CHUYÊN ĐỀ
XỬ LÝ XÂU NÂNG CAO

MỤC LỤC

PHẦN I: MỞ ĐẦU

Trong các kì thi học sinh giỏi, việc sử dụng cấu trúc dữ liệu xâu thường xuyên xuất hiện. Có nhiều các thuật toán khác nhau để xử lí xâu. Trong chuyên đề này, tôi xin được trình bày về kĩ thuật Hashing.

Trong quá trình thực hiện chuyên đề, do giới hạn về thời gian và hạn chế về kinh nghiệm công tác nên không thể tránh được những thiếu sót. Kính mong các thầy cô và các em học sinh góp ý để tôi có thể hoàn thiện tốt hơn nội dung cho chuyên đề.

PHẦN II: NỘI DUNG

I. LÝ THUYẾT

Để giải quyết vấn đề so sánh hai chuỗi một cách hiệu quả, ta có cách so sánh đơn giản nhất là duyệt lần lượt từng kí tự ở cả hai chuỗi để so sánh. Với cách này, độ phức tạp là $O(n_1, n_2)$ với $n_1, n2$ là độ dài tương ứng của xâu 1 và xâu 2.

Ta có thể làm tốt hơn với ý tưởng như sau: chuyển đổi mỗi xâu tương ứng thành một số nguyên. Sau khi chuyển đổi việc so sánh hai xâu chỉ còn độ phức tạp là $O(1)$.

Để chuyển đổi xâu thành số nguyên, ta cần có hàm băm Hash function. Sau khi chuyển đổi, nếu hai xâu s và t thì ta có $hash(s) = hash(t)$.

Lưu ý: điều kiện trên không nhất thiết phải đảm bảo theo hướng ngược lại. Tức là nếu $hash(s) = hash(t)$ thì không nhất thiết là s phải bằng t.

Hàm băm:

Một hàm được dùng để xác định giá trị băm của chuỗi thường được sử dụng là:

$$hash(s) = (s[0] + s[1] \times P^1 + s[2] \times P^2 \ldots + S[n-1] \times P^{n-1}) \; mod \; m$$

Với P và m là các số nguyên được chọn trước. Số P thường được chọn là một số nguyên tố gần bằng số lượng kí tự trong dữ liệu đầu vào. Ví dụ nếu trong xâu đầu vào chỉ gồm các kí tự in thường trong bảng chữ cái tiếng Anh thì $P = 31$ là một lựa chọn tốt. Nếu trong xâu đầu vào bao gồm cả chữ cái viết thường lẫn viết hoa trong bảng chữ cái tiếng Anh thì $P = 53$ là một lựa chọn tốt.

Rõ ràng, m phải là một số lớn vì xác xuất để hai xâu ngẫu nhiên khác nhau có mã $hash$ trùng nhau (gọi là va chạm) là $\approx \frac{1}{m}$. Trong thực tế, ta thường chọn m là một số nguyên tố lớn. Trong chuyên đề này tôi sử dụng $m = 10^9 + 9$. Đây là một số nguyên lớn nhưng đủ nhỏ để thực hiện phép nhân hai giá trị bằng cách sử dụng số nguyên 64 bit.

Dưới đây là một ví dụ về hàm băm của một chuỗi S chỉ gồm các chữ cái in thường. Ta chuyển đổi từng kí tự của S thành một số nguyên.

a	1
b	2
c	3
d	4
e	5
f	6
g	7
h	8

i	9
j	10
k	11
l	12
m	13
n	14
o	15
p	16
q	17
r	18
s	19
t	20
u	21
v	22
w	23
x	24
y	25
z	26

Không lựa chọn chuyển đổi $a \to 0$ vì chuỗi dạng $aaaaaa$ ... sẽ có mã hash là 0.

```
long long compute_hash(string const& s) {
    const int p = 31;
    const int m = 1e9 + 9;
    long long hash_value = 0;
    long long p_pow = 1;
    for (char c : s) {
        hash_value = (hash_value + (c - 'a' + 1) * p_pow) % m;
        p_pow = (p_pow * p) % m;
    }
    return hash_value;
}
```

II. BÀI TẬP ÁP DỤNG.

Bài tập 1:

Cho n xâu $S_1, S_2, \ldots, S_n$ mỗi xâu có độ dài không quá m kí tự. Hãy tìm tất cả các xâu trùng nhau trong chuỗi và chia chúng vào các nhóm.

