比赛链接

补题情况

题目	蒋贤蒙	王赵安	王智彪
A	2	0	0
B	2	0	0
C	2	1	0
D	0	0	0
F	0	0	0
G	2	0	0
I	1	0	2
J	2	0	2

题解

A. A Math Challenge

题意

$$ \sum_{i=0}^n\sum_{1\le cj\le ai+b}i^pj^q $$

题解

设 $F(n)=\sum_{i=1}^n i^q$于是上式转化为

$$ \sum_{i=0}^ni^pF(\lfloor \frac {ai+b}c\rfloor) $$

由于 $F(n)$ 是 $q+1$ 次多项式，所以高斯消元可以暴力求出 $F(n)$ 的表达式。于是问题转化为计算 $\sum_{i=0}^ni^p(\lfloor \frac {ai+b}c\rfloor)^k(0\le k\le q)$

上式用万能欧几里得算法板子可以 $O\left(p^2q^2\log c\right)$题目正解是类欧几里得算法 $O\left((p+q)^3\log c\right)$不过卡卡常还是能过的。

```
const int mod=998244353,MAXK=53;
int C[MAXK][MAXK];
struct Node{
    int cntr,cntu,f[MAXK][MAXK];
    Node(int cntr=0,int cntu=0){
        this->cntr=cntr;
        this->cntu=cntu;
        mem(f,0);
    }
    Node operator * (const Node &b)const{
        static int px[MAXK],py[MAXK];
```

```
        static int b1[MAXK][MAXK],b2[MAXK][MAXK];
        Node c;
        int dx=cntr,dy=cntu;
        px[0]=py[0]=1;
        _for(i,1,MAXK)
        px[i]=1LL*px[i-1]*dx%mod;
        _for(i,1,MAXK)
        py[i]=1LL*py[i-1]*dy%mod;
        _for(i,0,MAXK)_rep(j,0,i){
            b1[i][j]=1LL*C[i][j]*px[i-j]%mod;
            b2[i][j]=1LL*C[i][j]*py[i-j]%mod;
        }
        c.cntr=(cntr+b.cntr)%mod;
        c.cntu=(cntu+b.cntu)%mod;
        _for(i,0,MAXK)_for(j,0,MAXK){
            c.f[i][j]=f[i][j];
            _rep(i2,0,i)_rep(j2,0,j)
c.f[i][j]=(c.f[i][j]+1LL*b.f[i2][j2]*b1[i][i2]%mod*b2[j][j2])%mod;
        }
        return c;
    }
};
Node quick_pow(Node n,int k){
    Node ans=Node(0,0);
    while(k){
        if(k&1)ans=ans*n;
        k>>=1;
        if(k)n=n*n;
    }
    return ans;
}
Node asgcd(int a,int b,int c,int n,Node su,Node sr){
    if(a>=c)
    return asgcd(a%c,b,c,n,su,quick_pow(su,a/c)*sr);
    int m=(1LL*a*n+b)/c;
    if(!m)
    return quick_pow(sr,n);
    else
    return quick_pow(sr,(c-b-1)/a)*su*asgcd(c,(c-
b-1)%a,a,m-1,sr,su)*quick_pow(sr,n-(1LL*c*m-b-1)/a);
}
Node cal(int a,int b,int c,int n){
    Node su=Node(0,1),sr=Node(1,0);
    _for(i,0,MAXK)
    sr.f[i][0]=1;
    return quick_pow(su,b/c)*asgcd(a,b%c,c,n,su,sr);
}
int a[MAXK][MAXK],pw[MAXK][MAXK],A[MAXK];
void Init(){
    C[0][0]=1;
```

```
    _for(i,1,MAXK){
        C[i][0]=1;
        _rep(j,1,i)
        C[i][j]=(C[i-1][j-1]+C[i-1][j])%mod;
    }
    _for(i,0,MAXK){
        pw[i][0]=1;
        _for(j,1,MAXK)
        pw[i][j]=1LL*pw[i][j-1]*i%mod;
    }
}
int quick_pow(int n,int k){
    int ans=1;
    while(k){
        if(k&1)ans=1LL*ans*n%mod;
        n=1LL*n*n%mod;
        k>>=1;
    }
    return ans;
}
void build(int n){
    _for(i,0,n){
        _for(j,0,n)
        a[i][j]=pw[i][j];
        _rep(j,1,i)
        a[i][n]=(a[i][n]+pw[j][n-2])%mod;
    }
    _for(i,0,n){
        int pos=-1;
        _for(j,i,n){
            if(a[j][i]){
                pos=j;
                break;
            }
        }
        if(pos!=i)
        swap(a[i],a[pos]);
        int div=quick_pow(a[i][i],mod-2);
        _rep(j,i,n)
        a[i][j]=1LL*a[i][j]*div%mod;
        _rep(j,0,n){
            if(j==i)continue;
            for(int k=n;k>=i;k--)
            a[j][k]=(a[j][k]+mod-1LL*a[j][i]*a[i][k]%mod)%mod;
        }
    }
    _for(i,0,n)
    A[i]=a[i][n];
}
int main()
{
```

```
    Init();
    int
a=read_int(),b=read_int(),c=read_int(),p=read_int(),q=read_int(),n=read_int
();
    Node ans=cal(a,b,c,n);
    int base=1;
    _for(i,0,MAXK){
        ans.f[0][i]=(ans.f[0][i]+base)%mod;
        base=1LL*base*(b/c)%mod;
    }
    build(q+2);
    int s=0;
    _rep(i,0,q+1)
    s=(s+1LL*ans.f[p][i]*A[i])%mod;
    enter(s);
    return 0;
}
```

