清华机试真题

2017 interview

生活在在外星球X上的小Z想要找一些小朋友组成一个舞蹈团，于是他在网上发布了信息，一共有 n 个人报名面试。**面试必须按照报名的顺序**依次进行。小Z可以选择在面试完若干小朋友以后，在所有**已经面试过**的小朋友中进行任意顺序的挑选，以组合成一个舞蹈团。虽然说是小朋友，但是外星球X上的生态环境和地球上的不太一样，这些小朋友的身高可能相差很大。小Z希望组建的这个舞蹈团要求**至少**有 m 个小朋友，并且这些小朋友的最高身高和最低身高之差不能超过 k 个长度单位。现在知道了这些小朋友的身高信息，问小Z至少要面试多少小朋友才能在已经面试过的小朋友中选出不少于 m 个组成舞蹈团。

轩轩

过了14个测试点！！！

```
# include<algorithm>
# include<vector>
# include<iostream>
using namespace std;
typedef struct{int id;int h;} student;
```

```
bool operator <(const student& a,const student & b){return (a.h)
<(b.h);}

int n,m,k,tall=0;
vector<student> stu;
vector<int> highid;
void init(){
    student temp;
    cin >> n >> m >> k;
    for(int i=0;i<n;i++){
        temp.id = i;
        cin >> temp.h;
        stu.push_back(temp);
    }
}
void pick(){
    for(int i = 0;i<=n-m;i++){ //排序
        sort(stu.begin(),stu.begin()+m+i);
        for(int j = 0;j<=i;j++){//检查个头
            if(stu[j+m-1].h-stu[j].h<=k){
                for(int y=j;y<j+m;y++){//找最大id
                    if(stu[y].id>tall) tall = stu[y].id;
                }
                cout<<tall+1<<endl;
                return;
            }
        }
    }
    cout<<"impossible"<<endl;
}
int main(){
    init();
    pick();
    return 0;
}
```

旭哥

测试点未通过

```
#include<iostream>
#include<map>
```

```cpp
#include<list>
#include<vector>
#include<algorithm>
using namespace std;
int n,m,k;
int h;
vector<int> max_nums;
map<int,vector<int> > mp;
int main()
{
    //input
    cin>>n>>m>>k;
    for(int i = 1;i<=n;++i){
        cin>>h;
        mp[h].push_back(i);
    }
    /*operation:
    (1)统一遍历一遍面试名单：从小到大，以遇到的每个身高h为区间起点，检测在[h,h+k)这个cell中，是否能找够m:
        是:则记录下他们最大的面试者编号；放入一个记录每个小区间max_num的vector中
        否:则跳过当前，进行下一个节点循环。
    (2)扫描一遍后，输出max_num的向量中，值最小的，是他们最少需要面试的人数。
     (3) 复杂度：应该是O(m*n);
    */
    map<int,vector<int> >::iterator it1 = mp.begin();//用了两个指针，构造数轴上的一个小区间
    map<int,vector<int> >::iterator it2 = it1;
    for(;it1!=mp.end();++it1){
        int counts = 0;
        int max_num = -1;
        bool enough_flag = false;
        int min_high = it1->first;//记录检测小cell中的最小身高（即基准身高），与其他身高之差，应不大于k
        for(it2 = it1;it2!=mp.end();++it2){
            if(it2->first - min_high <= k && counts <= m){
                counts += it2->second.size();
                if(it2->second[it2->second.size()-1] > max_num ){
                    max_num = it2->second[it2->second.size()-1];
                }
            }else if(counts == m) {
```

```
                enough_flag = true;
                break;
            }else  break;
        }
        if(max_num !=-1 && enough_flag ==true )
max_nums.push_back(max_num);
    }
    //搜索结束，找到所有小区间上的最小的编号
    int i= max_nums.size();
    vector<int>::iterator it3 = max_nums.begin();
    sort(it3,it3+i);
    if(!max_nums.empty()){
        cout<<max_nums.front()<<endl;
    }else
        cout<<"impossible"<<endl;
   return 0;
}
```

2017多项式求和

小K最近刚刚习得了一种非常酷炫的多项式求和技巧，可以对某几类特殊的多项式进行运算。非常不幸的是，小K发现老师在布置作业时抄错了数据，导致一道题并不能用刚学的方法来解，于是希望你能帮忙写一个程序跑一跑。给出一个 m 阶多项式 $f(x)=\sum_{i=0}^{m} b_i x^i$对给定的正整数 a，求$S(n)=\sum_{k=0}^{n} a^k f(k)$由于这个数可能比较大，所以你只需计算 $S(n)$ 对 10^9+7 取模后的值（即计算除以 10^9+7 后的余数）。

