המשך בעיית התרמיל (ולא הגנב)

נזכיר את מה שעשינו בשיעור שעבר.

תזכורת

הבעיה:

- מופע: איברים עם משקלים $w_1,\dots,w_n$ וערכים $v_1,\dots,v_n$ וכן קיבולת W.
- פתרון חוקי: קבוצה $I \subseteq \{1,\dots,n\}$ ממשקל $w(I)=\sum_{i\in I} w_i \le W$.
- יש למצוא: קבוצה חוקית I עם ערך $v(I)=\sum_{i\in I} v_i$ מקסימלי.

הגדרנו תת-בעיות:

לכל $k\in\{0,\dots,n\}$ וְ- $U\in\{0,\dots,W\}$:

$$\mathcal{T}(k,U) := \{I \subseteq \{1,\dots,k\} \mid w(I)\le U\}$$

$$OPT(k,U) := \text{ערך פתרון טוב ביותר ב-}\mathcal{T}(k,U)$$

הבעיה המקורית היא מציאת $OPT(n,W)$.

פיתחנו נוסחה:

$$OPT(k,U)=\begin{cases} 0, & k=0 \\ OPT(k-1,U), & k>0 \wedge w_k > U \\ \max\{OPT(k-1,U), v_k + OPT(k-1,U-w_k)\} & k>0 \wedge w_k \le U \end{cases}$$

המשך

מקרה ב' + הוכחת טענה 2

- מקרה א': פתרון $\mathcal{T}(k,U)$ **שלא** כולל את k.
  - טענה 1: פתרון אופטימלי מסוג זה, ערכו $OPT(k-1,U)$.
- מקרה ב': פתרון $\mathcal{T}(k,U)$ **שכן** כולל את k.
  - טענה 2: פתרון אופטימלי כזה, ערכו הוא $v_k + OPT(k-1,U-w_k)$.

    הוכחה:

    האיבר ה- k תורם ערך v_k לפתרון.

    בכל פתרון מסוג זה, שאר האיברים בפתרון שייכים ל- $\{1,\dots,k-1\}$ (לפי הגדרה).

    אז פתרון כזה הוא מהצורה $Q\cup\{k\}$ עבור $Q\subseteq\{1,\dots,k-1\}$, ומשקלו הכולל הוא:

לפי הגדרה

$$w(Q\cup\{k\}) = \underbrace{w_k + w(Q) \le U}$$

$$\Downarrow$$

$$w(Q) \le U - w_k$$

ולכן $Q \in \mathcal{T}(k-1, U-w_k)$.

כמו-כן, בחירת k לא מגבילה את האפשרות לבחור Q כזה (כל קבוצה $Q \in \mathcal{T}(k-1, U-w_k)$ ניתן להרחיב לפתרון חוקי $Q\cup\{k\}$ ממשקל $U \ge$).

לכן הערך המרבי שניתן לקבל עבור פתרון כזה הוא:

$$\max_{Q\in\mathcal{T}(k-1,U-w_k)}\left(v(Q\cup\{w\})\right) = \max_{Q\in\mathcal{T}(k-1,U-w_k)}\left(v_k + v(Q)\right) = v_k + \max_{Q\in\mathcal{T}(k-1,U-w_k)} v(Q) =$$
$$= v_k + OPT(k-1, U-w_k)$$

