Hash Functions and String Matching

Sriram Sankaranarayanan

Data Structures and Algorithms

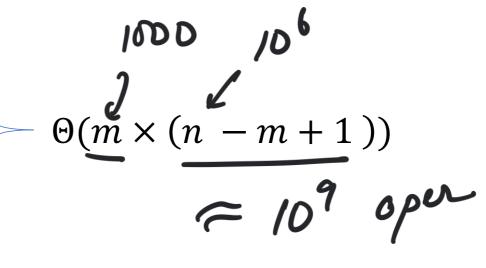
Rabin - Karp

String Matching

- Does a given pattern Pof size m occur in a string Sof of size ?
- Example:
 - P = "GATTACA" of size m = 7
 - S="GTAGATAGATTTAATTACGATTACATGATGTTGATTAGGATGATTACATATATGAATA ATAGCGCCGATATAGAT"
 - 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+m-1] equals P



Speeding up Matching Using Hash-Functions

Idea: Hash the pattern P using hash function: h(P)

```
compute pattern hash: r = h(P)

for each s = 1, ..., n-m+1

compute: q = h(S[s]...S[s+m-1])

if r = G:

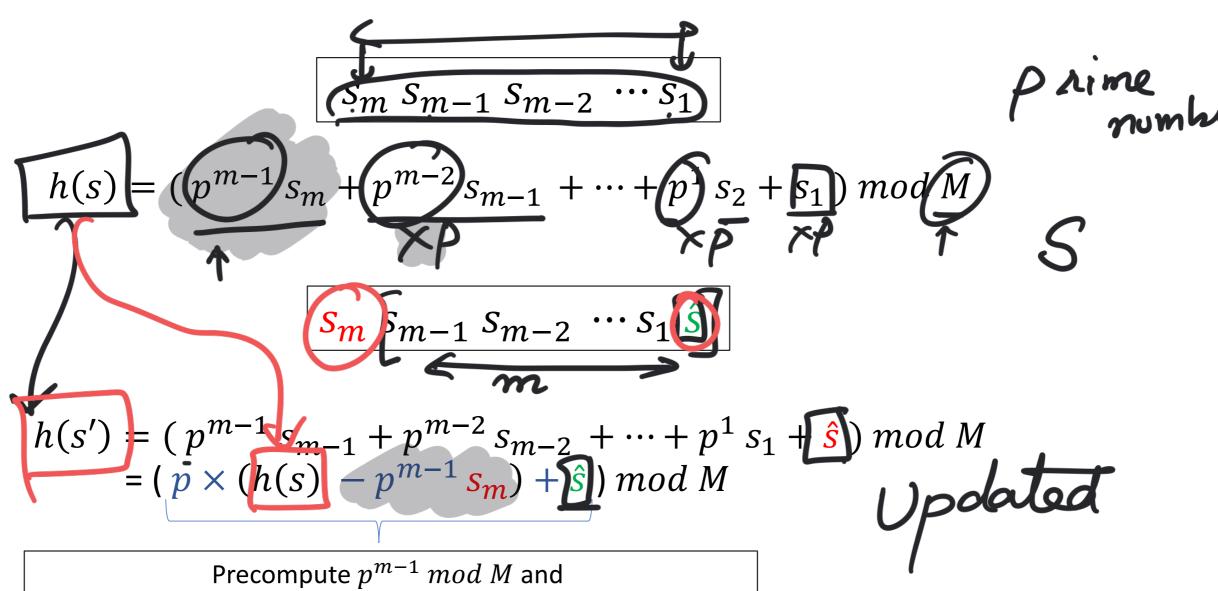
compare S[s]...S[s+m-1] with P.
```

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



Rolling Hash Functions

M



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



Using Rolling Hash Function

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

Running time: Worst case continues to be $\Theta(m \times (n - m + 1)) T$ $m = (0^3) \qquad n = (0^4)$

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

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 = "GATATATACAGACAATAGATAGACACACG<mark>TAGGTGCACAGT"</mark>
 - S2 = "AGGATTTAGGTGGAACCCAGAGAGTTTAGGACCAGATTAGAT"
 - m = 5
 - Answer: True

Simple Algorithm

- for i = 1 to n1-m+1

 - Use previous problem to search for pattern in S2
 - If pattern P found, then return True.
 - Else, continue.
 - Return False

Assuming good hash function:

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

Worst Case:

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

Idea: Use a hash table and hash functions

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

Improved Algorithm

```
For i = 1 to n1 - m+1
• Compute rolling hash h_i = h(S1[i], ...S1[i+m-1])
Insert \{(h_1(1), (h_2, 2), \cdots, (h_{n_1-m+1}, n_1-m+1)\} into perfect hash 0(n_1-m)
                                                                                                          \Theta(n_1 - m + 1)
    table H
    For j = 1 to n2 - m + 1
         • Compute rolling hash r_j = h(S2[j] ... S2[j+m-1])
         • Is key r_i h hashtable H?
         • If yes, let k be the associated value with the key r_i
              • Compare S1[k]..., S1[k+m-1] with S2[j]...S2[j+m-1]
                                                                          \Theta(m)
                  210 6 + 103
If there are no spurious collisions: \Theta(n_1 + n_2 + m) Otherwise: \Theta(n_1 + (n_2 - m + m))
```