错题集 1

1□Fake Maxpooling

链接

题意

给定一个 $n\times m$ 矩阵□$a_{i,j}=\text{lcm}(i,j)$□设 $f(i,j)=\{\max a_{x,y}|i\le x\lt i+k,j\le y\lt j+k\}$□求

\begin{equation}\sum_{i=1}^{n-k+1}\sum_{j=1}^{m-k+1}f(i,j)\end{equation}

题解

记忆化搜索求 gcd□两次单调队列维护最大值。

第一次单调队列维护每行长度为 k 的区间的最大值，第二次单调队利用 k 列每行长度为 k 的区间的最大值。

时间复杂度 $O(nm)$□

```
#include <iostream>
#include <cstdio>
#include <cstdlib>
#include <algorithm>
#include <string>
#include <sstream>
#include <cstring>
#include <cctype>
#include <cmath>
#include <vector>
#include <set>
#include <map>
#include <stack>
#include <queue>
#include <ctime>
#include <cassert>
#define _for(i,a,b) for(int i=(a);i<(b);++i)
#define _rep(i,a,b) for(int i=(a);i<=(b);++i)
#define mem(a,b) memset(a,b,sizeof(a))
using namespace std;
typedef long long LL;
inline int read_int(){
    int t=0;bool sign=false;char c=getchar();
    while(!isdigit(c)){sign|=c=='-';c=getchar();}
```

```
    while(isdigit(c)){t=(t<<1)+(t<<3)+(c&15);c=getchar();}
    return sign?-t:t;
}
inline LL read_LL(){
    LL t=0;bool sign=false;char c=getchar();
    while(!isdigit(c)){sign|=c=='-';c=getchar();}
    while(isdigit(c)){t=(t<<1)+(t<<3)+(c&15);c=getchar();}
    return sign?-t:t;
}
inline char get_char(){
    char c=getchar();
    while(c==' '||c=='\n'||c=='\r')c=getchar();
    return c;
}
inline void write(LL x){
    register char c[21],len=0;
    if(!x)return putchar('0'),void();
    if(x<0)x=-x,putchar('-');
    while(x)c[++len]=x%10,x/=10;
    while(len)putchar(c[len--]+48);
}
inline void space(LL x){write(x),putchar(' ');}
inline void enter(LL x){write(x),putchar('\n');}
const int MAXN=5005;
int a[MAXN][MAXN],b[MAXN][MAXN],que[MAXN];
int main()
{
    int n=read_int(),m=read_int(),k=read_int(),u,v;
    _rep(i,1,n)
    _rep(j,1,m){
        if(!a[i][j]){
            for(int k=1;k*i<=n&&k*j<=m;k++)
            a[i*k][j*k]=i*j*k;
        }
    }
    _rep(i,1,n){
        int front=1,tail=0;
        _for(j,1,k){
            while(front<=tail&&a[i][que[front]]<=a[i][j])front++;
            que[++tail]=j;
        }
        _rep(j,k,m){
            while(front<=tail&&j-que[front]>=k)front++;
            while(front<=tail&&a[i][que[front]]<=a[i][j])front++;
            que[++tail]=j;
            b[i][j]=a[i][que[front]];
        }
    }
    LL sum=0;
    _rep(j,k,m){
```

```
        int front=1,tail=0;
        _for(i,1,k){
            while(front<=tail&&b[que[front]][j]<=b[i][j])front++;
            que[++tail]=i;
        }
        _rep(i,k,n){
            while(front<=tail&&i-que[front]>=k)front++;
            while(front<=tail&&b[que[front]][j]<=b[i][j])front++;
            que[++tail]=i;
            sum+=b[que[front]][j];
        }
    }
    enter(sum);
    return 0;
}
```

2□Columns Swaps

链接

题意

给定一个 $2\times n$ 的表格，定义一次操作为交换同一列的上下两个元素。

问至少要经过多少次操作才能使表格两行均为 $1\sim n$ 的全排列。(如果无解，输出 -1)

