המשך ההוכחה של האלגוריתם של Dijkstra

<u>אמרנו:</u>

- $d[v]$ – ניחוש למרחק מ- s ל- v
- $\delta(s,v)$ – המרחק מ- s ל- v (אורך/משקל מק"ב)

<u>הנחה:</u>

- כל הקדקודים נגישים מ- s
- כל המשקלות ≤ 0

האלגוריתם של Dijkstra הוא מימוש של האלגוריתם הגנרי שפותר את הבעיה כאשר הנחות אלה נכונות.

[[את האלגוריתם ניתן למצוא בהרצאה הקודמת.]]

משפט [ניחשנו נכון]

בסוף האלגוריתם, $d[v]=\delta(s,v)$ לכל קדקוד $v\in V$.

<u>הערה</u>: לפי משפט הנכונות של האלגוריתם הגנרי, האלגוריתם של דיקסטרה מחזיר עץ מק"ב מ- s.

הוכחנו את המשפט באמצעות:

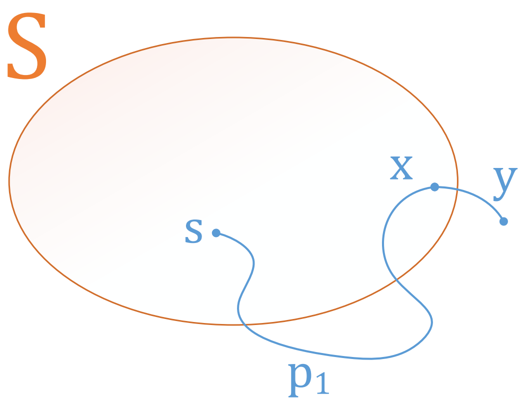
טענת עזר

בשלב בו בחרנו את u בתור הקדקוד שנכנס ל- S, כבר התקיים $d[u]=\delta(s,u)$.

<u>הוכחה:</u>
באינדוקציה על סדר הצעדים.

- <u>בסיס האינדוקציה</u>: בשלב הראשון בוחרים את s, ואכן מתקיים $d[s]=0=\delta(s,s)$.
- <u>צעד אינדוקטיבי:</u>
  נניח את הנחת האינדוקציה לכל הקדקודים שנכנסו לפני u.

  - <u>"טענת עזר"-עזר</u>: אם קיים מק"ב מ- s לקדקוד y שבו הקשת האחרונה היא (x,y) כאשר $x\in S$ ו- $y\notin S$ – אזי בשלב הזה]] של האינדוקציה [[מתקיים $d[y]=\delta(s,y)$.



  - <u>הוכחה:</u>
    נקרא למסלול $p=p_1\circ(x,y)$.
    זה מק"ב, ולכן גם הרישא p_1 היא מק"ב מ- s ל- x.
    כלומר $w(p_1)=\delta(s,x)$.
    לפי הנחת האינדוקציה, כש- x נכנס

ל- S [[וזה קרה כי הוא כרגע ב- S]], התקיים $\delta(s,x)=d[x]$, ופעולות $\text{Relax}(x,y)$ באותו שלב הביאו לכך ש:

$$d[y]\le d[x]+w(x,y)=\delta(s,x)+w(x,y)=\delta(s,y)$$

לכן, בכל שלב עתידי (כולל בצעד הנוכחי) יתקיים $d[y]\le\delta(s,y)$ (כי $d[\cdot]$ יכול רק לרדת) – אבל הראינו שלכל מימוש של האלגוריתם הגנרי תמיד מתקיים $\forall v\in V\ d[v]\ge\delta(s,v)$.

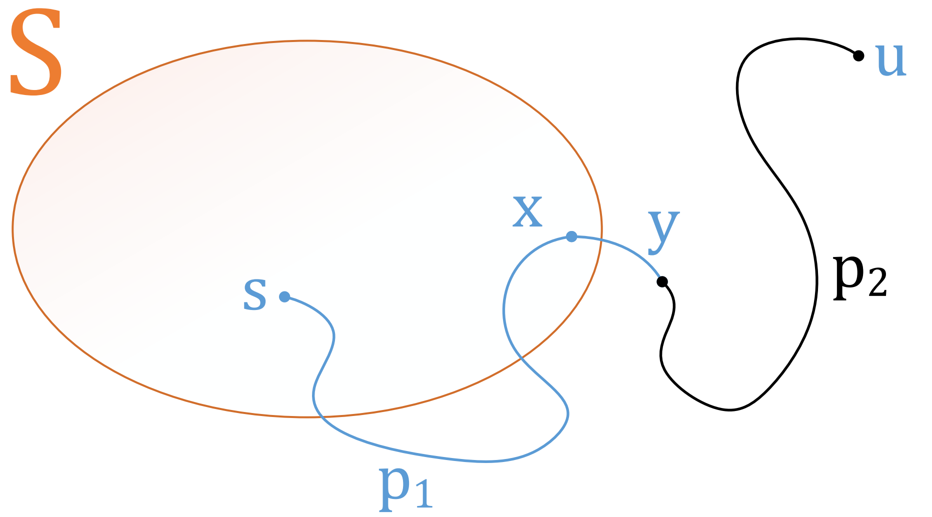
לכן $d[y]=\delta(s,y)$.