旭哥

只能过两个测试点！！！

```
#include<algorithm>
#include<cmath>
#include<stdio.h>
typedef long long ll;
using namespace std;

ll n,m,a;
int mod = 1e9+7;
```

```
//计算(a*b)%c
ll mul(ll a,ll b,ll mod) {
    ll res = 0;
    while(b > 0){
        if( (b&1) != 0)  // 二进制最低位是1 --> 加上 a的 2^i 倍, 快速幂是乘上a的2^i )
            res  = ( res + a) % mod;
        a = (a << 1) % mod;    // a = a * 2    a随着b中二进制位数而扩大 每次 扩大两倍。
        b >>= 1;               // b -> b/2     右移  去掉最后一位 因为当前最后一位我们用完了,
    }
    return res;
}

//幂取模函数
ll pow1(ll a,ll n,ll mod){
    ll res = 1;
    while(n > 0){
        if(n&1)
            res = (res * a)%mod;
        a = (a * a)%mod;
        n >>= 1;
    }
    return res;
}


// 计算 ret = (a^n)%mod
ll pow2(ll a,ll n,ll mod) {
    ll res = 1;
    while(n > 0) {
        if(n & 1)
            res = mul(res,a,mod);
        a = mul(a,a,mod);
        n >>= 1;
    }
    return res;
}

//递归分治法求解
ll pow3(ll a,ll n,ll Mod){
```

```
    if(n == 0)
        return 1;
    ll halfRes = pow1(a,n/2,Mod);
    ll res = (ll)halfRes * halfRes % Mod;
    if(n&1)
        res = res * a % Mod;
    return res;
}

ll fun(ll b[],ll x)
{
    ll fx = 0;
    for(ll i = 0;i <= m;++i){
        fx +=  mul(b[i],pow3(x,i,mod),mod);
    }
    return fx;
}

ll Sum(ll a,ll n,ll b[])
{
    ll sum=0;
    for(ll i = 0;i <= n;++i){
        sum += mul(pow3(a,i,mod),fun(b,i),mod);
    }
    return sum%mod;
}

int main()
{
    //input:
    ll temp;
    //cin>>n>>m>>a;
    scanf("%lld %lld %lld",&n,&m,&a);
    ll b[m+1];
    for(ll i = 0;i <= m;++i){
        scanf("%lld",&b[i]);
    }
    printf("%lld",Sum(a,n,b));
    //cout<<Sum(a,n,b)<<endl;
    return 0;
}
```

轩轩

只能过两个测试点！！！

```
# include <stdio.h>
# include <iostream>
# define  MAX 1000000007
# define ll long long

using namespace std;

ll n,m,a,*b;


long long mul(long long a,long long b,long long mod) {
    long long res = 0;
    while(b > 0){
        if( (b&1) != 0)  // 二进制最低位是1 --> 加上 a的 2^i 倍, 快速幂是乘上a的2^i )
            res  = ( res + a) % mod;
        a = (a << 1) % mod;     // a = a * 2     a随着b中二进制位数而扩大 每次 扩大两倍。
        b >>= 1;                // b -> b/2      右移   去掉最后一位 因为当前最后一位我们用完了,
    }
    return res;
}



long long pow(long long a,long long n,long long mod) {
    long long res = 1;
    while(n > 0) {
        if(n & 1)
            res = mul(res,a,mod);
        a = mul(a,a,mod);
        n >>= 1;
    }
    return res;
}
```

```
void init(){
    cin>>n>>m>>a;
    b = new ll[m+1];
    for(ll i=0;i<=m;i++){
    cin>>b[i];
    }
}


ll f(ll x){
    ll result=0;
    for(ll i=0;i<=m;i++){
        result+=mul(b[i],pow(x,i,MAX),MAX);
        result%=MAX;
    }
    return result;
}

ll s(){
    ll result=0;
    for(ll i = 0;i<=n;i++){
        result+=mul(pow(a,i,MAX),f(i),MAX);
        result%=MAX;
    }
    return result;
}

int main(){
    init();
    cout<<s()<<endl;
    return 0;
}
```

2018葱的战争

一个n乘m的棋盘，上面有k根葱，每根葱面朝方向为d（0123分别表示上下左右），没跟葱一个战斗力f。每隔时间葱会向面朝方向走一格，如果遇到棋盘边界，那么他将把面朝方向转180度（此回合葱不会走动），如果某个时刻有两个或以上的葱在同一位置，那么他们将发生战争，只有战斗力最高的葱存活，其他的葱全部原地枯萎，不在走动，求经过t时间后所有葱的位置

