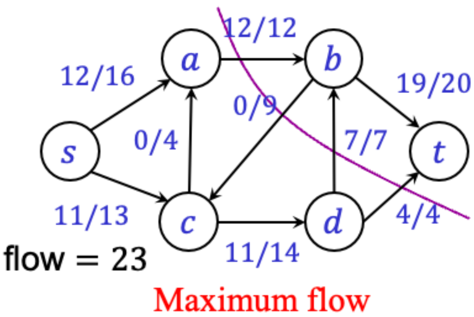
- נדחוף Δ יחידות זרימה לאורך המסלול P (בשאר הקשתות אין זרימה).
- אילוצי שימור הזרימה (חוק הצומת) יתקיימו לכל Δ. נגדיר את $c(e) - f(e)$ להיות **הקיבול השיורי של הקשת e**.
- אילוצי הקיבול (חוק הקשת) יתקיימו אם $\Delta \le \min_{e\in P} c(e) - f(e)$. כאשר $\min_{e\in P} c(e) - f(e)$ הוא **הרוחב של מסלול מכון**, הקיבול השיורי המינימלי של קשת במסלול.

אבחנה: בהינתן זרימה f, דחיפה של Δ יחידות זרימה במסלול מכון נותנת זרימה ⟺ Δ אינו גדול מהרוחב של המסלול. בסיום הפעולה, **הערך של הזרימה המתקבלת גדול ב-Δ מזה של f**.





Blocking flow



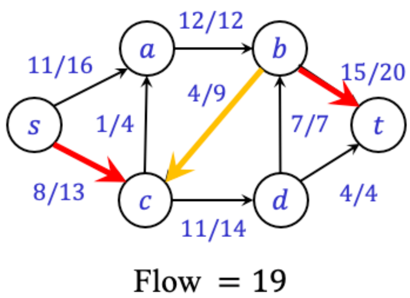
זרימה חסומה (blocking flow): אם אין מסלול מכוון בזה, הזרימה נקראת זרימה חסומה. כל המסלולים חסומים (יש קשת רוויה), ואי אפשר להגדיל את הזרימה באף מסלול.

אבחנה: נשים לב כי **זרימה חסומה היא לא בהכרח זרימה מקסימלית** – יש אפשרות אחרת להגדיל את הזרימה. נחשב את הזרימה בחתך: מה שיוצא $12+7+4=23$, מה שנכנס 4 ונקבל זרימה של 19. היא **לא שווה לקיבול של החתך** שהוא 23.
נוכל להשיג זרימה מקסימלית שבה הזרימה בחתך היא 23 והיא שווה לקיבול של החתך שהוא 23.

דחיפת זרימה במסלול תשתית:

מסלול תשתית/משפר (augmenting path): ברשת זרימה, מדובר במסלול מהמקור ליעד כאשר **מתעלמים מכיווני הקשתות**. נניח שאין לנו בקלט קשתות מקבילות (יש תמיד קשת רק בכיוון אחד). במסלול כזה יכולים להיות שני סוגי קשתות:

- קשתות קדימה (forward edges) – קשתות שהן בכיוון המסלול, שם **נגדיל את הזרימה**.
- קשתות אחורה (backward edges) – קשתות שהן נגד כיוון המסלול, שם **נקטין/נבטל את הזרימה**.
 נסמן $E^R=\{(v,u)\mid(u,v)\in E\}$.



נרצה לבצע פעולת שיפור על הזרימה f, במסלול תשתית P ופרמטר Δ.

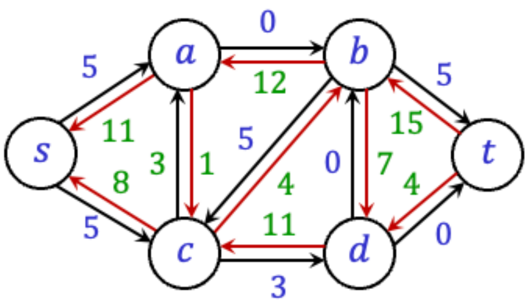
- בקשת קדמית – **נגדיל** את הזרימה ב-Δ, $f(e)+\Delta\le c(e)$ ולכן $\boldsymbol{\Delta\le c(e)-f(e)}$.
- בקשת אחורית – **נקטין** את הזרימה ב-Δ, $f(e)-\Delta\ge 0$ כלומר $\boldsymbol{\Delta\le f(e)}$.

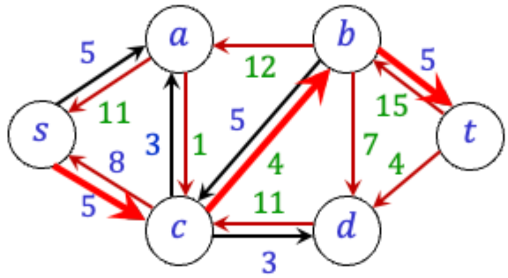
כעת נגדיר, עבור קשת **קדמית** הקיבול השיורי יהיה הכמות המקסימלית של זרימה שאפשר **להוסיף** לקשת (ההפרש בין הקיבול לזרימה), ועבור קשת **אחורית** הקיבול השיורי הוא הכמות המקסימלית של זרימה שאפשר **להוריד** מהזרימה הנוכחית בקשת (שווה לזרימה הנוכחית).

קיבול שיורי (residual capacity): קיבול שיורי של **קשת** במסלול תשתית P **יחסית לזרימה f** מוגדר על ידי:

$$c_f(e)=\begin{cases}c(e)-f(e) & e\in E\\ f(e) & e\in E^R\end{cases}$$

הרוחה (residual/bottleneck capacity) של מסלול תשתית P הוא $c_f(P)=\min_{e\in P}c_f(e)$.





רשת שיורית (residual network): הרשת הזו היא עבור זרימה מסוימת, לכן נסמנה G_f. ניקח את כל הקשתות והקשתות ההפוכות, נזרוק את כל אלה שיש להם קיבול שיורי 0 (מדובר בקשתות רוויות): אז עבור $f:E\to\mathbb{R}^+$ נקבל כי הרשת השיורית היא $G_f=(V,E_f,c_f,s,t)$ עבור:

$$E_f=\{e\mid e\in E\cup E^R,c_f(e)>0\}$$

כלומר, לכל קשת $e\in E$ מוסיפים קשת הפוכה, כאשר e מקבלת קיבול שיורי של מה שנשאר להזרים $c(e)-f(e)$, והקשת ההפוכה שלה מקבלת קיבול שיורי של מה שנשאר להוריד $f(e)$. לאחר מכן מוחקים את כל הקשתות בעלות קיבול שיורי 0.

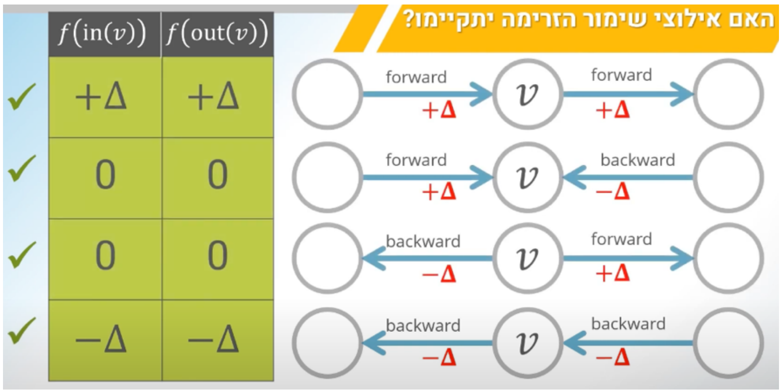
מסלול תשתית/משפר הוא בעצם מסלול מכוון מ-s ל-t **ברשת השיורית**. נשים לב כי מסלול מכוון הוא מקרה פרטי של מסלול תשתית. **כל מסלול מכוון בעל רוחב Δ הוא גם מסלול תשתית בעל רוחב Δ.**

פעולת השיפור (augmenting flow): בהינתן רשת שיורית $G=(V,E,c,s,t)$, זרימה f, ומסלול תשתית P. נרצה לדחוף זרימה ברשת השיורית כדי ליצור **augmented flow** באופן הבא:

$$f'(e)=\begin{cases}f(e)+c_f(P) & e\in P\\ f(e)-c_f(P) & e^R\in P\\ f(e) & else\end{cases}$$

כלומר, אנחנו מסתכלים על הקשתות במסלול המכוון שמצאנו ברשת השיורית, ומוסיפים או מורידים בהתאמה, לפי המקור של הקשת: קשת קדמית? נוסיף, קשת אחורית? נוריד.

אבחנה: בהינתן זרימה f, דחיפה של Δ יחידות זרימה במסלול תשתית P נותנת זרימה ⟺ Δ אינו גודל מהרוחב $\Delta_f(P)$ של המסלול.
הערך של הזרימה המתקבל גדול ב-Δ מזה של f: $|f'|=f+c_f(P)$.

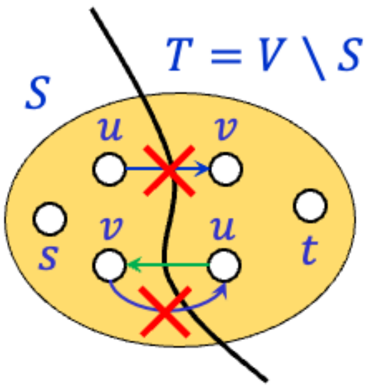


למה: f' היא זרימה חוקית, ומתקיים $|f'| = |f| + \Delta_f(P)$.

- לאחר הפעלת פעולת השיפור: נראה כי בכל 4 המקרים הכול תקין.
- אילוצי שימור הזרימה ימשיכו להתקיים (חוק הצומת).
- אילוצי הקיבול ימשיכו להתקיים (חוק הקשת) אם $\Delta \leq \Delta_f(P)$.
- ערך הזרימה יגדל ב-Δ.

טענה: תהי f זרימה **ברשת שיורית** $G = (V, E, c, s, t)$. שלושת התנאים הבאים שקולים:

1. f היא זרימה **מקסימלית** ב-G.
2. **אין מסלול תשתיתי** עבור f (שמאפשר לנו להגדיל את ערך הזרימה באמצעות פעולת השיפור).
3. יש חתך (S, T) של G כך שמתקיים $|\boldsymbol{f}| = \boldsymbol{c}(\boldsymbol{S}, \boldsymbol{T})$.



(2) ⇐ (3): אם אין מסלול משפר מ-s ל-t, אז נוכל למצוא את כל הצמתים הנגישים מ-s (עם DFS למשל), ונגדיר את קבוצת הצמתים הזו בתור הקבוצה S. קיבלנו חתך, כאשר S לא מכיל את t, ונגדיר את T בתור שאר הצמתים. זה חתך (S, T).

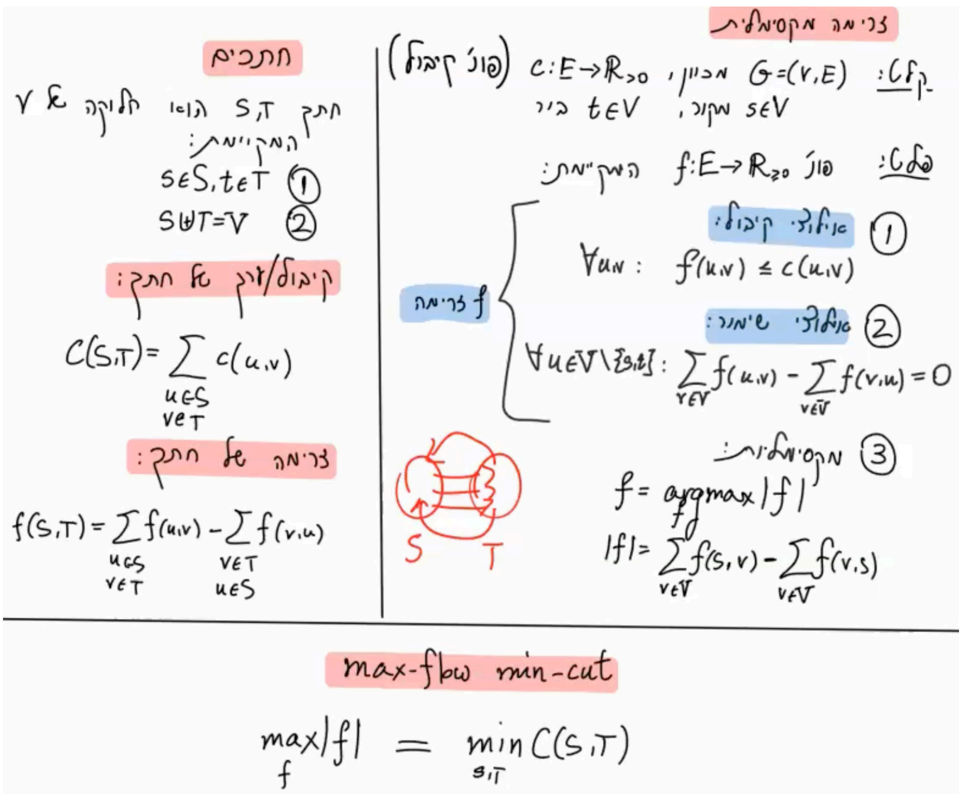
מתי הזרימה שווה לקיבול? רק אם בכל קשת חוצה זורם הקיבול (המקסימום האפשרי), ובכל קשת חוזרת זורם 0. נניח בשלילה שבקשת חוצה לא זורם מקסימום, אז הקשת הזאת הייתה ברשת השיורית ויכולנו להגיע לצומת שנמצא בקבוצה T.

לכן, אם הקשת החוצה $e \in E(S, T)$ אז $e \notin E_f$ ולכן $f(e) = c(e)$. אם הקשת החוזרת $e \in E(T, S)$ אז $e \notin E_f$ ולכן $f(e) = 0$. לכן נקבל כי ערך הזרימה $|f| = c(S, T)$.

אבחנה: לכל רשת זרימה יש זרימה מקסימלית.

מסקנה (MFMC): בכל רשת זרימה **ערך הזרימה המקסימלית שווה לקיבול של החתך המינימלי.**

זרימה 1 (תרגול 9)

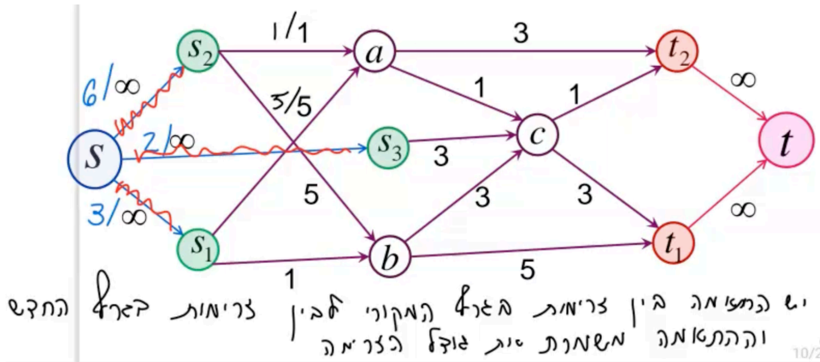


תרגיל 1 (מספר מקורות ובורות): נתונה רשת זרימה עם מספר מקורות ומספר בורות. הראו שניתן למצוא זרימה מקסימלית על ידי שימוש באלגוריתם לבעיה המקורית (עם מקור אחד ובור אחד).

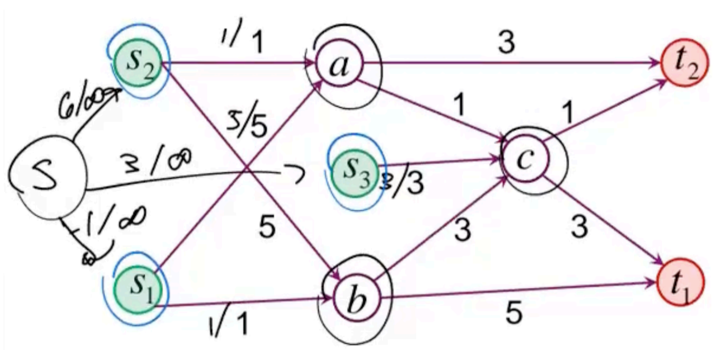
פתרון: נאחד את המקורות ואת הבורות למקור יחיד ובור יחיד. נוסיף צומת חדש s וממנו נמתח קשתות למקורות הקודמים, וצומת חדש t שאליו יכנסו קשתות מהבורות הקודמים. לקשתות אלו יהיו קיבול אינסופי. נחפש זרימה מקסימלית על הגרף החדש.

נכונות: יש התאמה בין זרימות בגרף המקורי לבין זרימות רגילות (מקור אחד ובור אחד) בגרף החדש. ההתאמה משמרת את גודל הזרימה. לכן בפרט, זרימה מקסימלית בגרף החדש מתאימה לזרימה מקסימלית בגרף המקורי.

⇐ נתונה לנו זרימה בגרף החדש. מה שיוצא מכל המקורות בגרף המקורי, זה בדיוק מה שנכנס מהמקור החדש.



⇐ נתונה זרימה בגרף המקורי. נעביר מ-s החדש בדיוק את הכמות שיוצאת מכל מקור בגרף המקורי.



תרגיל 2 (זרימה בשלמים): נתונה רשת זרימה מ-s ל-t עם קיבולים שלמים. הוכיחו שערך הזרימה המקסימלית שלם.

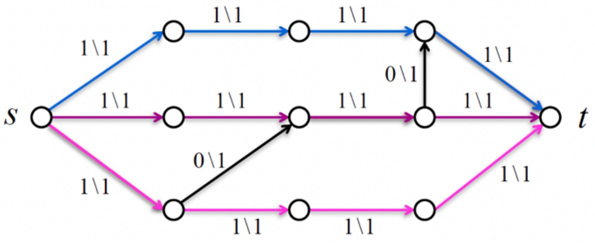
פתרון: כל הקיבולים ברשת שלמים, זה אומר שבפרט ערך הזרימה של כל חתך ברשת הוא שלם (כי ערך זרימה של חתך הוא סכום של קיבולים), בפרט הערך של החתך המינימלי הוא שלם. לפי MFMC ערך הזרימה המקסימלית שלם.

***אפשר להוכיח משהו יותר חזק**, שבפרט קיימת זרימה מקסימלית שעל כל קשת בה יש ערך שלם (בתרגול הבא).*

תרגיל 3 (מסלולים זרים בקשתות): נתון גרף מכוון וצמתים $s, t \in V$. תארו אלגוריתם אשר מוצא את המספר המקסימלי של מסלולים זרים בקשתות מ-s ל-t.

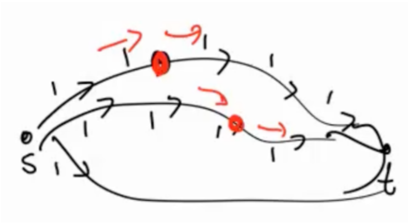
פתרון: מה לא עובד? אלגוריתם חמדן שכל פעם מחפש מסלול, ואז מוריד אותו ומחפש מסלול אחר, לא נוכל להתחרט על קשתות שלקחנו. עלינו לקבל מסלולים זרים לחלוטין בקשתות. מה כן עובד?

- ניתן לכל קשת קיבול 1. רשת כזאת נקראת רשת 0-1. מהטענה שנוכיח בתרגול הבא: קיימת זרימה מקסימלית שלכל קשת בה יש ערך שלם – יזרום או 0 או 1 (יודעים גם למצוא אותה).
- נמצא זרימה מקסימלית מ-s ל-t, ונניח שכל קשת תזרים 0 או 1, או שנבחר קשת או שלא.
- **נקבל כי הגודל של הזרימה המקסימלית בדיוק שווה למספר המסלולים הזרים המקסימליים**. על כל מסלול זר אנחנו מזרימים 1, יש 3 מסלולים וכל אחד מהם תורם 1 לזרימה.

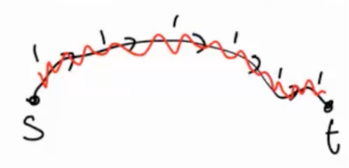


נכונות: נסמן ב-m את גודל הזרימה המקסימלי, וב-k את המספר המקסימלי של מסלולים זרים בקשתות. נוכיח כי $k = m$:

- $m \geq k$: יש לנו k מסלולים זרים בקשתות מ-s ל-t. נתאר זרימה על הגרף שהערך שלה הוא לפחות k. נזרים על כל מסלול נתון 1 בכל קשת. זה ברור, **שערך הזרימה הוא k** זה ברור, מ-s יוצאות בדיוק k קשתות שבכל אחת מזרימים בה 1. נראה שהזרימה חוקית:



- **קיבול** – בכל קשת אנו מזרימים 1, והקיבול בכל קשת הוא 1. המסלולים זרים בקשתות, כל קשת מופיעה רק במסלול אחד, ולכן על כל קשת הזרמנו לכל היותר 1.
- **שימור** – נסתכל על קודקוד באחד המסלולים, אנחנו מכניסים 1 ומוציאים מיד 1. קודקודים יכולים להופיע בכמה מסלולים, אבל גם במסלול השני מה שנכנס בו זה מה שיוצא. אם נסכום על פני כל המסלולים את מה שנכנס פחות מה שיוצא נקבל 0, בכל מסלול בנפרד זה מתקיים ולכן גם בסכום. זו תכונה שמקבלים ברגע שמזרימים במסלולים ערך קבוע לאורך המסלול (אפשר גם להזרים בכל מסלול ערך אחר).



- $m \le k$: יש לנו זרימה בגודל m. נראה שיש לפחות k מסלולים זרים. נוכיח באינדוקציה על m.
  - בסיס: $m = 0$, ברור שיש 0 מסלולים זרים.
  - נניח שהטענה נכונה עבור $m - 1$ ונוכיח עבור m. ניקח זרימה מקסימלית בגודל m (בכל קשת זורם 0 או 1). נסיר את הקשתות עם זרימה 0 (סתם מפריעות, לא מעניין). נחפש מסלול מ-s ל-t, בפרט כל הקשתות במסלול מזרימות 1. **נוריד את כל המסלול**, ונקבל רשת זרימה חדשה עם זרימה בערך $m - 1$. מה"א מתקיים יש $m - 1$ מסלולים זרים לפחות בגרף החדש. לכן, בגרף המקורי (הכולל את המסלול שהסרנו) יש m מסלולים זרים.

<u>תרגיל 4 (קשתות מנתקות):</u> נתון גרף מכוון וצמתים $s, t \in V$. תארו אלגוריתם אשר מוצא את מספר הקשתות המינימלי שיש להסיר מהגרף על מנת שלא יהיה מסלול מ-s ל-t.

<u>פתרון:</u> כל חתך s-t נותן לנו חסם עליון לערך שאנחנו מחפשים – מספר הקשתות המינימלי שיש להסיר מהגרף כדי לנתק את המסלול בין s ל-t. אנחנו רוצים למצוא את **החתך המינימלי**, ונוכל לעשות זאת **באמצעות הזרימה המקסימלית**, לפי **MFMC**. לא נתונה לנו רשת זרימה, לכן ניעזר במה שעשינו קודם **וניתן לכל קשת קיבול 1**.

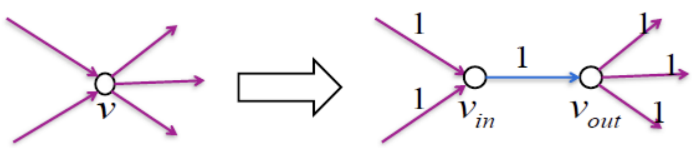
נכונות: נסמן ב-k את גודל החתך המינימלי, וב-j את מספר הקשתות המנתקות המינימלי. נרצה להראות $k = j$:

- $j \le k$: בהינתן חתך s-t בגודל k יש k קשתות שחוצות את החתך, ואם נסיר אותן אז ננתק את s מ-t. מצאנו מספר קשתות $j = k$ ולכן המינימלי בהכרח קטן או שווה לכמות זו.
- $j \ge k$: בהינתן קבוצה $|A| = j$ של קשתות מנתקות, נרצה להראות שיש חתך בגודל j ואז המינימלי יהיה בגודל קטן או שווה לו. נגדיר גרף חדש $G' = (V, E \backslash A)$, שבו s ו-t מנותקים. לכן, נגדיר את S: כל מי שנגיש מ-s ב-G', T: כל שאר הצמתים. קיבלנו חתך (S, T) ונרצה להראות שהגודל שלו הוא j (לכל היותר).
  - בגרף G' אין אף קשת שחוצה את החתך ולכן $C_{G'}(S, T) = 0$.
  - בגרף המקורי G מתקיים: לכן $C_G(S, T) \le C_{G'}(S, T) + |A| = j$.

<u>סיכום ביניים לרשת 0-1, יש שוויון בין:</u>

- ערך הזרימה המקסימלית.
- גודל החתך המינימלי.
- המספר המקסימלי של מסלולים זרים בקשתות.
- מספר הקשתות המינימלי שיש להסיר בכדי לנתק את הגרף.

