

# חיפוש מחרוזות – אלגוריתם Knuth-Morris-Prat (KMP)

## סימונים

כל המחרוזות הן מערכי תווים והמערכים מתחילים באינדקס 0. המחרוזת הריקה תסומן ב- $\epsilon$ . עבור מחרוזות  $X$  נסמן ב- $X_n$  את המחרוזת המכילה את תווים 0 עד  $n - 1$  ב- $X$ . (שימו לב ש- $X_0$  היא המחרוזת הריקה).

## הבעיה

נתונה מחרוזת  $T$  ומחרוזת  $P$  (בשביל pattern). רוצים למצוא את כל המופעים (הרצופים) של  $P$  בתוך  $T$ . נסמן  $m = \text{Length}(P)$ ,  $n = \text{Length}(T)$ . בעיה שקולה: עבור מחרוזות  $T, P$  כנ"ל רוצים למצוא את כל ה- $i$ ים עבורם  $P$  היא סיפא של  $T_i$ .

## האלגוריתם

לכל  $0 \leq i \leq m$  נגדיר את  $C[i]$  באופן הבא:

- $C[0] = -1$
- עבור  $i > 0$ ,  $C[i]$  הוא המספר המקסימלי  $0 \leq j < i$  המקיים ש- $P_j$  הוא סיפא של  $P_i$ . (שימו לב ש- $P_0$  היא המחרוזת הריקה ולכן היא סיפא של כל מחרוזת. זה אומר ש- $P[i] > 0$  לכל  $i > 0$ .)<sup>1</sup>

האלגוריתם מתחיל בחישוב הטבלה  $C$ . נדחה את ההסבר על איך לעשות זאת ביעילות לסוף. כעת מאתחלים שני מונים  $i, q$  ל-0.  $i$  יהיה האינדקס אותו נבדוק ב- $T$  ו- $q$  יהיה האינדקס אותו נבדוק ב- $P$ . כעת מבצעים:

1. כל עוד  $i < \text{Length}(T)$ :
  - a. אם  $T[i] = P[q]$ :
    - i.  $i += 1, q += 1$
    - ii. אם  $q = m$ :
      1. הדפס את  $i - m$
      2.  $q = C[q]$  (שימו לב ש- $C[m] = C[q]$ )
  - b. אחרת:
    - i. אם  $q > 0$ :  $q = C[q - 1] + 1$  (<sup>2</sup>)
    - ii. אחרת:  $i += 1$

<sup>1</sup> זה שונה ב-1 מההגדרה שנתתי.  
<sup>2</sup> זה היה נכון בסופו של דבר. אם האינדקסים מתחילים ב-1, הניסוח יותר פשוט.

יש מימושים שקולים, כמובן.

### דוגמא לריצה

נבחר  $P = aabaaba$  ו- $T = aaaabaabaacaabaaba$  (יש שני מופעים של  $P$  בתוך  $T$ ). האורך של  $P$  הוא  $m = 7$  והאורך של  $T$  הוא  $n = 18$ .

הטבלה  $C$  תהייה כדלקמן:

i	0	1	2	3	4	5	6	7
C[i]	-1	0	1	0	1	2	3	4
Maximal P-prefix which is a postfix of $P_i$ .	None		a		a	aa	aab	aaba

הפעם הראשונה בה תהייה אי התאמה היא כאשר  $i = 2, q = 2$ :

i: 2  
 T: aaabaabaacaabaaba  
 P: aabaaba  
 q: 2

מבצעים  $q = C[q - 1] + 1 = C[1] + 1 = 1$  וממשיכים להשוות:

i: 23  
 T: aaabaabaacaabaaba  
 P: aabaaba  
 q: 12

נעצרים כאשר  $i = 3, q = 2$ . מבצעים  $q = C[q - 1] + 1 = C[1] + 1 = 1$  וממשיכים להשוות:

i: 3 8  
 T: aaabaabaacaabaaba  
 P: aabaaba  
 q: 1 6

כאשר  $i = 8, q = 6$  מצאנו מופע של  $P$  ב- $T$ . לאחר שמגדילים את  $i, q$  ב-1 מדפיסים  $i - m = 9 - 7 = 2$  ומבצעים  $q = C[q] = C[7] = 4$ . ממשיכים להשוות:

i: 9  
 T: aaaabaabaacaabaaba  
 P: aabaaba  
 q: 4

נעצרים כאשר  $i = 10, q = 5$ . מבצעים  $q = C[q - 1] + 1 = C[4] + 1 = 2$ . ממשיכים להשוות:

i: 10  
 T: aaaabaabaacaabaaba

P: a**b**aaba  
q: 2

אך שוב נעצרים מייד. מבצעים  $q = C[q - 1] + 1 = C[1] + 1 = 1$  ממשיכים להשוות:

i: 10  
T: aaaabaabaac**a**aabaaba  
P: a**a**aaba  
q: 1

אך שוב נעצרים מייד. מבצעים  $q = C[q - 1] + 1 = C[0] + 1 = 0$  ממשיכים להשוות:

i: 10  
T: aaaabaabaac**a**aabaaba  
P: a**a**aaba  
q: 0

אך שוב נעצרים מייד. היות ו- $q = 0$ , מבצעים  $i = i + 1 = 11$  ממשיכים להשוות:

i: 11  
T: aaaabaabaac**aabaaba**  
P: **aabaaba**  
q: 0

כאשר  $i = 17, q = 6$  מצאנו מופע נוסף של התבנית. לאחר שמגדילים את  $i, q$  ב-1 מדפיסים  $i - m = 11 - 7 = 4$  ומבצעים  $q = C[q] = C[7] = 4$ . בשלב זה נצא מהלולאה (כי  $i \geq \text{Length}(T)$ ) והאלגוריתם יסתיים.

