

个人资料



a710128

访问：5687次

积分：162

等级：

排名：千里之外

原创：10篇 转载：0篇

译文：0篇 评论：3条

文章搜索

文章分类

这个不需要分类 (5)

算法 (5)

经典问题 (1)

文章存档

2015年11月 (1)

2015年10月 (1)

2015年02月 (2)

2014年11月 (2)

2014年09月 (1)

展开

阅读排行

支配树 与 tarjan算法 (2466)

【经典】进化树问题 (575)

费用流做二分图最大权匹 (463)

Javascript 解数独 (347)

一个低调的博客建立了。 (322)

多次背包 $O(NV)$ (258)

【清单】边角知识清单 (238)

终于把 VIJOS 源代码调了 (237)

网络流 -- 流量平衡？ (222)

解方程（有点精度问题） (199)

支配树 与 tarjan算法

标签：算法 支配树

2015-11-19 15:21

2478人阅读

评

分类： 算法 (4)

版权声明：本文为博主原创文章，未经博主允许不得转载。

目录(?)

[+]

简介

什么是支配树？支配树是什么？XD

对于一张有向图（可以有环）我们规定一个起点 r （为什么是 r 呢？因为网上都是这么规定的），从 r 点到图上另一个点 w 可能存在很多条路径（下面将 r 到 w 简写为 $r \rightarrow w$ ）。

如果对于 $r \rightarrow w$ 的任意一条路径中都存在一个点 p ，那么我们称点 p 为 w 的支配点（当然这也是 $r \rightarrow w$ 的必经点），注意 r 点不讨论支配点。下面用 $idom[u]$ 表示离点 u 最近的支配点。

对于原图上除 r 外每一个点 u ，从 $idom[u]$ 向 u 建一条边，最后我们可以得到一个以 r 为根的树。这个树我们就叫它“支配树”。

相似

这个东西看上去有点眼熟？

支配点和割点（删掉后图联通块数增加）有什么区别？

我们考虑问题给定一个起点 r 和一个终点 t ，询问删掉哪个点能够使 r 无法到达 t 。

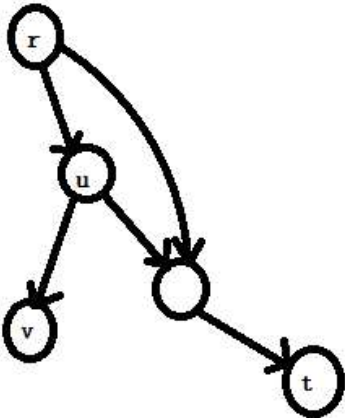
很显然，我们删掉任意一个 $r \rightarrow t$ 的必经点就能使 r 无法到达 t ，删掉任意一个非必经点， r 仍可到达 t 。

从支配树的角度来说，我们只需删掉支配树上 r 到 t 路径上的任意一点即可

从割点的角度来说，我们是不是只需要考虑所有割点，判断哪些割点在 $r \rightarrow t$ 的路径上即可？是否将某个割点删掉即可让 r 无法到达 t ？

这当然是不正确的，我们可以从两个方面来说明它的错误：

1. 删掉割点不一定使 r 无法到达 t



评论排行

- 支配树 与 tarjan算法 (1)
- 费用流做二分图最大权匹 (1)
- 【经典】进化树问题 (1)
- 【清单】边角知识清单 (0)
- 解方程（有点精度问题） (0)
- 网络流 -- 流量平衡？ (0)
- 多次背包 O(NV) (0)
- Javascript 解数独 (0)
- 终于把 VIJOS 源代码调通 (0)
- 一个低调的博客建立了。 (0)

推荐文章

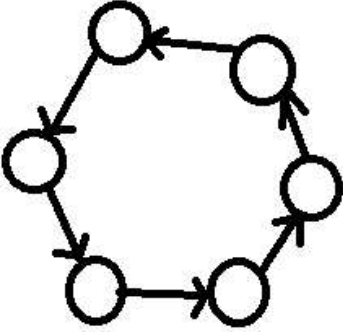
- * CSDN日报20170725——《新的开始，从研究生到入职亚马逊》
- * 深入剖析基于并发AQS的重入锁(ReentrantLock)及其Condition实现原理
- * Android版本的"Wannacry"文件加密病毒样本分析(附带锁机)
- * 工作与生活真的可以平衡吗？
- * 《Real-Time Rendering 3rd》提炼总结——高级着色：BRDF及相关技术
- * 《三体》读后思考-泰勒展开/维度打击/黑暗森林

最新评论

- 支配树 与 tarjan算法
YouSiki: 并非所有半必经点都一定是X的搜索树祖先，只有最终的那个DFS序最小的半必经点一定是其搜索树祖先。
- 费用流做二分图最大权匹配
a710128: 好像图片里把符号打反了
- 【经典】进化树问题
u010274222: 然而RQNoj上，直接把1~2，2~3.....N-1~N，N~1的和加起来除以2就能过...

