String Hashing

張晏誠

https://shorturl.at/gN77Z

Example

- There are n strings, and the following q queries
- For each query, answer whether s_i is identical to s_j
- $q \le 10^5$, the total length of strings $\le 10^6$

Brute Force Solution

Check every character of s_i and s_j.

Brute Force Solution

Time complexity: $O(max\{|s_i|\} \times q)$ $\rightarrow$ TLE

String Hashing

Do hashing for each strings:

$$h_i = H(s_i)$$

If s_i and s_j are identical, then $h_i = h_j$.

We introduce a simple method called “Rolling Hash”.

Rolling Hash

Let the characters of a string be the coefficients of polynomial. That is, convert the string to ***x***-based numeral system.

Ex：”abc” is mapped to $2x^2 + x$

Rolling Hash

If ***x*** is large enough (larger than size of character set), then ***H(s)*** will be an 1-1 mapping.

However, it is hard to achieve, since we only has restricted spaces.

Rolling Hash

So the ***H(s)*** must be in a range ***[0, m)*** for some ***m***

→ Modulo ***H(s)*** by ***m***

Rolling Hash

Hence, given ***x***, ***m***, we define

$$H(s) = \sum_{i=0}^{|s|-1} s[i] \times x^i \mod m$$

Rolling Hash

We usually pick a small prime for ***x*** and a large prime for ***m***, where ***x*** is larger than the size of character set.

Ex.
If the strings only contain lowercase alphabet, pick ***(31, 1000000007)***
If the strings contain lowercase and uppercase, pick ***(53, 1000000007)***

Rolling Hash

Note that the hashing function is not 1-1, so collision might occur.

If you are worried about it, use more pairs of ***(x, m)*** to check (?)

Rolling Hash

```
pw[0] = 1;
for(int i = 1; i < MxLen; i++)
    pw[i] = pw[i - 1] * M % mod;
string s;
cin >> s;
long long hash = 0;
for(int i = 0; i < s.size(); i++)
    hash = (hash + s[i] * pw[i] % mod) % mod;
```

Rolling Hash

Time complexity : $O(|s|)$

Fun With Strings

- Given ***n*** strings and ***q*** operation
- Each operation is in the following form:
  - $s_i := s_i + s_j$, here + means concatenation
  - Answer if s_i is identical to s_j
- $n, q \leq 100, \Sigma|s_i| <= 5000$

Fun With Strings

For the second operation, we can solve it simply by Rolling Hash.

```
cout << (hash[x] == hash[y] ? "Y" : "N") << '\n';
```

Fun With Strings

For the first operation, by the definition of hashing function :

$$H(s_i + s_j) = \sum_{k=0}^{|s_i|-1} s_i[k] \times x^k + \sum_{k=0}^{|s_j|-1} s_j[k] \times x^{|s_i|+k} \mod m$$

$$= H(s_i) + H(s_j) \times x^{|s_i|} \mod m$$

Fun With Strings

We can solve it by memorizing the length of each string.

```
hash[x] = (hash[x] + hash[y] * pw[len[x]] % mod) % mod;
len[x] += len[y];
```

Fun With Strings

If $|s_i|$ is too large that can't be stored in long long,
using *Fermat's Little Theorem*, we can modulo $|s_i|$ by ***m-1*** and use fast exponentiation to calculate $x^{|s_i|}$.

$$H(s_i + s_j) = \sum_{k=0}^{|s_i|-1} s_i[k] \times x^k + \sum_{k=0}^{|s_j|-1} s_j[k] \times x^{|s_i|+k} \mod m$$

$$= H(s_i) + H(s_j) \times x^{|s_i|} \mod m$$

Fun With Strings

Time complexity:

$O(\Sigma|s| + q)$ or
$O(\Sigma|s| + q \log m)$ if you apply the method on the previous page
(or $O(\Sigma|s| + q + \sqrt{m})$ using BSGS)

Fun With Strings

Code :

(Full version)

```
const int N = 110;
const int MxLen = 5010;
const long long mod = 1000000007, M = 37;

int n, q;
long long pw[5010];
long long hash[N], len[N];

void solve(){
    pw[0] = 1;
    for(int i = 1; i < MxLen; i++)
        pw[i] = pw[i - 1] * M % mod;
    cin >> n >> q;
    for(int i = 0; i < n; i++){
        string s;
        cin >> s;
        len[i] = s.size();
        for(int j = 0; j < len[i]; j++)
            hash[i] = (hash[i] + s[j] * pw[j] % mod) % mod;
    }
    while(q--){
        char c;
        int x, y;
        cin >> c >> x >> y;
        x--, y--;
        if(c == 'E')
            cout << (hash[x] == hash[y] ? "Y" : "N") << '\n';
        else{
            hash[x] = (hash[x] + hash[y] * pw[len[x]] % mod) % mod;
            len[x] += len[y];
        }
    }
}
```

Fun With Strings

Note :
"***hash***" has been declared in default.
You may get a compile error (CE) if you just simply copy and paste the code.