מבני נתונים ואלגוריתמים – תרגול 11#

התאמת מחרוזות

סימונים והגדרות:

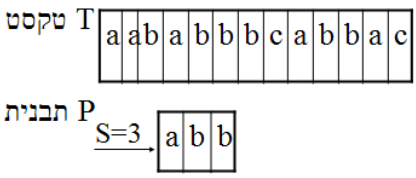
- **טקסט T** (מערך של תווים) באורך n – T[0,…,n-1] **ותבנית P** באורך m – P[0,…,m-1] כך ש- $m\leq n$.
- התווים של P ו-T נלקחים מאלפבית סופי Σ. לדוגמא: Σ={0,1}, {a,b,…,z}.
- Σ^* - קבוצת כל המחרוזות באורך סופי שניתן להרכיב מ-Σ.
- ε- המחרוזת הריקה. מתקיים ש- $\varepsilon \in \Sigma$

<u>בעיית התאמת מחרוזות:</u>

- 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,…,i-1].
- **רישא:** x רישא של y אם קיים z כך ש-y = xz (=שרשור של x עם z).
- **סיפא:** x סיפא של y אם קיים z כך ש-y = zx .

<u>הגדרה נוספת של הבעיה:</u>

- 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) → O(nm).

<u>**אלגוריתם רבין-קרפ**</u>

- מתייחסים למחרוזת כמספר (לפי ערך ה-ASCII של התווים).
- בגלל שמחרוזות הן מערכים, השוואה בין 2 מחרוזות תלויה באורך n שלהם, ולכן זה לא קבוע – O(n).
- הרעיון – נשתמש בפונקציית hash. נחשב את ערך ה-hash של התבנית, ונשווה אותו לערך ה-hash של כל תתי-המחרוזות בטקסט.

<u>מימוש נאיבי:</u>

- נחשב את H(P) – לוקח O(m) (m-אורך התבנית).
- עוברים על כל תתי המחרוזות *t* ב-T , מחשבים את H(*t*) ומשווים ל-H(P). O(nm).

<u>שיפור</u>: נשתמש ב-**Rolling Hash**

זוהי פונקציית Hash מיוחדת מחשבת את ערך ה-Hash עבור תת-המחרוזת הראשונה ב-T בזמן O(m) (כאורך התבנית), ואז מחשבת את תת המחרוזת הבאה מתוך תת-המחרוזת הראשונה ב-O(1) וכך הלאה.

סיבוכיות - O(n+m).

<u>אלגוריתם KMP</u>

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

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

דוגמא:

P = a b a b c

	0	1	2	3	4	5
Π	-1	0	0	1	2	0

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

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

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

וכן הלאה.

<u>הרעיון הכללי של האלגוריתם:</u> 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:

<u>KMP(P, T):</u>

```
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++
```

דוגמת הרצה:

j:	0	1	2	3	4	5	6
p[*j*]:	a	b	a	b	a	a	
b[*j*]:	-1	0	0	1	2	3	1

```
0 1 2 3 4 5 6 7 8 9 ...
a b a b b a b a a
a b a b a c
    a b a b a c
        a b a b a c
          a b a b a c
              a b a b a c
```

סיבוכיות: O(m+n).

הסברים נוספים:

http://www.inf.fh-flensburg.de/lang/algorithmen/pattern/kmpen.htm

בנוסף, KMP יכול לחשב את **σ – פונקציית הסיפא**.

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

לדוגמא:

T = a a a b a a c a a

P = a a b a a

	0	1	2	3	4	5	6	7	8	9
σ	0	1	2	2	3	4	5	0	1	2

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

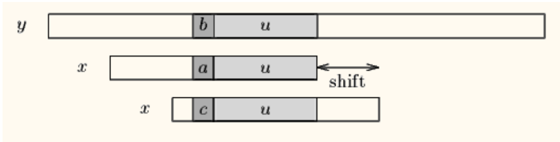
נחפש ב-$T_6 = a\ a\ a\ b\ a\ a$ את הסיפא המקסימלית שהיא רישא של P: $P_5 = a\ a\ b\ a\ a$.

אלגוריתם BM

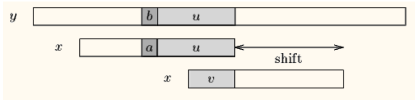
האלגוריתם משווה את התבנית מול תחילת הטקסט אך מתחיל את ההשוואה של מסוף התבנית עד לתחילתה. האלגורתם עושה preprocessing על התבנית ויוצר 2 פונקציות. מטרתן לקדם את התבנית תוך כדי "דילוג" על תווים שאנחנו יודעים בודאות שהתבנית לא תימצא בהם.