□

<u>חזרה לצעד האינדוקטיבי:</u>

תהי (x,y) קשת כלשהי במק"ב מ- s ל- u כך ש- $x\in S$ וְ- $y\notin S$.

נסמן את המסלול ב- $p=p_1\circ p_2$ [p_1 מסלול מ- s ל- y וְ- p_2 מסלול מ- y ל- u].



רישא של מק"ב היא מק"ב, ולכן p_1 היא מק"ב ל- y שמסתיים ב-(x,y) כנ"ל.

לכן לפי "טענת עזר"-העזר, מתקיים $d[y]=\delta(s,y)$.

מכך שהבחירה החמדנית בחרה את u בשלב זה (ולא את y) [[אם היא בחרה את u אז כבר סיימנו ישירות מ"טענת עזר"-העזר]] נובע ש- $d[u]\le d[y]$.

מצד שני:

$$d[u]\ge\delta(s,u)=w(p_1)+w(p_2)=\delta(s,y)+\underbrace{w(p_2)}_{\ge 0}\ge\delta(s,y)=d[y]$$

יש אי-שוויון בשני הכיוונים, ולכן מתקיים $d[y]=d[u]$ וכל אי-השוויונות הם הדוקים, ובפרט מתקיים כי $d[u]=\delta(s,u)$.

יש פה חלון! ← □

מימוש וניתוח זמן ריצה של דייקסטרה

<u>מימוש:</u>

נתחזק את S באמצעות מערך:

$$A[v]=\begin{cases}1, & v\in S\\ 0, & v\notin S\end{cases}$$

כדי לשלוף u עם $d[u]$ מינימלי נתחזק ערימה Q.

- <u>אתחול:</u>
  - מכניסים את כל הקדקודים v לפי מפתח $d[v]$
- <u>צעד:</u>
  - $u\leftarrow \text{Extract\_Min}(Q)$
  - $A[u]=1$
  - לכל שכן v של u, אם $A[v]=0$:
    - $\text{Relax}(u,v)$
    - $\text{Decrease\_Key}(Q,v,d[v])$

[[אנחנו מעדכנים כאן את $d[v]$, וזה המפתח שלפיו הערימה שלנו פועלת, לכן יש לעדכן את הערימה כך שתכלול את השינוי. בערימה רגילה זה נעשה ע"י הוצאת האיבר והכנסתו חזרה עם מפתח מעודכן, אך ישנם מימושים בהם ניתן לעשות זאת בצורה יעילה יותר. אתם יכולים לחשוב על זה בתור Update_Key אם אתם מעדיפים.]]

<u>זמן ריצה:</u>

נספור פעולות ערימה:

- $|V|$ פעמים Insert [[בעצם ניתן לבנות את הערימה בצורה קצת יעילה יותר, אבל זה לא באמת משנה פה]]
- $|V|$ פעמים Extract_Min
- $|E|$ פעמים Decrease_Key

ובנוסף אנו מבצעים $|E|$ פעולות Relax.

סה"כ: $O(|E|\log|V|)$ ע"י ערימה רגילה.

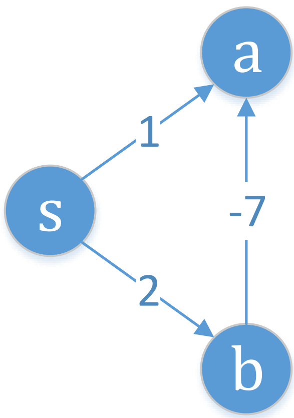
הערה [מימוש יעיל יותר]

ערימת Fibonacci מורידה את זמן הריצה הממוצע ל- $O(1)$ לכל פעולה חוץ מ- Extract_Min ו- Delete.

עם ערימת פיבונאצ'י זמן הריצה יהיה $O(|E|+|V|\log|V|)$.

הבעיה במשקלות שליליים

דוגמה:



בסוף ריצת האלגוריתם נקבל כי $d[a]=1$ ו- $d[b]=2$, אבל $\delta(s,a)=-5$.

מה משתבש בהוכחה?

היה מקום שבו אמרנו ש- $w(p_2)\geq 0$ ולכן המסלול הנוסף מ- y ל- u יכול רק להגדיל את המשקל הכולל. במקרה שבו יש משקולות שליליים זה לא נכון.

