错题集 4

1、小V的序列

链接

题意

给定一个长度为 n 的序列，保证序列中每个数取值随机。

接下来给定 m 个询问，每次给定一个数 b□问序列中是否存在数与 b 异或后二进制表示中 1 的个数不超过 3。

题解

考虑将 b 的二进制表示分成四部分，每部分表示一个长度为 16 二进制数。

则如果要与 b 异或后二进制表示中 1 的个数不超过 3，则至少四部分要有一个部分与 b 相同。

考虑将序列中的每个数分成四个，每个数投入对应位置二进制数的桶中，然后暴力查询，时间复杂度 $O\left(\cfrac {nm}{2^{16}}\log v\right)$□

```
const int MAXN=1e6+5,MAXM=1<<16,Mod=998244353;
LL a[MAXN];
vector<LL> c[4][MAXM];
uint64_t G(uint64_t x){
    x^=x<<13;
    x^=x>>7;
    x^=x<<17;
    return x;
}
bool check(LL x){
    int cnt=0;
    _for(i,0,64){
        if((x>>i)&1)
        cnt++;
    }
    return cnt<=3;
}
int main()
{
    int n=read_int(),m=read_int();
    a[0]=read_LL();
    _for(i,1,n)a[i]=G(a[i-1]);
    _for(i,0,n){
        _for(j,0,4)
```

```
        c[j][(a[i]>>(16*j))&((1<<16)-1)].push_back(a[i]);
    }
    int ans=0;
    while(m--){
        LL b=read_LL();
        bool flag=false;
        _for(i,0,4){
            int t=(b>>(16*i))&((1<<16)-1);
            _for(j,0,c[i][t].size()){
                if(check(c[i][t][j]^b)){
                    flag=true;
                    break;
                }
            }
            if(flag)
            break;
        }
        ans=(ans<<1|flag)%Mod;
    }
    enter(ans);
    return 0;
}
```

2□Monster Hunter

链接

题意

给定以 1 为根的 n 阶的点权树，假定可以用无费用无限制删去其中 k 个结点，问删除剩余结点的最小费用。

其中，对剩余的所有结点，删除它之前需要删除它的父结点，同时删除它的费用为它的点权 $+$ 当前的所有儿子结点的费用的和。

要求输出 $k=0,1\cdots n$ 时的答案。

题解

易知确认无费用无限制删除的结点后答案已经固定。考虑 $dp(u,k,0/1)$ 表示以结点 u 的子树中有代价删除 k 个结点的最小费用。

其中 $dp(u,k,0)$ 表示无代价删除 u 结点□ $dp(u,k,1)$ 表示有代价删除 u 结点。不难得到状态转移方程

$$ dp(u,i+j,0)=\min(dp(u,i+j,0),dp(u,i,0)+\min(dp(v,j,0),dp(v,j,1))) $$

$$ dp(u,i+j,1)=\min(dp(u,i+j,0),dp(u,i,1)+\min(dp(v,j,0),dp(v,j,1)+w_v)) $$

注意优化转移方式，结合代码，不难发现转移过程等效于枚举每个点对的 LCA所以复杂度为 $O(n^2)$

```
const int MAXN=2e3+5;
const LL Inf=1e18;
LL dp[MAXN][MAXN][2];
int head[MAXN],edge_cnt,w[MAXN];
struct Edge{
    int to,next;
}edge[MAXN];
void Insert(int u,int v){
    edge[++edge_cnt]=Edge{v,head[u]};
    head[u]=edge_cnt;
}
int sz[MAXN];
void dfs(int u){
    dp[u][0][0]=0;
    dp[u][1][1]=w[u];
    sz[u]=1;
    for(int i=head[u];i;i=edge[i].next){
        int v=edge[i].to;
        dfs(v);
        for(int j=sz[u];j>=0;j--)
        _rep(k,1,sz[v]){
dp[u][j+k][0]=min(dp[u][j+k][0],dp[u][j][0]+min(dp[v][k][0],dp[v][k][1]));
dp[u][j+k][1]=min(dp[u][j+k][1],dp[u][j][1]+min(dp[v][k][0],dp[v][k][1]+w[v
]));
        }
        sz[u]+=sz[v];
    }
}
int main()
{
    int T=read_int();
    while(T--){
        int n=read_int();
        _rep(i,1,n)_rep(j,1,n)dp[i][j][0]=dp[i][j][1]=Inf;
        edge_cnt=0;
        _rep(i,1,n)head[i]=0;
        _rep(i,2,n)Insert(read_int(),i);
        _rep(i,1,n)w[i]=read_int();
        dfs(1);
        for(int i=n;i;i--)
        space(min(dp[1][i][0],dp[1][i][1]));
        enter(0);
    }
    return 0;
```

```
}
```