

## הרצאה 7: תכנון דנאמי

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

מרצה: אורי שטמר

תכנון דנאמי היא טכניקה להפוך אלגוריתם רקורסיבי אקספוננציאלי לאלגוריתם פולינומי יעיל.

אינטואיציה: אם יש לנו אלג' רקורסיבי אקספ' שהרבה מהקריאות שלו הם אותו דבר, אז לא נצטרך לבצע אותם מחדש כל פעם אלא נשמור בזיכרון את הקריאות הקודמות. זה לא תמיד עובד...

### בעית הפעילויות הממושקלת

רוצים לבחור אוסף של פעילויות שלא חותכות אחת את השניה. הפעם הקודמת (כשדיברנו על אלגוריתמים חמדניים) לא התייחסנו למשקל של הפעילויות. הפעם רוצים אוסף של פעילויות זרות שהמשקל שלהן מקסימלי. (זאת הכללה של הבעיה הקודמת כאשר בבעיה הקודמת כל המשקולות היו 1).

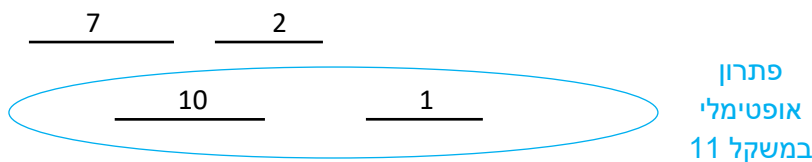
#### הגדרה פורמלית:

קלט: אוסף של  $n$  קטעים (פעילויות). לכל קטע זמן התחלה  $s_i$ , זמן סיום  $f_i$  (כאשר  $f_i > s_i$ ) ומשקל  $w_i$ .

פתרון חוקי: אוסף זר של פעילויות, כלומר לכל זוג פעילויות באוסף  $i_1, i_2$  מתקיים: או  $f_{i_1} \leq s_{i_2}$  או  $f_{i_2} \leq s_{i_1}$ .

יש למצוא: פתרון חוקי  $I$  שסכום משקלי הפעילויות באוסף הוא מקסימלי:  $w(I) = \sum_{i \in I} w(i)$

#### דוגמה:



**שאלה:** האם תמיד קיים פתרון?

**תשובה:** כן! תמיד קיים פתרון חוקי (למשל פתרון המכיל רק קטע אחד) ולכן תמיד קיים פתרון חוקי במשקל מקסימלי.

**שאלה:** האם האלגוריתם החמדן שהיה לנו ללא המשקולות עובד כאן?

**תשובה:** לא. למשל, הוא נכשל בדוגמה האחרונה... לא ידוע אלגוריתם חמדן לבעיה זו.

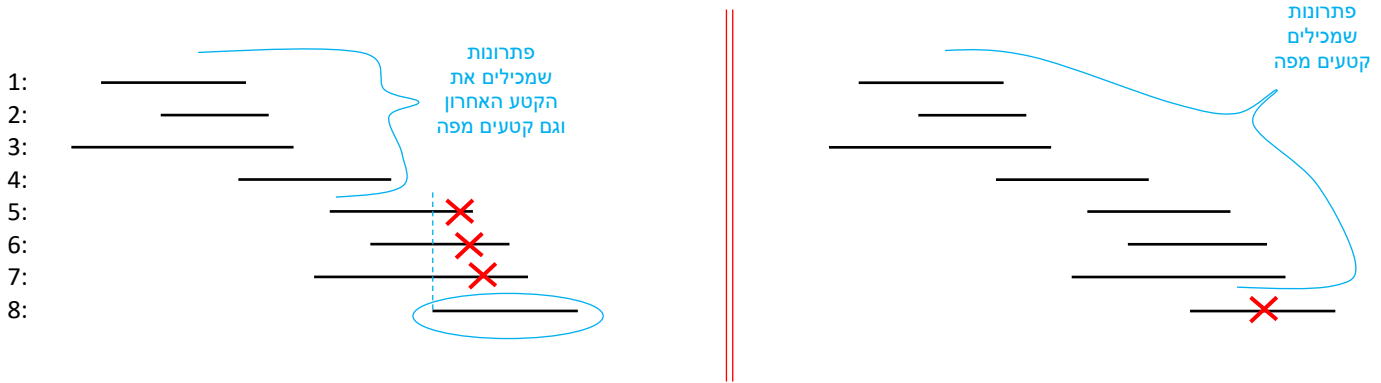
היום: נראה רק איך למצוא משקל של פתרון אופטימלי

