



מבני נתונים ואלגוריתמים

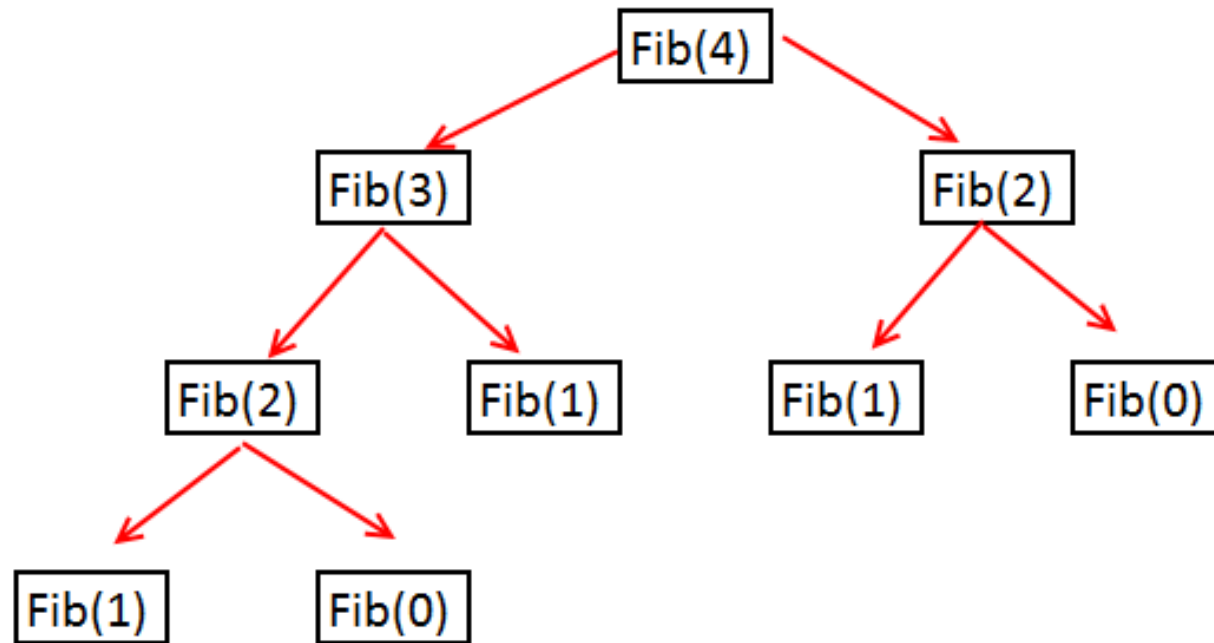
תכנון דינאמי


$$F_0 = 1$$

$$F_1 = 1$$

$$F_{n+2} = F_{n+1} + F_n$$

## Fibonacci Series



Fibonacci(N) = 0 for n=0  
= 1 for n=1  
= Fibonacci(N-1) + Fibonacci(N-2) for n>1

$$T(n) = T(n-1) + T(n-2) + 1 = 2^n = O(2^n)$$

# גישה ראשונה לשיפור

## **BOTTOM-UP APPROACH**

---

```
public int fibDP(int x) {
    int fib[] = new int[x + 1];
    fib[0] = 0;
    fib[1] = 1;
    for (int i = 2; i < x + 1; i++) {
        fib[i] = fib[i - 1] + fib[i - 2];
    }
    return fib[x];
}
```

# גישה שנייה לשיפור: TOP-DOWN APPROACH:

```
public int fibTopDown(int n) {  
    if(n==0) return 1;  
    if(n==1) return 1;  
    if(fib[n]!=0){  
        return fib[n];  
    }else{  
        fib[n] = fibTopDown(n-1) + fibTopDown(n-2);  
        return fib[n];  
    }  
}
```

---

# תכנות דינאמי

## תכנון דינמי

תכנון דינמי – שיטה לפיתרון בעיות רקורסיביות בהן משתמשים כמה פעמים בפיתרון של תתי בעיות. במקום לפתור את תתי הבעיות שוב ושוב, נחשב רק פעם אחת ונשמור את הפתרונות

# אלגוריתם "הפרד ומשול"

במדעי המחשב, הפרד ומשול היא פרדיגמת תכנון אלגוריתמים חשובה. היא מבוססת על שבירה רקורסיבית של הבעיה לשתיים או יותר תת-בעיות מאותה הצורה (או צורה דומה לה), עד שהבעיות הופכות לפשוטות דיין כדי שניתן יהיה לפתור אותן ישירות. לאחר מכן הפתרונות לתת הבעיות משולבים יחד כדי לתת פתרון לבעיה המקורית.

# מוטיבציה

נניח וסיימתם תואר והחלטתם לפתוח מפעל למוטות.

