

# 2020牛客暑期多校训练营（第六场）

[比赛网址](#)

## 训练结果

- 时间:2020-07-27 12:00~17:00
- rank:36/1019
- 完成情况 : 7/8/11

## 题解

### A.African Sort

#### 题意

给定排列  $p$  每次可以选一个下标集合等概率打乱包含的数并花费集合大小的代价，求给  $p$  排升序最优策略下最小代价的期望，对  $998244353$  取模

#### 题解

首先题解是说一定是每一个环内进行重排会是最优（我也不知道这是怎么想到的）

那么就推长度为  $n$  的环的期望步数，先进行一次重整，有  $n!$  种可能，这  $n!$  种可能中必定会出现一些大小更小的环，我们先考虑在  $n!$  的可能中，产生长度为  $i$  的环的次数，是  $(n-i)!(i-1)!$  种，于是根据贡献，我们得到了  $f[n]=\frac{\sum_{i=2}^n (n-i)!(i-1)!f[i]}{n!}+n$  然后处理前缀和即可。

但是!!!为什么那么操作就一定是最优的?出题人也表示不知道,所以这可能是一个错题。

```
#include<bits/stdc++.h>
using namespace std;
#define mem(a,b) memset(a,b,sizeof(a))
typedef long long LL;
typedef pair<int,int> PII;
#define X first
#define Y second
inline int read()
{
    int x=0,f=1;char c=getchar();
    while(!isdigit(c)){if(c=='-')f=-1;c=getchar();}
    while(isdigit(c)){x=x*10+c-'0';c=getchar();}
    return x*f;
}
```

```
const int MOD=998244353;
const int maxn=100010;
int n,to[maxn],size;
bool vis[maxn];
LL f[maxn];
LL quickpow(int a,int N)
{
    LL res=1,tmp=a;
    while(N)
    {
        if(N&1)res=(res*tmp)%MOD;
        tmp=(tmp*tmp)%MOD;
        N>>=1;
    }
    return res;
}
LL inv(LL a){return quickpow(a,MOD-2);}
int main()
{
    f[1]=0;f[2]=4;
    for(int i=3;i<=100000;i++)f[i]=(LL)i*(f[i-1]+1)%MOD*inv(i-1)%MOD;
    n=read();
    int T=read();
    while(T--)
    {
        LL ans=0;
        for(int i=1;i<=n;i++)to[i]=read(),vis[i]=0;
        for(int i=1;i<=n;i++)
            if(!vis[i])
            {
                int now=i,size=1;
                vis[now]=1;
                while(!vis[to[now]])
                    vis[to[now]]=1,now=to[now],size++;
                ans=(ans+f[size])%MOD;
            }
        printf("%lld\n",ans);
    }
    return 0;
}
```

## B.Binary Vector

### 题意

随机选取 $n$ 个二进制向量，问线性独立的概率

## 题解

对于每加到第 $i$ 个向量时，有 $2^{n-2^i}$ 个向量线性无关，每次乘上概率 $P = \frac{2^{n-2^i}}{2^n}$ 即可

```

#include<algorithm>
#include<iostream>
#include<cstdlib>
#include<cstring>
#include<cstdio>
#include<vector>
#include<queue>
#include<cmath>
#include<map>
#include<set>
#define LL long long int
#define REP(i,n) for (int i = 1; i <= (n); i++)
#define Redge(u) for (int k = h[u],to; k; k = ed[k].nxt)
#define cls(s,v) memset(s,v,sizeof(s))
#define mp(a,b) make_pair<int,int>(a,b)
#define cp pair<int,int>
using namespace std;
const int maxn = 20000005,maxm = 100005,INF = 0x3f3f3f3f,P = 1000000007;
inline int read(){
    int out = 0,flag = 1; char c = getchar();
    while (c < 48 || c > 57){if (c == '-') flag = 0; c = getchar();}
    while (c >= 48 && c <= 57){out = (out << 1) + (out << 3) + c - 48; c =
getchar();}
    return flag ? out : -out;
}
int n,p1[maxn],p2[maxn],N = 20000000;
int f[maxn];
void init(){
    p1[0] = 1;
    for (int i = 1; i <= N; i++) p1[i] = 1ll * p1[i - 1] * 2 % P;
    p2[0] = 1; p2[1] = 500000004;
    for (int i = 2; i <= N; i++) p2[i] = 1ll * p2[i - 1] * p2[1] % P;
    f[1] = 500000004;
    int u = 1,d = 500000004;
    for (int i = 2; i <= N; i++){
        u = 1ll * u * ((1ll * p1[i] - 1 + P) % P) % P;
        d = 1ll * d * p2[i] % P;
        f[i] = 1ll * u * d % P;
        f[i] = f[i] ^ f[i - 1];
    }
}
int main(){
    init();
    int T = read();
    while (T--){