**הנחה:** נמנין את הקטעים עפ"י סדר סיום. נניח  $f_1 \leq f_2 \leq \dots \leq f_n$ .  
 כלומר: מעכשיו לאורך כל הבעיה נניח כי הקטעים נתונים לנו ממויינים לפי סדר זמני סיום.

**הרעיון:** נחשוב על החלוקה הבאה של הפתרונות החוקיים:

- פתרונות שמכילים את הקטע ה-  $n$
- פתרונות שלא מכילים את הקטע ה-  $n$

**דוגמה:** איך נראית החלוקה שלנו לקבוצות?



אנחנו הולכים לתכנן אלגוריתם רקורסיבי לבעיה שלנו. איך בד"כ עובד אלג' רקורסיבי? הוא מגדיר ופותר תתי בעיות של הבעיה שלו, ואח"כ משתמש בפתרון שלהם כדי לפתור את הבעיה הנוכחית. איזה תתי בעיות אני צריך לפתור במקרה שלנו כדי לפתור את הבעיה המקורית?

**נגדיר:** לכל  $1 \leq j \leq n$ :

תת הבעיה ה- $j$  היא בעיה לבעיה המקורית, מלבד שהבחירה מוגבלת לקטעים  $1, 2, \dots, j$  במיון.

**הגדרה:**

$Sol(j) =$  קבוצת כל הפתרונות החוקיים עבור תת הבעיה ה- $j$ , כאשר  $Sol(0) = \emptyset$ .

כלומר,

$$Sol(j) = \{I \subseteq \{1, 2, \dots, j\} : I \text{ פתרון חוקי לבעית הפעילויות הממושקלת}\}$$

**הגדרה:**

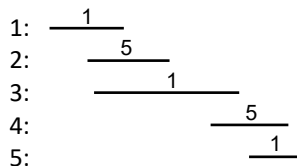
$OPT(j) =$  משקל פתרון אופטימלי עבור תת הבעיה ה- $j$ , כלומר משקל פתרון אופטימלי עבור  $j$  הקטעים הראשונים במיון.

כלומר

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

**אנחנו מחפשים את  $OPT(n)$ .**

**דוגמה:**



$$Sol(4) = \{\emptyset, \{1\}, \{2\}, \{3\}, \{4\}, \{1,4\}, \{2,4\}\}$$

$$OPT(4) = 10$$

**סימון:** עבור  $1 \leq j \leq n$  נסמן ב-  $p(j)$  את האינדקס של הקטע המקסימלי במיון שמסתיים לכל המאוחר בזמן  $s_j$ . כלומר,

$$p(j) = \max\{i : f_i \leq s_j\}$$

למשל, בדוגמה האחרונה מתקיים ש-  
 $p(5) = 3, \quad p(4) = 2, \quad p(3) = p(2) = p(1) = 0$

עכשיו אנחנו רוצים לקשר את  $OPT(j)$  ל- $OPT$ -ים קטנים יותר.

### טענה (נוסחת מבנה):

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

$$OPT(0) = 0$$

שימו לב: הנוסחה הזאת זה לא אלגוריתם ואין פה שום רקורסיה. זאת נוסחה מתמטית שמדברת על שיויון בין מספרים.

אז קודם הגדרנו את  $OPT(j)$  בתור  $OPT(j) = \max_{I \in Sol(j)} \{w(I)\}$  ועכשיו אנחנו טוענים שהערכים  $OPT(j)$  האלה מקיימים איזשהו קשר מתמטי. זה אולי נראה ברור (לאור הציוורים למעלה), אבל זה ממש לא מיידי וצריך להוכיח את זה.

### הוכחת נוסחת המבנה:

הערה: ההוכחה היא לא באינדוקציה. אנחנו מראים שיויון בין מספרים.

