String Hashing

Tries

Trees &

Binary numbers

kmp / 2 algorithms

string matching

1 or 2

100 — 30 — 10 — 68-69

string problems

hashing

tries

- Priyansh Agarwal

100 kmp / z algorithm problems

↓ 99% of these

can also be solved using

Hashing

String matching $\underline{O(1)}$

"Priyansh" $\propto$ "Priyansh"

$O(n)$ $O(m)$

$O(\min(n, m))$

"abcabcabcabc" $\longrightarrow$ S $\longrightarrow$ $\underline{O(n)}$

how many times does the string "abc" come as a substring $\longrightarrow$ P $\longrightarrow$ $\underline{O(m)}$

$\underline{O(n \times m)}$

abc abc abc abc → S

1 1 m 1 1

abc → P

m

substring(i, i+m-1) in S

compare with P

$O(n \times m)$

1. iterate over all indices → $O(n)$
2. compare the current substring with the pattern → $O(m)$

$O(n \times m)$

$O(\log m)$

$O(1)$ time

① kmp | z

①

$O(n+m)$ time

worst case T.C

10^5

② Hashing

$O(n \log m)$ time

$O(n)$ time

100

Why String Hashing?

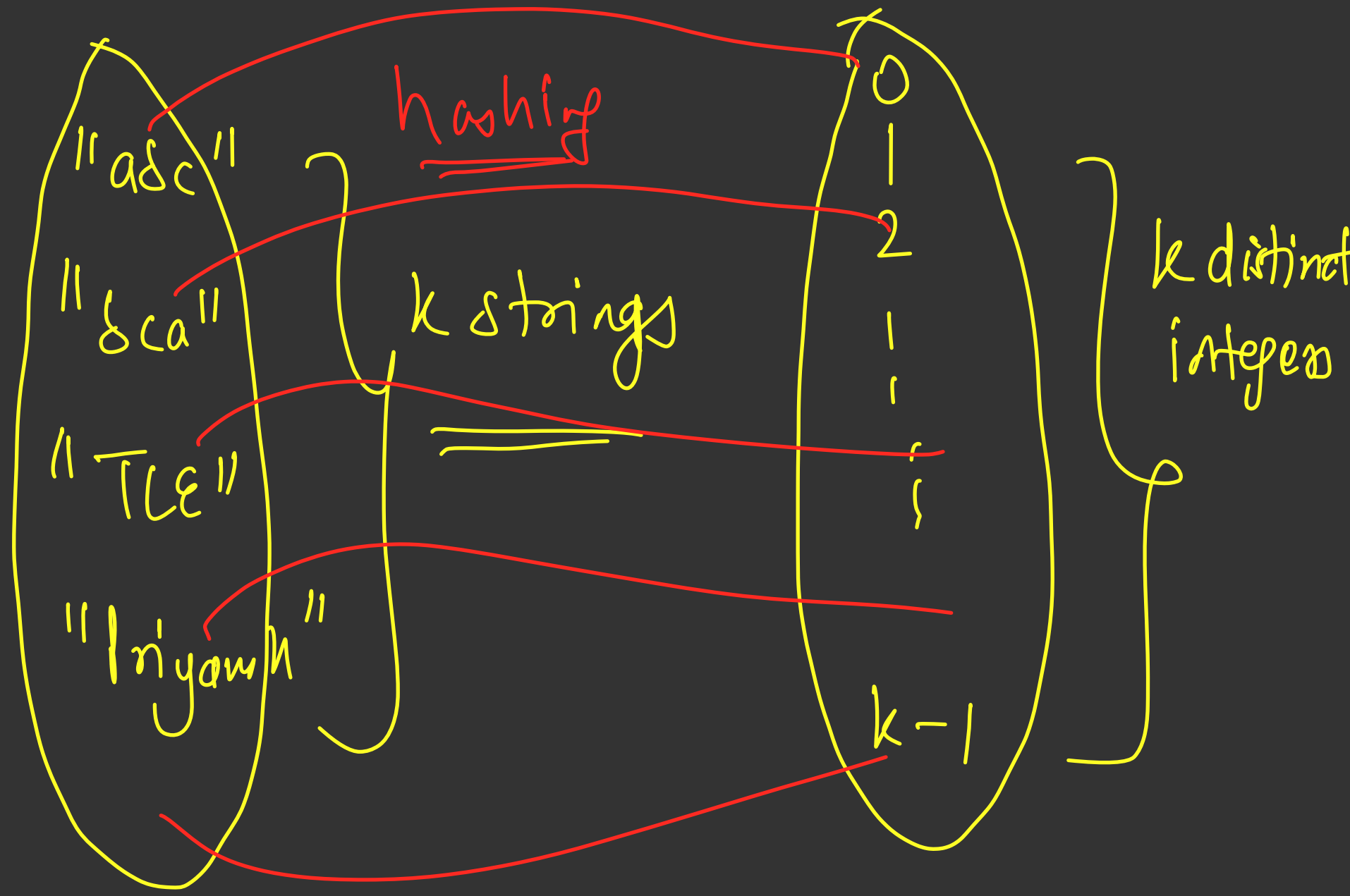
- Optimizing Brute Force solutions
- Comparing 2 strings
- Daddy of String Algos