```

```
printf("%d\n", f[read()]);  
}  
return 0;  
}
```

## C. Combination of Physics and Maths

### 题意

给了一个  $n \times m$  的矩阵，让你选一个子矩阵，使得子矩阵的总权值和除以最下面一行的和最大

### 题解

答案肯定只有一列，而且要尽可能多的往上选，所以我们枚举每个点，把上面的全选了，求最大值即可。

```
#include<iostream>  
#include<cstdio>  
#include<algorithm>  
#include<cstring>  
#define ll long long  
using namespace std;  
int read()  
{  
    int k=0,f=1;char c=getchar();  
    for(;!isdigit(c);c=getchar()) if(c=='-') f=-1;  
    for(;isdigit(c);c=getchar()) k=k*10+c-'0';return k*f;  
}  
const int N=205;  
int T,n,m,a[N][N];  
int main()  
{  
    for(T=read();T;T--)  
    {  
        long double ans=0,sum=0;  
        n=read();m=read();  
        for(int i=1;i<=n;i++)  
            for(int j=1;j<=m;j++)  
                a[i][j]=read();  
        for(int i=1;i<=m;i++)  
        {  
            sum=0;  
            for(int j=1;j<=n;j++)  
            {  
                sum+=a[j][i];  
                ans=max(ans,sum/(1.*a[j][i]));  
            }  
        }  
    }  
}
```

```

    }
}
printf("%.10Lf\n",ans);
}
return 0;
}

```

## E.Easy Construction

### 题意

构造一个长度为 $n$ 的排列，使得对于任意长度的连续区间都至少存在一个区间使得区间和模区间长度为 $k$

### 题解

先考虑当区间长度为 $n$ 的情况，那么就可以分奇偶进行讨论。

奇数：182736459

偶数：84172635

```

#include<bits/stdc++.h>
using namespace std;
#define mem(a,b) memset(a,b,sizeof(a))
typedef long long LL;
typedef pair<int,int> PII;
#define X first
#define Y second
inline int read()
{
    int x=0,f=1;char c=getchar();
    while(!isdigit(c)){if(c=='-')f=-1;c=getchar();}
    while(isdigit(c)){x=x*10+c-'0';c=getchar();}
    return x*f;
}
const int maxn=5010;
int n,k;
int main()
{
    n=read();k=read();
    if((n*(n+1)/2)%n!=k)
    {
        puts("-1");
        return 0;
    }
    if(n%2)

```

```
{
    for(int i=1;i<n;i+=2)printf("%d %d ",i,n-i);
    printf("%d\n",n);
    return 0;
}
else
{
    printf("%d %d",n,n/2);
    for(int i=1;i<n/2;i++)printf(" %d %d",i,n-(i));
    puts("");
    return 0;
}
return 0;
}
```

## G.Grid Coloring

### 题意

给一个  $n \times n$  的网格所有短边使用  $k$  种颜色上色，满足以下条件：

- 1、每种颜色使用次数相同
- 2、不存在一个同色的回路
- 3、不存在任意一条长边是同色的

### 题解

先将每个网格的左边和上边涂成一种颜色，视作该网格的颜色，保证相邻网格是异色的，发现这样一定不存在回路，且最后会剩下最下面一排和最右边一排，在此基础上只需保证这两排相邻的边不重色即可满足题目条件。