<u>תרגיל 5 (מסלולים זרים בקודקודים):</u> נתון גרף מכוון וצמתים $s, t \in V$. תארו אלגוריתם אשר מוצא את המספר המקסימלי של מסלולים זרים בקודקודים מ-s ל-t.



<u>פתרון:</u> באופן כללי, מסלולים זרים בקודקודים בהכרח זרים בקשתות, אך לא בהכרח להפך. נמיר את הגרף: כל צומת בגרף המקורי v נפצל לשני צמתים v_{in}, v_{out}, נכניס ונוציא את הקשתות באופן מתאים. ניעזר ברשת 0-1 עם קיבול 1 על כל הקשתות. בגרף החדש שהתקבל G', **מסלולים זרים בקשתות אמ"מ הם זרים בקודקודים**. בגרף החדש נמצא זרימה מקסימלית, וזה מתאים למספר המסלולים הזרים בקודקודים.

<u>נכונות:</u>

- נראה התאמה בין מסלולים זרים בקודקודים ב-G לבין G': אם נסתכל על מסלול ב-G שעובר ב-v, המסלול שמתאים לו הוא עובר דרך v_{in} ויוצא דרך v_{out}.
- נראה התאמה בין מסלולים זרים בקודקודים ב-G' למסלולים זרים בקשתות ב-G':
  - ⇐ מסלולים זרים בקודקודים בהכרח זרים בקשתות.
  - ⇒ תהי קבוצת מסלולים זרים בקשתות, נניח בשלילה שהם לא זרים בקודקודים, יש להם קודקוד משותף שהוא מהצורה v_{in} או v_{out}. אם יש לנו שני מסלולים שעוברים דרך v_{in}, אחריו חייב לבוא v_{out} ואז הקשת (v_{in}, v_{out}) משותפת, בסתירה. (**כי בעצם אנחנו כופים על כל קודקוד v להפוך לזוג צמתים מחוברים בקשת**).

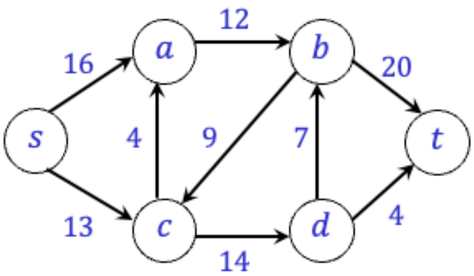
שיטת פורד-פלקרסון (FF)

<u>הרעיון:</u> נתחיל עם זרימה חוקית כלשהי f בגרף G: למשל $f = 0$. נבצע איטרציות שיפור:

- נבנה את **הרשת השיורית G_f** (אפשר לבנות אותה בזמן $O(|E|)$, עוברים כל הקשתות ורושמים את הקיבולים).
- נחפש **מסלול שיפור P (מסלול מכוון מ-s ל-t)** ברשת זו (אפשר למצוא באמצעות BFS/DFS בזמן $O(|E|)$).
- אם לא מצאנו כזה – הזרימה מקסימלית.
- אם מצאנו מסלול כזה P, אז ניקח את הקיבולת השיורית שלו $c_f(P)$ ונבצע את **פעולת השיפור** (חסם עליון על אורך המסלול זה מספר הצמתים בגרף, אז העלות היא $O(|V|)$).

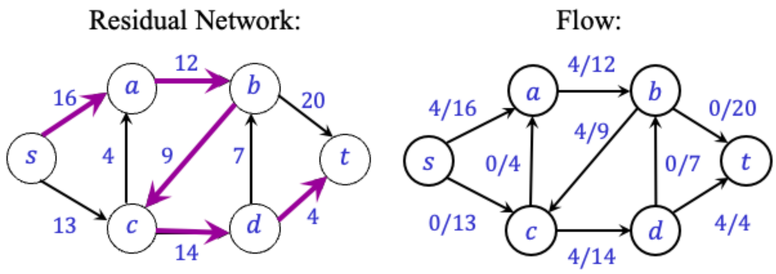
<u>טענה:</u> אם כל הקיבולים על הקשתות הם **שלמים**, ומתקיים $c: E \to \mathbb{Z}^+$, והזרימה ההתחלתית f היא **שלמה** ($f = 0$). כל הזרימות והקיבולים השיוריים בדרך יהיו שלמים, ובכל איטרציה $c_f(P) \geq 1$ וזה חייב להסתיים. לכן:
(1) אם כל הקיבולים הם **שלמים**, אז יש **זרימה מקסימלית שלמה f^***.
(2) פעולת FF, המתחילה מ-$f = 0$, עם כל בחירה של מסלולים משפרים, מוצאת **זרימה שלמה מקסימלית** לאחר לכל היותר $|f^*|$ שיפורים, וזמן הריצה הכולל הוא $O(|E| \cdot |f^*|)$.

<u>דוגמה:</u>

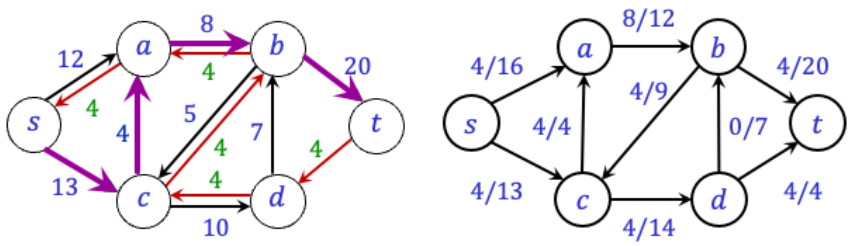


נתחיל עם זרימה 0, לכן ביחס אליה הרשת המקורית היא גם הרשת השיורית. הסיבה לכך היא שהקיבול השיורי של כל קשת אחורה הוא 0, ולכן מסלול תשתית בעל רוחב חיובי, ישתמש רק בקשתות קדימה.

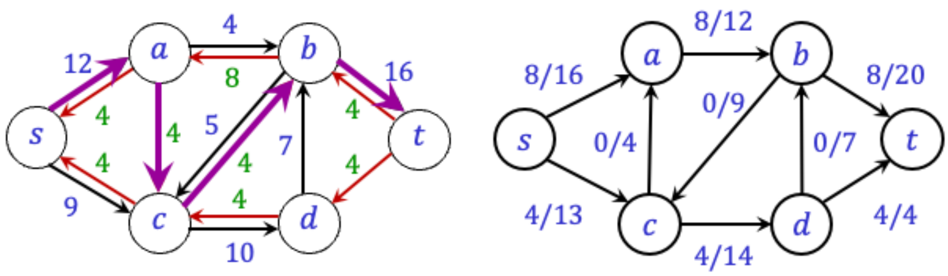
כעת, נסתכל על מסלול משפר, כאשר הקיבולת השיורית של המסלול (הרוחב שלו) $c_f(P) = 4$, המינימלי מבין כל הקיבולים השיוריים על המסלול. לכן, נשפר ב-4 יחידות זרימה: על כל קשת בכיוון המסלול נוסיף 4.



נמצא מסלול משפר נוסף, גם שם הרוחב הוא 4, לכן נוסיף 4 יחידות זרימה (כל הקשתות בכיוון המסלול).
*אינטואיציה: על קשתות שהן בכיוון המקורי (E), **הקיבולים ברשת השיורית הם מה שנשאר להזרים** $c(e) - f(e)$, בכיוון ההפוך נגדיר בדיוק את $f(e)$, מה שזורם כרגע. זאת על מנת שנדע כמה אפשר לשפר, **רוחב/קיבול המסלול יהיה המינימלי מבין כל הערכים הללו על גבי המסלול**.*



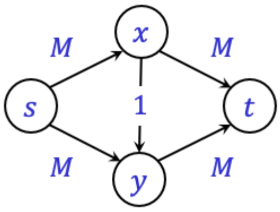
נמצא שוב מסלול משפר. הפעם, יש לנו גם קשתות אחוריות. ניזכר כי עבור **קשת קדימה**, הקיבול השיורי הוא הכמות המקסימלית שאפשר **להוסיף**, בעוד עבור **קשת אחורית** הקיבול השיורי הוא הכמות המקסימלית שאפשר **להוריד**. נקבל כי עבור הקשתות הקדמיות הקיבולים השיוריים הם 12 ו-16, עבור האחוריות הקיבולים השיוריים הם 4 ו-4. לכן, $c_f(P) = 4$, נוסיף ונוריד בהתאם.

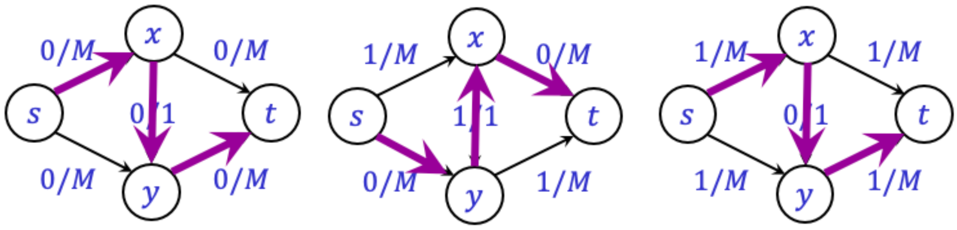


<u>דוגמה איטית:</u>

ברשת הנ"ל הזרימה המקסימלית היא 2M. אם נבצע בחירה לא יעילה של מסלולים משפרים, FF יבצע 2M שיפורים.

כל פעם נעבור דרך מסלול שה-bottleneck (רוחב המסלול) שלו הוא 1. נבחר קודם את s-x-y-t, ואז את s-y-x-t ונקבל זרימה של 0 על האמצעית (הזרמנו ואז ביטלנו), ונקבל במסלול העליון 1 ובתחתון 1 אחרי 2 שיפורים. כך אחרי 2M איטרציות נקבל זרימה של M על כל אחד מהמסלולים העליון והתחתון.



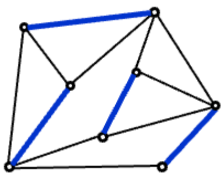


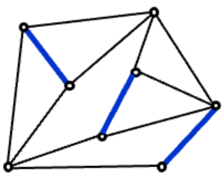
לכן, **החסם $O(|E||f^*|)$ הוא הדוק.**

<u>דוגמה לריצה אינסופית (Zwick, 1995):</u> אם יש אפילו רק קיבול אחד שהוא אי-רציונלי, האלגוריתם יכול לא להסתיים.

זיווגים בגרפים דו-צדדיים

<u>בעיית זיווג מקסימום:</u> נתון גרף $G = (V, E)$ לא מכוון. **קבוצה $M \subseteq E$ היא זיווג/שידוך** של הגרף ⇔ אין שתי קשתות של M שנוגעות זו בזו, **הקשתות זרות בקודקודים**. כלומר $\{a, b\} \neq \{c, d\} \in M \rightarrow$ a, b, c, d *are distinct*. המטרה היא למצוא **זיווג מקסימום.**

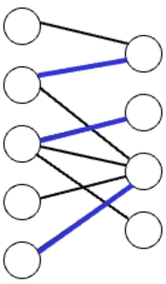

maximum matching
perfect matching

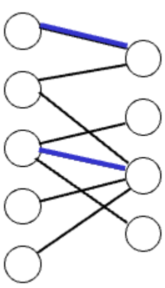

maximal but not
maximum matching

<u>זיווג מקסימום:</u> זיווג עם **המספר הכי גדול של קשתות מבין כל הזיווגים.** זיווג **מקסימלי הוא כזה שלא ניתן להוסיף לו עוד קשתות,** אך זה לא אומר שהוא מקסימום (הוא לא בעל הכי הרבה קשתות).

<u>זיווג מושלם:</u> זיווג מקסימלי שנוגע **בכל הצמתים בגרף.** כלומר, לכל צומת יש קשת בזיווג שנוגעת בו.

maximum
but not
perfect
matching





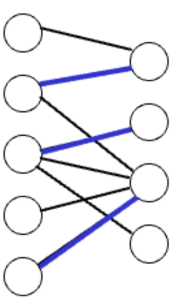
maximal
but not
maximum
matching

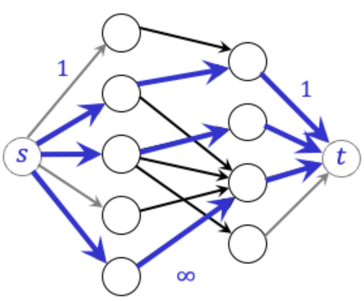
<u>בגרף דו-צדדי:</u> גרף $G = (V, E)$ הוא דו-צדדי, אם קיימות קבוצות $L, R \subseteq V$ כאשר $L \cap R = \emptyset$, $L \cup R = V$ ו-$E \subseteq L \times R$. גם כאן, יכול להיות זיווג שהוא לא זיווג מקסימום (לא מכיל את המספר המקסימלי של קשתות, יש זיווג עם 3 קשתות). וזיווג מקסימום שהוא לא בהכרח מושלם.

עבור גרפים דו-צדדיים, הבעיה יכולה לעבור **רדוקציה לבעיית זרימה מקסימלית max flow.**

<u>האלגוריתם:</u>

- בהינתן גרף דו-צדדי $G = (L, R, E)$ נבנה רשת זרימה $G' = (V', E', 1, s, t)$ כאשר כל הקיבולים הם 1, נוסיף צומת מקור s וצומת בור t שיחוברו ל-L ול-R בהתאמה.
- נמצא זרימה שלמה f^* באמצעות FF.
- נחזיר את M, קבוצת כל הקשתות המקוריות שיש להן זרימה של 1 (יכולה להיות רק זרימה של 0 או 1, ניקח את אלו עם זרימה 1).



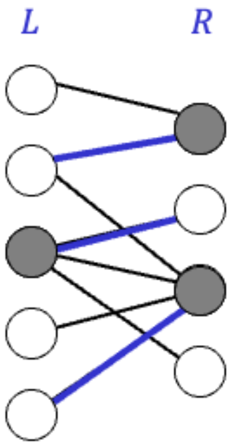


<u>סיבוכיות:</u> מספר הקשתות ברשת הזרימה החדשה הוא $|V| + |E|$ ולכן זמן הריצה הוא $O(|E||V|) = O\big((|E| + |V|)|f^*|\big)$ כיוון שמתקיים $|V| + |E| = O(|E|)$ כי מספר הצמתים קטן ממספר הקשתות, ו-$f^* = O(|V|)$ כי סך הזרימה שיכול להיכנס ל-t הוא 1 לכל צומת.

<u>משפט:</u> בגרף דו-צדדי גודל הזרימה המקסימלית = גודל הזיווג המקסימלי.

<u>הוכחה:</u> מתקיים שיש **התאמה אחד לאחד בין matching לבין integral flow**. ברור כי כל matching אפשר להרחיב לזרימה ברשת. בכיוון השני, מכל זרימה ברשת אפשר לחלץ matching (הקשתות המרכזיות): לא יכול להיות שיש שתי קשתות בחולות שנכנסות לאותו צומת, וכיוון שהזרימה בהן שלמה, היה צריך לצאת מהצומת 2, אבל הקיבול הוא 1 וזה לא יכול לקרות.

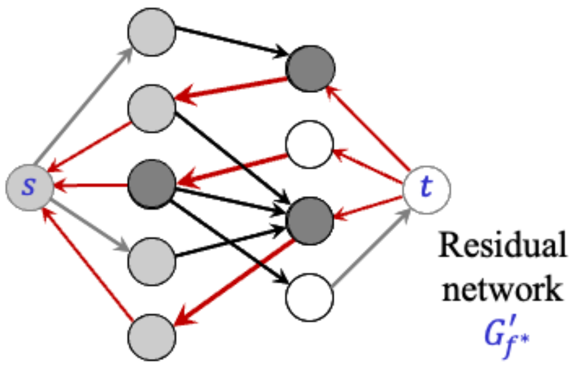
<u>משפט החתונה של הול:</u> נתון גרף דו-צדדי $G=(L,R,E)$, עבור $X\subseteq L$, נסמן $N(X)$ את קבוצת השכנים של X ב-R. קיים זיווג מכל הצמתים ב-L $\Leftrightarrow$ $|N(X)|\geq|X|$.



<u>כיסוי צמתים מינימלי:</u> נתון גרף דו-צדדי $G=(L,R,E)$, קבוצה $C\subseteq V=L\cup R$ היא vertex cover (כיסוי צמתים) $\Leftrightarrow$ היא נוגעת בכל הקשתות, כלומר לפחות קצה אחד של כל קשת נמצא ב-C: לכל $\{u,v\}\in E$ מתקיים $\{u,v\}\cap C\neq\emptyset$. המטרה היא למצוא כיסוי צמתים מינימלי.

<u>האלגוריתם:</u>

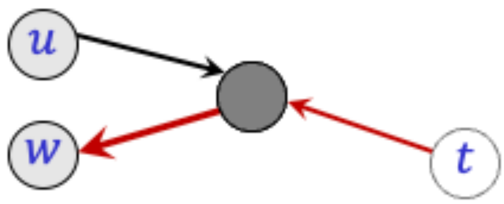
- נבנה רשת זרימה כמו קודם מהגרף הדו-צדדי. נמצא זרימה מקסימלית f^* וזיווג מקסימום M.
- עבור הזרימה הזו f^* נבנה את הרשת השיורית G_{f^*}. כעת נרצה לחלץ מכאן vertex cover.
- נעבור על כל הקשתות בזיווג $(u,v)\in M$, ונוסיף את u או את v ל-C: אם v נגיש מ-s ברשת השיורית נוסיף אותו, אחרת נוסיף את u.
  - ניתן לראות בשרטוט כי הצמתים באפור בהיר הם הצמתים שנגישים מ-s, והצמתים שבאפור כהה הם אלו שנבחרו לכיסוי: כאשר u בצד שמאל, v בצד ימין.
- מספר הצמתים שהוספנו (גודל הכיסוי שיצרנו) זה **בדיוק גודל הזיווג**, $|C|=|M|$: כי הוספנו צומת אחד מכל קשת בזיווג. **נותר לנו להראות שזה מכסה את כל הקשתות בגרף, לא רק את הזיווג.**



<u>משפט:</u> בגרף דו-צדדי גודל כיסוי הצמתים המינימלי = גודל הזיווג המקסימלי.

<u>הוכחה:</u>

- $\leq$: ברור שגודל הזיווג המקסימלי קטן או שווה לגודל כיסוי הצמתים המינימלי: בשביל לכסות את כל הקשתות בגרף, בפרט צריך לכסות את הקשתות של ה-matching (לקחת קצה כלשהו של הקשתות מה-matching). לכן, בהינתן זיווג, ניקח את כל הצדדים השמאליים של הקשתות בזיווג, יהיה צומת לכל קשת ולא יהיה חיתוך בין הצמתים (כי הקשתות זרות בזיווג), קיבלנו חסם תחתון לגודל כיסוי הצמתים.
- $\geq$: נמשיך מהנקודה שבה יצרנו את הכיסוי C וגודלו הוא כגודל M, הזיווג שנעזרנו בו. כל קשת של M מכוסה, לכן נסתכל על $(u,v)\in E\backslash M$.
  - <u>מקרה 1:</u> **u, ולכן גם v, נגישים מ-s ברשת השיורית.** t (הבור בצד ימין) לא נגיש מ-s ברשת השיורית. מדובר בזרימת מקסימום ולכן אין מסלול משפר. לכן יש לנו קשת הפוכה מ-t ל-v ברשת השיורית (אין קשת ל-t שניתן להזרים אליה עוד, הזרימה היא 1). לכן, יש גם זרימה מ-w ל-v של 1 (לפי חוק השימור/הצומת, צריכה גם להיכנס זרימה של 1 ל-v כדי שתצא 1 מ-v), ועל כן ברשת השיורית תהיה קשת $(w,v)\in M$, כי כל הקשתות שיש עליהן זרימה בגרף המקורי נמצאות בזיווג, כאשר $v\in C$. לפי האלגוריתם, עברנו על הקשת בזיווג (w,v) **והוספנו את v כי הוא נגיש מ-s. כיסינו את הקשת (u,v).**



  - מקרה 2: **u לא נגיש מ-s ברשת השיורית.** נסתכל על הצד השמאלי, אם u לא נגיש זה אומר יש קשת הפוכה ברשת השיורית (u,s). לכן, חייבת להיות קשת $(u,w)\in M$ בזיווג כאשר $u\in C$. לפי האלגוריתם, עברנו על הקשת בזיווג (u,w), **והוספנו את u כי הצד הימני לא נגיש מ-s. כיסינו את הקשת (u,v).**

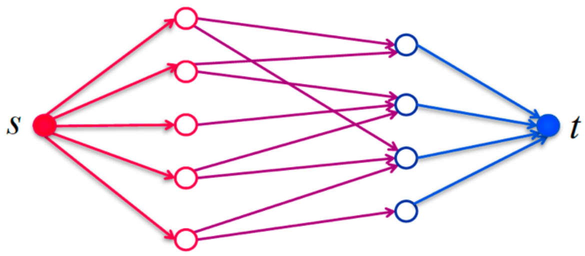
זרימה 2 (תרגול 10)

<u>תרגיל 1 (זרימה בשלמים):</u> נתונה רשת זרימה מ-s ל-t עם קיבולים שלמים. הוכיחו שהשיטה של FF מוצאת זרימה מקסימלית f כך שלכל $u, v \in V$ הערך $f(u, v)$ הינו שלם.

<u>פתרון:</u> נוכיח את הטענה באינדוקציה על מספר הזרימות שהאלגוריתם חיבר עד כה. נרצה להראות שבכל שלב באינדוקציה פונקציית הזרימה מזרימה ערך שלם על כל קשת:

- בסיס – מתחילים מזרימת 0, הכול שלם.
- צעד האינדוקציה – עד כה מצאנו זרימה f שבה בכל קשת עוברת זרימה שלמה. באיטרציה הבאה אנחנו מחפשים מסלול P ברשת השיורית של f, ומשפרים אותו לפי ה-bottleneck שלו – **מהנחת האינדוקציה כל הקיבולים השיוריים הם שלמים** ולכן גם המינימלי שלם. כאשר נוסיף/נחסיר אותו מערכים שלמים, **הזרימה תישאר שלמה**.

<u>תרגיל 2 (סוסים ורוכבים):</u> יש 100 סוסים ו-100 רוכבים. כל רוכב מכין רשימה של סוסים שהוא מוכן לרכב עליהם. מצא אלגוריתם שימצא את המספר המקסימלי של סוסים ורוכבים שיוכלו לדהור ביחד. נרצה לצמד כמה שיותר זוגות.



<u>פתרון:</u> ניעזר בגרף דו-צדדי, ונרצה למצוא זיווג מקסימום בגרף. לצורך כך נשתמש בזרימה. נוסיף צומת עזר s שממנו נמתח קשתות לכל הצמתים ב-L, ונוסיף צומת בור t שנמתח קשתות אליו מ-R. נשייך לכל הקשתות את הקיבול 1, ונקבל **רשת 0-1 בה נחפש זרימה מקסימלית**. נריץ אלגוריתם למציאת זרימה מקסימלית.

<u>נכונות:</u> נוכיח שגודל הזרימה המקסימלית $|f|$ שווה לגודל זיווג המקסימום m:

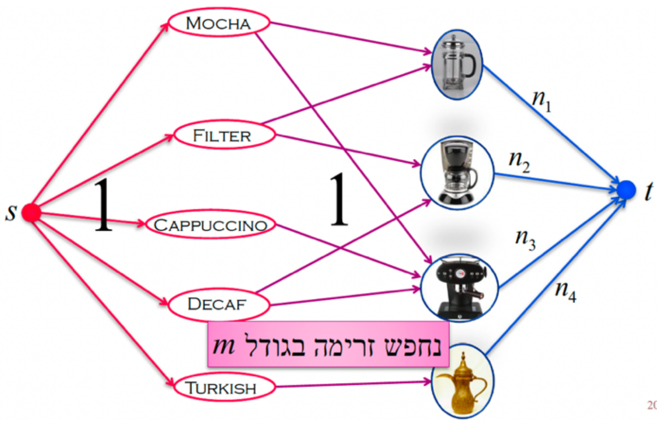
- $|f| \geq m$: בהינתן זיווג, נמצא זרימה מתאימה באותו גודל (m) באופן הבא: עבור כל קשת (u, w) בזיווג, נעביר זרם 1 דרך הקשתות $(s, u), (u, w), (w, t)$. כלומר לכל קשת בזיווג יצרנו מסלול זרימה, כאשר כמות המסלולים כגודל הזיווג.
- $|f| \leq m$: בהינתן זרימה מקסימלית, נוכל לקחת רק את הקשתות מ-U ל-W (מצד שמאל של הגרף הדו-צדדי לצד ימין) שהן רוויות (לוקחות חלק בזרימה), ולקבל את הזיווג המתאים.

<u>סיבוכיות:</u> בניית הרשת לוקחת זמן לינארי, ולכן זמן הריצה תלוי רק באלגוריתם למציאת זרימה מקסימלית. בשיעור נראה שבמקרה כזה האלגוריתם של דיניץ רץ בזמן $O(|E|\sqrt{|V|})$.

