Vendetta Blogs

JVxie的个人博客

昵称：JVxie
园龄：4年8个月
粉丝：29
关注：0
+加关注

< 2020年6月 >

日	一	二	三	四	五	六
31	1	2	3	4	5	6
7	8	9	10	11	12	13
14	15	16	17	18	19	20
21	22	23	24	25	26	27
28	29	30	1	2	3	4
5	6	7	8	9	10	11

搜索

找找看

谷歌搜索

随笔分类

NOIP游记 by JVxie(2)
经典算法 by JVxie(4)
数据结构 by JVxie(2)
题解思想 by JVxie(5)

Friends

AYOJ(原METO CODE)
skida
cytaz
hello,world
ifAandBisC
METO

首页 □□□ 联系 管理 随笔- 9 文章- 0 评论- 58

最近公共祖先**LCA(Tarjan**算法)的思考和算法实现

LCA 最近公共祖先

Tarjan(离线)算法的基本思路及其算法实现

小广告：**METO CODE** 安溪一中信息学在线评测系统**(OJ)**

//由于这是第一篇博客..有点瑕疵...比如我把**false**写成了**flase**...看的时候注意一下!

//还有...这篇字比较多 比较杂....毕竟是第一次嘛 将就将就 后面会重新改！！！

首先是最近公共祖先的概念(什么是最近公共祖先？)：

在一棵没有环的树上，每个节点肯定有其父亲节点和祖先节点，而最近公共祖先，就是两个节点在这棵树上深度最大的公共的祖先节点。

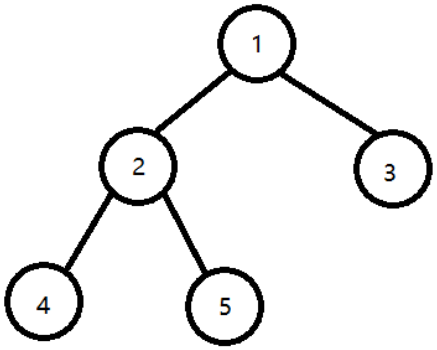
换句话说，就是两个点在这棵树上距离最近的公共祖先节点。

所以LCA主要是用来处理当两个点仅有唯一一条确定的最短路径时的路径。

有人可能会问：那他本身或者其父亲节点是否可以作为祖先节点呢?

答案是肯定的，很简单，按照人的亲戚观念来说，你的父亲也是你的祖先，而LCA还可以将自己视为祖先节点。

举个例子吧，如下图所示 4 和 5 的最近公共祖先是 2， 5 和 3 的最近公共祖先是 1， 2 和 1 的最近公共祖先是 1 。



这就是最近公共祖先的基本概念了，那么我们该如何去求这个最近公共祖先呢？

通常初学者都会想到最简单粗暴的一个办法：对于每个询问，遍历所有的点，时间复杂度为**O(n*q)**，很明显，**n**和**q**一般不会很小。

常用的求LCA的算法有：Tarjan/DFS+ST/倍增

后两个算法都是在线算法，也很相似，时间复杂度在**O(logn)~O(nlogn)**之间，我个人认为较难理解。

有的题目是可以用线段树来做的，但是其代码量很大，时间复杂度也偏高，在**O(n)~O(nlogn)**之间，优点在于也是简单粗暴。

这篇博客主要是要介绍一下Tarjan算法(其实是我不会在线...)。

什么是Tarjan(离线)算法呢？顾名思义，就是在一次遍历中把所有询问一次性解决，所以其时间复杂度是**O(n+q)**。

Tarjan算法的优点在于相对稳定，时间复杂度也比较居中，也很容易理解。

下面详细介绍一下Tarjan算法的基本思路：

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**已经被访问过了，则可以确认**u**和**v**的最近公共祖先为**v**被合并到的父亲节点**a**。