注意特判  $k$  和  $n$  等于 1 的情况！

```
#include<algorithm>
#include<iostream>
#include<cstdlib>
#include<cstring>
#include<cstdio>
#include<vector>
#include<queue>
#include<cmath>
#include<map>
#include<set>
#define LL long long int
#define REP(i,n) for (int i = 1; i <= (n); i++)
```

```

#define Redge(u) for (int k = h[u],to; k; k = ed[k].nxt)
#define cls(s,v) memset(s,v,sizeof(s))
#define mp(a,b) make_pair<int,int>(a,b)
#define cp pair<int,int>
using namespace std;
const int maxn = 205,maxm = 100005,INF = 0x3f3f3f3f;
inline int read(){
    int out = 0,flag = 1; char c = getchar();
    while (c < 48 || c > 57){if (c == '-') flag = 0; c = getchar();}
    while (c >= 48 && c <= 57){out = (out << 1) + (out << 3) + c - 48; c =
getchar();}
    return flag ? out : -out;
}
struct node{
    int cnt,i;
}tmp[maxm];
bool operator <(const node& a,const node& b){
    return a.cnt == b.cnt ? a.i > b.i : a.cnt < b.cnt;
}
priority_queue<node> q;
int n,k,m,left[maxm],ti;
int h[maxn][maxn],v[maxn][maxn];
bool check(int i,int j,node u){
    return (u.cnt >= 2 && (i == 1 || u.i != h[i - 1][j]) && (j == 1 || u.i
!= v[i][j - 1]));
}
void work(){
    while (!q.empty()) q.pop();
    REP(i,k) q.push((node){m,i});
    int flag = true,x = 0,y = 0;
    for (int i = 1; i <= n; i++){
        for (int j = 1; j <= n; j++){
            ti = 0;
            while (!q.empty()){
                tmp[++ti] = q.top(); q.pop();
                if (check(i,j,tmp[ti])){
                    break;
                }
            }
            if (!check(i,j,tmp[ti])){
                flag = false; x = i; y = j;
                for (int t = 1; t <= ti; t++) if (tmp[t].cnt > 0)
q.push(tmp[t]);
                break;
            }
            else {
                v[i][j] = h[i][j] = tmp[ti].i; tmp[ti].cnt -= 2;
                for (int t = 1; t <= ti; t++) if (tmp[t].cnt > 0)
q.push(tmp[t]);
            }
        }
    }
}

```

```
        if (!flag) break;
    }
    if (!flag){
        for (int i = x; i <= n; i++){
            for (int j = (i == x ? y : 1); j <= n; j++){
                ti = 0;
                while (!q.empty()){
                    tmp[++ti] = q.top(); q.pop();
                    if (tmp[ti].i != h[i - 1][j]) break;
                }
                h[i][j] = tmp[ti].i; tmp[ti].cnt--;
                for (int t = 1; t <= ti; t++) if (tmp[t].cnt > 0)
q.push(tmp[t]);

                ti = 0;
                while (!q.empty()){
                    tmp[++ti] = q.top(); q.pop();
                    if (tmp[ti].i != v[i][j - 1]) break;
                }
                v[i][j] = tmp[ti].i; tmp[ti].cnt--;
                for (int t = 1; t <= ti; t++) if (tmp[t].cnt > 0)
q.push(tmp[t]);
            }
        }
    }
    //left sides
    for (int i = 1; i <= n; i++){
        ti = 0;
        while (!q.empty()){
            tmp[++ti] = q.top(); q.pop();
            if (tmp[ti].i != h[n + 1][i - 1]) break;
        }
        h[n + 1][i] = tmp[ti].i; tmp[ti].cnt--;
        for (int t = 1; t <= ti; t++) if (tmp[t].cnt > 0) q.push(tmp[t]);
    }
    v[n + 1][n + 1] = h[n + 1][n];
    for (int i = n; i; i--){
        ti = 0;
        while (!q.empty()){
            tmp[++ti] = q.top(); q.pop();
            if (tmp[ti].i != v[i + 1][n + 1]) break;
        }
        v[i][n + 1] = tmp[ti].i; tmp[ti].cnt--;
        for (int t = 1; t <= ti; t++) if (tmp[t].cnt > 0) q.push(tmp[t]);
    }
    //print
    for (int i = 1; i <= n + 1; i++){
        printf("%d",h[i][1]);
        for (int j = 2; j <= n; j++){
            printf(" %d",h[i][j]);
        }
    }
}
```

```

    }
    puts("");
}
for (int i = 1; i <= n + 1; i++){
    printf("%d",v[1][i]);
    for (int j = 2; j <= n; j++){
        printf(" %d",v[j][i]);
    }
    puts("");
}
}
int main(){
    int T = read();
    while (T--){
        n = read(); k = read();
        if (n == 1 || k == 1 || 2 * n * (n + 1) % k) {puts("-1");
continue;}
        m = 2 * n * (n + 1) / k;
        work();
    }
    return 0;
}