<u>הערה, נגדיר רשת 0-1 מטיפוס II:</u>

- *כל הקיבולים 1.*
- *לכל קודקוד יש דרגת כניסה/יציאה לכל היותר 1.*

<u>תרגיל 3 (מכונות ומשימות):</u> נתונות n מכונות ו-m משימות שצריכות להתבצע על ידי המכונות. מכונה i יכולה לבצע n_i משימות. בנוסף, עבור כל משימה נתונה רשימה של מכונות שיודעות לבצע אותה. תארו אלגוריתם אשר בוחר לכל משימה את המכונה שתבצע אותה (תחת ההגבלות הנ"ל), או מודיע שלא קיימת השמה כזו.



<u>פתרון:</u> באופן דומה נבנה גרף דו-צדדי, משימות בשמאל ומכונות בימין. כל מסלול שזורם עליו משהו, חייב להתחיל מ-s ל-cappuccino, ומשם רק לאחת המכונות שיכולות לבצע אותו, ואנחנו חייבים להזרים 1 על ה-n_3 משימות שהמכונה 3 יכולה לבצע.
נרצה למצוא זרימה שהגודל שלה בדיוק כמספר המשימות, **כלומר זרימה בגודל m**. הזרימה המקסימלית חייבת להיות לכל היותר m, אם נסתכל על החתך באמצע, הגודל שלו הוא m (וכל חתך מהווה חסם עליון על הזרימה ולכן היא לא יכולה להיות יותר מ-m).

<u>סיבוכיות:</u> נסמן את סך אורכי הרשימות של המשימות ב-M, מה שמיוצג בגרף על ידי כל הקשתות שחוצות באמצע. לא מדובר ברשת 0-1, אז אי אפשר להשתמש בניתוח המוצלח של דיניץ.

- ניקח את הניתוח הרגיל: יש לנו $m + n + 2$ קודקודים, ו-$m + n + M$ קשתות. נקבל $O(|E||V|^2) = O\big((m + n + m)(m + n)^2\big)$.
- החסם שנקבל מ-FF עדיף: $O(|E||f^*|) = O\big((m + n + M)m\big)$. (אפשר להראות חסם דומה על הרצת דיניץ).

<u>תרגיל 4 (k-קשירות בקשתות):</u> גרף מכוון נקרא k-קשיר בקשתות אם בין כל זוג קודקודים שלו יש לפחות k מסלולים זרים בקשתות (בכל כיוון). נתון גרף מכוון $G=(V,E)$ וקבוע k, תארו אלגוריתם אשר בודק האם הוא k-קשיר בקשתות.

<u>פתרון:</u> בתרגיל 3 מתרגול 9, ראינו כי אפשר למצוא מסלולים זרים בקשתות באמצעות זרימה מקסימלית. עבור כל זוג קודקודים $u,v\in V$, נבדוק האם יש ביניהם k מסלולים זרים בקשתות (בשני הכיוונים). ראינו בתרגול הקודם איך לעשות את זה באמצעות זרימה ברשתות 0-1. הסיבוכיות: $O\left(|V|^2|E|\min\left\{|V|^{\frac{2}{3}},|E|^{\frac{1}{2}}\right\}\right)$.

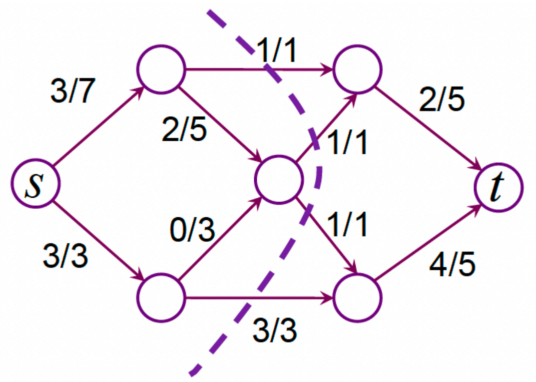
<u>פתרון משופר:</u> אין צורך למצוא זרימה מקסימלית. צריך רק לבדוק האם קיימת זרימה בגודל k (יכול להיות שהמקסימלית גדולה ממש מ-k). נוכל להריץ k איטרציות של FF. **כל איטרציה מזרימה לפחות 1 על מסלול כלשהו כי הקיבולים שלמים**, לכן הזרימה תהיה k או פחות. נקבל סיבוכיות: $O(k|V|^2|E|)$.

<u>פתרון משופר מאוד:</u> נבחר קודקוד יחיד כלשהו $v\in V$ ונבדוק שיש k מסלולים זרים בינו לבין כל אחד מהקודקודים האחרים (בשני הכיוונים). יהיו לנו $|V|$ בדיקות לעשות מהעוגן v. זה מספיק – אם יש ממנו k מסלולים לכולם, אז בפרט בין כל זוג יש (טרנזיטיביות).

<u>נכונות:</u> **נוכיח באמצעות זרימה**. הבדיקות שהאלגוריתם מבצע נכשלות ⇔ הגרף אינו k-קשיר.

- ⇐ אם אחת מהבדיקות נכשלה, ברור שהגרף אינו k-קשיר.
- ⇒ אם כל הבדיקות עברו, אז לכל $u,w\in V$: **יש זרימה בגודל k (שקול לכך שיש k מסלולים זרים בקשתות) מ-u ל-v, יש זרימה בגודל k מ-v ל-w, ולכן יש זרימה (טרנזיטיביות) בגודל k מ-u ל-w**. נסתכל על החתך המינימלי s-t כך ש-u נמצא ב-S ו-w נמצא ב-T. נניח בשלילה שאין זרימה בגודל k מ-u ל-w ונגיע לסתירה. מתקיים $C(S,T)<k$ כי אין זרימה בגודל k:
  - אם $v\in S$: כל זרימה קטנה מגודל החתך, שהוא קטן מ-k. מצאנו חתך מגודל קטן מ-k בין v ל-w ולכן לא ייתכן שיש זרימה בגודל k (גדול מכך) מ-v ל-w.
  - אם $v\in T$: סימטרי הפוך. אפשר להגיד שמצאנו חתך בין u ל-v שהגודל שלו קטן מ-k, ואז לא ייתכן שיש זרימה בגודל k מ-u ל-v.

<u>תרגיל 5 (מציאת חתך מינימלי):</u> נתונה רשת זרימה $G=(V,E)$ וזרימה מקסימלית עבורה f. תארו אלגוריתם למציאת חתך מינימלי של הרשת.



<u>אבחנה:</u> אם f זרימה מקסימלית, אז ב-G_f אין אף מסלול בין s ל-t (אם היה כזה, נוכל לשפר את הזרימה המקורית).

<u>פתרון:</u> נבנה את הרשת השיורית ביחס לזרימה f. נריץ BFS מהמקור, נגדיר את S להיות כל מי שנגיש מ-s ב-G_f, ו-T: כל שאר הקודקודים. אנחנו יודעים כי $s\in S$, $t\in T$. קיבלנו חתך מינימלי.

<u>סיבוכיות:</u> $O(|V|+|E|)$.

<u>נכונות:</u> החתך s-t שהאלגוריתם מוצא הוא מינימלי:

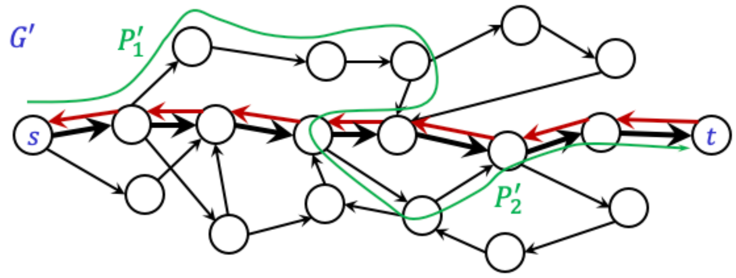
- מתקיים $|f|=f(S,T)=\sum_{(u,v)\in E(S,T)}f(u,v)-\sum_{(v,u)\in E(T,S)}f(v,u)$.
  - האם יכולה להיות קשת בגרף מ-T ל-S? כלומר קשת (v,u) עבור $u\in S$, $v\in T$ שזורמת עליה זרימה חיובית? זה אומר שברשת השיורית G_f קיימת הקשת (u,v) עם אותו ערך זרימה. זה לא ייתכן, כי אז זה אומר ש-v גם נגיש מ-s למרות ש-v נמצא ב-T. לכן, **כל הצד הימני הוא 0**.
  - האם יכולה להיות קשת בגרף מ-S ל-T שזורמת עליה זרימה שקטנה מהקיבול? נניח קשת (u,v) שמתקיים עבורה $f(u,v)<c(u,v)$? זה אומר שגם ברשת השיורית תופיע קשת דומה עם מה שנותר להזרים (ההפרש), ואז שוב v יהיה נגיש מ-s. לכן, **כל הקשתות בצד השמאלי שיכולות להיות הן רוויות**.
- סה"כ קיבלנו כי $C(S,T)=|f|$.

זרימה II

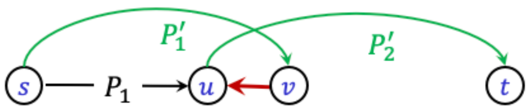
מסלולי שיפור קצרים ביותר

למה על מק"בים (המרחקים יכולים רק לגדול): יהי $G=(V,E)$ גרף מכוון, וצמתים $s,t\in V$. יהי P מק"ב מ-s ל-t (מספר קשתות). ניצור גרף חדש $G'=(V,E')$ כאשר אנחנו מוסיפים לגרף את הקשתות ההפוכות מהמסלול: $E'=E\cup P^R$. אז מתקיים כי $\delta_{G'}(s,t)\ge\delta_G(s,t)$.

הוכחה: נסתכל על P' המק"ב מ-s ל-t בגרף החדש G'. אם הוא לא משתמש בקשתות החדשות שהוספנו P^R אז P הוא גם מסלול בגרף המקורי G וסיימנו. אחרת, קיימת על המסלול הזה קשת $(v,u)\in P^R$.

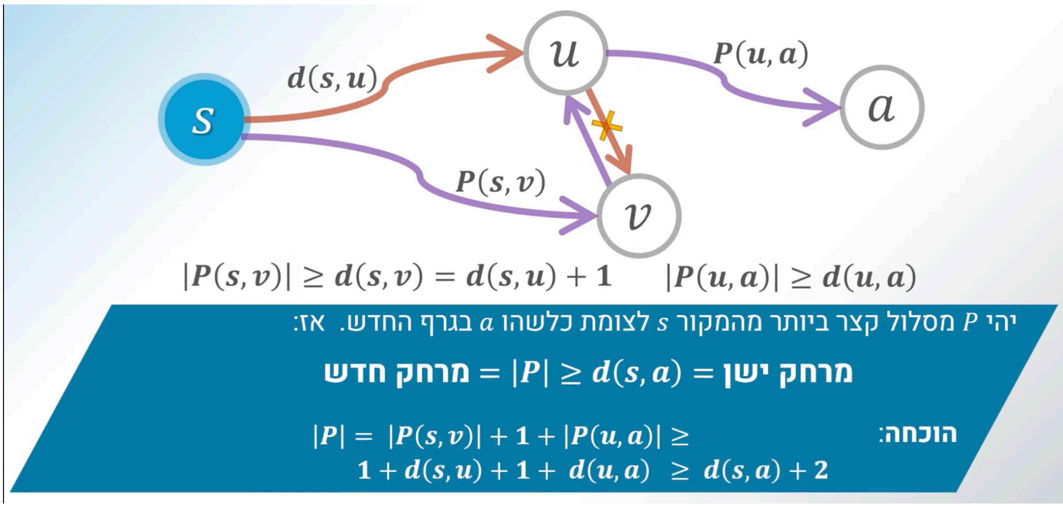


- נסתכל על החלקים P_1' מ-s ל-v ו-P_2' מ-u ל-t.
- אנחנו יודעים כי ב-P_1' אין קשתות מ-P^R ולכן הוא גם מסלול ב-G ומתקיים $|P_1'|\ge\delta_G(s,v)$.
- אנחנו יודעים כי $(u,v)\in P$, לכן מתקיים בגרף G כי $\delta_G(s,v)=\delta_G(s,u)+1$.
- נסתכל על המסלול המקוצר P_1 שהולך מ-s ישר ל-u. לכן, P_1P_2' הוא מסלול בגרף החדש שהינו קצר יותר מהמסלול P', וזו סתירה לכך ש-P' מק"ב.



למה על מק"בים (המרחקים יכולים רק לגדול) 2: דומה ללמה 1, כאשר $E'\subseteq E\cup P^R$, הפעם רק מוכל ולא שווה בדיוק לקבוצה הזאת. אפשר לעשות את התוספת ולהשמיט קשתות וזה עדיין יעבוד. **השמטת קשתות יכולה רק להגדיל מרחקים ולא להקטין.**

הוכחה של עבור גרף מכוון כללי: אם נהפוך את כיוון הקשת (u,v) במק"ב מצומת מקור s ל-v, אז המרחק מהמקור לצומת כלשהו a יכול רק לגדול. כאשר המק"ב אל הצומת a מכיל את הקשת ההפוכה, אז המרחק גדל ב-2.



מרחק ברשת השיורית: תהי f זרימה ב-G, כאשר G_f הרשת השיורית. יהי $\delta_f(v)=\delta_{G_f}(s,v)$ המרחק מ-s לכל צומת v ברשת השיורית, כאשר לכל קשת יש אורך 1 (המרחקים יכולים להיות ∞, לא בהכרח כל הצמתים נגישים מ-s).

למה 1: יהי P המסלול המשפר הקצר ביותר (במונחי מספר קשתות) מ-s ל-t עבור f. תהי f' הזרימה המתקבלת לאחר השיפור לפי המסלול P. אז $\delta_{f'}(v)\ge\delta_f(v)$ לכל צומת $v\in V$. כלומר, המרחקים שהגדרנו ברשת השיורית **רק גדלו**.

הוכחה: מה קורה כשמשפרים לפי מסלול P? אפשר להפוך קשתות לרוויות – נוסיף זרימה על קשתות קדמיות, ואז **קשת רוויה תיפול מהרשת השיורית**. במקרה אחר, **מצטרפות קשתות הפוכות**. בכל אופן זה נופל תחת למה 2 שראינו קודם, ולכן המרחקים לא יכולים להפוך לקטנים יותר. **נסמן קשת $(u,v)\in P$ בקריטית**, אם מתקיים $c_f(u,v)=c_f(p)$, כלומר הן יהפכו לרוויות אחרי שנשפר לפי המסלול P (נביא את הזרימה למקסימום הקיבול עליהן). לכל מסלול משפר P יש לפחות קשת אחת קריטית. אם $(u,v)\in P$ היא קריטית, אז הקשת עצמה לא תהיה ברשת השיורית.

אלגוריתם Edmonds-Karp

<u>האלגוריתם:</u> איך בוחרים מסלול משפר כדי לממש את שיטת FF? שתי בחירות טבעיות למימוש EK:

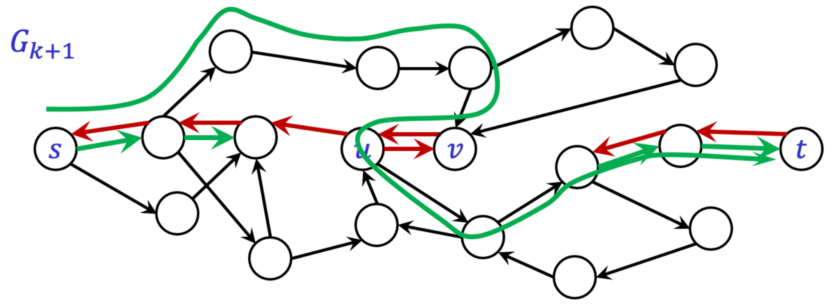
- EK1 – נפעל באופן greedy וניקח את זה שמשפר לנו בצורה הכי גדולה, המסלול בעל הקיבול השיורי הכי גדול.
- EK2 – נמצא את המסלול המשפר הקצר ביותר (באמצעות BFS).

נבחר ב-EK להיות EK2, ונטען כי **לכל סוגי קיבול (לאו דווקא שלמים)**, EK2 ישתמש לכל היותר ב-$O(|E||V|)$ מסלולי שיפור.

תהא $P_1, P_2, \dots$ סדרת מסלולים משפרים לשימוש על ידי EK. הזרימות המתקבלות הן $0 = f_1, f_2, \dots$, כלומר המסלול P_i משפר ב-G_{f_i}. לפי הלמה, לכל $u \in V$ מתקיים $\delta_{f_1}(u) \le \delta_{f_2}(u) \le \cdots$. ניזכר כי $c_f(P) = \min\{c_f(u,v) | (u,v) \in P\}$. קשת במסלול $(u,v) \in P$ היא קריטית אם $c_f(u,v) = c_f(P)$.

<u>למה 2:</u> אם קשת $(u,v) \in E \cup E^R$ קריטית גם על המסלול P_i וגם על P_j (היא מתייבשת אבל היא גם קמת לתחייה, מושמטת ומוחזרת), כאשר $i < j$, אז $\delta_{f_i}(u) < \delta_{f_j}(u)$. למעשה, המרחק גדול לפחות ב-2: $\delta_{f_i}(u) + 2 \le \delta_{f_j}(u)$.

<u>הוכחה:</u> אחרי שיפור לפי P_i, הקשת הקריטית (u,v) מושמטת מהרשת השיורית. הקשת מוחזרת לרשת רק אחרי $(v,u) \in P_k$ עבור $i < k < j$. המרחק ל-v הוא המרחק ל-u + 1, המרחק ל-u הוא המרחק ל-k + 1.



$(u,v) \in P_i$, and P_i is a shortest path in G_{f_i} $\Rightarrow$ $\boldsymbol{\delta_{f_i}(u) = \delta_{f_i}(v) - 1}$

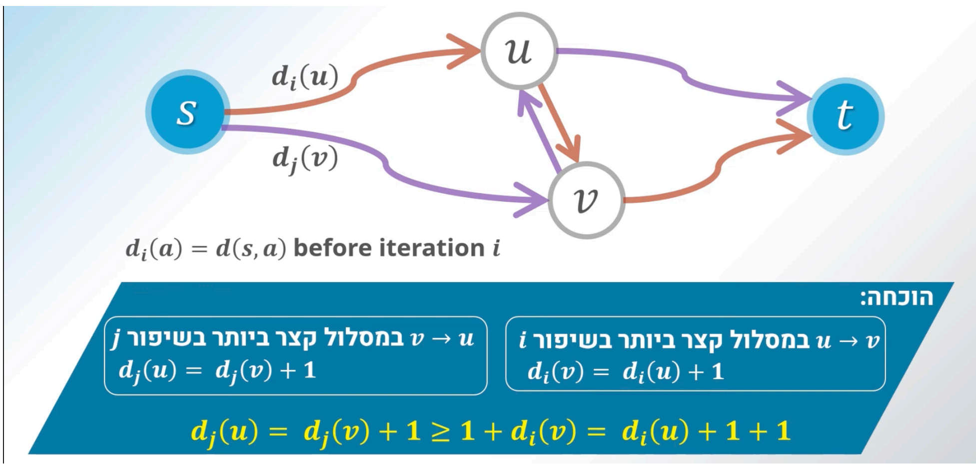
After using P_i, (u,v) *leaves* the res. network, i.e., $(u,v) \notin E_{f_{i+1}}$

(u,v) *reenters* $E_{f_{k+1}}$, for some $i < k < j$, only after $(v,u) \in P_k$

$(v,u) \in P_k$, and P_k is a shortest path in G_{f_k} $\Rightarrow$ $\boldsymbol{\delta_{f_k}(v) = \delta_{f_k}(u) - 1}$

$$\boldsymbol{\delta_{f_i}(u) = \delta_{f_i}(v) - 1}$$
$$\boldsymbol{\delta_{f_k}(v) = \delta_{f_k}(u) - 1 \Rightarrow \delta_{f_k}(u) = \delta_{f_k}(v)} + 1$$
$$\Rightarrow \delta_{f_i}(u) + 2 \le \delta_{f_k}(u) \le \delta_{f_j}(u) \quad \blacksquare$$

ברשת השיורית, אם קשת (u,v) מושמטת ומוחזרת, אז המרחק מ-s ל-u גדל ב-2, לכן קשת מושמטת לכל היותר $\frac{|V|}{2}$ פעמים, לפני שנגיע למרחק המקסימלי מ-s ל-u שהוא $|V|$.

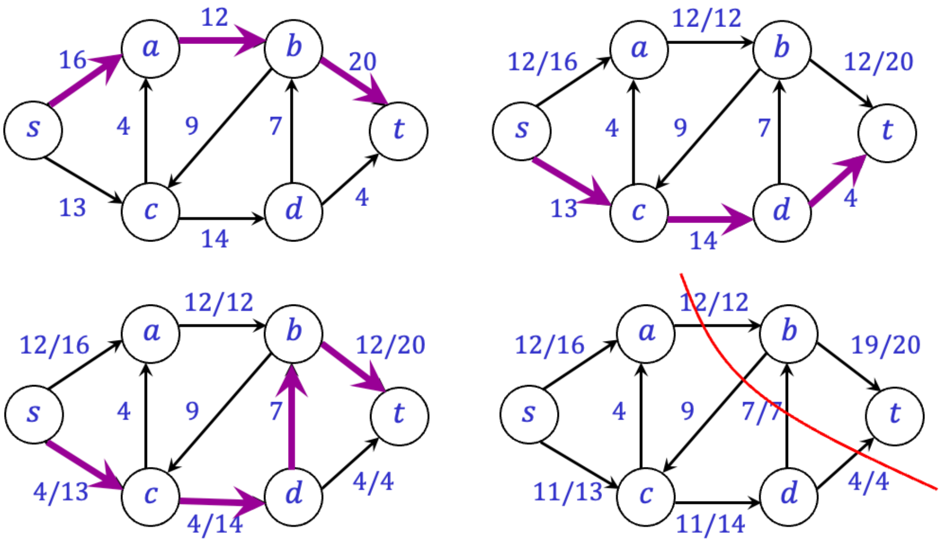


סיבוכיות:

- כל מסלול משפר מכיל לפחות קשת קריטית אחת. סה"כ, כמות המסלולים המשפרים היא $|E \cup E^R| \cdot \frac{|V|}{2} = |E| \cdot |V|$.
- בכל איטרציה אנחנו מחפשים מק"ב ברשת השיורית באמצעות BFS, בזמן $O(|E|)$.
- לכן, זמן הריצה הכולל של EK2 הוא $\boldsymbol{O(|E|^2|V|)}$.

דוגמת הרצה של EK:

- הזרימה ההתחלתית היא 0, ולכן הרשת היא גם הרשת השיורית ההתחלתית.
- יש לנו שני מסלולי שיפור קצרים ביותר באורך 3, המסלול העליון s→a→b→t והתחתון s→c→d→t. האלגוריתם של EK רשאי לבחור כל אחד מהם. **נניח שבחרנו את העליון**. רוחב המסלול הוא 12.
- כעת, ראינו שיש עוד מסלול שיפור באורך 3. אחת התכונות שהשתמשנו בהן בניתוח של EK הוא **שהמרחק בין s ל-t ברשת השיורית לא קטן**. אם 3 היה הכי קצר קודם אז 3 זה הכי קצר גם עכשיו. **נבחר את המסלול התחתון**, שרוחבו 4.
- כעת נשפר לפי המסלול s→c→d→b→t, היחיד שקיים בשלב זה, מאורך 4, ורוחבו 7. לאחר השיפור, קיבלנו זרימת מקסימום: כי מצאנו חתך שגודלו שווה לגודל הזרימה. ערך הזרימה המקסימאלית הוא 23.



חיבור וחיסור זרימות

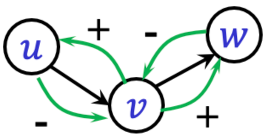
עד כה, כל האלגוריתמים שראינו הגדילו את הזרימה לאורך מסלול אחד (FF, EK): בחרו מסלול שיפור, הגדילו עליו את הזרימה וכו'. כעת, נרצה להגדיל את הזרימה במקביל במספר מסלולים. על כן, נגדיר איך אנחנו מגדילים זרימה באופן הזה: נמצא זרימה ברשת השיורית, ונרצה להוסיף את הזרימה הזאת לרשת המקורית.

תהא $G = (V, E, c, s, t)$ רשת זרימה, f זרימה ב-G, ו-G_f הרשת השיורית. **הרשת השיורית היא גם רשת זרימה בפני עצמה.**

זרימה שיורית: זרימה g ברשת הזרימה השיורית G_f היא זרימה שיורית.

