

## דחיסה - אלגוריתם Lempel Ziv Welch

### קידוד (Encoding)

יש לנו אלף בית - כל האותיות הבסיסיות. (לדוגמה - א' וב').  
נגדיר מילון, שבהתחלה יכיל רק את האותיות הבסיסיות (א' = 0, ב' = 1, כל אות היא integer).

1. תרגם את המילה הכי ארוכה שאפשר עם המילון הנוכחי.

2. הוסיף אותה ואת האות שאחריה למילון.

אם לדוגמה נרצה לתרגם: אאבבאבבאאאבבבבבב.  
המילון שלנו יהיה:

מילה	תרגום
א	0
ב	1
אא	2
אאב	3
בב	4
בא	5
אב	6
באב	7
באבב	8
בבא	9
אאא	10
אאבב	11
בבב	12
בבבב	13

והמילה שתרגמנו תהיה:

4, 12, 4, 3, 2, 4, 7, 5, 0, 1, 1, 2, 0

### פענוח (Decoding)

1. אם הקוד קיים - תרגם. הוסיף למילון את כל המילה הקודמת + אות ראשונה של מילה נוכחית.

2. אם הקוד לא קיים - הוסיף למילון את כל המילה הקודמת + אות ראשונה של המילה הקודמת. תרגם.

דוגמא:

מילון:

מילה	קוד
א	0
ב	1

פענחו את הקידוד הקודם:

4	12	4	3	2	4	7	5	0	1	1	2	0

(1) אתחול

מילון:

מילה	קוד
א	0
ב	1

4	12	4	3	2	4	7	5	0	1	1	2	0
												א

(2) '2' לא נמצא במילון לכן זו האפשרות השניה באלגוריתם הפענוח

נעדכן את המילון:

מילה	קוד
א	0
ב	1
אא	2

נפענח:

4	12	4	3	2	4	7	5	0	1	1	2	0
											אא	א

(3) '1' כן נמצא במילון לכן זו האפשרות הראשונה באלגוריתם הפענוח

4	12	4	3	2	4	7	5	0	1	1	2	0
										ב	אא	א

נעדכן את המילון:

מילה	קוד
א	0
ב	1
אא	2
אאב	3

וכן הלאה.

## התאמת מחרוזות

סימונים והגדרות:

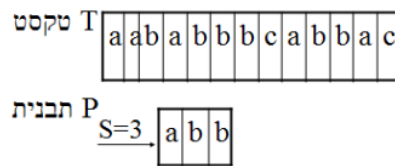
- **טקסט T** (מערך של תווים) באורך  $n$  –  $T[0, \dots, n-1]$  ו**תבנית P** באורך  $m$  –  $P[0, \dots, m-1]$  כך ש-  $m \leq n$ .
- התווים של P ו-T נלקחים מאלפבית סופי  $\Sigma$ . לדוגמא:  $\Sigma = \{0, 1\}$ ,  $\{a, b, \dots, z\}$ .
- $\Sigma^*$  - קבוצת כל המחרוזות באורך סופי שניתן להרכיב מ- $\Sigma$ .
- $\varepsilon$  - המחרוזת הריקה. מתקיים ש-  $\varepsilon \in \Sigma$

בעיית התאמת מחרוזות:

- רוצים למצוא את כל המופעים של P ב-T.
- רוצים למצוא את כל האינדקסים (=היסטים) ב-T כך ש-

$$T[i, \dots, i + m - 1] = P[0, \dots, m - 1]$$

לדוגמא:



- תהי X מחרוזת. נסמן ב- $X_i$  את המחרוזת שמכילה את התווים מ-0 עד  $i$  ב-X ואורכה  $i$ . כלומר  $X_i = X[0, \dots, i-1]$
- **רישא:** x רישא של y אם קיים z כך ש- $xz = y$  (=שרשור של x עם z).
- **סיפא:** x סיפא של y אם קיים z כך ש- $zx = y$ .

הגדרה נוספת של הבעיה:

- רוצים למצוא את כל ההיסטים  $m$  כך ש-P סיפא של  $T_m$ .

**האלגוריתם הנאיבי**

- בודקים את כל ההיסטים שהאפשריים באמצעות לולאה שרצה מ- $i=0$  עד  $n-m$ .

סיבוכיות:  $O((n-m+1)m) \rightarrow O(nm)$ .