```

## H.Harmony Pairs

### 题意

让你求有多少点对  $(A,B)$  满足  $0 \leq A \leq B \leq N$  ,使  $A$  的各位上的和大于  $B$  各位上的和。

### 题解

$f[i][j][1/0]$  表示从  $i$  位往后所有位上和为  $j$  且高位是否有限制的数字的数目 dp 的时候当前为从大往小 dp 并同时用树状数组统计答案。

```

#include<iostream>
#include<cstdio>
#include<algorithm>
#include<cstring>
#include<set>
#define ll long long
using namespace std;
int read()
{
    int k=0,f=1;char c=getchar();
    for(;!isdigit(c);c=getchar()) if(c=='-') f=-1;
    for(;isdigit(c);c=getchar()) k=k*10+c-'0';return k*f;
}

```

```
}
const int mod=1e9+7;
char s[105];
set<int> st[105],st1[105];
int n,f[105][1005][2],vis[105][1005][2];
int ans[105][1005],sum[105][1005];
int ans2[105][1005],sum2[105][1005];
void up(int &x,int y)
{
    x=(x+y)%mod;
}
void add(int x,int y,int v)
{
    y+=1;
    // if(x==1) cout<<x<<" "<<y<<" "<<v<<endl;
    for(int i=y;i<=1001;i+=i&-i)
        up(sum[x][i],v);
}
int query(int x,int y)
{
    y+=1;
    if(y<=0) return 0;
    int res=0;
    for(;y;y-=y&-y)
        up(res,sum[x][y]);
    return res;
}
void add2(int x,int y,int v)
{
    y+=1;
    // if(x==1) cout<<x<<" "<<y<<" "<<v<<endl;
    for(int i=y;i<=1001;i+=i&-i)
        up(sum2[x][i],v);
}
int query2(int x,int y)
{
    y+=1;
    if(y<=0) return 0;
    int res=0;
    for(;y;y-=y&-y)
        up(res,sum2[x][y]);
    return res;
}
int main()
{
    scanf("%s",s+1);
    n=strlen(s+1);
    for(int i=1;i<=n;i++)
        s[i]-='0';
    set<int>::iterator it;
```

```

f[n+1][0][0]=f[n+1][0][1]=1;
st[n+1].insert(0);st1[n+1].insert(0);
for(int i=n;i;i--)
{
    for(it=st1[i+1].end(),it--;;it--)
    {
        up(f[i][s[i]+*it][1],f[i+1][*it][1]),st1[i].insert(s[i]+*it);
up(ans[i][s[i]+*it],1ll*f[i+1][*it][1]*query(i,s[i]+*it-1)%mod);
//    if(i==1) cout<<*it<<endl;
        add(i,s[i]+*it,f[i+1][*it][1]);
        if(it==st1[i+1].begin()) break;
    }
    for(int j=s[i]-1;j>=0;j--)
        for(it=st[i+1].end(),it--;;it--)
        {
            up(f[i][j+*it][1],f[i+1][*it][0]),st1[i].insert(j+*it);
            up(ans[i][j+*it],1ll*f[i+1][*it][0]*query(i,j+*it-1)%mod);
            add(i,j+*it,f[i+1][*it][0]);
            if(it==st[i+1].begin()) break;
        }
    for(int j=9;j>=0;j--)
        for(it=st[i+1].end(),it--;;it--)
        {
            up(f[i][j+*it][0],f[i+1][*it][0]),st[i].insert(j+*it);
up(ans2[i][j+*it],1ll*f[i+1][*it][0]*query2(i,j+*it-1)%mod);
            add2(i,j+*it,f[i+1][*it][0]);
            if(it==st[i+1].begin()) break;
        }
}
int now=0,as=0;
// for(int i=1;i<=n;i++)
for(int i=1;i<=n;i++)
{
    now=(1ll*now*10+s[i])%mod;
    for(int j=0;j<=1000;j++)
    {
        up(as,ans[i][j]);
        up(as,1ll*now*ans2[i+1][j]%mod);
    }
}
cout<<as<<endl;
return 0;
}

