A

B

给定一棵n个节点的树，有边权（维护成本），给定m组顶点，代表一条线路，同时给出了该线路带来的利润。可以删除树上的边，以及放弃若干条线路，求：最大利润。

问题即求最大权闭合子图。

建图方式：建立汇点和源点，将源点与m条线路相连，边权为利润，将汇点与树上节点相连，边权为成本，若第i条线路需要经过树上的第j条边，就把线路i和树边j连接，边权为INF

最后的答案为：全部利润和-该图最大流

由于点的数量为1e4最坏情况有1e8条边，无法直接建图，可以用线段树优化建图。对给出的树进行树链剖分，将每条线路的路径转化为若干连续区间，再使用线段树优化对区间的连边(若区间[L,R]在线段树上用一个节点维护，则直接将边连到改节点，不需要连到叶节点)。优化过的图，点数为5n+m=6e4边数为5n+m(1+logn)=2e5

树剖+线段树+网络流

```
#include<bits/stdc++.h>
using namespace std;
const int maxn=1e6+5,INF=2e9;
int n,m,u,v,c,x,y;
vector<pair<int,int> >ve[maxn];
int nxt[maxn],to[maxn],head[maxn],val[maxn],cur[maxn];
int cnt;
void add_edge(int u,int v,int va){
    to[cnt]=v;
    nxt[cnt]=head[u];
    head[u]=cnt;
    val[cnt]=va;
    cnt++;
}
int sz[maxn],hson[maxn],fa[maxn],dep[maxn],w[maxn];
int id[maxn],re[maxn],idx,top[maxn];
void dfs1(int p,int Fa,int weigh){
    sz[p]=1;
    fa[p]=Fa;
    dep[p]=dep[Fa]+1;
    w[p]=weigh;
    int SIZE=ve[p].size();
    int maxx=0;
    for(int i=0;i<SIZE;i++){
        if(ve[p][i].first==Fa)continue;
        dfs1(ve[p][i].first,p,ve[p][i].second);
        sz[p]+=sz[ve[p][i].first];
        if(maxx<sz[ve[p][i].first]){
            maxx=sz[ve[p][i].first];
            hson[p]=ve[p][i].first;
```

```
        }
    }
}
void dfs2(int p,int Fa,int t){
    id[p]=++idx;
    re[id[p]]=p;
    top[p]=t;
    if(hson[p])dfs2(hson[p],p,t);
    int SIZE=ve[p].size();
    for(int i=0;i<SIZE;i++){
        if(ve[p][i].first==Fa||ve[p][i].first==hson[p])continue;
        dfs2(ve[p][i].first,p,ve[p][i].first);
    }
}
int en;
void pushup(int rt){
    add_edge(rt,rt<<1,INF);
    add_edge(rt<<1,rt,0);
    add_edge(rt<<1|1,rt,0);
    add_edge(rt,rt<<1|1,INF);
}
void build(int l,int r,int rt){
    if(l==r){
        add_edge(rt,en,w[re[l]]);
        add_edge(en,rt,0);
        return;
    }
    int mid=(l+r)>>1;
    build(l,mid,rt<<1);
    build(mid+1,r,rt<<1|1);
    pushup(rt);
}
void query(int l,int r,int L,int R,int va,int rt){
    if(l>=L&&r<=R){
        add_edge(va,rt,INF);
        add_edge(rt,va,0);
        return;
    }
    int mid=(l+r)>>1;
    if(L<=mid)query(l,mid,L,R,va,rt<<1);
    if(R>mid)query(mid+1,r,L,R,va,rt<<1|1);
}
int dis[maxn];
bool f[maxn];
queue<int>qu;
bool bfs(){
    memset(dis,0,sizeof(dis));
    memset(f,0,sizeof(f));
    dis[0]=1;
    f[0]=1;
```

```
    qu.push(0);
    while(!qu.empty()){
        int fr=qu.front();
        qu.pop();
        for(int i=head[fr];i!=-1;i=nxt[i]){
            if(val[i]>0&&dis[to[i]]==0){
                dis[to[i]]=dis[fr]+1;
                if(!f[to[i]]){
                    f[to[i]]=1;
                    qu.push(to[i]);
                }
            }
        }
    }
    return dis[en]>0;
}
int dfs(int p,int maxflow){
    if(p==en)return maxflow;
    int rest=maxflow;
    for(int i=cur[p];i!=-1&&rest>0;i=nxt[i]){
        cur[p]=i;
        if(val[i]>0&&dis[to[i]]==dis[p]+1){
            int tmp=dfs(to[i],min(rest,val[i]));
            val[i]-=tmp;
            val[i^1]+=tmp;
            rest-=tmp;
        }
    }
    return maxflow-rest;
}
int dinic(){
    int res=0;
    while(bfs()){
        int tmp=0;
        for(int i=0;i<=4*n+m+1;i++)cur[i]=head[i];
        do{
            tmp=dfs(0,INF);
            res+=tmp;
        }while(tmp>0);
    }
    return res;
}
int main(){
    cin>>n>>m;
    memset(head,-1,sizeof(head));
    for(int i=1;i<n;i++){
        cin>>u>>v>>c;
        ve[v].push_back(make_pair(u,c));
        ve[u].push_back(make_pair(v,c));
    }
    dep[1]=1;
```

```
    dfs1(1,0,0);
    dfs2(1,0,1);
    en=4*n+m+1;
    build(1,n,1);
    int ans=0;
    for(int i=1;i<=m;i++){
        cin>>u>>v>>x>>y;
        if(x<=y)continue;
        add_edge(0,4*n+i,x-y);
        add_edge(4*n+i,0,0);
        ans+=x-y;
        while(top[u]!=top[v]){
            if(dep[top[u]]<dep[top[v]])swap(u,v);
            query(1,n,id[top[u]],id[u],4*n+i,1);
            u=fa[top[u]];
        }
        if(dep[u]<dep[v])swap(u,v);
        query(1,n,id[v]+1,id[u],4*n+i,1);
    }
    int flow=dinic();
    cout<<ans-flow<<endl;
    return 0;
}
```