## אלגוריתם KMP

על מנת להימנע מהשוואות מיותרות, נשתמש במידע שיש בתבנית P עצמה. מחשבים את פונקציית  $\Pi$  – פונקציית הרישא עבור תבנית P באורך m.

$\Pi(i) = j$  – האינדקס המקסימלי j ( $0 \leq j < i$ ) המקיים ש- $P_j$  היא סיפא של  $P_i$  (לכל  $0 \leq i < m$ ).

דוגמא:

<b>P = a b a b c</b>		0	1	2	3	4	5
	$\Pi$	-1	0	0	1	2	0

$\Pi(1) = 0$  – צריך למצוא את הרישא הכי ארוכה של  $P_0 = \varepsilon$  שהיא סיפא של  $P_1 = a$ . אין - לכן  $\Pi(1) = 0$ .

$\Pi(2) = 0$  – צריך למצוא את הרישא הכי ארוכה של  $P_1 = a$  שהיא סיפא של  $P_2 = ab$ . אין - לכן  $\Pi(2) = 0$ .

$\Pi(3) = 1$  – צריך למצוא את הרישא הכי ארוכה של  $P_2 = ab$  שהיא סיפא של  $P_3 = aba$ , לכן  $\Pi(3) = 1$ .

וכן הלאה.

הרעיון הכללי של האלגוריתם: מחפשים את P ב-T. בכל פעם שיש אי-התאמה, נסיט את P ימינה בהיסט הקטן ביותר האפשרי שמקיים שעדיין יש התאמה של התווים הראשונים ב-P לבין התווים האחרונים ב-T. האלגוריתם:

KMP(P, T):

q = i = 0

while i < length(T)

if T[i] == P[q]

σ[i] = q

i ++

q ++

if q == m

print i-m

q = Π(q)

else

if q > 0

q = Π(q-1) + 1

else

i ++

**סיבוכיות:**  $O(m+n)$ .

האינדקס של ההתקדמות לאורך הטקסט הוא j, האינדקס הנוכחי של מספר האותיות המתאימות בין התבנית לטקסט הוא q.

בנוסף, KMP יכול לחשב את  $\sigma$  – פונקציית הסיפא.

$j - \sigma(i) = j$  הוא אורך הרישא המקסימלית של התבנית  $P_j$  שהיא סיפא של  $T_i$  (לכל  $0 < i \leq 9$ ).

לדוגמא:

**T = a a b a a c a a**

	0	1	2	3	4	5	6	7	8	9
$\sigma$	0	1	2	2	3	4	5	0	1	2

**P = a a b a a**

נחפש ב- $T_4 = a a b$  את הסיפא המקסימלית שהיא רישא של  $P_3 = a a b$ .

נחפש ב- $T_6 = a a b a a$  את הסיפא המקסימלית שהיא רישא של  $P_5 = a a b a a$ .

פונקציית הסיפא (סיגמא, שורה 4 באלגוריתם) אינה חיונית למימוש האלגוריתם אלא מוסיפה מידע שמסייע בפתרון בעיות עם אלגוריתם KMP, כפי שנראה בהמשך.

אלגוריתם KMP (מילולית)

Preprocessing : בנה את טבלת פונקציית הרישא.

ריצה: כאשר יש חוסר התאמה במיקום ה- $q+1$ , קדם את התבנית  $P$  ימינה  $\pi(q) - q$  מקומות.

(כי על פי טבלת פונקציית הרישא, לאחר שהתקדמנו  $q$  מקומות  $\pi(q)$  היא רישא נכונה).

דוגמא: <https://www.youtube.com/watch?v=kBW6oPaVjq0>

The screenshot shows a video player with the title "KMP algorithm - Example". It displays the following data:

String - T: a b c d a b a b c d a b c d a b d e

Pattern - P: a b c d a b d

Prefix table:

q	1	2	3	4	5	6	7
P[q]	a	b	c	d	a	b	d
$\pi[q]$	0	0	0	0	1	2	0

# KMP algorithm - Example

i	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
String - T	a	b	c	d	a	b		a	b	c	d	a	b	c	d	a	b	d	e
Pattern - P					a	b	c	d	a	b	d								

## Prefix table

q	1	2	3	4	5	6	7
P[q]	a	b	c	d	a	b	d
$\pi[q]$	0	0	0	0	1	2	0

# KMP algorithm - Example

i	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
String - T	a	b	c	d	a	b		a	b	c	d	a	b	c	d	a	b	d	e
Pattern - P					a	b	c	d	a	b	d								

