2016ACM香港网赛题解

PROBLEM A

A+B PROBLEM

Given N integers in the range [−50000,50000], how many ways are there to pick three integers ai, aj, ak, such that i, j, k are pairwise distinct and ai+aj=ak? Two ways are different if their ordered triple (i,j,k) of indices are different.

Input

The first line of input consists of a single integer N (1≤N≤200000). The next line consists of N space-separated integers a1,a2,…,aN.

Output

Output an integer representing the number of ways.

Sample Input 1

```
4
1 2 3 4
```

Sample Output 1

```
4
```

Sample Input 2

```
6
1 1 3 3 4 6
```

Sample Output 2

```
10
```

题意

从输入数据中找这样的组合（i，j，k），使ai+aj=ak。输出组数。应是FFT,看不懂。自己暴力的超时了...

AC代码

```cpp
#include <iostream>
#include <cmath>
#include <cstring>
#include <algorithm>
using namespace std;

const int N = 2e5+10;
const double pi = acos(-1.0);

char s1[N],s2[N];
int len,res[N];

struct Complex{
    double r,i;
    Complex(double r=0,double i=0):r(r),i(i) {

    };
    Complex operator+(const Complex &rhs){
        return Complex(r + rhs.r,i + rhs.i);
    }
    Complex operator-(const Complex &rhs){
        return Complex(r - rhs.r,i - rhs.i);
    }
    Complex operator*(const Complex &rhs){
        return Complex(r*rhs.r - i*rhs.i,i*rhs.r + r*rhs.i);
    }
} va[N],vb[N];

//雷德算法--倒位序
void rader(Complex F[],int len){ //len = 2^M,reverse F[i] with F[j] j为i二进制反转
    int j = len >> 1;
    for(int i = 1;i < len - 1;++i){
        if(i < j) swap(F[i],F[j]);  // reverse
        int k = len>>1;
        while(j>=k){
            j -= k;
            k >>= 1;
        }
        if(j < k) j += k;
    }
}
//FFT实现
void FFT(Complex F[],int len,int t){
    rader(F,len);
    for(int h=2;h<=len;h<<=1){ //分治后计算长度为h的DFT
        Complex wn(cos(-t*2*pi/h),sin(-t*2*pi/h)); //单位复根e^(2*PI/m)用欧拉公式展开
        for(int j=0;j<len;j+=h){
            Complex E(1,0); //旋转因子
            for(int k=j;k<j+h/2;++k){
                Complex u = F[k];
                Complex v = E*F[k+h/2];
                F[k] = u+v; //蝴蝶合并操作
                F[k+h/2] = u-v;
                E=E*wn; //更新旋转因子
            }
        }
    }
    if(t==-1){   //IDFT
        for(int i=0;i<len;++i){
            F[i].r/=len;
        }
    }
}
//数组开小会RE
long long a[4*N];
Complex F[4*N];
long long num[4*N],sum[4*N];
int n;
long long zero;

//求卷积
void Conv(Complex F[],int len){
    FFT(F,len,1);
    for(long long i=0;i<len;++i) F[i] = F[i]*F[i];
    FFT(F,len,-1);
}

void init(){
    memset(num,0,sizeof(num));
    cin >> n;
    zero=0;
    for(int i=0; i<n; i++){
        cin >> a[i];
        if(a[i]==0)zero++;
        num[a[i]+50000]++;
    }
    sort(a, a + n);
    int len1 = a[n-1] + 50000 + 1;
    len = 1;
    while(len < len1*2){
        len <<= 1;
    }
    for(int i=0; i<len1; i++){
        F[i] = Complex(num[i],0);
    }
    for(int i=len1; i<len; i++){
        F[i] = Complex(0,0);
    }
}

void gao(){
    Conv(F,len);
    //num数组就是卷积后的结果，表示两两组合
    for(int i=0; i<len; i++){
        num[i] = (long long)(F[i].r+0.5);
    }
    len = a[n-1]*2;
    //本身和本身组合是不行的,减掉取两个相同的组合
    for(int i=0; i<n; i++)
        num[a[i]+a[i]+2*50000]--;
    long long cnt = 0;
    for(int i=0; i<n; i++){
      if(a[i]!=0){
        cnt+=num[a[i]+2*50000];
        cnt-=zero*2;
      }else{
          cnt+=num[a[i]+2*50000];
          cnt-=(zero-1)*2;
      }
    }
    cout << cnt << endl;
}

int main(){
    int t;
    init();
    gao();
    return 0;
}
```