עבור  $j = 0$  הטענה טריוויאלית. נקבע  $j > 0$ .

עלינו לנתח את  $OPT(j)$ , כלומר את המשקל של פתרון אופטימלי עבור  $j$  הקטעים הראשונים במיון. נזכור ש-

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

ולכן כדי לנתח את  $OPT(j)$  נרצה לנתח את  $Sol(j)$ , כלומר נרצה להבין איך יכול להראות פתרון חוקי מתוך  $j$  הפעילויות הראשונות במיון.

**אבחנה 0:** כל פתרון ב-  $Sol(j)$  או מכיל את הקטע ה- $j$  או לא מכיל את הקטע ה- $j$ .

**אבחנה 1:** קבוצת הפתרונות ב-  $Sol(j)$  שלא כוללים את  $j$  היא בדיוק  $Sol(j-1)$ . כלומר,  
 $\{I \in Sol(j) : j \notin I\} = Sol(j-1)$

**הסבר:** נסתכל על כל הפתרונות החוקיים שמוכלים ב-  $\{1, 2, \dots, j\}$  ולא מכילים את  $j$ . כלומר אלו כל הפתרונות החוקיים שמוכלים ב-  $\{1, 2, \dots, j-1\}$ , כלומר אלו בדיוק כל הפתרונות ב-  $Sol(j-1)$ .

**טענה 2:** הפתרונות ב-  $Sol(j)$  שכן כוללים את  $j$  הם בדיוק כל הפתרונות מהסוג  $J \cup \{j\}$  עבור  $J \in Sol(p(j))$ , כלומר,

$$\{I \in Sol(j) : j \in I\} = \{J \cup \{j\} : J \in Sol(p(j))\}$$

זאת טענה אינטואיטיבית, אבל היא דורשת הוכחה.

## הוכחת טענה 2: נראה הכלה בשני הכיוונים.

**כיוון ראשון:** יהי  $K \in \{J \cup \{j\} : J \in \text{Sol}(p(j))\}$ . כלומר  $K$  מכיל אוסף של פעילויות זרות מתוך  $\{1, 2, \dots, p(j)\}$  ובנוסף מכיל את הפעילות  $j$ , אשר לפי הגדרת  $p(j)$  אינה נחתכת עם אף פעילות מ- $\{1, 2, \dots, p(j)\}$ . לכן  $K$  מכיל אוסף של פעילויות זרות מתוך  $\{1, 2, \dots, j\}$ , כלומר  $K \in \text{Sol}(j)$  ובנוסף  $j \in K$  ולכן  $K \in \{I \in \text{Sol}(j) : j \in I\}$

**כיוון שני:** יהי  $K \in \{I \in \text{Sol}(j) : j \in I\}$ . בפרט  $j \in K$ . נגדיר  $\tilde{K} = K \setminus \{j\}$ . מכיון ש- $K$  הוא פתרון חוקי המכיל את  $j$ , מהגדרת  $p(j)$  אנו יודעים ש- $K$  לא מכיל אף אחת מהפעילויות  $1, 2, \dots, j-1$ ,  $p(j)+1, p(j)+2, \dots$  כי פעילויות אלה מתנגשות עם פעילות  $j$ . לכן  $\tilde{K}$  מוכל בקטעים  $1, 2, \dots, p(j)$  והוא פתרון חוקי (כי  $K$  חוקי). לכן

$$K \in \{J \cup \{j\} : J \in \text{Sol}(p(j))\}$$

מ.ש.ל. (טענה 2)

עכשיו נוכל להשלים את הוכחת נוסחת המבנה בעזרת הטענה והאבחנות:

$$OPT(j) \stackrel{\text{הגדרת } OPT}{=} \max_{I \in \text{Sol}(j)} \{w(I)\} \stackrel{\text{אבחנה 0}}{=} \max \left\{ \max_{\substack{I \in \text{Sol}(j) \\ j \notin I}} \{w(I)\}, \max_{j \in I} \{w(I)\} \right\}$$

$$\stackrel{\text{טענה 2 ואבחנה 1}}{=} \max \left\{ \max_{I \in \text{Sol}(j-1)} \{w(I)\}, \max_{J \in \text{Sol}(p(j))} \{w(J \cup \{j\})\} \right\}$$

$$\stackrel{\text{הגדרת } w(\cdot)}{=} \max \left\{ \max_{I \in \text{Sol}(j-1)} \{w(I)\}, w_j + \max_{J \in \text{Sol}(p(j))} \{w(J)\} \right\}$$

