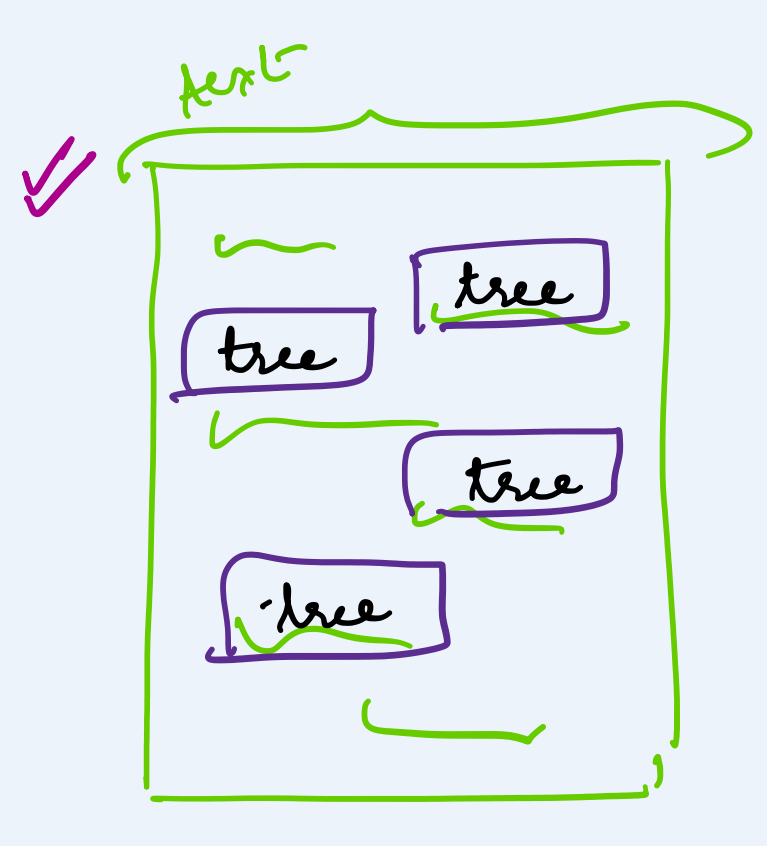
Monday, 18 October 2021 8:32 PM

String Algorithms



→ Ctrl + F

tree → pattern

size of pattern << size of text

① Naive pattern matching

text = aabaacaadaabaabc → $T[1 \ldots n]$

pat = aaba → $P[1 \ldots m]$

$m << n$

Time Complexity

① Best case m = length of text

t → aabccaaddec

p → fmaa

TC → $O(m)$

② Worst case

→ when all the chars of pat & text are same.

t → aaaaaaaa

p → aaa

TC → $O((m-n) * m)$

⇓

$O(m * n)$

Space Complexity: $O(1)$

Implementation :

Instead of matching the whole char of pattern in the text, compare the hash value.

text → aaba (43)

string aabc → Black box → hash value 43 } $O(1)$

text → aabaacaadaabaabc

pat ⇒ aaba

① pass the pat to the black box.

② From the text, take equal no of characters = to the pattern and pass it to the black box.

Blackbox → 43, 37, 37

37 = 37

43 ≠ 37

Earlier for each window,

t		p
a	→	a
a	→	a
b	→	b
a	→	a

4 comparisons

Naive → (T*P)

Total windows → $(T - P + 1)$

For each window, you just have to compare the hash value.

→ $(T - P + 1) * 1$

→ $(T - P + 1)$ → $T >> P$

→ $O(T)$

How does this black box does the computations ??

unique string → Black box → unique hash value

Black box → Hash Function

string → int

abc → 97 + 98 + 99 ⇒ 294

adding ascci values

abc → 294

Hash Function

$str[i] + str[i+1] ---- + str[i+n]$

abc, bac → (x) = 294

If a hash function, gives same hash value for two different strings, then it is the case of collision.

If for a hash function, the frequency of collision is very high, then it is not a good hash function.