Ta có thể thực hiện sắp xếp các xâu rồi lần lượt so sánh các xâu liền kề với độ phức tạp là $O(n\,m \log n)$. Để phân loại cần thực hiện n log (n) phép so sánh với mỗi so sánh mất $O(m)$. Tuy nhiên, bằng cách sử dụng hàm băm, mỗi phép so sánh ta chỉ mất thời gian thực hiện là $O(1)$ và độ phức tạp giảm xuống còn $O(nm + n \log n)$

Ta sẽ thực hiện hàm băm cho từng chuỗi, sắp xếp các giá trị băm kèm theo chỉ số sau đó nhóm các chỉ số có hàm băm giống nhau.

```
vector<vector<int>> group_identical_strings(vector<string> const& s)
{
    int n = s.size();
    vector<pair<long long, int>> hashes(n);
    for (int i = 0; i < n; i++)
        hashes[i] = {compute_hash(s[i]), i};
```

```
    sort(hashes.begin(), hashes.end());

    vector<vector<int>> groups;
    for (int i = 0; i < n; i++) {
        if (i == 0 || hashes[i].first != hashes[i-1].first)
            groups.emplace_back();
        groups.back().push_back(hashes[i].second);
    }
    return groups;
}
```

Bài tập 2:

Tính toán nhanh hàm băm của một xâu con trong xâu cho trước.

Vấn đề: cho xâu S và cặp chỉ số i, j. Hãy tính hàm băm của xâu con $S[i \dots j]$.

Theo định nghĩa, ta có

$$hash(S[i \dots j]) = \sum_{k=i}^{j} s[k] \times p^{k-i} \ mod\ m$$

Nhân với P^i:

$$hash(S[i \dots j]) \times p^i = \sum_{k=i}^{j} s[k] \times p^k \ mod\ m$$

$$= \ hash(s[0 \dots j]) - hash(s[0, i-1])\ mod\ m$$

Bài tập 3:

Cho một xâu S và một xâu mẫu P. Xác định tất cả các vị trí xẫu hiện của xâu mẫu P trong xâu S.

Input:

Gồm một số thử nghiệm, mỗi thử nghiệm gồm 3 dòng

Dòng 1: ghi độ dài của xâu mẫu P;

Dòng 2: ghi xâu mẫu P;

Dòng 3: ghi xâu S.

Output:

Đối với mỗi thử nghiệm, ghi tất cả các vị trí xuất hiện của xâu P trong xâu S. (vị trí của xâu S được đánh chỉ số bắt đầu từ 0). Giữa mỗi thử nghiệm ngăn cách nhau bởi một kí tự xuống dòng.

Example:

INPUT	OUTPUT
2 dh dhbbdhbb	0 4

Code:

```
#include <bits/stdc++.h>
#define pb push_back
#define MAX 1000006
#define mod 1000000009
#define read freopen("input.txt","r",stdin);
#define base 10
using namespace std;
typedef long long ll;
typedef unsigned long long llu;

llu ary[MAX],ah,bh[MAX];

int main()
{
    ary[0]=1;
    for(int i=1; i<=MAX; i++) ary[i]=ary[i-1]*base;
    string a,b;
    int lena,lenb;
    while(cin>>lena>>a>>b)
    {
        lenb=b.size();
        ah=0;
        memset(bh,0,sizeof(bh));
        for(int i=lena-1; i>=0; i--) ah=ah*base+a[i];
        for(int i=lenb-1; i>=0; i--) bh[i]=bh[i+1]*base+b[i];
        bool flag=false;
        if(lena<=lenb)
        {
            for(int i=0; i<=lenb-lena; i++)
            {
                llu c=bh[i]-(bh[i+lena]*ary[lena]);
                if(ah==c )
                {
                    printf("%d\n",i);
                    flag=true;
                }
            }
        }
        if(!flag) puts("");
    }
    return 0;
}
```

Bài tập 4:

Cho một xâu S gồm không quá 10^6 kí tự. Một xâu con *P* của *S* thỏa mãn các điều kiện sau được gọi là xâu đặc biệt:

- P là tiền tố của S;
- P là hậu tố của S;
- P xuất hiện ở giữa xâu S.

Tức là xâu P xuất hiện ít nhất 3 lần trong xâu S ở các vị trí tiền tố, hậu tố và giữa xâu S.

Input:

- Một dòng duy nhất ghi xâu S có độ dài nằm trong khoảng từ 1 đến 10^6.

Output:

- In ra xâu P. Nếu không tìm được xâu như vậy, in ra "None".

