PS10--201300086 史浩男

1、DFS应用：深搜

思路：

- 以s为root开始DFS，忽略所有t的出度
- 从s出发到t的路径数=从s的所有孩子v出发到t的路径数之和

正确性：

- 因为是无圈图，从s出发不可能回到s

```
DFSAll(G):
for (each node u)
  u.color = WHITE
  u.count=0//记录从u到目标t的路径数
DFS(G,s)
return s.count
```

```
DFS(G,s):
for (each edge (s,v) in E)
    if(v==t)
        s.count=s.count+1
    else if (v.color == BLACK)//v出发的路径数已经统计完成，直接加上去
        s.count=s.count+v.count
    else if (v.color == WHITE)
        DFS(G,v)
s.color = BLACK
```

时间

- 对小于|V|个点进行一次DFS，O（|V|+|E|）

2、强连通分支SCC

思路

- 为保证E最小，且SCC不变，则在每个SCC中取一个代表点即可
- 步骤一：计算SCC，每个SCC用一个点替代，得到G1
- 步骤二：对G1进行DFS，保留所有点和所有树边，删除其他边

步骤一伪代码

```
STRONGLY-CDNNECTED-COMPONENTS(G)
//结点分类，给每个分量中的结点赋予一个相同的SCC标号:u.SCC
for(each SCC Ci in G)//遍历每个SCC
    for(each vertex u in Ci)//遍历SCC内的每个点
        for(each (u,v) in E)
            if(v.SCC!=u.SCC)//遍历SCC内点的出度边
                add (g_u.SCC,g_v.SCC) to G1 //在G1中添加有向边
```

时间：每条边和点都遍历常数次，O（|V|+|E|）

步骤二伪代码

```
DFSAll(G1):
for (each node u)
  u.color = WHITE
for (each node u)
  if (u.color == WHITE)
      v=ancestor(u)
      DFS(G1,v)
return G2
```

```
ancestor(u)://返回u的源点祖先
    while(there is v and (v,u) in E)
        u=v
    return v
```

```
DFS(G,s):
s.color = GRAY
for (each edge (s,v) in E)
  if (v.color == WHITE)
    //初始的G2只有G1中所有点，没有边
    add (s,v) to G2//只有树边保留在G2中
    DFS(G,v)
s.color = BLACK
```

时间分析：

比原始的DFS只多了一步ancestor操作，由于这是在判断u的颜色后才决定是否进行的，所以所有ancestor操作的最大开销为O（|E|）
因此总时间仍为O（|V|+|E|）

3、市长吹牛皮

（a）牛皮1:任意点可达任意点

初步分析

- 把交叉口当作点，单行路段当作有向边
- 先假设市长吹的牛皮1是真的
- 依次证明以下命题

命题1：一定没有源点和汇点

> 这是因为：没有点可以到达源点，从汇点出发哪都去不了

命题2：从任意点出发，一定存在路径可以回到这个点

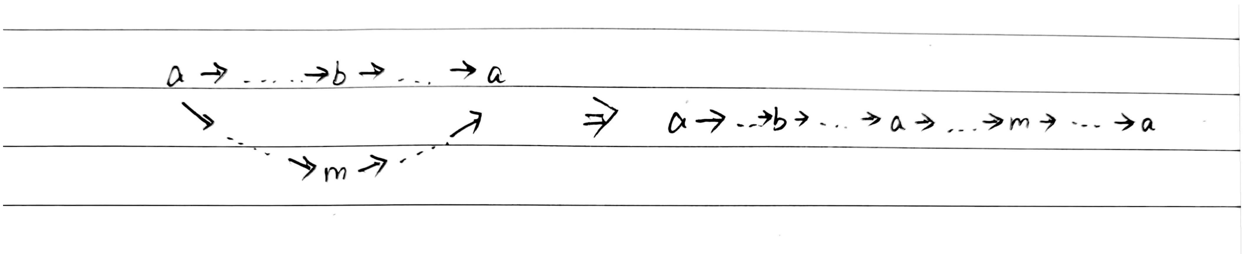
> 反证法：假设从v出发，不存在路径可以回到v
> 设点集A表示存在路径通往v的所有点，点集B表示从v出发可以到达的所有点
> 首先，A与B非空，否则v是源点或汇点
> 其次，A与B交集为空，否则一定存在从v回到v的路径
> 因此，从任意一个B中的点出发，都无法到达A中任意一个点，否则，一定存在从v回到v的路径
> 而这与市长吹的牛皮矛盾，所以假设不成立，命题2成立

