CPCFI

Lecture: 9.1 – STRING MATCHING I

Unit: 9 STING PROCESSING + COMPUTATIONAL GEOMETRY

Instructor: WÉSTON

9.1.1. KNUTH-MORRIS-PRATT'S ALGORITHM

MOTIVATING PROBLEM: FIND ALL OCCURRENCES OF A SUBSTRING P IN A LONG STRING T

- $|P| = m$
- $|T| = n$

→ [PREFIX FUNCTION] (STRING s, $n = |S|$)

↳ ARRAY π OF LENGTH n. $\pi[i]$: LENGTH OF THE LONGEST PROPER PREFIX OF SUBSTRING $S[0, \dots, i]$ WHICH IS ALSO A SUFFIX OF THIS SUBSTRING

- $\pi[0] = 0$

BY DEFINITION

PREFIX ≠ O.G. STRING

- $\pi[i] = \max\limits_{k=0,\dots,i}\{k : S[0,\dots,k-1] = S[i-(k-1),\dots,i]\}$

EXAMPLE:

$s =$ "ABCABCD"
$n = 7$
$\pi = [0, 0, 0, 1, 2, 3, 0]$
1 2 3 4 5 6 7

$s =$ "AABAAAB"
$n = 7$
$\pi = [0, 1, 0, 1, 2, 2, 3]$
1 2 3 4 5 6 7

* GO TO TRIVIAL ALGORITHM IN C++

↳ $O(n^3)$

↑

LET'S DO BETTER

① FIRST OPTIMIZATION

- THE VALUES OF π CAN ONLY INCREASE BY ONE AT MOST
- WHEN MOVING TO $\pi[i+1]$ ITS VALUE $+1$, -1 OR STAY THE SAME

PROOF: IF $\pi[i+1] > \pi[i]+1$ THEN WE CAN TAKE THE SUFFIX ENDING AT POSITION $i+1$ WITH LENGTH $\pi[i+1]$ AND REMOVE THE LAST CHARACTER FROM IT. WE END UP WITH A SUFFIX ENDING AT POSITION i WITH LENGTH $\pi[i+1]-1$ WHICH IS BETTER THAN $\pi[i]$!

→ $\pi[i]$ SHOULD BE THE LONGEST

$$\underbrace{\overbrace{s_0\ s_1}^{\pi[i]=2}\ s_2\ s_3}_{\pi[i+1]=4} \quad \dots \quad \underbrace{s_{i-2}\ \overbrace{s_{i-1}\ s_i}^{\pi[i]=2}\ s_{i+1}}_{\pi[i+1]=4}$$

$$\bigvee$$

$s_0 s_1 s_2 s_3 = s_{i-2} s_{i-1} s_i s_{i+1}$ $\quad \pi[i+1]=4$

$\downarrow$

$s_0 s_1 s_2 = s_{i-2} s_{i-1} s_i$ $\quad \pi[i]=3 \longrightarrow \pi[i]=2$!

② SECOND OPTIMIZATION

- IF $s[i+1] = s[\pi[i]] \longrightarrow \pi[i+1] = \pi[i]+1$

$$\underbrace{\overbrace{s_0\ s_1\ s_2}^{\pi[i]}\ \overbrace{s_3}^{s_3 = s_{i+1}}}_{\pi[i+1]=\pi[i]+1} \quad \dots \quad \underbrace{\overbrace{s_{i-2}\ s_{i-1}\ s_i}^{\pi[i]}\ \overbrace{s_{i+1}}^{s_3 = s_{i+1}}}_{\pi[i+1]=\pi[i]+1}$$

- IF $s[i+1] \neq s[\pi[i]] \longrightarrow$ TRY A SHORTER STRING j : $j < \pi[i]$ AND $s[0,\dots,j-1] = s[i-j+1,\dots,i]$

$$\overbrace{\underbrace{s_0\ s_1}_{j}\ s_2\ s_3}^{\pi[i]} \quad \dots \quad \overbrace{s_{i-3}\ s_{i-2}\ \underbrace{s_{i-1}\ s_i}_{j}}^{\pi[i]}\ s_{i+1}$$

$$\bigvee$$

IF WE FIND j, WE COMPARE $s[i+1]$ WITH $s[j]$. IF $s[i+1] = s[j] \longrightarrow s[i+1] = j+1$, OTHERWISE, WE KEEP DECREASING j UNTIL $j=0$:

IF $s[i+1] = s[0] \longrightarrow \pi[i+1] = 1$

OTW. $\pi[i+1] = 0$

KMP ALGORITHM $O(n) \longrightarrow$ DOES NOT PERFORM ANY STRING COMPARISONS

1) COMPUTE $\pi[i]$ ITERATING FROM $i=1 \rightarrow n-1$ $\quad (\pi[0]=0)$ $\quad$ [ZERO-BASED]

2) TO GET $\pi[i]$ WE SET $j = \pi[i-1]$, j DENOTES THE LENGTH OF THE BEST SUFFIX FOR $i-1$

3) TEST IF SUFFIX OF LENGTH $j+1$ IS ALSO A PREFIX BY COMPARING $s[j]$ AND $s[i]$

↳ IF $s[j] = s[i] \rightarrow \pi[i] = j+1$

OTW. $j = \pi[j-1]$ AND REPEAT

[KMP IS AN ONLINE ALGORITHM]

4) IF $j=0 \rightarrow \pi[i]=0$

* GO TO KMP.CPP

APPLICATIONS:

- [SEARCH FOR A SUBSTRING IN A STRING]
  - GIVEN A TEXT t AND A STRING s, DISPLAY ALL POSITIONS OF OCCURRENCES OF s IN t

    $$\left.\begin{array}{l} \circ\ n = |s| \\ \circ\ m = |t| \end{array}\right\} \quad m > n$$

  - GENERATE STRING $s\#t$ WHERE $\#$ IS A SEPARATOR $\# \notin s$, $\# \notin t$
  - CALCULATE PREFIX FUNCTION FOR $s\#t$
  - $\pi[i] \leq n \quad \forall i \in [0, ..., n+m]$ DUE TO THE SEPARATOR

    ↳ IF $\pi[i] = n \rightarrow$ s APPEARS COMPLETELY AT POSITION i AND APPEARS AT POSITION $\underbrace{i-2n}$ IN t

    $$i \in [0, ..., n+m] \qquad \downarrow \; i-(n+1)-n+1 = i-2n$$

EXTRA: STRING HASHING

→ MOTIVATING PROBLEM: COMPARE STRINGS EFFICIENTLY

○ BRUTE FORCE: $O(\min(s_1, s_2))$

○ EFFICIENTLY: $O(1)$ → CONVERT s_1 AND s_2 TO NUMBERS AND COMPARE BOTH NUMBERS

↙ HASHES

[IF TWO STRINGS ARE EQUAL, ALSO THEIR HASHES SHOULD BE EQUAL]

CONDITION ↗ HAS TO HOLD

↳ HOWEVER IF HASH(S) = HASH(T) THEN S AND T DO NOT NECESSARILY ARE EQUAL DUE TO COLLISIONS

<u>HASH STRING</u> : (OF LENGTH m)

$$\text{HASH}(s) = s[0] + s[1]p + s[2]p^2 + \cdots + s[n-1]p^{n-1} \mod m$$

$$= \sum_{i=0}^{n-1} s[i]p^i \mod m$$

- p AND m ARE CHOSEN, POSITIVE INTEGERS
- HASH(S) IS A POLYNOMIAL ROLLING HASH FUNCTION

- P SHOULD BE A PRIME NUMBER ≈ # OF CHARS IN THE INPUT ALPHABET

  ↳ P = 31 → ONLY LOWERCASE

  P = 53 → AND UPPERCASE

- $m = 10^9 + 9$

* GO TO STRING_HASHING.CPP

<u>APPLICATIONS</u> :

- [SEARCH FOR DUPLICATE STRINGS IN AN ARRAY OF STRINGS]

  ∘ n = |ARRAY| AND EACH STRING IS AT MOST m CHARACTERS LONG

  ∘ TRIVIAL ALGORITHM: SORT ARRAY AND COMPARE PAIR OF STRINGS → O(nm log n) SINCE EACH COMPARISON TAKES O(m)

  ∘ BETTER ALGORITHM: SORT ARRAY AND PERFORM COMPARISONS IN O(1) USING HASHES → O(nm + n log n)

  O(nm) TO COMPUTE HASHES

$$A = [\underset{\uparrow}{\text{"ABC"}}, \underset{\uparrow}{\text{"AA"}}, \text{"BC"}, \underset{\uparrow}{\text{"ABC"}}, \text{"AAA"}, \underset{\uparrow}{\text{"AA"}}]$$

- [FAST HASH CALCULATION OF SUBSTRINGS OF GIVEN STRING]

  ∘ FIND HASH(S[i ... j])

  ∘ BY DEFINITION, $\text{HASH}(S[i \ldots j]) = \sum_{k=i}^{j} S[k] \cdot p^{k-i} \mod m$

  $p^k = p^{k-i} p^i = p^{k-i+i}$

  ∘ MULTIPLYING BY p^i : $\text{HASH}(S[i \ldots j])\, p^i = \sum_{k=i}^{j} S[k] \cdot \boxed{p^k} \mod m$

$$= \text{HASH}(s[0 \dots j]) - \text{HASH}(s[0 \dots i-1]) \bmod m$$

- $\text{HASH}(s[0]) = 0$
- $\text{HASH}(s[0 \dots i]) = (\text{HASH}(s[0 \dots i-1]) + s[i]p^{i}) \bmod m$

EXTRA: RABIN-KARP ALGORITHM (STRING MATCHING) $O(|s|+|t|)$

→ <u>MOTIVATING PROBLEM</u>: GIVEN TWO STRINGS (PATTERN s AND TEXT t) DETERMINE IF s APPEARS IN t. IF IT DOES, ENUMERATE ALL ITS OCCURRENCES

<u>EXAMPLE</u>:

- PATTERN $s =$ "ABC" $n = |s|$
- TEXT $t =$ " AACCBCABCCCABBBCABCACC" $m = |t|$

0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21

$$\text{AACCBC}\underbrace{\text{ABC}}_{6}\text{CCABBBC}\underbrace{\text{ABC}}_{16}\text{ACC}$$

<u>TRIVIAL ALGORITHM</u>: ITERATE OVER EACH CHAR. IN t, IF $t[i] = s[0]$, THEN ITERATE OVER EACH CHAR j IN s AND INCREASE COUNTER IF $s_0 s_1 \dots s_n$ IS EQUAL TO $t_i t_{i+1} \dots t_{i+n}$

$O(nm)$

<u>BETTER ALGORITHM</u>: RABIN-KARP:

$O(n+m)$

1) CALCULATE HASH FOR s
2) CALCULATE HASH VALUES FOR ALL PREFIXES IN t
3) COMPARE EACH SUBSTRING OF LENGTH $|s|$ WITH PATTERN s