11.11.2020 ***תכנון אלגוריתמים***

הרצאה 8: תכנון דנאמי

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

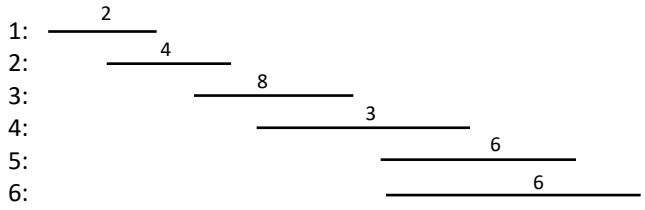
מרצה: אורי שטמר

בעית הפעילויות הממושקלת – שחזור פתרון

בפעם הקודמת ראינו איך למצוא משקל של פתרון אופטימלי, כלומר למצוא את $OPT(n)$. יתרה מכך, בסוף השיעור שעבר ראינו אלגוריתם אשר מחזיר מערך M ערך שעבור $0 \le j \le n$ מתקיים $M[j] = OPT(j)$.

עכשיו נראה איך בהינתן מערך M כנ"ל, נוכל לשחזר ממנו פתרון אופטימלי בזמן $O(n)$. (לא כולל זמן חישוב המערך...)

דוגמה:



M[0]=0, M[1]=2, M[2]=4, M[3]=10, M[4]=10, M[5]=16, M[6]=16

אלגוריתם שחזור:

אלגוריתם השחזור שנראה מבוסס על 2 האבחנות הבאות:

אבחנה 1: אם $OPT(j) = OPT(j-1)$ אזי קיים פתרון אופטימלי לתת הבעייה $\{1,2,\dots,j\}$ אשר לא מכיל את j

לכן, במקרה זה, אם מבקשים ממני לשחזר פתרון אופטימלי עבור תת הבעיה ה j, אז אני יכול (ברקורסיה) להחזיר פתרון אופטימלי לתת הבעיה ה $(j-1)$

אבחנה 2: אם $OPT(j) > OPT(j-1)$ אזי כל פתרון אופטימלי לתת הבעייה $\{1,2,\dots,j\}$ מכיל את j

מנוסחת המבנה אנחנו יודעים ש-

$$OPT(j) = \max\{OPT(j-1)\ ,\ w_j + OPT(p(j))\}$$

ולכן אם $OPT(j) \neq OPT(j-1)$ אזי $OPT(j) = w_j + OPT(p(j))$.

לכן, במקרה זה, אם מבקשים ממני לשחזר פתרון אופטימלי עבור תת הבעיה ה j, אז אני יכול (ברקורסיה) להחזיר פתרון אופטימלי לתת הבעיה ה $p(j)$ ולהוסיף לפתרון זה את הפעילות j (כמו שראינו בשיעור הקודם, הפתרון המתקבל חוקי).

Recunstruct(j)

- If $j = 0$ then return $\emptyset$
- If $M[j] \neq M[j-1]$ then return $\{j\} \cup \text{Recunstruct}\big(p(j)\big)$
- Else return $\text{Recunstruct}(j-1)$

<u>משפט:</u> נניח כי לכל $0 \leq j \leq n$ מתקיים $M[j] = OPT(j)$.
אזי $\text{Recunstruct}(n)$ מחזיר פתרון אופטימלי לבעיית הפעילויות הממושקלת, כלומר מחזיר פתרון $I \in Sol(n)$ במשקל $w(I) = OPT(n)$.

<u>הוכחה:</u>

נוכיח באינדוקציה על j שקריאה ל- $\text{Recunstruct}(j)$ מחזירה פתרון $I \in Sol(j)$ במשקל $w(I) = OPT(j)$.

<u>בסיס:</u> $j = 0$. הפתרון היחיד ב- $Sol(0)$ הטא הפתרון הריק. מהגדרת האלגוריתם מוחזרת קבוצה ריקה.

<u>צעד:</u>
נניח כי לכל $\ell < j$ מתקיים שקריאה ל- $\text{Recunstruct}(\ell)$ מחזירה פתרון $I \in Sol(\ell)$ במשקל $w(I) = OPT(\ell)$.

מקרה א: $M[j] \neq M[j-1]$. אזי, כפי שכתבנו בטקסט האדום מתחת לאבחנה 2, מתקיים

$$M[j] = OPT(j) = w_j + OPT\big(p(j)\big) = w_j + M[p(j)] \qquad \big((1)\big)$$

במקרה זה $\text{Recunstruct}(j)$ מחזיר $I = \{j\} \cup \text{Recunstruct}\big(p(j)\big)$.

נסמן $K = \text{Recunstruct}\big(p(j)\big)$. לפי הנחת האינדוקציה, K הוא פתרון אופטימלי עבור תת הבעיה ה- $p(j)$, כלומר $K \in Sol\big(p(j)\big)$ ו- $w(K) = OPT\big(p(j)\big)$.