Example:

INPUT	OUTPUT
fixprefixsuffix	fix

Code:

```
#include<iostream>
#include<algorithm>
#include<cstdio>
#include<cstring>
#include<vector>
#define REP(i,m) for(int i=0;i<m;++i)
#define REPN(i,m,in) for(int i=in;i<m;++i)
#define ALL(t) (t).begin(),(t).end()
#define pb push_back
#define mp make_pair
#define fr first
#define sc second
#define dump(x)  cerr << #x << " = " << (x) << endl
#define prl cerr<<"called:"<< __LINE__<<endl
using namespace std;
static const int INF =500000000;
template<class T> void debug(T a,T b){ for(;a!=b;++a) cerr<<*a<<'
';cerr<<endl;}
typedef long long int lint;
typedef pair<int,int> pi;
char s[1000005],key[1000005];
int n;
int table[1000005];
int maxlen;
void check(){
   memcpy(key,s,sizeof(s));

   table[0]=-1;
```

```
    REPN(i,n+1,1){
        int j=table[i-1];
        while(j>=0 && key[j+1]!=key[i]) j=table[j];
        if(key[j+1]==key[i]) ++j;
        table[i]=j;
    }

    int j=-1;
    REPN(i,n-1,1){
        while(j>=0 && key[j+1]!=s[i]) j=table[j];
        if(key[j+1]==s[i]) ++j;
        maxlen=max(maxlen,j+1);
    }
}

lint hash[1000005],hash2[1000005];
lint B=100000007;

int main(){
    scanf("%s",s);
    n=strlen(s);

    hash[0]=s[0];
    REP(i,n) hash[i+1]=hash[i]*B+s[i+1];

    hash2[n]=0;
    lint Bsum=1;
    for(int i=n-1;i>=0;--i) hash2[i]=hash2[i+1]+s[i]*Bsum,Bsum*=B;




    check();

    int lb=0;
    for(int i=maxlen;i>=1;--i){
        if(hash[i-1]==hash2[n-i]){
            lb=i;
            break;
        }
    }

    if(lb==0) puts("Just a legend");
    else{
        s[lb]='\0';
        printf("%s\n",s);
    }

    return 0;
}
```

Bài tập 5:

Cho xâu S. Hãy tìm xâu đối xứng dài nhất xuất hiện trong S.

Input:

- Dòng 1 ghi số nguyên N ($N \leq 50000$) là số kí tự của xâu S;

- Dòng 2 ghi xâu S.

Output:

- Một dòng duy nhất ghi độ dài của xâu đối xứng dài nhất trong xâu S.

Example:

INPUT	OUTPUT
5 abacd	3

Code:

```
#include <cstring>
#include <cstdio>
#include <iostream>
#define REP(i, a, b) for(int i = (a); i <=(b); ++i)
#define REPD(i, a, b) for(int i = (a); i >=(b); --i)
#define long long long

const int N = 50005;
const int MOD = 1000000007;
const long MM = (long)MOD * MOD;

using namespace std;

long H[N], R[N], power[N];
char s[N];
int n;

int getHash(int i, int j)
    { return (H[j] - H[i - 1] * power[j - i + 1] + MM) % MOD; }
int getRash(int i, int j)
    { return (R[i] - R[j + 1] * power[j - i + 1] + MM) % MOD; }
bool isPalin(int i, int j)
    { return getHash(i, j) == getRash(i, j); }

int main() {
    #ifdef _LAD_
        freopen("PALINY.in", "r", stdin);
    #endif // _LAD_
```

```
    scanf("%d\n%s", &n, s + 1);
    REP(i, 1, n) H[i] = (H[i - 1] * 26 + s[i] - 'a') % MOD;
    REPD(i, n, 1) R[i] = (R[i + 1] * 26 + s[i] - 'a') % MOD;
    power[0] = 1;
    REP(i, 1, n) power[i] = power[i - 1] * 26 % MOD;
    int ans = 0;
    REP(i, 1, n) {
        // even palindrome
        int l = 0, r = min(i, n - i);
        while (l <= r) {
            int mid = l + r >> 1;
            if (isPalin(i - mid + 1, i + mid))
                ans = max(ans, mid << 1), l = mid + 1;
            else
                r = mid - 1;
        }
        // odd palindrome
        l = 0, r = min(i - 1, n - i);
        while (l <= r) {
            int mid = l + r >> 1;
            if (isPalin(i - mid, i + mid))
                ans = max(ans, mid << 1 | 1), l = mid + 1;
            else
                r = mid - 1;
        }
    }
    printf("%d\n", ans);
    return 0;
}
```

Bài tập 6:

Cho một xâu S không quá 50000 kí tự và một số nguyên K. Hãy cho biết độ dài của xâu dài nhất mà xuất hiện ít nhất K lần trong xâu S.

Input:

- Dòng thứ nhất gồm hai số nguyên N và K ($1 \leq N \leq 50\,000; 1 \leq K \leq 20$);
- Dòng thứ hai ghi xâu S gồm N kí tự chữ cái in thường.