חיבור - שיפור זרימה ע"י זרימה שיורית: השיפור של f באמצעות g, המסומנת $f \uparrow g$ (f מוגדל ב-g) מוגדר כך שלכל קשת $(u, v) \in E$ בזרימה המקורית, נוסיף את הזרימה השיורית ונחסיר את הזרימה השיורית על הקשת ההפוכה:

$$(f \uparrow g)(u, v) = f(u, v) + g(u, v) - g(v, u)$$



- יכול להיות שהקשת (u, v) לא קיימת ברשת השיורית, ואז נגדיר $g(u, v) = 0$.
- יכול להיות שהקשת ההפוכה (v, u) לא קיימת ברשת השיורית, ואז נגדיר $g(v, u) = 0$.

למה: אם f זרימה ב-G, ו-g זרימה ב-G_f, אז $f \uparrow g$ זרימה ב-G ומתקיים $|f \uparrow g| = |f| + |g|$.

הוכחה:

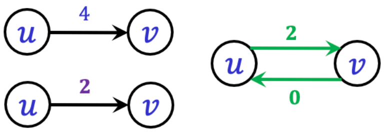
- נוכיח ש-$f \uparrow g$ היא זרימה חוקית:
  - **שימור** – עבור צומת $v \in V$, נבדוק כמה זרימה יוצאת מ-v וכמה נכנסת ל-v: באמצעות שימור הזרימה על f ועל g נקבל שימור זרימה על כל הביטוי.
  - **קיבול** – ניעזר בכך שגם ברשת השיורית לכל זרימה g מתקיים $g(e) \le c_f(e)$.
- גודל הזרימה, אם $v = s$ נקבל כי הביטוי שווה ל-$|f| + |g|$.

הערה: נשים לב, כי עד כה מצאנו מסלול משפר והגדלנו עליו את הזרימה. **זהו מקרה פרטי של הלמה שהוכחנו.** אם P הוא מסלול שיפור עבור זרימה f, בעל קיבול שיורי/רוחב/bottleneck $c_f(P)$, אנחנו יכולים באמצעות המסלול P להגדיר זרימה מאוד פשוטה ברשת השיורית, שנקרא לה g_P: נזרים $c_f(P)$ יחידות זרימה על הקשתות של המסלול P. **לפי הלמה אפשר לקחת את הזרימה g_P ולהגדיל באמצעותה את הזרימה f: $f\uparrow g_P$.**

חיסור בין זרימות: תהא G רשת זרימה, בה יש שתי זרימות f, g, כאשר הרשתות השיוריות המתאימות הן G_f, G_g. החיסור של g מ-f, המסומן $f\downarrow g$ (ההקטנה של f ב-g) מוגדר כך:

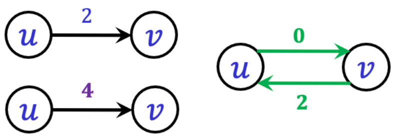
- $(f\downarrow g)(u,v)=\left(f(u,v)-g(u,v)\right)^{+}$ עבור הקשת $(u,v)\in E$.

$$\text{If } f(u,v)\geq g(u,v)\text{, then}$$
$$(f\downarrow g)(u,v)=f(u,v)-g(u,v)$$
$$(f\downarrow g)(v,u)=0$$



- $(f\downarrow g)(v,u)=\left(g(u,v)-f(u,v)\right)^{+}$ עבור הקשת $(u,v)\in E$. זאת כי ברשת השיורית יש גם קשתות הפוכות.

$$\text{If } f(u,v)< g(u,v)\text{, then}$$
$$(f\downarrow g)(u,v)=0$$
$$(f\downarrow g)(v,u)=g(u,v)-f(u,v)$$



כאשר האופרטור $x^{+}=\max\{0,x\}$ אומר שאם נקבל תוצאה שלילית נחליף אותה ב-0.

למה: אם f, g זרימות ב-G, אז $f\downarrow g$ היא זרימה ב-G_g ומתקיים $|f\downarrow g|=|f|-|g|$. בנוסף, $g\uparrow(f\downarrow g)=f$.

למה: אם f זרימה מקסימלית ב-G, ו-g זרימה מקסימלית ב-G_f, אז $g\uparrow f$ זרימה מקסימלית ב-G.

אלגוריתם Dinic

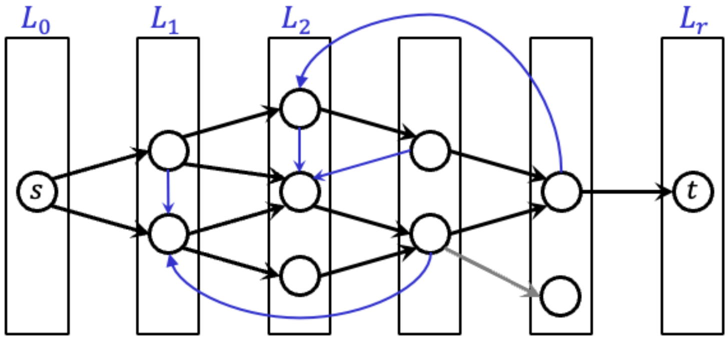
האלגוריתם: נתחיל עם זרימה חוקית, למשל זרימת האפס, $f=0$. נבנה את הרשת השיורית G_f.

- אם אין מסלול מכוון מ-s ל-t ברשת השיורית, אנו יודעים כי f היא זרימת מקסימום.
- אחרת, נבנה את G_f^L, רשת שיורית לפי רמות, **המכילה את כל הקשתות שמופיעות על מק"בים (ביחס לכמות קשתות) מ-s ל-t ברשת השיורית.**
  - נמצא זרימה חוסמת g (blocking flow) ברשת G_f^L.
  - בפרט הזרימה החוסמת היא זרימה ב-G_f, לכן נשפר את הזרימה המקורית באמצעות הזרימה הזו: $f=f\uparrow g$.

הערות:

- לכל איטרציה של האלגוריתם נקרא blocking step, בו אנו מוצאים זרימה חוסמת.
- זרימה חוסמת – f זרימה חוסמת אם אין אף מסלול P מהמקור לבור שכל הקשתות בו אינן רוויות (מסלול שניתן לשפר עליו).
- EK הציע למצוא בכל איטרציה מסלול שיפור קצר ביותר, ועליו לשפר וכך הלאה. מה שדיניץ מציע, לנסות למצוא בבת אחת מספר גדול בכל האפשר של מסלולי שיפור קצרים בכל האפשר. כיוון שאנחנו מתעניינים רק במק"בים, דיניץ מציע להסתכל רק על הקשתות שיכולות להופיע על מסלולי שיפור קצרים ביותר, ובאמצעותם נבנה תת-רשת G_f^L.

רשת השכבות השיורית (Leveled residual network): תהא G רשת זרימה עם קיבולים חיוביים על כל הקשתות, f זרימה ב-G, G_f היא הרשת השיורית. כמו ב-EK, נגדיר עבור כל צומת את המרחק מ-s אליו ברשת השיורית: $\delta_f(v)=\delta_f(s,v)$. נגדיר את הרשת השיורית לפי רמות: $G_f^L=(V,E_f^L,c_f,s,t)$, תת-רשת של G_f כאשר: $\boldsymbol{E_f^L=\{(u,v)\in E_f|\delta_f(v)=\delta_f(u)+1\}}$. כלומר, **קשתות שמופיעות על מק"בים מ-s, לא בהכרח ל-t.** (אפשר לחזק את הטענה ולכלול קשתות במק"בים מ-s עד ל-t)



<u>הגדרת השכבות:</u> נגדיר את $L_i = \{v \in V | \delta_f(s,v) = i\}$.
ברמה L_0 יהיה רק הצומת s, ובכל רמה L_i יופיעו הצמתים שמרחקם מ-s הוא i.

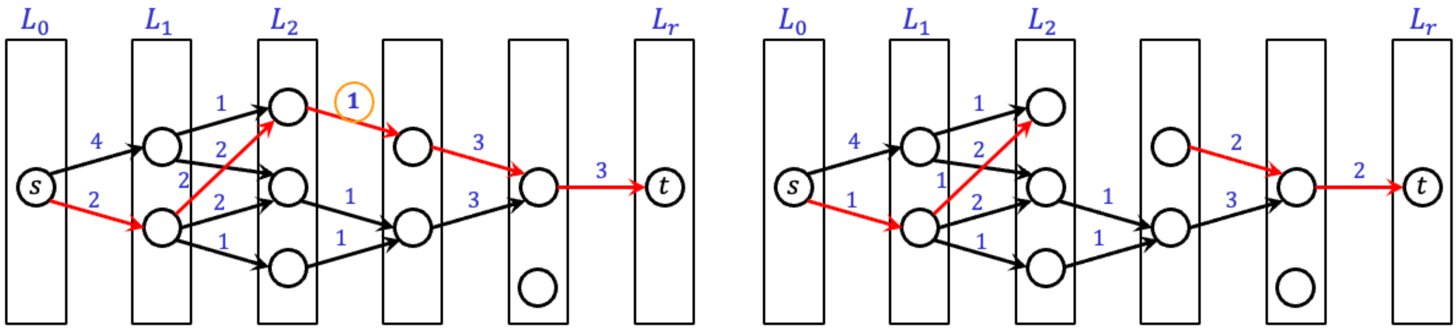
- את כל הקשתות הכחולות אנחנו זורקים, הן לא שייכות לרשת הרמות, כי הן לא הולכות מרמה אחת לרמה הבאה.
- רק קשתות מ-L_i ל-L_{i+1} יהיו ב-G_f^L.
- בנוסף, קשתות שהולכות מרמה לרמות שנמצאות לפניה אנחנו גם זורקים.

אפשר למצוא את E_f^L באמצעות BFS בזמן $O(|E|)$.

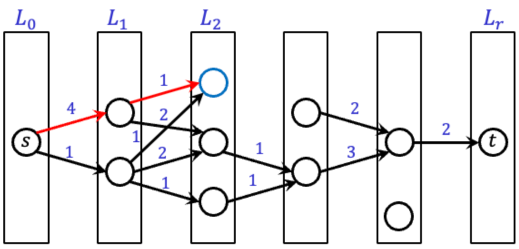
<u>מציאת blocking flow:</u>

איך נמצא זרימה חוסמת ברשת זרימה חסרת מעגלים G? בהמשך נפעיל את זה על הרשת השיורית G_f:

- נאתחל זרימה $f = 0$.
- בלולאה:
  - **נמצא מסלול מכוון P מ-s ל-t**. אם אין כזה, f היא כבר זרימה חוסמת.
  - אם מצאנו כזה, נשפר לפי המסלול בדומה למה שביצענו בעבר לפי bottleneck של מסלול משפר: **נשפר את הזרימה על P לפי** $c(P) = \min_{(u,v)\in P} c(u,v)$. **נקטין את הקיבול על כל הקשתות של P ב-$c(P)$.** נמחק את כל הקשתות שהקיבול שלהן ירד ל-0 (לא נשתמש בהן עוד).

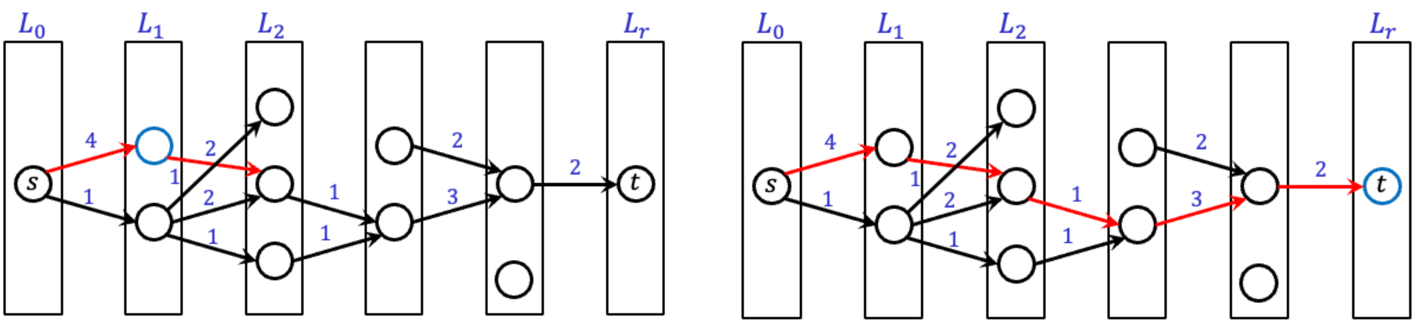


איך נמצא מסלול מכוון מ-s ל-t? באמצעות DFS עם שינוי קל: נאתחל $P = \emptyset, v = s$. כל עוד $v \neq t$:



- אם יש קשת $(v,w) \in E$ **נתקדם** עליה: $P = P \cup (v,w)$, נעדכן $v = w$.
- אחרת (אם אין יותר קשתות שיוצאות מ-v, כלומר מחקנו אותן באיטרציות קודמות), **ניסוג** עליה: נמחק את הקשת האחרונה על המסלול.
  - אם $v = s$ (אז P ריק) ואנחנו תקועים – לכן נחזיר שאין יותר מסלולים מכוונים מ-s ל-t.
  - אחרת, תהא (u,v) הקשת האחרונה על P, נמחק את (u,v) מ-P וגם מ-G, ונגדיר את $v = u$.

נחזיר את הקבוצה P – זה המסלול מ-s ל-t.



נכונות:

למות על מק"בים:

למה 1: יהי $G=(V,E)$ גרף מכוון, וצמתים $s,t\in V$. נסמן ב-$(E^L)^R=\{(v,u)|(u,v)\in E \text{ and } \delta_G(s,v)=\delta_G(s,u)+1\}$, הקשתות ההופכיות לקשתות של רשת השכבות. נגדיר את הגרף G' שהוא הוספה של הקשתות החדשות ל-E. אז נקבל כי המרחקים לא קטנים: $\delta_{G'}(s,t)\geq\delta_G(s,t)$.
בנוסף, אם המרחקים זהים, ו-P מק"ב מ-s ל-t ב-G', אז P גם מק"ב מ-s ל-t בגרף המקורי G.

למה 2: יהי $G=(V,E)$ גרף מכוון, וצמתים $s,t\in V$. נסמן ב-$(E^L)^R=\{(v,u)|(u,v)\in E \text{ and } \delta_G(s,v)=\delta_G(s,u)+1\}$, הקשתות ההופכיות לקשתות של רשת השכבות. נגדיר את הגרף G' שהוא הוספה של הקשתות החדשות ל-E תוך הסרת הקשתות של B כאשר זוהי קבוצת הקשתות כך שעל כל מק"ב מ-s ל-t חייבת להיות לפחות קשת אחת של B. כאן, המרחק חייב לגדול ממש, כלומר $\delta_{G'}(s,t)>\delta_G(s,t)$.

סיבוכיות:

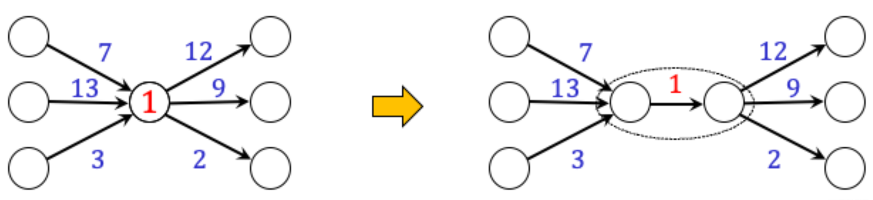
- למה: האלגוריתם של דיניץ מוצא זרימה מקסימלית לאחר לכל היותר $|V|$ צעדים (blocking steps).
  - הוכחה: לפי למה 2, כל פעם שחישבנו blocking flow מ-s ל-t הגדלנו את המרחק מ-s ל-t ברשת השיורית. המרחק ההתחלתי ברשת השיורית הוא לפחות 1, וברשת השיורית האחרונה הוא לכל היותר $|V|-1$ לכן מספר ה- blocking steps הוא לכל היותר $|V|-2$.
- למה: אפשר למצוא זרימה חוסמת ברשת שיורית לפי רמות בזמן $O(|E||V|)$.
  - הוכחה: כל פעולת התקדמות (**advance**) או נסיגה (**retreat**) מקשת כלשהי לוקחת $\boldsymbol{O(1)}$. כל **פעולת שיפור** הזרימה על מסלול מכוון שמצאנו לוקחת $\boldsymbol{O(|V|)}$ לכל היותר אורך המסלול P. כל **פעולת שיפור**, לפחות קשת אחת הפכה להיות רוויה (זו שהגדירה את רוחב המסלול), לכן **מספר המסלולים שנמצא הוא לכל היותר $\boldsymbol{|E|}$**.
    - כמה פעמים נבצע retreat? כל פעם אנחנו מוחקים קשת, לכן לכל היותר נבצע זאת $|E|$ פעמים.
    - כמה פעמים נבצע advance? יש שני סוגים:
      - כזה שאין אחריו צעדי retreat (ממשיכים עד שמגיעים למסלול שמגיע ל-t). המסלול מכיל לכל היותר $|V|$ קשתות שעשינו עליהם advance, ויש סה"כ $|E|$ מסלולים, סה"כ $|V||E|$.
      - כזה שאחריו יש צעד retreat, זה חסום על ידי מספר צעדי ה-retreat, שזה $|E|$.
    - סה"כ כמות צעדי advance היא $|V||E|+|E|$, והעלות הכוללת למציאת זרימה חוסמת היא $O(|E||V|)$.
- סה"כ נקבל כי האלגוריתם של דיניץ מוצא זרימה מקסימלית בזמן $\boldsymbol{O(|E||V|^2)}$. זה עדיף על EK כיוון שלרוב מספר הקשתות יהיה הרבה יותר ממספר הצמתים.

הרצת Dinic ברשתות מיוחדות

רשתות יחידה ("סוג 2"):

רשת יחידה: רשת $G=(V,E,c,s,t)$ שבה לכל צומת $v\in V\backslash\{s,t\}$ יש **או רק קשת נכנסת אחת** (עם קיבול 1) **או רק קשת יוצאת אחת** (עם קיבול 1). נניח **שכל הקיבולים הם 1**.

לחלופין, אפשר לנסח את התנאי בתור הוספת אילוץ קיבול לצמתים. בהינתן רשת זרימה כללית, לכל צומת יש אילוץ על כמות הזרימה המקסימלית שיכולה לעבור דרכו. נרצה שכמות הזרימה הזאת תהיה לכל היותר 1. כיצד? נפצל כל צומת שאנחנו רוצים לשים עליו אילוץ, לשני צמתים שיחוברו בקשת במכוונת, עם קיבול 1.



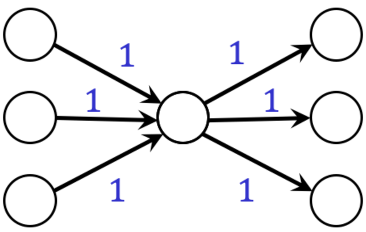
זיווג מקסימום בגרף דו-צדדי: **רדוקציה מזיווג לבעיית זרימה מקסימאלית מייצרת רשת יחידה**. בהרצת FF, נקבל זמן ריצה של $O(|E||V|)$ לזיווג מקסימום. אמנם, **אם נריץ את האלגוריתם של דיניץ נקבל זמן ריצה של $\boldsymbol{O(|E|\sqrt{|V|})}$**. נראה שאותו אלגוריתם עובד באופן כללי על רשתות יחידה ולאו דווקא רשתות מהסוג שמקבלים מבעיית זיווג בגרף דו-צדדי.

למה (flow decomposition): תהא $G=(V,E,c,s,t)$ **רשת יחידה**, וזרימה $f:E\to\{0,1\}$ מסוג 0-1 ברשת. אז, אפשר להציג את f כסכום של זרימות לאורך $\boldsymbol{|f|}$ **מסלולים זרים בצמתים** מ-s ל-t (פרט ל-s ול-t), ואוסף של **מעגלים זרים בצמתים**.

ניתוח סיבוכיות דיניץ ברשתות סוג 2:

- למה: ברשתות יחידה, האלגוריתם מבצע בדיוק $\boldsymbol{O}(\sqrt{|\boldsymbol{V}|})$ צעדי blocking steps.
  - הוכחה: נראה כי האלגוריתם מבצע $2\sqrt{|V|}$ צעדים. אם G רשת יחידה ו-f זרימה שלמה ב-G, אז גם G_f, G_f^L הן רשתות יחידה. **אחרי** $\sqrt{|V|}$ **צעדים**, המרחק מ-s ל-t ברשת השכבות השיורית הוא לפחות $\sqrt{|V|}+2$ (כיוון שהמרחק גדל לפחות ב-1 בכל צעד, והמרחק היה לפחות 1 בהתחלה).
  אם רק נסתכל מבחינת מרחק, לא נראה שהתקדמנו יותר מידי. כיוון שאנחנו ברשת יחידה, נסתכל על גודל הזרימה וניעזר בלמה של flow decomposition: זרימה g ב-G_f ניתן להציג כסכום של $|g|$ מסלולים זרים בצמתים מ-s ל-t. כל מסלול כזה מכיל לפחות $\sqrt{|V|}$ צמתי ביניים, לכן **בזרימה מקסימלית** יכולים להיות לכל היותר $\frac{|V|}{\sqrt{|V|}}=\sqrt{|V|}$ מסלולים כאלה. המסקנה היא שההפרש בין גודל זרימת מקסימום לזרימה שכרגע בידינו הוא לכל היותר $\sqrt{|V|}$. עכשיו אפשר לשכוח מדיניץ, כל זרימה חוסמת שמצאנו מגדילה את הזרימה בלפחות 1, אז נצטרך **לכל היותר** $\sqrt{|\boldsymbol{V}|}$ **איטרציות נוספות של שיפורים עד שנגיע לזרימה מקסימאלית.**
- למה: אפשר למצוא זרימה חוסמת ברשת השכבות השיורית של רשת יחידה בזמן $\boldsymbol{O}(|\boldsymbol{E}|)$.
  - הוכחה: את כל הקשתות על המסלול שמצאנו (מכיוון שהקיבול שלהן הוא 1 והן הופכות להיות רוויות) אפשר לזרוק. לכן, מספר צעדי ה-advance/retreat הוא $O(|E|)$. הפעם, אחרי advance גם אם אין retreat לאחר מכן, היא מופיעה על מסלול ומיד מוחקים אותה. העלות הכוללת של ההגדלות היא $O(|E|)$, לא נזרים זרימה על קשת יותר מפעם אחת.
- סה"כ נקבל כי דיניץ מוצא זרימה מקסימאלית ברשת יחידה בזמן $\boldsymbol{O}(|\boldsymbol{E}|\sqrt{|\boldsymbol{V}|})$.

רשתות קיבול יחידה ("סוג 1"):



רשת קיבול יחידה: רשת זרימה שכל הקיבולים על הקשתות שלה הם 1 (אין קשתות מקבילות).

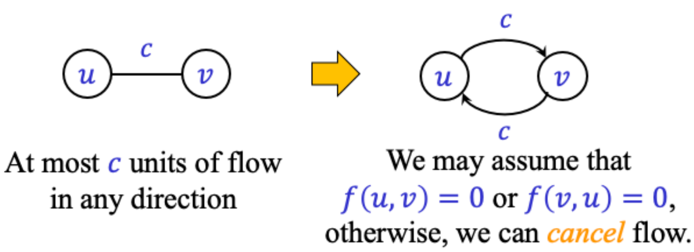
ניתוח סיבוכיות דיניץ ברשתות סוג 1:

- למה: ברשתות קיבול יחידה, האלגוריתם מבצע $\boldsymbol{O}\left(\min\left\{|\boldsymbol{E}|^{\frac{1}{2}}, |\boldsymbol{V}|^{\frac{2}{3}}\right\}\right)$ צעדי blocking steps.
  - הוכחה: אנחנו לא יודעים שהזרימה מתפרקת למסלולים זרים בצמתים, רק למסלולים **זרים בקשתות**. ניתן להראות כמו קודם כי האלגוריתם מבצע לכל היותר $\boldsymbol{2}\sqrt{|\boldsymbol{E}|}$ צעדים. כעת נראה כי האלגוריתם מבצע בדיוק $2|V|^{\frac{2}{3}}$ צעדים. נניח **שביצענו כבר** $|\boldsymbol{V}|^{\frac{2}{3}}$ **צעדים**. המרחק הוא לפחות $|V|^{\frac{2}{3}}$, ולכן יש לפחות $|V|^{\frac{2}{3}}$ רמות ברשת השכבות השיורית. לכן, יש לפחות רמה אחת שמכילה לכל היותר $|V|^{\frac{1}{3}}=\frac{|V|}{|V|^{\left(\frac{2}{3}\right)}}$ צמתים. נטען שיש **לפחות זוג רמות עוקבות שמכילות** $\boldsymbol{2}|\boldsymbol{V}|^{\frac{1}{3}}$ **צמתים**. החתך שמוגדר על ידי שתי הרמות האלה מכיל לכל היותר $|V|^{\frac{2}{3}}$ קשתות (יכולות לחבר בין הרמות האלה לכל היותר כמות קשתות כמו המכפלה בין כמויות הצמתים), לכן לכל זרימה g ב-G_f, מתקיים $|g|\le|V|^{\frac{2}{3}}$, זה ההפרש מזרימת המקסימום, ולכן לאחר לכל היותר $|V|^{\frac{2}{3}}$ צעדים נוספים נמצא זרימת מקסימום.
- סה"כ נקבל כי דיניץ מוצא זרימה מקסימאלית ברשת יחידה בזמן $O\left(\min\left\{|E|^{\frac{3}{2}}, |E||V|^{\frac{2}{3}}\right\}\right)$.

