茶余饭后自学资料-tarjan求LCA

> Robert Tarjan（萝卜糖尖）的确是一位伟大的计算机科学家，灵活巧妙的利用dfs树，解决了图论部分连通性问题

LA问题与RMQ问题，是信息学比较经典的两种问题，某些题目可以转化为这两种问题进行求解。

LCA：最近公共祖先，RMQ：区间最值询问

LCA 问题有多种求法

前面已经学习了倍增求解LCA

今天来学习一下另外一种解法：tarjan求LCA，相对来说tarjan的解法更容易写和理解。

Tarjan算法是一个常见的用于解决LCA问题的离线算法，它结合了DFS和并查集，整个算法为线性处理时间，O（n + Q），n为点数，Q为查询次数。

tarjan算法求LCA主要基于深度优先遍历，可以看成是对朴素**向上标记法求LCA**的优化，在深度优先遍历的过程中将遍历节点的状态分为三大类：

- 当前还未搜到的点，状态可以标记为0
- 正在遍历的节点，状态可以标记为1
- 已经遍历过且回溯完的节点，状态可以标记为2

算法过程：

1.任选一个点为根节点，从根节点开始。

2.遍历该点u所有子节点v，并标记这些子节点v已被访问过。

3.若是v还有子节点，返回2，否则下一步。

4.合并v到u上。

5.寻找与当前点u有询问关系的点v。

6.若是v已经被访问且回溯过了，则可以确认u和v的最近公共祖先为:v的父亲节点a。

算法伪代码:

```
Tarjan(u)//merge和find为并查集合并函数和查找函数
    {
        for each(u,v)    //访问所有u子节点e
        {
            Tarjan(v);         //继续往下遍历
            merge(u,v);     //合并v到u上
            标记v被访问过;
        }
        for each(u,v' )     //访问所有和u有询问关系的v'
       {
            如果 v' 被访问过;
```

```
            u,v' 的最近公共祖先为find( v' );
        }
    }
```

参考代码

```
#include <bits/stdc++.h>

using namespace std;
const int N = 500010;
int head[N],dis[N],f[N],cnt,deep[N],ans[N],status[N];
int n,m;
vector<pair<int,int> > v[N];
struct node
{
    int to,w,ne;
} e[N];
void add(int x,int y)  //建图
{
    cnt++;
    e[cnt].to = y;
    e[cnt].ne = head[x];
    head[x] = cnt;
}
int  find(int x)  //并查集
{
    if(x==f[x])
        return x;
    return f[x]=find(f[x]);
}
void dfs(int u)  //dfs求节点深度
{
    for(int i=head[u]; i!=-1; i=e[i].ne)
    {
        int v = e[i].to;
        if(!deep[v])
        {
            deep[v] = deep[u]+1;
            dfs(v);
        }
    }
}
void tarjan(int u)
{
    status[u] = 1;  //标记访问，未回溯
    for(int i=head[u]; i!=-1; i=e[i].ne)  //遍历该点u所有子节点v，并标记这些子节点v已被访问过。
    {
        int now = e[i].to;
        if(!status[now])
        {
            tarjan(now);
            f[now] = u;  //合并
        }
    }
    for(auto each:v[u])  //寻找与当前点u有询问关系的点v
```

```
    {
        int cx = each.first,cy = each.second;
        if(status[cx] == 2)
        {
            int lca = find(cx); //求LCA，v的父亲就是u和v的LCA
            ans[cy] = deep[u]+deep[cx]-2*deep[lca];  //求距离
        }
    }

    status[u] = 2; //标记访问，且已回溯
}
int main()
{
    memset(head,-1,sizeof(head));
    cin >> n;
    for(int i=1; i<n; i++){
        int x,y;
        cin >> x >>y;
        add(x,y);
        add(y,x);
    }
    deep[1] = 1;
    for(int i=1; i<=n; i++)
        f[i] = i;
    dfs(1);
    cin>>m;
    for(int i=1; i<=m; i++) { //存储所有询问
        int x,y;
        cin >> x >>y;
        v[x].push_back({y,i});
        v[y].push_back({x,i});
    }
    tarjan(1);  //以1为根节点访问
    for(int i=1; i<=m; i++){
        cout <<ans[i]<<endl;
    }
    return 0;
}
```

现在你又掌握了一种tarjan解法。

LCA问题还可以通过DFS序(准确说是欧拉序)，转换成RMQ问题求解，感兴趣还可以先去学习一下。