Output:

- Gồm một dòng duy nhất là độ dài của xâu thỏa mãn yêu cầu bài toán.

Example:

INPUT	OUTPUT
5 2 xxxxx	4

Code:

```
#include <bits/stdc++.h>
using namespace std;

typedef long long ll;
typedef pair<int,int> ii;
typedef unsigned long long ull;

#define X first
#define Y second
#define pb push_back
#define mp make_pair
#define ep emplace_back
#define EL printf("\n")
#define sz(A) (int) A.size()
#define FOR(i,l,r) for (int i=l;i<=r;i++)
#define FOD(i,r,l) for (int i=r;i>=l;i--)
#define fillchar(a,x) memset(a, x, sizeof (a))
#define faster ios_base::sync_with_stdio(false); cin.tie(NULL); cout.tie(NULL);


const int N = 1e5+5, base = 1e6+9;
int n, k, st[base];
ll p[N], H[N];
string s;

ll getH(int i, int j) {
   if (i == 0) return H[j];
   ll ans = (H[j]-H[i-1]*p[j-i+1]%base)%base;
   while (ans < 0) ans += base;
   return ans;
}
```

```
bool check(int len) {
	FOR(i,0,base-1) st[i] = 0;
	FOR(i,0,n-1-len+1) {
		ll code = getH(i,i+len-1);
		if (++st[code] >= k) return true;
	}
	return false;
}


int main() {

	scanf("%d%d\n", &n,&k);
	getline(cin, s);

	p[0] = 1;
	FOR(i,1,n) p[i] = p[i-1] * 27 % base;

	H[0] = s[0]-'a'+1;
	FOR(i,1,n-1)
	H[i] = (H[i-1]*27 + s[i]-'a'+1) % base;

	int ans = 0;

	int L = 1, R = n;
	while (L <= R) {
		int mid = (L+R)/2;
		if (check(mid))
			ans = mid, L = mid+1;
		else
			R = mid-1;
	}

	cout << ans << endl;

	return 0;
}
```

Bài tập 7:

Cho hai xâu kí tự A và B chỉ gồm các kí tự chữ cái latin thường và một danh sách gồm Q truy vấn dạng l, r, u, v với ý nghĩa cần so sánh thứ tự từ điền của xâu con $A[i \dots j]$ với xâu con $B[u \dots v]$ (các kí tự của xâu được đánh thứ tự từ 1).
Với mỗi truy vấn hãy trả lời xem hai xâu đó có thứ tự từ điển như thế nào?
Lưu ý: xâu $A = A_1A_2 \dots A_n$ được gọi là có thứ tự từ điển nhỏ hơn xâu $B = B_1B_2 \dots B_m$ nếu:

+ $n < m$ và $A_i = B_i$ $(\forall i = 1 \to n)$;

+ Với k $(1 \le k \le \min(n, n))$ là giá trị nhỏ nhất thỏa mãn $A_k \ne B_k$ thì $A_k < B_k$

Hai xâu được gọi là bằng nhau nếu không xác định được xâu nào nhỏ hơn xâu nào.

Input:

- Dòng thứ nhất gồm hai số nguyên dương n, m $(1 \le n, m \le 10^6)$ là độ dài của xâu A và độ dài của xâu B;
- Dòng thứ 2 ghi xâu A;
- Dòng thứ 3 ghi xâu B;
- Dòng thứ 5 ghi Q là số lượng truy vấn;
- Q dòng tiếp theo, mỗi dòng ghi 4 số nguyên l, r, u, v $(1 \le l \le r \le n; 1 \le u \le v \le m)$ mô tả một truy vấn cần trả lời.

Output:

- Với mỗi truy vấn in ra một kí tự '>', '>', '=' tương ứng với câu trả lời. Tất cả các câu trả lời được ghi trên một dòng.

Example:

INPUT	OUTPUT
13 14 bomthichdacau bomthichdabanh 3 1 10 1 10 1 10 1 11 1 11 1 11	=<>