输入：第一行n m k，然后接下来k行每根葱的信息x y d f（坐标，方向，战斗力），最后一行输入时间t 输出：k行，分别表示每个葱的位置。 数据范围：n和m在100内，k在1000内，t在1000内，f在1000内，保证初始每颗葱位置不同，战斗力不同。

轩轩

以下代码测试点通过

```
# include<iostream>
# include<map>
# include<vector>
using namespace std;
typedef struct{int x,y;} position;
typedef struct{int id,f;} idfight;
typedef struct{int id;position p;int d;int f;bool live;} cong;
typedef vector<cong> conglist;
conglist all_cong;
map<int,vector<idfight> >war_map;
int n,m,k,times;
void init(){
    cin>>n>>m>>k;
    for(int i=0;i<n;i++){
        int x,y,d,f;
        cin >> x >>y>>d>>f;
        cong c1 ={i,{x,y},d,f,1};
        all_cong.push_back(c1);
    }
    cin >> times;
}
void action(cong &c){
    if(c.live){
    switch(c.d){
    case 0: if(c.p.y==m) c.d=1;else c.p.y++;break;
    case 1: if(c.p.y==1) c.d=0;else c.p.y--;break;
    case 2: if(c.p.x==1) c.d=3;else c.p.x--;break;
    case 3: if(c.p.y==n) c.d=2;else c.p.x++;break;
    default:;break;
    }
    int pi = c.p.x*1000+c.p.y;
    idfight idf = {c.id,c.f};
    war_map[pi].push_back(idf);
    }
```

```
}
void printans(){
        for(vector<cong>::iterator i =
all_cong.begin();i!=all_cong.end();i++)
        cout<<(*i).p.y<<" "<<(*i).p.x<<endl;
}
void fight(){
    map<int,vector<idfight> >::iterator it;
    it = war_map.begin();
    while(it!=war_map.end()){
        if((*it).second.size()>1){
            int max = 0;
            for(vector<idfight>::iterator i =
(*it).second.begin();i!=(*it).second.end();i++){
                if((*i).f>max)max = (*i).f;
            }
            for(vector<idfight>::iterator i =
(*it).second.begin();i!=(*it).second.end();i++){
                if((*i).f<max) all_cong[(*i).id].live=0;
            }
        }
    it++;
    }
}
int main() {
    init();
    while(times--){
    for(vector<cong>::iterator i =
all_cong.begin();i!=all_cong.end();i++){
        action(*i);
    }
    fight();
    war_map.clear();
    }
    printans();
    return 0;
}
```

2018路径

有n个点，每个点有一个权值，每两点间的不同边的个数为他们权值相与得到的值的二进制数字中的1的个数（边为有向边，有第i指向第j，i小于j）求第1个点到第n个点的路径个数（当且仅当不止一条边不同才被称为两条不同的路径），由于数据很大，对991247取模。

输入：第1行n，第二行分别试每个点权值 输出：路径个数 数据范围:n在2e5内，权值大小在1e9内

旭哥

以下代码测试点通过

```
#include<iostream>
using namespace std;
struct Node
{
    int n;
    int w;
}node[200003];
int countOnes(int a,int b)
{
    //count the number in the ans(&)
    int c = a&b;
    int ones = 0;
    while(c != 0){
        if(c%2 == 1)
            ones++;
        c = c>>1;
    }
    return ones;
}

int Routes(int n)
{
    if(n == 1 || n==0 )
        return 0;
    else if(n > 1)
        return countOnes(1,n) + Routes(n-1)*countOnes(n-1,n);
}
int main()
{
    int n,f,routes = 0;
```

```
    cin>>n;
    for(int i = 1;i <= n;++i){
        cin>>f;
        node[i].n = i;
        node[i].w= f;
    }
    routes = Routes(5);
    cout<<routes<<endl;
    return 0;

}
```

2018四种操作

有一个n个元素的数列,元素的值只能是0 1 2三个数中的一个，定义四种操作，(1 i x)表示为把第i位替换成x，x也只能是0 1 2三个数中的一个，(2 i j)表示把第i个数到第j个数所有的元素值加1，并对3取模，(3 i j)表示把第i个数到第j个数之间的序列的颠倒顺序，(4 i j)表示查询第i个数到第j个数之间的序列是否存在三个或以上相同数，如果有，输出yes，否则输出no

输入：第一行输入n，接下来一行输入n个数，保证是0 1 2中的一个，第三行输入一个数q，表示操作个数，接下来q行输入q种操作 输出：每次第四次操作时，输出yes或者no 数据范围：不记得了