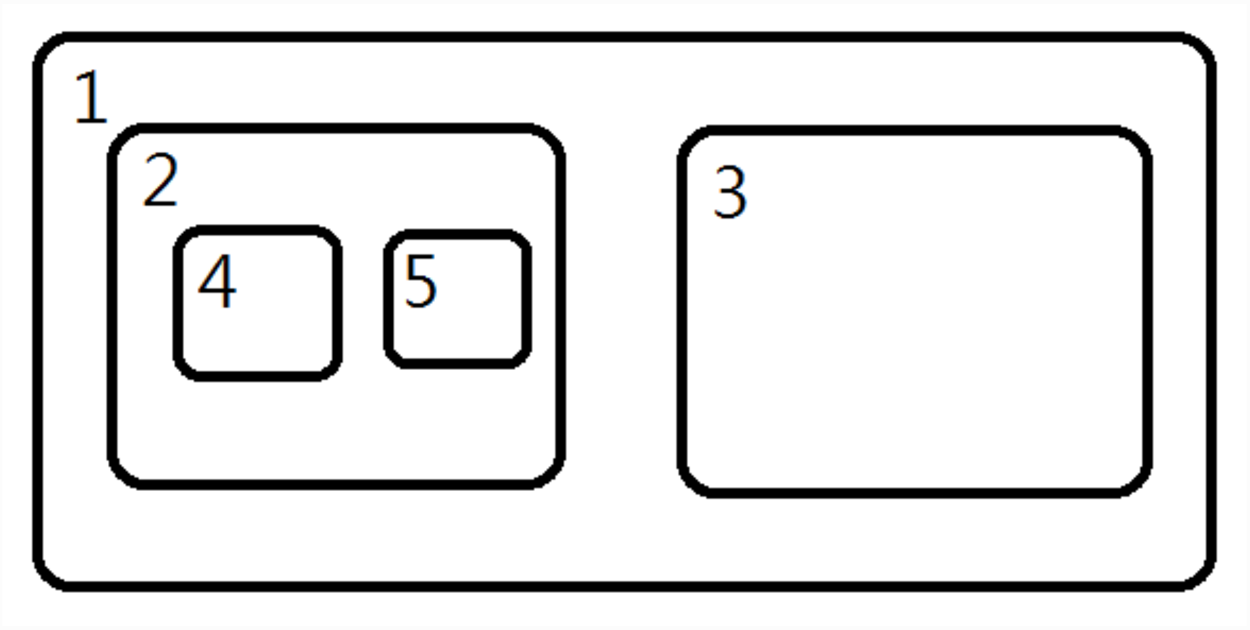
PROBLEM B

BOXES

There are N boxes, indexed by a number from 1 to N. Each box may (or not may not) be put into other boxes. These boxes together form a tree structure (or a forest structure, to be precise).

You have to answer a series of queries of the following form: given a list of indices of the boxes, find the total number of boxes that the list of boxes actually contain.

Consider, for example, the following five boxes.



- If the query is the list "1", then the correct answer is "5", because box 1 contains all boxes.
- If the query is the list "4 5", then the correct answer is "2", for boxes 4 and 5 contain themselves and nothing else.
- If the query is the list "3 4", then the correct answer is "2".
- If the query is the list "2 3 4", then the correct answer is "4", since box 2 also contains box 5.
- If the query is the list "2", then the correct answer is "3", because box 2 contains itself and two other boxes.

Input

The first line contains the integer N (1≤N≤200000), the number of boxes.

The second line contains Nintegers. The ith integer is either the index of the box which contains the ith box, or zero if the ith box is not contained in any other box.

The third line contains an integer Q (1≤Q≤100000), the number of queries. The following Q lines will have the following format: on each line, the first integer M(1≤M≤20) is the length of the list of boxes in this query, then M integers follow, representing the indices of the boxes.

Output

For each query, output a line which contains an integer representing the total number of boxes.

Sample Input 1

```
5
0 1 1 2 2
5
1 1
2 4 5
2 3 4
3 2 3 4
1 2
```

Sample Output 1

```
5
2
2
4
3
```

题意

 题意：n个箱子，其中一些箱子装在另一些里面。Q次询问，每次给定m个箱子，问这m个箱子里面一共包含了几个箱子（包括m个箱子本身）。

建树，若第i个箱子装在第j个箱子里面，j为i的父节点。

箱子编号1~n, 箱子组成树或者森林结构，确定一个根节点0，这样箱子组成的森林就合成一棵根节点为0的树。

从根节点0开始DFS遍历整个树, 记录每个节点DFS到的顺序编号DFN(i).

