比赛链接

补题情况

题目	蒋贤蒙	王赵安	王智彪
A	0	0	0
B	0	0	0
C	0	0	0
D	2	0	0
E	0	0	0
G	0	0	0
J	2	0	0
K	0	0	0

题解

J. Defend Your Country

题意

给定 n 个点 m 条边的连通图，要求删去任意条边，最大化图的点权和。

其中第 i 个点的点权的绝对值为 a_i，如果该点所在连通块大小为偶数则权值为正，否则为负。

题解

不难发现如果 n 为偶数则不需要任何删边。

如果 n 为奇数，如果最优解中存在一个大小不为 1 的奇连通块，任取该连通块的一个点删去所有相关连边一定不会使得答案变劣。

因此不妨考虑强制删一个点，如果这个点不是割点，显然删除该点后不需要额外删点，统计此时的答案。

如果这个点是割点，则需要考虑删去这个点得到的剩余连通分量，如果每个连通分量大小都是偶数，显然也不需要额外删点。

否则，删去该点后得到至少两个奇连通分量，还要继续在奇连通分量中删点，事实上，这种策略一定不是最优的，下面给出证明：

假设这是最优策略，则所有删去的点一定是原图中的割点，否则如果存在非割点一开始删去该点才是最优。

另外，每次删去割点一定得到的都是奇连通块，否则考虑 (偶连通块-第二个割点-偶连通块)-第一个割点-奇连通块 的情况。

其中原策略是删除第一个割点后得到奇连通块 (偶连通块-第二个割点-偶连通块)，再删除第二个割点。这样一定不如直接删去第二个割点。

于是由于每个割点删完后一定存在与他相邻的奇连通块，然后又要在奇数连通块中删割点，于是每个割点

一定有两个相邻割点。

考虑在原图建立点双树，易知点双树的叶子割点一定没有两个相邻割点，矛盾。因此假设不成立。

至于维护删除割点后的其他连通分量奇偶性可以在跑 dfs 树时顺便维护子树大小，根据子树奇偶性判断。

特别注意即使 u 是割点，删除 u 也不能保证 u 的每个子树都构成独立连通分量。

事实上如果 $\text{low[v]}<\text{dfn}[u]$ 则说明 v 和 u 的祖先结点属于同一个点双连通分量，不要重复判定。**比赛的时候就这里假了，长了个教训**

时间复杂度 $O(n+m)$

```
const int MAXN=1e6+5,MAXM=2e6+5,Inf=1e9;
struct Edge{
    int to,next;
}edge[MAXM<<1];
int head[MAXN],edge_cnt;
void Insert(int u,int v){
    edge[++edge_cnt]=Edge{v,head[u]};
    head[u]=edge_cnt;
}
int low[MAXN],dfn[MAXN],sz[MAXN],dfs_t;
bool iscut[MAXN],fib[MAXN];
void dfs(int u,int fa){
    low[u]=dfn[u]=++dfs_t;
    int child=0;
    sz[u]=1;
    for(int i=head[u];i;i=edge[i].next){
        int v=edge[i].to;
        if(v==fa)continue;
        if(!dfn[v]){
            dfs(v,u);
            sz[u]+=sz[v];
            if(sz[v]%2&&low[v]>=dfn[u])
            fib[u]=true;
            low[u]=min(low[u],low[v]);
            if(low[v]>=dfn[u]&&u!=fa)
            iscut[u]=true;
            if(u==fa)child++;
        }
        low[u]=min(low[u],dfn[v]);
    }
    if(u==fa&&child>=2)
    iscut[u]=true;
}
int a[MAXN];
void solve(){
    int n=read_int(),m=read_int();
    LL ans=0;
```

```
    edge_cnt=0,dfs_t=0;
    _rep(i,1,n){
        a[i]=read_int();
        head[i]=0;
        dfn[i]=0;
        iscut[i]=0;
        fib[i]=0;
        ans+=a[i];
    }
    while(m--){
        int u=read_int(),v=read_int();
        Insert(u,v);
        Insert(v,u);
    }
    if(n%2==0){
        enter(ans);
        return;
    }
    dfs(1,1);
    int det=Inf;
    _rep(i,1,n){
        if(!iscut[i])
        det=min(det,a[i]);
        else if(!fib[i])
        det=min(det,a[i]);
    }
    enter(ans-det*2);
}
int main(){
    int T=read_int();
    while(T--){
        solve();
    }
    return 0;
}
```