H. travel

题意

给定一棵点权树，从树上选三条不相交的路径，每条路径的权值定义为路径上的点权和，要求最大化三条路径权值和。

题解

设 $\text{dp}(u,0/1/2,i)$ 表示只考虑 u 的子树，结点 u 的状态为 $0/1/2$ 时，已经选中了 i 条链此时的最大路径权值和。

我们需要维护一条正在生成的链，这条链不包含在已经选中的 i 条链当中，如果 u 状态为 0 表示 u 不在生成链中。

如果 u 状态为 1 表示 u 在生成链中且 u 只有一个儿子在生成链中 u 状态为 2 表示 u 在生成链中且 u 有两个儿子在生成链中。

考虑状态转移，利用生成链的合并，不难有

$$ \text{dp}(u,0,i+j)\gets \text{dp}(u,0,i)+\text{dp}(v,0,j)\\ \text{dp}(u,1,i+j)\gets \text{dp}(u,0,i)+\text{dp}(v,1,j)+a_u\\ \text{dp}(u,1,i+j)\gets \text{dp}(u,1,i)+\text{dp}(v,0,j)\\ \text{dp}(u,2,i+j)\gets \text{dp}(u,1,i)+\text{dp}(v,1,j)\\ \text{dp}(u,2,i+j)\gets \text{dp}(u,2,i)+\text{dp}(v,0,j) $$

注意上式的 $\gets$ 表示取最大值，另外为了防止选中复数条从 v 生成的链，需要开一个临时数组存储中间量。

初始状态为 $\text{dp}(u,1,0)=a_u$ 最后转移完要考虑将正在生成的链转化为已经选中的链，于是有

$$ \text{dp}(u,0,i)\gets \max(\text{dp}(u,1,i-1),\text{dp}(u,2,i-1)) $$

最终答案为 $\text{dp}(1,0,3)$ 时间复杂度 $O(nk^2)$ 其中 k 表示最多能选中的链的个数。

参考资料

```
const int MAXN=4e5+5,MAXK=4;
struct Edge{
    int to,next;
}edge[MAXN<<1];
int a[MAXN],head[MAXN],edge_cnt;
LL dp[MAXN][3][MAXK],tmp[3][MAXK];
void Insert(int u,int v){
    edge[++edge_cnt]=Edge{v,head[u]};
    head[u]=edge_cnt;
}
void Max(LL &a,LL b){
    if(b>a)
    a=b;
}
```

```
void dfs(int u,int fa){
    dp[u][1][0]=a[u];
    for(int i=head[u];i;i=edge[i].next){
        int v=edge[i].to;
        if(v==fa)continue;
        dfs(v,u);
        _for(i,0,3)_for(j,0,MAXK)
        tmp[i][j]=dp[u][i][j];
        _for(i,0,MAXK)_for(j,0,MAXK-i){
            Max(tmp[0][i+j],dp[u][0][i]+dp[v][0][j]);
            Max(tmp[1][i+j],dp[u][0][i]+dp[v][1][j]+a[u]);
            Max(tmp[1][i+j],dp[u][1][i]+dp[v][0][j]);
            Max(tmp[2][i+j],dp[u][1][i]+dp[v][1][j]);
            Max(tmp[2][i+j],dp[u][2][i]+dp[v][0][j]);
        }
        _for(i,0,3)_for(j,0,MAXK)
        dp[u][i][j]=tmp[i][j];
    }
    _for(i,1,MAXK)
    Max(dp[u][0][i],max(dp[u][1][i-1],dp[u][2][i-1]));
}
int main()
{
    int n=read_int();
    _rep(i,1,n)
    a[i]=read_int();
    _for(i,1,n){
        int u=read_int(),v=read_int();
        Insert(u,v);
        Insert(v,u);
    }
    dfs(1,0);
    enter(dp[1][0][3]);
    return 0;
}
```