tot(i)表示箱子i包含的箱子总数（包括箱子i本身）。

每个箱子包含的箱子的集合在DFS遍历的顺序下是连续的。

range(i)表示箱子i包含的箱子的DFN编号范围，即DFN编号为range(i).first~range(i).second的所有箱子都包含在箱子i里面。

举个例子，样例建树DFS遍历：

其中，range(1).first=1,range(1).second=5; DFN编号为1~5的箱子都包含在箱子1中。

range(2).first=2,range(2).second=4; DFN编号为2~4的箱子都包含在箱子4中。

range(3).first=5,range(3).second=5; DFN编号为5的箱子包含在箱子3中。

range(4).first=3,range(4).second=3; DFN编号为3的箱子包含在箱子4中。

range(5).first=4,range(5).second=4; DFN编号为4的箱子包含在箱子5中。

得到了每个包含的箱子总数tot(i),和每个箱子i包含的箱子编号范围range(i)，

对于每一组查询q,暴力枚举一遍每个箱子q(i)是否被另一个箱子包含，如果q(i)被包含，删掉q(i)，最后没删掉的节点的tot求和。

AC代码

```
#include <algorithm>
#include <cstring>
#include <string.h>
#include <iostream>
#include <list>
#include <map>
#include <set>
#include <stack>
#include <string>
#include <utility>
#include <vector>
#include <cstdio>
#include <cmath>

#define LL long long
#define N 200005
#define INF 0x3ffffff

using namespace std;

int n;
int belong[N];        //belong[i]表示箱子i在箱子elong[i]里面
int qnum;              //查询次数
int m;
int q[25];
vector<int>vec[N];

int DFN[N];                           // 每个节点DFS到的顺序编号DFN[i]
pair<int,int>range[N];     //range[i]表示箱子i包含的箱子的DFN编号范
```

```
围
int tot[N];                         //tot[i]表示箱子i包含的箱子总数
 (包括箱子i本身
int pos;


void dfs(int u)          //DFS遍历树
{
    DFN[u]=pos++;
   range[u].first=pos;
    tot[u]=1;
    for(int i=0;i<vec[u].size();i++){
        int v=vec[u][i];
        dfs(v);
        tot[u]+=tot[v];
    }
    range[u].second=pos;
   return;
}

int main()
{
   while(scanf("%d",&n)!=EOF){
    pos=0;
    memset(tot,0,sizeof(tot));
     for(int i=1;i<=n;i++) vec[i].clear();

    for(int i=1;i<=n;i++)
    {
        scanf("%d",&belong[i]);
        vec[belong[i]].push_back(i);
    }
    dfs(0);      //从根节点0开始遍历

/*
     for(int i=1;i<=n;i++)
        {
            cout<<tot[i]<<endl;
            cout<<DFN[i]<<endl;
            cout<<range[i].first<<‘ ‘<<range[i].second<<endl;
        }
  */

    scanf("%d",&qnum);
    while(qnum--)
    {
        scanf("%d",&m);
        for(int i=0;i<m;i++)
        {
            scanf("%d",&q[i]);          //q数组为查询的箱子集合
        }
        int flag=0;                     //用flag来记录哪些箱子被
删掉了，如果(flag&(1<<i))，箱子q[i]就被删掉了。
        for (int i=0;i<m;i++)       //暴力枚举每个箱子q[i]，看是否包含在另一个箱子里面
        {
            if(!(flag&(1<<i)))
                {
                    for(int j=i+1;j<m;j++)
                    {
                        if(!(flag&(1<<j)))
                        {
                            if (range[q[i]].first<=range[q[j]].first && range[q[i]].second>=range[q[j]].second){          //箱子q[j]包含在箱子q[i]里面
                                flag|=(1<<j);
                                }
                            else if (range[q[j]].first<=range[q[i]].first && range[q[j]].second>=range[q[i]].second) {      //箱子q[i]包含在箱子q[j]里面
                                flag|=(1<<i);
                                break;
                                }
                        }
                    }
                }
        }

        int ret= 0;
        for (int i=0;i<m;i++)
        {
                if(!(flag&(1<<i)))  {     //箱子q[i]没被其他箱子包含
                    ret+=tot[q[i]];
            }
        }

        printf("%d\n",ret);
    }
   }
    return 0;
}
```

PROBLEM C

CLASSROOMS

The new semester is about to begin, and finding classrooms for orientation activities is always a headache.

There are k classrooms on campus and n proposed activities that need to be assigned a venue. Every proposed activity has specfic starting time si and ending time fi. Any such an activity should take place at one of the classrooms. Any of the k classrooms is big enough to hold any of the proposed activities, and each classroom can hold at most one activity at any time. No two proposed activities can take place at the same classroom at the same time. Even if two proposed activities overlap momentarily (the ending time of one activity equals the starting time another activity), they cannot be assigned to the same classroom.