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

מ.ש.ל. (נוסחת המבנה)

אז הוכחנו את נוסחת המבנה. עכשיו אנחנו רוצים להשתמש בה כדי לתכנן אלגוריתם שמחשב את  $OPT(n)$ , כלומר את ערך פתרון אופטימלי עבור הקטעים  $\{1, 2, \dots, n\}$ .

## נתחיל עם אלגוריתם רקורסיבי לא יעיל:

**C\_OPT(j):**

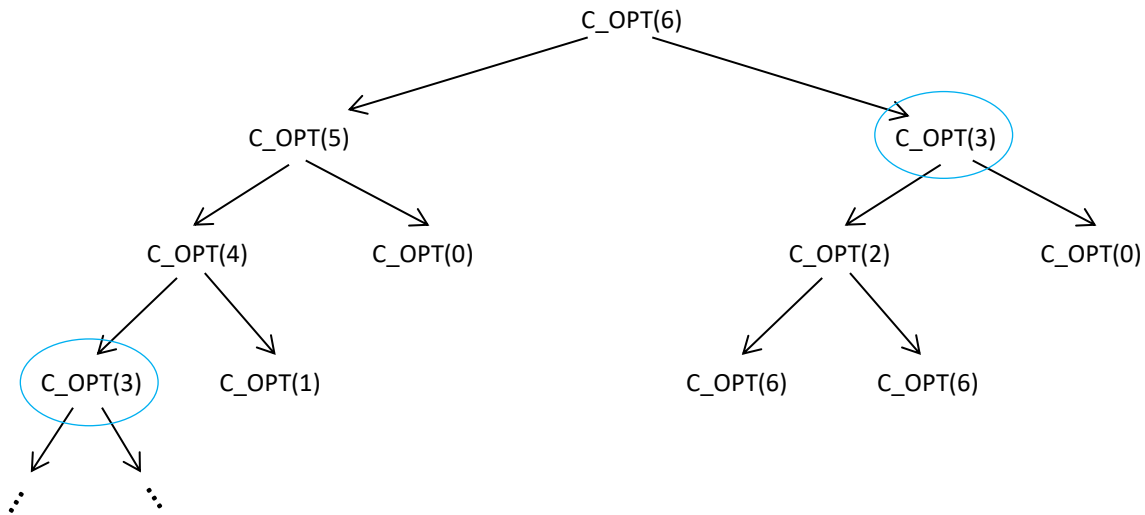
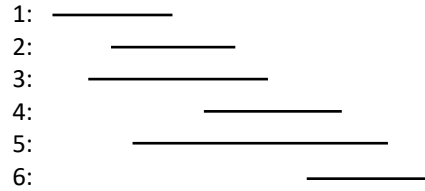
if  $j=0$  return 0

else return  $\max\{ C\_OPT(j-1), w_j + C\_OPT(p(j)) \}$

שימו לב: בניגוד לנוסחת המבנה ששם אין שום רקורסיה והיא מדברת על שיויון בין מספרים, פה זה כן קריאות רקורסיביות לפונקציה  $C\_OPT$ .

סיביות: במקרה הגרוע  $2^n$ .

דוגמת הרצה:



אבחנה: מספר הקריאות הרקורסיביות יכול להיות גדול, אבל יש לכל היותר  $n$  קריאות רקורסיביות עם ערכים שונים.

רעיון של תכנון דינאמי: לשמור את ערך הפתרון עבור הקריאה הבאה.

