2023.8.5 思维好题 题单

1. 11

【问题描述】

长度为n+1的序列a.其中[1..n]每个数都至少出现一次.（n<=1e5）

对每个k从1到n+1，询问长度为k的不同的子序列有多少个?

【链接】https://www.luogu.com.cn/problem/AT2649

【输入格式】

第一行输入n

第二行输入n个数

【输出格式】

对于k从1到n+1，输出答案。

【样例输入】

3
1 2 1 3

【样例输出】

3
5
4
1

题解

考虑总的方案数是C（n+1，k），从中去掉那些重复的。

重复的两个数字如果是i和j，那么重复的方案一定是

xxxixxx

xxxjxxx

也就是i到j的区间内一定没选

从i到j区间之外选了k-1个数，把这个组合数去掉即可。

代码

```cpp
#include <iostream>
#include <algorithm>
#include <cstring>
#include <cstdio>
#include <vector>
#include <cstdlib>
#include <iomanip>
#include <cmath>
#include <ctime>
#include <map>
#include <set>
#include <queue>
#include <stack>
using namespace std;
#define lowbit(x) (x&(-x))
#define max(x,y) (x>y?x:y)
#define min(x,y) (x<y?x:y)
#define MAX 100000000000000000
#define MOD 1000000007
#define pi acos(-1.0)
#define ei exp(1
#define PI 3.141592653589793238462
#define INF 0x3f3f3f3f3f
#define mem(a) (memset(a,0,sizeof(a)))
typedef long long ll;
ll gcd(ll a,ll b){
    return b?gcd(b,a%b):a;
}
const int N=200005;
const int maxn = 1e5 + 10;
ll pos[maxn],fac[maxn],facm[maxn];
ll quick_pow(ll a,ll n,ll p)
{
    ll x = a;
    ll res = 1;
    while(n){
        if(n & 1){
            res = ((ll)res * (ll)x) % p;
        }
        n >>= 1;
        x = ((ll)x*(ll)x) % p;
    }
    return res;
```

```
}
ll C(ll n,ll k){
    if(k > n) return 0ll;
    ll ans = fac[k]*fac[n-k]%MOD;
    ans = (fac[n]*quick_pow(ans,MOD-2ll,MOD))%MOD;
    return ans;
}
int main()
{
    fac[0] = 1;
    for(int i = 1;i < maxn;i++)
        fac[i] = (fac[i-1]*i)%MOD;
    ll n;
    scanf("%lld", &n);
    n++;
    ll m, x;
    for(int i = 1;i <= n;i++){
        scanf("%lld", &x);
        if (pos[x]){
            m = n - (i - pos[x] + 1);
            break;
        }
        pos[x] = i;
    }
    for(int i = 1;i <= n;i++)
        printf("%lld\n", (C(n, i) - C(m, i-1)+MOD) % MOD);
    return 0;
}
```

2. Card Eater

【问题描述】

有一堆牌，每张牌上有一个数字。 每次可以取出其中3张，丢掉数字最大的和数字最小的牌，把中间那张再放回牌堆。 要求最后所有剩余牌上的数字互不相同，求最多能剩几张牌。牌数不超过1e5