□

שלב 3: הגדרת סדר של התת-בעיות

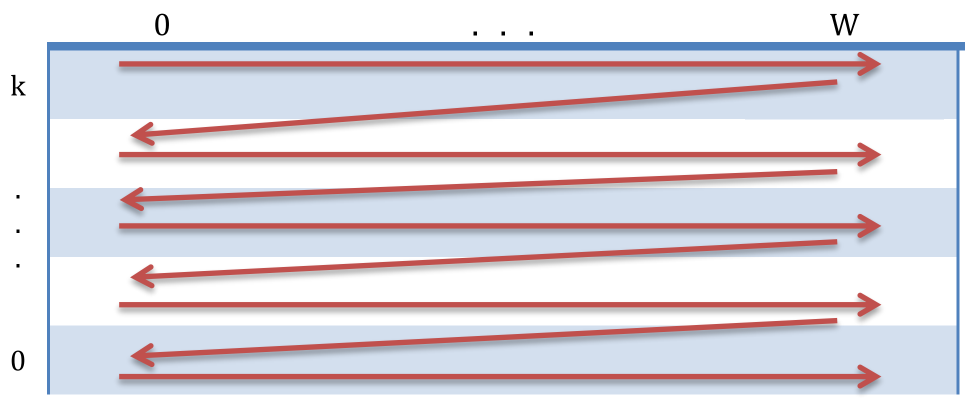
סדר חייב לכבד את הנוסחה:

הזוג (k,U) חייב להופיע **אחרי** הזוגות $(k-1,U)$ ו-$(k-1,U-w_k)$ שמופיעים בצד ימין.

[[כלומר, כיוון שע"מ לחשב את הזוג (k,U) צריך את $(k-1,U)$ או את $(k-1,U-w_k)$, אנו חייבים שכאשר נחשב את (k,U) כבר יהיו לנו את הערכים של $(k-1,U)$ ושל $(k-1,U-w_k)$, ולכן שני אלה צריכים להופיע לפני (k,U) בסדר החישוב.]]

<u>סדר אפשרי</u>: סדר לקסיקוגרפי:

$$(k_1,U_1) < (k_2,U_2)$$
אם $k_1 < k_2$
או $k_1 = k_2$ וְ- $U_1 < U_2$



[זהו לא הסדר היחיד שיעבוד; ניתן גם ללכת עמודה אחר עמודה במקום שורה אחר שורה.]

מימושים אפשריים

רקורסיה נאיבית: כבר ברור לנו שזה לא יהיה יעיל.

רקורסיה עם ממואיזציה (Memoization):

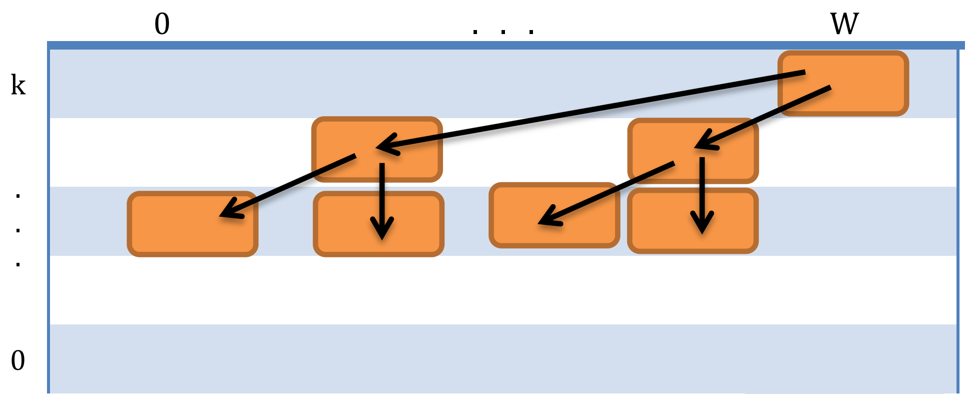
- אתחול:
  - נגדיר מערך $M[\cdot,\cdot]$ עם כניסות k,U $(0 \le k \le n,\ 0 \le U \le W)$.
  - נשים -1 בכל הכניסות.
  - נקרא ל-$Recurse(n,W)$.
- $Recurse(k,U)$:
  - אם $M[k,U] \ne -1$ נחזיר $M[k,U]$.
  - אחרת:
    - אם $k=0$ אזי $M[k,U] \leftarrow 0$.
    - אם $k>0$ וְ- $w_k > U$ נבצע:

$$M[k,U] \leftarrow Recurse(k-1,U)$$

    - אחרת נבצע:

$$M[k,U] \leftarrow \max\left\{Recurse(k-1,U), v_k + Recurse(k-1,U-w_k)\right\}$$

  - נחזיר את $M[k,U]$.



