Styx 的博客

初探tarjan算法（求强连通分量）

2018-07-23 20:33:32

51

> “tarjan陪伴强联通分量
> 生成树完成后思路才闪光
> 欧拉跑过的七桥古塘
> 让你 心驰神往”---《膜你抄》

tarjan是一种求强连通分量、双连通分量的常用算法，其拓展例如求缩点、割点、割桥以及2-SAT等都是非常实用的（tarjan orz）

所以大概写这篇博客的目的就是简单介绍一下这个算法，至于2-SAT什么的……咳咳，你说什么？我没听见 ╮(´∀`)╭

一、tarjan求强连通分量

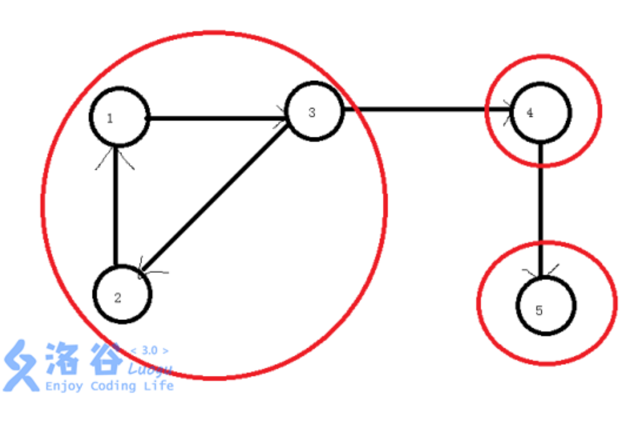
1.什么是强连通分量？

引用来自度娘的一句话：

> “有向图强连通分量：在有向图G中，如果两个顶点vi,vj间（vi>vj）有一条从vi到vj的有向路径，同时还有一条从vj到vi的有向路径，则称两个顶点强连通(strongly connected)。如果有向图G的每两个顶点都强连通，称G是一个强连通图。有向图的**极大**强连通子图，称为强连通分量(strongly connected components)。”

没准会一脸懵逼，不过仔细想想是可以理解的

反正就是在图中找到一个最大的图，使这个图中每个两点都能够互相到达。这个最大的图称为强连通分量，同时一个点也属于强连通分量。

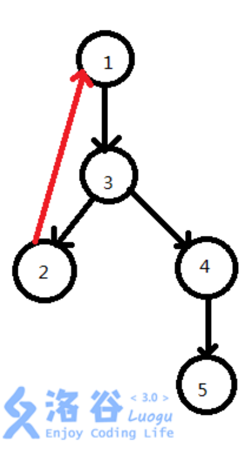


如图中强连通分量有三个：1-2-3,4,5

2.强连通分量怎么求？

噫……很明显，通过肉眼可以很直观地看出1-2-3是一组强连通分量，但很遗憾，机器并没有眼睛，所以该怎么判断强连通分量呢？

如果仍是上面那张图，我们对它进行dfs遍历。



可以注意到红边非常特别，因为如果按照遍历时间来分类的话，其他边都指向在**自己之后**被遍历到的点，而红边指向的则是**比自己先**被遍历到的点。

如果存在这么一条边，那么我们可以yy一下

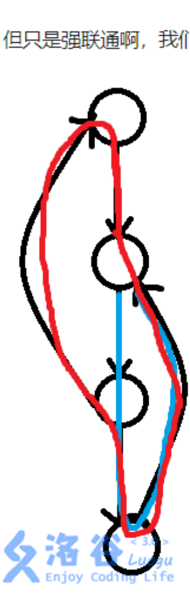
从一个点出发，一直向下遍历，然后忽得找到一个点，那个点竟然有条指回这一个点的边！