Good-suffix shift: נבדוק את הסיפא של תת-המחרוזת שבה הייתה התאמה. נקדם את התבנית בבטחה כך שהתווים בהם הייתה התאמה עדיין זהים.

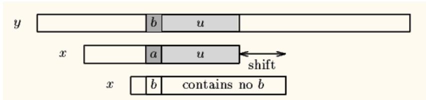
נניח ששי אי-האתמה בין x[i]=a לבין y[i+j] = b. כלומר יש התאמה בין הטקסט לתבנית מסוף התבנית עד ל-i+1 (תת-מחרוזת u). המטרה היא להזיז את התבנית כך שהמופע הבא הימני ביותר של u יהיה מול הטקסט במיקום j+i+1.



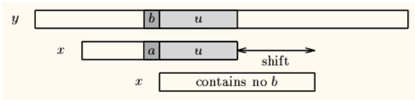
אם אין תת-מחרוזת כזאת, נזיז את התבנית כך שהסיפא הארוכה ביותר v של y[i+j+1…j+m-1] תהיה מול הרישא המתאימה של התבנית.



Bad-char shift: המטרה להזיז את התבנית עד למופע הבא של התו y[i+j] בתבנית.



אם אין, אפשר להזיז את כל התבנית מול y[i+j+1].



האלגוריתם בוחר את ההזזה המקסימלית מבין ה-GSS וה-BCS.

סיבוכיות: O(n+m).

דוגמא והסברים נוספים: http://www-igm.univ-mlv.fr/~lecroq/string/node14.html

תרגילים

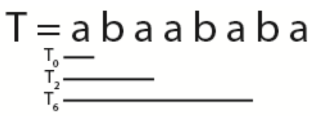
תרגיל:

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

T פלינדרום אם $T = T^R$.

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

לדוגמא-



אז נחזיר j=6.

פיתרון:

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

אם x פלינדרום אז $T^R = y^Rx$ (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_n.

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

<u>תרגיל:</u>

נתונה מחרוזת T שאורכה $n \geq 10$. תארו אלגוריתם המוצא חלוקה $T = xyx$ כך ש-$|y| \geq 10$ והאורך של x מקסימלי.

*הערה – תמיד קיימת חלוקה כזו, כיוון ש- x יכולה להיות ε.

דוגמא:

$$T = \underline{aabaa}\overbrace{aaabbcbbaaa}^{|y|=11}\underline{aabaa}$$

x = aabaa

<u>פיתרון:</u>

מה יכול להיות האורך המקסימלי של x? $|x| \leq \left\lfloor \frac{(n-10)}{2} \right\rfloor = k$

- בדוגמא : $k = \left\lfloor \frac{21-10}{2} \right\rfloor = 5$.

רוצים למצוא את הרישא המקסימלית של T באורך לכל היותר k שהיא גם סיפא של T.

← נריץ KMP כאשר התבנית היא T_k והטקסט הוא k התווים האחרונים ב-T.

סיבוכיות: O(n).

דוגמא –

P   T

T = aabaadaabbccccbbacaaabaa

c a a a b a a
| | | | |
a a b a a d a

|x| = 5

<u>תרגיל:</u>

הגדרה: מחרוזת T' היא סיבוב מעגלי של מחרוזת $T = t_1t_2 \ldots t_n$ אם קיים $1 \leq i \leq n$ כך שמתקיים:

$T' = t_it_{i+1} \ldots t_nt_1t_2 \ldots t_{i-1}$.

דוגמא: a r c <-> c a r

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

<u>פיתרון:</u>

נשים לב ש-T' סיבוב מעגלי של T אם"ם T' מופיע ב-TT.

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

← נריץ KMP עם טקסט TT ותבנית 'T.

סיבוכיות: O(2n+n) = O(n).

תרגיל:

נתונות 2 מחרוזות X,Y באורך n. X אנגרמה של Y אם קיימת תמורה $p \in S_{\{0,1,\dots,n-1\}}$ כך שמתקיים

$$Y = X[p(o)]\, X[p(1)] \dots X[p(n-1)]$$

כתבו אלגוריתם שמקבל T ו-P ו מוצא את כל תתי המחרוזות של T שהן אנגרמות של P.

פיתרון:

נריץ RB עם Rolling hash שמחשב את ערך ה-Hash על המחרוזת ללא תלות בסדר האותיות ואז אם x ו-y הן אנגרמות אז RH(x) = RH(y).

לדוגמא – לכל אות $a \in \Sigma$ נבחר מספר גדול N_a (כדי שסכום 2 מספרים יהיה ייחודי) ונגדיר:

$$RH(s) = \sum_{i=0}^{len(s)-1} N_{s_{[i]}}$$