Code:

```
#include <iostream>
#include <string>
#include <vector>
using namespace std;


uint64_t pow[1000000];
```

```cpp
vector<uint64_t> make_hash(string &a) {
    vector<uint64_t> h(a.size());
    h[0] = a[0] - 'a';
    for (int i=1; i<(int)a.size(); i++) {
        h[i] = h[i-1] * 29 + (a[i]-'a');
    }
    return h;
}

uint64_t get_hash(const vector<uint64_t> &h, int l, int r) {
    if (l == 0) return h[r];
    return h[r] - h[l-1] * pow[r - l + 1];
}

int main() {
    pow[0] = 1;
    for (int i=1; i<1000000; i++) pow[i] = pow[i-1] * 29;

    ios::sync_with_stdio(false); cin.tie(0);
    int m, n; cin >> m >> n;
    string a, b; cin >> a >> b;
    auto ha = make_hash(a), hb = make_hash(b);
    int Q; cin >> Q;
    while (Q--) {
        int la, ra, lb, rb;
        cin >> la >> ra >> lb >> rb;
        la--, ra--, lb--, rb--;
        int len = min(ra-la+1, rb-lb+1);

        int low = 1, high = len;
        while (low <= high) {
            int k = (low + high) / 2;
            if (get_hash(ha, la, la+k-1) == get_hash(hb, lb, lb+k-1)) {
                low = k + 1;
            } else {
                high = k - 1;
            }
        }
```

```
        if (high == len) {
            if (ra-la == rb-lb) cout << '=';
            else if (ra-la < rb-lb) cout << '<';
            else cout << '>';
        } else {
            cout << (a[la+high]<b[lb+high]? '<':'>');
        }
    }
    return 0;
}
```

Bài tập 8:

Một biểu thức dạng $A + B = C$ đã được viết ra nhưng hiện tại nó đã bị xóa mất dấu $+$ và dấu $=$. Nhiệm vụ của bạn là phải khôi phục lại biểu thức đó.

Cho xâu ban đầu gồm các kí tự chữ số từ '0' đến '9'. Hãy chèn vào xâu đó một dấu '+' và một dấu '=' sao cho:

- Kí tự '+' được đặt bên trái kí tự '=';
- Các kí tự '+' và '=' chia dãy ra thành ba thành phần tương ứng là A, B và C;
- Cả ba phần A, B và C đều không chứa các số 0 vô nghĩa ở đầu;
- $A + B = C$

Dữ liệu đầu vào đảm bảo luôn có câu trả lời tồn tại.

Input:

- Gồm một xâu duy nhất không rỗng, chứa các kí tự chữ số. Độ dài của xâu không quá 10^6 kí tự.

Output:

- In ra xâu sau khi khôi phục thành biểu thức. Nếu có nhiều hơn một cách, in ra một cách bất kì.

Example:

INPUT	OUTPUT
12345168	123 + 45 = 168

Code:

```
#include<bits/stdc++.h>

using namespace std;
const int maxn = 1e6+9;
const int HM = 2;
typedef long long ll;
int Hash[maxn][HM],mod[1000],rt;
int power[maxn][HM];
```

```
char que[maxn];
void getprime(int l,int r,int P[]) {
    int cnt = 0;
    for(int i = l; i <= r; ++i) {
        int flag = 0;
        for(int j = 2; j*j <= i; ++j)
            if(i%j==0)flag = 1;
        if(flag==0)P[cnt++] = i;
    }
}
bool cheak(int l1,int l2,int l3,int n) {
    if(l1<=0||l2<=0||l3<=0)return 0;
    if(l2>l1||l3>l1)return 0;
    int cnt1 = 0,cnt2 = 0,cnt3 = 0;
    if(l1!=1&&que[l3+l2+1]=='0')return 0;
    if(l2!=1&&que[l3+1]=='0')return 0;
    if(l3!=1&&que[1]=='0')return 0;
    for(int i = 0; i < HM; ++i) {
        int tp1,tp2,tp3;
        tp1 = tp2 = tp3 = 0;
        tp1 = Hash[n][i] - Hash[n-l1][i]*power[l1][i]%mod[i];
        tp2 = Hash[n-l1][i] - Hash[l3][i]*power[l2][i]%mod[i];
        tp3 = Hash[l3][i];
        if((tp2+tp3-tp1)%mod[i]!=0)return 0;
    }
    for(int i = 1; i<=l3; ++i)putchar(que[i]);
    putchar('+');
    for(int i = l3+1; i<=l2+l3; ++i)putchar(que[i]);
    putchar('=');
    for(int i = l3+l2+1; i<=n; ++i)putchar(que[i]);
    putchar('\n');
    return 1;
}
int main() {
    memset(Hash,0,sizeof Hash);
    getprime(10007,12005,mod);
    for(int i = 0;i<maxn;++i)
        for(int j = 0;j<HM;++j)power[i][j] = 1;
    rt = 10;
```

```
    scanf("%s",que+1);
    int len = strlen(que+1);
    for(int i = 1; i <= len; ++i) {
        for(int j = 0; j<HM; ++j)
            Hash[i][j] = (Hash[i-1][j]*rt%mod[j] + que[i]-'0')%mod[j],
            power[i][j] = power[i-1][j]*rt%mod[j];
    }
    for(int i = len; i >= 1&&i>=len/4; --i) {
        if(cheak(len-i+1,len-i+1,len-2*(len-i+1),len))return 0;
        if(cheak(len-i+1,len-i,len-2*(len-i)-1,len))return 0;
        if(cheak(len-i+1,len-2*(len-i+1),len-i+1,len))return 0;
        if(cheak(len-i+1,len-2*(len-i)-1,len-i,len))return 0;
    }
    return 0;
}
```