B. Best Subgraph

题意

定义 $k-\text{degree}$ 子图为每个点在子图中度数至少为 k 的连通极大子图。

定义每个 G 的子图 S 的分数为 $M\times m(S)-N\times n(S)+B\times b(S)$。

其中 $m(S)$ 表示 S 的边数，$n(S)$ 表示 S 的点数，$b(S)$ 表示 $|\{(u,v)|(u,v)\in G,u\in S,v\not\in S\}|$。

求分数最大的 $k-\text{degree}$ 子图，并求出分数最大的 $k-\text{degree}$ 子图中 k 的最大值。

输入保证图连通。

题解

首先，定义 $V(k)$ 表示所有 $k-\text{degree}$ 子图的并集，易知 $V(k+1)\subseteq V(k)$。

构建分层图，其中第 k 层的点集为 $V(k)-V(k+1)$，同时对每个点 $u\in V(k)-V(k+1)$，$w(u)=k$。

然后考虑按层加点，动态维护当前每个 $k-\text{degree}$ 子图的答案。对于每个新加入的点 u，首先他本身产生贡献 $-N$。

对于 u 的所有连边 (u,v)，如果 $w(v)\gt w(u)$，则 (u,v) 会被加入 $m(S)$，同时从 $b(S)$ 删除，产生贡献 $M-B$。

如果 $w(v)=w(u)$，则 (u,v) 也会被加入 $m(S)$，但为了贡献被计算两次，于是每个点的贡献为 $\frac M2$。

如果 $w(v)\lt w(u)$ 则 (u,v) 会被加入 $b(S)$ 产生贡献 B

于是上述过程可以将所有贡献都转化为每个点的点权。考虑加入点时并查集维护每个连通块的点权和。

每层点全部加完后查询每个连通块的点权和的最大值即为所有 $k-\text{degree}$ 子图的最大答案。

至于如果计算每个点的 $w(u)$ 首先输入保证图连通，所有图 G 本身就是 $1-\text{degree}$ 子图。

考虑先删去所有度数为 1 的点，删点后可能会导致原来度数不为 1 的点度数不大于 1，继续删除，直到无点可删，得到 $2-\text{degree}$ 子图。

将删去的点的 $w(u)$ 全部设为 1，然后再求 $3-\text{degree}$ 子图不断重复上述过程，即可得到所有 $w(u)$

时间复杂度 $O\left(\text{Na}(n+m)\right)$

```
const int MAXN=1e6+5;
const LL inf=1e18;
struct Edge{
    int to,next;
}edge[MAXN<<1];
int head[MAXN],edge_cnt;
void Insert(int u,int v){
    edge[++edge_cnt]=Edge{v,head[u]};
    head[u]=edge_cnt;
}
bool vis[MAXN];
vector<int> q[MAXN],vec[MAXN];
int deg[MAXN],w[MAXN],p[MAXN];
LL sum[MAXN];
int Find(int x){
    return x==p[x]?x:p[x]=Find(p[x]);
}
int main()
{
    int n=read_int(),m=read_int(),M=read_int(),N=read_int(),B=read_int();
    while(m--){
        int u=read_int(),v=read_int();
        Insert(u,v);
        Insert(v,u);
        deg[u]++;
        deg[v]++;
    }
    _rep(i,1,n)
    q[deg[i]].push_back(i);
    _for(k,1,MAXN){
        _for(j,0,q[k].size()){
            int u=q[k][j];
            if(vis[u])continue;
            vis[u]=true;
            w[u]=k;
```

```