Calculation of hash of a string

The good and widely used way to define the hash value of a string (s), of length (n) is,

$$hash(s) = [s[0] + s[1] * p + s[2] * p^2 + s[3] * p^3 + \ldots s[n-1] * p^{n-1}] \bmod m$$

$$= \left[\sum_{i=0}^{n-1} s[i] * p^i\right] \bmod m$$

p, m → chosen positive integers

polynomial hash function

p ≈ prime no ≈ no. of char input alphabet.

Eg: lowercase chars

Eg: p = 31

m → should be a large no. since the probability of two random strings to be colliding is $\frac{1}{m}$.

string → Blackbox → polynomial hash function → value

$O(p)$

t → aabacadaab

p → aaba

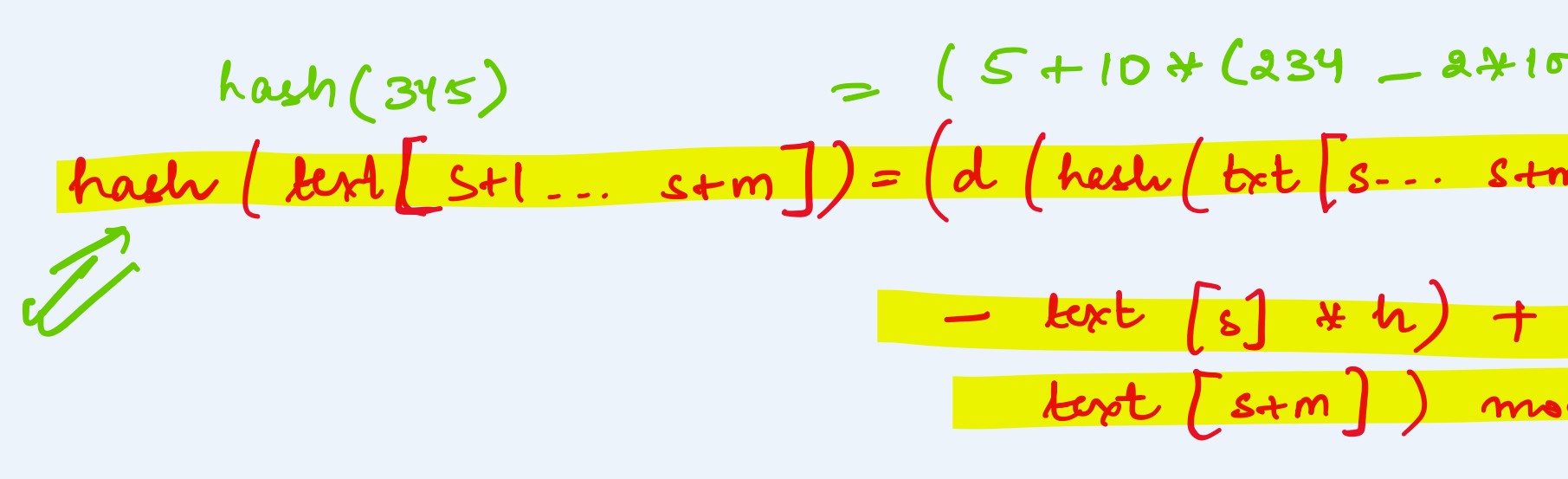
For all windows → $(T - P + 1)$ passing all windows to black box.

one computation of black box = $O(P)$

TC → $(T - P + 1) * P$

→ $O(T * P)$

Rolling Hash



s → index

length (pattern) = m - 1

previous window → (s... s+m-1)

new window → (s+1 .... s+m)

hash(345) = (5 + 10 * (234 - 2*100)) mod q

hash (text[s+1 ... s+m]) = (d (hash(txt[s... s+m-1])

− text [s] * h) +

text [s+m]) mod q

Here, hash (txt[s+ ... s+m-1]) = Hash value of str at idx (s)

Hash (txt [s+1 .... s+m]) = Hash value at index (s+1)

d → no of char in the alphabet

q → prime no.

h → d^(m-1) (m → length of pattern)

text → 23456

234

length (pattern) → ③

Since, the length of pattern ③, then the length of each window in the text → ③.

234 → BB → 234

① Calculate the hash value of pattern

② Calculate hash value of first window of text.

TC →

Best case → $O(T - P + 1) * 1$

Worst case → $O(T * P)$