תכנון דינאמי (Dynamic Programming)

Super Mario

מכירים, נכון?
אם לא, לכו מפה. <_>

אחת המטרות של שחקנים במשחק זה הוא לצבור כמה שיותר מטבעות (מקבלים עוד פסילות מזה).

שלבי המשחק אינם לינאריים – לעתים ניתן לבחור ללכת בדרך אחת מתוך כמה, ובכל דרך יש כמות מטבעות כלשהי.

[גרף שמתאר שלב]

השאלה שנשאל: באיזו דרך כדאי ללכת ע"מ להשיג את מס' המטבעות המקסימלי?

ניתן לבחון את כל הדרכים האפשריות, אך אם למשל יש בכל נקודה 2 אפשרויות בחירה, מס' המסלולים יכול להיות אקספוננציאלי. אז זו לא דרך טובה במיוחד.

נציע משהו אחר:
בכל נקודה בגרף יש מס' מקסימלי של מטבעות שניתן להשיג עד שמגיעים אליו.
אז נחשב את הערך הטוב ביותר בכל נקודה בצורה סדרתית, החל מהנקודה הראשונה, כאשר כל פעם שאנו מגיעים לנקודה שיש מספר דרכים להגיע אליה, נבחן את כל הקדקודים מהם יש קשת לנקודה שלנו ונסמן את הערך של הנקודה ע"י בחירת המקסימום מתוך כל הערכים של הקדקודים הללו (תוך התחשבות בכמות המטבעות על הקשתות שמחברות אותן אליה).

דוגמה [פחות כיפית]

- <u>מופע</u>: סדרת מספרים אי-שליליים $a_1,\ldots,a_n$.
- יש <u>למצוא</u>: $f(n)$, כאשר:

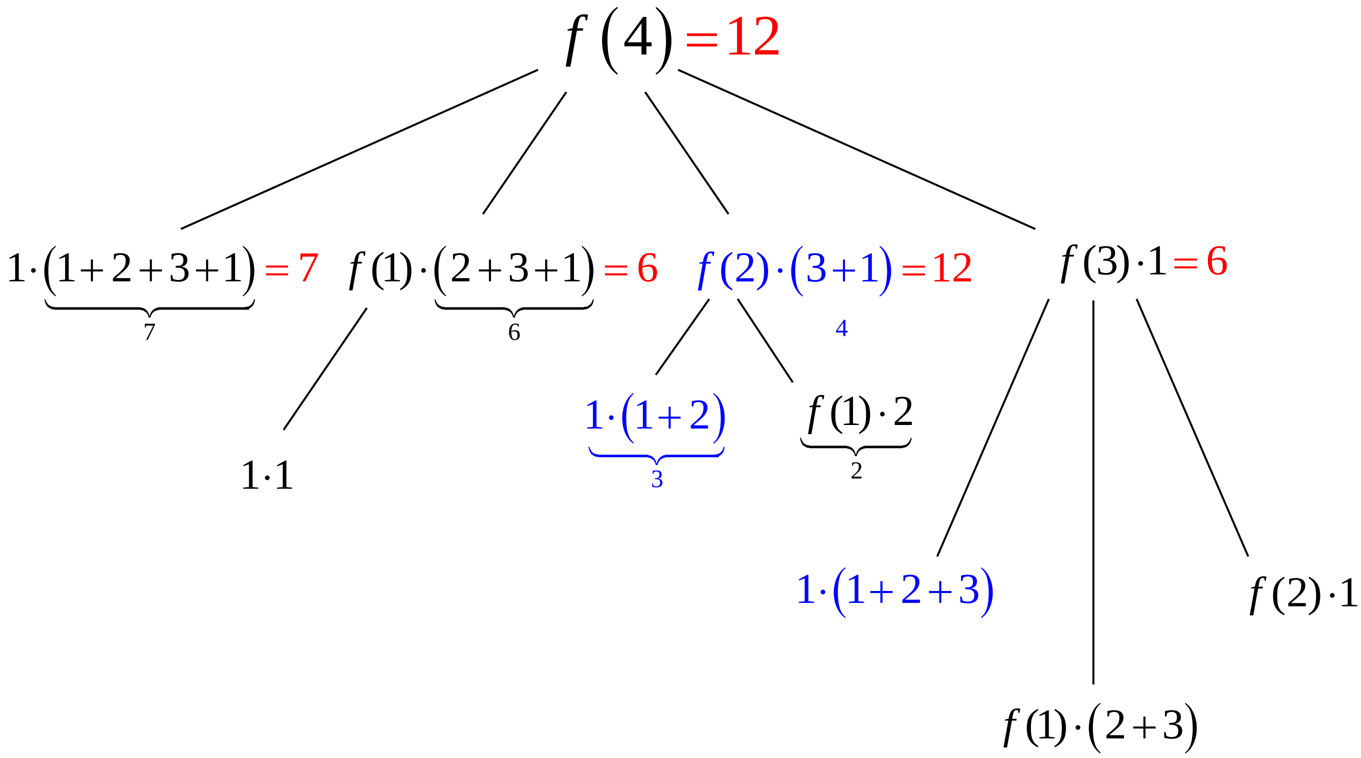
$$f(k)=\begin{cases} 1, & k=0 \\ \max\limits_{0\le i\le k-1}\left(f(i)\cdot\sum\limits_{j=i+1}^{n}a_j\right), & 1\le k\le n\end{cases}$$

