

מבני נתונים ואלגוריתמים – תרגול #13

דחיסה - המשך

תרגיל

נתונה שפה עם האותיות והשכיחויות הבאות:

| A | B | C | D | E | F | G |
|------|------|------|------|------|------|------|
| 0.13 | 0.14 | 0.15 | 0.16 | 0.17 | 0.18 | 0.07 |

- מה הקידוד האופטימלי?
- מה שיעור הדחיסה לעומת האופטימום (אנטרופיה)?

תזכורת – קוד הופמן:

Huffman(C):

$n = |C|$

Q.insert(C)

for i=1 to n-1

new node A

$X = \text{Left}(A) = \text{Q.Extract-Min}()$

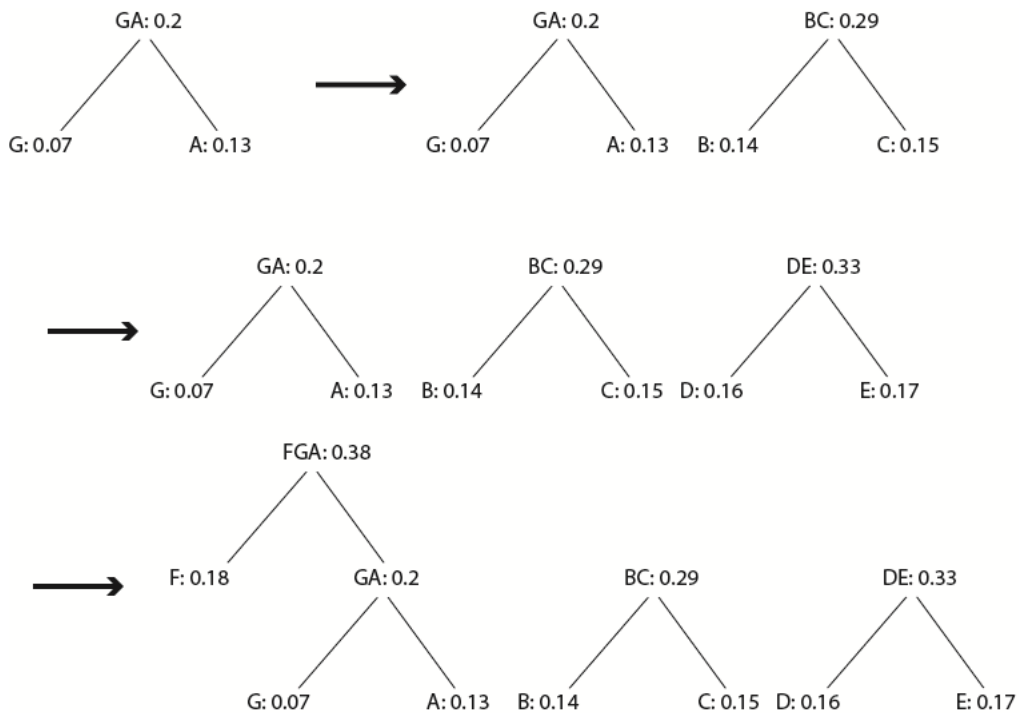
$Y = \text{Right}(A) = \text{Q.Extract-Min}()$

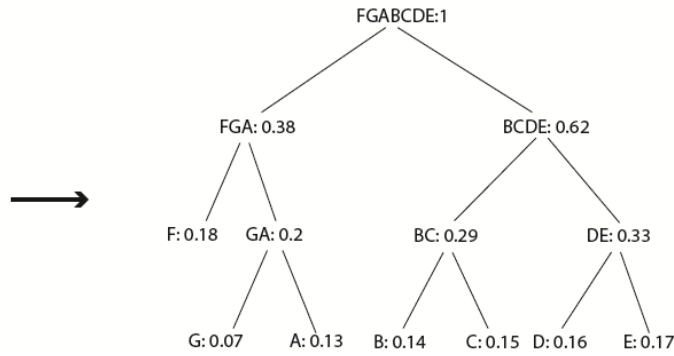
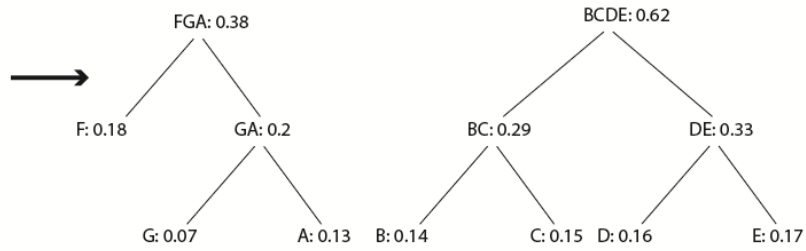
$f(A) = f(X) + f(Y)$

Q.Insert(A)

return Q.ExtractMin()

פיתרון:





חישוב שיעור הדחיסה:

$$E(X) = 0.18 * 2 + (0.13 + 0.14 + 0.17 + 0.16 + 0.15 + 0.07) * 3 = 2.82$$

$$H(X) = -0.13(\log 0.13) - 0.14(\log 0.14) \dots = 2.76$$

כלומר צריך מינמום 2.76 ביטים במוצע לייצוג השפה, ובפועל קיבלנו קוד עם 2.82 ביטים במוצע!