נראה כי הפתרון המוחזר I הוא חוקי (כלומר $I \in Sol(j)$) ואופטימלי (כלומר $w(I) = OPT(j)$).

<u>חוקיות:</u>
K מכיל אוסף של פעילויות זרות מתוך $\{1,2,\dots,p(j)\}$. לפי הגדרת $p(j)$, הפעילות j נחתכת עם אף פעילות מ- $\{1,2,\dots,p(j)\}$. לכן $I = K \cup \{j\}$ מכיל אוסף של פעילויות זרות מתוך $\{1,2,\dots,j\}$, כלומר $I = K \cup \{j\} \in Sol(j)$

<u>משקל:</u>
משקל הפתרון המוחזר $I = K \cup \{j\}$ הוא

$$w(I) = w(K \cup \{j\}) = OPT\big(p(j)\big) + w_j = OPT(j)$$

כאשר השיוויון האחרון נובע מ- $\big((1)\big)$

מקרה ב: $M[j] = M[j-1]$. במקרה זה $\text{Recunstruct}(j)$ מחזיר $I = \text{Recunstruct}(j-1)$.
לפי הנחת האינדוקציה

$$I \in Sol(j-1) \subseteq Sol(j)$$

וגם

$$w(I) = OPT(j-1) = M[j-1] = M[j] = OPT(j)$$

מ.ש.ל.

ניתוח זמן ריצה:

הפרוצדורה $\text{Recunstruct}(j)$ מבצעת קריאה רקורסיבית אחת עם ערך ℓ, עבור $\ell < j$.
אז סה"כ יש לכל היותר n קריאות רקורסיביות.
כל קריאה (ללא עלות הקריאות הרקורסיביות במהלכה) היא בזמן $O(1)$.
לכן סה"כ זמן ריצה (לא כולל חישוב המערך) הוא $O(n)$.

מסקנה: סיבוכיות האלגוריתם כולל חישוב המערך ושחזור פתרון היא $O(n \cdot \log n)$.

הערות כליות לגבי תכנון דנאמי

- מתי אפשר להפעיל תכנון דנאמי?
  - יש מספר קטן של תתי בעיות
  - יש נוסחת מבנה שמקשרת בין ערכים של פתרונות אופטימלים לתתי הבעיות
  - יש סדר על תתי הבעיות

- שלבים בפיתוח אלגוריתם מבוסס על תכנון דנאמי:
  - הגדרה של תתי הבעיות ושל Sol ושל OPT
  - פיתוח נוסחת מבנה (כולל מקרי בסיס) והסבר כיצד מחשבים ערך פתרון אופטימלי
  - הוכחת נוסחת המבנה
  - הגדרת מערך שמכיל את OPT והצגת אלגוריתם לחישוב ערכי המערך
  - שחזור פתרון אופטימלי לאחר שהמערך הנ"ל חושב

בעיית התרמיל

נתונה חנות שבה יש n מוצרים. לכל מוצר יש משקל ומחיר. גנב פורץ לחנות ורוצה לגנוב מוצרים במחיר גבוה ביותר, אבל יכול רק לגנוב מוצרים שסך כל משקלם הוא לכל היותר T (זה הקיבולת של התרמיל שלו...)
* כאן אפשר או לקחת מוצר או לא לקחת (אי אפשר לקחת חלק ממוצר)

הגדרה פורמלית של הבעיה:

קלט:
- n מוצרים $\{1,2,\dots,n\}$. לכל מוצר i יש משקל w_i ומחיר p_i
- משקל תרמיל T (הנחה: T וכל אחד המשקלים w_i הם שלמים אי שליליים)

פתרון חוקי: אוסף $I \subseteq \{1,2,\dots,n\}$ של מוצרים כך שסכום המשקלים באוסף הוא לכל היותר T. כלומר,

$$w(I) = \sum_{i \in I} w_i \leq T$$

יש למצוא: פתרון חוקי I עם מחיר $p(I) = \sum_{i \in I} p_i$ מקסימלי

פתרון נאיבי:
ראשית נשים לב שאנחנו יכולים לפתור את הבעייה, כי יש מספר סופי של אפשרויות ואנחנו יכולים לנסות את כולם ולקבל אלגוריתם אקספוננציאלי...

נסיונות כושלים לתכנן אלגוריתם חמדן לבעייה:

רעיון 1: בכל שלב, נבחר מוצר עם מחיר מקסימלי מבין המוצרים שעוד אפשר להכניס לתרמיל (כלומר מבין המוצרים שאם נוסיף אותם לתרמיל אז לא נחרוג מהמשקל המותר T).

רעיון 2: בכל שלב, נבחר מוצר עם משקל מינימלי מבין המוצרים שעוד אפשר להכניס לתרמיל.

