

אלגוריתמים / תרגיל #5

אריאל סטולרמן

קבוצה 02

(1)

להלן אלגוריתם *greedy* לפתרון בעיית *Fractional Knapsack*:יהיו  $O = \{o_1, \dots, o_n\}$   $n$  חפצים בעלי ערכים  $\{v_1, \dots, v_n\}$  ומשקלים  $\{w_1, \dots, w_n\}$  בהתאמה. להלן האלגוריתם:

- נגדיר פונקציית  $R: O \rightarrow \mathbb{R}$  המחזירה לכל  $o_i \in O$  את ערכו היחסי למשקלו, כלומר:  $R(o_i) := \frac{v_i}{w_i}$ .  $\forall o_i \in O$ .
- נסמן את ה"שק" אותו נמלא ב- $B$ , שיאותחל להיות ריק.
- כל עוד  $M > 0$  נבצע:

○ קח את  $o_i$  כך ש- $R(o_j) \mid o_j \in O$ .○ אם  $M \geq w_i$  אז הכנס ל- $B$  את  $o_i$ , הוצא את  $o_i$  מ- $O$ , ושנה את  $M$  כך:  $M := M - w_i$ .○ אחרת הכנס ל- $B$  את  $\frac{M}{w_i} \cdot o_i$ .**נכונות:**

תחילה, חישוב פונקציית  $R$  נותנת סידור של האובייקטים ב- $O$  לפי ערכם היחסי למשקלם, זהו ערכם האמיתי והמדד לקבוע את יוקרתו של כל אובייקט, ולפיכך העדפתו בהכנסה ל- $B$ . אם נדאג להכניס בכל רגע נתון כמה שניתן מהאובייקט היקר ביותר באותו רגע, זהו האלגוריתם החמדני למילוי  $B$ .

בלולאה על  $M$ : לאחר שלקחנו את האובייקט היקר ביותר (באופן יחסי למשקלו) שהוא  $o_i$ , אם  $M \geq w_i$  אזי ניתן להכניס לשק  $B$  את כל האובייקט. לאחר הכנסתו נדאג לעדכן את  $O$  (כדי שלא נבחר ב- $o_i$  שנית) ואת  $M$  (כיוון שלאחר ההכנסה הצטמצם המשקל המותר שנותר). אם  $M < w_i$  אזי לא ניתן להכניס את כל  $o_i$ , אז נכניס כמה שאנו יכולים מתוכו, שזה בדיוק  $M$  שנותר לנו בשלב זה, או באופן יחסי למשקלו המלא של  $o_i$ , יהיה זה השבר  $\frac{M}{w_i}$ . לאחר שלב זה  $M$  יתאפס והלולאה תפסק.

(2)

- להלן ניתוח הסיבוכיות של האלגוריתם הנתון:

- עבור  $n = 0$  מבצעים מספר קבוע של פעולות -  $T(0) = O(1)$
- עלות הקריאות הרקורסיביות + עלות הפעולות הקבועות -  $T(1) = O(1) + T(0) = 2 \times O(1)$
- הערה: את עלות כל איטרציה בלולאה ה- $for$  נחייב באופן *amortized* על הקריאה הרקורסיבית (שעלותה  $\leq O(1)$ ) באותה איטרציה, כך יוותר חיוב  $O(1)$  בלבד על הריצה הנוכחית של  $Fun$ , לא כולל הקריאות הרקורסיביות.
- משיקולים דומים -  $T(2) = O(1) + T(1) + T(0) = O(1) + 2 \times O(1) + O(1) = 4 \times O(1)$
- ... (באינדוקציה)
- $T(n) = O(1) + T(n-1) + \dots + T(1) + T(0) = O(1) + 2^{n-1} \times O(1) + \dots + 2 \times O(1) + O(1) = 2^n \times O(1) = O(2^n)$  - כנדרש