波哥

```
#include<iostream>
#include<algorithm>
#include<cstdio>
using namespace std;
int a[100000];
void replace(int a[],int i,int x ){
     a[i-1]=x;
}

void addx_y(int a[],int x,int y){
    for(int i=x-1;i<y;i++){
       a[i]=(a[i]+1)%3;
    }
}
void ser_com3(int a[],int x,int y){
```

```
    int zeros=0;
    int ones=0;
    int twos=0;
    bool flag=0;
    for(int k=x-1;k<y;k++){
       if(a[k]==0)zeros++;
       if(a[k]==1)ones++;
       if(a[k]==2)twos++;
       if(zeros>=3||ones>=3||twos>=3){
           flag=1;
           cout<<"yes"<<endl;
           break;
       }
    }
    if(!flag)cout<<"no"<<endl;
}

int main(){
    int n;
    int op,x,y;
    while(scanf("%d",&n)!=EOF){
        for(int i=0;i<n;i++)scanf("%d",&a[i]);
        int q;
        cin>>q;
        while(q--){
            cin>>op>>x>>y;
            switch(op){
                case 1: replace(a,x,y);
                case 2: addx_y(a,x,y);
                case 3: reverse(a+x-1,a+y);
                case 4: ser_com3(a,x,y);
                default:break;
            }
        }
    }
    return 0;
}
```

旭哥

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

```
int a[10000001];
void rep(int a[],int i,int e)
{
    a[i] = e;
}

void addall(int a[],int i,int j)
{
    for(int k = i;k <= j;++k){
        a[k] = (a[k]+1)%3;
    }
}

void rev(int a[],int i,int j)
{
    int temp;
    if((j-i)%2 == 0){
        for(int k = i,m = j;k != m ;++k,--m){
            temp = a[k];
            a[k] = a[m];
            a[m] = temp;
        }
    }else{
        for(int k = i,m = j;k != m-1 ;++k,--m){
            temp = a[k];
            a[k] = a[m];
            a[m] = temp;
        }
    }
}

void quire(int a[],int i,int j)
{
    int zeros,ones,twos;
    for(int k = i;k<j;++k){
        if(a[k] == 0)
            zeros++;
        else if(a[k] == 1)
            ones++;
        else if(a[k] == 2)
            twos++;
```

```
        if(zeros >= 3||ones >= 3 || twos >= 3){
            cout<<"yes"<<endl;
            break;
        }
    }
    cout<<"no"<<endl;
}

int main()
{
    int n,q,x,op,s,e;

    //input
    for(int i = 0;i<n;++i){
        cin>>x;
        a[i] = x;
    }
    cin>>q;
    for(int i = 0;i < q ;++i){
        cin>>op>>s>>e;
        if(s == 1)
            rep(a,s,e);
        else if(s == 2)
            addall(a,s,e);
        else if(s == 3 )
            rev(a,s,e);
        else if(s == 4 )
            quire(a,s,e);
    }

    return 0;
}
```

轩轩

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

typedef struct {int x,y,z;} comm;
```

```
vector<int> numlist;
vector<comm> commlist;
int n,q;

void init(){
    int temp;
    cin>>n;
    while(n--){
        cin>>temp;
        numlist.push_back(temp);
    }
    cin>>q;
    int x,y,z;
    while(q--){
        cin>>x>>y>>z;
        comm command = {x,y,z};
        commlist.push_back(command);
    }
}
void printdata(){
    vector<int>::iterator it;
    it = numlist.begin();
    while(it!=numlist.end()){
    cout<<(*it)<<" ";
    it++;
    }
    cout<<endl;
}
void action_1(int i,int x){
    numlist[i-1]=x;
}

void action_2(int i ,int j){
    i--;j--;
    for(;i<=j;i++){
        numlist[i]=(numlist[i]+1)%3;
    }

}
void action_3(int i ,int j){
    i--;
    reverse(numlist.begin()+i,numlist.begin()+j);
```

```
}
void action_4(int i , int j){
    i--;
    int a,b,c;
    a =  count(numlist.begin()+i,numlist.begin()+j,0);
    b =  count(numlist.begin()+i,numlist.begin()+j,1);
    c =  count(numlist.begin()+i,numlist.begin()+j,2);
    if((a>2)||(b>2)||(c>2))cout<<"yes"<<endl;
    else cout<<"no"<<endl;
}
int main(){
    init();
    vector<comm>::iterator i;
    i = commlist.begin();
    while(i!=commlist.end()){
        switch((*i).x){
        case 1:action_1((*i).y,(*i).z);break;
        case 2:action_2((*i).y,(*i).z);break;
        case 3:action_3((*i).y,(*i).z);break;
        case 4:action_4((*i).y,(*i).z);break;
        default:;
        }
    i++;
    }
    printdata();
    return 0;
}
```