这个图中点u是关键点（删掉后图联通块个数增加）并且u在r->t的路径上，然而删掉点u后r仍然可以到达t

2. 图中不一定存在割点



在这个图中不存在任何割点

所以我们没有办法使用割点来解决这个问题。

简化问题

- 树

对于一棵树，我们用r表示根节点，u表示树上的某个非根节点。很容易发现从r->u路径上的所有点都是支配点，而idom[u]就是u的父节点。

这个可以在 $O(n)$ 的时间内实现。

- DAG（有向无环图）

因为是有向无环图，所以我们可以按照拓扑序构建支配树。

假设当前我们构造到拓扑序中第x个节点编号为u，那么拓扑序中第1 ~ X-1个节点已经处理好了，考虑所有能够直接到达点u的节点，对于这些节点我们求出它们在支配树上的最近公共祖先v，这个点v就是点u在支配树上的父亲。

如果使用倍增求LCA，这个问题可以在 $O((n + m)log_2n)$ 的时间内实现。

对于这两个问题我们能够很简便的求出支配树。

有向图

对于一个有向图，我们应该怎么办呢？

简单方法

我们可以考虑每次删掉一个点，判断哪些点无法从r到达。

假设删掉点u后点v无法到达，那么点u就是r->v的必经点（点u就是v的支配点）。

这个方法我们可以非常简单的在 $O(nm)$ 的时间内实现。

其中n是点数，m是点数。

更快的方法

这里，我将介绍Lengauer-Tarjan算法。

这个算法能在很快的时间内求出支配树。

要介绍这个算法我们还需引入两个定理和一些概念

大概步骤

关闭

首先来介绍一些这个算法的大概步骤

1. 对图进行DFS（深度优先遍历）并求出搜索树和DFS序。这里我们用 $dfn[x]$ 表示点 x 在dfs序中的位置。
2. 根据半必经点定理计算出所有的半必经点作为计算必经点的根据
3. 根据必经点定理修正我们的半必经点，求出支配点

半必经点

我们用 $idom[x]$ 表示点 x 的最近支配点，用 $semi[x]$ 表示点 x 的半必经点。
那什么是半必经点呢？

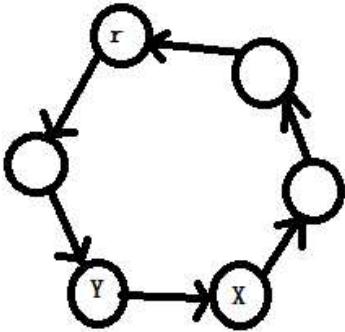
对于一个节点 Y ，存在某个点 X 能够通过一系列点 p_i （不包含 X 和 Y ）到达点 Y 且 $\forall i$
 $dfn[i] > dfn[Y]$ ，我们就称 X 是 Y 的半必经点，记做 $semi[Y] = X$

当然一个点 X 的“半必经点”会有多个，而且这些半必经点一定是搜索树中点 X 的祖先（具...，...，...
详细解释，请自行思考）。

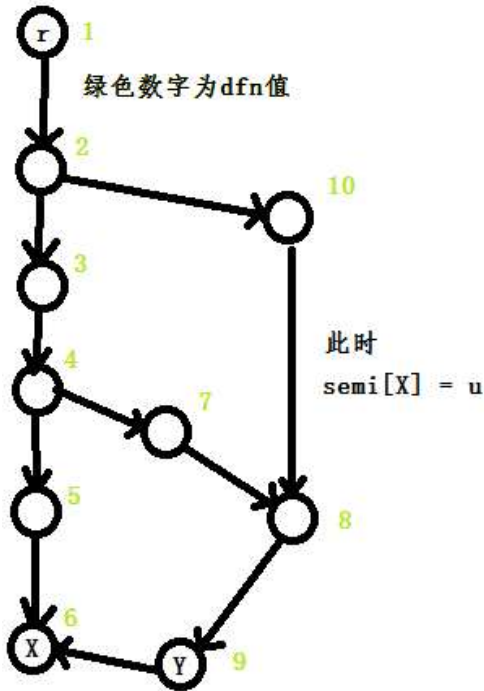
对于每个点，我们只需要保存其半必经点中 dfn 最小的一个，下文用 $semi[x]$ 表示点 x 的半必经点中 dfn 值最小的点的编号。

我们可以更书面一点描述这个定理：

- 对于一个节点 Y 考虑所有能够到达它的节点，设其中一个为 X
- 若 $dfn[X] < dfn[Y]$ ，则 X 是 Y 的一个半必经点



- 若 $dfn[X] > dfn[Y]$ ，那么对于 X 在搜索树中的祖先 Z (包括 X)，如果满足 $dfn[Z] > dfn[Y]$ 那么 $semi[Z]$ 也是 Y 的半必经点