            vec[k].push_back(u);
            for(int i=head[u];i;i=edge[i].next){
                int v=edge[i].to;
                deg[v]--;
                q[deg[v]].push_back(v);
            }
        }
    }
    _rep(u,1,n){
        p[u]=u;
        sum[u]=-N;
        for(int i=head[u];i;i=edge[i].next){
            int v=edge[i].to;
            if(w[u]<w[v]){
                sum[u]+=M;
                sum[u]-=B;
            }
            else if(w[u]==w[v]&&u<v)
            sum[u]+=M;
            else if(w[u]>w[v])
            sum[u]+=B;
        }
    }
    int st=MAXN-1;
    while(vec[st].empty())st--;
    pair<LL,int> ans=make_pair(-inf,0);
    for(int k=st;k;k--){
        for(int u:vec[k]){
            for(int i=head[u];i;i=edge[i].next){
                int v=edge[i].to;
                if(w[v]<k)continue;
                int x=Find(u),y=Find(v);
                if(x!=y){
                    p[x]=y;
                    sum[y]+=sum[x];
                }
            }
        }
        for(int u:vec[k])
        ans=max(ans,make_pair(sum[Find(u)],k));
    }
    space(ans.second);enter(ans.first);
    return 0;
}
```

C. Cells

题意

给定一个二维平面，求满足如下条件的 n 元路径组个数：

1. 第 i 条路径 $(a_i,0)\to (0,i)$
2. 每次移动只能选择 $(x,y)\to (x-1,y),(x,y+1)$

数据保证 $a_{i-1}\lt a_i$。

题解

显然有

$$ M= \begin{bmatrix} {a_1+1\choose 1}&{a_1+2\choose 2}&\cdots&{a_1+n\choose n}\\ {a_2+1\choose 1}&{a_2+2\choose 2}&\cdots&{a_2+n\choose n}\\ \vdots&\vdots&\ddots&\vdots\\ {a_n+1\choose 1}&{a_n+2\choose 2}&\cdots&{a_n+n\choose n}\\ \end{bmatrix} = \prod_{i=1}^n \frac 1{i!} \begin{bmatrix} \frac {(a_1+1)!}{a_1!}&\frac {(a_1+2)!}{a_1!}&\cdots&\frac {(a_1+n)!}{a_1!}\\ \frac {(a_2+1)!}{a_2!}&\frac {(a_2+2)!}{a_2!}&\cdots&\frac {(a_2+n)!}{a_2!}\\ \vdots&\vdots&\ddots&\vdots\\ \frac {(a_n+1)!}{a_n!}&\frac {(a_n+2)!}{a_n!}&\cdots&\frac {(a_n+n)!}{a_n!}\\ \end{bmatrix} $$

