

## רבין קרפ

אחד האלגוריתמים הראשונים של Pattern Matching. הבעיה בצורה לא פורמלית - למצוא את  $P$  ב  $T$ . הרעיון - **finger print**: למצוא פונקציית hash  $h$  שמקיימת:

1. אם  $P = t_i \dots t_{i+m-1}$  אזי  $h(P) = h(i)$  כאשר  $h(i)$  הוא סימון ל  $h(i)$ .  
 $h(t_i \dots t_{i+m-1})$

2. אם  $P \neq t_i \dots t_{i+m-1}$  אזי בהסתברות גבוהה  $h(P) \neq h(i)$ .

3. בהינתן  $h(i)$  ו  $t_{i+m}$ , ניתן לחשב את  $h(i+1)$  "מהר".

## סיבוכיות תקשורת

יש שני שחקנים - Alice (A) ו Bob (B). לכל אחד יש ווקטור: לאליס יש את  $X = (x_0, \dots, x_{n-1})$  ולבוב יש את  $Y = (y_0, \dots, y_{n-1})$ . רוצים לדעת אם הווקטורים זהים או לא.

אליס צריכה לשלוח לבוב ביטים, כאשר כל ביט שהיא שולחת מאוד יקר (ולכן רוצים לצמצם את מספר הביטים), ובסופו של דבר, בוב צריך לדעת אם לשניהם יש אותו ווקטור או לא.

אפשר להראות בצורה די פשוטה שאם רוצים תשובה דטרמיניסטית שתמיד צודקת חייבים לשלוח  $n$  ביטים.

אם רוצים לשלוח פחות מ  $n$  ביטים חייבים להסתפק בתשובה הסתברותית, אבל עדיין רוצים הסתברות מאוד גבוהה.

אפשר להשתמש בפרוטוקול הבא:

- A: בחר באקראי מספר ראשוני  $p \in \text{PRIME}(n^2)$
- חשב:  $h(X) = \text{NUMBER}(X) \pmod{p}$ , כאשר  $\text{NUMBER}(X)$  הוא המספר המיוצג ע"י  $X$ .

$$\text{NUMBER}(X) = \sum_{i=0}^{n-1} x_i \cdot 2^i$$

- שלח ל  $B$  את  $(h(x), p)$ .

B: עושה את אותו חישוב עם  $p$  שהוא קיבל ועם הווקטור  $Y$  שלו ובודק אם יצא אותו דבר.

כמה ביטים שולחים?  $p \leq n^2$  ולכן  $2 \log n = \log n^2 \leq |p|$ . כמו כן  $h(x) < p$  ולכן  $|x| \leq |p| \leq 2 \log n$ . לכן סה"כ צריך לשלוח  $4 \log n$  ביטים.

$\text{PRIME}(n^2)^1$  זה כל המספרים הראשוניים בין  $2$  ל  $n^2$ .

## נכונות:

מה ההסתברות ש  $X$  ו  $Y$  שונים, אבל  $h(X) = h(Y)$ ?  
נאמר שמספר ראשוני הוא "רע" אם הוא מטעה את האלגוריתם. צריך לחשב את  $\frac{\text{number of bad primes}}{|\text{PRIME}(n^2)|}$ .  
יש משפט בתורת המספרים שאומר שלכל  $x > 67$ ,  $|\text{PRIME}(x)| > \frac{x}{\ln x}$ . זה אומר שבערך לכל  $\log$  מספרים יש מספר ראשוני - כלומר יש יחסית הרבה מספרים ראשוניים.  
אם  $n \geq 9 \iff n^2 > 67 \iff |\text{PRIME}(n^2)| > \frac{n^2}{2 \ln n}$   
נרצה לחסום את מספר הראשוניים הרעים. נקבע  $X$  ו  $Y$  ונסתכל על "p" שהוא רע:

$$\left. \begin{array}{l} \text{NUMBER}(X) = x' \cdot p + s \\ \text{NUMBER}(Y) = y' \cdot p + s \end{array} \right\} \implies h(X) = h(Y) = s$$

אבל  $X \neq Y$ . נגדיר:

$$w = \text{NUMBER}(X) - \text{NUMBER}(Y) = p \cdot (x' - y')$$

כלומר  $w$   $p$ . כמו כן  $\text{NUMBER}(X) < 2^n$  ולכן  $w < 2^n$ .  
יש משפט נוסף בתורת המספרים שאומר של  $w$  קיים פירוק יחודי:

$$(2^n >) w = p_1^{i_1} \cdot p_2^{i_2} \cdot \dots \cdot p_k^{i_k} \quad \forall_j i_j \geq 1, p_j < p_{j+1}$$

מכיוון ש  $\forall_j i_j \geq 1$ , אפשר להעריך אותם:

$$w = p_1^{i_1} \cdot p_2^{i_2} \cdot \dots \cdot p_k^{i_k} \geq p_1 \cdot p_2 \cdot \dots \cdot p_k$$

עכשיו, נשים לב ש  $p_1 \geq 2$ ,  $p_2 \geq 3$ ,  $p_3 \geq 5$  וכו' - ובאופן עקרוני  $p_i > i$ , ולכן אפשר לכתוב:

$$w = p_1^{i_1} \cdot p_2^{i_2} \cdot \dots \cdot p_k^{i_k} \geq p_1 \cdot p_2 \cdot \dots \cdot p_k > 1 \cdot 2 \cdot 3 \cdot 4 \cdot \dots \cdot k = k! > 2^k$$

מה זה  $k$ ? מספר הראשוניים הרעים, שמחלקים את  $w$ . אבל  $n > k$ , ולכן מספר הראשוניים הרעים  $n >$  לכן:

$$\frac{\text{Number of bad primes}}{|\text{PRIME}(n^2)|} < \frac{n}{n^2/2 \log n} = \frac{2 \log n}{n}$$

## תרגיל לבית

מה קורה בבסיס  $b$ : כאשר המספרים יכולים להיות  $\{0, 1, 2, \dots, b-1\}$ .

## חזרה ל Pattern Matching

הפונקציה  $h$  שהשתמשנו בה קודם מקיימת את שני התנאים הראשונים:

1. מחרוזות שוות נותנות hash שווה.

2. מחרוזות שונות נותנות hash שונה בסיכוי מאוד גבוה.

ומה עם התנאי השלישי? נראה איך אפשר לחשב במהירות את hash הבא מתוך hash הקודם:

- כדי להזיז את הביטים שנשארים מכפילים ב2
- כדי להוסיף את הביט החדש מוסיפים את  $t_{i+m}$
- כדי להעיף את הביט הישן מחסירים  $t_i \cdot 2^m$

$$h(i+1) = (h(i) \cdot 2 + t_{i+m} - t_i \cdot 2^m) \pmod{p}$$

בעיה: כדי לייצג  $2^m$  צריך  $\frac{m}{\log n}$  מילים. אבל נשים לב שאפשר לחשב מראש, פעם אחת, את  $2^m \pmod{p}$  ולשמור את זה לשימוש בשאר החישוב.

## יתרון על KMP

בKMP יש אוטומט, ולפעמים צריך לחזור אחורה. בעלות לשיעורין זה מתקזז, אבל worst case לפעולה יכול להיות הרבה זמן. ברבין קרפ יש זמן קבוע לכל תו. יש תוכניות של real time שבהן מוכנים להתפשר על טעות בשביל לקבל זמן worst case קטן לכל תו.