רעיון 3: בכל שלב, נבחר מוצר שעוד אפשר להכניס לתרמיל כך ש $\frac{p_i}{w_i}$ מקסימלי. (הרעיון הוא לנסות למקסם את הרווח ליחידת משקל)

דוגמה נגדית לרעיון 1:

$$T = 8$$
$$p_1 = w_1 = 1$$
$$p_2 = w_2 = 4$$
$$p_3 = w_3 = 4$$
$$p_4 = w_4 = 5$$

הפתרון האופטימלי הוא כמובן $\{2,3\}$ במחיר ומשקל 8. אבל האלגוריתם החמדן שהצענו יחזיר $\{4,1\}$ במחיר ומשקל 6.

דוגמה נגדית לרעיון 2: אותה דוגמה ממקודם. הפעם נחזיר פתרון $\{1,2\}$ במחיר ומשקל 5...

דוגמה נגדית לרעיון 3: בדוגמה האחרונה מתקיים ש- $\frac{p_i}{w_i} = 1$ לכל i ולכן האלגוריתם שהצענו יכול להיכשל.
דוגמה נוספת:

$$T = 12$$
$$p_1 = 20,\ w_2 = 10 \implies \frac{p_1}{w_1} = 2$$
$$p_2 = 11,\ w_2 = 6 \implies \frac{p_2}{w_2} = 1.8333$$
$$p_3 = 11,\ w_3 = 6 \implies \frac{p_2}{w_2} = 1.8333$$

הפתרון האופטימלי הוא כמובן $\{2,3\}$ במחיר 22 ובמשקל 12. אבל האלגוריתם שהצענו יבחר קודם אם מוצר 1 ויתקע...

כלומר, נסינו לתכנן אלגוריתן חמדן וזה לא עבד.
לא ידוע אלגוריתם חמדן לבעיה.

נראה אלגוריתם מבוסס תכנון דנאמי לבעיית התרמיל

הערה: בפתרון שלנו לא יהיה לנו צורך להניח שהמוצרים ממויינים (לפי מחיר או משקל). כלומר המוצרים ניתנו לנו כקלט לפי סדר מסויים, ועם הסדר הזה נעבוד. (במילים אחרות, אנחנו לא יודעים להרוויח משהו ממיון המוצרים...)

נסיון 1 (שלא יצליח):

ננסה ללכת בדרך דומה למה שעשינו בבעיית הפעילויות הממושקלת

הגדרת תתי בעיות:

עבור $0 \le j \le n$, תת הבעייה ה- j היא בעייה זהה לבעיה המקורית, מוגבלת ל- j המוצרים הראשונים.

הגדרת Sol:

$\boldsymbol{Sol(j)}$ = קבוצת כל הפתרונות החוקיים עבור תת הבעיה ה- j, כלומר $I \in Sol(j)$ אם

1. $I \subseteq \{1,2,\dots j\}$
2. $w(I) \le T$

הגדרת OPT:

$\boldsymbol{OPT(j)}$ = מחיר מקסימלי של פתרון חוקי עבור תת הבעייה ה- j, כלומר

$$OPT(j) = \max_{I \in Sol(j)} p(I)$$

ננסה לכתוב נוסחת מבנה:

$$OPT(j) = \max\left\{ \; OPT(j-1) \;\; , \;\; p_j + \textbf{?} \; \right\}$$

$OPT(j-1)$: לא מכיל מוצר j

$p_j + \textbf{?}$: מה נשים פה?

היינו רוצים לכתוב כאן

"מחיר מקסימלי של פתרון חוקי מתוך j-1 המוצרים הראשונים שמשקלי לכל היותר $T - w_j$"

אי אפשר לבטא זאת באמצעות OPT כמו שהגדרנו אותו...

הגדרה נכונה של תתי הבעיות:

לכל $0 \le j \le n$ ולכל $0 \le T' \le T$ נגדיר תת בעייה (j, T'):

מצא תת קבוצה של $\{1, 2, \dots, j\}$ כך שמשקלה הוא לכל היותר T' ומחירה מקסימלי

הגדרת Sol:

$\boldsymbol{Sol(j,T')}$ = קבוצת כל הפתרונות החוקיים עבור תת הבעיה ה- (j, T'), כלומר $I \in Sol(j,T')$ אם

1. $I \subseteq \{1,2,\dots j\}$
2. $w(I) \le T'$

הגדרת OPT:

$\boldsymbol{OPT(j,T')}$ = מחיר מקסימלי של פתרון חוקי עבור תת הבעייה ה- (j, T'), כלומר

$$OPT(j,T') = \max_{I \in Sol(j,T')} p(I)$$

נוסחת מבנה:

$$OPT(j,T') = \begin{cases} 0 & , \quad j = 0 \\ OPT(j-1,T') & , \quad j > 0 \;\&\; w_j > T' \\ \max\left\{ \; OPT(j-1,T') \; , \; p_j + OPT\left(j-1, T' - w_j\right) \right\} & , \quad j > 0 \;\&\; w_j \le T' \end{cases}$$