遍历的话需要用到**dfs**来遍历(我相信来看的人都懂吧...)，至于合并，最优化的方式就是利用并查集来合并两个节点。

下面上伪代码：

```
Tarjan(u)//marge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v
    {
        Tarjan(v);          //继续往下遍历
        marge(u,v);     //合并v到u上
        标记v被访问过;
    }
    for each(u,e)     //访问所有和u有询问关系的e
    {
        如果e被访问过;
        u,e的最近公共祖先为find(e);
    }
}
```

个人感觉这样还是有很多人不太理解，所以我打算模拟一遍给大家看。

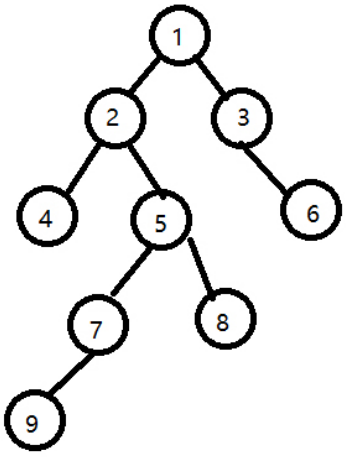
建议拿着纸和笔跟着我的描述一起模拟！！

假设我们有一组数据 9个节点 8条边 联通情况如下：

1--2，1--3，2--4，2--5，3--6，5--7，5--8，7--9 即下图所示的树

设我们要查找最近公共祖先的点为9--8，4--6，7--5，5--3；

设f[]数组为并查集的父亲节点数组，初始化f[i]=i，vis[]数组为是否访问过的数组，初始为0;



f[1]=1; vis[1]=flase;
f[2]=2; vis[2]=flase;
f[3]=3; vis[3]=flase;
f[4]=4; vis[4]=flase;
f[5]=5; vis[5]=flase;
f[6]=6; vis[6]=flase;
f[7]=7; vis[7]=flase;
f[8]=8; vis[8]=flase;
f[9]=9; vis[9]=flase;