קשתות אנטי-מקבילות:

כשהגדרנו את הרשת השיורית G_f, הנחנו שב-G אין קשתות אנטי-מקבילות. אם יש קשת $(u,v)\in E$ הנחנו שאין לה קשת הופכית. זה אפשר לנו להניח כי $E\cap E^R=\emptyset$. כפי שנראה, ההנחה הזו לא הכרחית.

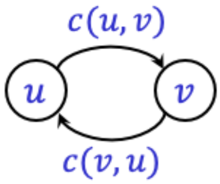
למה שנרצה להרשות אותן? לפעמים באופן טבעי אנחנו מקבלים קשתות אנטי-מקבילות. למשל ברשתות זרימה לא מכוונות (צינורות, חשמל). איך נתרגם זרימה לצורה החדשה? נכניס קשתות אנטי-מקבילות.



פתרון 1: פתרון פשוט, אך לא אלגנטי. **על אחת הקשתות נוסיף באמצע צומת**. מספר הצמתים יכול לגדול באופן משמעותי (ועל כל אחת מבין כל שתי קשתות מוסיפים צומת). למרבה המזל זמן הריצה של דיניץ נשאר אותו דבר. כשניתחנו את האלגוריתם, הדבר העיקרי שבו השתמשנו ב-$|V|$ זה היה כדי לחסום את המרחק מ-s ל-t, מרחק שגדל פי 2 לכל היותר כאן.

Number of vertices may increase substantially.
If $G = (V, E, c, s, t)$ is the original network, and $G' = (V', E', c', s, t)$ is the new network, then $|V'| \le |V| + \frac{|E|}{2}$ and $|E'| \le \frac{3}{2}|E|$.

פתרון 2: **נשנה את ההגדרות** של רשת שיורית, קיבול שיורי, ושיפור זרימה $(f \uparrow g)$. אנחנו מניחים שהקשתות באות בזוגות של קשתות אנטי-מקבילות. לכל קשת יש קשת אנטי-מקבילה, ולכן $E^R = E$. נעדכן את הגדרת הקיבול השיורי, בתוספת $f(v,u)$ כיוון שהזרימה נכנסת לצומת.



If $(u,v) \in E$ and $(v,u) \notin E$,
we add (v,u) to E with $c(v,u) = 0$.

Now $E^R = E$.

Residual capacities:

$$c_f(u,v) = \big(c(u,v) - f(u,v)\big) + f(v,u) \quad , \quad (u,v) \in E$$

Residual network: $G_f = (V, E_f, c_f, s, t)$,

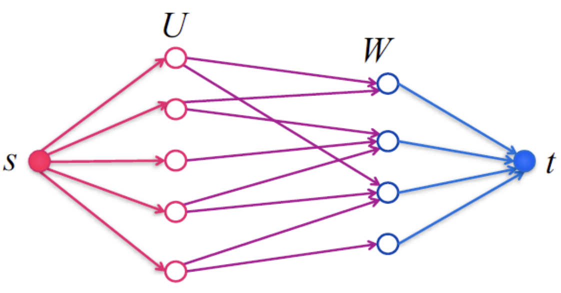
$$E_f = \{\, (u,v) \mid (u,v) \in E \,,\ c_f(u,v) > 0 \,\}$$

זרימה 3 (תרגול 11)

תרגיל 0 (זיווג מושלם בלוח שח): בהינתן גרף עם מספר זוגי של צמתים, **זיווג מושלם** הינו זיווג שבו כל צומת מזווג. ייתכן זיווג מושלם בגרף דו צדדי רק אם $|U| = |W|$. זה תנאי בסיסי שחייב להתקיים. **בלוח שח** אנחנו רוצים לזווג שחורים ולבנים, אז אם נסיר 2 משבצות לבנות לא נוכל למצוא זיווג מושלם – יישארו 2 שחורים ללא זיווג.

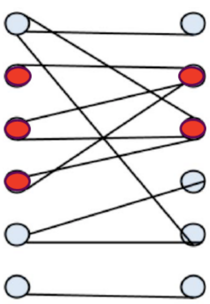
נתון לוח שח בגודל $n \times n$ לאחר שהוסרו ממנו חלק מהמשבצות. תארו אלגוריתם שבודק האם ניתן לכסות אותו על ידי אבני דומינו.

- נבנה גרף באופן הבא: ניצור קודקוד עבור כל משבצת, ניצור קשת עבור כל זוג משבצות סמוכות.
- זהו גרף דו צדדי – צד אחד מכיל את המשבצות הלבנות, והשני מכיל את המשבצות השחורות.
- קיים כיסוי חוקי של אבני דומינו $\Leftrightarrow$ קיים זיווג מושלם בגרף. זיווג מושלם מתאים לקבוצת אבני דומינו שאינן חופפות (לכן זה זיווג), ומכסות את כל המשבצות (לכן הוא מושלם).
- נבנה רשת מטיפוס 2 (רשת יחידה), ונוכל למצוא זיווג מקסימום על ידי דיניץ בזמן $\boldsymbol{O}(|\boldsymbol{E}|\sqrt{|\boldsymbol{V}|})$.



תרגיל 1 (משפט החתונה של Hall): עבור גרף דו צדדי לא מכוון $G = (U, W, E)$ ותת קבוצה $X \subseteq U$ נגדיר את השכנים של הקבוצה להיות כל הקודקודים שהם שכנים של קודקודים ב-X: $N(X) = \{y \in W : (x,y) \in E \ for\ some\ x \in X\}$. נתון גרף כזה כך שמתקיים $|U| = |W| = n$. הוכיחו שקיים זיווג מושלם בגרף $\Leftrightarrow$ לכל תת קבוצה $X \subseteq U$ מתקיים $|X| \le |N(X)|$.

פתרון: אם מצאנו קבוצה X כלשהי שעבורה $|X| \ge |N(X)|$, את אחד האיברים ב-X לא נצליח לזווג, יש מעט מידי בני זוג פוטנציאליים.

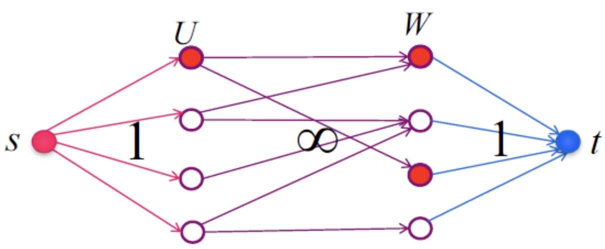


לא.

⇐ נניח שקיים זיווג מושלם בגרף, נוכיח שלכל תת קבוצה $X \subseteq U$ מתקיים התנאי. נבחר זיווג כזה ונסמן ב-$Y \subseteq W$ את קבוצת הקודקודים שמתאימים בזיווג לקודקודי X. אנחנו יודעים כי $|Y| = |X|$. כל קודקוד ב-Y בפרט שכן של קודקוד ב-X. לכן $Y \subseteq N(X)$. לכן $|Y| \leq |N(X)|$ וקיבלנו $|X| \leq |N(X)|$.

⇒ נניח שלכל תת קבוצה התנאי מתקיים. נראה שקיים זיווג מושלם = נראה שעל הרשת הזאת יש זרימה בגודל n (מקסימאלית). נהפוך את הגרף לרשת זרימה, וניתן לקשתות מ-U ל-W קיבול אינסופי (כל שאר הקיבולים הם 1). נוכיח שהחתך המינימלי בגודל n, מספיק שנוכיח **שכל חתך הוא בגודל לפחות n**, **לכן להגדיל את הקיבולים רק יעזור לנו להוכיח שהחתכים הם גדולים** (וזה לא פוגע לנו בשאלה, כי שאר הקיבולים משני הצדדים מ-s, ול-t מגבילים את הזרימה ל-1 על כל צומת.

נראה שכל חתך (S,T) שהוא בגודל **סופי**, הוא לפחות בגודל n. נסמן באדום קודקודים מ-S, ובכחול קודקודים מ-T. אנחנו רוצים להבין מה הגודל של החתך, כמה "תורם" כל קודקוד לחתך? כל הקבוצות האלה זרות, וביחד נותנות את כל הקודקודים.



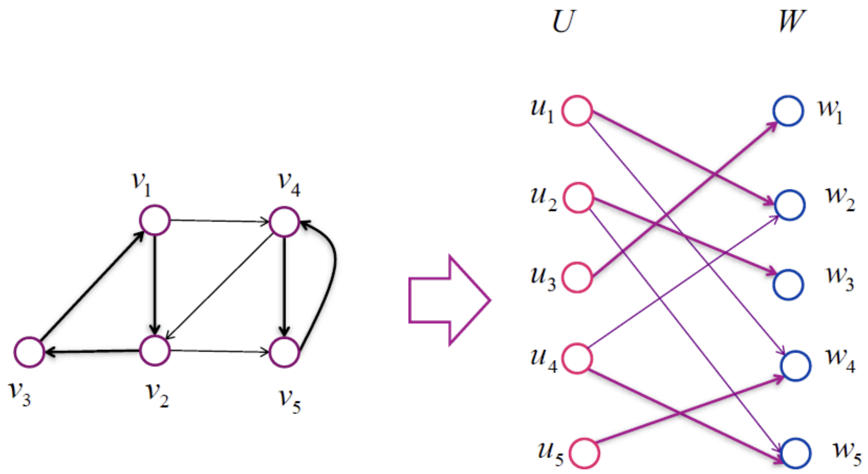
1. קודקודים כחולים (T) ב-U: יש לנו קשת שתורמת 1 לחתך (s,u).
2. קודקודים אדומים (S) ב-U: אין קשתות שתורמות לחתך. זו תת קבוצה של U ולכן לפי הנתון $|S \cap U| \leq |N(S \cap U)|$ כל השכנים של האדומים הם גם אדומים (אחרת הם היו כחולים והייתה לנו קשת אינסופית). לכן $N(S \cap U) \subseteq S \cap W$. סה"כ נקבל כי $|S \cap U| \leq |S \cap W|$.
3. קודקודים כחולים (T) ב-W: אין קשתות שתורמות לחתך.
4. קודקודים אדומים (S) ב-W: יש לנו קשת 1 שתורמת לחתך, (w,t).

גודל החתך הוא סכום של כל מי שתורם לנו לחתך (1 ו-4): $c(S,T) = |T \cap U| + |S \cap W| \geq |T \cap U| + |S \cap U| = |U| = n$.

<u>תרגיל 2 (תת-גרף עם דרגות 1):</u> נתון גרף מכוון $G = (V,E)$. תארו אלגוריתם המוצא קבוצת קשתות $E' \subseteq E$ (נוריד חלק מהקשתות המקוריות) כך שלכל קודקוד בגרף $G' = (V,E')$ דרגת כניסה 1 ודרגת יציאה 1.

<u>פתרון:</u> נרצה לשדך בין קודקודים, **כך שכל קודקוד יבחר קודקוד אחר (דרגת יציאה 1) וגם כל קודקוד יבחר על ידי קודקוד אחר (דרגת כניסה 1), נרצה שידוך מושלם כזה.** לכן, נניח שהקודקודים בגרף הם $v_1, \dots, v_n$.

- ניצור גרף דו-צדדי ונשכפל כל קודקוד: הקבוצה U תייצג דרגת יציאה, והקבוצה W תייצג דרגת כניסה: $u_1, \dots, u_n$ ו-$w_1, \dots, w_n$. נוסיף s ו-t שיתחברו לכולם.
- הקשתות בין U ל-W: אם יש קשת (v_i, v_j) נרצה ליצור קשת (u_i, w_j).
- נבדוק האם בגרף החדש קיים זיווג מושלם, אם כן נחזיר את קבוצת הקשתות בגרף המקורי שמתאימות לזיווג.



<u>נכונות:</u> נוכיח שבגרף החדש יש זיווג מושלם ⟺ יש תת קבוצה של קשתות שעונה על הדרישה:

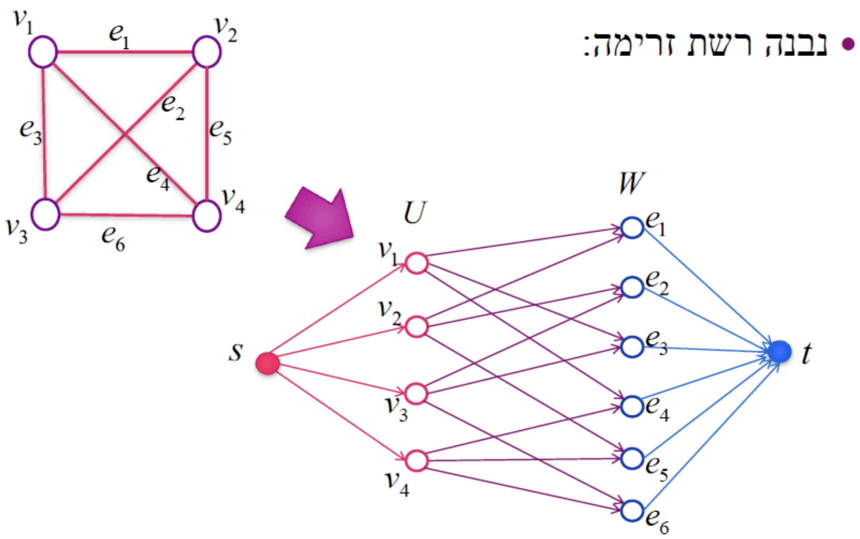
⇐ נניח שיש זיווג מושלם בגרף החדש, אז לכל קשת בזיווג (u_i, w_j) נבחר את הקשת (v_i, v_j) לקבוצה. למה זה מקיים את התנאים של השאלה? כיוון שזה זיווג מושלם בחרנו ב-u_i בדיוק פעם אחת ולכן v_i יופיע פעם אחת בקודקוד שיוצאת ממנו קשת. כך בהתאם יופיע פעם אחת v_j בקודקוד שנכנסת אליו קשת. לכן לכל קודקוד v תהיה דרגת יציאה בדיוק 1 ודרגת כניסה בדיוק 1.

⇒ נניח שיש תת קבוצה של קשתות שעונה על הדרישה, אז לכל קשת (v_i, v_j) בקבוצה נבחר את הקשת (u_i, w_j). קבוצת הקשתות שנבחר מהווה זיווג מושלם: כל קודקוד $u \in U$ מזווג, ובדיוק לקודקוד אחד, וגם כל $w \in W$.

תרגיל 3 (אילוצי דרגות): נתון גרף לא מכוון $G = (V, E)$. תארו אלגוריתם שמכוון את קשתות הגרף, כך שבגרף שיתקבל לכל קודקוד תהיה דרגת יציאה לכל היותר 3.

פתרון: נחשוב על זה בצורה הבאה: **שכל קודקוד צריך לבחור את הקשתות שיוצאות ממנו, ומותר לו לבחור לכל היותר 3 כאלו. בנוסף, נצטרך שכל קשת תיבחר על ידי מישהו**. נבנה רשת זרימה בהתאם.

- זה מזכיר לנו את בעיית המכונות והמשימות מהתרגול הקודם, המכונות הן U (הקודקודים שצריכים לבחור קשתות), והמשימות הן W (לכוון כל קשת פעם אחת). נמתח קשת בין כל קודקוד ב-U, ל-"קשת" ב-W, אם הקודקוד נוגע בקשת (יכול לבחור בה). **דרגת הכניסה של כל קודקוד (המייצג קשת) ב-W היא 2: יש שני קודקודים שנוגעים בכל קשת.**
- הקיבולים: לקודקודים נגביל את הקיבול ב-3, ואת הבחירה נגביל ב-1 (כל קשת נבחרת פעם אחת בלבד).



נכונות: קיים כיוון תקין של הגרף ⇔ גודל הזרימה המקסימאלית הינו $|E|$ (אנחנו רוצים לכוון את כל הקשתות).

⇐ נניח שקיים כיוון תקין של הגרף. נבנה זרימה בהתאם לכיוון הגרף: אם בגרף הקשת e_i יוצאת מהקודקוד v_j אז הקודקוד שמייצג את e_i ברשת יקבל זרימה בגודל 1 מהקודקוד שמייצג את v_j. בגרף לכל קודקוד דרגת יציאה לכל היותר 3, ולכן גם ברשת הזרימה מכל קודקוד בחלק השמאלי של הרשת תצא זרימה בגודל לכל היותר 3. מכיוון שתגיע זרימה בגודל 1 לכל קודקוד בחלק הימני של הרשת, נקבל זרימה בגודל $|E|$.

⇒ נניח שקיימת זרימה בגודל $|E|$ ונראה שקיים כיוון תקין של הגרף. נכוון כל קשת e_i כך שתצא מהקודקוד שמעביר אליה זרימה ברשת הזרימה (הוא בחר בה שתצא ממנו). מכיוון שהזרימה בגודל E, כל הקשתות יכוונו. דרגת היציאה של קודקוד בגרף זהה לזרם שעובר דרכו ברשת, כי לכל קודקוד יכולה להיכנס זרימה בגודל לכל היותר 3, לא קיים קודקוד עם דרגת יציאה גדולה מ-3.

סיבוכיות:

- יצרנו רשת זרימה בגודל הבא: $|V'| = |V| + |E| + 2$, $|E'| = |V| + 3|E|$.
- זמן הריצה: דיניץ (לא רשת 0-1) $O(|E'||V'|^2) = O(|V| + |E|)^3$, כיוון הקשתות $O(|E|)$.
- אפשר לשפר את החסם: הרצת FF $O(|E||E'|)$.
- אפשר לשפר עוד יותר: יש לנו רשת שהיא לא 0-1. נעשה רדוקציה פשוטה: נפצל כל קודקוד u לשלושה קודקודים שלכל אחד מהם נכנס קיבול 1. הוספנו פי 3 קודקודים וקשתות, אבל מדובר בקבוע. אנחנו חושבים על הפעם הראשונה ש-u בוחר קשת, על הפעם השנייה ועל הפעם השלישית. זאת באמת רשת 0-1 ונקבל: $O\left(|E'|\sqrt{|V'|}\right) = O(|V| + |E|)^{\frac{3}{2}}$.

תכנות ליניארי

רקע והגדרות

בעיה של תכנות ליניארי: נרצה להביא למקסימום/למינימום פונקציה ליניארית של n משתנים, כך שהמשתנים צריכים לקיים אוסף של אילוצים ליניאריים (שוויונות, או אי-שוויונות).
למשתנים $x_1, \dots, x_n$ מותר לקבל ערכים ממשיים, **פונקציית המטרה** (linear objective function) שלנו תהיה מיוצגת על ידי $c_1x_1 + \cdots + c_nx_n$, הכפופה **לאילוצים** ליניאריים (linear constraints):

- $a_{1,1}x_1 + a_{1,2}x + \cdots + a_{1,n}x_n \le b_1$
- $a_{2,1}x_1 + a_{2,2}x + \cdots + a_{2,n}x_n = b_2$

פתרון פיזיבילי: נקודה $(x_1, \dots, x_n) \in \mathbb{R}^n$ היא פתרון פיזיבילי של בעיית LP ⟺ היא מקיימת את כל האילוצים של הבעיה.

פתרון אופטימלי: נקודה פיזיבילית $(x_1, \dots, x_n) \in \mathbb{R}^n$ היא פתרון אופטימלי של בעיית LP ⟺ לכל נקודה פיזיבילית אחרת $(x_1', \dots, x_n') \in \mathbb{R}^n$ מתקיים $c_1x_1' + \cdots c_nx_n' \le c_1x_1 + \cdots + c_nx_n$. כלומר, הערך של הצבת הנקודה האחרת בפונקציית המטרה קטן או שווה לערך של הצבת הנקודה האופטימאלית.

$$\begin{array}{l} \max \quad c_1x_1 + c_2x_2 + \cdots + c_nx_n \\ \text{subject to:} \\ a_{1,1}x_1 + a_{1,2}x_2 + \cdots + a_{1,n}x_n \le b_1 \\ a_{2,1}x_1 + a_{2,2}x_2 + \cdots + a_{2,n}x_n \le b_2 \\ a_{3,1}x_1 + a_{3,2}x_2 + \cdots + a_{3,n}x_n \le b_3 \\ \quad \vdots \qquad\qquad \vdots \qquad\qquad \vdots \qquad \vdots \\ a_{m,1}x_1 + a_{m,2}x_2 + \cdots + a_{m,n}x_n \le b_m \\ x_1, x_2, \dots, x_n \ge 0 \end{array}$$

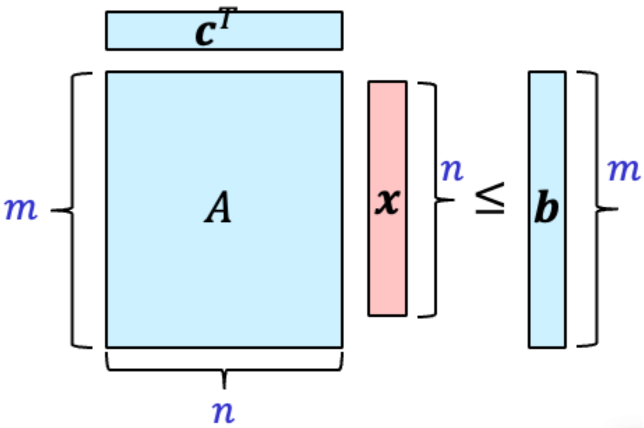
הגדרות על בעיית LP:

- **פיזיבילית**: בעיית LP היא פיזיבילית אם יש לה פתרון פיזיבילי.
- **חסומה**: בעיית LP היא חסומה אם קיים B כך שלכל פתרון פיזיבילי של הבעיה מתקיים $\sum_{i=1}^{n} c_ix_i \le B$. למשל, אם אין אילוצים בכלל, זו בעיה לא חסומה.
- **לבעיית LP יש פתרון אופטימלי** ⟺ היא פיזיבילית וחסומה.

הצורה הסטנדרטית של LP:

נסתכל רק על בעיות של **מציאת מקסימום**, נניח שכל האי-שוויונות הם מהצורה ≤. כל המשתנים הם **אי-שליליים**. יש n משתנים, m אילוצים ליניאריים (ועוד n אילוצי אי-שליליות על המשתנים). הייצוג הראשוני תופס הרבה מקום, נרצה לייצג בצורה יותר ותמציתית: נרצה להביא למקסימום את הסכום $\max \sum_{j=1}^{n} c_jx_j$, תחת האילוצים הללו: $\sum_{j=1}^{n} a_{i,j}x_j \le b_i$ עבור $i = 1, \dots, m$, $x_j \ge 0$ עבור $j = 1, \dots, n$. נשתמש ב-i עבור אינדקס של שורה ו-j עבור עמודה.

אפשר להציג את זה גם **בכתיב מטריציוני**: יש לנו מטריצה בגודל $m \times n$ עבור המקדמים של האילוצים, וקטור b עבור המקדמים של האילוצים, ועוד וקטור c^T שהוא בגודל n שאותו אנו רוצים למקסם. נכתוב: $\max c^Tx$ כך שמתקיים $Ax \le b, x \ge 0$. כאשר $x \le y$ מתקיים כאשר זה בכל קוואורדינטה: $\forall i.\, x_i \le y_i$.

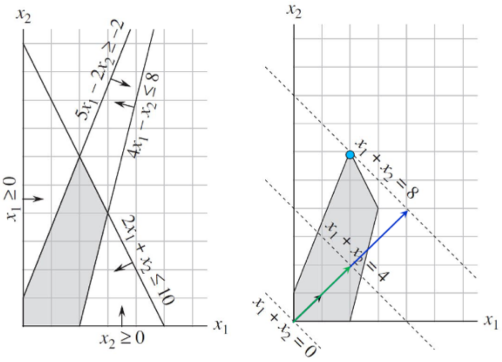


ראינו כי ניתן להמיר כל LP לצורה הסטנדרטית.