题解

首先如果某个数字在表格中出现次数不为 2 必无解。

否则记录每个数的两次出现的行号和列号。

建图，每个列代表一个点。对图进行某种染色，若一个点染黑色代表该列要交换，若一个点染白色代表该列不要交换。

考虑每对相同数的行号。如果行号相同，则两个数所在列必须有一个交换，一个不交换，两列所代表点连一条黑边，代表这两点颜色不同。

如果行号不相同，则两个数所在列要么都交换，要么都不交换，两列所代表点连一条白边，代表这两点颜色相同。

对每个连通块进行染色，初始点颜色设为 0，与初始点不同的颜色设为 1。最后如果 0 色点多则 0 色为白色，否则 1 色为白色。

由于染色过程遇到已标记点会跳过，所以最后需要在染色后枚举边集确定每条边的两个点颜色是否满足条件。

```
#include <iostream>
#include <cstdio>
```

```cpp
#include <cstdlib>
#include <algorithm>
#include <string>
#include <sstream>
#include <cstring>
#include <cctype>
#include <cmath>
#include <vector>
#include <set>
#include <map>
#include <stack>
#include <queue>
#include <ctime>
#include <cassert>
#define _for(i,a,b) for(int i=(a);i<(b);++i)
#define _rep(i,a,b) for(int i=(a);i<=(b);++i)
#define mem(a,b) memset(a,b,sizeof(a))
using namespace std;
typedef long long LL;
inline int read_int(){
    int t=0;bool sign=false;char c=getchar();
    while(!isdigit(c)){sign|=c=='-';c=getchar();}
    while(isdigit(c)){t=(t<<1)+(t<<3)+(c&15);c=getchar();}
    return sign?-t:t;
}
inline LL read_LL(){
    LL t=0;bool sign=false;char c=getchar();
    while(!isdigit(c)){sign|=c=='-';c=getchar();}
    while(isdigit(c)){t=(t<<1)+(t<<3)+(c&15);c=getchar();}
    return sign?-t:t;
}
inline char get_char(){
    char c=getchar();
    while(c==' '||c=='\n'||c=='\r')c=getchar();
    return c;
}
inline void write(LL x){
    register char c[21],len=0;
    if(!x)return putchar('0'),void();
    if(x<0)x=-x,putchar('-');
    while(x)c[++len]=x%10,x/=10;
    while(len)putchar(c[len--]+48);
}
inline void space(LL x){write(x),putchar(' ');}
inline void enter(LL x){write(x),putchar('\n');}
const int MAXN=2e5+5;
struct Edge{
    int to,type,next;
}edge[MAXN<<1];
int head[MAXN],edge_cnt;
```

```
void Insert(int u,int v,int type){
    edge[++edge_cnt]=Edge{v,type,head[u]};
    head[u]=edge_cnt;
}
vector<int> r[MAXN],c[MAXN];
int block_id[MAXN],block_cnt,color[MAXN],color_cnt[MAXN][2];
void dfs(int u,int c){
    block_id[u]=block_cnt;color[u]=c;
    color_cnt[block_cnt][c]++;
    for(int i=head[u];i;i=edge[i].next){
        int v=edge[i].to;
        if(block_id[v])
        continue;
        dfs(v,c^edge[i].type);
    }
}
int main()
{
    int t=read_int();
    while(t--){
        mem(block_id,0);block_cnt=0;
        mem(head,0);edge_cnt=0;
        mem(color_cnt,0);
        int n=read_int(),a;
        _rep(i,1,n)
        r[i].clear(),c[i].clear();
        _rep(i,1,n){
            a=read_int();
            c[a].push_back(i);
            r[a].push_back(1);
        }
        _rep(i,1,n){
            a=read_int();
            c[a].push_back(i);
            r[a].push_back(2);
        }
        bool flag=true;
        _rep(i,1,n){
            if(c[i].size()>2){
                puts("-1");
                flag=false;
                break;
            }
        }
        if(!flag)
        continue;
        _rep(i,1,n){
            Insert(c[i][0],c[i][1],r[i][0]==r[i][1]);
            Insert(c[i][1],c[i][0],r[i][0]==r[i][1]);
        }
        int ans=0;
```

```
        _rep(i,1,n){
            if(!block_id[i]){
                block_cnt++;
                dfs(i,0);
                ans+=min(color_cnt[block_cnt][0],color_cnt[block_cnt][1]);
            }
        }
        for(int i=1;i<edge_cnt;i+=2){
            if(color[edge[i].to]^color[edge[i+1].to]^edge[i].type){
                puts("-1");
                flag=false;
                break;
            }
        }
        if(!flag)
        continue;
        enter(ans);
        _rep(i,1,n){
if(color[i]^(color_cnt[block_id[i]][0]<color_cnt[block_id[i]][1]))
            space(i);
        }
        puts("");
    }
    return 0;
}
```