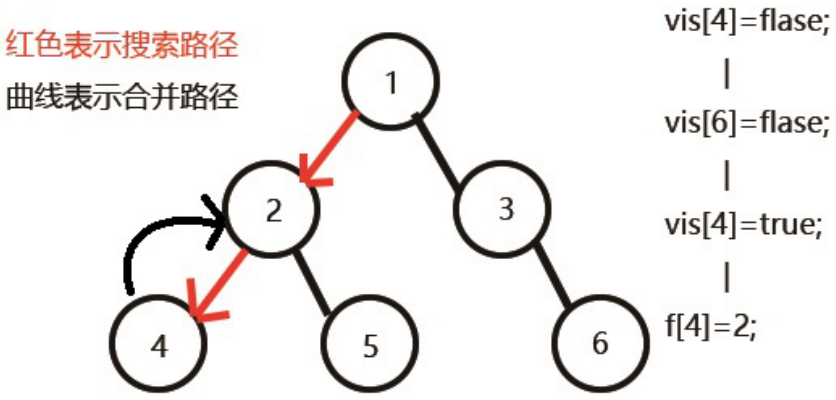
下面开始模拟过程：

取1为根节点，往下搜索发现有两个儿子2和3；

先搜2，发现2有两个儿子4和5，先搜索4，发现4没有子节点，则寻找与其有关系的点；

发现6与4有关系，但是**vis[6]=0**，即6还没被搜过，所以不操作；

发现没有和4有询问关系的点了，返回此前一次搜索，更新**vis[4]=1**；



表示4已经被搜完，更新**f[4]=2**，继续搜**5**，发现5有两个儿子7和8；

先搜**7**，发现7有一个子节点9，搜索**9**，发现没有子节点，寻找与其有关系的点；

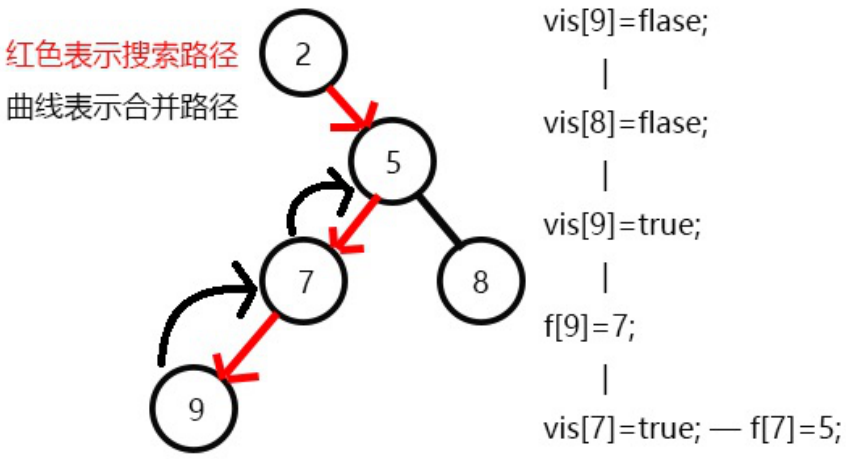
发现8和9有关系，但是**vis[8]=0**,即8没被搜到过，所以不操作；

发现没有和9有询问关系的点了，返回此前一次搜索，更新**vis[9]=1**；

表示9已经被搜完，更新**f[9]=7**，发现7没有没被搜过的子节点了，寻找与其有关系的点；

发现5和7有关系，但是**vis[5]=0**，所以不操作；

发现没有和7有关系的点了，返回此前一次搜索，更新**vis[7]=1**；



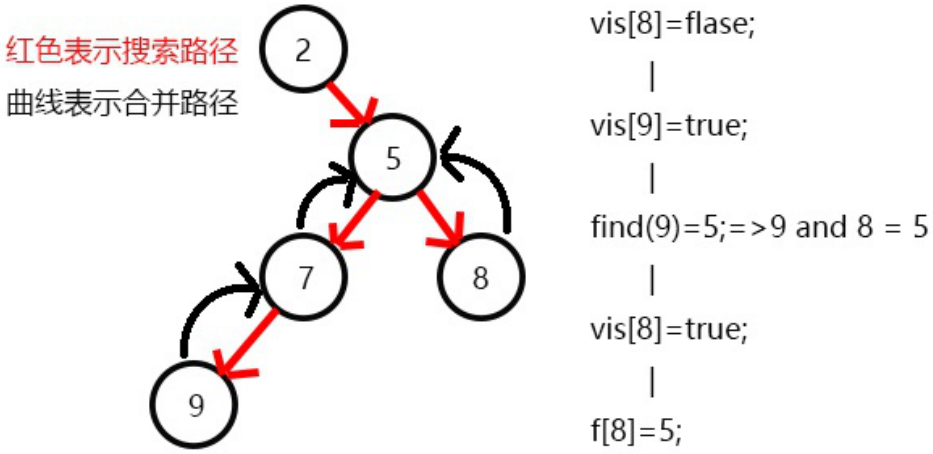
表示7已经被搜完，更新**f[7]=5**，继续搜**8**，发现8没有子节点，则寻找与其有关系的点；

发现9与8有关系，此时**vis[9]=1**，则他们的最近公共祖先为**find(9)=5**；

(find(9)的顺序为f[9]=7-->f[7]=5-->f[5]=5 return 5;)