There are so many proposed activities that there may not be enough classrooms to hold all the activities. It is desirable to have as many activities as possible. At most how many proposed activities can be assigned to the classrooms?

Input

- The first line contains two positive integers n and k (1≤k≤n≤200000), representing the number of proposed activities and number of classrooms, respectively.
- The following n lines each contains two positive integers: the ith line among these n lines contains si and fi (1≤si≤fi≤109), indicating the starting time and ending time of proposed activity i

Output

Output an integer indicating the maximum number proposed activities that can be scheduled.

Sample Input 1

```
4 2
1 4
2 9
4 7
5 8
```

Sample Output 1

```
3
```

题意

题意，n个活动，k个教室，给定每个活动开始和结束时间，在同一个教室举行的连续两个活动结束时间和开始时间之间必须有间隔。问最多能举办多少个活动。

贪心，把每个活动按结束时间排序，然后从头到尾扫一遍。

multiset里存放每个教室正在进行的活动的结束时间。

如果multiset里有某个教室的活动在活动i开始之前就结束的，活动i就可以举办，把原来的结束时间删掉，再把活动i的结束时间存进去。

如果multiset里没有比a(i).begin小的结束时间，即当前有活动的教室在活动i开始之前都结束不了活动，此时multiset里元素的数量表示有多少个教室在同时进行活动，如果还有空教室，活动i就可以在这个教室进行，把活动i的结束时间存入multiset。

注：实际存入multiset的是 (-a(i).ed-1),而查找时用的是（-a(i).begin）。因为要使用lower_bound函数，而lower_bound（start,end,k）返回的是集合里大于等于k的第一个数的下标，而题目里面要查找的是 比 开始时间 小的 第一个 结束时间，加个负号就刚好。

AC代码

```cpp
#include <algorithm>
#include <cstring>
#include <string.h>
#include <iostream>
#include <list>
#include <map>
#include <set>
#include <stack>
#include <string>
#include <utility>
#include <vector>
#include <cstdio>
#include <cmath>

#define LL long long
#define N 200005
#define INF 0x3ffffff

using namespace std;

int n , m;
struct node{
    int bg,ed;                  //m每场活动开始时间，结束时间
}a[N];

bool cmp(node a , node b){             //按照结束时间排序
    if(a.ed== b.ed) return a.bg < b.bg;
    return a.ed< b.ed;
}
```

```

int main(){
    while (~scanf("%d%d" , &n , &m))
        {
            for (int i = 0 ; i < n ; ++i)
            scanf("%d%d",&a[i].bg ,&a[i].ed);
            sort(a,a+n,cmp);

            multiset<int>endtime;         //h存放每个教室正在进行的活动的结束时间
            endtime.clear();
            int ans = 0;
            for (int i = 0 ; i < n ; ++i){
                multiset<int> :: iterator iter;
                iter = endtime.lower_bound(-a[i].bg);          //是否存在某个教室的活动在i开始时间前前就结束了
                if (iter == endtime.end()){                    //如果没有在活动i开始前就结束活动的教室，就另找一个教室
                    if (endtime.size() < m){
                        endtime.insert(-a[i].ed- 1);
                        ++ans;
                    }
                    continue;
                }
                endtime.erase(iter);                           //找到了某个教室活动已经结束了，活动i在这个教室进行
                endtime.insert( - a[i].ed - 1);       //更新活动的结束时间
                ++ans;
            }
        printf("%d\n" , ans);
        }
}
```

PROBLEM D

CURIOUS CUPID

There are *K* different languages in the world. Each person speaks one and only one language. There are exactly *N* single men and N single women.

Cupid, the god of love, wants to match every single man to a single woman, and vice versa. Everybody wants to find a partner who speaks the same language as s/he does. Communication between the couple is very important! Cupid asks these *N* men to stand in a line, and likewise for the *N* women. Cupid knows that the ith man speaks language *ai* and the ith woman speaks language *bi*.

It is too hard for Cupid to match all people at the same time. What Cupid does is to repeatedly look at some specific interval in these two lines, pick the men and women in that interval and find the maximum number of man-woman pairs who speak the same language and can be matched.

Input

- The first line contains three integers N, M and K ($1≤N≤50000, 1≤M≤50000, 1≤K≤1000000$).
- The second line contains N integers $a0,a1,a2,…,aN−1$, where ai ($0≤ai<K$) is the language spoken by the ith man.
- The third line contains N integers $b0,b1,b2,…,bN−1$, where bi ($0≤bi<K$) is the language spoken by the ith woman.
- In the next M lines, each line contains two integers L and R ($0≤L≤R<N$), representing the starting and ending index of the interval. That is, Cupid is looking at men $L,L+1,…,R$ and women $L,L+1,…,R$.