那么想必这个点能够从自身出发再回到自身

想必这个点和其他向下遍历的该路径上的所有点构成了一个环，

想必这个环上的所有点都是**强联通**的。

但只是强联通啊，我们需要求的可是强连通分量啊……



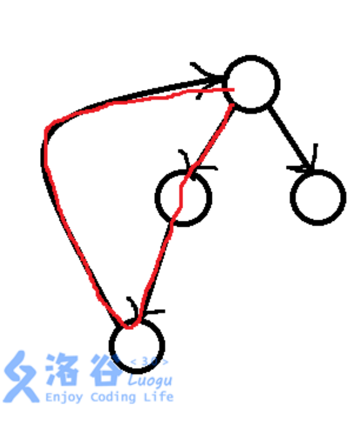
比如说图中红色为**强连通分量**，而蓝色只是**强联通图**

因此我们只需要知道这个点u下面的所有子节点**有没有连着这个点的祖先**就行了。

但似乎还有一个问题啊……

我们怎么知道这个点u它下面的所有子节点一定是都与他强联通的呢？

这似乎是不对的，这个点u之下的所有点不一定都强联通



那么怎么在退回到这个点的时候，知道所有和这个点u构成强连通分量的点呢？

开个栈记录就行了

什么？！这么简单？

没错~就是这么简单~

如果在这个点之后被遍历到的点已经能与其下面的一部分点（也可能就只有他一个点）已经构成强连通分量，即它已经是最大的。

那么把它们**一起从栈里弹出来**就行了。

所以最后处理到点u时如果u的子孙没有指向其祖先的边，那么它之后的点肯定**都已经处理好**了，一个常见的思想，可以理解一下。

所以就可以保证栈里留下来u后的点都是能与它构成强连通分量的。

似乎做法已经明了了，用程序应该怎么实现呢？

3.tarjan求强连通分量的程序实现

首先需要介绍一些辅助数组

> (1)、dfn[ ]，表示这个点在dfs时是**第几个被搜到的。**
> (2)、low[ ]，表示这个点以及其子孙节点连的所有点中**dfn最小的值**
> (3)、stack[ ]，表示当前**所有可能能构成强连通分量的点。**
> (4)、vis[ ]，表示一个点是否在stack[ ]数组中。

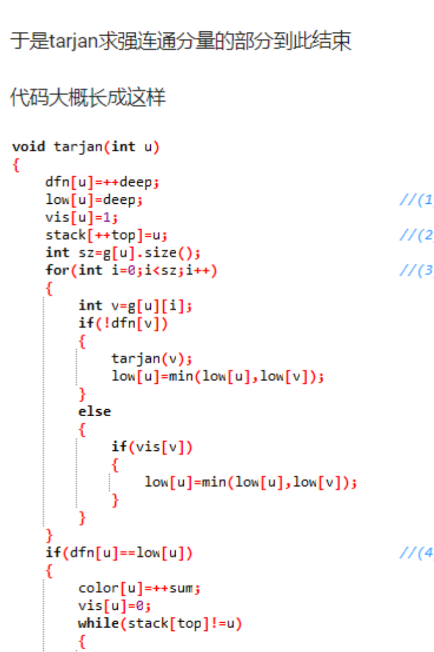
那么按照之上的思路，我们来考虑这几个数组的用处以及算法的具体过程。

假设现在开始遍历点u：

1. **首先初始化dfn[u]=low[u]=第几个被dfs到**
   dfn可以理解，但为什么low也要这么做呢？
   因为low的定义如上，也就是说如果没有子孙与u的祖先相连的话，dfn[u]一定是它和它的所有子孙中dfn最小的（因为**它的所有子孙一定比他后搜到**）。
2. **将u存入stack[ ]中，并将vis[u]设为true**
   stack[ ]有什么用？
   如果u在stack中，u之后的所有点在u被回溯到时u和栈中所有在它之后的点都构成强连通分量。(也就是上文中所说的开个栈记录)
3. **遍历u的每一个能到的点，如果这个点dfn[ ]为0，即仍未访问过，那么就对点v进行dfs，然后low[u]=min{low[u],low[v]}** low[ ]有什么用？
   应该能看出来吧，就是记录一个点它最大能连通到哪个祖先节点（当然包括自己）
   如果遍历到的这个点已经被遍历到了，那么看它当前有没有在stack[ ]里,如果有那么low[u]=min{low[u],low[v]}
   如果已经被弹掉了，说明无论如何这个点也不能与u构成强连通分量，因为它不能到达u
   如果还在栈里，说明这个点肯定能到达u，同样u能到达他，他俩强联通。
4. **假设我们已经dfs完了u的所有的子树，那么之后无论我们再怎么dfs，u点的low值已经不会再变了。**
   那么如果dfn[u]=low[u]这说明了什么呢？
   再结合一下dfn和low的定义来看看吧
   dfn表示u点被dfs到的时间，low表示u和u所有的子树所能到达的点中dfn最小的。
   这说明了u点及u点之下的所有子节点没有边是指向u的祖先的了，即我们之前说的u点与它的子孙节点构成了一个最大的强连通图即强连通分量
   此时我们得到了一个强连通分量，把所有的u点以后压入栈中的点和u点一并弹出，将它们的vis[ ]置为false，如有需要也可以给它们染上相同颜色（后面会用到）