设 $x_i=a_i+1$，则

$$ M= \prod_{i=1}^n \frac 1{i!} \begin{bmatrix} x_1&x_1(x_1+1)&\cdots&\prod_{i=0}^{n-1}(x_1+i)\\ x_2&x_2(x_2+1)&\cdots&\prod_{i=0}^{n-1}(x_2+i)\\ \vdots&\vdots&\ddots&\vdots\\ x_n&x_n(x_n+1)&\cdots&\prod_{i=0}^{n-1}(x_n+i)\\ \end{bmatrix} $$

从左到右用列消元，可以得到

$$ M= \prod_{i=1}^n \frac 1{i!} \begin{bmatrix} x_1&x_1^2&\cdots&x_1^n\\ x_2&x_2^2&\cdots&x_2^n\\ \vdots&\vdots&\ddots&\vdots\\ x_n&x_n^2&\cdots&x_n^n\\ \end{bmatrix} $$

每行都提出一个 x_i 就可以得到一个范德蒙行列式，于是有

$$ \det M=\prod_{i=1}^n \frac {a_i+1}{i!}\prod_{1\le i\lt j\le n}(a_j-a_i) $$

考虑 NTT 计算每个值在 $\prod_{1\le i\lt j\le n}(a_j-a_i)$ 出现的次数。

具体的，可以设 $f(x)=\sum_{i=1}^n x^{a_i},g(x)=\sum_{i=1}^n x^{-a_i}$，则每个值 k 出现次数就是 $[x^k]f(x)g(x)$。

注意还有 $i\lt j$ 的限制，根据对称性，只考虑 $k\gt 0$ 的部分贡献然后做快速幂即可，时间复杂度 $O(V\log V)$。

```
const int mod=998244353,MAXN=1e6+5;
int quick_pow(int n,int k){
    int ans=1;
    while(k){
        if(k&1)ans=1LL*ans*n%mod;
```

```
        n=1LL*n*n%mod;
        k>>=1;
    }
    return ans;
}
namespace Poly{
    const int Mod=998244353,G=3;
    int rev[MAXN<<2],Wn[30][2];
    void init(){
        int m=Mod-1,lg2=0;
        while(m%2==0)m>>=1,lg2++;
        Wn[lg2][1]=quick_pow(G,m);
        Wn[lg2][0]=quick_pow(Wn[lg2][1],Mod-2);
        while(lg2){
            m<<=1,lg2--;
            Wn[lg2][0]=1LL*Wn[lg2+1][0]*Wn[lg2+1][0]%Mod;
            Wn[lg2][1]=1LL*Wn[lg2+1][1]*Wn[lg2+1][1]%Mod;
        }
    }
    int build(int k){
        int n,pos=0;
        while((1<<pos)<=k)pos++;
        n=1<<pos;
        _for(i,0,n)rev[i]=(rev[i>>1]>>1)|((i&1)<<(pos-1));
        return n;
    }
    void NTT(int *f,int n,bool type){
        _for(i,0,n)if(i<rev[i])
        swap(f[i],f[rev[i]]);
        int t1,t2;
        for(int i=1,lg2=0;i<n;i<<=1,lg2++){
            int w=Wn[lg2+1][type];
            for(int j=0;j<n;j+=(i<<1)){
                int cur=1;
                _for(k,j,j+i){
                    t1=f[k],t2=1LL*cur*f[k+i]%Mod;
                    f[k]=(t1+t2)%Mod,f[k+i]=(t1-t2)%Mod;
                    cur=1LL*cur*w%Mod;
                }
            }
        }
        if(!type){
            int div=quick_pow(n,Mod-2);
            _for(i,0,n)f[i]=(1LL*f[i]*div%Mod+Mod)%Mod;
        }
    }
    void mul(int *f,int _n,int *g,int _m){
        int n=build(_n+_m-2);
        _for(i,_n,n)f[i]=0;_for(i,_m,n)g[i]=0;
        NTT(f,n,1);NTT(g,n,1);
```

```
        _for(i,0,n)f[i]=1LL*f[i]*g[i]%Mod;
        NTT(f,n,0);
    }
}
int frac[MAXN],invf[MAXN],a[MAXN<<2],b[MAXN<<2];
int main(){
    frac[0]=1;
    _for(i,1,MAXN)
    frac[i]=1LL*frac[i-1]*i%mod;
    invf[MAXN-1]=quick_pow(frac[MAXN-1],mod-2);
    for(int i=MAXN-1;i;i--)
    invf[i-1]=1LL*invf[i]*i%mod;
    int n=read_int(),base=1e6,ans=1;
    _rep(i,1,n){
        int t=read_int();
        ans=1LL*ans*(t+1)%mod*invf[i]%mod;
        a[t]++;
        b[base-t]++;
    }
    Poly::init();
    Poly::mul(a,base+1,b,base+1);
    _rep(i,base+1,base*2)
    ans=1LL*ans*quick_pow(i-base,a[i])%mod;
    enter((ans+mod)%mod);
    return 0;
}
```