דוגמה מספרית

ניקח את הסדרה:

1 2 3 1

אזי:

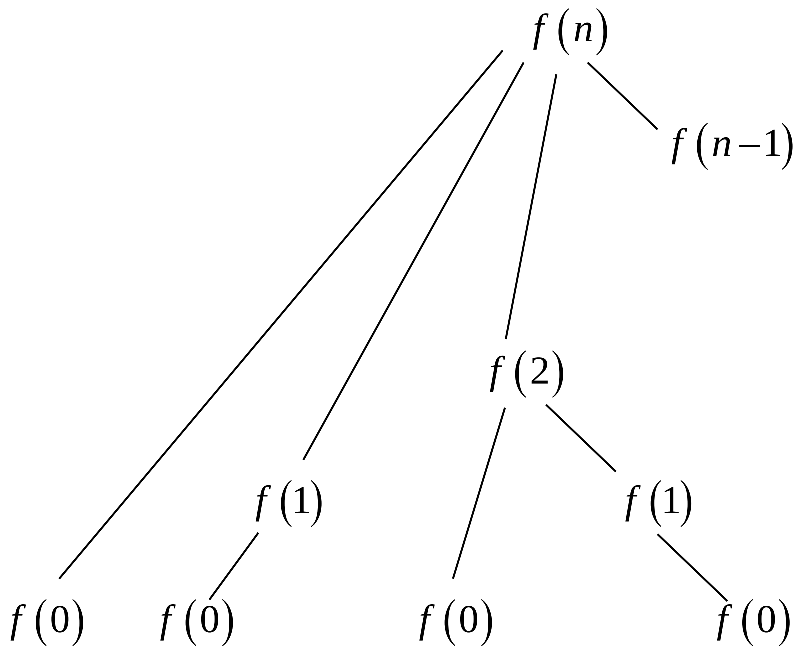


[משמעות הפונקציה: מקסימום שניתן לקבל ע"י חלוקת הסדרה לקטעים:

$$(a_1+\cdots+a_5)(a_{5+1}+\cdots)(\quad)$$

]

פתרון נאיבי: רקורסיה פשוטה



[בפתרון זה יש כ- 2^{n-1} קריאות ל- $f(0)$.]
$f(k)$ יבצע k קריאות רקורסיביות.
גודל עץ הרקורסיה הוא $O\left(2^n\right)$.

אלגוריתם איטרטיבי

חישוב bottom-up במקום top-down.

