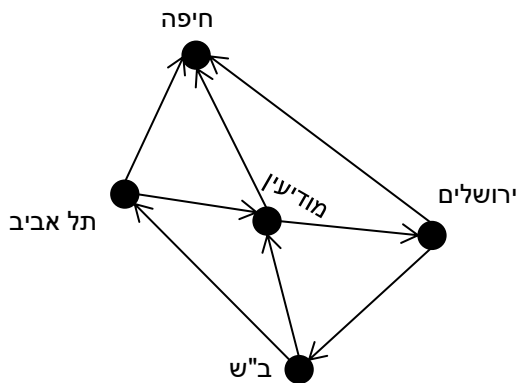


## הרצאה 10: מסלולים קלים ביותר בגרף מכוון

Textbook: Cormen, Leiserson, Rivest,  
Stein. Introduction to Algorithms.

מרצה: אורי שטמר

**דוגמה:** יש לנו ערים ויש לנו בין כל זוג ערים מרחק. אנחנו רוצים לדעת מהי הדרך הקלה ביותר להגיע מב"ש לכל אחת משאר הערים.



**הגדרה:** בהינתן מסלול  $P = (v_1, v_2, \dots, v_k)$  בגרף עם משקולות על הקשתות, משקל המסלול הוא סכום המשקולות של הקשתות במסלול:

$$w(P) = \sum_{i=1}^{k-1} w(v_i, v_{i+1})$$

**הגדרה:**

$$\delta(u, v) = \begin{cases} \text{משקל מסלול קל ביותר מ } u \text{ ל } v & , \\ \infty & , \end{cases} \text{ אחרת}$$

### בעיית המסלולים הקלים ביותר עם מקור יחיד

**קלט:** גרף  $G = (V, E)$  מכוון ופונקציית משקל  $w: E \rightarrow \mathbb{R}$  וצומת  $s \in V$  (צומת זה נקרא "המקור")

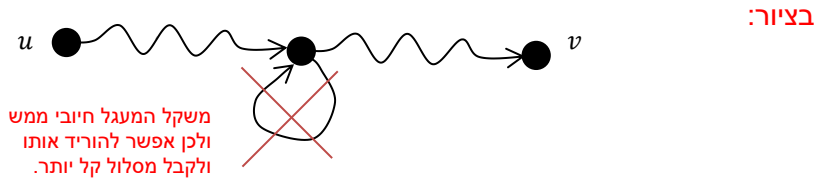
**יש למצוא:** לכל  $v \in V$  מסלול קל ביותר מ-  $s$  ל-  $v$

תזכורת: כשאנו אומרים מסלול קל ביותר – הכוונה היא לפי משקולות. כשאנו אומרים מסלול קצר ביותר – הכוונה היא לפי מספר צלעות.

**מקרה פשוט:**  $w(u, v) = 1$  עבור כל  $(u, v) \in E$ . במקרה זה משקל מסלול שווה לאורך המסלול, וקיבלנו את בעיית המסלול הקצר ביותר. פותרים על ידי BFS בסיבוכיות  $O(|V| + |E|)$ .

## נתחיל מלהבחין בכמה תכונות שימושיות

**אבחנה 1:** אם כל המשקולות בגרף חיוביים (כלומר גדולים ממש מאפס), אזי אם קיים מסלול מ- $s$  ל- $v$  אזי מסלול קל ביותר הוא מסלול פשוט.



**אבחנה 2:** אם כל המשקולות בגרף הם אי-שליליים אזי אם קיים מסלול מ- $s$  ל- $v$  אזי קיים מסלול קל ביותר מ- $s$  ל- $v$  שהוא פשוט.

מדוע אנחנו מציינים את 2 האבחנות האלה? עכשיו אנחנו יודעים שלפחות במקרה של משקלים אי-שליליים, כשנתכנן אלגוריתם לבעייה נוכל להתרכז במציאת מסלולים פשוטים.

