

Bright


Action is more important than imagination... ^_^

目录视图

摘要视图

RSS 订阅

个人资料



小小程序员

访问：200936次

积分：3351

等级：BLOG 5

排名：第5699名

原创：113篇

转载：11篇

译文：5篇

评论：51条

文章分类

DM & ML (10)

Python (13)

C/C++ (16)

Algorithm (10)

Programming (20)

Interview Related (13)

Personal (5)

Online Judge (4)

Dynamic Programming (16)

Math (8)

Reposts (9)

Java (2)

R (1)

basic (2)

spark (1)

engineerin (5)

Good Links

欢迎加群—

ml&dm&programming(489553410)

UESTC

CDOJ(UESTC OJ)

文章搜索

2016软考项目经理实战班 学院周年礼-顶尖课程钜惠呈现 Hadoop英雄会—暨Hadoop 10周年生日大趴 【博客专家】有奖试读—Windows PowerShell实战指南

Tarjan算法求解桥和边双连通分量（附POJ 3352 Road Construction解题报告）

标签：Tarjan 双连通分量

2011-07-21 11:03 6161人阅读 评论(6) 收藏 举报

分类：Algorithm (9)

版权声明：本文为博主原创文章，未经博主允许不得转载。

在说Tarjan算法解决桥和边双连通分量问题之前我们先来回顾一下Tarjan算法是如何求解强连通分量的。

Tarjan算法在求解强连通分量的时候，通过引入dfs过程中对一个点访问的顺序dfsNum（也就是在访问该点之前已经访问的点的个数）和一个点可以到达的最小的dfsNum的low数组，当我们遇到一个顶点的dfsNum值等于low值，那么该点就是一个强连通分量的根。因为我们在dfs的过程中已经将点仍入到栈中，因此我们只需要将栈中的元素出栈直到遇到根，那么这些点就组成一个强连通分量。

对于边双连通分量，我们需要先了解一些概念：

边连通度：使一个子图不连通所需要删除的最小的边数就是该图的边连通度。

桥（割边）：当删除一条边就使得图不连通的那条边称为桥或者是割边。

边双连通分量：边连通度大于等于二的子图称为边双连通分量。

理解了这些概念之后我们来看看Tarjan是如何求解边双连通分量的，不过在此之前我们先说说Tarjan是怎样求桥的。同样引入了dfsNum表示一个点在dfs过程中所被访问的时间，然后就是low数组表示该点最小的可以到达的dfsNum。我们分析一下桥的特点，删除一条边之后，那么如果dfs过程中的子树没有任何一个点可以到达父亲节点及父亲节点以上的节点，那么这个时候子树就被封死了，这条边就是桥。有了这个性质，也就是说当我们dfs过程中遇到一条树边a->b，并且此时low[b]>dfsNum[a]，那么a-b就是一座桥。

呵呵桥都求出来了，还怕边双连通分量吗？我们把所有的桥去掉之后那些独立的分量就是不同的边双连通分量，这个时候就可以按照需要灵活的求出边双连通分量了。

下面附上POJ 3352的解题思路吧：

这道题的意思是说，给你一个无向图，然后问你至少需要添加几条边，可以使整个图变成边双连通分量，也就是说任意两点至少有两条路可以互相连通。我们这样考虑这个问题，属于同一个边双连通分量的任意点是至少有两条通路是可以互相可达的，因此我们可以将一个边双连通分量缩成一个点。然后考虑不在边双连通分量中的点，通过缩点后形成的是一棵树。对于一个树型的无向图，需要添加（度为1的点的个数+1）/2条边使得图成为双连通的。这样问题就是变成缩点之后，求图中度为1的点的个数了。

这个题目的条件给的很强，表示任意两个点之间不会有重边，因此我们可以直接经过Tarjan的low值进行边双连通分量的划分，最后求出度为1点数就可以解决问题了。如果是有重边的话，那么不同的low值是可能是属于同一个边双连通分量的，这个时候就要通过将图中的桥去掉然后求解边双连通分量，这个请见我的博客的另外一篇解题报告。

阅读排行

如何让自己的博客被搜索 (17956)
 python中创建指定大小的 (6435)
 Tarjan算法求解桥和边双 (6158)
 Ubuntu下看不见pthread (5743)
 逆波兰表达式算法 (5388)
 国内外OJ简介 (4914)
 (算法) Tarjan离线算法 (4348)
 C语言函数调用时参数压 (3587)
 求解最长回文子串 Mana (3583)
 Tarjan求有向图的强连通 (3376)

文章存档

2016年01月 (1)
 2015年12月 (1)
 2015年11月 (1)
 2015年09月 (1)
 2015年05月 (4)