לסיכום, הנה תקציר של "מה שעשה האלגוריתם":

T: aa**a**abaabaacaabaaba  
P: a**a**babaaba  
T: aaa**a**abaabaacaabaaba  
P: a**a**babaaba  
T: aaa**aabaaba**acaabaaba  
P: **aabaaba**  
T: aaaabaabaac**a**aabaaba  
P: aaba**a**ba  
T: aaaabaabaac**a**aabaaba  
P: a**a**babaaba  
T: aaaabaabaac**a**aabaaba  
P: a**a**babaaba  
T: aaaabaabaac**aabaaba**  
P: **aabaaba**

ניתן לראות כי האלגוריתם מנסה לחפש את  $P$ -ב- $T$  וכל פעם כאשר נמצאת אי התאמה (באדום), הוא מסיט את  $P$  ימינה בהיסט הקטן ביותר האפשרי בו עדיין יש התאמה בין התווים הראשונים של  $P$  לתווים האחרונים שנסקרו ב- $T$ . אז ממשיך האלגוריתם בחיפוש מאותו מקום בו עצר לפי ההסטה של  $P$ .

### איך בונים את הטבלה

בניית הטבלה נשענת על התכונה הבאה:  $P_{i+1}$  היא סיפא של  $P_{k+1}$  אם ורק אם  $P_i$  סיפא של  $P_k$  וגם  $P[k] = P[i]$ .

נניח שחישבנו את  $C[i]$  עבור  $i \leq k$  ואנו רוצים לחשב את  $C[k+1]$ . אזי לפי הגדרת  $C$ ,  $P_{C[k]}$  היא הרישא הגדולה ביותר של  $P$  שהיא סיפא של  $P_k$ . אם  $P[C[k]] = P[k]$  אז  $C[k+1] = C[k] + 1$  (לפי התכונה). אחרת, הרישא הבאה בגודלה של  $P$  שהיא סיפא של  $P_k$  היא  $P_{C[C[k]]}$  (נובע מהגדרת  $C$ ) ושוב, אם  $P[C[C[k]]] = P[k]$  אז  $C[k+1] = C[C[k]] + 1$ . אפשר להמשיך כך והתהליך יעצר כאשר נגיע למסקנה שהרישא הארוכה ביותר של  $P$  שיכולה להיות סיפא של  $P_{k+1}$  היא  $P_0$  (מחרוזת ריקה) ואז בהכרח  $C[k+1] = 0$ .

האלגוריתם:

1.  $C[0] = -1$
2.  $C[1] = 0$
3. עבור  $k = 2, \dots, m$ :
  - a.  $d = C[k-1]$
  - b. כל עוד  $P[d] \neq P[k-1]$  וגם  $d \geq 0$  בצע  $d = C[d]$ .
  - c.  $C[k] = d + 1$

לדוגמא, עבור המחרוזת  $P = aabaabab$  החישוב יתבצע כדלקמן:

1.  $C[0] = -1$
2.  $C[1] = 0$
3. חישוב  $C[2]$ :  $d = C[2-1] = 0$  ולכן הלולאה  $b$  לא תתבצע ונקבל  $C[2] = 0 + 1 = 1$
4. חישוב  $C[3]$ :  $d = C[3-1] = 1$  ולכן לאחר הלולאה  $b$  יתקיים  $d = -1$   
 $C[3] = -1 + 1 = 0$  אז נקבל  $C[3] = 0$
5. חישוב  $C[4]$ :  $d = C[4-1] = 0$  ולכן הלולאה  $b$  לא תתבצע (כי  $P[3] = a$ )  
 $C[4] = 0 + 1 = 1$  אז נקבל  $C[4] = 1$
6. חישוב  $C[5]$ :  $d = C[5-1] = 1$  ולכן הלולאה  $b$  לא תתבצע (כי  $P[4] = a$ )  
 $C[5] = 1 + 1 = 2$  אז נקבל  $C[5] = 2$
7. חישוב  $C[6]$ :  $d = C[6-1] = 2$  ולכן הלולאה  $b$  לא תתבצע (כי  $P[5] = b$ )  
 $C[6] = 2 + 1 = 3$  אז נקבל  $C[6] = 3$
8. חישוב  $C[7]$ :  $d = C[7-1] = 3$  ולכן הלולאה  $b$  לא תתבצע (כי  $P[6] = a$ )  
 $C[7] = 3 + 1 = 4$  אז נקבל  $C[7] = 4$
9. חישוב  $C[8]$ :  $d = C[8-1] = 4$  ולכן הלולאה  $b$  לא תתבצע (כי  $P[7] = b$ )  
 $C[8] = 4 + 1 = 5$  אז נקבל  $C[8] = 5$

$$P[C[C[d]]] = P[C[1]] = P[0] = a$$

ולכן הלולאה תעצור כאשר  $d = -1 = C[C[4]]$  ונקבל  $C[8] = -1 + 1 = 0$ .