**משפט:** אם  $(v_1, v_2, \dots, v_k)$  הוא מסלול קל ביותר מ- $v_1$  ל- $v_k$  אזי לכל  $1 \leq i < \ell \leq k$  מתקיים ש- $(v_i, v_{i+1}, \dots, v_\ell)$  הוא מסלול קל ביותר מ- $v_i$  ל- $v_\ell$ .

### רעיון ההוכחה:

נניח בשלילה כי המסלול  $(v_i, v_{i+1}, \dots, v_\ell)$  אינו קל ביותר. כלומר קיים מסלול  $P$  קל יותר מ- $v_i$  ל- $v_\ell$ .

$$(v_1, \dots, v_{i-1}) \circ P \circ (v_{i+1}, \dots, v_k)$$

זהו מסלול קל יותר מ- $v_1$  ל- $v_k$ . סתירה.

**מסקנה (טענת הרישא):** אם  $(v_1, v_2, \dots, v_k)$  הוא מסלול קל ביותר מ- $v_1$  ל- $v_k$  אזי לכל  $1 \leq i \leq k$  מתקיים ש- $(v_1, v_2, \dots, v_i)$  הוא מסלול קל ביותר מ- $v_1$  ל- $v_i$ .

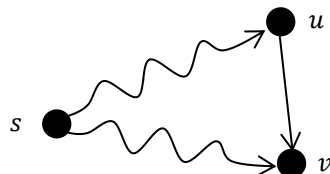
**מסקנה נוספת:** אם  $(v_1, v_2, \dots, v_{k-1}, v_k)$  מסלול קל ביותר מ- $v_1$  ל- $v_k$  אזי  $\delta(v_1, v_k) = \delta(v_1, v_{k-1}) + w(v_{k-1}, v_k)$

### הוכחת המסקנה הנוספת:

$$\delta(v_1, v_k) = w(v_1, v_2, \dots, v_{k-1}, v_k) = w(v_1, v_2, \dots, v_{k-1}) + w(v_{k-1}, v_k) = \delta(v_1, v_{k-1}) + w(v_{k-1}, v_k)$$

**אבחנה 3 (טענת הקשת):** לכל קשת  $(u, v) \in E$  מתקיים  $\delta(s, v) \leq \delta(s, u) + w(u, v)$

בציור:



## איך אפשר לפתור את הבעיה?

יש לנו גרף מכוון עם משקולות ואנחנו רוצים למצוא מסלול קל ביותר מ- $s$  לכל קודקוד בגרף.

**אלגוריתם נאיבי:** כדי למצוא מסלול קל ביותר מ- $s$  לקודקוד  $v$  נעבור על כל האפשרויות למסלול פשוט מ- $s$  ל- $v$  ונבדוק אם קיבלנו מסלול בגרף ואם כן ניקח מסלול עם משקל מינימלי (לפי מה שאמרנו קודם, זה עובד לפחות כאשר המשקולות בגרף הם אי-שליליות...).  
סיבוכיות:  $O(n!) = 2^{O(n \cdot \log n)}$ .

היום נדבר רק על איך מוצאים משקל מסלול קל ביותר מ- $s$  לכל קודקוד בגרף

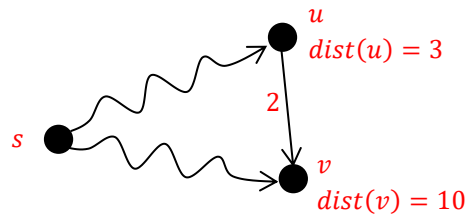
**רעיון לאלגוריתם יעיל יותר:**

נחזיק לכל צומת משתנה  $dist(v)$  כך שבסוף ריצת האלגוריתם יתקיים  $dist(v) = \delta(s, v)$ .

אינטואיטיבית, בכל שלב במהלך הריצה, המשתנה  $dist(v)$  יחזיק את "משקל המסלול הקל ביותר שאנחנו יודעים עליו עד עכשיו". לאורך הריצה נעדכן את המשתנים האלו כאשר "נגלה" מסלולים קלים יותר. בסוף הריצה נדאג שמשתנים אלו יכילו את הערך הנכון.

