Hash Functions and String Matching

Sriram Sankaranarayanan

Data Structures and Algorithms

Rabin - Karp

Very important problem with lots of applications!

String Matching

- Does a given pattern $\boxed{\text{'P'}}$ of size m occur in a string 'S' of size $\boxed{n}$?
- Example:
  - P = "$\boxed{\text{GATTACA}}$" of size m = 7
  - S="GTAGATAGATTTAATTACGATTACATGATGTTGATTAGGATGATTACATATATGAATAATAGCGCCGATATAGAT"
  - Answer: P occurs in S at the position shown in red.

- Simple algorithm:
  - For each s = 1,…, n-m+1
    - Check if substring S[s],..,S[s+m-1] equals $\boxed{P}$

$$\Theta(\overset{1000}{m} \times (\underbrace{n}_{10^6} - m + 1))$$

$\approx 10^9$ oper

m

m

Speeding up Matching Using Hash-Functions

Idea: Hash the pattern P using hash function: $\boxed{h(P)}$ ← hash of this pattern

$S[1] \ldots\ldots S[s] \ldots S[s+m-1]$

$\xleftrightarrow{m}$

```
compute pattern hash: r = h(P)
for each s = 1,…, n-m+1
   compute: q = h(S[s]…S[s+m-1])
   if r = q:
         compare S[s]…S[s+m-1] with P.
```

← $\Theta(m)$ filtering

Running time: $\Theta(m \times (n - m + 1))$

S[i] . . .

Rolling Hash Functions

m

$s_m \; s_{m-1} \; s_{m-2} \; \cdots \; s_1$

prime number

$$h(s) = (p^{m-1} s_m + p^{m-2} s_{m-1} + \cdots + p^1 s_2 + s_1) \bmod M$$

×p ×p ×p

S

$s_m \; s_{m-1} \; s_{m-2} \; \cdots \; s_1 \; \hat{s}$

m

$$h(s') = (p^{m-1} s_{m-1} + p^{m-2} s_{m-2} + \cdots + p^1 s_1 + \hat{s}) \bmod M$$

$$= (\underbrace{p \times (h(s) - p^{m-1} s_m) + \hat{s}}) \bmod M$$

Updated

Precompute $p^{m-1} \bmod M$ and
perform two multiplications + one subtraction + one addition

Using Rolling Hash Function

S[1] . . . S[m-1]

compute pattern hash: $\boxed{r = h(P)}$
for each s = 1,…, n-m+1
    compute: q = $h(S[s] \ldots S[s+m-1])$ ⟵ $\Theta(1)$ use rolling hash function
    if r = q:
        compare S[s]…S[s+m-1] with P.

collision

Running time: Worst case continues to be $\Theta(m \times (n - m + 1))$

$m = 10^3$ $n = 10^6$

10^9

Assuming low probability of hash collision: we can improve the running time to $\Theta(m + n)$

$\boxed{\times 1000}$

10^6

Problem # 2 : Check if two strings have a common substring of size m.

- Inputs: Two strings S1 of size n1, and S2 of size n2.
- Output: True if S1 and S2 have a common substring of size m, FALSE otherwise.
- Example:
  - S1 = “GATATATACAGACAATAGATAGACACACGTAGGTGCACAGT”
  - S2 = “AGGATTTAGGTGGAACCCAGAGAGTTTAGGACCAGATTAGAT”
  - m = 5
  - Answer: True

Simple Algorithm

- for i = 1 to n1-m+1 m.
  - P = S1[i]…S1[i+m-1] ← hash
  - Use previous problem to search for pattern in S2 Rabin Karp
    - If pattern P found, then return True.
    - Else, continue.
  - Return False

Assuming good hash function:

$$\Theta(n_1 \times (m + n_2))$$

Worst Case:

$$\Theta\left(n_1 \times \left((n_2 - m + 1)\, m\right)\right)$$

Idea: Use a hash table and hash functions

However, we will need extra space $\boxed{\Theta(n_1)}$ for the hashtable.

Improved Algorithm

For i = 1 to n1 - m+1

- Compute rolling hash $\boxed{\mathrm{h_i}} = h(S1[i], \dots S1[i+m-1])$ $\Theta(n_1)$

Insert $\{(h_1, 1), (h_2, 2), \cdots, (h_{n_1-m+1}, \boxed{n_1 - m + 1})\}$ into perfect hash table H. $\Theta(n_1 - m + 1)$

For j = 1 to n2 – m + 1

- Compute rolling hash $r_j = h(S2[j] \dots S2[j+m-1])$ $\Theta(n_2)$
- Is key r_j in hashtable H?
- If yes, let k be the associated value with the key r_j
  - Compare S1[k]…, S1[k+m-1] with S2[j]…S2[j+m-1] $\Theta(m)$

$2 \cdot 10^6 + 10^3$

If there are no spurious collisions: $\Theta(n_1 + n_2 + m)$ Otherwise: $\Theta(n_1 + (\underbrace{n_2 - m + 1}_{10^6}) \times \underbrace{m}_{10^3})$ $\sim 10^9$

10^6