יש לכם מוטות בגודל מסוים, ואתם רוצים למכור אותם ולמקסם את הרווח.

עבור כל גודל תקבלו תמורה כספית שונה (לא בהכרח שחתיכת מוט גדולה יותר תקבל תמורה גדולה יותר!) ועליכם להחליט כיצד לחתוך את המוטות (אם בכלל) על מנת להרוויח כמה שיותר כסף.



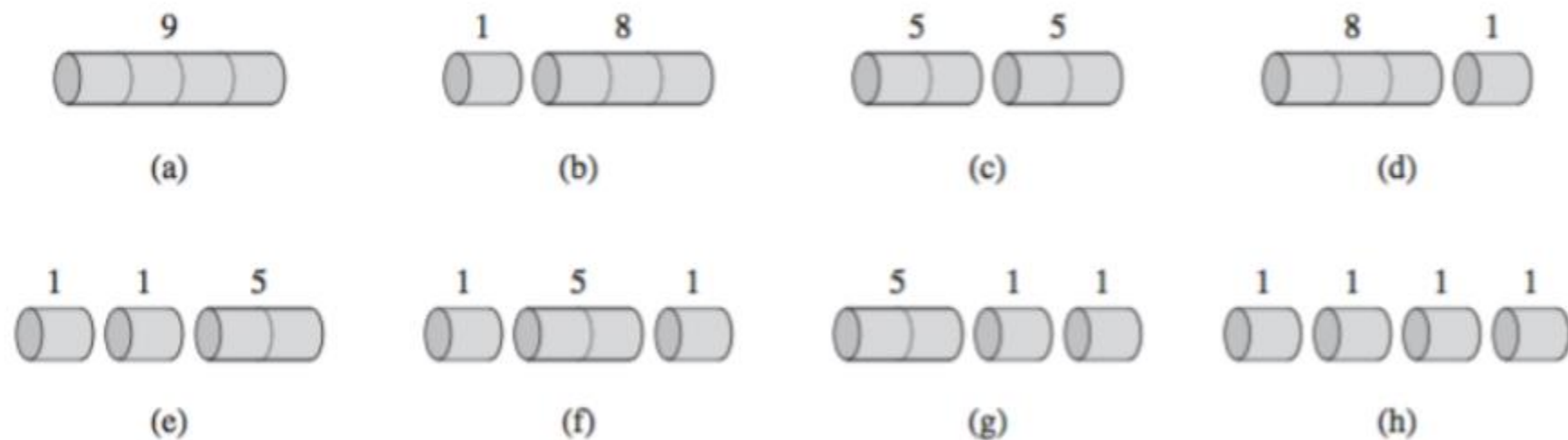
# תיאור הבעיה

בהנתן מוט שאורכו  $n > 0$ , והוא יכול להיחתך למספר  $k$  כלשהו של חתיכות  
( $k \leq n$ ). המחיר לכל חתיכה מאורך  $i$  מיוצג על ידי  $p(i)$  והמקסימום רווח שניתן  
להרוויח עבור חתיכת מוט באורך  $i$  מיוצג על ידי  $r(i)$ .

מצאו את  $r(n)$ .

length $i$	1	2	3	4	5	6	7	8	9	10
price $p_i$	1	5	8	9	10	17	17	20	24	30

**Figure 15.1** A sample price table for rods. Each rod of length  $i$  inches earns the company  $p_i$  dollars of revenue.



**Figure 15.2** The 8 possible ways of cutting up a rod of length 4. Above each piece is the value of that piece, according to the sample price chart of Figure 15.1. The optimal strategy is part (c)—cutting the rod into two pieces of length 2—which has total value 10.

- נגדיר את  $C(i)$  להיות המחיר האופטימלי של המוט עד האורך  $i$ .  
נוכל להגיע לנוסחה הרקורסיבית הבאה:
1. תמכרו את החתיכה בעלת המחיר האופטימלי.
  2. תמצאו דרך לחלק את החתיכה הנותרת (באופן אופטימלי).

וככה אנחנו נמשיך לחתוך בכל האופציות, וניקח את הרווח הכי גדול.

$$r_n = \max_{1 \leq i \leq n} (p_i + r_{n-i})$$

# פתרון נאיבי

CUT-ROD( $p, n$ )