在这些必经点中，我们仅需要 dfn 值最小的
这个半必经点有什么意义呢？

我们求出深搜树后，考虑原图中所有非树边（即不在树上的边），我们将这些边删掉，加入一些新的边 $\{(semi[w], w) | w \in V \text{ and } w \neq r\}$ ，我们会发现构建出的新图中每一个点的支配点是不变的，通过这样的改造我们使得原图变成了DAG

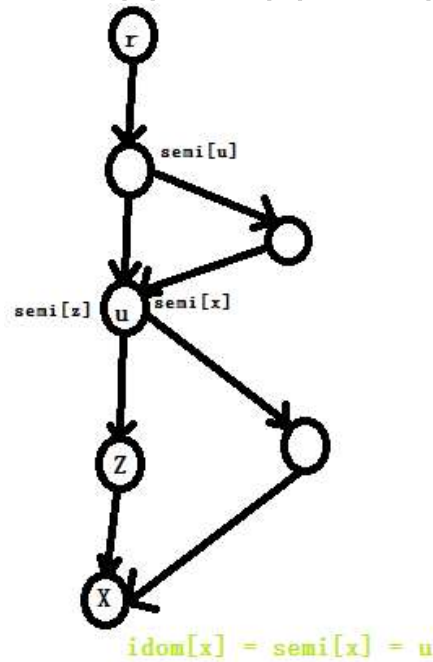
是否接下来使用DAG的做法来处理就可以做到 $n \log_2 n$ 呢？我没试过，不过我有更好的方法。

必经点

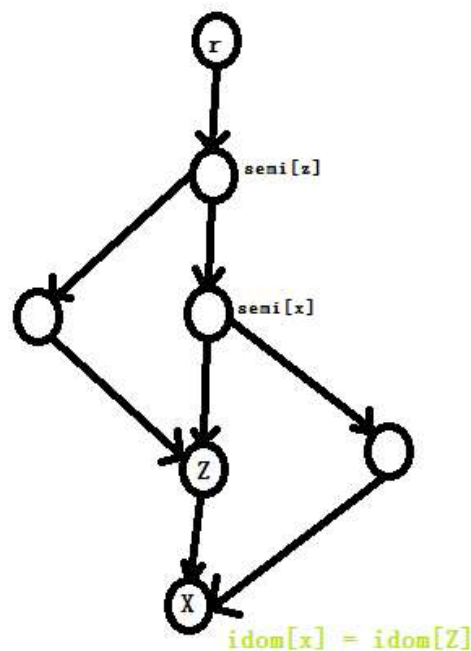
一个点的半必经点有可能是一个点的支配点，也有可能不是。我们需要使用必经点定理对这个半必经点进行修正，最后得到必经点

对于一个点 X ，我们考虑搜索树上 $semi[X]$ 到 X 路径上的所有点 $p_0, p_1, p_2, p_3 \dots$ 有 $p_i (1 \leq i < k)$ ，我们找出 $dfn[semi[p_i]]$ 最小的一个 p_i 记为 Z

- 考虑搜索树上 X 与 $semi[X]$ 之间的其他节点（即不包含 X 和 $semi[X]$ ），其中半必经点的记为 Z
- 如果 $semi[Z] = semi[X]$ ，则 $idom[X] = semi[X]$



- 如果 $semi[Z] \neq semi[X]$ ，则 $idom[X] = idom[Z]$



具体实现

对于求半必经点与必经点我们都需要处理一个问题，就是对于一个节点 X 的前驱 Y ，我们需要计算 Y 在搜索树上所有 dfn 值大于 $dfn[X]$ 的祖先中 $semi$ 值最小的一个，我们可以按 dfn 从大到小的顺序处理，使用并查集维护，这样处理到节点 X 值时所有 dfn 值比 X 大的点都被维护起来了。