A _ _ _ _ _ _ _ _ _ n characters

B _ _ _ _ _ _ _ _ _ n

$O(n)$

whether A == B or not


hashing
"abc"
"bca"
"TCE"
"Priyanh"
k strings
0
1
2
i
k-1
k distinct integers

"abc" → 0

"bca" → 1

"aaa" → 2

"aac" → 3

"abc"

↓ hashing function

0

a

b

c

size 100

26^{100}

10^{18} integers

2

log long

Given a string $\longrightarrow$ map it to an integer

"Priyansh ....." $\longrightarrow$ 6

$\downarrow$

"....." $\longrightarrow$ 7

Requirements

every possible string cannot be mapped to a unique integer

always

- If a == b, Hash(a) == Hash(b)
- If a != b, Hash(a) != Hash(b)

a and b are strings

  - If this is true with a very high probability we are good to go
- Calculating Hash function should be fast enough
- Hash of a string shouldn't change in the code
- Hash value should be itself O(1)

"abc" → 1

"baa" → 2

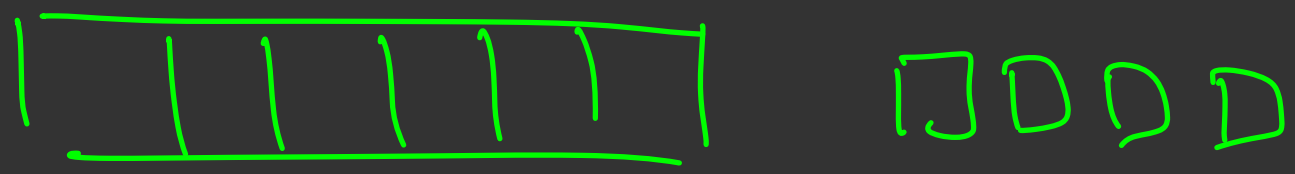
"caa" → 2

"aba" → 3

$$\alpha \begin{cases} 1 \\ 2 \\ 3 \end{cases}$$

n strings $\longrightarrow$ m integers

$\underline{n > m}$



m boxes

n balls

$n > m$

if we want to put all balls in m boxes then atleast 1 box must have more than 1 ball

n strings

m integers

rand_int(0, m) → 0 to m-1

rand_int(0, m) → 0 to m-1

"abc"

"bca"

"cab"

"aac"

1 2 3

1/3 1/3 1/3

1

2

3

rand-int(1,3)

n strings

m integer

$\boxed{a} \; != \; \boxed{b}$

what is the probability that

hash(a) == hash(b)

A $\quad 1 = \quad$ B

but

hash(A) = hash(B)