$\min \sum_{j=1}^{n} c_jx_j$	⟺	$\max \sum_{j=1}^{n} (-c_j)x_j$
$\sum_{j=1}^{n} a_{i,j}x_j \ge b_i$	⟺	$\sum_{j=1}^{n} (-a_{i,j})x_j \le -b_i$
$\sum_{j=1}^{n} a_{i,j}x_j = b_i$	⟺	$\sum_{j=1}^{n} a_{i,j}x_j \le b_i$ $\sum_{j=1}^{n} (-a_{i,j})x_j \le -b_i$
$x \lesseqgtr 0$ (x not restricted)	⟺	$x = x^+ - x^-$ $x^+, x^- \ge 0$

שימוש גנרי (כלכלי): נסתכל על ה-LP הבא: $\max \sum_{j=1}^{n} c_jx_j$ תחת האילוצים $\sum_{j=1}^{n} a_{i,j}x_j \le b_i$ עבור $i = 1, \dots, m$, $x_j \ge 0$ עבור $j = 1, \dots, n$. ניתן לכך את הפירוש הבא:

- נניח שיש m חומרי גלם, שאפשר לייצר מהם n מוצרים שונים. יש לנו b_i יחידות של חומר הגלם i, כאשר $i = 1, \dots, m$.
- כדי לייצר יחידה אחת של המוצר j נצטרך $a_{i,j}$ יחידות של חומר הגלם i.
- כל יחידה של המוצר j אפשר למכור ברווח c_j.
- נגדיר את המשתנה x_j להיות כמות היחידות של המוצר j שנייצר.
- כמה יחידות מכל מוצר לייצר, כך שהרווח שלנו יהיה הכי גדול?



<u>LP בשני מימדים – גישה גיאומטרית:</u> כל אילוץ מגדיר חצי-מישור. החיתוך של כל חצאי המישורים האלו מגדיר פוליגון, שנקרא **תחום הפיזיביליות**. פונקציית המטרה זה וקטור (וקטור המקדמים שלה). אנחנו מסתכלים על הישר שממשיך את וקטור המטרה, ואנחנו מחפשים את הנקודה בתחום הפיזיבילי שההיטל שלה על הישר הוא כמה שיותר רחוק מהראשית. תמיד יש פתרון אופטימלי שהוא קודקוד (חיתוך של הישרים).

<u>LP בצורת slack:</u> נעבור מהצורה הסטנדרטית לצורה שהאלגוריתם simplex יעבוד איתה, צורת ה-slack. לכל אילוץ נוסיף **משתנה slack**, שהוא מוגדר על ידי ההפרש. נדרוש על המשתנה הזה שיהיה אי-שלילי. היתרון בצורה הזו, היא שהאי שוויונות היחידים שצריך לעבוד איתם הם יחסית פשוטים. לרוב פשוט נתייחס אליהם כמשתנים רגילים לכל דבר: $s_i = x_{n+i}$, ויהיו לנו $n+m$ משתנים. תוכנית LP כזו תיקרא תכנית **בצורת slack**.

Standard form | **Slack form**

$$\max\ \boldsymbol{c}^T\boldsymbol{x} \quad \text{s.t.}\ A\boldsymbol{x} \le \boldsymbol{b},\ \boldsymbol{x} \ge 0$$

$$\max\ \boldsymbol{c}^T\boldsymbol{x} \quad \text{s.t.}\ \boldsymbol{s} = \boldsymbol{b} - A\boldsymbol{x},\ \boldsymbol{s},\boldsymbol{x} \ge 0$$

$$\begin{aligned} &\max\ \textstyle\sum_{j=1}^{n} c_j x_j \\ \text{s.t.}\quad &\textstyle\sum_{j=1}^{n} a_{i,j}x_j \le b_i\ ,\ i = 1,2,\dots,m \\ &x_j \ge 0 \quad ,\quad j = 1,2,\dots,n \end{aligned}$$

$$\begin{aligned} &\max\ \textstyle\sum_{j=1}^{n} c_j x_j \\ \text{s.t.}\quad &s_i = b_i - \textstyle\sum_{j=1}^{n} a_{i,j}x_j\ ,\ i = 1,2,\dots,m \\ &x_j \ge 0 \quad ,\quad j = 1,2,\dots,n \\ &s_i \ge 0 \quad ,\quad i = 1,2,\dots,m \end{aligned}$$

אלגוריתם Simplex - מבוא

<u>דוגמה:</u>

Start with an LP in standard form:

$$\begin{array}{llllllll} \max & 3x_1 & + & x_2 & + & 2x_3 & & \\ \text{s.t.} & x_1 & + & x_2 & + & 3x_3 & \le & 30 \\ & 2x_1 & + & 2x_2 & + & 5x_3 & \le & 24 \\ & 4x_1 & + & x_2 & + & 2x_3 & \le & 36 \\ & x_1 & , & x_2 & , & x_3 & \ge & 0 \end{array}$$

Convert it into slack form:

$$\begin{array}{llllllll} \max & z & = & & 3x_1 & + x_2 & + 2x_3 \\ \text{s.t.} & x_4 & = & 30 & - x_1 & - x_2 & - 3x_3 \\ & x_5 & = & 24 & - 2x_1 & - 2x_2 & - 5x_3 \\ & x_6 & = & 36 & - 4x_1 & - x_2 & - 2x_3 \\ & & & \multicolumn{4}{l}{x_1, x_2, x_3, x_4, x_5, x_6 \ge 0} \end{array}$$

$$x_6 = 36 - 4x_1 - x_2 - 2x_3$$

$$x_1 = 9 - \frac{1}{4}x_6 - \frac{1}{4}x_2 - \frac{1}{2}x_3$$

נתחיל עם בעיית LP בצורה הסטנדרטית, יש לנו 3 משתנים, 3 אילוצים. נמיר אותה לצורת slack, נוסיף 3 משתנים x_4, x_5, x_6. למשתנים שנמצאים כרגע בצד שמאל, נקרא משתנים **בסיסיים (basic)**, בהתחלה אלו משתני ה-slack. לשאר המשתנים בצד שמאל נקרא **לא-בסיסיים (non-basic)**.

ניתן לראות שיש **פתרון פיזיבילי בסיסי (bfs)** מערך $z = 0$: אם ניתן 0 לכל אחד מהמשתנים הלא-בסיסיים $(x_1, x_2, x_3) = (0,0,0)$ נקבל את ערכי המשתנים הבסיסיים $(x_4, x_5, x_6) = (30,24,36)$.

איך אפשר לשפר את הערך של הפתרון? באמצעות Pivoting step:

1. נבחר משתנה לא-בסיסי שהמקדם שלו חיובי, למשל x_1 (המקדם שלו הוא 3). עד כמה אפשר להגדיל אותו ולשמור על פיזיביליות? במשוואה הראשונה יש לנו $30 - x_1$ לכן אי אפשר להגדיל מעבר ל-30. לפי המשוואה השנייה מתקיים $24 - 2x_1$ לכן אי אפשר להגדיל מעבר ל-12.
נציב לו את הערך שהוא המינימום בין כל אלה: $x_1 = \min\left\{\frac{30}{1}, \frac{24}{2}, \frac{36}{4}\right\} = 9$.
2. למען הקיצור, לא נכתוב בצורה מפורשת את אילוצי הסימון $x_1, \dots, x_6 \ge 0$ אבל הם תמיד קיימים.
3. המשתנה הלא-בסיסי שבחרנו להגדיל אותו, יהפוך להיות בסיסי, במקום המשתנה הבסיסי שנמצא במשוואה שנתנה לנו את ערך המינימום בשלב 1. x_1 יהפוך להיות בסיסי (יכנס לבסיס), ו-x_6 יפסיק להיות בסיסי (יצא מהבסיס). נעדכן את המשוואה בהתאם, לא שינינו את האילוצים בכלל. ניקח את המשוואה החדשה ונציב במקום x_1 בשתי המשוואות האחרות של x_4, x_5 וגם בפונקציית המטרה z.

$$\begin{aligned} x_4 &= 30 - x_1 - x_2 - 3x_3 \\ &= 30 - \left(9 - \frac{1}{4}x_6 - \frac{1}{4}x_2 - \frac{1}{2}x_3\right) - x_2 - 3x_3 \\ &= 21 + \frac{1}{4}x_6 - \frac{3}{4}x_2 - \frac{5}{2}x_3 \end{aligned}$$

4. קיבלנו צורת slack חדשה: פונקציית המטרה נראית אחרת, כאשר כל האילוצים נשמרו. ה-bfs החדש שיש לנו הפעם הוא $(x_6, x_2, x_3) = (0,0,0)$, ונקבל $(x_4, x_5, x_1) = (21,6,9)$ וערך הפתרון הוא $z = 27$.

נבצע Pivoting step נוסף: נגדיל את x_3 (בעל מקדם חיובי), נקבל כי המינימום הוא $\frac{3}{2} = \frac{6}{4}$ ולכן נחליף אותו עם x_5. לאחר עדכון המשוואות נקבל צורת slack (טבלת simplex) נוספת, כאשר הפתרון קצת יותר גדול מ-27.

$$\begin{array}{llcl}
\max & z & = & 27 - \frac{3}{4}x_6 + \frac{1}{4}x_2 + \frac{1}{2}x_3 \\
\text{s.t.} & x_4 & = & 21 + \frac{1}{4}x_6 - \frac{3}{4}x_2 - \frac{5}{2}x_3 \\
 & x_5 & = & 6 + \frac{1}{2}x_6 - \frac{3}{2}x_2 - 4x_3 \\
 & x_1 & = & 9 - \frac{1}{4}x_6 - \frac{1}{4}x_2 - \frac{1}{2}x_3
\end{array}$$

$$\begin{array}{llcl}
\max & z & = & \frac{111}{4} - \frac{11}{16}x_6 + \frac{1}{16}x_2 - \frac{1}{8}x_5 \\
\text{s.t.} & x_4 & = & \frac{69}{4} - \frac{1}{16}x_6 + \frac{3}{16}x_2 + \frac{5}{8}x_5 \\
 & x_3 & = & \frac{3}{2} + \frac{1}{8}x_6 - \frac{3}{8}x_2 - \frac{1}{4}x_5 \\
 & x_1 & = & \frac{33}{4} - \frac{5}{16}x_6 - \frac{1}{16}x_2 + \frac{1}{8}x_5
\end{array}$$

נבצע Pivoting step אחרון: נגדיל את x_2 (בעל מקדם חיובי), נקבל כי המינימום הוא 4 ולכן נחליף אותו עם x_3. לאחר עדכון המשוואות נקבל צורת slack סופית, כאשר הפתרון קצת יותר גדול מ-27.

$$\begin{array}{llcl}
\max & z & = & \frac{111}{4} - \frac{11}{16}x_6 + \frac{1}{16}x_2 - \frac{1}{8}x_5 \\
\text{s.t.} & x_4 & = & \frac{69}{4} - \frac{1}{16}x_6 + \frac{3}{16}x_2 + \frac{5}{8}x_5 \\
 & x_3 & = & \frac{3}{2} + \frac{1}{8}x_6 - \frac{3}{8}x_2 - \frac{1}{4}x_5 \\
 & x_1 & = & \frac{33}{4} - \frac{5}{16}x_6 - \frac{1}{16}x_2 + \frac{1}{8}x_5
\end{array}$$

$$\begin{array}{llcl}
\max & z & = & 28 - \frac{2}{3}x_6 - \frac{1}{6}x_3 - \frac{1}{6}x_5 \\
\text{s.t.} & x_4 & = & 18 - \frac{1}{2}x_3 + \frac{1}{2}x_5 \\
 & x_2 & = & 4 + \frac{1}{3}x_6 - \frac{8}{3}x_3 - \frac{2}{3}x_5 \\
 & x_1 & = & 8 - \frac{1}{3}x_6 + \frac{1}{6}x_3 + \frac{1}{6}x_5
\end{array}$$

קיבלנו את הפתרון הפיזיבילי $(x_4, x_2, x_1) = (18,4,8)$ עם הערך $z = 28$, וזהו גם הפתרון האופטימלי של בעיית ה-LP (הוא גם בסיסי כי $(x_6, x_3, x_5) = (0,0,0)$). **כל המקדמים של המשתנים הלא-בסיסיים הם שליליים**, לכן לא נוכל לשפר עוד.

אם אנחנו רוצים רק את הפתרון של הבעיה המקורית נסתכל על $(x_1, x_2, x_3) = (8,4,0)$, ומשתני ה-slack הם $(x_4, x_5, x_6) = (18,0,0)$ כלומר האילוץ הראשון לא התקיים עם שוויון, יהיה הפרש של 18 שם. האילוצים השני והשלישי התקיימו עם שוויון.

האלגוריתם:

1. נתחיל מבעיה בצורה **סטנדרטית**. נהפוך אותה לבעיה בצורת **slack** (על ידי הוספה של משתני slack), כך שנוכל למצוא בקלות **פתרון פיזיבילי בסיסי** (bfs).
   a. אם כל ה-b_i חיוביים זה פשוט ונוסיף משתני slack, אחרת נצטרך לעבוד קצת יותר (auxiliary problem).
   b. אם אי אפשר למצוא דרך להעביר לצורת slack עם bfs, **הבעיה לא פיזיבילית**.
2. נבחר משתנה **לא-בסיסי** x_e (enter) עם **מקדם חיובי** בפונקציית המטרה (שהולך להיכנס לבסיס). נמצא משתנה **בסיס** x_ℓ (leave) שנותן את החסם העליון הכי קטן על הגדלת x_e. נחליף את x_ℓ עם x_e (**pivot step**).
   a. אם יש כמה משתנים לא-בסיסיים עם מקדמים חיוביים, אפשר לבחור כל אחד, זה נעשה על ידי **pivoting rule**.
   b. אם אף אחד מהמשתנים הבסיסיים לא נותן חסם להגדלת x_e, **הבעיה לא חסומה**.
   c. יכול להיות שלקחנו את x_e ואפשר להגדיל אותו במקסימום 0 – זה נקרא צעד מנוון (**degenerate step**). הבסיס השתנה אבל ערך הפתרון לא השתנה. צריך להימנע מצעדים מנוונים כאלה.
   d. נחזור על 2.
3. אם כל המקדמים בפונקציית המטרה הם שליליים, אז ה-bfs הנוכחי הוא **אופטימלי**.

סימונים:

- B – קבוצת המשתנים הבסיסיים: $B = \{n+1, \dots, n+m\}$.
- N – קבוצת המשתנים הלא-בסיסיים: $N = \{1, \dots, n\}$.
- simplex tableau: **פונקציית המטרה מורכבת מערך הפתרון הבסיסי v, וצ"ל של המשתנים הלא-בסיסיים ($j \in N$)**. יש לנו משוואה ליניארית עבור כל משתנה בבסיס ($i \in B$). יש כאן מינוס כי זה מה שקורה כשממירים מהצורה הסטנדרטית.
- פתרון בסיסי הוא $\bar{x}_j = b_j \ if \ j \in B$, אחרת $\bar{x}_j = 0 \ if \ j \in N$. נשים לב שיש שני פירושים ל-x_j: משתנה שמופיע בהגדרה הפורמלית של הבעיה, מצד שני $\bar{x}_j$ זה הערך של המשתנה בפתרון ספציפי שאנחנו מסתכלים עליו. פתרון בסיסי הוא פיזיבילי (bfs) $\Leftrightarrow \forall i. b_i \geq 0$.

$$\boxed{\begin{array}{l} z = v + \sum_{j \in N} c_j x_j \\ x_i = b_i - \sum_{j \in N} a_{i,j} x_j \quad , \quad i \in B \end{array}}$$

- pivoting step – בחירת המשתנים שיכנסו ויצאו מהבסיס. נבחר משתנה עם מקדם חיובי $c_e > 0$. עבור כל משוואה אחרת שבה $a_{i,e} > 0$, יש מגבלה על הגודל שניתן להוסיף ל-x_e. על כן, נבחר את האינדקס שעבורו נשיג את החסם המינימלי על המשתנה x_e.

$$\boxed{\begin{aligned} z &= v + c_e x_e + \textstyle\sum_{j\in N\setminus\{e\}} c_j x_j \\ x_i &= b_i - a_{i,e}x_e - \textstyle\sum_{j\in N\setminus\{e\}} a_{i,j}x_j, \quad i \in B \end{aligned}}$$

$$\ell = \operatorname{argmin}_{i\in B}\left\{\frac{b_i}{a_{i.e}} \,\middle|\, a_{i,e} > 0\right\}$$

- עדכון הטבלה: **המשתנה x_e הופך לבסיסי, והמשתנה x_ℓ הופך ללא-בסיסי**. נעדכן את המשוואה עבור x_e שנכנס עכשיו לבסיס באמצעות x_ℓ (נעביר אגפים ונחלק) ונקבל עבורו מקדמים חדשים. לאחר מכן נעדכן את כל שאר השורות עבור משתנים שאינם x_e: מציבים את הביטוי החדש של x_e בכל משוואה כזאת.

$$x_\ell = b_\ell - a_{\ell,e}x_e - \textstyle\sum_{j\in N\setminus\{e\}} a_{\ell,j}x_j$$

$$x_e = \underbrace{\frac{b_\ell}{a_{\ell,e}}}_{b'_e} - \underbrace{\frac{1}{a_{\ell,e}}}_{a'_{e,\ell}}x_\ell - \sum_{j\in N\setminus\{e\}}\underbrace{\frac{a_{\ell,j}}{a_{\ell,e}}}_{a'_{e,j}}x_j$$

These are the coefficients of the new row corresponding to x_e

Next, we update the row of x_i, for $i \neq e$:
(This is essentially an elementary row operation on A.)

$$x_i = b_i - a_{i,e}\left(b'_e - a'_{e,\ell}x_\ell - \textstyle\sum_{j\in N\setminus\{e\}} a'_{e,j}x_j\right) - \textstyle\sum_{j\in N\setminus\{e\}} a_{i,j}x_j$$

$$x_i = \underbrace{\left(b_i - a_{i,e}b'_e\right)}_{b'_i} + \underbrace{a_{i,e}a'_{e,\ell}}_{a'_{i,\ell}}x_\ell - \sum_{j\in N\setminus\{e\}}\underbrace{\left(a_{i,j} - a_{i,e}a'_{e,j}\right)}_{a'_{i,j}}x_j$$

- לבסוף, נעדכן את פונקציית המטרה שלנו עם הצבת x_e:

$$z = v + c_e\left(b'_e - a'_{e,\ell}x_\ell - \textstyle\sum_{j\in N\setminus\{e\}} a'_{e,j}x_j\right) + \textstyle\sum_{j\in N\setminus\{e\}} c_j x_j$$

$$z = \underbrace{\left(v + c_e b'_e\right)}_{v'} - \underbrace{c_e a'_{e,\ell}}_{a'_{i,\ell}}x_\ell - \sum_{j\in N\setminus\{e\}}\underbrace{\left(c_j - c_e a'_{e,j}\right)}_{a'_{i,j}}x_j$$

אלגוריתם Simplex - ניתוח

<u>סיום (Termination):</u>

האם האלגוריתם תמיד יעצור? אם לא ניתקל אף פעם בצעד מנוון (בכל צעד הערך גדל), התהליך חייב לעצור. האם יכול להיות שנתכנס לפתרון האופטימלי אבל אף פעם לא נגיע אליו? יש מספר סופי של בסיסים. בסיס הוא בחירה של m משתנים מבין $n+m$ משתנים כלליים: $\binom{n+m}{m}$. אם הערך כל הזמן גדל, לא נוכל לחזור לאותו בסיס, ולכן נעצור אחרי מספר סופי של צעדים.

- יכול להיות שיהיו צעדים מנוונים ואז פונקציית המטרה לא תשתנה. למשל אחרי ה-pivot השני, הפתרון שלנו נשאר 8 ולא גדל. הכנסנו את x_3 לבסיס, המקדם לא מופיע ולכן אסור לנו להגדיל אותו בשום דבר. הצעד שלאחריו הוא כבר לא מנוון.

Recall that $v' = v + c_e b'_e = v + c_e(b_\ell / a_{\ell,e})$

Thus, if $b_\ell = 0$, then $v' = v$, and we get a *degenerate* iteration.

Example of degeneracy:

$$\begin{aligned} z &= x_1 + x_2 + x_3 \\ x_4 &= 8 - x_1 - x_2 \\ x_5 &= x_2 - x_3 \end{aligned} \qquad \begin{aligned} z &= 8 + x_3 - x_4 \\ x_1 &= 8 - x_2 - x_4 \\ x_5 &= x_2 - x_3 \end{aligned}$$

degenerate step

$$\begin{aligned} z &= 8 + x_2 - x_4 - x_5 \\ x_1 &= 8 - x_2 - x_4 \\ x_3 &= x_2 - x_5 \end{aligned} \qquad \begin{aligned} z &= 16 - x_1 - 2x_4 - x_5 \\ x_2 &= 8 - x_1 - x_4 \\ x_3 &= 8 - x_1 - x_4 - x_5 \end{aligned}$$

- יכול להיות שניקלע למעגלים: רצף של צעדים מנוונים שחוזרים ל-bfs שכבר ביקרנו בו. אפשר לשנות טיפה את המקדמים של הבעיה, ואז הסיכוי שיהיה צעד מנוון הוא קטן. פתרון טוב יותר הוא להשתמש ב**כלל Bland**: אם יש מספר משתנים לא-בסיסיים אם מקדמים לא חיוביים, נבחר את המשתנה בעל האינדקס הכי קטן. **אם נשתמש בכלל Bland לא יהיה מעגל באלגוריתם ה-simplex.**

<u>כללי pivoting נפוצים:</u>

- Dantzig – נבחר e עבורו c_e מקסימלי.
- השיפור הגדול ביותר – נבחר את e שמגדיל את פונקציית המטרה בצורה הגדולה ביותר.
- Zadeh – נבחר e עם מקדם $c_e > 0$ שהכנסנו אותו לבסיס B מספר קטן של פעמים עד כה.
- Random – נבחר משתנה מקרי.

עבור כל הכללים הללו זמן הריצה של simplex **הוא לא פולינומי ב-WC**.

<u>סיבוכיות:</u>

- כל צורת slack מכילה n משתנים לא בסיסיים ו-m משתנים בסיסיים. חסם עליון טריוויאלי הוא $\binom{m+n}{m} < 2^{(m+n)}$, זה מספר הדרכים לבחור את קבוצת המשתנים הבסיסיים של צורת ה-slack.
- סיבוכיות צעד פיבוט: $O(nm)$: בחירת משתנה לא בסיסי, מציאת המשתנה הבסיסי המתאים, החלפה ביניהם באילוץ המתאים ובשאר האילוצים ובפונקציית המטרה.
- מספר הצעדים שהאלגוריתם מבצע: $O(2^{m+n})$.
- סה"כ $\boldsymbol{O(nm \cdot 2^{m+n})}$.

<u>מציאת bfs (באמצעות תכנית העזר):</u>

אם לאחר המרה לצורת slack מתקיים $b < 0$, אז ה-LP אולי לא פיזיבילי. כדי לבדוק זאת, נציג בעיית עזר (auxiliary LP):

- נציג משתנה חדש x_0 ודרשנו שהוא יהיה אי-שלילי.
- אנחנו מתעלמים מפונקציית המטרה המקורית (רק האילוצים משפיעים על פיזיביליות), ומגדירים את הפונקציה החדשה $-x_0$. חישוב $\max -x_0$ שקול לחישוב $\min x_0$.
- מפחיתים את x_0 מכל אחד מהאילוצים: $Ax - x_0 \le b$.

$$\begin{aligned} &\max\ \boldsymbol{c}^T\boldsymbol{x} \\ &\text{s.t.}\ \ \boldsymbol{Ax} \le \boldsymbol{b} \\ &\qquad \boldsymbol{x} \ge 0 \end{aligned}$$

Original LP

$$\begin{aligned} &\max\ -x_0 \\ &\text{s.t.}\ \ \boldsymbol{Ax} - x_0 \le \boldsymbol{b} \\ &\qquad \boldsymbol{x} \ge 0\ ,\ x_0 \ge 0 \end{aligned}$$

Auxiliary LP

בעיית העזר תמיד פיזיבילית, כי אם נבחר את x_0 מספיק גדול אי השוויונות יתקיימו. $x_0 = -min\ b_i$ ו-$x = 0$. כדי להחליט האם הבעיה המקורית פיזיבילית נחליט אם **הפתרון של בעיית העזר הוא 0**. המצב הכי טוב הוא $x_0 = 0$. זה אומר שכל האילוצים מתקיימים באופן המקורי $Ax \le b$.

