[报告]poj 3728 The merchant

[Source]

http://poj.org/problem?id=3728

[Description]

给出一颗树，每个点有一个权值，有很多次询问，每次询问会给出树上两点分别作为起点和终点，这两点就决定了树上的一条有向的路径，求这条路径的任意两点间最大的点权之差，唯一的要求是点权小的那个点必须更靠近起点。

[Solution]

用倍增法求 lca 的时候会用到一个辅助数组 parent，parent[i][j]表示 i 节点的第 2^j 个祖先是哪个节点。我们可以用同样的方法定义 min 数组，min[i][j]表示从 i 节点到它的 2^j 个祖先中最小的点权是多少，同样我们定义 max 数组，max[i][j]表示从 i 节点到它的 2^j 个祖先中最大的点权是多少，定义 up 数组，up[i][j] 表示从 i 节点到它的 2^j 个祖先这条路上的最大收益是多少，定义 down 数组，down[i][j] 表示从 i 节点的 2^j 个祖先到它的这条路上的最大收益是多少。这四个数组都可以用和预处理 parent 数组同样的方法预处理出来。

对于每个询问 q(a,b),首先求出 a 和 b 的 lca（假设为 p），然后用倍增法求出从 a 到 p 的最大收益 x（运用 up 数组），用倍增法求出从 p 到 b 的最大收益 y（运用 down 数组），用倍增法求出从 a 到 p 的最小值 z 和 p 到 b 的最大值 w（运用 max 和 min 数组），然后 x，y，w-z 这三个数的最大值就是所求的答案。

可以看出，此题的复杂度和用倍增法求 lca 的复杂度是一样的，预处理是 nlog(n),对每次询问的处理时间是 log(n).

[Code]

```
#include<stdio.h>
#include<string.h>
#include<stdlib.h>
#include<queue>
#include<iostream>
using namespace std;
#define size 50100
int dep[size];
int mm[size][18],nn[size][18],zh[size][18],re[size][18],par[size][18];
/*分别对应了上文提到的max,min,up,down,parent数组*/
int v[size];
typedef struct E{
    int to,next;
}E;
```

```
E edge[size<<1];
int org[size];
int most(int a,int b,int c)
{
    if(a<b)
        a=b;
    if(a<c)
        a=c;
    return a;
}
void dfs(int cur,int up)//对那5个数组进行预处理
{
    int temp;

    if(up==0)
        dep[cur]=1;
    else{
        dep[cur]=dep[up]+1;
        mm[cur][0]=max(org[cur],org[up]);
        nn[cur][0]=min(org[cur],org[up]);
        zh[cur][0]=org[up]-org[cur];
        re[cur][0]=org[cur]-org[up];
        par[cur][0]=up;
    }
        for(int j=1;par[cur][j-1]!=0;j++){
            int temp=par[par[cur][j-1]][0];
            if(temp==0||par[temp][j-1]==0)
                par[cur][j]=0;
            else{
                par[cur][j]=par[temp][j-1];
                mm[cur][j]=max(mm[cur][j-1],mm[temp][j-1]);
                nn[cur][j]=min(nn[cur][j-1],nn[temp][j-1]);

    zh[cur][j]=most(mm[temp][j-1]-nn[cur][j-1],zh[cur][j-1],zh[temp][j-1]);

    re[cur][j]=most(mm[cur][j-1]-nn[temp][j-1],re[cur][j-1],re[temp][j-1]);
            }
        }
    for(int x=v[cur];x!=-1;x=edge[x].next)
        if(edge[x].to!=up)
            dfs(edge[x].to,cur);
}
int lca(int a,int b)//求两点的lca
{
```

```
    int i;
    if(dep[a]>dep[b])
        swap(a,b);
    while(dep[b]>dep[a]){
        for(i=0;par[b][i+1]!=0&&dep[par[b][i+1]]>=dep[a];i++);
        b=par[b][i];
    }
    while(a!=b){
        for(i=0;par[a][i+1]!=par[b][i+1];i++);
        a=par[a][i];
        b=par[b][i];
        if(a!=b){
            a=par[a][0];
            b=par[b][0];
        }
    }
    return a;
}
int query(int a,int b)//回答询问
{
    if(a==b)
        return 0;
    int p=lca(a,b);
    int ans=0;
    int sma=org[a],big=org[b];
    int temp,i;

    temp=a;
    while(temp!=p){
        for(i=0;par[temp][i+1]!=0&&dep[par[temp][i+1]]>=dep[p];i++);
        ans=most(ans,zh[temp][i],mm[temp][i]-sma);
        sma=min(sma,nn[temp][i]);
        temp=par[temp][i];
    }
    temp=b;
    while(temp!=p){
        for(i=0;par[temp][i+1]!=0&&dep[par[temp][i+1]]>=dep[p];i++);
        ans=most(ans,re[temp][i],big-nn[temp][i]);
        big=max(big,mm[temp][i]);
        temp=par[temp][i];
    }
    ans=max(ans,big-sma);
    return ans;
}
```

```
int main()
{
    int n,q,root,i,j,a,b;

    while(scanf("%d",&n)==1){
        memset(v,-1,sizeof(v));
        for(i=1;i<=n;i++){
            scanf("%d",&org[i]);
        }
        for(i=1;i<n;i++){
            scanf("%d %d",&a,&b);
            edge[i*2-1].to=a;
            edge[i*2-1].next=v[b];
            v[b]=i*2-1;
            edge[i*2].to=b;
            edge[i*2].next=v[a];
            v[a]=i*2;
        }
        root=1;
        memset(par,0,sizeof(par));
        dfs(root,0);
        scanf("%d",&q);
        while(q--){
            scanf("%d %d",&a,&b);
            printf("%d\n",query(a,b));
        }
    }
    return 0;
}
```