1	① $1/m \cdot 1/m \rightarrow 1/m^2$		1
2	② $1/m \cdot 1/m \rightarrow 1/m^2$		2
3	③ $1/m \cdot 1/m \rightarrow 1/m^2$		3
4			⋮
⋮			
m	⋮		m
	⋮		
	ⓜ $1/m \cdot 1/m \rightarrow 1/m^2$		

$$\left(\frac{1}{m^2} + \frac{1}{m^2} + \frac{1}{m^2} \cdots\right)^{m \text{ times}} \rightarrow \underline{1/m}$$

$$\underline{m = 10^9}$$

$$\boxed{1/10^9}$$

$$\left\{ n \text{ strings} \begin{array}{c} 1 \\ 2 \\ \vdots \\ 10^9 \end{array} \right\}$$

(A , B)

if (A != B but h(A) = h(B))

(A == B) hash(A) == hash(B)

"abc" → 1

"bac" → 2

"aac" → 3

"baa" → 1

1

2

3

is "abc" == "baa"

1 1

What is the probability that this fails

if ($A \mid_{\sigma} = B$ but $hash(A) == hash(B)$)

$hash(A) \longrightarrow$	1	$1/m$	$hash(B) \rightarrow$	1	$\rightarrow 1/m$
	2	$1/m$		2	$1/m$
	3			3	
	⋮				
	m				

if my algorithm has a probability of failing as $1/m$

how many times can I run my algorithm on average before it fails

m comparisons

Coin — H $1/2$ $\longrightarrow$ 2

Coin — T $1/2$

Probability of event happening = p

then

average no. of tries required for event to happen is $1/p$

A and B & my

we are comparing A and B

↓

$1/m$ average

10^8

$m \to 10^9$

$1/10^9$

$\boxed{10^9}$

10^8

Codeforces problem $\longrightarrow$ 100 test cases

every test case require 10^8 comparisons

$\underline{10^{10}}$ comparisons $\underline{10^9}$

$S \longrightarrow$ randomly generating an integer from 1 to m

Polynomial Rolling Hash

$$\boxed{\text{hash}(s)} = s[0] + s[1] \cdot p + s[2] \cdot p^2 + \ldots + s[n-1] \cdot p^{n-1} \mod m$$

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

if (A == B) then

hash(A) = hash(B)

(p, m)

$$hash(s) = (s_0 \cdot p^0 + s_1 \cdot p^1 + s_2 \cdot p^2 \ldots s_{n-1} \cdot p^{n-1}) \% m$$

<u>p and m</u> → range of integers allowed

(0 to m-1)

① if $A = B$ then

$hash(A) = hash(B)$ with probability 1

② if $A \neq B$ then $hash(A) \neq hash(B)$ with a very high probability

③ $hash(\text{any string}) \longrightarrow \underline{0 \text{ to } m-1}$

m distinct integer

Assume that mod m did not exist

'asc'

'baa'

292

300

a

b

27

a b c

b a a

1

2

3

d → 4

e → 5

$a \to 1$

$b \to 2$

$c \to 3$

$\vdots$

$z \to 26$

a b z

1 2 26

$$\boxed{1 \cdot 1 + 2 \cdot 27 + (26)\, 27^2}$$

A

B

integer

$$292 \longrightarrow \boxed{2 \cdot 10^2} + 9 \cdot 10^1 + 2 \cdot 1$$

$$300 \longrightarrow \boxed{3 \cdot 10^2} + 0 \cdot 10^1 + 0 \cdot 1$$

$$\begin{array}{ccc} a & b & c \\ \swarrow & \downarrow & \searrow \\ 1 & 2 & 3 \\ \downarrow & \downarrow & \downarrow \\ 1 & 27 & 27^2 \end{array}$$