אלגוריתם של Bellman-Ford

תכונות

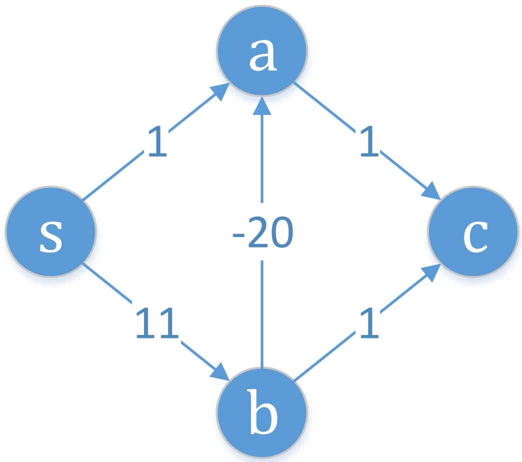
- זהו מימוש נוסף של האלגוריתם הגנרי
- עובד גם למשקולות שליליים
- בודק אם יש מעגל שלילי
- אם אין, האלגוריתם מחזיר $\forall v:\ d[v]=\delta(s,v)$
  $\Leftarrow$ מחזיר גם עץ מק"ב

האלגוריתם

- <u>אתחול</u>: (כמו בכל מימוש גנרי (למשל בדייקסטרה))
- <u>לולאה</u>: $|V|-1$ פעמים נבצע:
  - נעבור על כל הקשתות, ולכל קשת (u,v) נבצע $\text{Relax}(u,v)$
- <u>בודקים את תנאי הסיום</u>:
  - נבדוק האם לכל $(u,v)\in E$ מתקיים $d[v]\leq d[u]+w(u,v)$
  - אם זה לא מתקיים נחזיר "קיים מעגל שלילי"
  - אם זה כן מתקיים, נחזיר $\forall v: d[v],\pi[v]$

[הערה: אם באיטרציה כלשהי של הלולאה הראשית רואים שאף Relax אינו אפקטיבי, ניתן לעצור מיד ולסיים את ריצת האלגוריתם, שכן גם אם נמשיך את ריצתו הערכים לא ישתנו.]

דוגמה



נריץ על יבש את האלגוריתם – נניח שנעבור על הקשתות לפי הסדר (משמאל לימין):

$$(a,c),(b,c),(s,a),(b,a),(s,b)$$

[[סדר המעבר על הקשתות, או סדר הבדיקה שלהם בתנאי הסיום, חסר כל חשיבות – ניתן לבחור איזה סדר שרוצים.]]

	$d[s]$	$d[a]$	$d[b]$	$d[c]$
אתחול	0	∞	∞	∞
שלב 1	0	1	11	∞
שלב 2	0	-9	11	2
שלב 3	0	-9	11	-8

לאחר שלב 3 כבר כל פעולות ה- Relax אינן אפקטיביות ועוצרים.

הוכחת נכונות

טענה [האלגוריתם מגלה מעגל שלילי]

אם יש מעגל שלילי, האלגוריתם מגלה זאת.

<u>הוכחה:</u>

לפי משפט נכונות האלגוריתם הגנרי, תנאי הסיום מתקיים רק אם חישבנו נכון את $d[v]=\delta(s,v)$ לכל v. אם יש מעגל שלילי אז לכל קדקוד v במעגל, $\delta(s,v)<r$ לכל r סופי, ולכן לא יכול להתקיים $d[v]=\delta(s,v)$.

[[ניתן לחזור על המעגל כמה פעמים שנרצה עד שנקבל ערך נמוך יותר מ- r כלשהו שנבחר.]]

□

טענה [כשאין מעגל שלילי יש מק"ב פשוט]

אם אין מעגל שלילי, לכל קדקוד v קיים מק"ב מ-s ל-v שהוא מסלול פשוט.

<u>הסבר</u>: אם מסלול ל-v מכיל מעגל, אפשר למחוק את המעגל מהמסלול ולקבל מסלול לא יותר יקר/ארוך.

הוכחת נכונות כשאין מעגל שלילי

- <u>משפט</u>: אם אין מעגלים שליליים, אלגוריתם B-F [[Bellman-Ford =]] מחזיר ערכים $\forall v:\ d[v]=\delta(s,v)$.
- <u>טענת עזר</u>: לכל קדקוד $v\in V$, אם יש מק"ב מ-s ל-v עם k קשתות אזי בסוף הצעד ה-k בלולאה יתקיים $d[v]\le\delta(s,v)$.

<u>הוכחת המשפט</u>:

לכל v, לפי טענה 2, קיים מק"ב עם $|V|-1\ge k$ צלעות, ולפי טענת העזר, בסוף הצעד ה-k באלגוריתם מתקיים $d[v]\le\delta(s,v)$.

כיוון שערכי $d[\cdot]$ לא יורדים, מתקיים $d[v]\le\delta(s,v)$ גם בסוף האלגוריתם – אבל $d[v]\ge\delta(s,v)$ תמיד (באלגוריתם הגנרי), ולכן מתקיים $d[v]=\delta(s,v)$.

[את טענת העזר נוכיח אחרי חופשת פסח, כשכבר לא נזכור כלום. ☺]