**Tarjan三大算法之双连通分量（割点，桥）**

Robert Endre Tarjan是一个美国计算机学家，他传奇的一生中发明了无数算法，统称为Tarjan算法。其中最著名的有三个，分别用来求解   
1) 无向图的双连通分量   
2) 有向图的强连通分量   
3) 最近公共祖先问题   
接下来几篇博客将分别讲述三个算法，首先是无向图的双连通分量，我们先从无向图的割点和桥讲起。

下面介绍中**无向图**中割点和桥的概念：   
**割点：**一个结点称为割点（或者割顶）当且仅当去掉该节点极其相关的边之后的子图不连通。   
**桥：**一条边称为桥（或者割边）当且仅当去掉该边之后的子图不连通。   
**割点：**   
首先我们考虑一个连通图（非连通图可以分别考虑连通块），我们从任意一个起点开始进行深度优先搜索，可以得到一棵树，并且这棵树中所有结点的子树之间不存在边，即没有跨越两棵子树的边（考虑一下，如果存在，那么与深度优先搜索树的定义互相矛盾）。于是有如下定理：   
在无向连通图G中，   
1、根结点u为割顶当且仅当它有两个或者多个子结点；   
2、非根结点u为割顶当且仅当u存在结点v，使得v极其所有后代都没有反向边可以连回u的祖先（u不算）   
在Tarjan算法里面，有两个时间戳非常重要，一个是dfn，意为深度优先数，即代表访问顺序；一个是low，意为通过反向边能到达的最小dfn。于是，上述定理中第二个条件（非根结点）可以简单地写成low[v]>=dfn[u]。   
**代码如下：**

int n,m,stamp,low[1005],dfn[1005],iscut[1005];

vector<int> vec[1005];

void tarjan(int index,int fa){

int child=0;

low[index]=dfn[index]=++stamp;

for(int i=0;i<vec[index].size();i++)

{

int tmp=vec[index][i];

if(!dfn[tmp])

{

child++;

tarjan(tmp,index);

low[index]=min(low[index],low[tmp]);

if(low[tmp]>=dfn[index])

iscut[index]=1;

}

else if(dfn[tmp]<dfn[index] && tmp!=fa)

{

low[index]=min(low[index],dfn[tmp]);

}

}

if(fa<0 && child==1)

iscut[index]=0;

}

**桥：**   
桥的求法其实也是类似的，它的求法可以看成是割顶的一种特殊情况，当结点u的子结点v的后代通过反向边只能连回v，那么删除这条边(u, v)就可以使得图G非连通了。用Tarjan算法里面的时间戳表示这个条件，就是low[v]>dfn[u]。   
**代码如下：**

int n,stamp,dfn[1005],low[1005];

int cnt,ansx[10005],ansy[10005];

vector<int> vec[1005];

int rank[1005];

void addAns(int x,int y)

{

if(x>y)

swap(x,y);

ansx[cnt]=x, ansy[cnt]=y;

cnt++;

}

void tarjan(int index,int fa)

{

int tmp;

dfn[index]=low[index]=++stamp;

for(int i=0;i<vec[index].size();i++)

{

tmp=vec[index][i];

if(!dfn[tmp])

{

tarjan(tmp,index);

low[index]=min(low[index],low[tmp]);

if(low[tmp]>dfn[index])

addAns(index,tmp);

}

else if(dfn[tmp]<dfn[index] && tmp!=fa)

{

low[index]=min(low[index],dfn[tmp]);

}

}

}