展开

下面贴上POJ 3352的ac代码，供网友们参考：

```
[cpp]
01. #include <iostream>
02. #include <cstring>
03. #include <stdlib.h>
04. #include <stdio.h>
05. #include <vector>
06. using namespace std;
07. const int Max=1010;
08. int top[Max],edge[Max][Max]; //memset(top,0,sizeof(top));
09. int dfsNum[Max],dfsnum; //memset(dfsNum,0,sizeof(dfsNum)),dfsNum=1;
10. int low[Max];
11. int degree[Max];
12. int ans;
13.
14. void tarjan(int a,int fa)
15. {
16.     dfsNum[a]=low[a]=++dfsnum;
17.     for(int i=0;i<top[a];i++)
18.     {
19.         if(edge[a][i]!=fa)
20.         {
21.             if(dfsNum[edge[a][i]]==0)
22.             {
23.                 tarjan(edge[a][i],a);
24.                 if(low[a]>low[edge[a][i]])
25.                     low[a]=low[edge[a][i]];
26.             }
27.             else
28.             {
29.                 if(low[a]>dfsNum[edge[a][i]])
30.                     low[a]=dfsNum[edge[a][i]];
31.             }
32.             // if(low[edge[a][i]]>dfsNum[a])
33.             // {
34.             // }
35.             // }
36.         }
37.     }
38. }
39.
40. int solve(int n)
41. {
42.     int i,j;
43.     int a,b;
44.     for(i=1;i<=n;i++)
45.     {
46.         a=i;
47.         for(j=0;j<top[i];j++)
48.         {
49.             b=edge[a][j];
50.             if(low[a]!=low[b])
51.             {
52.                 degree[low[a]]++;
53.                 degree[low[b]]++;
54.             }
55.         }
56.     }
57.     int leaves=0;
58.     for(i=1;i<=n;i++)
59.     {
60.         if(degree[i]==2)
61.         {
62.             leaves++;
63.         }
64.     }
65.     return (leaves+1)/2;
66. }
67.
68. int main()
69. {
70.     int n,m;
71.     int i,a,b;
72.     while(scanf("%d %d",&n,&m)!=EOF)
73.     {
74.         memset(top,0,sizeof(top));
75.         memset(degree,0,sizeof(degree));
```

```

76.         for(i=0;i<m;i++)
77.         {
78.             scanf("%d %d",&a,&b);
79.             edge[a][top[a]++]=b;
80.             edge[b][top[b]++]=a;
81.         }
82.
83.         memset(dfsNum,0,sizeof(dfsNum));
84.         dfsnum=0;
85.
86.         tarjan(1,-1);
87.         ans=solve(n);
88.         printf("%d\n",ans);
89.     }
90.     return 0;
91. }

```

上面的代码的写法确实有问题，因为在同一个双连通分量中的点的low值并不一定相等，所以使用low值来判断是否在同一分量中显然是有问题的。因此我们最安全的做法是在tarjan的过程中，把桥标记出来，然后使用dfs跑每一个连通分量，最后使用桥统计每个点的度。

对于上面的问题，给大家带来的误导，非常抱歉。

下面的代码是对上面代码的修改，写的有点乱，供大家参考一下。

```

[cpp]
01. #include <iostream>
02. #include <cstring>
03. #include <cstdlib>
04. #include <cstdio>
05. #include <vector>
06. using namespace std;
07. const int Max=1010;
08. int top[Max]; //memset(top,0,sizeof(top));
09. int dfsNum[M]; //memset(dfsNum,0,sizeof(dfsNum)),dfsNum=1;
10. int low[Max]
11. int degree[M]
12. int ans;
13.
14. bool exist[Max][Max],
15.
16. void tarjan(int u, int fa,
17. {
18.     dfsNum[u] = ++dfsnum;
19.     for(int i=0;i<top[u];i++)
20.     {
21.         if(edge[u][i]!=fa)
22.         {
23.             if(dfsNum[edge[u][i]]==0)
24.             {
25.                 tarjan(edge[u][i],u);
26.                 if(low[u]>low[edge[u][i]])
27.                     low[u]=low[edge[u][i]];
28.                 if(dfsNum[u] < low[edge[u][i]])
29.                 {
30.                     exist[u][edge[u][i]] = exist[edge[u][i]][u] = true;
31.                 }
32.             }
33.             else
34.             {
35.                 if(low[u]>dfsNum[edge[u][i]])
36.                     low[u]=dfsNum[edge[u][i]];
37.             }
38.         }
39.     }
40. }
41.
42. int cc[Max], ccCnt;
43.
44. void dfs(int fa, int u)
45. {
46.     cc[u] = ccCnt;
47.     for(int i=0; i<top[u]; i++)
48.     {
49.         int v = edge[u][i];
50.         if(v != fa && !exist[u][v] && !cc[v])