דוגמה:

נניח שמתישו במהלך הריצה, יש לנו 2 צמתים  $u, v$  כך



איך נעדכן?

**Relax( $u, v$ )**

- If  $(u, v) \in E$  and  $dist(v) > dist(u) + w(u, v)$
- Then  $dist(v) \leftarrow dist(u) + w(u, v)$

**הערה:** שימו לב ש- $dist(v)$  קטן ממש בכל עדכון

**אלגוריתם גנרי למציאת משקלי מסלולים קלים ביותר:**

**אתחול:**  $dist(s) = 0$  ולכל  $v \neq s$  בצע  $dist(v) = \infty$

**צעד:** כל עוד קיימת קשת  $(u, v) \in E$  כך ש- $dist(v) > dist(u) + w(u, v)$ , בחר קשת  $(u, v)$  כנ"ל ובצע Relax( $u, v$ )

כפי שנראה בהמשך ההרצאה, אם האלגוריתם הזה עוצר אזי בסיום הריצה לכל  $v$  מתקיים  $dist(v) = \delta(s, v)$ . כלומר אם האלגוריתם עוצר אז הוא מצליח לחשב את מה שרצינו...

**הבעיה:** מספר פעולות ה Relax שהאלגוריתם עושה יכול להיות אקספוננציאלי. אנחנו נראה איך לממש את האלגוריתם הזה בצורה כזאת שזמן הריצה שלו יהיה פולינומי. נעשה זאת על ידי בחירה חכמה של הצלע הבאה בכל שלב שנבצע עליה Relax.

לפני שדבר על מימושים ספציפיים כאלה, נראה כמה תכונות של האלגוריתם הגנרי (בפרט, תכונות אלו יהיו נכונות עבור המימושים הספציפיים שנראה בהמשך).

**טענה נשמרת עבור האלגוריתם:** אם  $dist(v) \neq \infty$  אזי קיים מסלול מ- $s$  ל- $v$  בגרף שמשקלו  $dist(v)$ .

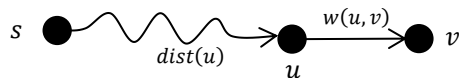
**הוכחה:** באינדוקציה על מספר העדכונים.

**בסיס:** עבור הקודקוד  $s$  קיים מסלול במשקל אפס.

**צעד:** נניח שביצענו עדכון  $dist(v) \leftarrow dist(u) + w(u, v)$ . מהנחת האינדוקציה קיים מסלול מ- $s$  ל- $u$  במשקל  $dist(u)$ . משקל מסלול זה בתוספת הקשת  $(u, v)$  הוא

$$dist(v) = dist(u) + w(u, v)$$

בצורה:



מ.ש.ל.

**מסקנה (טענת החסם העליון):** בכל שלב באלגוריתם  $\delta(s, v) \leq dist(v)$ .  
 משקל מסלול או  $\infty$  או משקל מסלול כלשהו קל ביותר

**משפט העצירה:** אם האלגוריתם הגנרי עוצר אזי לכל צומת  $u$  מתקיים  $dist(u) = \delta(s, u)$ .

**הוכחה:**

נניח בשלילה כי האלגוריתם עצר אבל קיים צומת  $u$  כך ש- $dist(u) \neq \delta(s, u)$ . מטענת החסם העליון, במקרה זה  $dist(u) > \delta(s, u)$ . לכן קיים מסלול מ- $s$  ל- $u$  שמשקלו קטן ממש מ- $dist(u)$ . יהי  $P$  מסלול כזה.

