isekai

by Phero

ในข้อนี้จะใช้ Rabin-Karp Algorithm ในการแก้ไขโจทย์ โดยก่อนอื่นเราจะทำการ Preprocess เพื่อเก็บค่า hash ของ prefix และ suffix ทุกตำแหน่งของสตริง S

- โดยจะนิยามให้ prefix ในตำแหน่งที่ i ของสตริงเรียกว่า $l[i]$ และ suffix ในตำแหน่งที่ i ของสตริงเรียกว่า $r[i]$ (โดยที่ $1 \leq i \leq N$)

การที่จะตรวจสอบว่าสตริงในช่วง $[S_aS_{a+1}...\ S_{b-1}S_bS_cS_{c+1}...S_{d-1}S_d]$ เป็น**พาลินโดรม**หรือเปล่า เราจะรู้ค่าอยู่สองค่าคือค่า hash ที่ได้จากสตริงข้างบนตามปกติ และค่า hash ที่ได้จากการ reverse สตริงข้างบน

- นิยามให้เครื่องหมาย $-$ นี้หมายถึงทำการ**ลบตำแหน่ง**ของสตริงนั้นออก
- นิยามให้เครื่องหมาย $+$ หมายถึงทำการ**เชื่อม**ทั้งสองสตริงเข้าด้วยกัน

เราสามารถหาค่า hash ของสตริงตามปกติได้จากการเอา $(l[b] - l[a-1]) + (l[d] - l[c-1])$

ส่วนค่า hash ของสตริงที่ทำการ reverse สามารถหาได้จากการเอา $(r[c] - r[d+1]) + (r[a] - r[b+1])$

ดังนั้นถ้าค่า hash ทั้งสองค่ามีค่าเท่ากันก็แปลว่าสตริงดังกล่าวนั้นเป็นพาลินโดรมนั่นเอง

Model Code

```
#include <bits/stdc++.h>
using namespace std;
#define ll long long
#define N 2000005
#define MUL 101 // power value (must be a prime number)
#define MOD 1000000007 // mod value (must be a prime number)
int n,q,l1,r1,l2,r2;
ll l[N],r[N],pw[N]; // values of i_th prefix, i_th suffix, and mul^i
char s[N];

ll s2n(int p){ return (ll)s[p] - 'A' + 1; } // change char to int
ll add(ll a,ll b){ return (a + b) % MOD;} // plus function
ll sub(ll a,ll b){ return (a - b + MOD) % MOD;}  // minus function
ll mul(ll a,ll b){ return (a * b) % MOD;} // multiply function

bool is_parin(int a,int b,int c,int d){
    ll ab = sub(l[b], mul(l[a-1], pw[b-a+1])); // calcualte S[a:b]
    ll cd = sub(l[d], mul(l[c-1], pw[d-c+1])); // calcualte S[c:d]
    ll abcd = add(mul(ab, pw[d-c+1]), cd); // merge ab and cd

    ll ba = sub(r[a], mul(r[b+1], pw[b-a+1])); // calcualte S[b:a]
    ll dc = sub(r[c], mul(r[d+1], pw[d-c+1])); // calcualte S[d:c]
    ll dcba = add(mul(dc, pw[b-a+1]), ba); // merge dc and ba

    return abcd == dcba; // check if they equal
}

int main(){
    scanf("%d%d",&n,&q);
    scanf("%s",s+1);

    pw[0] = 1;
    for(int i=1;i<=n;i++) pw[i] = mul(pw[i-1], MUL);
    for(int i=1;i<=n;i++) l[i] = add(mul(l[i-1], MUL), s2n(i));
    for(int i=n;i>=1;i--) r[i] = add(mul(r[i+1], MUL), s2n(i));

    while(q--){
        scanf("%d%d%d%d",&l1,&r1,&l2,&r2);
        is_parin(l1,r1,l2,r2) ? printf("YES\n") : printf("NO\n");
    }
}
```

Time Complexity: $\mathcal{O}(N+Q)$