对于 Y 的所有满足条件的祖先，就是在并查集中 Y 的祖先，可以通过带权并查集的方法，维护祖先中的最小值，并记下 $semi$ 最小的具体是哪个节点。

这样我们就能够在 $O((n + m) \times \alpha(n))$ 时间内解决这个问题。

代码

这里提供 Codechef May15 中 GRAPHCNT 题目的代码

```
1  #include <iostream>
2  #include <cstdio>
3  using namespace std;
4  typedef long long lld;
5  const int MaxN = 100000 + 10, MaxE = (5 * 100000) * 2 + MaxN;
6  int head[MaxN], pre[MaxN], dom[MaxN], to[MaxE], nxt[MaxE], top;
7  void addedge(int *h, int fr, int tt)
8  {
9      top ++;
10     nxt[top] = h[fr];
11     to[top] = tt;
12     h[fr] = top;
13 }
14 int n, m;
15 void init()
16 {
17     scanf("%d%d", &n, &m);
18     int a, b;
19     for(int i = 1; i <= m; ++i)
20     {
21         scanf("%d%d", &a, &b);
22         addedge(head, a, b);
23         addedge(pre, b, a);
24     }
25 }
26 int bcj[MaxN], semi[MaxN], idom[MaxN], best[MaxN], dfn[MaxN], id[MaxN], fa[MaxN], dfs_clock;
27 int push(int v)
28 {
29     if(v == bcj[v]) return v;
30     int y = push(bcj[v]);
31     if(dfn[semi[best[bcj[v]]]] < dfn[semi[best[v]]]) best[v] = best[bcj[v]];
32     return bcj[v] = y;
33 } //带权并查集路径压缩
34 void dfs(int rt)
35 {
36     dfn[rt] = ++dfs_clock;
37     id[dfs_clock] = rt;
38     for(int i = head[rt]; i; i = nxt[i])
39         if(!dfn[to[i]])
40         {
41             dfs(to[i]);
42             fa[to[i]] = rt;
43         }
44 }
45 } //求出dfs序
46 void tarjan()
47 {
48     for(int i = dfs_clock, u; i >= 2; --i)
49         { //按dfs序从大到小计算
50             u = id[i];
51             for(int j = pre[u]; j; j = nxt[j]) //semi
52             {
53                 if(!dfn[to[j]]) continue;
```

```

54         push(to[j]);
55         if(dfn[semi[best[to[j]]]] < dfn[semi[u]]) semi[u] = semi[best[to[j]]];
56     }
57     addedge(dom, semi[u], u);
58     bcj[u] = fa[u]; u = id[i - 1];
59     for(int j = dom[u]; j; j = nxt[j])//idom
60     {
61         push(to[j]);
62         if(semi[best[to[j]]] == u) idom[to[j]] = u;
63         else idom[to[j]] = best[to[j]];
64     }
65 }
66 for(int i = 2, u; i <= dfs_clock; ++i)
67 {
68     u = id[i];
69     if(idom[u] != semi[u]) idom[u] = idom[idom[u]];
70 }
71 }
72 int sons[MaxN];
73 lld ans;
74 void calc_son()
75 {
76     for(int i = dfs_clock, u; i >= 2; --i)
77     {
78         u = id[i];
79         ++ sons[u];
80         if(idom[u] != 1) sons[idom[u]] += sons[u];
81         else ans -= ((lld)sons[u] * (lld)(sons[u] - 1)) / 211;
82     }
83 }
84 void solve()
85 {
86     for(int i = 1; i <= n; ++i) bcj[i] = semi[i] = best[i] = i;
87     dfs_clock = 0;
88     dfs(1);
89     tarjan();
90     ans = ((lld)dfs_clock * (lld)(dfs_clock - 1)) / 211;
91     calc_son();
92     cout << ans << endl;
93 }
94 int main()
95 {
96     init();
97     solve();
98     return 0;
99 }

```

顶 1 踩 0

[上一篇](#) [费用流做二分图最大权匹配](#)

相关文章推荐

- the hdu
- 快速构造支配树的Lengauer-Tarjan算法
- Tarjan算法求有向图强连通分量
- 在流程图中求支配点的一种快速算法+[CodeChef ...
- 强连通分量算法之——Tarjan
- hdu4694 Important Sisters 支配树
- LCA离线算法Tarjan (2) 案例1——求二叉树中节...
- 【算法】【树】最近公共祖先LCA——Tarjan算法

- Codechef GRAPHCNT 支配树学习及tarjan算法求解
- Tarjan算法求强连通分量

猜你在找

【直播】机器学习&数据挖掘7周实训—韦玮

【直播】3小时掌握Docker最佳实战—徐西宁

【直播】计算机视觉原理及实战—屈教授

【直播】机器学习之矩阵—黄博士

【直播】机器学习之凸优化—马博士

【套餐】系统集成项目管理工程师顺利通关—徐朋

【套餐】机器学习系列套餐（算法+实战）—唐宇迪

【套餐】微信订阅号+服务号Java版 v2.0—翟东平

【套餐】微信订阅号+服务号Java版 v2.0—翟东平

【套餐】Javascript 设计模式实战—曾亮

查看评论

1楼 [YouSiki](#) 2017-03-06 10:21发表



并非所有半必经点都一定是X的搜索树祖先，只有最终的那个DFS序最小的半必经点一定是其搜索树祖先。

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

公司简介 | 招贤纳士 | 广告服务 | 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 400-660-0108 | 北京创新乐知信息技术有限公司 版权所有 | 江苏知之为计算机有限公司 |

江苏乐知网络技术有限公司

京 ICP 证 09002463 号 | Copyright © 1999-2017, CSDN.NET, All Rights Reserved 