Output

For each interval, print the maximum number of couples Cupid could match.

Sample Input 1

```
3 4 2
0 0 1
0 0 0
0 0
2 2
0 1
1 2
```

Sample output 1

```
1
0
2
1
```

题意

给你两串数，一串代表男的，一串代表女的，每个数代表这个人说的语言，现在要给一个区间(L,R)里面的人配对，只有语言相同的才能配一对，问对于每个区间，最多能配成几对。莫队算法处理。

AC代码

```cpp
#include <bits/stdc++.h>
using namespace std;
#define N 51000
#define K 1000100
int n, m, k;
int a[N], b[N], pos[N], c[2][K], ans[N], cnt;
struct node {
    int l, r, id;
    void sc(int i){
        scanf("%d%d", &l, &r);
        id = i;
    }
}p[N];
bool cmp(node a, node b) {
    if(pos[a.l] == pos[b.l]) return a.r < b.r;
    return pos[a.l] < pos[b.l];
}
void update(int x, int y) {
    if(a[x] != b[x]) cnt -= min(c[0][a[x]], c[1][a[x]]) + min(c[0][b[x]], c[1][b[x]]);
    else cnt -= min(c[0][a[x]], c[1][a[x]]);
    c[0][a[x]] += y;
    c[1][b[x]] += y;
    if(a[x] != b[x]) cnt += min(c[0][a[x]], c[1][a[x]]) + min(c[0][b[x]], c[1][b[x]]);
    else cnt += min(c[0][a[x]], c[1][a[x]]);
}
void pri() {
    for(int j = 0;j < 2;j++)
        for(int i = 0;i < n;i++)
            printf("%d%c", c[j][i], " \n"[i==n-1]);
}
void solve() {
    memset(c, 0, sizeof(c));
    int pl = 0, pr = 0;
    cnt = a[0] == b[0] ? 1 : 0;
    c[0][a[0]]++; c[1][b[0]]++;
    for(int i = 0;i < m;i++) {
        int id = p[i].id, l = p[i].l, r = p[i].r;
        for(int j = pr + 1;j <= r;j++)
            update(j, 1);
        for(int j = pr;j > r;j--)
            update(j, -1);
        for(int j = pl;j < l;j++)
            update(j, -1);
        for(int j = pl-1; j >= l;j--)
            update(j, 1);
        pr = r; pl = l;
        ans[id] = cnt;
    }
    for(int i = 0;i < m;i++)
        printf("%d\n", ans[i]);
}
int main() {
    while(~scanf("%d%d%d", &n, &m, &k)) {
        for(int i = 0;i < n;i++)
            scanf("%d", &a[i]);
        for(int i = 0;i < n;i++)
            scanf("%d", &b[i]);
        int bk = sqrt(n + 1.0);
        for(int i = 0;i < n;i++)
            pos[i] = i / bk;
        for(int i = 0;i < m;i++)
            p[i].sc(i);
        sort(p, p+m, cmp);
        solve();
    }
    return 0;
}
```

PROBLEM E

DIM SUM

There are n people (labelled 1,...,n counterclockwise) sitting around a round table, equally spaced. There are n dishes (labelled 1,...,n) on a rotating tray on the table. At the beginning, dish i is right in front of person i (i=1,...,n). Person i has an unordered set of dishes Si(possibly empty) that he/she wants. At time 0, the tray can start rotating either clockwise or counterclockwise at a speed 1/n revolution per second (i.e., one position per second). The direction of rotation cannot change once the tray is in motion. At any time, if there is a dish right in front of a person that he/she wants, he/she can get his/her portion instantaneously (assume there is sufficient number of portions in each dish to satisfy everyone). We choose the direction of rotation that minimizes the time needed for everyone to get all the food they want.

For example, in the above figure, S1={},S2={1},S3={1,3}. If we rotate clockwise, then Person 2 will get Dish 1 at time 2, Person 3 gets Dish 3 at time 0, and will get Dish 1 at time 1, and hence the time needed is 2. One can check that the time needed for counterclockwise is also 2. Therefore the minimum time is 2.

The list of sets of dishes for each person (S1,S2,…,Sn) is called a food assignment. In this problem, we not only consider one food assignment, but all food assignment that satisfies certain constraints. Assume person i can only have dishes in the set Ti due to dietary constraints (so Si must be a subset of Ti). The task is to find the sum of the minimum time needed over all possible food assignment (S1,S2,…,Sn) satisfying the dietary constraints, modulo 1000000009.