【链接】https://www.luogu.com.cn/problem/AT2299

【输入格式】

第一行输入n

接下来输入n个数字

【输出格式】

输出答案。

【样例输入】

5
1 2 1 3 7

【样例输出】

3

题解

可以发现，只要有两张相同的，那么就可以选这两个牌再任选一张，然后把这两张牌的任一张与任选的一张扔掉

因此，任何两张重复的卡片都可以被直接去掉，如果重复卡片的数量是奇数，需要再额外付出1张卡片的代价。因此判断重复卡片数量的奇偶性就可以了。

代码

```cpp
#include<iostream>
#include<cstdio>
#include<cstdlib>
#include<string>
#include<cstring>
#include<cmath>
#include<ctime>
#include<algorithm>
#include<utility>
#include<stack>
#include<queue>
#include<vector>
#include<set>
#include<map>
#define EPS 1e-9
#define PI acos(-1.0)
#define INF 0x3f3f3f3f
#define LL long long
const int MOD = 1E9+7;
const int N = 100000+5;
using namespace std;
int bucket[N];
int main(){
    int n;
    scanf("%d",&n);
    for(int i=1;i<=n;i++){
        int x;
        scanf("%d",&x);
```

```
29          bucket[x]++;
30      }
31
32      int type=0;
33      for(int i=1;i<=N;i++)
34          if(bucket[i]!=0)
35              type++;
36
37      int res;
38      if(type%2)
39          res=type;
40      else
41          res=type-1;
42
43      printf("%d\n",res);
44      return 0;
45  }
```

3. Contiguous Repainting

【问题描述】

有 *N* 个格子排成一列，从左起第 i 个格子中写着整数 *ai*。

开始时，每个格子被涂成白色。sunuke君将重复进行以下操作。

- 选择连续的 K 个格子，将它们全部涂成白色或全部涂成黑色。此操作将会覆盖掉格子原来的颜色。

ssunuke君希望在操作完成后，黑色格子中整数的和最大。请求出此最大值。n不超过1e5

【链接】https://www.luogu.com.cn/problem/AT2264

【输入格式】

第一行输入n，k

接下来一行输入n个数字

【输出格式】

输出答案

【输入样例】

5 3
-10 10 -10 10 -10

【输出样例】

10

题解

除了最后一次染色的那个长度为k的区间以外的所有格子，都可以任意确定颜色

先贴着整个序列两侧用区间涂色，然后慢慢地把涂色区间向中间挪，这样就可以一格一格地确定颜色

代码

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

const int N = 1e5 + 10;

int a[N];
long long pre[N], suf[N], sum[N];

int main(){
        int n, k;
        long long ans;
        scanf("%d %d",&n,&k);
        for (int i = 1; i <= n; i++) scanf("%d",&a[i]);
        sum[0] = 0;
        for (int i = 1; i <= n; i++) sum[i] = sum[i - 1] + a[i];
        pre[0] = 0;
        for (int i = 1; i <= n; i++) pre[i] = pre[i - 1] + max(a[i], 0);
        suf[n + 1] = 0;
        for (int i = n; i >= 1; i--) suf[i] = suf[i + 1] + max(a[i], 0);
        ans = 0;        //枚举区间
        for (int i = 0; i <= n - k; i++)
                ans = max(ans, pre[i] + suf[i + k + 1] + max(sum[i + k] - sum[i]
        printf("%lld\n", ans);
        return 0;
}

```

4. Ranking

【问题描述】

题意：给定去年一场比赛的排名，还有一些pair，每个pair表示今年的排名中这两个选手的相对排名变了。有且仅有这些pair的相对位置发生了变化，求出今年的排名或者不存在，或者某些位置不确定。

【链接】https://codeforces.com/gym/101404/attachments E题

【输入】

多组数据，第一行是数据组数。

每组数据第一行包含整数n，表示队伍的数量，不超过500

接下来是n个整数，表示去年的球队排名，从第一名到最后一名。

接下来是m，表示修改的pair数量,不超过25000

接下来m行，每行是两个数，表示一个pair。

【输出】

打印一行n个数，表示今年的排名。如果不存在输出IMPOSSIBLE，如果有不确定的位置输出?

【输入】

3

5

5 4 3 2 1

2

2 4

3 4

3

2 3 1

0

4

1 2 3 4

3

1 2

3 4

2 3

【输出】

5 3 2 4 1

2 3 1

IMPOSSIBLE

题解

建立竞赛图，表示去年的比赛状况。这个竞赛图的拓扑序就是去年的排名。

然后根据pair的信息将一些边反向。依然是一张竞赛图。

然后进行拓扑排序即可。如果有环则证明信息错误。

但是还有不确定的问题。实际上不可能有不确定的情况出现。出现不确定的清空当且仅当某一时刻图中有两个入度为0的点。

在整个拓扑排序的过程中一直是竞赛图。而竞赛图不可能存在两个入度为0的点。因为两个点之间总有一条边相连，要么指a要么指b。

代码

拓扑排序模板题，就不提供代码啦

5. Median Sum

【问题描述】

一个集合由N个整数组成，请求出它的所有非空子集和的中位数

- $1 \leqslant N \leqslant 2000$
- $1 \leqslant Ai \leqslant 2000$

【链接】https://www.luogu.com.cn/problem/AT3857

【输入格式】

第一行输入n

接下来一行输入n个数

【输出格式】

输出答案

【样例输入】

3
1 2 1

【样例输出】

2

题解

我们记所有元素权值的和为s。

那么如果把空序列也考虑进来的话，对于任意一个权值为x的子序列，一定能找到一个权值为$s-x$的子序列。

也就是说，所有子序列的权值在s/2左右两边是对称的。

然而我们不考虑空序列，所以中位数一定是权值大于等于s/2的子序列中权值最小的那一个子序列的权值。

这个可以直接用背包+bitset加速。

代码

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

#define MAXN 2010bitset<2000010> f;
int a[MAXN],n,sum;

int main(){
    scanf("%d",&n);
    for(int i=1;i<=n;++i)scanf("%d",a+i),sum+=a[i];
    f[0]=1;
    for(int i=n;i>=1;--i)f|=f<<a[i];
    for(int i=(sum+1)>>1;i<=sum;++i)
        if(f[i])
        {
            printf("%d\n",i);
            return 0;
        }
}
```