נריץ simplex על בעיית העזר. ראשית צריך להמיר את הבעיה לצורת slack. גם אם $b < 0$, נוכל למצא bfs.

$$\begin{aligned} &\max\ -x_0 \\ &\text{s.t.}\ \ \boldsymbol{Ax} - x_0 \le \boldsymbol{b} \\ &\qquad \boldsymbol{x} \ge 0\ ,\ x_0 \ge 0 \end{aligned} \quad\Longrightarrow\quad \begin{aligned} &\max\ -x_0 \\ &\text{s.t.}\ \ \boldsymbol{s} = \boldsymbol{b} + x_0 - \boldsymbol{Ax} \\ &\boldsymbol{s} \ge 0\ ,\ \boldsymbol{x} \ge 0\ ,\ x_0 \ge 0 \end{aligned}$$

- תזכורת: את s_i מסמנים גם ב-x_{n+i}. נניח כי $b < 0$, יש לו איבר שלילי. נסמן ב-k את האינדקס של ה-b_i הכי קטן: $\forall i. b_k \le b_i$. נעבוד עם האיבר x_{n+k}.
- נעשה את צעד ה-pivot המבוקר שבו אנחנו מחליפים את x_0 ב-x_{n+k} וזה יפתור לנו את הבעיה. נסתכל על המשוואה ה-k-ית. **נהפוך את x_0 למשתנה בסיסי ואת x_{n+k} למשתנה לא-בסיסי**. כבר יש לנו את $-b_k$ **וזה חיובי**.
- עכשיו צריך לקחת את כל השורות האחרות ולהביע אותן באמצעות המשתנים הלא-בסיסיים החדשים אחרי העדכון. אנחנו שמחים כי $b_i - b_k \ge 0$ וקיבלנו **פתרון פיזיבילי**. יש לנו bfs התחלתי לבעיית העזר ואפשר להתחיל לפתור אותה.
- בעיית העזר חסומה (פונקציית המטרה היא לכל היותר 0). לכן אלגוריתם simplex יחזיר פתרון אופטימלי. אם x_0 בפתרון האופטימלי של בעיית העזר הוא 0, אם נשכח אותו נישאר עם **פתרון פיזיבילי לבעיה המקורית**.

כדי לפתור LP מההתחלה ראינו כי אנחנו צריכים להריץ simplex פעמיים: פעם אחת כדי לפתור את בעיית העזר, ואז פעם נוספת כדי לפתור את הבעיה המקורית. יש הרבה מקרים שבהם קל לנו לקבל פתרון פיזיבילי התחלתי ואז אפשר לחסוך את בעיית העזר.

<u>פתרונות בסיסיים אופטימליים:</u>

- <u>טענה</u>: לכל בעיית LP בצורת slack שהיא פיזיבילית וחסומה, יש **פתרון אופטימלי שהוא בסיסי**. בפרט יש לכל היותר m משתנים שאינם 0 (לכל היותר m אילוצים אינם הדוקים).
  - כיוון שאלגוריתם ה-simplex עובד רק עם פתרונות בסיסיים, וכשהוא עוצר אנחנו יודעים שהפתרון הוא בסיסי, זה אומר בפרט שלכל בעיה כזו יש פתרון אופטימלי שהוא בסיסי.
- <u>טענה</u>: לכל בעיית LP בצורה הסטנדרטית שהיא פיזיבילית וחסומה, יש **פתרון אופטימלי שבו לפחות n מהאילוצים מתקיימים עם שוויון**.

תכנות לינארי 1 (תרגול 12)

הצגה גרפית:

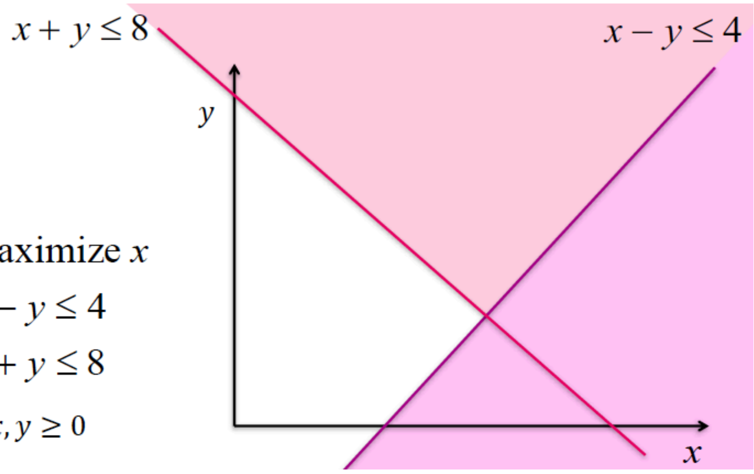
כל pivoting step באלגוריתם ה-simplex מקדם אותנו לקודקוד הבא שמקרב אותנו לפתרון האופטימלי. בנינו את צורת ה-slack:

$$s_1 = 4 - x + y$$
$$s_2 = 8 - x + y$$

התחלנו ב-$(0,0)$, ואחרי הצעד הראשון הגענו ל-$(4,0)$, כאשר בחרנו את ההפרש המינימלי 4 ולכן x הופך להיות בסיסי, ו-s_1 הופך ללא-בסיסי.

לאחר הצעד הנוסף נקבל $(6,2)$, וערך פונקציית המטרה הוא 6.

אם היינו לוקחים משתנה עם מקדם 0, היינו עושים **צעד מנוון**.



$$\text{maximize } x$$
$$x - y \leq 4$$
$$x + y \leq 8$$
$$x, y \geq 0$$

תרגיל 3: יהיו $G = (V, E)$ גרף לא מכוון, תארו אלגוריתם מבוסס LP שיחשב את הערך המירבי T, כך **שלכל** בחירת משקלים אי-שליליים על הקשתות המקיימים $\sum_{e \in E} w(e) = 1$, **יש** צומת $v \in V$ המקיים $\sum_{e: v \in e} w(e) \geq T$.

יש בעיה עם דרישת "לכל" בחירת משקלים, ואז "יש" צומת כלשהו. זה הפוך לוגית ממה שאנחנו רוצים: ה-T שאנחנו מחפשים הוא המקסימום על פני כל ההבטחות t: לכל הצבה שלא נביא, קיים קודקוד כך שהסכום של מה שנוגע בו הוא לפחות t:

$$T = \max \{t | \forall w(e). \sum w(e) = 1, \quad \exists v \in V \sum_{e: v \in e} w(e) \geq T\}$$

מהי השלילה? כל ה-t-ים כך **שקיימת הצבה אחת** (דוגמה חוסמת מלמעלה את t), **שלכל קודקוד** הסכום קטן מ-t (מעדיפים קטן שווה).

$$T^* = \min \{t | \exists w(e). \sum w(e) = 1, \quad \forall v \in V \sum_{e: v \in e} w(e) \leq T\}$$

$$\min T$$
$$\forall v \in V, \sum_{e: v \in e} w(e) \leq T$$
$$\sum_{e \in E} w(e) = 1$$
$$\forall e \in E, w(e) \geq 0$$

כלומר: ההבטחה הכי גבוהה שאפשר להשיג, היא שווה לחסם התחתון הכי טוב שאפשר להשיג על ידי הצבה קונקרטית. תמצאו לי את ההצבה הכי גבוהה שבה המקסימום על הקודקודים הכי קטן שאפשר, וזה T.

נוכיח את נכונות השוויון בין שני הפתרונות: יהי פתרון אופטימלי לבעיית המינימום, T^*, w^*, כאשר T פתרון לבעיית המקסימום:

- הוא מקיים $T \geq T^*$: נניח בשלילה כי $T < T^*$. מהגדרת התוכנית, לכל צומת $v \in V$ מתקיים $\sum_{e: v \in e} w^*(e) \leq T^*$, ואם זה קטן מ-T, זה סתירה לכך שמדובר ב-T חוקי. אין קודקוד שסכום הקשתות שלו גדול שווה T.
- הוא מקיים $T \leq T^*$: נניח בשלילה כי $T^* > T$. אז בפרט עבור $T^* > T' = \frac{T^* + T}{2} > T$ שנמצא בדיוק ביניהם. לקחנו $T' > T$ ולכן קיימת פונקציה w' עבורה אין צומת שמקיים $\sum_{e: v \in e} w'(e) \geq T'$. כלומר, לכל צומת v מתקיים $\sum_{e: v \in e} w'(e) < T'$. לכן קיבלנו כי T', w' פתרון פיזיבילי, בסתירה למינימליות של T^*.

הבעיה הדואלית

<u>הגדרת התוכנית הדואלית:</u>

לכל **תוכנית LP פריימלית (primal)** בצורה הסטנדרטית, נתאים **תכנית דואלית (dual)**.

primal	$\max \sum_{j=1}^{n} c_j x_j$ s.t. $\sum_{j=1}^{n} a_{i,j} x_j \le b_i \ , \ i = 1,2,\dots,m$ $x_j \ge 0 \ , \ j = 1,2,\dots,n$	$\max \boldsymbol{c}^T \boldsymbol{x}$ s.t. $A\boldsymbol{x} \le \boldsymbol{b}$ $\boldsymbol{x} \ge 0$
dual	$\min \sum_{i=1}^{m} b_i y_i$ s.t. $\sum_{i=1}^{m} a_{i,j} y_i \ge c_j \ , \ j = 1,2,\dots,n$ $y_i \ge 0 \ , \ i = 1,2,\dots,m$	$\min \boldsymbol{b}^T \boldsymbol{y}$ s.t. $A^T\boldsymbol{y} \ge \boldsymbol{c}$ $\boldsymbol{y} \ge 0$

- לכל אילוץ בתוכנית הפריימלית הוספנו משתנה (dual variable) y_i.
- לכל משתנה בתוכנית הפריימלית הגדרנו אילוץ בתוכנית הדואלית.
- הוקטורים b ו-c החליפו תפקידים. המטריצה A עברה transpose. אי השוויון שינה כיוון.
- בעיית מקסימום הופכת לבעיית מינימום.

<u>משפט הדואליות (LP duality theorem):</u> אם שתי התוכניות הפריימלית והדואלית הן פיזיביליות וחסומות, אז הערך האופטימלי שלהן שווה. נניח ש-$\bar{x}$ הוא פתרון פיזיבילי של הפריימלית, ו-$\bar{y}$ הוא פתרון פיזיבילי של הדואלית, אז הערך של $\bar{x}$ בבעיה הפריימלית קטן שווה מהערך של $\bar{y}$ בבעיה הדואלית: $\sum_{j=1}^{n} c_j \bar{x}_j \le \sum_{i=1}^{m} b_i y_i$. בנוסף, שניהם אופטימליים $\Leftrightarrow$ יש שוויון ביניהם.

איך מקבלים את הבעיה הדואלית? יש לנו את הבעיה הפריימלית, ואנחנו רוצים לקבל חסם עליון על הערך האופטימלי שלה. ניקח צ"ל של m האילוצים יחד עם המקדמים הבאים: $y_1, \dots, y_m$. ניקח את האילוץ ה-i, נכפיל ב-y_i ונסכם.

$$\sum_{i=1}^{m}\Big(\sum_{j=1}^{n} a_{i,j} x_j\Big) y_i = \sum_{j=1}^{n}\Big(\sum_{i=1}^{m} a_{i,j} y_i\Big) x_j \ \le \ \boxed{\sum_{i=1}^{m} b_i y_i}$$

If $\boxed{\sum_{i=1}^{m} a_{i,j} y_i \ge c_j}$, for every j, then for every feasible $\boldsymbol{x}$:

$$\sum_{j=1}^{n} c_j x_j \le \sum_{j=1}^{n}\Big(\sum_{i=1}^{m} a_{i,j} y_i\Big) x_j \ \le \ \sum_{i=1}^{m} b_i y_i$$

<u>דואליות חלשה (weak duality):</u> טענה יותר חלשה שמראה רק צד אחד של אי השוויון.

If $\bar{\boldsymbol{x}}$ is a feasible solution of the primal,
and $\bar{\boldsymbol{y}}$ is a feasible solution of the dual, then

$$\sum_{j=1}^{n} c_j \bar{x}_j \le \sum_{i=1}^{m} b_i \bar{y}_i$$

אפשר לקבל אותה גם דרך הכתיב המטריציוני:

$$\text{Suppose } \underbrace{A\bar{\boldsymbol{x}} \le \boldsymbol{b} \ , \ \bar{\boldsymbol{x}} \ge 0}_{\bar{\boldsymbol{x}} \text{ feasible in primal}} \ \text{ and } \ \underbrace{A^T\bar{\boldsymbol{y}} \ge \boldsymbol{c} \ , \bar{\boldsymbol{y}} \ge 0}_{\bar{\boldsymbol{y}} \text{ feasible in dual}}$$

$$\boldsymbol{c}^T\bar{\boldsymbol{x}} \ \le \ (A^T\bar{\boldsymbol{y}})^T\bar{\boldsymbol{x}} \ = \ \bar{\boldsymbol{y}}^T(A\bar{\boldsymbol{x}}) \ \le \ \bar{\boldsymbol{y}}^T\boldsymbol{b} \ = \ \boldsymbol{b}^T\bar{\boldsymbol{y}}$$

פירוש בלבלי:

התוכנית הפריימלית: כמה יחידות x_j מכל מוצר j לייצר, כך שהרווח שלנו יהיה הכי גדול?

- נניח שיש m חומרי גלם, שאפשר לייצר מהם n מוצרים שונים. יש לנו b_i יחידות של חומר הגלם i, כאשר $i = 1, \ldots, m$.
- כדי לייצר יחידה אחת של המוצר j נצטרך $a_{i,j}$ יחידות של חומר הגלם i.
- כל יחידה של המוצר j אפשר למכור ברווח c_j.
- אי השוויון $\sum_{j=1}^{n} a_{i,j}x_j \le b_i$ מוודא שהכמות של חומר הגלם ה-i היא לא יותר מהכמות שיש לנו כרגע ביד.

התוכנית הדואלית: מה ההצעה שהעלות שלה מינימלית שבה לא תהיה לנו עוד סיבה מיידית לדחות את ההצעה?

במקום לייצר את המוצרים, ננסה למכור את חומרי הגלם שיש לנו (הצעה לקנות מאיתנו את חומרי הגלם, כאשר הרצון הוא כמובן לקנות אותם בעלות המינימלית).

- ההצעה היא y_i לחומר גלם i עבור $i = 1, \ldots, m$. סה"כ $\sum_{i=1}^{n} b_i y_i$ לכל חומרי הגלם.
- אבל, לא נרצה שיתקיים $\sum_{i=1}^{n} a_{i,j}y_i < c_j$ עבור j כלשהו, כי עדיף יהיה לנו לייצר את המוצר j בעצמנו, נקבל יותר ממה שהוא מציע לנו על חומרי הגלם שצריך להשתמש בהם.

הוכחת משפט הדואליות (segment 8) – מיותר.

הגרסה "המלאה" של משפט הדואליות:

(1) אם P פיזיבילית וחסומה, אז גם D פיזיבילית וחסומה, והערכים האופטימליים שלהן שווים.
(2) אם P פיזיבילית אבל לא חסומה, אז D לא פיזיבילית.
(3) אם D פיזיבילית אבל לא חסומה, אז P לא פיזיבילית.

זוגות פריימלית/דואלית נוספים:

- אם יש אי-שוויונות בתוכנית הפריימלית, המשתנים בדואלית צריכים להיות אי-שליליים.
- אם יש שוויונות, אין אילוצי סימון על המשתנים בתוכנית הדואלית.

Standard form	Equational form	Inequalities form
$\max\ \mathbf{c}^T\mathbf{x}$ s.t. $A\mathbf{x} \le \mathbf{b}$ $\mathbf{x} \ge 0$	$\max\ \mathbf{c}^T\mathbf{x}$ s.t. $A\mathbf{x} = \mathbf{b}$ $\mathbf{x} \ge 0$	$\max\ \mathbf{c}^T\mathbf{x}$ s.t. $A\mathbf{x} \le \mathbf{b}$
$\min\ \mathbf{b}^T\mathbf{y}$ s.t. $A^T\mathbf{y} \ge \mathbf{c}$ $\mathbf{y} \ge 0$	$\min\ \mathbf{b}^T\mathbf{y}$ s.t. $A^T\mathbf{y} \ge \mathbf{c}$	$\min\ \mathbf{b}^T\mathbf{y}$ s.t. $A^T\mathbf{y} = \mathbf{c}$ $\mathbf{y} \ge 0$

הלמה של פרקש – מיותר.

מתכון לדואליות:

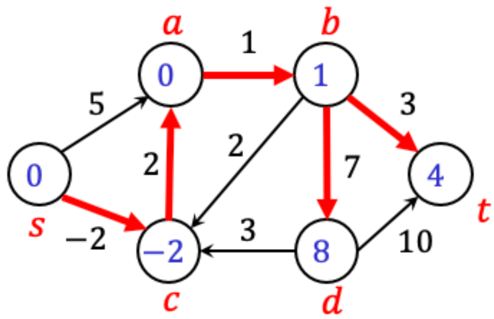
	Max problem (P)	Min problem (D)
variables	$x_1, x_2, \ldots, x_n$	$y_1, y_2, \ldots, y_m$
matrix	A	A^T
constants	$\mathbf{b}$	$\mathbf{c}$
objective	$\max\ \mathbf{c}^T\mathbf{x}$	$\min\ \mathbf{b}^T\mathbf{y}$
i-constraint in (P) $i = 1,2,\ldots,m$	*i*-th row of *A* $a_{i,*}x \le b_i$ $a_{i,*}x = b_i$ $a_{i,*}x \ge b_i$	$y_i \ge 0$ y_i unrestricted $y_i \le 0$
j-constraint in (D) $j = 1,2,\ldots,n$	$x_j \ge 0$ x_j unrestricted $x_j \le 0$	*j*-th column of *A*, as a row vector $a_{*,j}^T y \ge c_j$ $a_{*,j}^T y = c_j$ $a_{*,j}^T y \le c_j$

ההבדל בין תכנות לינארי לתכנות בשלמים (segment 10) – מיותר.

איך מתבון הדואליות עובד? נמיר מצורה לא סטנדרטית לסטנדרטית, נמיר לדואלית, ונחליף את $y_1 - y_2$ במשתנה בודד y:

$$\begin{cases} \max \sum c_i x_i \\ \sum e_i x_i = b \\ x_i \geq 0 \end{cases} \Longleftarrow \begin{matrix} \max \sum c_i x_i \\ \sum e_i x_i \leq b \\ \sum -e_i x_i \leq -b \\ x_i \geq 0 \end{matrix} \overset{\text{Duel}}{\Longleftarrow} \begin{matrix} \min b(y_1 - y_2) \\ e_i(y_1 - y_2) \geq c_i \\ y_1, y_2 \geq 0 \end{matrix} \Longleftrightarrow \begin{matrix} \min by \\ e_i y \geq c_i \end{matrix}$$

בעיית SSSP כ-LP



<u>תזכורת:</u> יש לנו גרף מכוון וממושקל ללא מעגלים שליליים, עם מקור s ובור t. אנו **רוצים למצוא מק"בים מ-s**. נגדיר $\delta(u) = \delta(s, u)$ כאורך המק"ב מ-s ל-u בגרף. נניח שכל הצמתים נגישים מ-s. הקשתות האדומות הן חלק מעץ המק"בים.

<u>למה:</u> נניח שלכל צומת u יש לנו ערך y_u.
(1) אם $y_s = 0$ וגם $y_v \leq y_u + w(u, v)$ לכל $(u, v) \in E$, אז $y_u \leq \delta(u)$, כלומר הערך הזה הוא חסם תחתון למרחק מ-s ל-u.
(2) אם בנוסף לכל צומת v שאינו s שעבורה מתקיים השוויון, הערכים הם האורכים עצמם (הקשתות הדוקות), כלומר $y_u = \delta(u)$.

<u>בעיית SSSP כ-LP:</u> נרצה להביא למקסימום את סכום של כל ערכי ה-y, תחת האילוץ שעבור כל קשת $y_v - y_u \leq w(u, v)$.

- אנחנו רוצים למצוא מסלולים קצרים ביותר, למה זו בעיית מקסימום? הסיבה היא שכל אוסף ערכי y הם חסמים תחתונים על המרחקים, לכן נרצה למצוא את החסמים הכי גדולים.
- הצגת הבעיה בבעיית מינימום: אפשר לכתוב את הבעיה הדואלית (זה מתקשר גם לבעיות זרימה).

$$\text{(LP)} \qquad \begin{array}{ll} \max & \sum_{u \in V} y_u \\ \text{s.t.} & y_v - y_u \leq w(u, v)\ ,\ \ (u, v) \in E \\ & y_s = 0 \end{array}$$

<u>טענה:</u> אם $\bar{y}$ פתרון אופטימלי לבעיית ה-LP, אז $\bar{y}_u = \delta(u)$.

הוכחה: הבעיה פיזיבילית כי $\bar{y}_u = \delta(u)$ הוא פתרון פיזיבילי. הבעיה חסומה כי לפי הלמה, $\bar{y}_u \leq \delta(u)$. לכן, לבעיה יש פתרון אופטימלי $\bar{y}$. נוכיח שלכל צומת v שאינו s קיימת קשת $(u, v) \in E$ נכנסת כך שהיא הדוקה: $\bar{y}_v = \bar{y}_u + w(u, v)$ ולפי למה 2 נקבל שוויון $\bar{y}_u = \delta(u)$. אם כן, נניח בשלילה כי קיים צומת v שעבור כל הקשתות הנכנסות אליו מתקיים: $\bar{y}_v < \bar{y}_u + w(u, v)$. לכן, זה לא הפתרון האופטימלי, אפשר להגדיל טיפה את y_v כך שכל האילוצים ימשיכו להתקיים, כלומר שיפרנו את הפתרון.

- מה קורה אם ב-G יש מעגלים שליליים? הבעיה לא פיזיבילית.
- מה קורה אם יש צומת שאינו נגיש מ-s? הבעיה לא חסומה (אפשר להגדיל את הערך של אותו צומת כרצוננו).

<u>הצגה כבעיה פיזיקלית:</u>

נרצה להביא למקסימום את y_t, למצוא את המק"ב מ-s ל-t בלבד. לכל קשת נבין חתיכת חבל שהאורך שלה בדיוק $w(u, v)$. נבנה מכל החתיכות האלה מודל פיזיקלי של הגרף. מחזיקים ביד אחת את s ובשנייה את t ומושכים עד שאי אפשר יותר. ברגע שעצרנו המרחק הגאומטרי בין s ל-t זה המרחק בין הצמתים בגרף.

בעיות זרימה כ-LP

<u>בעיית זרימה כ-LP:</u> נרצה **להביא למקסימום את גודל הזרימה**: כמה נכנס לבור (אפשר להגדיר באופן שקול כמה יוצא מהמקור). הגורם הראשון מתאר את כל הזרימה הנכנסת לבור, והגורם השני מתאר את כל הזרימה שיוצאת מהבור.

- לכל קשת ערך הזרימה הוא $f_{u,v}$ והוא חייב להיות אי-שלילי.
- יש לנו את אילוץ הצומת/שימור: מה שנכנס זה מה שיוצא.
- יש לנו את אילוץ הקשת/קיבול: לא ניתן להזרים יותר מהקיבול.

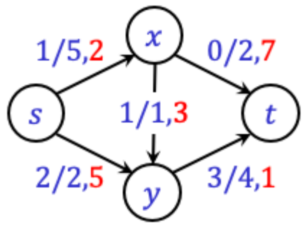
$$\begin{array}{ll} \max & \sum_{u:(u,t) \in E} f_{u,t} - \sum_{w:(t,w) \in E} f_{t,w} \\ \text{s.t.} & \sum_{u:(u,v) \in E} f_{u,v} - \sum_{w:(v,w) \in E} f_{v,w} = 0\ ,\ \ v \in V \setminus \{s, t\} \\ & f_{u,v} \leq c_{u,v}\ ,\ (u, v) \in E \\ & f_{u,v} \geq 0\ ,\ (u, v) \in E \end{array}$$

נשים לב כי הבעיה תמיד פיזיבילית – 0 תמיד פתרון. בנוסף, הבעיה תמיד חסומה – יש קיבול מקסימלי לכל קשת. לכן לבעיה יש פתרון אופטימלי.

וריאציה 1 - Minimum cost flow:

לכל קשת יש גם **עלות** בנוסף לקיבול, לפי הפונקציה $a: E \to \mathbb{R}$. **הזרמת זרימה בקשת e עולה** $\boldsymbol{a(e) \cdot f(e)}$, כלומר משלמים על כל יחידת זרימה שמעבירים. נרצה לשלוח קלט נוסף לבעיה d (demand), של כמות יחידות הזרם שנרצה להזרים מ-s ל-t **בעלות כמה שיותר נמוכה**. כשנעביר יחידת זרימה נרצה למצוא את המק"ב מבחינת העלות a של הקשתות (כמו משקל על הקשתות).

- נעביר תחילה יחידה 1 על המסלול s→x→y→t.
- נעביר 2 יחידות נוספות על המסלול s→y→t.
- נעביר יחידה אחרונה על המסלול s→x→t.



Sending 3 units at the cost of :
$1 \cdot 2 + 2 \cdot 5 + 1 \cdot 3 + 3 \cdot 1 = 18$

$$\min \ \sum_{(u,v)\in E} a_{u,v} f_{u,v}$$
$$\text{s.t.} \ \sum_{u:(u,t)\in E} f_{u,t} - \sum_{w:(t,w)\in E} f_{t,w} = d$$
$$\sum_{u:(u,v)\in E} f_{u,v} - \sum_{w:(v,w)\in E} f_{v,w} = 0 \ , \ v \in V \setminus \{s,t\}$$
$$f_{u,v} \le c_{u,v} \ , (u,v) \in E$$
$$f_{u,v} \ge 0 \ , (u,v) \in E$$

אילוצי השימור והקיבול נותרו זהים. אמנם, בפונקציית המטרה נוסיף את האלמנט של $a_{u,v}$. בנוסף, יש לנו ביקוש של d יחידות בבור. מתי הבעיה פיזיבילית? כאשר $flow_{max} \ge d$. אם הזרמנו את הזרימה המקסימלית, נוכל להזרים גם זרימה קטנה יותר. אם היא פיזיבילית היא גם חסומה: אפשר לקחת את הערך המוחלט של המחירים כפול הקיבולות וזה בטוח מהווה חסם עליון לפונקציית המטרה, בהתאם גם חסם תחתון עם מינוס המחירים. על כן יש לה פתרון אופטימלי.