I. Incentive Model

题意

有两个人争夺 n 个物品，每一轮争夺一个物品，每次争夺都有成功概率。给定 x,y 代表初始双方的 $stake$ 分别为 ${\frac x y}$ 和 $1-{\frac x y}$ ，每次争夺双方成功概率为自己的 $stake$ 比双方相加的 $stake$ ，然后每一轮获胜者的 $stake$ 要加上 w ，问第一个人争夺成功轮数的期望，结果对 998244353 取模。

题解

我们设对于第一个人，第 i 轮获胜的概率为 X_{i} ，第 i 轮的 $stake$ 期望为 S_{i} 。

我们有 $S_{0}={\frac x y}$

然后我们有 $X_{i+1}={\frac {S_{i}} {1+w×i}}$ ， $S_{i+1}=S{i}+w×X_{i+1}$

所以有 $S_{i+1}=S_{i}+w×({\frac {S_{i}} {1+w×i}})=S_{i}×(1+{\frac w {1+w×i}})=S_{i}×{\frac {1+(i+1)×w} {1+i×w}}$

所以有 ${\frac {S_{i+1}} {1+(i+1)×w}}={\frac {S_{i}} {1+i×w}}=...={\frac {S_{0}}

{1+0×i}}={\frac x y}$

所以有 $S_{n}={\frac x y}×(1+n×w)$

又因为获胜轮数可以表示为 ${\frac {S_{n}-{\frac x y}} {w}}$

所以化简得到答案为 ${\frac x y}×n$

```cpp
#include <bits/stdc++.h>
#define int long long
using namespace std;
const int mod=998244353;
int qpow(int a,int b,int m=mod) {
    int r=1;
    while(b) {
        if (b&1) r=r*a%m;
        a=a*a%m,b>>=1;
    }
    return r;
}
int n,x,y,w;
#undef int

int main() {
    scanf("%lld %lld %lld %lld",&n,&w,&x,&y);
    printf("%lld\n",n*x%mod*qpow(y,mod-2)%mod);
    return 0;
}
```

J. Jam

题目链接

题意

现在有 N,W,S,E 四个方向，每个方向可以左拐右拐直走，然后会有一些互相撞车的情况，单位时间每个路线可以通过一辆车，我们要保证不能相撞，然后给出这十二个路来的车的量数，问最少多少时间能让所有的车都走完。

题解

由图显然右拐不用考虑，我们求出其他最长时间再分别和这四个数取 max 就可以了，所以只需要求出满足剩下八个的最小的时间就好了。

然后有很多不是最优的情况不用考虑，比如从 N 直走 和 从 N 直走+从 S 直走，前者的情况在任何情况下都不如后者显然不用考虑，于是我们就挑选最优的方案，可以找出十二种，最后的最优策略一定是这十二种的线性组合，我们分别设出他们的执行次数，然后就变成了八个不等式，每个不等式都是大于等