[שימו לב: ההבדל בין שיטה זו ל-bottom-up היא שכאן אנו לא מחשבים קודם את הערכים הנמוכים, אלא כן מתחילים מלמעלה, אך מחשבים ערך אך ורק אם לא חישבנו אותו בעבר.]

שחזור פתרון אופטימלי

- <u>אבחנה</u>: לפי טענה 1, עבור $k>0$ קיים פתרון אופטימלי ב-$\mathcal{T}(k,U)$ **שלא כולל את** $k \Leftrightarrow OPT(k,U)=OPT(k-1,U)$.
- <u>הערה</u>: אם $k>0$ וחישבנו את $M[k,U]$ (ב-$Recurse(k,U)$) אז חישבנו גם את $M[k-1,U]$ ולכן קיים פתרון אופטימלי ב-$\mathcal{T}(k,U)$ שלא כולל את $k \Leftrightarrow M[k,U]=M[k-1,U]$.
  אם תנאי זה לא מתקיים אז חישבנו את $M[k-1,U-w_k]$.

<u>שחזור פתרון</u>:

- אתחול:
  - $U \leftarrow W$
  - $k \leftarrow n$
  - $I \leftarrow \varnothing$
- כל עוד $k>0$, נבצע:
  - אם $M[k,U]=M[k-1,U]$ אז:
    - $k \leftarrow k-1$
  - אחרת:
    - $I \leftarrow I \cup \{k\}$
    - $k \leftarrow k-1$
    - $U \leftarrow U-w_k$

זמן ריצה

- <u>שחזור פתרון אופטימלי</u>: $O(n)$ [כי מבצעים n פעמים פעולות של $O(1)$]
- <u>חישוב OPT</u>:
  - <u>אתחול</u>: $O(n \cdot W)$
  - <u>רקורסיה</u>:

  זה נראה כמו $O(n \cdot W)$, אבל צריך לבדוק זאת בזהירות, שכן ייתכן שקריאה רקורסיבית עבור (k,U) מתבצעת מספר רב של פעמים.
  נעשה זאת בעזרת ניתוח פחת (amortized analysis).
  נשתמש בשיטת האסימונים (tokens):
  - כל קריאה תייצר מס' כלשהו של אסימונים.
  - כל קריאה תשלם אסימון עבור עצמה.
  - צריך לבדוק שמס' הקריאות $\leq$ למס' האסימונים שייצרנו.

  נעשה זאת כך:

- נייצר אסימון ונעבר אותו ל- $Recurse\left(n,W\right)$.
- כל קריאה $Recurse\left(k,U\right)$ תייצר 2 אסימונים בפעם הראשונה (כאשר $M\left[k,U\right]=-1$) ותעביר אותם לקריאות הרקורסיביות לפי הצורך.
- כל קריאה מתבצעת במקביל לשליחת אסימון, לכן כל קריאה יכולה לשלם עבור עצמה.

[שרטוט העברת האסימונים]

$\Leftarrow$ סה"כ מס' האסימונים:

$$1+2\cdot\left(n+1\right)\left(W+1\right)=\mathrm{O}\left(n\cdot W\right)$$

לכן הרקורסיה אכן לוקחת $\mathrm{O}\left(n\cdot W\right)$.

אבל כמה זה $\mathrm{O}\left(n\cdot W\right)$?

W יכול להיות אקספוננציאלי במס' הביטים שלו [[כלומר יתכן ש- $W=\mathrm{O}\left(2^n\right)$]].

למשל, אם:

$$\begin{array}{cccc} w_1 & \cdots & w_n & \underset{n^2\ \text{bits}}{W} \\ \underset{n^2\ \text{bits}}{v_1} & \cdots & \underset{n^2\ \text{bits}}{v_n} & \end{array}$$

אז אם נסמן $N=$ מספר הביטים, נקבל ש- $N=\Theta\left(n^3\right)$.