Bài tập 9:

Cho một 'bản đồ' thứ nhất là một lưới có kích thước $N \times M$ mà mỗi phần tử trong lưới là một kí tự chữ cái in thường; và một bản đồ thứ 2 là một lưới có kích thước $M \times N$ tương tự. Bạn cần phải căn chỉnh sao cho hai bản đồ này chồng lên nhau trên một phần có kích thước $M \times M$ sao cho các phần tử trong phần này giống hệt nhau. Hãy xác định vị trí của phần chồng lên nhau này trên hai bản đồ đã cho.

Input:

- Dòng thứ nhất chứa hai số nguyên N, M $(1 \leq N \leq 2000; 1 \leq M \leq 200; M \leq N)$;
- N dòng tiếp theo, mỗi dòng chứa M kí tự latin viết thường biểu diễn bản đồ thứ nhất;
- M dòng tiếp theo, mỗi dòng chứa N kí tự latin viết thường biểu diễn bản đồ thứ 2.

Output:

- Gồm hai số nguyên i, j ngăn cách bởi một dấu cách biểu thị phần có kích thước $M \times M$ trong bản đồ đầu tiên bắt đầu từ hàng thứ i bằng với phần của bản đồ thứ hai bắt đầu từ dòng thứ j. Các hàng và cột được đánh thứ tự bắt đầu từ 1. Nếu có nhiều hơn một đáp án, bạn chỉ cần in ra một đáp án bất kì. Dữ liệu đảm bảo có ít nhất một đáp án tồn tại.

Example:

INPUT	OUTPUT
10 5 somer andom noise mayth eforc ebewi thyou hctwo again noise somermayth andomeforc noiseebewi againthyou noisehctwo	4 6

Code:

```
#ifdef DEBUG
#define _GLIBCXX_DEBUG
#endif

#include <bits/stdc++.h>

using namespace std;

typedef long double ld;

#ifdef DEBUG
#define eprintf(...) fprintf(stderr, __VA_ARGS__), fflush(stderr)
#else
#define eprintf(...) ;
#endif
```

```
#define sz(x) ((int) (x).size())
#define TASK "text"

const int inf = (int) 1.01e9;
const long long infll = (long long) 1.01e18;
const ld eps = 1e-9;
const ld pi = acos((ld) -1);

mt19937 mrand(random_device{} ());

int rnd(int x) {
  return mrand() % x;
}

const int mod[2] = {(int) 1e9 + 7, (int) 1e9 + 9};

struct Hash {
  static const int n = 2;
  int a[n];

  Hash() {
    for (int i = 0; i < n; i++) {
      a[i] = 0;
    }
  }

  Hash(int x) {
    for (int i = 0; i < n; i++) {
      a[i] = x % mod[i];
    }
  }

  Hash operator + (const Hash &h) const {
    Hash res;
    for (int i = 0; i < n; i++) {
      res.a[i] = a[i] + h.a[i];
      if (res.a[i] >= mod[i]) {
        res.a[i] -= mod[i];
      }
```

```
    }
    return res;
  }

  Hash operator - (const Hash &h) const {
    Hash res;
    for (int i = 0; i < n; i++) {
      res.a[i] = a[i] + mod[i] - h.a[i];
      if (res.a[i] >= mod[i]) {
        res.a[i] -= mod[i];
      }
    }
    return res;
  }

  Hash operator * (const Hash &h) const {
    Hash res;
    for (int i = 0; i < n; i++) {
      res.a[i] = (long long) a[i] * h.a[i] % mod[i];
    }
    return res;
  }

  bool operator == (const Hash &h) const {
    for (int i = 0; i < n; i++) {
      if (a[i] != h.a[i]) {
        return false;
      }
    }
    return true;
  }
};

const int maxn = 4005;
Hash p[maxn];

void precalc() {
  p[0] = Hash(1);
  p[1].a[0] = rnd(500) + 500;
```

```
  p[1].a[1] = rnd(500) + 500;
  for (int i = 2; i < maxn; i++) {
    p[i] = p[i - 1] * p[1];
  }
}

int n, m;
char a[maxn][maxn], b[maxn][maxn];

int read() {
  if (scanf("%d%d", &n, &m) < 2) {
    return false;
  }
  for (int i = 0; i < n; i++) {
    scanf("%s", a[i]);
  }
  for (int i = 0; i < m; i++) {
    scanf("%s", b[i]);
  }
  return true;
}

Hash ha[maxn], hb[maxn];
Hash s[maxn];
int z[maxn];

bool check(int row) {
  for (int i = 0; i < m; i++) {
    s[i] = ha[i];
  }
  for (int i = 0; i < n; i++) {
    s[m + i] = hb[i];
  }
  for (int i = 0; i < n + m; i++) {
    z[i] = 0;
  }
  for (int i = 1, l = 0, r = -1; i < n + m; i++) {
    if (i + z[i - l] < r) {
      z[i] = z[i - l];
```

```
    } else {
      l = i;
      r = max(r, i);
      while (r < n + m && s[r - l] == s[r]) {
        r++;
      }
      z[i] = r - l;
    }
  }
  for (int i = 0; i < n; i++) {
    if (z[m + i] >= m) {
      printf("%d %d\n", row + 1, i + 1);
      return true;
    }
  }
  return false;
}

void solve() {
  for (int i = 0; i < n; i++) {
    auto &cur = hb[i];
    cur = Hash();
    for (int j = 0; j < m; j++) {
      cur = cur * p[1] + Hash(b[j][i]);
    }
  }
  for (int i = 0; i < m; i++) {
    auto &cur = ha[i];
    cur = Hash();
    for (int j = 0; j < m; j++) {
      cur = cur * p[1] + Hash(a[j][i]);
    }
  }
  if (check(0)) {
    return;
  }
  for (int i = 0; i + m < n; i++) {
    for (int j = 0; j < m; j++) {
      ha[j] = ha[j] - Hash(a[i][j]) * p[m - 1];
```

```
    }
    for (int j = 0; j < m; j++) {
      ha[j] = ha[j] * p[1] + Hash(a[i + m][j]);
    }
    if (check(i + 1)) {
      return;
    }
  }
}

int main() {
  precalc();
  while (read()) {
    solve();
  }
  return 0;
}
```