于的关系，然后有十二个未知数。跑一个单纯型就结束了，然后对应的哪个未知数，自己画个图就好了。

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

const double eps = 1e-8;
const int maxn=13,maxm=13;

int n, e, l, r;
int id[maxn+maxm],tmp[maxn];
double m[maxm][maxn],b[maxm],*c=m[0],ans[maxn+maxm];

void pivot(int r, int c) {
    int i, j;
    double coe = 1.0 / m[r][c];
    swap(id[n + r], id[c]);
    m[r][c] = 1.0;
    for(int j = 1; j <= n; ++j)
        m[r][j] *= coe;
    b[r] *= coe;
    for(int i = 0; i <= e; ++i) {
        if(i == r) continue;
        coe = m[i][c];
        m[i][c] = 0.0;
        for(j = 1; j <= n; ++j) m[i][j] -= coe * m[r][j];
        b[i] -= coe * b[r];
    }
}

bool simplex() {
    int bas, fr;
    double G;
    while(true) {
        bas = fr = 0;
        G = INFINITY;
        for(int i = 1; i <= n; ++i)
            if(c[i] > c[fr]) fr = i;
        if(!fr) return true;
        for(int j = 1; j <= e; ++j)
            if(m[j][fr] > eps && b[j] < G * m[j][fr]) {
                G = b[j] / m[j][fr];
                bas = j;
            }
        if(!bas) return false;
        pivot(bas, fr);
    }
}

int main() {
//    scanf("%d%d", &n, &e);
```

```
//    for(int i = 1; i <= n; i++) scanf("%lf", c + i);
//    for(int i = 1; i <= e; ++i) {
//        scanf("%d%d%lf", &l, &r, b + i);//从l到r 费用都是bi 然后让1~n的所有值都大于等于ci 且让费用最小
//        for(int j = l; j <= r; ++j)
//            m[i][j] = 1.0;
//    }
//    if(simplex()) {
//        printf("%.0f\n",-b[0]);
//    }
    int T;
    scanf("%d",&T);
    while(T--) {
        memset(id,0,sizeof(id));
        memset(b,0,sizeof(b));
        memset(c,0,sizeof(c));
        memset(ans,0,sizeof(ans));
        memset(m,0,sizeof(m));
        int cnt=0,cnt1=0;
        for(int i=1;i<=4;i++) {
            for(int j=1,ba;j<=4;j++) {
                if(i==j||(((j%4)+1)%4==(i%4))) {
                    if(i==j) scanf("%d",&ba);
                    else scanf("%d",&tmp[++cnt1]);
                    continue;
                }
                scanf("%lf",&c[++cnt]);
                //printf("%d %d\n",i,j);
            }
        }
        n=8,e=12;
        for(int i=1;i<=12;i++) b[i]=1;
        m[1][4]=m[1][8]=1.0;
        m[2][8]=m[2][1]=1.0;
        m[3][7]=m[3][5]=1.0;
        m[4][7]=m[4][3]=1.0;
        m[5][4]=m[5][6]=1.0;
        m[6][6]=m[6][1]=1.0;
        m[7][2]=m[7][5]=1.0;
        m[8][2]=m[8][3]=1.0;
        m[9][7]=m[9][8]=1.0;
        m[10][5]=m[10][6]=1.0;
        m[11][3]=m[11][4]=1.0;
        m[12][1]=m[12][2]=1.0;
        simplex();
        //printf("%.0f\n",-(b[0]-1e-8));
        int ans=(ceil)(-b[0]);
        for(int i=1;i<=cnt1;i++) {
            ans=max(ans,tmp[i]);
        }
```

```
        printf("%d\n",ans);
    }
    return 0;
}
```

当然这东西可能被卡？看了一眼题解，我们可以发现每一种方案都只对应两个情况，所以我们可以将他们代表的点连线然后我们发现这是一个多了两条边的二分图，枚举一下两条边选了多少次，剩下的跑网络流就好了。这个复杂度是 $O(n^{2}C),$ 其中 C 是网络流复杂度。