```

## J. Josephus Transform

### 题意

初始排列为 $1-n$ ，接下来有 $m$ 次操作，每次操作把当前序列进行 $k$ 约瑟夫环的删除顺序进行重排 $x$ 次，最后输出最终排列。

## 题解

对于每一个操作，先用线段树在 $O(n \log n)$ 时间内快速模拟把 $k$ 约瑟夫环的删除序列做出来，这个显然是一个置换，然后将置换倍增即可。

```
#include<bits/stdc++.h>
using namespace std;
#define mem(a,b) memset(a,b,sizeof(a))
typedef long long LL;
typedef pair<int,int> PII;
#define X first
#define Y second
inline int read()
{
    int x=0,f=1;char c=getchar();
    while(!isdigit(c)){if(c=='-')f=-1;c=getchar();}
    while(isdigit(c)){x=x*10+c-'0';c=getchar();}
    return x*f;
}
const int maxn=100010;
int n,m,x,k,here[maxn],To[maxn],T0[maxn],temp[maxn],a[32][maxn],now;
#define ls (u << 1)
#define rs (u << 1 | 1)
int sum[4 * maxn],to[maxn],pos,L,R;
void modify(int u,int l,int r){
    if (l == r) {sum[u] = 0; return;}
    int mid = (l + r) >> 1;
    if (mid >= pos) modify(ls,l,mid);
    else modify(rs,mid + 1,r);
    sum[u] = sum[ls] + sum[rs];
}
int query(int u,int l,int r){
    if (l >= L && r <= R) return sum[u];
    int mid = (l + r) >> 1;
    if (mid >= R) return query(ls,l,mid);
    else if (mid < L) return query(rs,mid + 1,r);
    return query(ls,l,mid) + query(rs,mid + 1,r);
}
int kth(int u,int l,int r,int k){
    if (l == r) return l;
    int mid = (l + r) >> 1;
    if (sum[ls] >= k) return kth(ls,l,mid,k);
    else return kth(rs,mid + 1,r,k - sum[ls]);
}
void build(int u,int l,int r){
```

```
if (l == r){
    sum[u] = 1;
    return;
}
int mid = (l + r) >> 1;
build(ls,l,mid);
build(rs,mid + 1,r);
sum[u] = sum[ls] + sum[rs];
}
int main()
{
    n=read();m=read();
    for(int i=1;i<=n;i++)here[i]=i;
    while(m--)
    {
        k=read();x=read();
        now = k;
        build(1,1,n);
        for (int i = 1; i <= n; i++)
        {
            pos = kth(1,1,n,now);
            to[pos] = i;
            if (i == n) break;
            modify(1,1,n);
            L = pos; R = n;
            int r = query(1,1,n);
            if (r >= k) now = now + k - 1;
            else {
                now = (k - r) % (n - i);
                if (!now) now = n - i;
            }
        }
        for(int i=1;i<=n;i++)a[0][i]=to[i];
        for(int Log=1;(1<<Log)<=x;Log++)
            for(int i=1;i<=n;i++)
                a[Log][i]=a[Log-1][a[Log-1][i]];
        for(int Log=0;(1<<Log)<=x;Log++)
            if(x&(1<<Log))
            {
                for(int i=1;i<=n;i++)temp[a[Log][i]]=here[i];
                for(int i=1;i<=n;i++)here[i]=temp[i];
            }
    }
    for(int i=1;i<n;i++)printf("%d ",here[i]);
    printf("%d\n",here[n]);
    return 0;
}
```