לכן:

$$W\approx 2^{n^2}=2^{\Theta\left(N^{\frac{2}{3}}\right)}$$

[בעיה זו נחשבת קשה כפי שהבעיה למציאת מסלול המילטון נחשבת קשה.
היא שייכת למשפחת בעיות קשות שנכיר לקראת סוף הקורס.
לעומת זאת, מציאת פתרון מקורב זו דווקא בעיה לא כ"כ קשה.
בפירוט נוסף:
למצוא פתרון עם ערך $=OPT\longrightarrow$ קשה.
למצוא פתרון עם ערך $\leq 0.99999\cdot OPT\longrightarrow$ קל (אפשר לעשות בזמן $\mathrm{O}\left(n^2\right)$).]

חידוד העקרונות של תכנון דינאמי

זוכרים את הפונקציה שנתנו בתור הדוגמה הראשונה בה עשינו אלגוריתם בתכנון דינאמי?
אם כן – למה?
היא הייתה די נוראית.

נראה עכשיו כמה דוגמאות נוספות, קצת יותר נעימות, בשביל לחזור על העקרונות שלמדנו.

בעיית מסלול המילטון

נזכיר את בעיית מציאת מסלול המילטון בגרף מכוון:

- <u>מופע</u>: גרף מכוון $G=(V,E)$.
- <u>יש למצוא</u>: תשובה בוליאנית (T / F) לשאלה: האם קיים מסלול פשוט שעובר בכל הקדקודים?
- <u>אלגוריתם נאיבי</u>: לעבור על כל הפרמוטציות של $V \Leftarrow$ זמן ריצה $\Omega(|V|!)$ [יתכן שעבור כל פרמוטציה נעשה חישוב של יותר מ-$\mathrm{O}(1)$, ואז יתכן שהסיבוכיות אף גדולה מ-$\mathrm{O}(|V|!)$].

<u>פתרון בתכנון דינאמי:</u>

1) פירוק לתת-בעיות

נגדיר לכל קבוצה $\varnothing \neq U \subseteq V$ ולכל $s,t \in U$:

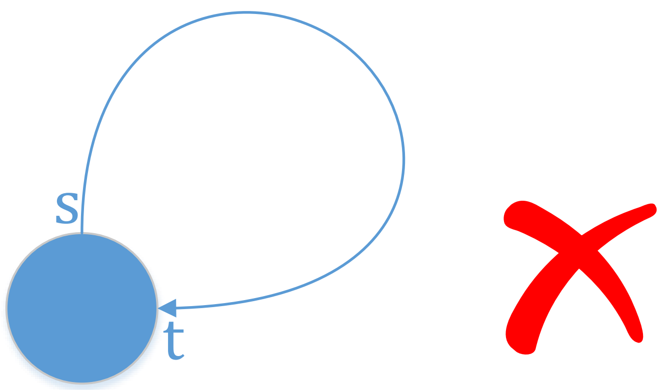
האם קיים מסלול פשוט מ-s ל-t שעובר בדיוק בקדקודי $P(U,s,t)=U$

הבעיה המקורית:

$$\bigvee_{s,t\in V} P(V,s,t)$$

[[זו פעולת "או" ($\vee$, OR) בין כל האפשרויות]]

- <u>מקרה קצה</u>: $s=t$.



$\Leftarrow P(U,s,t)=F$.

  - מקרה קצה למקרה קצה: $|U|=1$

$P(U,s,t)={\color{blue}T} \Leftarrow$

- מקרה כללי: $(s\neq t, |U|>1)$

אם הקשת הראשונה היא (s,v), שאר המסלול יהיה מ- v ל- t וישתמש בקדקודים $U\setminus\{s\}$.