1 **if**  $n == 0$

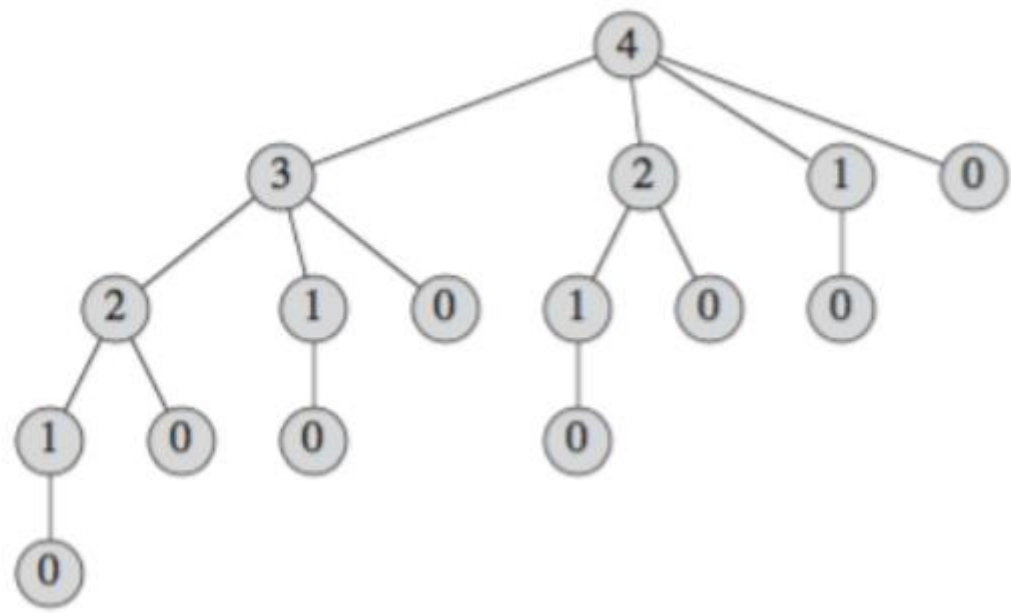
2     **return** 0

3  $q = -\infty$

4 **for**  $i = 1$  **to**  $n$

5      $q = \max(q, p[i] + \text{CUT-ROD}(p, n - i))$

6 **return**  $q$



# פתרון בעזרת תכנות דינאמי

**BOTTOM-UP-CUT-ROD**( $p, n$ )

```
1  let  $r[0..n]$  be a new array
2   $r[0] = 0$ 
3  for  $j = 1$  to  $n$ 
4       $q = -\infty$ 
5      for  $i = 1$  to  $j$ 
6           $q = \max(q, p[i] + r[j - i])$ 
7       $r[j] = q$ 
8  return  $r[n]$ 
```

הגדרת הבעיה:

קלט: 2 מחרוזות (באזורים שונים):

$$X = x_1x_2 \dots x_n$$

$$Y = y_1y_2 \dots y_m$$

פלט: תת סדרה משותפת מקסימלית:

$$Z = z_1 \dots z_k$$

כך שקיימים:

$$i_1 < i_2 < \dots < i_k$$

$$j_1 < j_2 < \dots < j_k$$

כך ש-

$$Z = x_{i_1}x_{i_2} \dots x_{i_k} = y_{j_1}y_{j_2} \dots y_{j_k}$$

(כלומר התווים לא חייבים להיות רציפים).

דוגמא –  $X = ABCBDAB, Y = BDCABA$  אורך תת הסדרה המקסימלית היא 4: BCBA, BDAB.

פיתרון נאיבי:

- נעבור על כל תתי הסדרות של X
- מחפש האם הן קיימות ב-Y.

← סיבוכיות – אקספוננציאלי



## פיתרון רקורסיבי:

אם נסתכל על התו האחרון בכל מחרוזת:

$$X = x_1 x_2 \dots x_{n-1} | x_n$$

$$Y = y_1 y_2 \dots y_{m-1} | y_m$$

**אם**  $y_m = x_n$  : אז זה יכול להיות חלק מתת הסדרה הארוכה ביותר. נחתוך את התווים האלו ונסתכל על:

$$X' = x_1 x_2 \dots x_{n-1}$$

$$Y' = y_1 y_2 \dots y_{m-1}$$

אם נמצא ב- $X'$  ו- $Y'$  תת סדרה מקסימלית, נוכל להוסיף את  $x_n$  ו- $y_m$ .

**אם**  $y_m \neq x_n$  : אפשר להגיד בודאות ש-  $x_n$  לא שייך לתת הסדרה המקסימלית או  $y_m$ .

לכן נחפש את התת הסדרה המקסימלית ב-2 המקרים הבאים:

מקרה 1:

$$X' = x_1 x_2 \dots x_{n-1}$$

$$Y = y_1 y_2 \dots y_m$$

מקרה 2:

$$X = x_1 x_2 \dots x_n$$

$$Y' = y_1 y_2 \dots y_{m-1}$$

ננסח בצורה מסודרת את האלגוריתם הרקורסיבי:

$LCS(X,n, Y,m )$ :

if  $n=0$  or  $m=0$

return 0

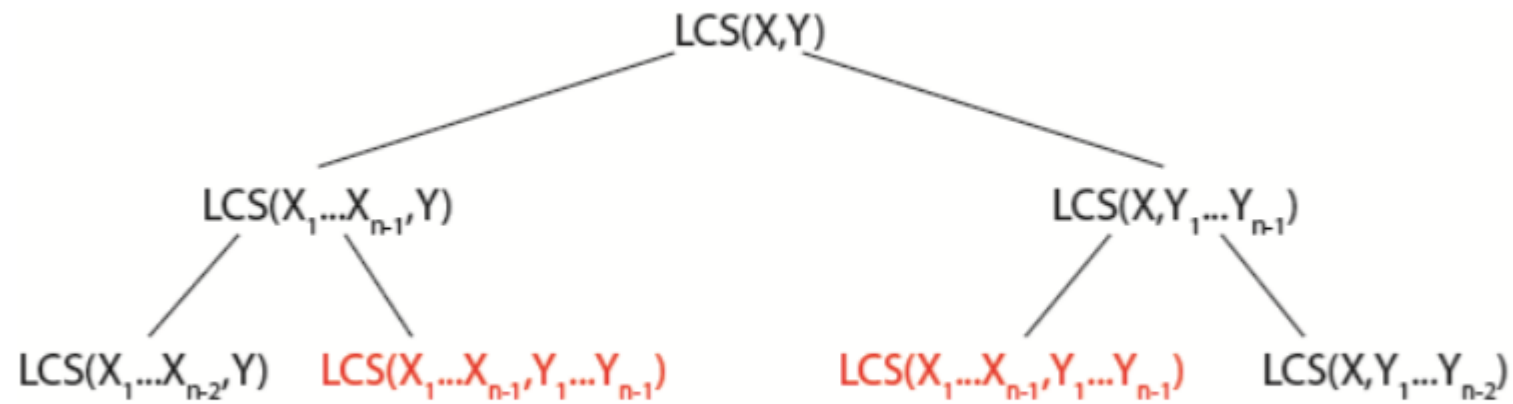
else if  $X_n = Y_m$

return  $LCS(X,n-1,Y,m-1) + 1$

else //  $X_n \neq Y_m$

return  $\max\{LCS(X,n,Y,m-1), LCS(X,n-1,Y,m)\}$

← סיבוכיות: גם אקספוננציאלי – במקרה הגרוע בכל קריאה לפונקציה, נעשה שוב 2 קריאות וכו':



← יש הרבה חישובים שחוזרים על עצמם!

## פיתרון באמצעות תכנון דינמי:

- נשתמש בטבלה D שגודלה  $(n+1)(m+1)$  (1 עבור המחזורות הריקה)
- בונים את הטבלה תא אחרי תא – תא  $LCS(i,j)$  מתאר את אורך תת הסדרה המקסימלית עבור המחזורות:

$$X = x_1x_2 \dots x_i$$

$$Y = y_1y_2 \dots y_j$$

- מסתכלים על 3 תאים – למעלה, אלכסון, שמאלה.
- אפשר לשמור את המסלול וכך לשחזר את הפיתרון האופטימלי.

```

LCS(X, Y):
for i=0 to n
    D(i,0) = 0
for j=0 to m
    D(0,j) = 0
for i=1 to n
    for j=1 to m
        if  $X_i = Y_j$ 
            D(i,j) = D(i-1, j-1) + 1
            P(i,j) = אלכסון
        else
            D(i,j) = max{D(i,j-1), D(i-1,j)}
            P(i,j) = שמאלה/למעלה

```

P- טבלה בגודל זהה ל-D שבה שומרים את המסלול.

← סיבוכיות: זמן ומקום  $O(nm)$ .

- אם לא רוצים לשחזר את הפיתרון, אפשר לצמצם את סיבוכיות המקום ל-2 שורות (הושרה הנוכחית והשורה הקודמת) ואז  $O(\max(n,m))$ .

דוגמא:

	j	0	1	2	3	4	5
i		-	B	D	C	A	B
0	-	0	0	0	0	0	0
1	A	0	0	0	0	1	1
2	B	0	1	1	1	1	2
3	C	0	1	1	2	2	2
4	B	0	1	1	2	2	3

← הפיתרון האופטימלי במקרה זה נמצא ב- $D(n+1,m+1) = 3$

שחזור הפיתרון:

- מתחילים מ- $P(n+1,m+1)$  והולכים בכיוון החצי.
- כאשר עולים למעלה באלכסון - מדפיסים את התו המתאים.

הפלט: B C B.

## לסיכום – שלבי התכנון הדינמי:

- הגדרה רקורסיבית של הבעיה
- בניית מרחב כל תתי הפתרונות
- מציאת הפיתרון האופטימלי
- שחזור הפיתרון האופטימלי