- <u>אתחול</u>: נגדיר מערך M עם כניסות [[כלומר אינדקסים]] $0,\dots,n$.
- <u>לולאה</u>: עבור $k=0$ ועד n נבצע:
  - אם $k=0$: $M[0]\leftarrow 1$
  - אחרת: $M[i]\leftarrow \max\limits_{0\le i\le k-1} M[i]\cdot\sum\limits_{j=i+1}^{k} a_j$
- <u>סיום</u>: נחזיר את $M[n]$.

הוכחת נכונות

- <u>משפט</u>: האלגוריתם מחזיר את $f(n)$.
- <u>טענת עזר</u>: בסוף האלגוריתם, $M[k]=f(k)$ לכל $k\in\{0,\dots,n\}$.

<u>הוכחת המשפט</u>: מיידית מטענת העזר.

<u>הוכחת טענת העזר</u>:
באינדוקציה על k.

- <u>$k=0$</u>:
בצעד הראשון $M[0]\leftarrow 1$, וערך זה לא משתנה עד סוף האלגוריתם.

- <u>צעד אינדוקטיבי</u>:
נניח ש- $M[i]=f(i)$ לכל $0\le i\le k-1$ **בסוף האלגוריתם**.
ערכים אלה חושבו לפני $M[k]$, ולא שונו עד סוף האלגוריתם, לכן **גם בתחילת הצעד ה-** k מתקיים:

$$M[i]=f(i)\qquad\qquad \forall 0\le i\le k-1$$

לכן בזמן חישוב $M[k]$ מתקיים:

$$\max_{0\le i\le k-1} M[i]\sum_{j=i+1}^{k} a_j=\max_{0\le i\le k-1} f(i)\sum_{j=i+1}^{k} a_j$$

וערך זה שווה ל- $f(k)$.

כלומר, בצעד ה- k מתבצע $M[k] \leftarrow f(k)$ וערך זה לא משתנה עד סוף האלגוריתם.

□

זמן ריצה

- יש $n+1$ צעדים.
- בצעד ה- k צריך לקרוא את $M[0],\dots,M[k-1]$ ($O(k)$)
- צריך לחשב את הסכומים:

$$\begin{aligned}&a_k\\&a_{k-1}+a_k\\&a_{k-2}+a_{k-1}+a_k\\&\vdots\\&a_1+\cdots+a_{k-2}+a_{k-1}+a_k\end{aligned}$$

כמה זמן לוקח לעשות זאת?

ניתן לעשות זאת נאיבית, ואז זה $O(k^2)$, או שאפשר לעשות זאת גם ע"י תכנון דינאמי, להתחיל מהסכום S_k ולרדת למטה:

$$\begin{aligned}S_k &= a_k &&= S_k\\S_{k-1} &= a_{k-1}+a_k &&= S_k+a_{k-1}\\S_{k-2} &= a_{k-2}+a_{k-1}+a_k &&= S_{k-1}+a_{k-2}\\&\vdots &&\vdots\\S_1 &= a_1+\cdots+a_{k-2}+a_{k-1}+a_k &&= S_2+a_1\end{aligned}$$

זאת ניתן לעשות ב-$O(k)$.

- יש למצוא את המקסימום בין הערכים המתוארים בנוסחה ($O(k)$).
- סה"כ: $O(1+2+\cdots+n)=O(n^2)$

רכיבים עיקריים בתכנון דינאמי

- פירוק לתת-בעיות (בדוגמה: $f(0), f(1),\dots, f(n)$)
- מס' התת-בעיות צריך להיות **<u>פולינומי</u>**
- הפתרון צריך להיות = אחת התת-בעיות או חישוב קל שמסתמך עליהן
- **<u>נוסחת נסיגה</u>** שמתארת את הקשר ביניהן
- מקרה קצה בנוסחת נסיגה (בדוגמה: $k=0$)
- סדר של תת-בעיות כך שהביטויים הרקורסיביים בצד ימין יבואו לפני הביטוי שמחושב בצד שמאל (בדוגמה: $f(\underline{\underline{k}})=\begin{cases}\cdots & \cdots\\ \max\limits_{j\underline{\underline{<k}}}(\cdots) & \cdots\end{cases}$)

הדברים החשובים ביותר כאן הם ניסוח נכון של **נוסחת הנסיגה** ווידוא שמס' התת-בעיות הוא **פולינומי**.