于是tarjan求强连通分量的部分到此结束

代码大概长成这样



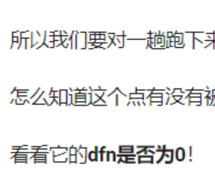
```
void tarjan(int u)
{
    dfn[u]=++deep;
    low[u]=deep;                    //(1)
    vis[u]=1;
    stack[++top]=u;                 //(2)
    int sz=g[u].size();
    for(int i=0;i<sz;i++)           //(3)
    {
        int v=g[u][i];
        if(!dfn[v])
        {
            tarjan(v);
            low[u]=min(low[u],low[v]);
        }
        else
        {
            if(vis[v])
            {
                low[u]=min(low[u],low[v]);
            }
        }
    }
    if(dfn[u]==low[u])              //(4)
    {
        color[u]=++sum;
        vis[u]=0;
        while(stack[top]!=u)
        {
            color[stack[top]]=sum;
            vis[stack[top--]]=0;
        }
        top--;
    }
}
```

对了，tarjan一遍不能搜完所有的点，因为**存在孤立点**或者其他

所以我们要对一趟跑下来还没有被访问到的点继续跑tarjan

怎么知道这个点有没有被访问呢？

看看它的**dfn是否为0**！



```
for(int i=1;i<=n;i++)
{
    if(!dfn[i])
    {
        tarjan(i);
    }
}
```

~~好的，谢谢各位观看~~

等等，还没完呢QAQ，我们还没有证过复杂度啊

4.非常简短的tarjan复杂度证明

思考每个点最多被dfs一次，所以均摊下来复杂度是**O(n)**的

证毕

二、tarjan缩点

1.什么时候要用缩点

众所周知，有向无环图总是有着一些蜜汁优越性，因为没有环，你可以放心的在上面跑dfs，搞DP，但如果是一张有向有环图，事情就会变得尴尬起来了

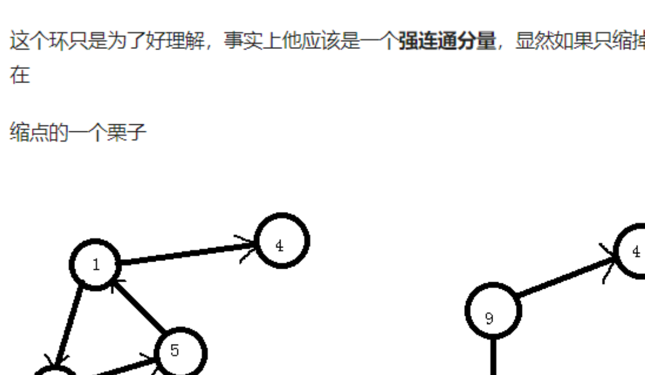
思考一下会发现如果不打vis标记就会t飞（一直在环里绕啊绕），但是如果打了，又不一定能保证最优解