发现没有与8有关系的点了，返回此前一次搜索，更新**vis[8]=1**；

表示8已经被搜完，更新**f[8]=5**，发现5没有没搜过的子节点了，寻找与其有关系的点；



发现7和5有关系，此时**vis[7]=1**，所以他们的最近公共祖先为**find(7)=5**；

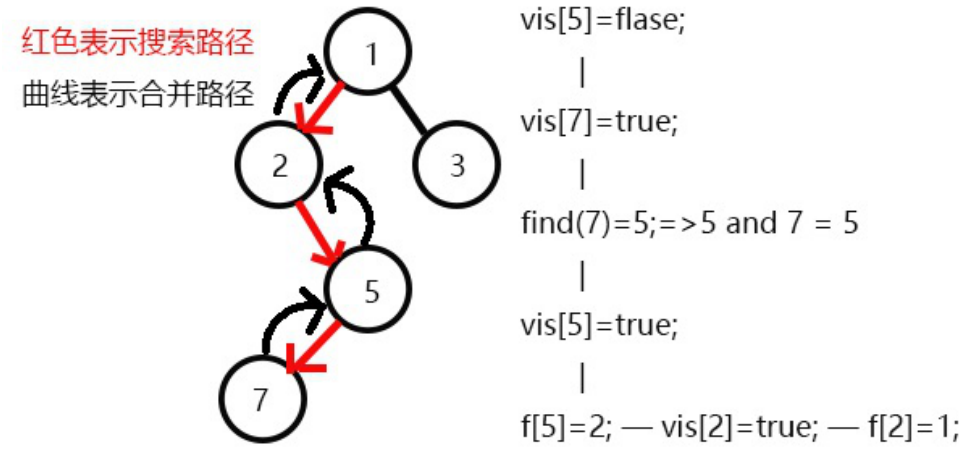
(find(7)的顺序为f[7]=5-->f[5]=5 return 5;)

又发现5和3有关系，但是**vis[3]=0**，所以不操作，此时5的子节点全部搜完了；

返回此前一次搜索，更新**vis[5]=1**，表示5已经被搜完，更新**f[5]=2**；

发现2没有未被搜完的子节点，寻找与其有关系的点；

又发现没有和2有关系的点，则此前一次搜索，更新**vis[2]=1**；



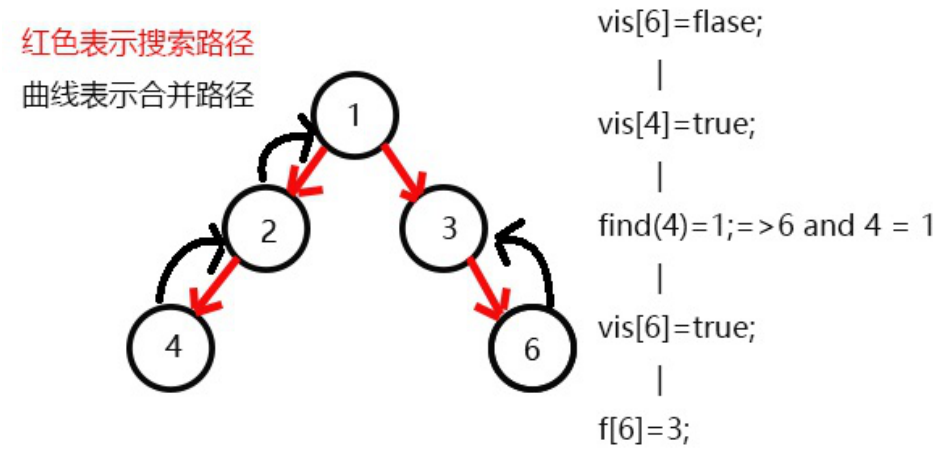
表示2已经被搜完，更新**f[2]=1**，继续搜**3**，发现3有一个子节点6；

搜索**6**，发现6没有子节点，则寻找与6有关系的点，发现4和6有关系；

此时**vis[4]=1**，所以它们的最近公共祖先为**find(4)=1**；

(find(4)的顺序为f[4]=2-->f[2]=2-->f[1]=1 return 1;)

