# חיפוש מחרוזת

באלגוריתם הנאיבי, תבנית החיפוש נבדקת כל פעם מול הקלט, אז זזים תו אחד ובודקים שוב. בד"כ אפשר לראות בתווים הראשונים שהתבנית לא מתאימה למיקום מסויים, אבל לא תמיד זה ככה. למשל:

$$\begin{array}{ccccccccc} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 \\ A & B & C & A & B & C & A & B & C \\ A & B & C & A & B & C & A & B & D \end{array}$$

. גם במקום 1 וגם במקום 4 אנו בודקים שוב חלק גדול מהמחרוזת - וזה מיותר.

# אלגוריתם קנות-מוריס-פראט

הרעיון הוא שלא חוזרים אחורה. בכל פעם ש"נתקעים" בבדיקת המחרוזת - מחפשים את הרישא הכי ארוכה ששווה לסיפה, ולפי זה קופצים.

לוקחים את התבנית, ובונים טבלת הזזות שהאורך שלה $m$, ובמקום ה$i$ רושמים מה הרישא הארוכה ביותר בתבנית ששווה לסיפא. ניתן לבצע את זה ב$o\left(m^2\right)$, אבל קנות-מוריס-פראט מראים גם איך לבצע את זה בזמן לינארי.

אבל למה מכינים טבלה לתבנית ולא למחרוזת? הסיבה היא טרנזיטיביות של יחס השוויון.

---

יש לנו $T$ באורך $n$ מעל א"ב $\Sigma \cup \{\phi\}$. $P$ באורך $m$ מעעל $\Sigma \cup \{\phi\}$. מחפשים את כל המקומות בהם $P$ מופיע ב$T$, כאשר $\phi$ יכול להתאים לכל תו.

זה דומה לבעיית חיפוש מחרוזות, לכן נרצה לנסות את האלגוריתם של קנות-מוריס-פראט.

הבעיה היא שהטבלה שלנו היא של המקומות בתבנית מול המקומות בטקסט, והשתמשנו בטרנזיטיביות. הבעיה היא שכאן אין לנו טרנזיטיביות - $A = \phi = B$ אבל $A \neq B$.

נשים לב שזה דומה לכפל שני מספרים - שמים מספר אחד מעל השני, וכל פעם מזיזים, מכפילים ומחברים:

$$\begin{array}{rcccccc} T & = & a_0 & a_1 & a_2 & a_3 & a_4 \\ S & = & b_0 & b_1 & b_2 & & \end{array}$$

נגדיר

$$a \cdot b = \begin{cases} 0 & a = b \vee a = \phi \vee b = \phi \\ 1 & otherwise \end{cases}$$

לכן אם נכפיל את $T$ ב$S$ לפי הפעולה הזו, נקבל לא רק אם יש התאמה - אלא גם כמה שגיאות היו.

# בעיה

לא הורדנו כלום - גם הכפל הוא פעולה של $O\left(n^2\right)$.

### פתרון

יש לנו אלגוריתם FFT שמכפיל 2 פולינומים!

## עוד בעיה!

$FFT$ משתמש בשורשי היחידה - כלומר בעובדה שאנו משתמשים בשדה סגור אלגברי. אבל הכפל שלנו, לא רק שהוא לא נותן לנו שדה סגור אלגברי - הוא בכלל לא נותן לנו שדה.

### פתרון

אמנם, אנחנו לא יודעים להכפיל עם $FFT$ בכפל החדש, אבל אנחנו יודעים להכפיל מספרים רגילים.

נבחר $\Sigma = \{0, 1\}$.

## מסקנה ראשונה

בזמן $O(n \log m)$ בא"ב $\{0, 1\}$ אנחנו יודעים לספור את כל ההתאמות של "1"ים בתבנית עם הטקסט בכל אינדקס.

### לכן

קודם נבדוק התאמות של אחדים, ואז נהפוך את התבנית, ואז ההתאמות של אפסים יהיו אחדים ונוכל לספור אותם.

## הפונקציה האופיינית

$$\chi_a(x) = \begin{cases} 1 & x = a \\ 0 & otherwise \end{cases}$$

$$\Sigma = \{A, B, C\}$$

מספר ההתאמות יחושב ע"י

$$\chi_A(T) \times \chi_A(P) + \chi_B(T) \times \chi_B(P) + \chi_C(T) \times \chi_C(P)$$

## מסקנה

עבור א"ב $|\Sigma| = c$, ניתן לפתור את בעיית אי ההתאמות בזמן $O(c \cdot n \log n)$

## אבל מה אם יש לנו א"ב אינסופי?

מכיוון ש$P$ סופי, נוכל לפתור את הבעיה ב$O(mn \log m)$ - שזה פחות טוב מ$O(mn)$

## פתרון

אם כל תו מופיע פעם אחת, נבנה טבלה שתספור התאמות בתבנית לכל אינדקס בטקסט. בכל פעם שמציבים את התבנית במקום כלשהו על הטקסט, נקדם מונה קודם לפי האינדקס. כלומר אם אנחנו מציבים את התבנית במקום 15, והתבנית היא "$ABCD$", ובטקסט במקום 15 נמצא התו $D$, אז מכיוון שבתבנית זהו התו הרביעי - האינדקס שלו זה 3, ולכן נקדם את מונה $15-3=12$

אם כל תו מופיע $c$ פעמים, זה יתבצע ב$O\left(cn\right)$

## אלגוריתם

כל תו בתבנית שמופיע יותר מ$c$ פעמים נקרא <u>שכיח</u> ונספור התאמות שלו ע"י $FFT$.
כל שאר התוים נספור אותם יחד בזמן $O\left(nc\right)$.

סה"כ - $O\left(nc+\dfrac{m}{c}n\log m\right)$

#### אופטימום

$$nc=\frac{m}{c}\log m$$

$$c^2=m\log m$$

$$\boxed{c=\sqrt{m\log m}}$$