תרגיל:

- (1) יהי X מ"מ המקבל ערכים $1, 2, \dots, n$ בהסתברויות p_1, p_2, \dots, p_n בהתאמה. נניח ש- $2p_i < p_{i+1}$. איך ייראה עץ הקידוד האופטימלי של ערכי X ?
- (2) נניח ש- X מקבל את הערכים $1, 2$ בהסתברויות $1/4, 3/4$ בהתאמה. מה ניתן לעשות כדי לדחוס ביעילות גדולה יותר סדרת ערכים של X ?

פיתרון:

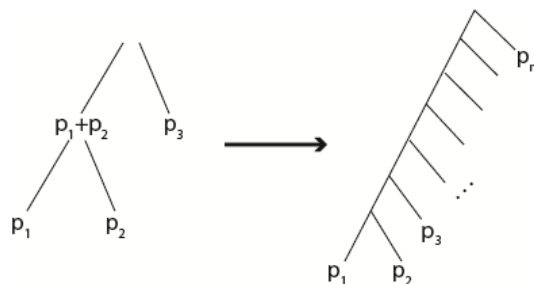
(1) נשים לב ש-

$$p_1 + p_2 < 2p_2 + p_2 < 2p_2 < p_3$$

לכן:

$$p_1 + p_2 + \dots + p_k < p_{k+1}$$

לכן תמיד נאחד את 2 הנמוכים, ואז הנמוך הבא עם הקודקוד החדש שנוצר וכן הלאה:



(2) העץ האופטימלי לפי הופמן הוא:  כלומר כמות הביטים הממוצעת היא 1.

אבל האנטרופיה נמוכה יותר:

$$H(X) = -\frac{1}{4} \log \frac{1}{4} - \frac{3}{4} \log \frac{3}{4} = 0.81$$

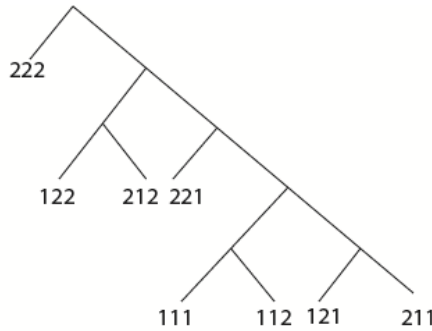
כלומר אפשר לייצג בפחות ביטים (בממוצע).

← כדי לדחוס ביעילות גבוהה יותר, אפשר לדחוס זוגות או שלשות של תווים.

דוגמא – שלשות:

| | | | | | | | |
|------|------|------|------|------|------|------|-------|
| 111 | 112 | 121 | 211 | 122 | 212 | 221 | 222 |
| 1/64 | 3/64 | 3/64 | 3/64 | 9/64 | 9/64 | 9/64 | 27/64 |

נקבל את העץ הבא:



כמות הביטים הממוצעת של כל שלשה:

$$E(X) = \frac{1}{64} (27 * 1 + 9 * 3 * 3 + 1 * 5 + 3 * 5 * 3) = \frac{158}{64} = 2.46$$

כלומר מקודדים כל שלשה ב-2.46 ביטים, לכן כל תו מקודד ב-0.82 ביטים בממוצע!

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 בטבלה.

בעיה לדוגמא –

בעיית תת הסדרה המשותפת הארוכה ביותר – Largest Common Subsequence (LCS)

הגדרת הבעיה:

קלט: 2 מחרוזות (באורכים שונים):

$$X = x_1 x_2 \dots x_n$$

$$Y = y_1 y_2 \dots y_m$$

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:

$$X = x_1 x_2 \dots x_n$$

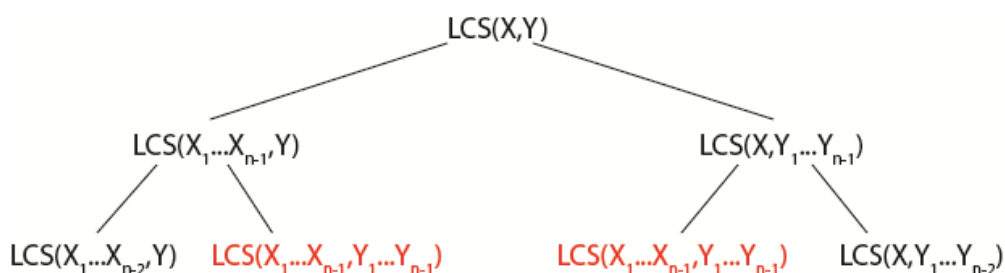
$$Y' = y_1 y_2 \dots y_{m-1}$$

ננסח בצורה מסודרת את האלגוריתם הרקורסיבי:

```

LCS(X,n, Y,m):
if n=0 or m=0
    return 0
else if Xn = Ym
    return LCS(X,n-1,Y,m-1) + 1
else // Xn ≠ Ym
    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:

$$X = x_1 x_2 \dots x_i$$

$$Y = y_1 y_2 \dots y_j$$

- 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 Xi = Yj
            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 | ↑ 1 | 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 הכתרון הדינמי:

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