2020牛客暑期多校训练营（第十场）

比赛情况

题号	A	B	C	D	E	F	G	H	I	J
状态	O	-	-	Ø	O	-	-	-	!	O

O 在比赛中通过 Ø 赛后通过 ! 尝试了但是失败了 - 没有尝试

比赛时间

2020-08-10 12:00-17:00

题解

D - Hearthstone Battleground

容易想到要贪心的用植物去破盾，所以优先用有亡语的随从生成植物去破盾，注意每次攻击前要清除对方的植物。

```
/*
#pragma GCC optimize(2)
#pragma GCC optimize(3,"Ofast","inline")
*/
#include<bits/stdc++.h>
#define ALL(x) (x).begin(),(x).end()
#define ll long long
#define db double
#define ull unsigned long long
#define pii_ pair<int,int>
#define mp_ make_pair
#define pb push_back
#define fi first
#define se second
#define rep(i,a,b) for(int i=(a);i<=(b);i++)
#define per(i,a,b) for(int i=(a);i>=(b);i--)
#define show1(a) cout<<#a<<" = "<<a<<endl
#define show2(a,b) cout<<#a<<" = "<<a<<"; "<<#b<<" = "<<b<<endl
using namespace std;
const ll INF = 1LL<<60;
const int inf = 1<<30;
const int maxn = 2e5+5;
inline void fastio() {ios::sync_with_stdio(false);cin.tie(0);cout.tie(0);}

int a[5],b[5];
void act1()
{
```

```
    if(b[3]) b[3]--,b[0]++,a[3]++,a[1]--;
    else if(b[4]) b[4]--,a[3]++,a[1]--;
    else if(b[1]) b[1]--,b[3]++,a[3]++,a[1]--;
    else if(b[2]) b[2]--,b[4]++,a[3]++,a[1]--;
}
void act2()
{
    if(b[3]) b[3]--,b[0]++,a[2]--,a[4]++;
    else if(b[4]) b[4]--,a[2]--,a[4]++;
    else if(b[1]) b[1]--,b[3]++,a[2]--,a[4]++;
    else if(b[2]) b[2]--,b[4]++,a[2]--,a[4]++;
}
void act4()
{
    if(b[3]) b[3]--,b[0]++,a[4]--;
    else if(b[4]) b[4]--,a[4]--;
    else if(b[1]) b[1]--,b[3]++,a[4]--;
    else if(b[2]) b[2]--,b[4]++,a[4]--;
}
void act3()
{
    if(b[3]) b[3]--,a[0]++,b[0]++,a[3]--;
    else if(b[4]) b[4]--,a[0]++,a[3]--;
    else if(b[1]) b[1]--,b[3]++,a[0]++,a[3]--;
    else if(b[2]) b[2]--,b[4]++,a[0]++,a[3]--;
}
void act0()
{
    if(b[1]) b[1]--,b[3]++,a[0]--;
    else if(b[2]) b[2]--,b[4]++,a[0]--;
}
bool win()
{
    int aa = a[1] + a[2] + a[3] + a[4];
    int bb = b[1] + b[2] + b[3] + b[4];
    if(bb==0) {
        if(aa>0 || a[0]>b[0]) return 1;
    }
    return 0;
}
int main()
{
    fastio();
    int _;
    for(cin>>_;_;_--){
        memset(a,0,sizeof(a));
        memset(b,0,sizeof(b));
        rep(i,1,4) cin>>a[i];
        rep(i,1,4) cin>>b[i];
        while(a[1]+a[2]+a[3]+a[4] && b[1]+b[2]+b[3]+b[4]){
```

```
            if(a[3]){
                b[0] = 0;
                act3();act0();
            }else if(a[1]){
                b[0] = 0;
                act1();
            }else if(a[4]){
                b[0] = 0;
                act4();
            }else if(a[2]){
                b[0] = 0;
                act2();
            }
        }
        //rep(i,0,4) show1(a[i]);
        //rep(i,0,4) show1(b[i]);
        if(win()) cout<<"Yes"<<endl;
        else cout<<"No"<<endl;
    }
    return 0;
}
```

E - Game

可以证明如果一个形状的右侧都是小于等于某一高度的时候可以保证左侧的高于某一高度的砖块可以被填进坑里，所以我们可以二分答案，然后贪心验证即可。

```
#include <bits/stdc++.h>
using namespace std;
const int N = 1e5+5;
typedef long long ll;
int a[N],n;
ll sum[N];
bool check(int hei) {
    ll rem = 0;
    for (int i = n;i>= 1;i--) {
        if (a[i]>hei)rem+=a[i]-hei;
        if (a[i]<hei)rem=max(0ll,rem-hei+a[i]);
        if (rem != 0 && rem > (ll)hei*(i-1)-sum[i-1])return false;
    }
    return true;
}
int main() {
    int cas;
    scanf("%d",&cas);
    while (cas--) {
        scanf("%d",&n);
```

```
        for (int i = 1;i<= n;i++)
        {
            scanf("%d",&a[i]);
            sum[i] = sum[i-1]+a[i];
        }
        int l = 1,r = 1e9+5;
        while (l<r) {
            int mid = (l+r)>>1;
            if (check(mid))r = mid;
            else l = mid+1;
        }
        printf("%d\n",l);
    }
    return 0;
}
```