1 0 1 0 1

0 0 1 0 1

base 2

$$1 \cdot 2^4 + 0 \cdot 2^3 + 1 \cdot 2^2 + 0 \cdot 2^1 + 1 \cdot 2^0$$

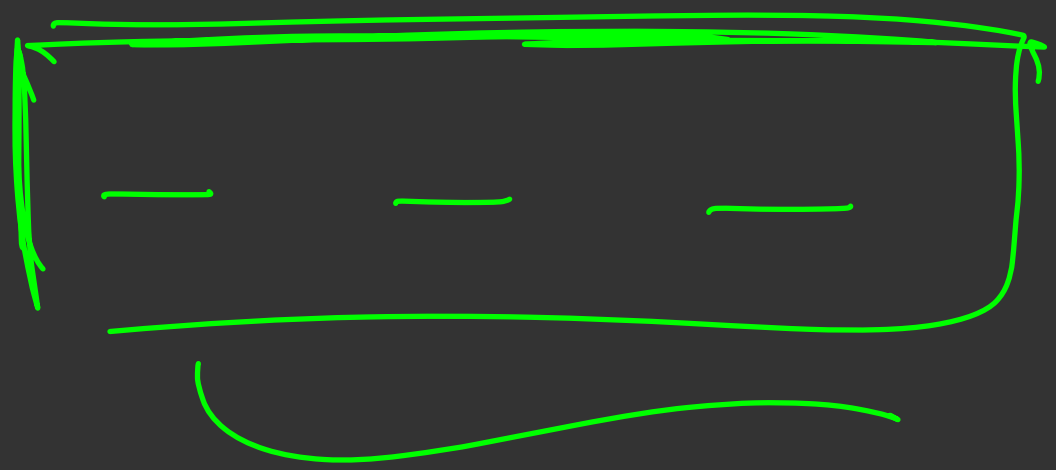
$$0 \cdot 2^4 + 0 \cdot 2^3 + 1 \cdot 2^2 + 0 \cdot 2^1 + 1 \cdot 2^0$$

3 9 2 4

2 9 2 4

$$3 \cdot 10^3 + 9 \cdot 10^2 + 2 \cdot 10^1 + 4 \cdot 10^0$$

$3 \cdot 10^3$



3000

$a \quad b \quad c \quad d$

$27^3 \quad 27^2 \quad 27^1 \quad 27^0$

$\_\_\_ \quad \_\_\_ \quad \_\_\_$

$\boxed{26^3}$

$$\text{hash}(s) = \left[s_0 \cdot p^0 + s_1 \cdot p^1 + s_2 \cdot p^2 \cdots s_{n-1} \cdot p^{n-1} \right] \% m$$

$p >$ no. of unique characters

$\boxed{(p, m)}$

string can contain characters a, b, c

3 unique characters

if $p = 10^5$ and $m \simeq 10^9$

$$\text{hash}(s) = \left[s_0 \cdot p^0 + s_1 \cdot p^1 + s_2 \cdot p^2 \cdots s_{n-1} \cdot p^{n-1} \right] \% m$$