בעיית הפעילויות הממושקלות

- מופע: סדרה של קטעים פתוחים $(s_1, f_1), \ldots, (s_n, f_n)$ ולכל קטע i, משקל w_i.
- פתרון חוקי: כמקודם, $I \subseteq \{1, \ldots, n\}$ כך שהקטעים $\{(s_i, f_i) | i \in I\}$ זרים.
- יש למצוא: פתרון חוקי I בעל משקל $\sum_{i \in I} w_i$ מקסימלי.

אם כל ערכי המשקל שווים אז זוהי אותה הבעיה שפתרנו בעבר.
לא ידוע על אלגוריתם חמדן שפותר בעיה זו; במקום, נראה אלגוריתם שמשתמש בתכנון דינאמי.

תכנון האלגוריתם

- שלב 1: פירוק לתת-בעיות
  - נמיין את הקטעים לפי זמני סיום כך ש- $f_1 \le f_2 \le \cdots \le f_n$.
  - לכל $k \le n$, נסתכל על התת-בעיה שכוללת רק את הקטעים $1, \ldots, k$.
  - נגדיר $T(k) = \{1, \ldots, k\}$ (וְ- $T(0) = \varnothing$).
  - נגדיר $OPT(k) =$ ערך הפתרון הטוב ביותר המוכל ב- $T(k)$.
    [כלומר המשקל $\sum_{i \in O} w_i$ של הפתרון הטוב ביותר $O \subseteq T(k)$.
    כמובן, $OPT(0) = 0$.]

  מספר התת-בעיות וזיהוי הבעיה המקורית:
  סה"כ מספר התת-בעיות פולינומי $(n+1)$ והבעיה המקורית $= OPT(n)$.
  [אנו נבנה באלגוריתם את הערך המספרי $OPT(n)$ ונראה שתוך-כדי נבנה גם את הקבוצה הרצויה.]

- שלב 2: מציאת נוסחת הנסיגה
  חלוקה למקרים של פתרונות ל- $OPT(k)$.
  [**שימו לב**: זהו אחד המקומות שאנשים הכי נוטים לטעות בהם, כי מפספסים מקרים.
  ע"מ לוודא שלא מפספסים דבר, מומלץ להכין טבלה כפי שנראה כאן.]

  טבלה עבור $T(k)$:

האם הקטע שייך לפתרון?	נוסחה
כן	
לא	

  נמלא את טבלה זו תוך כדי בניית האלגוריתם.

  - מקרה א': פתרון אופטימלי ל- $T(k)$ **לא כולל** את k.

<u>טענה 1</u>: במקרה זה, פתרון אופטימלי יהיה בעל הערך $OPT\left(k-1\right)$.

<u>הוכחה</u>: מיידי מהגדרת $OPT\left(\ \right)$, כי אי-בחירת הקטע k לא מגבילה את האפשרויות לבחור קטעים נוספים מתוך $T\left(k-1\right)$.

[זה נראה ממש ברור כאן, אבל בעתיד נראה מקרים שבהם זה לא ברור, ולכן חשוב לציין את זה.]

[[נמלא בטבלה:

האם הקטע שייך לפתרון?	נוסחה
כן	
לא	$OPT\left(k-1\right)$

]]

- <u>מקרה ב'</u>: פתרון אופטימלי ל-$T(k)$ **כן כולל** את הקטע k.

נתבונן במופע של הבעיה שלנו:

[שרטוט קטעים וזמנים]

[אם הקטע k מופיע בפתרון האופטימלי, אז אף קטע שנחתך עמו לא מופיע.
כיוון שכבר מיינו את הקטעים, k הוא הקטע שנגמר אחרון מתוך הקטעים ב-$T(k)$, לכן הקטעים שלא נחתכים עם k הם כל אלה ב-$T(k)$ שנגמרים לפני s_k (ושוב, כיוון שהקטעים ממוינים, ניתן למצוא את הקטע האחרון שנגמר לפני s_k ולהגביל את הקטעים לכל מי שנמצא עד אליו).]