- להלן אלגוריתם לפתרון הבעיה ב- $O(n^2)$ :

```

int Fun(int n){
    int[n] arr; // הנחה: הערכים מאותחלים ל-0
    int result = 0;
    if (n == 0) then return 0; //else:
    for (int i = 0; i < n; i++){
        if (i == 0) then arr[i] = 0; //טרוויאלי אך לשם קריאות הקוד
        else{
            for (int j = 0; j < i; j++) arr[i] = arr[i] + arr[j];
            arr[i] = i * arr[i] + i;
        }
    }
    for (i = 0; i < n; i++) result = result + arr[i];
    result = n * result + n;
    return result;
}

```

הסבר:

האלגוריתם מחזיק מערך בגודל  $n$ , וע"י לולאה מעדכן לכל  $0 \leq i \leq n - 1$  את המערך כך שבמקום ה- $i$  יישב הערך  $Fun(i)$ . שלב זה נעשה ע"י לולאה פנימית הסוכמת את כל איברי המערך עד  $i$ , ולבסוף מעדכנת ל- $i * arr[i] + i$ . הלולאה האחרונה עושה אותו דבר כמו הלולאה הפנימית, רק עבור המקרה  $i = n$ . סה"כ סיבוכיות האלג' כמבוקש:  $O(n^2)$ .

הערה:

ניתן היה לבצע את האלגוריתם ב- $O(n)$  ע"י החלפת הלולאה הפנימית לעדכון  $arr[i]$  במספר קבוע של פעולות באופן הבא:

$$arr[i] = \underbrace{((arr[i-1] - (i-1)) / (i-1) + arr[i-1])}_{=Fun(0)+Fun(1)+\dots+Fun(i-2)} * i + i;$$

ובכך לנצל את העבודה שעבור חישוב  $i - 1$  כבר סכמנו את כל  $Fun(0) + \dots + Fun(i - 2)$ . אותו שינוי לעדכון  $result$ .

(3)

תחילה נשים לב כי:

$$F(n) = \underbrace{F(0) \cdot F(1) + F(1) \cdot F(2) + \dots + F(n-3) \cdot F(n-2)}_{=F(n-1)} + F(n-2) \cdot F(n-1) \Rightarrow$$

$$F(n) = [F(n-2) + 1] \cdot F(n-1)$$

להלן אלגוריתם מובסס תכנות דינאמי לחישוב  $F(n)$ :

```

int F(int n){ // הנחה: n ≥ 2
    int[n+1] arr; // הנחה: הערכים מאותחלים ל-0
    arr[0] = arr[1] = 2;
    for(int i = 2; i < n+1; i++) arr[i] = (arr[i-2] + 1) * arr[i-1];
    return arr[n]; // מחזיר את הערך האחרון שחושב במערך, הוא הערך הרצוי
}

```

ברור כי האלגוריתם פועל בסיבוכיות לינארית  $O(n)$ . להלן מספר הפעולות האריתמטיות המתבצעות ע"י האלגוריתם כפונקציה של הקלט  $n$  (כולל אתחולי משתנים והשמות למשתנים):

$$T(n) =$$

מחובר	הסבר
$n + 1$	אתחול המערך
2	השמת ערכי התחלה לשני האיברים הראשונים במערך
1	אתחול $i$
$n + 1 - 2 = n - 1$	מספר ההשוואות שה"כ בלולאה
$n + 1 - 2 - 1 = n - 2$	מספר ההעלאות של $i$ ב-1 שה"כ בלולאה
$(n - 2) \cdot (1 + 1 + 1 + 1 + 1) = 5n - 10$	מספר עדכוני המערך בתוך הלולאה כפול: השמה ל- $arr[i]$ + חישוב $i - 2$ + חישוב החיבור עם 1 + חישוב ההכפלה + חישוב $i - 1$ שה"כ 5 פעולות אריתמטיות
<b><math>8n - 9</math></b>	<b>סה"כ:</b>

כאמור, אכן מתקיים כי:  $T(n) = O(n)$ .