As an example, consider the first sample input. We have T1={},T2={1},T3={1,3}, so S1={}. There are 8 possible food assignments:

S2	S3	Min time needed	Direction

{}	{}	0	either
{}	{1}	1	clockwise
{}	{3}	0	either
{}	{1,3}	1	clockwise
{1}	{}	1	counterclockwise
{1}	{1}	2	either
{1}	{3}	1	counterclockwise
{1}	{1,3}	2	either

The sum of the minimum time needed is 8.

Input

The first line contains the integer n (1≤n≤500000). Each of the next n lines describes a set Ti (i=1,…,n). The first integer in each line is ∣Ti∣ (the size of Ti), followed by ∣Ti∣ integers which are the elements of Ti. It is guaranteed that ∣T1∣+…∣Tn∣≤500000.

Output

Output the sum of the minimum time needed over all possible food assignment satisfying the dietary constraints, modulo 1000000009.

Sample Input 1

```
3
0
1 1
2 1 3
```

Sample output 1

```
8
```

sample Input 2

```
5
2 2 4
4 1 3 4 5
1 5
3 2 3 4
2 1 5
```

sample output 2

```
14676
```

题意

AC代码

```

```

PROBLEM F

CRAZY DRIVER

In the Linear City, there are N gates arranged in a straight line. The gates are labelled from 1 to N. Between adjacent gates, there is a bidirectional road. Each road takes one hour to travel and has a toll fee. Since the roads are narrow, you can only travel from gates to gates but cannot U-turn between gates.

Crazy driver Gary starts at Gate 1 at time 0 and he wants to drive through Gate N while minimizing the cost of travelling. However, Gate i only allows a car to pass through after a certain time Ti. As Gary is crazy, his car will always be traveling on any one of the roads, i.e., it will not stop at a gate. What is the minimum cost for him to drive through Gate N ?

As an example, consider the sample input below. An optimal solution is the following:

- Gate 1 to Gate 2 (cost 5)
- Gate 2 to Gate 1 (cost 5)
- Gate 1 to Gate 2 to Gate 3 (cost 9)
- Go between Gate 3 and Gate 4 until 7-th hour (cost 6)
- Go to and pass through Gate 5(cost 8)

Input

The first line contains an integer, N(2≤N≤105), the number of gates. The second line has N−1 integers, C1,…,CN−1. Ci (1≤Ci≤106) represents the toll fee of the road between Gate i and Gate i+1. The third line has N integers, T1,…,TN. Ti (0≤Ti≤106) represents the opening time (in hour) for each gate. T1 will always be 0.

Output

Output an integer representing the minimum cost of traveling.

Sample Input 1

```
5
5 4 2 8
0 2 4 4 8
```

Sample output 1

```
33
```

题意

题意：n个门编号1~n,从门i到i+1有一条双向通路，每条路花费的时间都是1小时，每条路花的路费分别是Ci, 每个门开的时刻分别是Ti，一个司机从门1开到门n，中间不停车，即如果到达门i的时候门没开就必须往返于前面的路上直到门开的时刻，问到门n最少花多少路费。

记录每扇门之前的路的最小路费。

AC代码

```
#include <algorithm>
#include <cstring>
#include <string.h>
#include <iostream>
#include <list>
#include <map>
#include <set>
#include <stack>
#include <string>
#include <utility>
#include <vector>
#include <cstdio>
#include <cmath>

#define LL long long
```

```cpp
#define N 100005
#define INF 0x3ffffff

using namespace std;

int n;
int c[N];               //门i-1到门i的路费是ci
int m[N];             //门i之前的路的路费最小值
int t[N];               //每个门开的时刻

int main()
{
    scanf("%d",&n);
    for(int i=1;i<=n-1;i++) {
        scanf("%d",&c[i]);
        if(i==1) m[i]=c[i];
        else m[i]=min(m[i-1],c[i]);
    }
      for(int i=0;i<n;i++) {
          scanf("%d",&t[i]);
      }

    int tt=0;           //当前时刻
    int i=0;
    long long ret=0;
    while(i<n)
        {
            i++;
            tt++;
            ret+=(long long)(c[i]);
            int tmp=t[i]-tt;                   //离门开还有多久

            while(tmp>0){
                tmp-=2;
                ret+=(long long)(m[i]*2);
                tt+=2;
            }
        }
        cout<<ret<<endl;
    return 0;
}
```

PROBLEM G

K-COLOURING OF A GRAPH

You are given a simple graph with N nodes and M edges. The graph has the special property that any connected component of size s contains no more than s+2 edges. You are also given two integers k and P. Find the number of k-colourings of the graph, modulo P.