## Prefix table

q	1	2	3	4	5	6	7
P[q]	a	b	c	d	a	b	d
$\pi[q]$	0	0	0	0	1	2	0

# KMP algorithm - Example

i	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
String - T	a	b	c	d	a	b	a	b	c	d	a	b	c	d	a	b	d	e	
Pattern - P							a	b	c	d	a	b	d						

## Prefix table

q	1	2	3	4	5	6	7
P[q]	a	b	c	d	a	b	d
$\pi[q]$	0	0	0	0	1	2	0

# KMP algorithm - Example

i	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
String - T	a	b	c	d	a	b		a	b	c	d	a	b	c	d	a	b	d	e
Pattern - P								a	b	c	d	a	b	d					

## Prefix table

q	1	2	3	4	5	6	7
P[q]	a	b	c	d	a	b	d
$\pi[q]$	0	0	0	0	1	2	0

# KMP algorithm - Example

i	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
String - T	a	b	c	d	a	b		a	b	c	d	a	b	c	d	a	b	d	e
Pattern - P								a	b	c	d	a	b	d					

## Prefix table

q	1	2	3	4	5	6	7
P[q]	a	b	c	d	a	b	d
$\pi[q]$	0	0	0	0	1	2	0

# KMP algorithm - Example

i	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
String - T	a	b	c	d	a	b		a	b	c	d	a	b	c	d	a	b	d	e
Pattern - P												a	b	c	d	a	b	d	

## Prefix table

q	1	2	3	4	5	6	7
P[q]	a	b	c	d	a	b	d
$\pi[q]$	0	0	0	0	1	2	0



## תרגילים

### תרגיל:

הגדרה: נסמן ב- $T^R$  את המחזורת ההופכית של  $T$  (באורך  $n$ ), כלומר  $T^R = T(n-1)T(n-2)...T(0)$ .

$T$  פלינדרום אם  $T = T^R$ .

בהינתן מחזורת  $T$  באורך  $n$ , תארו אלגוריתם שמוצא את אורך הרישא המקסימלית של  $T$  שהינה פלינדרום.

$T = a b a a b a b a$

לדוגמא-

$T_0 = a$   
 $T_2 = ab$   
 $T_6 = ababab$

אז נחזיר  $j=6$ .

### פיתרון:

אם  $x$  רישא של  $T$ , אז קיים  $y$  כך ש- $T = xy$ . לכן מתקיים  $T^R = y^R x^R$ .

אם  $x$  פלינדרום אז  $T^R = y^R x$  (גם הכיוון השני נכון).

לכן מספיק למצוא את הרישא הארוכה ביותר של  $T$  שהיא סיפא של  $T^R$ .

← נריץ KMP כך הטקסט שלנו הוא  $T^R$  והתבנית היא  $T$  ונחזיר  $\sigma(n) = j$ -ה המקסימלית כך ש- $T_j$  סיפא של  $T^R$ .

סיבוכיות:  $O(m+n) = O(2n) = O(n)$ . (נאיבי:  $O(n^2)$ ).

הערה:  $x$  פלינדרום אם  $x = x^r$

### תרגיל:

הגדרה: מחזורת  $T$  היא סיבוב מעגלי של מחזורת  $T = t_1 t_2 \dots t_n$  אם קיים  $n \geq 1$  כך שמתקיים:

$$T' = t_i t_{i+1} \dots t_n t_1 t_2 \dots t_{i-1}$$

דוגמא:  $car \leftrightarrow arc$

נתונות 2 מחזורות  $T, T'$  באורך  $n$ . תארו אלגוריתם הבודק האם  $T'$  הינה סיבוב של  $T$ .

### פיתרון:

נשים לב ש- $T'$  סיבוב מעגלי של  $T$  אם הם  $T'$  מופיע ב- $TT$ .

$$TT = t_1 t_2 t_3 \dots t_{i-1} \underbrace{t_i t_{i+1} t_{i+1} \dots t_n t_1 t_2 \dots t_{i-1} t_i}_{T'} \dots t_n$$

← נריץ KMP עם טקסט  $TT$  ותבנית  $T'$ .

סיבוכיות:  $O(2n+n) = O(n)$ .