מימוש ראשון – שיטת הפתקאות (memoization):

נחזיק מערך גלובלי  $M[0], M[1], \dots, M[n]$ .  
אתחול המערך:  $M[i] = -1$  לכל  $i$  (המשמעות של -1 היא שעדיין לא ידוע ערך הפתרון)

```

C_OPT(j):
if j=0 then M[0]=0
else{
  if M[j-1]=-1 then C_OPT(j-1)
  if M[p(j)]=-1 then C_OPT(p(j))
  M[j] = max{ M[j-1] , wj + M[p(j)] }
}

```

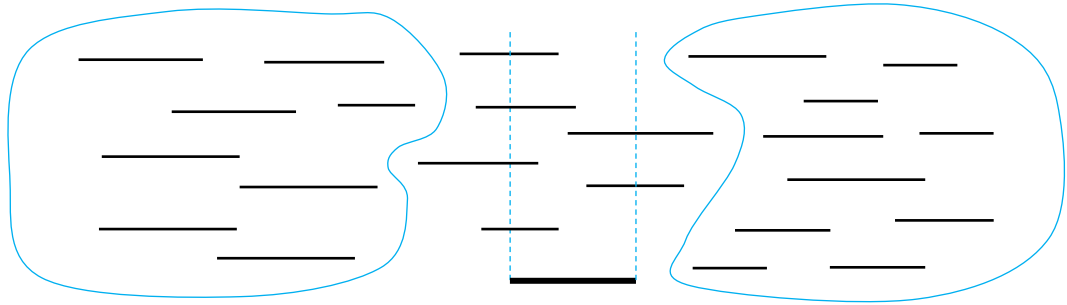
רעיון הוכחת הנכונות (שלא נראה):  
באינדוקציה על  $j$  מראים כי לאחר הקריאה ל  $C\_OPT(j)$  הערך  $M[j]$  מכיל את  $OPT(j)$ .

שאלה: איך מחשבים את ערך הפתרון לבעיה שלנו?  
תשובה: מריצים  $C\_OPT(n)$  ולאחר מכן ערך הפתרון נמצא בתא  $M[n]$ .

### ניתוח זמן ריצה:

- מיון  $O(n \cdot \log n)$
- לאחר המיון, חישוב  $p(1), p(2), \dots, p(n)$  אורך זמן  $O(n \cdot \log n)$
- זמן ביצוע הלולאה ללא הקריאות הרקורסיביות הוא  $O(1)$ . ישנן  $n$  קריאות רקורסיביות לכל היותר. לכן זמן כללי ללולאה הוא  $O(n)$
- סה"כ זמן ריצה  $O(n \cdot \log n)$

שאלה: למה היינו צריכים למיין את הקטעים?



אם היינו בוחרים את הקטע המודגש אז איינו צריכים לפתור רקורסיבית עבור צד ימין ועבור צד שמאל ולא היינו מצליחים להפעיל תכנון דנאמי. כלומר גם לאחר תכנון דנאמי היינו נתקעים עם אלגוריתם אקספוננציאלי...

תכנון דנאמי עובד טוב כאשר מספר הקריאות הרקורסיביות השונות האלגוריתם הוא קטן.

### מימוש שני – אלגוריתם איטרטיבי:

באלגוריתם הרקורסיבי התחלנו את החישוב מחיפוש הערך  $OPT(n)$  שרצינו ע"י קריאה ל-  $C\_OPT(n)$  ואח"כ בצענו קריאות רקורסיביות ל-  $C\_OPT(j)$  עבור ערכים יותר קטנים של  $j$ . זה נקרא Top-Down. עכשיו נראה אלגוריתם איטרטיבי שבו נעבוד הפוך: נתחיל מערכי  $j$  קטנים, נחשב אותם, ונתקדם כלפי מעלה. זה נקרא Bottom-Up.

### האלגוריתם האיטרטיבי:

נגדיר מערך  $M[0], M[1], \dots, M[n]$

אתחול:  $M[0]=0$

צעד: עבור  $j=1$  עד  $n$  בצע:

$$M[j] = \max\{ M[j-1], w_j + M[p(j)] \}$$

החזר:  $M[n]$

### הוכחת נכונות (באינדוקציה על $j$ ):

בסיס:  $M[0]=0=OPT(0)$

צעד: נניח  $M[j-1]=OPT(j-1)$  וגם  $M[p(j)]=OPT(p(j))$   
לכן

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

כאשר השיוויון האחרון נובע מנכונות נוסחת המבנה. מ.ש.ל.

זמן ריצה:  $O(n) + O(n \cdot \log n)$