p should not divide m

m should not divide p

} p > unique characters

$m \simeq 10^9$

fermat's theorem

$$a^{m-1} \equiv 1 \bmod m$$

for m = prime

$$a^{m-2} \equiv a^{-1} \bmod m$$

Strings only contain lower case letters

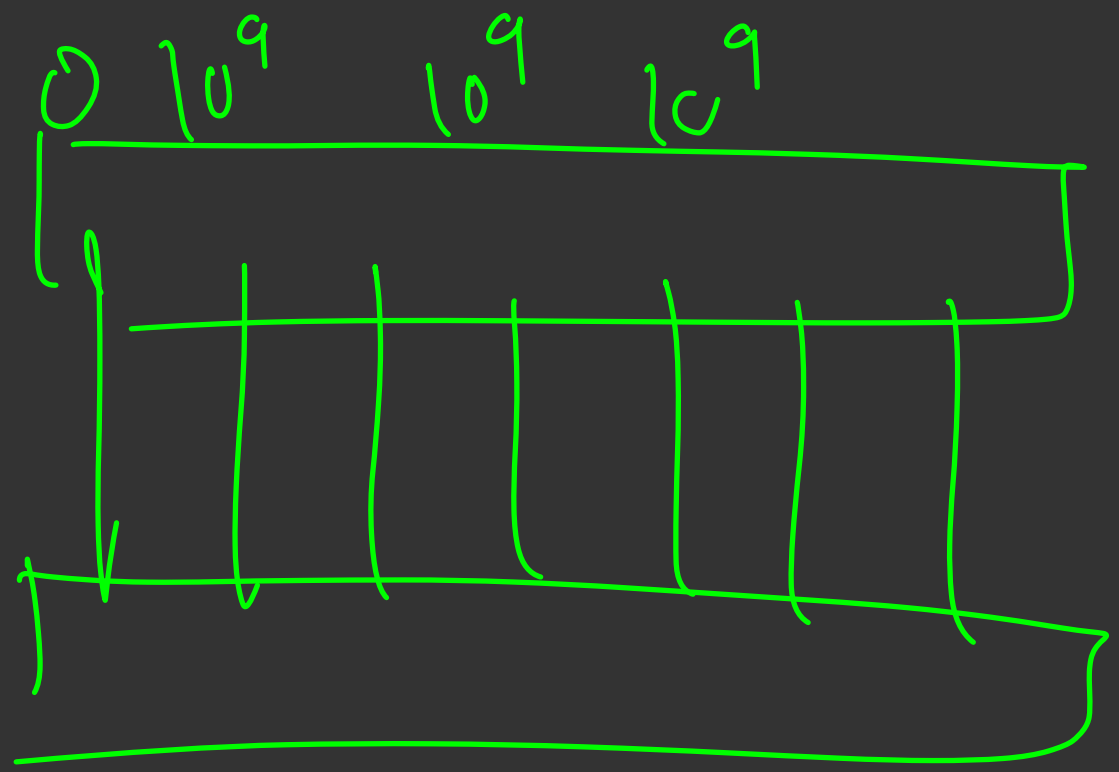
$l = 27$, $m = 10^9 + 7$

Strings with upper case & lower case

$l = 60$, $m = 10^9 + 7$

$f = 1001$

$m = 10^9 + 7$



$$f = \frac{10^9 + 5}{m = 10^9 + 7}$$

Hashing a string

$$hash(s) = (s_0 \cdot p^0 + s_1 \cdot p^1 \cdots s_{n-1} \cdot p^{n-1}) \% m$$

$p \rightarrow$ unique character

$\underline{\underline{0 \text{ to } 10^9}}$ $\quad m \simeq 10^9$ and prime

$$hash(arr) = (a[0] \cdot p^0 + a[1] \cdot p^1 \cdots a[n-1] \cdot p^{n-1}) \% m$$

$p \rightarrow$ unique elements $\quad p \rightarrow 10^9 + 3$

Hash of a string $\underline{\underline{S}} = H$

n

find hash of string $\underline{S + ch}$

$(H + ch \cdot p^n) \% m$

$$hash(s) = (s_0 \cdot p^0 + s_1 \cdot p^1 \cdots s_{n-1} p^{n-1}) \% m$$

$$hash(s + ch)$$

$$\rightarrow (H + ch \cdot p^n) \% m$$

hash of a string S is H

What is the hash of

(ch + S)

$$\text{hash}(s) = (\underline{s_0 \cdot p^0} + \underline{s_1 \cdot p^1} \cdots s_{n-1} p^{n-1}) \% m$$

$$\rightarrow H$$

$$\text{hash}(ch + s)$$

$$= (ch \cdot p^0 + \underbrace{s_0 \cdot p^1 + s_1 \cdot p^2 \cdots s_{n-1} \cdot p^n}) \% m$$

$$= (ch + \underline{(H \cdot p)}) \% m$$

Why Rolling?

- Hash(s) = H, what is Hash(s + x), x = character
  - $(H + xp^n) \% m$ → logn time
- Hash(s) = H, what is Hash(x + s), x = character
  - $(x + Hp) \% m$ → o(1) time
- How about calculating the Hash of a substring quickly?
- How about just comparing two strings?

hash of a substring :