במקרה זה ניתן להוסיף רק קטעים שמסתיימים עד זמן s_k.

נגדיר:

$$p(k) := \text{(0 אם לא קיים) } s_k \text{ הקטע האחרון שמסתיים עד } = \max\left(\left\{0\right\} \cup \left\{j \middle| f_j \le s_k\right\}\right)$$

[בדוגמה הנ"ל, $p(k) = k-3$]

<u>טענה 2</u>: פתרון אופטימלי מסוג זה הוא בעל ערך:

$$w_k + OPT\left(p(k)\right)$$

<u>הוכחה</u>:

הקטע k תורם w_k לפתרון מסוג זה.

לפי סדר המיון, כל קטע נוסף בפתרון מסתיים עד זמן f_k, ולכן כדי לא ליצור חפיפה, כל קטע נוסף מסתיים עד s_k.

לפי סדר המיון, זה אומר שמותר להוסיף בדיוק קטעים מתוך $T\left(p(k)\right) = \left\{1, \dots, p(k)\right\}$.

דהיינו, פתרון חוקי במקרה זה הוא מהצורה $\left\{k\right\} \cup Q$ עבור $Q \subseteq T\left(p(k)\right)$.

כמו-כן, בחירת k לא מגבילה את האפשרויות לבחור תת-קבוצה Q חוקית מתוך $T(p(k))$ (כלומר לכל קבוצה חוקית $Q \subseteq T(p(k))$ ניתן להרכיב פתרון חוקי $\{k\} \cup Q$). [שוב, **חשוב** לציין זאת!]

<u>מסקנה</u>: הערך המרבי שניתן לקבל עבור פתרון כזה הוא בדיוק:

$$\max_{\substack{Q\subseteq T(p(k)),\\ Q \text{ is valid}}}\left(\sum_{i\in Q\cup\{k\}} w_i\right) = w_k + \underbrace{\max_{\substack{Q\subseteq T(p(k)),\\ Q \text{ is valid}}}\left(\sum_{i\in Q} w_i\right)}_{OPT(p(k))} = w_k + OPT(p(k))$$

[[נמלא בטבלה:

האם הקטע שייך לפתרון?	נוסחה
כן	$w_k + OPT(p(k))$
לא	$OPT(k-1)$

]]

<u>מסקנה</u>:
הפתרון הטוב ביותר יהיה הטוב מבין שתי האפשרויות, ולכן, מטענה 1+2 נובע כי לכל $k>0$ מתקיים:

$$OPT(k) = \max\left\{OPT(k-1), w_k + OPT(p(k))\right\}$$

[נשים לב שהביטוי $OPT(k-1)$ חסר משמעות עבור $k=0$ (ולכן אמרנו "לכל $k>0$"), וזהו מקרה הקצה שלנו.]
<u>מקרה קצה</u>: $k=0 \Rightarrow OPT(k)=0$

לכן:

$$OPT(k) = \begin{cases} 0 & k=0 \\ \max\left\{OPT(k-1), w_k + OPT(p(k))\right\} & \text{otherwise} \end{cases}$$

למה צריך לציין "(אי)-בחירת k לא מגבילה את שאר הפתרון"?

<u>נעלה רעיון לבעיה קצת שונה</u>: בעיית פעילויות ממושקלות עם r קטעים **בדיוק**.

[בבעיה זו, אם אנו כן / לא בוחרים את קטע k, יש בכך השפעה על שאר הפתרון.]
כן בוחרים את $k \Leftarrow$ מגביל את האפשרות לבחור $Q \subseteq T(p(k))$, שכן צריך לבחור Q בגודל $r-1$
[לכן לא ניתן סתם למצוא פתרון לשאר ולהתעלם מכך, אלא יש לזכור שאנו תלויים בעוד פרמטרים].