Bài tập 10:

Cho hai xâu s và t chỉ gồm các chữ cái Latin viết thường. Giả sử S là tập các kí tự phân biệt của s còn T là tập hợp các kí tự phân biệt của t. Hai xâu s và t được gọi là đẳng cấu nếu có một ánh xạ một – một f giữa S và T mà $f(s_i) = t_i$. Tức là phải thỏa mãn:

1. $f(s_i) = t_i$ với bất kì chỉ số nào;
2. Với mỗi kí tự $x \in S$ có đúng một kí tự $y \in T$ sao cho $f(x) = y$;
3. Đối với mỗi kí tự $y \in T$ có đúng một kí tự $x \in S$ mà $f(x) = y$.

Ví dụ: cặp xâu "aababc" và "bbcbcz" là đẳng cấu nhưng cặp xâu "test" và "best" không phải là đẳng cấu.

Cho xâu S gồm n chữ cái Latin viết thường. Bạn phải trả lời Q quy vấn, mỗi có dạng x, y, len $(1 \le x, y \le n - len + 1)$. Đối với mỗi truy vấn, hãy kiểm tra xem hai chuỗi con $S[x \dots x + len - 1]$ và $S[y \dots y + len - 1]$ có đẳng cấu hay không.

Input:

- Dòng đầu tiên chứa hai số nguyên $n, Q (1 \le n \le 2 \times 10^5; 1 \le m \le 2 \times 10^5)$ là độ dài của xâu S và số lượng truy vấn;
- Dòng thứ hai chứa xâu S gồm n kí tự Latin viết thường;
- Q dòng tiếp theo mỗi dòng là một truy vấn gồm 3 số nguyên x_i, y_i và len_i $(1 \le x_i, y_i \le n, 1 \le len_i \le n - max(x_i, y_i) + 1)$ mô tả về cặp chuỗi con cần kiểm tra.

Output:

- Đỗi với mỗi truy vấn, in trên một dòng chữ "YES" nếu là đẳng cấu. Ngược lại, ghi "NO".