6. Vasya and String

【问题描述】

给你一个n个字符a与b组成的字符串，求至多改动k个字符后连续相同字符个数的最大值。

N和k不超过1e5

【链接】https://www.luogu.com.cn/problem/CF676C

【输入格式】

第一行输入n和k

接下来输入字符串

【输出格式】

输出最大值

【样例输入】

4 2
abba

【样例输出】

4

题解

设两个指针，分别记录最长字符串的左端点和右端点。首先右移右指针，若此位置上字符是 a 则 a 的数量加一，否则 b 的数量加一。随后若 a 或 b 的数量超过了 k，则右移左指针，将 a 或 b 的数量减一，否则尝试更新答案。

代码

```cpp
#include <cstdio>
#include <iostream>
using namespace std;
char s[100005];
int main() {
    int n, k;
    scanf("%d %d", &n, &k);
    scanf("%s", s);
    int l = 0, r = 0, ans = 0;
    int cnt1 = 0, cnt2 = 0;
    while(r < n) {
        if(s[r] == 'a')  cnt1++; else cnt2++;
        if(cnt1 <= k || cnt2 <= k)  ans++, r++;
        else {
            if(s[l] == 'a') cnt1--;  else cnt2--;
            l++, r++;
        }
    }
    printf("%d", ans);
    return 0;
}
```