וריאציה 2 - Multicommodity flow:

במקום זוג אחד של מקור ובור, יש לנו k כאלה. יש כאן k בעיות זרימה שונות ובלתי תלויות: נרצה להזרים d_i יחידות זרימה מ-s_i ל-t_i.

- **אילוץ הקיבול הוא המקום שבו הבעיות נפגשות**: לא משנה מאיפה הזרימה מגיעה, ואז הן נהיות תלויות.
- **אילוץ השימור מתקיים בנפרד** לכל בעיה עבור כל צומת, נסתכל על זרימה מסוג i שנכנסת ויוצאת מהצומת.

באופן כללי, נגדיר פרמטר $\alpha \ge 0$, שעבורו נוכל להזרים במקביל αd_i יחידות מטיפוס i מ-s_i ל-t_i.

$$\max \ \alpha$$
$$\text{s.t.} \ \sum_{u\in V} f_{i,u,t_i} - \sum_{w\in V} f_{i,t_i,w} = \alpha d_i \ , \ i \in [k]$$
$$\sum_{u:(u,v)\in E} f_{i,u,v} - \sum_{w:(v,w)\in E} f_{i,v,w} = 0 \ , v \in V \setminus \{s_i, t_i\} \ , i \in [k]$$
$$\sum_{i=1}^{k} f_{i,u,v} \le c_{u,v} \ , (u,v) \in E \ , \ i \in [k]$$
$$f_{i,u,v} \ge 0 \ , (u,v) \in E \ , \ i \in [k]$$

הבעיה הדואלית של Maximum flow:

ניזכר קודם כל **בבעיה הפרימלית של MF**.

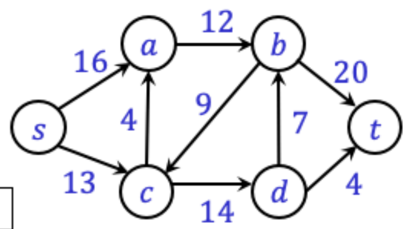
- השינוי האחד הוא שבעת נדבר על **מקיסום הזרימה שיוצאת מהמקור**.
- הקלה: נניח שאין קשתות בכלל שנכנסות למקור, ואין קשתות שיוצאות מהבור.

$$\max \sum_{v:(s,v)\in E} f_{s,v}$$
$$\text{s.t.} \quad \sum_{u:(u,v)\in E} f_{u,v} - \sum_{w:(v,w)\in E} f_{v,w} = 0 \ , \ v \in V \setminus \{s,t\}$$
$$f_{u,v} \le c_{u,v} \ , (u,v) \in E$$
$$f_{u,v} \ge 0 \ , (u,v) \in E$$

אנחנו רוצים למצוא את הבעיה הדואלית. נכתוב מטריצת אילוצים A. המשתנים מתאימים לעמודות המטריצה, יש לנו **עמודה לכל קשת**, ומשתנה פריימלי לכל קשת.

- **החלק התחתון** של המטריצה מדבר על אילוצי **הקיבול**, יש אחד כזה **לכל קשת**. לכן היא נראית כמו מטריצת הזהות. כל קשת צריכה להיות קטנה או שווה מהקיבולת שלה.
  - משתנה דואלי עבור כל צומת y_v.
- **החלק העליון** של המטריצה מדבר על אילוצי **שימור הזרימה**, יש אחד כזה **לכל צומת פנימי** שהוא לא s או t. נסתכל לדוגמה על צומת a: יש שתי קשתות שנכנסות אליו $(s,a),(c,a)$, ואחת שיוצאת (a,b). נרצה לוודא שסכום הזרימה על שתי הקשתות הנכנסות שווה לסכום הזרימה היוצאת על הקשת. יש לנו אילוץ לינארי עם מקדם 1 עבור הקשתות הנכנסות, ומקדם -1 עבור הקשת היוצאת, הסכום צריך להיות 0.
  - משתנה דואלי עבור כל קשת $z_{u,v}$.
- הוקטור העליון הוא פונקציית המטרה c. אנחנו רוצים למקסם את סך הזרימה שיוצאת מהמקור, אז נשים 1 על שתי הקשתות שיוצאות $(s,a),(s,c)$.

	(s,a)	(s,c)	(a,b)	(b,c)	(b,t)	(c,a)	(c,d)	(d,b)	(d,t)	
	1	1								
a	1		−1			1				
b			1	−1	−1			1		
c		1		1		−1	−1			
d							1	−1	−1	
(s,a)	1									16
(s,c)		1								13
(a,b)			1							12
(b,c)				1						9
(b,t)					1					20
(c,a)						1				4
(c,d)							1			14
(d,b)								1		7
(d,t)									1	4



Vertex-edge *incidence matrix* of G, without the rows of s and t.

Primal variables $f_{u,v}$ corresponding to columns.

Dual variables $y_v, z_{u,v}$ corresponding to rows.

- **ניקח את ה-transpose של המטריצה.**
  - הוקטור c פונקציית המטרה הופך להיות בצד ימין של האילוצים.
  - הוקטור b הופך להיות פונקציית המטרה: סוכמים רק על z כי ה-y מוכפל ב-0. לכן ה-y לא מופיע.
- עבור כל **קשת פנימית** יהיה לנו בצד העליון איפשהו 1 ואיפשהו -1 (יש צומת שהיא יוצאת ממנו, וצומת שהיא נכנסת אליו). בצד התחתון יהיה 1 כי זו מטריצת יחידה. לכן באילוץ הדואלי יהיו 3 מקדמים שונים מאפס עבור קשת פנימית. זה האילוץ $y_v - y_u + z_{u,v} \geq 0$.
- יש עוד **שני אילוצים עבור קשתות שיוצאות מ-s** (בצד העליון יהיו רק 1) וקשתות שיוצאות מ-t (בצד העליון יהיו רק -1). **עבור צומת המקור האילוץ יהיה $1 \leq$ כי פונקציית המטרה היא 1**, בניגוד לכל שאר הקשתות עבורן פונקציית המטרה 0.
- המשתנים הדואליים:
  - **אילוץ אחרון, $z_{u,v} \geq 0$** (משתנים דואלים של הקיבול על הקשתות) **כי במקור זה אי שוויון.**
  - **אנחנו לא דורשים על y** (משתנים דואלים על השימור של הצמתים) אי שוויון **כי במקור זה שוויון.**
- נרצה להפוך את כל האילוצים להיות $0 \leq$. נגדיר שני משתנים נוספים שמקובעים לערכים מסוימים $y_s = 1$ עבור האילוץ השני, ו-$y_t = 0$ עבור האילוץ השלישי.

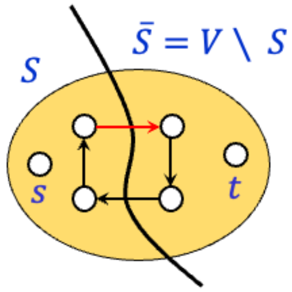
(P)
$$
\begin{aligned}
&\max \textstyle\sum_{u:(u,t)\in E} f_{u,t} \\
\text{s.t.}\quad & \textstyle\sum_{u:(u,v)\in E} f_{u,v} - \sum_{w:(v,w)\in E} f_{v,w} = 0 \ ,\ v \in V \setminus \{s,t\} \\
& f_{u,v} \leq c_{u,v} \ ,\ (u,v) \in E \\
& f_{u,v} \geq 0 \ ,\ (u,v) \in E
\end{aligned}
$$

(D)
$$
\begin{aligned}
&\min \textstyle\sum_{(u,v)\in E} c_{u,v} z_{u,v} \\
\text{s.t.}\quad & y_v - y_u + z_{u,v} \geq 0 \ ,\ (u,v) \in E \ ,\ u,v \in V \setminus \{s,t\} \\
& y_v \quad + z_{s,v} \geq 1 \ ,\ (s,v) \in E \\
& -y_u + z_{u,t} \geq 0 \ ,\ (u,t) \in E \\
& z_{u,v} \geq 0 \ ,\ (u,v) \in E
\end{aligned}
$$

(D)
$$
\begin{aligned}
&\min \textstyle\sum_{(u,v)\in E} c_{u,v} z_{u,v} \\
\text{s.t.}\quad & y_v - y_u + z_{u,v} \geq 0 \ ,\ (u,v) \in E \\
& y_s = 1 \ ,\ y_t = 0 \\
& z_{u,v} \geq 0 \ ,\ (u,v) \in E
\end{aligned}
$$

Equivalent form of the dual with $y_s = 1$ and $y_t = 0$.

בפתרון אופטימלי נקבל $\bar{z}_{u,v} = \max\{0, \bar{y}_u - \bar{y}_v\}$. יש לנו שני חסמים תחתונים עליו, הוא צריך לקיים את שניהם.



- נניח שהערכים של המשתנה הדואלי $\bar{y}_u$ הם או 0 או 1, זה משרה לנו חלוקה של הצמתים לשתי קבוצות. זו הנחה מאוד חזקה ולא טריוויאלית.
- נגדיר $S = \{u \in V | \bar{y}_u = 1\}$, $t \notin S$ כי קיבענו $y_t = 0$ וכל השאר יהיו הקבוצה T. קיבלנו חתך S-T.
- הקשתות שעבורן $\bar{z}_{u,v} = 1$ אלה הקשתות שחוצות את החתך, כי $y_u = 1, y_v = 0$. על חתך כזה נגיד שהקיבולת שלו היא סכום הקיבולות של כל הקשתות שחוצות אותו מ-S ל-T.

<u>הבעיה הפריימלית של SSSP (בעיית מינימום):</u>

ניקח את הבעיה הדואלית של הבעיה (מקסימום) שהצגנו קודם. זה דומה לבעיית minimum cost flow כאשר x הוא הזרימה f, $d_v = 1$ בכל צומת שאינו המקור, **ואין אילוצי קיבול**. כדי שזה יהיה פיזיבילי, צריך שיהיה $d_s = -(n-1)$.

$$\text{(P)} \qquad \begin{aligned} &\min \ \sum_{(u,v)\in E} w_{u,v} x_{u,v} \\ &\sum_{u:(u,v)\in E} x_{u,v} - \sum_{w:(v,w)\in E} x_{v,w} = 1 \ , \ v \in V \setminus \{s\} \\ &x_{u,v} \geq 0 \ , (u,v) \in E \end{aligned}$$

תכנון ליניארי 2 (תרגול 13)

<u>שאלה 0 (מוטיבציה לתוכנית הדואלית):</u> נתונה תכנית ליניארית $\max x_1 + 6x_2$, כיתבו תוכנית ליניארית שמוצאת **חסם עליון מינימלי על ערך פונקציית המטרה**.

נחפש את החסם הכי טוב שנוכל להשיג על ידי מקדמים שנכפיל במשוואות, כדי להרכיב משוואה שחוסמת את ערך פונקציית המטרה. **נתאים לכל אילוץ משתנה** אי-שלילי (פקטור להכפלה), ונכפול אותו בו: $y_1, y_2, \dots$. כל המקדמים אי-שליליים כדי שלא נהפוך את כיוון האי-שוויון.

נקבל **בצד שמאל** ביטוי שצריך להיות גדול שווה לפונקציית המטרה (המקדם של x_1 צריך להיות גדול שווה 1, ושל x_2 גדול שווה 6), **ובצד ימין** את החסם העליון.

נסדר את המשוואה כדי לקבל את המקדמים של x_1, x_2, ונדרוש עליהם את התנאים אל מול פונקציית המטרה המקורית. **כל משתנה מכתיב לנו אילוץ.**

התוכנית המבוקשת היא min על החסם. זאת בדיוק התוכנית הדואלית.

maximize $x_1 + 6x_2$

$x_1 + 2x_2 \leq 18$	$/\cdot y_1$	$y_1(x_1 + 2x_2) \leq 18y_1$
$x_1 - 3x_2 \leq -9$	$/\cdot y_2$	$y_2(x_1 - 3x_2) \leq -9y_2$
$-5x_1 + 4x_2 \leq 7$	$/\cdot y_3$	$y_3(-5x_1 + 4x_2) \leq 7y_3$

$x_1, x_2 \geq 0$

- נסכום:

$$y_1(x_1 + 2x_2) + y_2(x_1 - 3x_2) + y_3(-5x_1 + 4x_2) \leq 18y_1 - 9y_2 + 7y_3$$

$x_1, x_2 \geq 0$

- נמצא את המקדמים של x_1, x_2:

$$(y_1 + y_2 - 5y_3)x_1 + (2y_1 - 3y_2 + 4y_3)x_2 \leq 18y_1 - 9y_2 + 7y_3$$

- אם מתקיים $y_1 + y_2 - 5y_3 \geq 1$ וגם $2y_1 - 3y_2 + 4y_3 \geq 6$

אז $18y_1 - 9y_2 + 7y_3$ חוסם מלמעלה את הערך של פונ' המטרה.

\

$x_2 \leq 18$
$x_2 \leq -9$
$+4x_2 \leq 7$
≥ 0

- התוכנית המבוקשת:

ערך החסם

$$\min \ 18y_1 - 9y_2 + 7y_3$$

s.t.

$$\begin{cases} y_1 + y_2 - 5y_3 \geq 1 \\ 2y_1 - 3y_2 + 4y_3 \geq 6 \\ y_1, y_2, y_3 \geq 0 \end{cases}$$

אילוץ שדורש ש-y_1, y_2, y_3 אכן נותנים חסם

- זאת בדיוק התוכנית הדואלית.

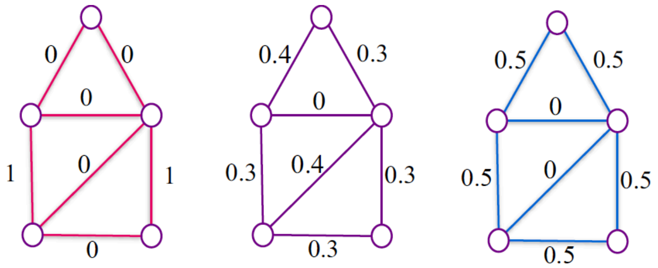
<u>התוכנית הדואלית</u>: נתונה לנו תוכנית בצורה הסטנדרטית: $\max c^t x, Ax \le b, x \ge 0$. כל אילוץ הופך למשתנה וכל משתנה הופך לאילוץ. המקסימום הופך להיות מינימום (מחפשים חסם מינימלי). אי שוויונות משנים כיוון, אילוצי החיוביות נשארים אבל בשביל המשתנים החדשים. נקבל $\min b^t y, A^t y \ge c, y \ge 0$.

$$\begin{array}{ll|ll}
\text{maximize} & \sum_{j=1}^{n} c_j x_j & \text{minimize} & \sum_{i=1}^{m} b_i y_i \\
\sum_{j=1}^{n} a_{ij} x_j \le b_i & \text{for } i = 1,2,\dots,m & \sum_{i=1}^{m} a_{ij} y_i \ge c_j & \text{for } j = 1,2,\dots,n \\
x_j \ge 0 & \text{for } j = 1,2,\dots,n & y_i \ge 0 & \text{for } i = 1,2,\dots,m
\end{array}$$

- הוקטור $c^t = (1{,}6)$ הפך להיות וקטור עמודה באילוצים $c = \binom{1}{6}$.
- הוקטור $b = \begin{pmatrix} 18 \\ -9 \\ 7 \end{pmatrix}$ הפך להיות וקטור מקדמים בפונקציית המטרה $b^t = (18, -9, 7)$.
- והמטריצה $A = \begin{pmatrix} 1 & 2 \\ 1 & -3 \\ -5 & 4 \end{pmatrix}$ הפכה להיות $A^t = \begin{pmatrix} 1 & 1 & -5 \\ 2 & -3 & 4 \end{pmatrix}$.

$$\begin{array}{ll|l}
\text{maximize } x_1 + 6x_2 & & \text{minimize } 18y_1 - 9y_2 + 7y_3 \\
x_1 + 2x_2 \le 18 & \leftarrow \boxed{y_1} & y_1 + y_2 - 5y_3 \ge 1 \\
x_1 - 3x_2 \le -9 & \leftarrow \boxed{y_2} & 2y_1 - 3y_2 + 4y_3 \ge 6 \\
-5x_1 + 4x_2 \le 7 & \leftarrow \boxed{y_3} & y_1, y_2, y_3 \ge 0 \\
x_1, x_2 \ge 0 & &
\end{array}$$

<u>שאלה 1 (זיווג שברי)</u>: מצאו LP לבעיה הבאה, והסבירו מהי הבעיה המתאימה לתוכנית הדואלית לה: נתון גרף לא מכוון. אנחנו רוצים לתת משקל אי-שלילי לכל קשת בגרף כך **שסכום משקלי הקשתות יהיה מקסימלי**. לכל קודקוד בגרף, סכום משקלי הקשתות **הסמוכות עליו הינו לכל היותר 1**.



איזו בעיה התוכנית הייתה פותרת אם היינו דורשים שהמשתנים יכילו רק ערכים שלמים? כל קשת תהיה 0 או 1. זה פותר לנו את בעיית הזיווג. אף קודקוד לא נבחר פעמיים, וזה מתאים לכך שקשת בחרה שני קודקודים, אז כל הקשתות שנוגעות בקודקודים האלה הן 0, אי אפשר שוב לבחור את הקודקודים האלה. אלו קבוצת קשתות זרה בקודקודים.

באופן כללי – אם יש בעיה P בצורה סטנדרטית, ונגדיר IP שזו אותה בעיה כאשר נגביל את המשתנים להיות שלמים. מה הקשר בין הפתרונות האופטימליים של הבעיות? $opt(IP) \le opt(P)$. אם מגבילים את המשתנים רק נפגע בערך האופטימלי של הפתרון.

<u>התוכנית הפריימלית</u>:

- x_e – משקל קשת ($|E|$ משתנים)
- לכל קודקוד יש אילוץ ($|V|$ אילוצים)

$$\begin{array}{ll}
\text{maximize} & \sum_{e \in E} x_e \\
\sum_{e: v \in e} x_e \le 1 & \text{for } v = 1,2,\dots,|V| \\
x_e \ge 0 & \text{for } i = 1,2,\dots,|E|
\end{array}$$

סכום על כל הקשתות אשר מכילות את הקודקוד.

התוכנית תלויה בגרף, וכל האילוצים הם פונקציה של הקשתות והצמתים בגרף. המטריצה A **לא כתובה בצורה מפורשת**. **כאן נגיע לדואלית ישירות בעזרת המוטיבציה** (לא בעזרת הנוסחה כמו תובי).

<u>התוכנית הדואלית</u>:

- נתאים לכל אילוץ (קודקוד) משתנה (נכפיל כל אילוץ במשתנה y_v).
- נסכום ונקבל את החסם העליון לפונקציית המטרה Σy_v (כי בצד ימין יש לנו 1).
- נקבל בצד שמאל ביטוי שצריך להיות גדול שווה לפונקציית המטרה (כל משתנה מכתיב לנו אילוץ חדש, לפי המקדמים שלו שצריכים להיות גדולים שווים ל-1).
- נקבל בצד ימין את החסם העליון שנרצה להביא למינימום.

$$\begin{array}{ll|ll}
\text{maximize} & \sum_{e=1}^{|E|} x_e & \text{minimize} & \sum_{v=1}^{|V|} y_v \\
\sum_{e: v \in e} x_e \le 1 & \text{for } v = 1,2,\dots,|V| & \sum_{v \in e} y_v \ge 1 & \text{for } e = 1,2,\dots,|E| \\
x_e \ge 0 & \text{for } e = 1,2,\dots,|E| & y_v \ge 0 & \text{for } v = 1,2,\dots,|V|
\end{array}$$

- משתנה לכל קשת
- אילוץ לכל קודקוד

- משתנה לכל קודקוד
- אילוץ לכל קשת

<u>משמעות הדואלית בשלמים:</u> כל קודקוד מקבל 0 או 1, לכן כל קודקוד נבחר או לא נבחר. מחפשים קבוצה בגודל מינימלי של קודקודים. האילוץ? לפחות קודקוד אחד יגע בכל קשת. זאת בעיית vertex cover – מציאת קבוצה מינימלית של קודקודים כך שכל קשת נוגעת בלפחות קודקוד אחד מהקבוצה. נקבל באופן כללי כי $opt(IP) \le opt(P) \le opt(D) \le opt(ID)$.

<u>משפט הדואליות (דואליות חזקה):</u> אם ל-LP יש פתרון פיזיבילי אופטימלי, אז ערך הפתרון האופטימלי של P יהיה זהה לערך הפתרון האופטימלי של D. קיימים וקטורים x, y כך שיש שוויון: $c^t x = b^t y$.

<u>דואליות חלשה:</u> זה מה שהיה ברור כשעשינו את המוטיבציה. הדואלית היא חסם עליון לפריימלית. לכל x, y פתרונות ל-P, D מתקיים כי $c^t x \le b^t y$. אם מצאנו $\bar{x}$ כך ש-$c^t \bar{x} = b^t y$. אם $b^t y$ חוסם כל פתרון של P, אז גם $c^t \bar{x}$ חוסם כל פתרון של P ולכן הוא אופטימלי. **זה כבר נובע מדואליות חלשה.**

<u>שאלה 2 (פיזיביליות לינארית):</u>

1) נתון אלגוריתם לפתרון בעיות LP – נוסיף לבעיה פונקציית מטרה כלשהי, נניח max 0 ונקבל בעיית LP. את הבעיה שהתקבלה אפשר לפתור בעזרת האלגוריתם הנתון.

2) נתון אלגוריתם לפתרון בעיות LF – נשתמש באילוצים של P וגם של D, ונרצה למצוא x, y כך שכל האילוצים מתקיימים וגם הפתרונות שווים: $c^t x = b^t y$. כך נקבל את הערך האופטימלי, ולא סתם שני פתרונות שונים.

<u>שאלה 3:</u> נתון אלגוריתם A שמקבל בקלט צורה סטנדרטית של תוכנית לינארית Q, עם n משתנים ו-2022 אילוצים ומחזיר פתרון אופטימלי של Q. כמו כן, נתונה תכנית לינארית P בצורה סטנדרטית עם n משתנים ו-2023 אילוצים, עבורה ידוע שקיים פתרון אופטימלי $\vec{y}$ לתוכנית הדואלית D, שבו המכפלה $y_1, \dots, y_{2023}$ מתאפסת. תארו אלגוריתם יעיל ככל הניתן למציאת ערך הפתרון האופטימלי של P.

נסתכל מנקודת המבט של הדואלית קודם:

- אם המכפלה מתאפסת, אחד המשתנים ב-D שווים לאפס: y_j כלשהו (כלומר יש אילוץ ב-P שלא משתמשים בו).
- נגדיר לכל j את D_j את התוכנית הדואלית שמתקבלת מהצבת $y_j = 0$. יש לנו 2023 תוכניות חדשות, עם משתנה אחד פחות (2022 משתנים) לכל $1 \le j \le 2023$.
- קודם כל $opt(P) = opt(D)$ מדואליות חזקה. אם נציב משתנה ספציפי רק אפשר לפגוע בפתרון האופטימלי ולכן $opt(D) \le opt(D_j)$ תמיד מתקיים (כי דואלית זה מינימיזציה, רצים על פחות משתנים ויש אותו ערך מטרה). לפי הנתון יש $y_j = 0$ בפתרון האופטימלי ולכן $opt(D) = \min_{1 \le j \le 2023} opt(D_j)$.
- **יכול להיות ש-D_j לא פיזיבילית**, ברגע שנעשה הצבה כזאת נפגע בפיזיביליות. האם D_j יכולה להיות לא חסומה? **לא**, רק צמצמנו את מרחב הפתרונות, זה עדיין חסום.

נתרגם חזרה ל-P:

- נסמן ב-P_j את הדואלית של D_j, שזו בדיוק התכנית P ללא האילוץ j. כלומר זו תוכנית לינארית עם 2022 אילוצים.
- מדואליות חזקה: $opt(P_j) = opt(D_j)$ (**יכול להיות ש-P_j לא חסומה** כי הורדנו אילוץ, אולי הוא גרם לכך שהיא חסומה. האם יכול להיות שפגענו בפיזיבליות כי הורדנו אילוץ? **לא**, רק הרחבנו את מרחב הפתרונות שלנו).
- נחשב את $\min_{1 \le j \le 2023} opt(P_j)$ וזה יתן לנו את $opt(P)$ על ידי 2023 קריאות מתאימות ל-A.