```
const int MAXN=12,MAXM=20,Inf=0x7fffffff;
struct Edge{
    int to,cap,next;
    Edge(int to=0,int cap=0,int next=0){
        this->to=to;
        this->cap=cap;
        this->next=next;
    }
}edge[MAXM<<1];
int head[MAXN],edge_cnt;
void Clear(){mem(head,-1);edge_cnt=0;}
void Insert(int u,int v,int c){
    edge[edge_cnt]=Edge(v,c,head[u]);
    head[u]=edge_cnt++;
    edge[edge_cnt]=Edge(u,0,head[v]);
    head[v]=edge_cnt++;
}
struct Dinic{
    int s,t;
    int pos[MAXN],vis[MAXN],dis[MAXN];
    bool bfs(int k){
        queue<int>q;
        q.push(s);
        vis[s]=k,dis[s]=0,pos[s]=head[s];
        while(!q.empty()){
            int u=q.front();q.pop();
            for(int i=head[u];~i;i=edge[i].next){
                int v=edge[i].to;
                if(vis[v]!=k&&edge[i].cap){
                    vis[v]=k,dis[v]=dis[u]+1,pos[v]=head[v];
                    q.push(v);
                    if(v==t)
                    return true;
                }
            }
        }
        return false;
    }
    int dfs(int u,int max_flow){
        if(u==t||!max_flow)
        return max_flow;
```

```
        int flow=0,temp_flow;
        for(int &i=pos[u];~i;i=edge[i].next){
            int v=edge[i].to;
if(dis[u]+1==dis[v]&&(temp_flow=dfs(v,min(max_flow,edge[i].cap)))){
                edge[i].cap-=temp_flow;
                edge[i^1].cap+=temp_flow;
                flow+=temp_flow;
                max_flow-=temp_flow;
                if(!max_flow)
                break;
            }
        }
        return flow;
    }
    int Maxflow(int s,int t){
        this->s=s;this->t=t;
        int ans=0,k=0;
        mem(vis,0);
        while(bfs(++k))
        ans+=dfs(s,Inf);
        return ans;
    }
}solver;
int c[4][4],idx[4][4];
int cal(){
    int ans=0,s=9,t=10;
    _for(i,0,4)
    ans+=c[i][(i+1)%4]+c[i][(i+2)%4];
    Clear();
    Insert(s,idx[0][1],c[0][1]);
    Insert(s,idx[1][3],c[1][3]);
    Insert(s,idx[2][0],c[2][0]);
    Insert(s,idx[3][0],c[3][0]);

    Insert(idx[0][1],idx[0][2],Inf);
    Insert(idx[0][1],idx[2][3],Inf);
    Insert(idx[0][1],idx[3][1],Inf);
    Insert(idx[1][3],idx[1][2],Inf);
    Insert(idx[1][3],idx[2][3],Inf);
    Insert(idx[1][3],idx[3][1],Inf);
    Insert(idx[2][0],idx[0][2],Inf);
    Insert(idx[2][0],idx[2][3],Inf);
    Insert(idx[3][0],idx[1][2],Inf);
    Insert(idx[3][0],idx[3][1],Inf);

    Insert(idx[0][2],t,c[0][2]);
    Insert(idx[1][2],t,c[1][2]);
    Insert(idx[2][3],t,c[2][3]);
    Insert(idx[3][1],t,c[3][1]);
    return ans-solver.Maxflow(s,t);
```

```
}
void solve(){
    int ans=Inf;
    _for(i,0,4){
        _for(j,0,4)
        c[i][j]=read_int();
    }
    _rep(i,0,min(c[0][2],c[1][2])){
        c[0][2]-=i;
        c[1][2]-=i;
        _rep(j,0,min(c[2][0],c[3][0])){
            c[2][0]-=j;
            c[3][0]-=j;
            ans=min(ans,cal()+i+j);
            c[2][0]+=j;
            c[3][0]+=j;
        }
        c[0][2]+=i;
        c[1][2]+=i;
    }
    _for(i,0,4)
    ans=max(ans,c[i][(i+3)%4]);
    enter(ans);
}
int main(){
    int cnt=0;
    _for(i,0,4){
        idx[i][(i+1)%4]=++cnt;
        idx[i][(i+2)%4]=++cnt;
    }
    int T=read_int();
    while(T--)
    solve();
    return 0;
}
```