7. 每条边的最小生成树

【问题描述】

给出一张图，对于每一条边，求出包含这条边的最小生成树。

【链接】https://codeforces.com/problemset/problem/609/E

【输入】

第一行包含两个整数 n 和 m (1 ⩽ n ⩽ 2·10^5， n - 1 ⩽ m ⩽ 2·10^5） — 图形中顶点和边的数量。

接下来m行，每行三个数，分别是两个顶点和其边权。

【输出】

打印 m 行。第 i 行应包含包含第 i 条边的生成树的最小可能权重。

边的编号按照输入的边的顺序。

【输入】

5 7

1 2 3

1 3 1

1 4 5

2 3 2

2 5 3

3 4 2

4 5 4

【输出】

9

8

11

8

8

8

9

题解

首先考虑最简单的情况，在最小生成树的边的答案就是最小生成树的大小。

对于剩下的边，考虑如何强制把一条边加入到最小生成树上。当把一条边加入后，就形成了一个环，所以只需要断掉这个环上的一条边即可

问题转化成了求出树上两点之间的最大值

代码

```
#include<iostream>
#include<cstdio>
#include<algorithm>
#include<cstring>
#include<cmath>
#include<set>
#include<vector>
#include<ctime>
#define ll long long
#define pr(x) cerr<<#x<<"="<<x<<endl
using namespace std;
#define N 410000
struct node
{
        ll l,r,id,ans,v;
}a[N];
struct node1
{
        ll to,next,v;
}e[N];
ll g[N],size,fa[N][20],maxx[N][20],dep[N],ans,n,m,i,tot,fa1[N],sum;
void add(ll o,ll p,ll q)
{
        e[++size].to=p;
        e[size].next=g[o];
        g[o]=size;
        e[size].v=q;
}
bool cmp(node x,node y)
{
 return x.v<y.v;
}
bool cmp1(node x,node y)
{
 return x.id<y.id;
}
ll find(ll x)
{
 if (x==fa1[x]) return x;
        fa1[x]=find(fa1[x]);
 return fa1[x];
}
void dfs(ll x)
{
```

```
 for (ll i=1;i<=18;i++)
        {
                fa[x][i]=fa[fa[x][i-1]][i-1];
                maxx[x][i]=max(maxx[x][i-1],maxx[fa[x][i-1]][i-1]);
        }
 for (ll k=g[x];k;k=e[k].next)
        {
                ll y=e[k].to;
  if (y!=fa[x][0])
                {
                        fa[y][0]=x;
                        maxx[y][0]=e[k].v;
                        dep[y]=dep[x]+1;
   dfs(y);
                }
        }
}
void solve(ll x,ll y)
{
 if (dep[x]<dep[y]) swap(x,y);
        ll d=dep[x]-dep[y];
 for (ll i=0;i<=18;i++)
        {
  if(1&(d>>i))
                {
                        ans=max(ans,maxx[x][i]);
                        x=fa[x][i];
                }
        }
 if (x==y) return ;
 for (ll i=18;i>=0;i--)
        {
  if (fa[x][i]!=fa[y][i])
                {
                        ans=max(ans,maxx[x][i]);
                        ans=max(ans,maxx[y][i]);
                        x=fa[x][i];
                        y=fa[y][i];
                }
        }
        ans=max(ans,max(maxx[x][0],maxx[y][0]));
 return ;
}
int main()
{
 scanf("%I64d %I64d",&n,&m);
 for (i=1;i<=m;i++)
```

```
92          {
93   scanf("%I64d %I64d %I64d",&a[i].l,&a[i].r,&a[i].v);
94                  a[i].id=i;
95          }
96  sort(a+1,a+1+m,cmp);
97          tot=0;
98  for (i=1;i<=n;i++) fa1[i]=i;
99  for (i=1;i<=m;i++)
100         {
101                 ll f1=find(a[i].l),f2=find(a[i].r);
102  if (f1!=f2)
103                 {
104                         fa1[f1]=f2;
105   add(a[i].l,a[i].r,a[i].v);
106   add(a[i].r,a[i].l,a[i].v);
107                         sum+=a[i].v;
108                         tot++;
109   if (tot==n-1) break;
110                 }
111         }
112  dfs(1);
113  for (i=1;i<=m;i++)
114         {
115                 ans=0;
116  solve(a[i].l,a[i].r);
117                 a[i].ans=sum-ans+a[i].v;
118         }
119  sort(a+1,a+1+m,cmp1);
120  for (i=1;i<=m;i++) printf("%I64d\n",a[i].ans);
121 }
```

---

8. Intersection

【问题描述】

给出一个平面上的n条直线，还有两条平行的直线形成一个长条状的区间，询问有多少的交点在这两条直线构成的区间内。

【输入】

第一行表示n

接下来n行，每行四个数字，x1，y1，x2，y2，表示一条直线表示的两个点的坐标

接下来两行，表示两个平行的直线。

【输出】

输出交点的数量

【数据范围】

1<=n<=10^5

坐标范围1e9

题解

两条直线在平板之间有交点当且仅当他们在平板之间有交叉，也就是和平板的交点横坐标的大小关系是相反的。所以我们处理出每条直线和平板的交点横坐标，按照其中的一个排序，在另外的一边求逆序对数即可。