Example:

INPUT	OUTPUT
7 4 abacaba 1 1 1 1 4 2 2 1 3 2 4 3	YES YES NO YES

Code:

```
#include <bits/stdc++.h>

#define fore(i, l, r) for(int i = int(l); i < int(r); i++)
#define sz(a) int(a.size())

#define x first
#define y second

using namespace std;

const int B = 2;

typedef array<int, B> ht;

ht MOD, BASE;

inline int norm(int a, const int &MOD) {
	while(a >= MOD)
		a -= MOD;
	while(a < 0)
		a += MOD;
	return a;
}

inline int mul(int a, int b, const int &MOD) {
	return int(a * 1ll * b % MOD);
}
```

```
inline ht operator +(const ht &a, const ht &b) {
	ht ans;
	fore(i, 0, sz(ans))
		ans[i] = norm(a[i] + b[i], MOD[i]);
	return ans;
}

inline ht operator -(const ht &a, const ht &b) {
	ht ans;
	fore(i, 0, sz(ans))
		ans[i] = norm(a[i] - b[i], MOD[i]);
	return ans;
}

inline ht operator *(const ht &a, const ht &b) {
	ht ans;
	fore(i, 0, sz(ans))
		ans[i] = mul(a[i], b[i], MOD[i]);
	return ans;
}

int CMODS[] = {int(1e9) + 7, int(1e9) + 9, int(1e9) + 21, int(1e9) + 33,
int(1e9) + 87, int(1e9) + 93, int(1e9) + 97, int(1e9) + 103};
int CBASE[] = {1009, 1013, 1019, 1021};

const int N = 200 * 1000 + 555;

int n, m;
char s[N];
int x[N], y[N], len[N];

inline bool read() {
	if(!(cin >> n >> m))
		return false;
	assert(scanf("%s", s) == 1);

	fore(i, 0, m) {
		assert(scanf("%d%d%d", &x[i], &y[i], &len[i]) == 3);
```

```
                x[i]--, y[i]--;
        }
        return true;
}
void setMods() {
        mt19937 rnd;
        unsigned int seed = n;
        fore(i, 0, n)
                seed = (seed * 3) + s[i];
        fore(i, 0, m) {
                seed = (seed * 3) + x[i];
                seed = (seed * 3) + y[i];
                seed = (seed * 3) + len[i];
        }
        rnd.seed(seed);

        set<int> mids;
        while(sz(mids) < sz(MOD))
                mids.insert(rnd() % 8);
        vector<int> vmids(mids.begin(), mids.end());
        fore(i, 0, sz(MOD)) {
                MOD[i] = CMODS[vmids[i]];
                BASE[i] = CBASE[i];
        }
}

ht pBase[N];
ht ph[27][N];
vector<int> ord[N];

ht getHash(int id, int l, int r) {
        return ph[id][r] - ph[id][l] * pBase[r - l];
}

inline void solve() {
        setMods();

        pBase[0] = {1, 1};
        fore(i, 1, N)
```

```
                pBase[i] = pBase[i - 1] * BASE;

        fore(c, 0, 26) {
                ph[c][0] = {0, 0};
                fore(i, 0, n) {
                        int val = (s[i] == c + 'a');
                        ph[c][i + 1] = ph[c][i] * BASE + ht{val, val};
                }
        }

        vector<int> curOrd(26, 0);
        iota(curOrd.begin(), curOrd.end(), 0);
        for(int i = n - 1; i >= 0; i--) {
                ord[i] = curOrd;
                auto it = find(ord[i].begin(), ord[i].end(), int(s[i] -
'a'));
                ord[i].erase(it);
                ord[i].insert(ord[i].begin(), int(s[i] - 'a'));
                curOrd = ord[i];
        }

        fore(q, 0, m) {
                int s1 = x[q], s2 = y[q];
                bool ok = true;

                fore(i, 0, 26) {
                        if(getHash(ord[s1][i], s1, s1 + len[q]) !=
                                getHash(ord[s2][i], s2, s2 + len[q])) {
                                ok = false;
                                break;
                        }
                }
                puts(ok ? "YES" : "NO");
        }
}

int main(){
#ifdef _DEBUG
        freopen("input.txt", "r", stdin);
```

```
	//freopen("output.txt", "w", stdout);
#endif
	if(read()) {
		solve();
	}
	return 0;
}
```

PHẦN III: KẾT LUẬN

Trong chuyên đề, tôi đã trình bày về kĩ thuật Hashing để giải quyết một số bài toán về xâu. Đây là một kĩ thuật giúp chúng ta linh động hơn trong việc xử lí các vấn đề về xâu.

Do thời gian và kiến thức có hạn nên chuyên đề này khó tránh khỏi những thiếu sót. Rất mong quý thầy cô đồng nghiệp đóng góp ý kiến để chuyên đề được hoàn thiện hơn. Tôi xin chân thành cảm ơn.

TÀI LIỆU THAM KHẢO

1. Introduction to Algorithms Third Edition – (Thomas H. Cormen; Charles E. Leiserson; Ronald L. Rivest; Clifford Stein)
2. https://vi.wikipedia.org/
3. https://www.spoj.com/
4. https://vn.spoj.com/
5. http://codeforces.com/