## K.K-Bag

### 题意

如果一个序列由若干个 $1$ 到 $k$ 的排列组成，那么称这个序列是 $k$ -bag。问一个给定序列是否是一个 $k$ -bag的子串

### 题解

讨论两种情况：

1.  $k > n$ ，那么这个序列只可能由左边一个和右边一个不完整的排列组成，扫一遍记录左边最远延伸的距离和右边最远延伸的距离即可

2.  $k \leq n$ 这个时候序列由中间若干个完整的排列加上左右不完整的排列，同样先记录左右最远延伸距离，同时用 $dp$ 计算每个位置为结束位置时中间完整排列的最长距离，判一下配合左右能否覆盖整个序列即可

```
#include<algorithm>
#include<iostream>
#include<cstdlib>
#include<cstring>
#include<cstdio>
#include<vector>
#include<queue>
#include<cmath>
#include<map>
#include<set>
#define LL long long int
#define REP(i,n) for (int i = 1; i <= (n); i++)
#define Redge(u) for (int k = h[u],to; k; k = ed[k].nxt)
#define cls(s,v) memset(s,v,sizeof(s))
#define mp(a,b) make_pair<int,int>(a,b)
#define cp pair<int,int>
using namespace std;
const int maxn = 500005,maxm = 100005,INF = 0x3f3f3f3f;
inline int read(){
    int out = 0,flag = 1; char c = getchar();
    while (c < 48 || c > 57){if (c == '-') flag = 0; c = getchar();}
    while (c >= 48 && c <= 57){out = (out << 1) + (out << 3) + c - 48; c =
getchar();}
    return flag ? out : -out;
}
int n,k;
int A[maxn],vis[maxn],cnt;
int tag[maxn],f[maxn];
int L,R;
```

```

set<int> S;
int main(){
    int T = read();
    while (T--){
        n = read(); k = read();
        REP(i,n) A[i] = read();
        if (k >= n){
            S.clear();
            L = 0;
            while (L + 1 <= n && !S.count(A[L + 1])) S.insert(A[++L]);
            S.clear();
            R = n + 1;
            while (R - 1 > 0 && !S.count(A[R - 1])) S.insert(A[--R]);
            puts(L >= R - 1 ? "YES" : "NO");
        }
        else {
            cnt = 0;
            REP(i,n) tag[i] = false;
            REP(i,k) vis[i] = false;
            L = 0;
            while (L + 1 <= n && !vis[A[L + 1]]) vis[A[++L]] = true;
            REP(i,k) vis[i] = false;
            R = n + 1;
            while (R - 1 > 0 && !vis[A[R - 1]]) vis[A[--R]] = true;
            REP(i,k) vis[i] = 0;
            int flag = false;
            for (int i = 1; i <= n; i++){
                if (!vis[A[i]]) cnt++;
                vis[A[i]]++;
                if (cnt == k) tag[i] = true;
                if (i >= k){
                    if (vis[A[i - k + 1]] == 1) cnt--;
                    vis[A[i - k + 1]]--;
                }
                if (tag[i]){
                    f[i] = i - k + 1;
                    if (i >= 2 * k && tag[i - k]) f[i] = min(f[i], f[i -
k]);
                    if (i >= R - 1 && f[i] <= L + 1) flag = true;
                }
            }
            puts(flag ? "YES" : "NO");
        }
    }
    return 0;
}

```

## 训练实况

开局fyh看E,wxg和hxm看K 12:33 fyh过E之前hxm写K wxg看C 12:39 hxm过K 12:46 wxg过C 12:46 fyhhxm发现B的规律,直接写结论 12:59 hxm过B,一起想G H之间 hxm想出G的构造 wxg在写H hxm和fyh调G 15:27 发现bug hxm过G 15:46 wxg过H 16:00 A放弃,想出J 16:39 wxg调过J

## 训练总结

wxg:本场过题较多,但是H题很早想出dp但统计答案一直没想清楚导致罚时较多。

hxm:本场前期过题速度较快,但中期两道题做的较慢,导致最后罚时并不是很理想,尤其是G题没有注意特殊情况,花费了很多时间与罚时

fyh:最后写J的时候倍增的数组开错,导致又RE了一发。。这也是罚时增加的原因。以后看队友代码的时候要多问。

From:  
<https://wiki.cvbbacm.com/> - CVBB ACM Team

Permanent link:  
[https://wiki.cvbbacm.com/doku.php?id=2020-2021:teams:die\\_java:front\\_page\\_summertrain7&rev=1596186345](https://wiki.cvbbacm.com/doku.php?id=2020-2021:teams:die_java:front_page_summertrain7&rev=1596186345)

Last update: 2020/07/31 17:05