J - Identical Trees

假设我们已经知道了两个树使任意两个子树标号相同的编辑代价，我们可以将两两之间同构的子树进行匹配，然后求一个最小完美匹配代价。于是我们可以记录每一个子树的树高，从树高小的开始算编辑代价，然后一层一层转移即可。其中判断树是否同构用树hash□其余部分就是一个用费用流转移的dp

```
#include <bits/stdc++.h>
using namespace std;
const int N = 505;
int pri[N],pcnt;
bool visp[100005];
int f[N][N],n;
void initprime() {
    for (int i = 2;pcnt<500;i++)
    {
        if (!visp[i]) pri[++pcnt] = i;
        for (int j = 2;i*j < 1e5;j++)
            visp[i*j] = true;
    }
}
struct Edge {
    int nxt,to;
};
struct Tree {
    int head[N],tot;
    Edge e[N<<1];
    void init() {
        memset(head,0,sizeof(head));
        tot = 0;
    }
```

```
    void add(int x,int y) {
        e[++tot].nxt = head[x];head[x] = tot;e[tot].to = y;
        e[++tot].nxt = head[y];head[y] = tot;e[tot].to = x;
    }
    unsigned int thash[N];
    int size[N],thei[N],fa[N];
    void dfs(int x,int f) {
        fa[x] = f;
        thash[x] = 1;
        thei[x] = 1;
        size[x] = 1;
        for (int i = head[x];i;i=e[i].nxt)if(e[i].to!=fa[x]) {
            dfs(e[i].to,x);
            thei[x] = max(thei[x],thei[e[i].to]+1);
            thash[x] += thash[e[i].to]*pri[size[e[i].to]];
            size[x]+=size[e[i].to];
        }
    }
}t1,t2;
bool cmp(const int &a,const int &b) {
    return t1.thei[a] < t1.thei[b];
}
int id[N];
struct Max_Flow{
    int
head[N<<1],last[N<<1],pre[N<<1],vis[N<<1],dis[N<<1],flow[N<<1],s,t,tot;
    struct edge {int u,v,w,c,nxt;}es[502010];
    void addedge(int u,int v,int w,int c)
    {
        es[++tot] = (edge){u,v,w,c,head[u]};
        head[u] = tot;
    }
    void add(int u,int v,int w,int c) {addedge(u,v,w,c);addedge(v,u,0,-c);}
    void init(int mx) {
        for (int i = 0;i <= mx+1;i++)head[i] = -1;
        tot = -1;
        s = 0,t = mx+1;
    }
    bool spfa(int mx)
    {
        for (int i = 0;i<= mx+1;i++)dis[i] = flow[i] = 0x3f3f3f3f,vis[i] =
0;
        queue<int> q;q.push(s);
        vis[s] = 1,dis[s] = 0,pre[t] = -1;
        while(!q.empty()){
            int u = q.front();q.pop();
            vis[u] = 0;
            for(int i=head[u];~i;i=es[i].nxt){
                int v = es[i].v,w = es[i].w,c = es[i].c;
                if(w>0 && dis[v] > dis[u] + c){
                    dis[v] = dis[u] + c;
```

```
                    pre[v] = u;
                    last[v] = i;
                    flow[v] = min(flow[u],w);
                    if(!vis[v]){
                        vis[v] = 1;
                        q.push(v);
                    }
                }
            }
        }
        return ~pre[t];
    }
    pair<int,int> MCMF(int mx)
    {
        pair<int,int> res = make_pair(0,0);
        while(spfa(mx)){
            int now = t;
            res.first += flow[t];
            res.second += flow[t] * dis[t];
            while(now != s){
                es[last[now]].w -= flow[t];
                es[last[now]^1].w += flow[t];
                now = pre[now];
            }
        }
        return res;
    }
}mcmf;
void calc(int x,int y) {
    vector<int>sonx,sony;
    sonx.clear();sony.clear();
    for (int i = t1.head[x];i;i=t1.e[i].nxt) if (t1.e[i].to!=t1.fa[x]) {
        sonx.push_back(t1.e[i].to);
    }
    for (int i = t2.head[y];i;i=t2.e[i].nxt) if (t2.e[i].to!=t2.fa[y]) {
        sony.push_back(t2.e[i].to);
    }
    mcmf.init(sonx.size()+sony.size());
    for (int i = 0;i < sonx.size();i++)
        for (int j = 0;j < sony.size();j++) {
            int sx = sonx[i],sy = sony[j];
            if (f[sx][sy]!=-1) {
                mcmf.add(i+1,j+1+sonx.size(),1,f[sx][sy]);
            }
        }
    for (int i = 0;i < sonx.size();i++) mcmf.add(0,i+1,1,0);
    for (int i = 0;i < sony.size();i++)
mcmf.add(i+1+sonx.size(),sonx.size()+sony.size()+1,1,0);
    f[x][y] = (x!=y) + mcmf.MCMF(sonx.size()+sony.size()).second;
}
```

```
int main() {
    int x;
    scanf("%d",&n);
    t1.init();
    t2.init();
    initprime();
    int root1=1,root2=1;
    for (int i = 1;i<= n;i++) {
        scanf("%d",&x);
        if (x!=0)t1.add(x,i);
        else root1 = i;
    }
    for (int i = 1;i<= n;i++) {
        scanf("%d",&x);
        if (x!=0)t2.add(x,i);
        else root2 = i;
        id[i] = i;
    }
    t1.dfs(root1,0);
    t2.dfs(root2,0);
    sort(id+1,id+n+1,cmp);
    for (int i = 1;i<= n;i++)
        for (int j = 1;j<= n;j++)
        {
            if (t1.thei[i]==1 && t2.thei[j]==1)
                f[i][j] = (i!=j);
            else f[i][j] = -1;
        }
    for (int i = 1;i<= n;i++)
    {
        int x = id[i];
        if (t1.thei[x]!=1)
            for (int j = 1;j<= n;j++) {
                int y = j;
                if (t2.thei[y] == t1.thei[x] && t1.thash[x] == t2.thash[y])
                    calc(x,y);
            }
    }
    printf("%d\n",f[root1][root2]);
    return 0;
}
```