因为交点的横坐标是实数，所以用归并排序比较好。

代码

```
#include<iostream>
#include<cstdio>
#include<algorithm>
#include<cstring>
#include<cmath>
#include<set>
#include<vector>
#include<ctime>
#define ll long long
#define pr(x) cerr<<#x<<"="<<x<<endl
using namespace std;
int ans,n,i;
double k1,b1,b2,ex[1001000],k,b,a[1001000];
struct node
{
 double a,b;
}tmp[1001000];
bool cmp (node x,node y)
{
 return x.a<y.a;
}
void merge(int l,int mid,int r)
{
 int x=l,y=mid+1,t=l;
 while (x<=mid&&y<=r)
        {
```

```
  if(a[y]<a[x]) ans+=mid-x+1,ex[t++]=a[y],y++;
  else ex[t++]=a[x],x++;
       }
 while (x<=mid) ex[t++]=a[x],x++;
 while (y<=r) ex[t++]=a[y],y++;
 for (int i=l;i<=r;i++) a[i]=ex[i];
}
void divide(int l,int r)
{
 if (l>=r) return;
 int mid=(l+r)/2;
 divide(l,mid);
 divide(mid+1,r);
 merge(l,mid,r);
}
int main()
{
 scanf("%lf %lf %lf",&k1,&b1,&b2);
 scanf("%d",&n);
 for (i=1;i<=n;i++)
       {
  scanf("%lf %lf",&k,&b);
               tmp[i].a=(double)(b-b1)*1.0/(k1-k);
               tmp[i].b=(double)(b-b2)*1.0/(k1-k);
       }
 sort(tmp+1,tmp+1+n,cmp);
 for (i=1;i<=n;i++) a[i]=tmp[i].b;//,printf("%.3lf\n",a[i]);
 divide(1,n);
 printf("%d\n",ans);
 //for (i=1;i<=n;i++) printf("%.3lf\n",a[i]);
 return 0;
}
```

9. Iroha and a Grid

【问题描述】

有一个H*W 的矩阵, 现在你正位于左上角的格子, 并且你只能向右移动或向下移动, 不幸的是, 矩阵的左下角A*B 的地方被划为了禁区, 即你不能在此行走, 那么现在你有多少种方法从左上角走到右下角的格子呢?