Recall that a simple graph is an undirected graph with no self loops and no repeated edges. A k-colouring of a graph is a way to assign to each node of the graph exactly one of k colours, such that if edge (u,v) is present in the graph, then u and v receive different colors.

Input

The first line of input consists of four integers, N,M,k, and P (1≤N≤50000, 0≤M≤1.5N, 1≤k≤109, 1≤P≤2·109). The next M lines of input each contains a pair of integers A and B (1≤A≤N, 1≤B≤N), describing an edge in the graph connecting nodes A and B.

Output

Output the number of k-colourings of the given graph, modulo P.

Sample Input 1

```
3 3 2 10000
1 2
2 3
3 1
```

Sample output 1

```
0
```

Sample Input 2

```
3 3 4 13
1 2
2 3
3 1
```

Sample output 2

```
11
```

题意

AC代码

```

```

PROBLEM H

LET'S MEET

MTR (subway) stations are common meetup locations in Hong Kong. Alice and Bob are going to meet up at one of the MTR stations. They each show up at the meeting spot in one of the stations right at noon, without remembering which station they plan to meet at. Having poor cellphone connection at the stations, they fail to contact the other person. Fortunately, they have agreed to look for each other with the following strategy.

Each station has a unique designated meeting spot. Every minute starting from noon, Alice and Bob look for the other person at the meeting spot of their current stations. If they happen to be at the same station, they can finally meet up. Otherwise, they each takes a train to reach a neighbouring station, uniformly at random from among the neighbouring stations. Upon arrival, they look for the other person at the meeting spot of their current stations. If they still cannot find each other, each of them again takes a train to reach a neighbouring station, uniformly at random. This process goes on until they are at (the meeting spot of) the same station at the same time.

Trains move very fast, so Alice and Bob will not see each other while on trains running in opposite directions. Suppose it takes exactly one minute to get from the meeting spot of one station to that of a neighbouring station. What is the expected time they will meet?

Input

The first line of input contains two integers n and m. n is the number of MTR stations ($1 \le n \le 20$, $0 \le m \le n(n-1)/2$) and m is the number of pairs of stations that are neighbours.

Each of the following m lines contains two distinct integers, u and v ($0 \le u, v < n$, $u \ne v$), indicating that stations u and v are neighbours. Every unordered pair of neighbours (u,v) will appear at most once in the input.

The last line consists of two integers s and t (0≤s,t<n), denoting the initial stations of Alice and Bob, respectively.

Output

If Alice and Bob will never meet, output "never meet" without the quotes. Otherwise, output a real number indicating the expected time (in minutes past noon) they will meet at the meeting spot of a certain station. Any solution with a relative or absolute error of 10−6 will be accepted.

Sample Input 1

```
3 2
0 1
1 2
0 2
```

Sample Output 1

```
1
```

Sample Input 2

```
4 2
0 1
2 3
0 3
```

Sample Output 2

```
never meet
```

题意

AC代码

```

```

PROBLEM I

PALINDROMES

Sam is starting a new company, and has just chosen its name. The name is a string $s=s1s2\ldots sn$ of length n. The name is so long that customers won't remember the full name. Rather, each customer is going to remember only a substring t of the full name, and different customers may remember different substrings.

Sam is curious how memorable the substring*t*will be to a customer. Sam believes that the more palindromes *t* contains, the more memorable it is. Recall that a string *w* is a palindrome if *w* reads the same forwards and backwards (for example, the strings radar and level are palindromes, while the string ever is not). After all, customers are sensitive to symmetries in the substring *t* that he or she remembers, and palindromes are full of symmetries. As such, Sam wants to find out the number of substrings *w* of *t* such that *w* is a palindrome.

The challenge is that Sam's company will have tons of customers. It is difficult for Sam to know how memorable his company name is to all these customers. Can you help Sam?

Input

The first line of input contains a string $s=s1s2\ldots sn$ consisting of n lowercase letters $(1\leq n\leq 100000)$. The second line contains an integer m $(1\leq m\leq 100000)$, denoting the number of customers. Then m lines follow, each containing a pair of integers ℓ and r $(1\leq \ell\leq r\leq n)$, indicating that a customer remembers precisely the substring $s\ell s\ell+1\ldots sr$between the ℓth and rth characters inclusive.

Output

For each customer, output a line containing an integer that indicates how memorable the company name is to the customer. More specifically, if this customer remembers the substring $s\ell s\ell+1\ldots sr$, output the number of pairs of indices (i,j) such that $\ell\leq i\leq j\leq r$ and $sisi+1\ldots sj$ is a palindrome.