```

```

51.         {
52.             // printf("v %d\n", v);
53.             dfs(u, v);
54.         }
55.     }
56. }
57.
58. int solve(int n)
59. {
60.     int i,j;
61.     int a,b;
62.
63.     memset(cc, 0, sizeof(cc));
64.     ccCnt = 1;
65.     for(i=1; i<=n; i++)
66.     {
67.         if(!cc[i])
68.         {
69.             dfs(-1, i);
70.             ccCnt++;
71.         }
72.     }
73.
74.     //cout<<"ccCnt " <<ccCnt<<endl;
75.
76.     for(i=1;i<=n;i++)
77.     {
78.         a=i;
79.         for(j=0;j<top[i];j++)
80.         {
81.             b=edge[a][j];
82.             if(cc[a] != cc[b])
83.             {
84.                 degree[cc[a]]++;
85.                 degree[cc[b]]++;
86.             }
87.         }
88.     }
89.     int leaves=0;
90.     for(i=1;i<ccCnt;i++)
91.     {
92.         if(degree[i]==2)
93.         {
94.             leaves++;
95.         }
96.     }
97.     return (leaves+1)/2;
98. }
99.
100. int main()
101. {
102.     int n,m;
103.     int i,a,b;
104.     while(scanf("%d %d",&n,&m)!=EOF)
105.     {
106.         memset(top,0,sizeof(top));
107.         memset(degree,0,sizeof(degree));
108.         for(i=0;i<m;i++)
109.         {
110.             scanf("%d %d",&a,&b);
111.             edge[a][top[a]++]=b;
112.             edge[b][top[b]++]=a;
113.         }
114.
115.         memset(dfsNum,0,sizeof(dfsNum));
116.         dfsnum=0;
117.
118.         memset(exist, false, sizeof(exist));
119.
120.         tarjan(1,-1);
121.         ans=solve(n);
122.         printf("%d\n",ans);
123.     }
124.     return 0;
125. }

```

顶

12

踩

0

上一篇（算法）Tarjan离线算法解决LCA问题（附POJ 1470 Closest Common Ancestors 代码）

下一篇计算几何 POJ 2826 An Easy Problem?!（线段位置判断并且求交点）

我的同类文章

Algorithm（9）

• 组合数学 容斥原理 专题

• 求解最长回文子串 Manacher算法 之 POJ 3974

• 关于字符串左移的解法

• （算法）Tarjan离线算法解决LCA问题（附POJ ...

• Tarjan求有向图的强连通分量（Tarjan算法描述）

• tarjan 离线求 lca（专题）

• 解析KMP算法

• 逆波兰表达式算法

• （算法）二分图的最大匹配（匈牙利算法）

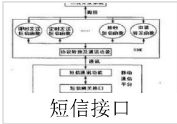
主题推荐

算法

poj


猜你在找

《C语言/C++学习指南》加解密密篇（安全相关算法） POJ 3352--Road Construction无向图增加最少的边成有趣的算法（数据结构） poj 3352 Road Construction双连通图Tarjan求至少数据结构和算法 poj 3177 & 3352 无向图双连通分量Tarjan CCIE专家讲解思科认证网络认证CCNA v2.0网络工程师 POJ 2186 Popular CowsTarjan算法求强连通分量 Java经典算法讲解 链式前向星kosarajuTarjanGabow算法的理解POJ 2186




查看评论

5楼 A_LeiQ 2015-12-30 21:20发表




挖个坟 讲的确实很清楚Orz

4楼 秋天的风 2014-07-18 20:24发表



Orz

3楼 xu245078703 2013-11-01 20:47发表



LZ 你的代码有错
测一下这个点

22 30

1 2

1 3

1 4

2 3

2 4

3 4

3 5

5 6

5 7

6 7

6 8

7 11

7 12
7 13
8 9
8 10
9 10
9 22
10 22
12 14
12 17
14 15
14 16
15 16
17 18
17 20
18 19
19 20
19 21
20 21

答案是3

Re: 小小程序员 2013-11-03 11:29发表



回复xu245078703：非常感谢你提出的样例，原来的写法确实有问题，不能使用low值标识连通分量，因此后面改成了删除桥，然后根据连通性标记出每个分量。最后根据桥统计分量的度，求出叶子节点。文章已经更新，谢谢提出问题。

2楼 softrice 2013-07-26 12:10发表



解释的很清晰。+1

1楼 kk303 2011-10-16 14:31发表



Orz....

您还没有登录,请[\[登录\]](#)或[\[注册\]](#)

* 以上用户言论只代表其个人观点，不代表CSDN网站的观点或立场

核心技术类目

全部主题 Hadoop AWS 移动游戏 Java Android iOS Swift 智能硬件 Docker
OpenStack VPN Spark ERP IE10 Eclipse CRM JavaScript 数据库 Ubuntu NFC
WAP jQuery BI HTML5 Spring Apache .NET API HTML SDK IIS Fedora XML
LBS Unity Splashtop UML components Windows Mobile Rails QEMU KDE Cassandra
CloudStack FTC coremail OPhone CouchBase 云计算 iOS6 Rackspace Web App
SpringSide Maemo Compuware 大数据 aptech Perl Tornado Ruby Hibernate ThinkPHP
HBase Pure Solr Angular Cloud Foundry Redis Scala Django Bootstrap

公司简介 | 招贤纳士 | 广告服务 | 银行汇款帐号 | 联系方式 | 版权声明 | 法律顾问 | 问题报告 | 合作伙伴 | 论坛反馈

网站客服 杂志客服 微博客服 webmaster@csdn.net 400-600-2320 | 北京创新乐知信息技术有限公司 版权所有 | 江苏乐知网络技术有限公司 提供商务支持
京 ICP 证 09002463 号 | Copyright © 1999-2014, CSDN.NET, All Rights Reserved