发现没有与6有关系的点了，返回此前一次搜索，更新**vis[6]=1**，表示6已经被搜完了；

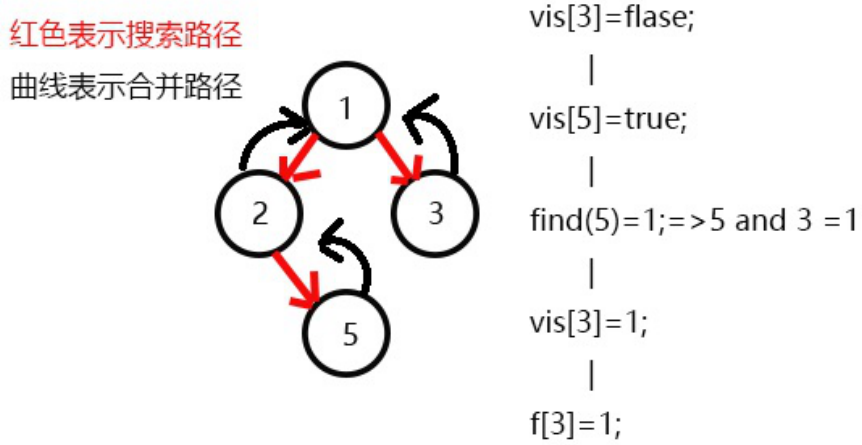


更新**f[6]=3**，发现3没有没被搜过的子节点了，则寻找与3有关系的点；

发现5和3有关系，此时**vis[5]=1**，则它们的最近公共祖先为**find(5)=1**；

(find(5)的顺序为f[5]=2-->f[2]=1-->f[1]=1 return 1;)

发现没有和3有关系的点了，返回此前一次搜索，更新**vis[3]=1**；



更新**f[3]=1**，发现1没有被搜过的子节点也没有有关系的点，此时可以退出整个**dfs**了。

经过这次dfs我们得出了所有的答案，有没有觉得很神奇呢？是否对Tarjan算法有更深层次的理解了呢？

如果有什么不懂可以在下面 留言提问 **or** 发送问题到**1136404654@qq.com**。

推荐几道**LCA**的题目

CODEVS 2370 小机房的树 传送门

CODEVS 1036 商务旅行 传送门

METO CODE 223 拉力赛 传送门

HDU 2586 How far way? 传送门

ZOJ 3195 Design the city 传送门

相应的题解以后可能会上，大家敬请期待吧。

分类: 经典算法 by JVxie

标签: LCA, Tarjan, 最近公共祖先, 公共祖先

JVxie
关注 - 0
粉丝 - 29
+加关注

» 下一篇： 基本数据结构——堆(Heap)的基本概念及其操作

posted @ 2015-10-04 16:57 JVxie 阅读(51678) 评论(52) 编辑 收藏

评论

#51楼 [楼主] 2020-04-15 02:27 | JVxie 回复 引用

@whosyourdaddyy
你可以查一下并查集时间复杂度的，是一个常数，接近o（1）

支持(0) 反对(0)

#52楼 2020-04-15 11:52 | whosyourdaddyy 回复 引用

@JVxie
好的，谢谢博主

支持(0) 反对(0)

刷新评论 刷新页面 返回顶部

发表评论

编辑 预览

提交评论 退出 订阅评论

[Ctrl+Enter快捷键提交]

相关博文:
· 最近公共祖先LCA Tarjan算法
· LCA 最近公共祖先
· 最近公共祖先LCA Tarjan 离线算法
· 最近公共祖先(LCA)的Tarjan算法
· lca最近公共祖先（模板）
» 更多推荐...

最新 **IT** 新闻:
· 《率土之滨》大型抗议活动之下，网易游戏帝国的盛世危言
· 有钱就放到余额宝里的人，这习惯恐怕要改一改了
· 丁磊首次直播选了快手，“老铁们”抢走2W个乳胶枕
· 王中军卖房，黄巍坠楼，但影院至暗时刻并非在疫情中才降临
· 罗永浩直播卖考拉黑卡 网友吐槽“你变了”
» 更多新闻...