$$1 \leqq H, W \leqq 100,000$$

【链接】 https://www.luogu.com.cn/problem/AT1974

【输入格式】

一行四个整数H,W,A,B

【输出格式】

方案数,由于方案数很大,请对10^9+7 取模

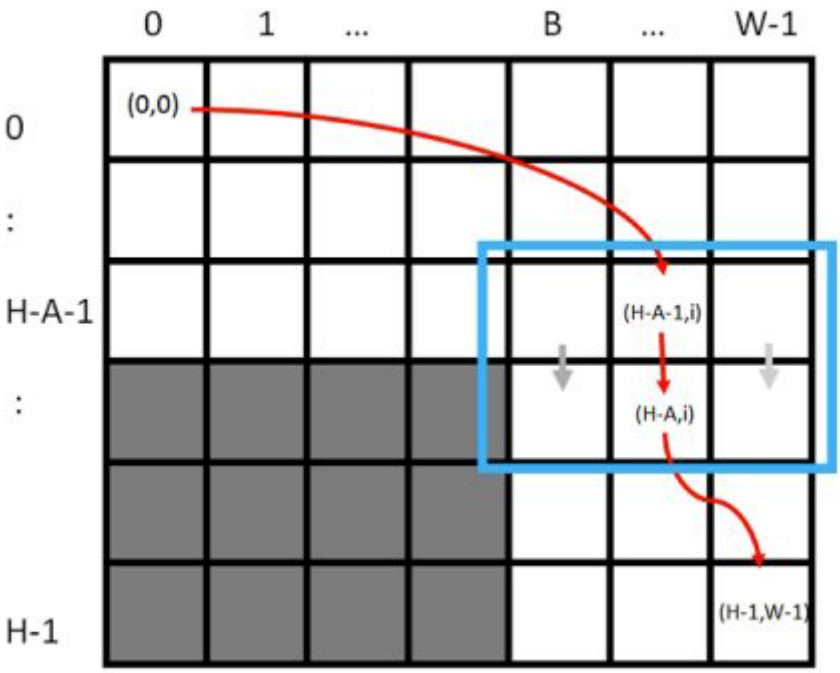
【样例输入】

2 3 1 1

【样例输出】

2

题解



如果没有A和B的限制，那么可以直接用组合数来计算答案。

否则把路径分成两部分，枚举一下这两部分跨越时候的位置即可。

C(h-a-2+i,i-1)*C(a-1+w-i,w-i)

代码

```
#include <iostream>
#include <cstdio>
#include <algorithm>
#include <queue>
#include <string>
using namespace std;
typedef long long ll;
const int N=200010;
```

```cpp
const int mod=1e9+7;

int h,w,a,b;
int fac[N],fnv[N];
ll ans;
ll quickmod(ll x,ll y)
{
    ll ans=1;
    for(;y;y>>=1)
    {
        if(y&1) ans=ans*x%mod;
        x=x*x%mod;
    }
    return ans;
}

void build()
{
    fac[0]=1;
    for(int i=1;i<N;i++) fac[i]=1ll*fac[i-1]*i%mod;
    fnv[N-1]=quickmod(fac[N-1],mod-2);
    for(int i=N-2;i>=0;i--) fnv[i]=1ll*fnv[i+1]*(i+1)%mod;
}

ll C(int x,int y)
{
    if(x<y) return 0;
    return 1ll*fac[x]*fnv[y]%mod*fnv[x-y]%mod;
}

int main()
{
    build();
    scanf("%d%d%d%d",&h,&w,&a,&b);
    for(int i=b+1;i<=w;i++)
    {
        ans=(ans+C(h-a-2+i,i-1)*C(a-1+w-i,w-i)%mod)%mod;
    }
    cout<<ans<<endl;
    return 0;
}


```

10. Inversion

【问题描述】

给出一个1-n的排列，有m次操作，每个操作给出一对l，r，表示将[l,r]区间reverse，问每次操作之后得到的排列的逆序对数是奇数还是偶数。

【输入】

第一行包含一个整数n（1 ⩽n⩽ 1500） — 排列的长度。

第二行包含n个数，表示排列。

第三行包含一个整数m(1 ⩽m ⩽ 2·10^5） — 要处理的操作数。

然后跟随 m行，第i行包含两个整数l,r 表示第i个操作是要反转[l，r]的排列。

【输出】

如果逆序对数是奇数打印odd，偶数打印even

【链接】https://codeforces.com/problemset/problem/911/D

【输入】

3

1 2 3

2

1 2

2 3

【输出】

odd

even

【输入】

4

1 2 4 3

4

1 1

1 4

1 4

2 3

【输出】

odd

odd

odd

even

题解

题意：给出一个长度为n的序列，每次翻转一段区间，问每次操作之后的逆序对数是奇数还是偶数。

这题很简单，只需要考虑奇偶性不需要统计个数。

每次翻转的一段区间pair的个数如果是偶数，那么逆序对数的奇偶性不变，否则一定改变。

代码

```
#include<bits/stdc++.h>
using namespace std;
int i,n,j,ans,a[2010],o,p,q;
int main()
{
 scanf("%d",&n);
 for (i=1;i<=n;i++) scanf("%d",&a[i]);
 for (i=1;i<=n;i++) for (j=i+1;j<=n;j++) if (a[i]>a[j]) ans^=1;
 scanf("%d",&q);
 while (q--)
        {
  scanf("%d%d",&o,&p);
  if (((p-o+1)*(p-o)/2)&1) ans^=1;
  if (ans&1) printf("odd\n");else printf("even\n");
        }
}
```