$P(U,s,t)=\bigvee_{v:(s,v)\in E} P(U\setminus\{s\},v,t) \Leftarrow$

2) נוסחה

$$P(U,s,t)=\begin{cases} {\color{blue}T}, & |U|=1 \\ {\color{red}F}, & s=t\wedge|U|>1 \\ \underbrace{\bigvee_{v:(s,v)\in E} P(U\setminus\{s\},v,t)}_{\text{or } {\color{red}F} \text{ if there isn't such a } v} & \text{otherwise} \end{cases}$$

[סדר: נסדר את השלשות לפי גודל U ואז לפי סדר הקדקודים (או משהו כזה; זה לא כזה משנה – זוהי קבוצה סופית של תת-בעיות ואין בעיה להגדיר עליה סדר).]

מדוע זה לא עובד

מדוע פתרון זה לא פותר את בעיית מציאת מסלול המילטון בזמן פולינומיאלי?
כי מספר התת-בעיות אינו פולינומיאלי!
מס' האפשרויות לבחור את s ו- t הוא אמנם n^2, אך מס' הדרכים לבחור את U הוא $O\left(2^{|V|}\right)$.

בעיית Super Mario

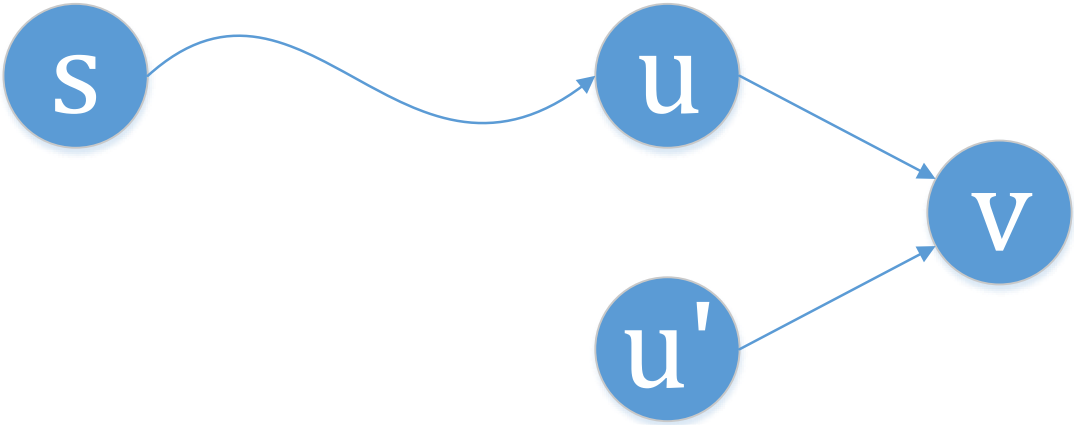
- מופע:
  - גרף מכוון $G=(V,E)$
  - לכל קשת $(u,v)\in E$ יש ערך $w(u,v)=$ מס' המטבעות שאפשר לאסוף באותו קטע
  - נקודת התחלה: $s\in V$
  - נקודת סיום: $t\in V$
- פתרון חוקי: מסלול מ- s ל- t
- יש למצוא: מסלול $p:s\rightsquigarrow t$ עם משקל $\sum_{(u,v)\in p} w(u,v)$ מקסימלי

תת-בעיות

- נגדיר: מסלול יקר ביותר מ- s ל- v = $OPT(v)$
- בעיה מקורית: $OPT(t)$

- <u>מס' התת-בעיות</u>: $O(|V|)$

נוסחה למקרה כללי



$$OPT(v)=\begin{cases}\dots\\ \max\limits_{u:(u,v)\in E}\{OPT(u)+w(u,v)\}\\ \dots\end{cases}$$

מה חסר בשביל לפתור את הבעיה?
חסר סדר על התת-בעיות.

למעשה, אם יש מעגל בגרף אז כלל אין פתרון בשיטה זו, שכן ניתן להמשיך ללכת במעגל ללא הגבלה ובכך להגדיל את המשקל הכולל כרצוננו.