Sample Input 1

```
aabacab
5
1 7
1 4
3 7
2 5
5 7
```

Sample output 1

```
11
6
7
5
3
```

题意

AC代码

```

```

PROBLEM J

0-1 SEQUENCES

You are given a sequence, in the form of a string with characters '0', '1', and '?' only. Suppose there are k '?'s. Then there are 2k ways to replace each '?' by a '0' or a '1', giving 2k different 0-1 sequences (0-1 sequences are sequences with only zeroes and ones).

For each 0-1 sequence, define its number of inversions as the minimum number of adjacent swaps required to sort the sequence in non-decreasing order. In this problem, the sequence is sorted in non-decreasing order precisely when all the zeroes occur before all the ones. For example, the sequence 11010 has 5 inversions. We can sort it by the following moves: 11010 → 11001 → 10101 → 01101 → 01011 → 00111.

Find the sum of the number of inversions of the 2k sequences, modulo 1000000007 (109+7).

Input

The first and only line of input contains the input string, consisting of characters ‘0’, ‘1’, and ‘?’ only, and the input string has between 1 to 500000 characters, inclusive.

Output

Output an integer indicating the aforementioned number of inversions modulo 1000000007.

Sample Input 1

```
?0?
```

Sample output 1

```
3
```

题意

注意每一个1能够往后移动的次数就是他后面0的位数，假设后面有zero个0，ques个？，那么这个位置可以往后移动（zero + (zero+1)+...+(zero+ques)），这个就可以用一个等差数列来记录，然后还要乘以这个数前面 2^(问号数），加起来就是答案了

AC代码

```cpp
#include <cstdio>
#include <cstring>
const long long m = 1000000007;
char a[600010];
int preq[600010], b[600010];
long long ques = 0, zeros = 0, tmp = 0, ans = 0;
int main() {
    scanf("%s", a);
    int n = strlen(a);
    if (a[0] == '?') preq[0] = 1;
    else preq[0] = 0;
    for (int i = 1; i < n; i++) {
        if (a[i] == '?') preq[i] = preq[i - 1] + 1;
        else preq[i] = preq[i - 1];
    }
    b[0] = 1;
    for (int i = 1; i <= n; i++) b[i] = b[i - 1] * 2 % m;
    for (int i = n - 1; i >= 1; i--) {
        if (a[i] == '0') {
            zeros++;
        } else if (a[i] == '1') {
            if (ques == 0) {
                ans = (ans + zeros) % m;
            } else {
                tmp = ((ques * b[ques - 1] % m) + (zeros * b[ques] % m)) % m * (b[preq[i - 1]]) % m;
                ans = (ans + tmp) % m;
            }
        } else {
            if (ques == 0) {
                ans = (ans + zeros) % m;
            } else {
                tmp = ((ques * b[ques - 1] % m) + (zeros * b[ques] % m)) % m * (b[preq[i - 1]]) % m;
                ans = (ans + tmp) % m;
            }
            ques++;
        }
    }
    if (a[0] != '0') {
        if (ques == 0) {
            ans = (ans + zeros) % m;
        } else {
            tmp = ((ques * b[ques - 1] % m) + (zeros * b[ques] % m)) % m;
            ans = (ans + tmp) % m;
        }
    }
    printf("%lld\n", ans);
}
```

PROBLEM K

TAKE TWO STONES

Alice and Bob are playing a new game of stones. There are N stones placed on the ground, forming a sequence. The stones are labeled from 1 to N.

Alice and Bob in turns take exactly two consecutive stones on the ground until there are no consecutive stones on the ground. That is, each player can take stone i and stone i+1, where $1 \leq i \leq N-1$. If the number of stone left is odd, Alice wins. Otherwise, Bob wins.

Assume both Alice and Bob play optimally and Alice plays first, do you know who the winner is?

Input

The input contains an integer N ($1 \leq N \leq 10000000$), the number of stones.

Output

Output the winner, “Alice” or “Bob” (without the quotes), on a line.

Sample Input 1

```
3
```

Sample output 1

```
Alice
```

Sample Input 2

```
2
```

Sample output 2

```
Bob
```

Sample Input 3

```
5
```

Sample output 3

```
Alice
```

题意

判断奇偶。

AC代码

```cpp
#include <iostream>

using namespace std;
int main(int argc, char *argv[]){
    int n;
    cin >> n;
    if(n%2==0){
        cout << "Bob";
    }else{
        cout << "Alice";
    }
    return 0;
}
```