לכל צומת  $x$  במסלול נסמן ב- $w_P(x)$  את משקל רישת המסלול מ- $s$  ל- $x$ . מההנחה על המסלול אנו יודעים שמתקיים

$$dist(u) > w_P(u)$$

אבל מצד שני,

$$dist(s) \leq 0 = w_P(s)$$

יהי  $y$  הצומת הראשון במסלול  $P$  כך ש

$$(1) \quad dist(y) > w_P(y)$$

יהי  $x$  הצומת לפני  $y$  במסלול. מהראשונות של  $y$  מתקיים

$$(2) \quad dist(x) \leq w_p(x)$$

מכיוון שהאלגוריתם עצר, אנו יודעים שאין פעולות Relax אפשריות. בפרט, אי אפשר לעשות Relax(x,y) ולכן

$$(3) \quad dist(y) \leq dist(x) + w(x, y)$$

נזכור ש-  $w_p(y)$  זהו משקל המסלול מ-  $s$  עד  $y$ , כאשר הקודקוד לפני  $y$  במסלול זה הוא  $x$ . לכן,

$$(4) \quad w_p(y) = w_p(x) + w(x, y)$$

כעת נראה שמתוך ((1)), ((2)), ((3)), ((4)) אנחנו מקבלים סתירה:

$$w_p(y) < \underset{(1)}{dist(y)} \leq \underset{(3)}{dist(x) + w(x, y)} \leq \underset{(2)}{w_p(x) + w(x, y)} = \underset{(4)}{w_p(y)}$$

מ.ש.ל.

מה הוכחנו? אם האלגוריתם עוצר אזי לכל צומת מתקיים שהערך  $dist$  שווה לערך הנכון. לא הוכחנו שהאלגוריתם עוצר. ניתן להוכיח שאם אין מעגלים שליליים אז האלגוריתם תמיד עוצר, אבל זמן הריצה יכול להיות אקספוננציאלי.

נראה שני מימושים של האלגוריתם הגנרי:

- האלגוריתם של Dijkstra עבור משקלים אי-שליליים
- האלגוריתם של Bellman-Ford אשר פועל גם עבור משקלים שליליים

### האלגוריתם של Dijkstra עבור גרפים עם משקולות אי-שליליים: $w(e) \geq 0$ לכל $e \in E$

הרעיון: בכל שלב נבחר צומת עם  $dist(v)$  מינימלי נבדוק אם אפשר לעדכן את השכנים שלו, ויותר לא נעדכן את  $v$  (יש כאן איזשהו צעד חמדני...)

**אתחול:**  $S \leftarrow \emptyset$  (לאורך הריצה,  $S$  תהייה קבוצת הצמתים שכבר "טיפלנו" בהם ועבורם  $dist(v) = \delta(s, v)$ )

$$dist(s) = 0$$

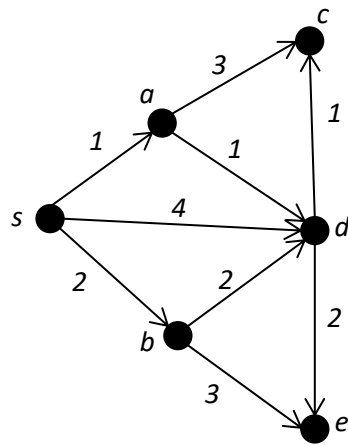
$$dist(v) = \infty \text{ בצע } v \neq s \text{ לכל}$$

**צעד:** כל עוד  $|S| < |V|$

- בחר צומת  $v \notin S$  עם  $dist(v)$  מינימלי
- $S \leftarrow S \cup \{v\}$
- לכל צומת  $x \notin S$  שכן של  $u$ , כלומר  $(v, x) \in E$ , בצע  $Relax(v, x)$

שימו לב שהאלגוריתם הזה הוא מקרה פרטי של האלגוריתם הגנרי שראינו קודם, כי שם אמרנו שצריך לבצע Relax עבו קשת כלשהי וכאן אנחנו בוחרים קשתות באופן יותר ספציפי

דוגמת ריצה:



	<i>dist(·)</i>					
	אתחול	שלב ראשון	שלב שני	שלב שלישי	שלב רביעי	שלב חמישי
<b>s</b>	0	0	0	0	0	0
<b>a</b>	∞	1	1	1	1	1
<b>b</b>	∞	2	2	2	2	2
<b>c</b>	∞	∞	4	4	3	3
<b>d</b>	∞	4	2	2	2	2
<b>e</b>	∞	∞	∞	5	4	4