而你一看题目却发现显然根据一些贪心的原则，这个环上每个点的最大贡献都是整个环的总贡献

这个时候缩点就显得很有必要了，因为单个点的贡献和整个环相同，为什么不去把整个环缩成一个超级点呢？

这个环只是为了好理解，事实上他应该是一个**强连通分量**，显然如果只缩掉一个强连通图，图中仍然有环存在

缩点的一个栗子



2.怎么缩点

还记得之前tarjan里的染色吗？

我们只需要把同一颜色的点权加到一块，然后把该颜色指向不同颜色的边建好就可以了

代码就不贴了，因为不同的题有不同的处理方法

三、tarjan求割点

1.什么是割点

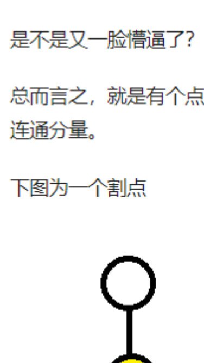
再次祭出度娘

> 在一个无向图中，如果有一个顶点集合，删除这个顶点集合以及这个集合中所有顶点相关联的边以后，图的连通分量增多，就称这个点集为割点集合。

是不是又一脸懵逼了？

总而言之，就是有个点维持着连通分量的继续，去掉那个点，这个连通分量就无法在维持下去，分成好几个连通分量。

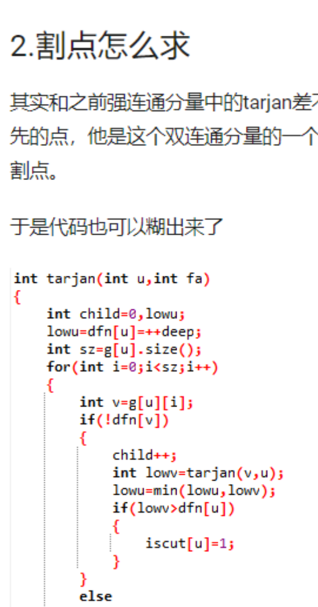
下图为一个割点



2.割点怎么求

其实和之前强连通分量中的tarjan差不多了，如果这个点的dfn比low要小，说明他的子树中没有能够到达他祖先的点，他是这个双连通分量的一个割点，但要加一个特判，根节点如果有两个及以上的儿子，那么他也是割点。

于是代码也可以糊出来了



```
int tarjan(int u,int fa)
{
    int child=0,lowu;
    lowu=dfn[u]=++deep;
    int sz=g[u].size();
    for(int i=0;i<sz;i++)
    {
        int v=g[u][i];
        if(!dfn[v])
        {
            child++;
            int lowv=tarjan(v,u);
            lowu=min(lowu,lowv);
            if(lowv>dfn[u])
            {
                iscut[u]=1;
            }
        }
        else
        {
            if(v!=fa&&dfn[v]<dfn[u])
            {
                lowu=min(lowu,dfn[v]);
            }
        }
    }
    if(fa<0&&child==1)
    {
        iscut[u]=false;
    }
    low[u]=lowu;
    return lowu;
}
```

四、一些例（shui）题

大概就是一些模板题吧，而且为了页面整洁，只有思路，没有代码

1. 洛谷P2863 牛的舞会
   用tarjan数出强连通分量的个数，给每个强连通分量的点染色，统计出每个强连通分量中点的个数，如果大于一，则答案加一
2. poj2186 Popular Cows
   显然一个强联通分量内的所有点都是满足条件的，我们可以对整张图进行缩点，然后就简单了。
   剩下的所有点都不是强连通的，现在整张图就是一个DAG（有向无环图）
   那么就变成一道水题了，因为这是一个有向无环图，不存在所有点的出度都不为零的情况。
   所以必然有1个及以上的点出度为零，如果有两个点出度为零，那么这两个点肯定是不相连的，即这两圈牛不是互相崇拜的，于是此时答案为零，如果有1个点出度为0，那么这个点就是被全体牛崇拜的，
   这个点可能是一个强联通分量缩成的超级点，所以应该输出整个强联通分量中点的个数。
3. 洛谷P3387 模板 缩点
   显然把每个强连通分量缩成一个点，权值就是强连通分量中所有点值之和，接着就可以跑dfs了，加个记忆化搜索复杂度是O(n)的
4. 洛谷P3388 模板 割点
   就是求割点的个数和位置

所以就讲到这里了~

结束

评论

前排
By Unknown_Error　At 2018-07-24 16:34:33

tarjan创造了原始对偶算法了解一下
By arfa　At 2018-07-24 19:44:15

%%%
By ilem　At 2018-07-27 11:17:07

~~前排兜售小面包，小闸蟹，大绿鸟~~
By Anguei　At 2018-08-09 23:59:36

前排兜售小面包
By 靛莲　At 2018-08-10 09:14:27

前排
By henrytb　At 2018-08-10 09:25:14

建议用上LaTeX
By Xeonacid　At 2018-08-13 15:03:35

前排
By little_gift　At 2018-08-13 15:08:17

前排
By Taday_Bule_Rainbow　At 2018-08-13 15:09:26

tarjin很多
By かねき　けん　At 2018-08-13 15:17:38

1 2 3 > >|