C

D

贪心

n个人共m道菜，循环选取k道，每个人对不同菜的喜爱程度给出，每个人都希望这k道菜的喜爱程度的和最大

感性地，预知前人的选择是没有意义的，没人能对干预过去，轮到他们的时候前人的结果就已经确定了

只会考虑后人的选择，那么，最后一个人一定有能力选择自己最喜欢的菜，以此类推

每个人只需要点后人点过的菜以外自己最喜欢的菜，就可以进行回推了

AC代码

```
#include <bits/stdc++.h>
using namespace std;
typedef pair<int,int>  PII;
const int N = 2010;
int a[N][N];
PII b[N][N];
int vis[N];
int main(){
    int t;
```

```
    cin>>t;
    while(t--){
        int n,m,k;
        cin>>n>>m>>k;
        for(int i=0;i<n;++i) for(int j=0;j<m;++j) scanf("%d",&a[i][j]);
        int count = 0;
        for(int i=0;i<k;++i){
            if(count==n) count=0;
            for(int j=0;j<m;++j){
               b[i][j] = {a[count][j],j};
            }
            count++;
        }
        for(int i=k-1;i>=0;--i){
            sort(b[i],b[i]+m);
            for(int j=m-1;j>=0;--j){
                if(!vis[b[i][j].second]){
                   vis[b[i][j].second] = 1;
                   break;
                }
            }
        }
        for(int i=0;i<m;++i) if(vis[i]) printf("%d ",i+1);
        printf("\n");
        memset(vis,0,sizeof(int)*m);
    }

}
```

E

这个数量级的k基本一眼枚举,那么问题其实就变成了对于固定的x和10的幂次，如何快速找到满足条件的y

由于这个函数单调，只需要找到第一个使得该函数大于等于乘积的y即可，我的第一感觉是二分，直接让队长写了

赛后一想这个函数太简单了可以直接开方，然后稍微调整一下即可剩省一个log的复杂度，对于复杂的单调函数可以考虑二分

AC代码

```
#include <bits/stdc++.h>
typedef long long ll;
using namespace std;
int main(){
    int t;
    cin>>t;
    while(t--){
        int flag = 0;
        ll ans = 0;
```

```
        ll tmp = 1;
        ll x;
        cin>>x;
        for(int k=0;k<18;++k){
            ans = (ll)sqrt(x*tmp);
            if(ans>=(ll(1e9))) break;
            while(ans*ans<x*tmp) ans++;
            if(ans*ans/tmp==x){
                flag = 1;
                break;
            }
            tmp*=10;
        }
        if(flag) cout<<ans<<endl;
        else cout<<"-1"<<endl;
    }
}
```

F

G

H

I

签到题

给定n*m大小的棋盘,按五子棋规则构造两人和棋情况

结论很显然

如果行数为偶数，对每一行构造一个连续四个断一个即可，下一行和上一行的情况完全相反

如果行数为奇数，考虑前n-1行采取偶数行情况的构造，最后一行两个棋子交替即可

本来第一直觉以为是考察五子棋的日字八卦阵，结果发现我属实想多了(虽然正解是这么做的，但我觉得很没必要)

八卦阵是和棋局面的充分条件，并非必要条件

AC代码

```
#include <bits/stdc++.h>
using namespace std;
int main(){
    int t;
    cin>>t;
    while(t--){
```

```
        int n,m;
        cin>>n>>m;
        int flag = 1;
        for(int i=0;i<(n%2?n-1:n);++i){
               int count = 1;
               for(int j=0;j<m;++j){
                      if(flag>0){
                         if(count<=4){
                               printf("x");
                               count++;
                         }
                         else{
                               count = 1;
                               printf("o");
                         }
                      }
                      else{
                         if(count<=4){
                               printf("o");
                               count++;
                         }
                         else{
                               count = 1;
                               printf("x");
                         }
                      }
               }
               flag = -flag;
               printf("\n");
        }
        if(n%2){
            for(int j=0;j<m;++j){
                if(j%2==0) printf("x");
                else printf("o");
            }
            printf("\n");
        }

        }
}
```

J

K

L