Given a string calculate its hash of size n

→ Time = $O(n)$ time

$O(1)$ time to add character at back

$O(1)$ time

$$\text{hash}(S) = \left(S_0 \cdot p^0 + S_1 \cdot p^1 \cdots S_{n-1} \cdot p^{n-1}\right) \% m$$

hash of substring from l to r

$$\underline{\left(S_l \cdot p^0 + S_{l+1} \cdot p^1 \cdots S_r \cdot p^{r-l}\right)}$$

$$\text{hash}(s) = \left[s_0 \cdot p^0 + s_1 \cdot p^1 \cdots s_{l-1} \cdot p^{l-1} + s_l \cdot p^l + \cdots s_r \cdot p^r + \cdots s_{n-1} \cdot p^{n-1} \right] \% m$$

$$\left[s_l \cdot p^0 + s_{l+1} \cdot p^1 \cdots s_r \cdot p^{r-l} \right] \% m$$

$\text{hash}[i]$

$$\rightarrow \quad s_0 \cdot p^0 + s_1 \cdot p^1 \; \text{----} \; s_i \cdot p^i$$

$$\text{hash}(r) - \text{hash}(l-1) = \left[\frac{s_l \cdot p^l + s_{l+1} \cdot p^{l+1} \; \text{--} \; s_r \cdot p^r}{p^l}\right]$$

$$(s_0 \cdot p^0 + s_1 \cdot p^1 \; \text{----} \; s_r \cdot p^r)$$

$$- \; (s_0 \cdot p^0 + s_1 \cdot p^1 \; \text{--} \; s_{l-1} \cdot p^{l-1})$$

hash(substring from l to r)

$$= \left[\frac{[\text{hash}[r] - \text{hash}[l-1]]}{\boxed{p^l}} \right] \% m$$

$$= \left[(\text{hash}[r] - \text{hash}[l-1]) \cdot \underline{\text{modinverse}(p^l)} \right] \% m$$

$\underline{O(\log m)}$

$$a \longrightarrow \left(\frac{1}{a}\right) \% m$$

$\hookrightarrow$ primo

$$\left(a^{m-2}\right)$$

$$\left(p^{l}\right) \rightarrow \left(\frac{1}{p^{l}}\right) \longrightarrow \left(p^{l}\right)^{m-2}$$

"abcabcabcabc" $\longrightarrow$ $O(n)$

find out how many abc occurs $O(m)$

$O(n) + O(m)$

Pitfalls?

10^{10} 10^{9}

- A != B but Hash(A) == Hash(b)
  - Probability = 1 / m
- Comparing 50 such strings, probability of a collision = 50 / m
- Let's look at an example problem to see how it fails
  - Substring comparison problem
- Solution????

What if you're doing 10^7 comparisons for $m = 10^9$

$$\frac{1}{10^9} \longrightarrow 10^7 \qquad \frac{10^7}{10^9} = \underline{1/100}$$

hash1 $\longrightarrow$ $p = 30$, $m = 10^9 + 7$

hash2 $\longrightarrow$ $p = 30$, $m = 10^9 + 9$

Single Integer $\longrightarrow$ $1/m$ $\longrightarrow$ $\underline{\underline{1/m^2}}$

10^{12} 10^{12} m

$(a \times b) \% m \longrightarrow$ overflow

$(a \times b) \% m$

Problems

- Number of different substrings
- Palindrome queries
- Largest string which repeats twice
- Longest palindromic substring

$O(n^2)$

5 3 1

6 4 2

5 length

3 len

1 —

a a b a b a b

5

4

3

2

1

Binary search on length of string

$O(\log n)$

$\underline{\underline{k}}$ for(int i=0; i≤n-k; i++)

Binary search on length $O(\log n)$

$\longleftrightarrow$ map<int, int> occur

```
for (int i=0 ; i<= n-k ; i++)
    int hash = substringHash(i, i+k-1)
    if occur.find(hash) != occur.end()
```

Longest Palindromic substring

5
3
1

6
4
2

6

```
for (int i=0 ; i<=n-k ; i++)
    int first half =
```

original string

a b c c b a

reverse string

$\longrightarrow$ x abc cbay z

z y abcabc x

$S \rightarrow xypqr\boxed{abc}cbcmn$

$S_{rev} \rightarrow nm\boxed{abc}cbarqpy$