命题3：如果存在从任意点出发回到自身的路径，其中经过了n个不同的点，则牛皮为真

> 这说明，有向图中存在一个允许有重复结点的圈，这个圈包含了n个不同的点
> 则牛皮显然成立，从任意点出发都能到达任意点

命题4：如果存在从任意点出发回到自身的路径最多只经历了小于n个点，则牛皮为假

> 如下图所示，假设包含最多点的返回自身的路径为a->.......->b->.......->a
> 设m为不包含在这一路径中的结点
> 由于从a出发可以到达m，从m出发可以到达a，则存在路径a->....->m->......->a
> 则可以合并为一条更长的路径如图，包含了更多结点，与假设中最多矛盾，因此命题4成立



验证方案

基于以上命题，线性时间解决方案如下：

- 先把所有点染为白色...........O(|V|)
- 从任意点u出发DFS，每条路径以回到u为结束
- 每经过一个白色点，就染为黑色
- DFS结束...........................O（|V|+|E|）
- 统计黑色点的个数.............O(|V|)
- 如果n个，说明牛皮是真的
- 如果<n个，说明市长说了假话

(b)牛皮2：定点出发可返回

初步分析

- 只需要验证，这个定点在一个有向圈里

算法

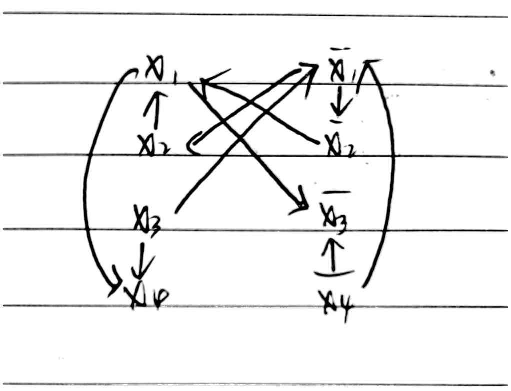
- 从town hall出发BFS
- 每个点初始染为白色，访问到时染为黑
- 每次进入下一层时，只访问白色的孩子
- 如果访问到了town hall，说明成功
- 如果访问完了n层或下一层不再有白色孩子时，还没有访问回town hall，则失败

时间

未必包含所有点和所有边的BFS，O（|V|+|E|）

4、构造SCC

（a）



（b）

- 有向线段意义：由a可推出b
- 如果x和x反在一个SCC中，说明x与x反是同真同假的，矛盾

（c）

- 执行Tarjan’s算法，给每个变量赋值一个SCC编号，则任意x与x反的编号不同
- 随机令x为真，那么在x所在SCC中的其他变量的真假随即确定，不会产生矛盾

- x所在SCC中其他变量的反都在其他SCC中，其真假也随之确定，从而其他SCC的真假性也确定下来了
- 每个SCC内部不会产生矛盾，SCC之间也不会产生矛盾，所以可以实现

（d）

1、执行Tarjan’s算法，给每个变量赋值一个SCC编号..................O（|V|+|E|）
2、检查每个SCC内部，如果出现x与x反在同一个SCC中，直接返回false...................O（|V|）
3、检查完毕后，随机令x为真，那么在x所在SCC中的其他变量的真假随即确定，不会产生矛盾
4、我们称x所在SCC中的其他变量的反所在SCC为关联SCC，则所有关联SCC中的变量真假值都随着x的确定而确定
5、如果此时还有SCC未确定值，则重复上述3、4操作，直到所有变量都定..................O（|V|+|E|）

时间：O（|V|+|E|）

5、寻找MST

（a）权互异则MST唯一

引理：MST必然包含权最小那条边

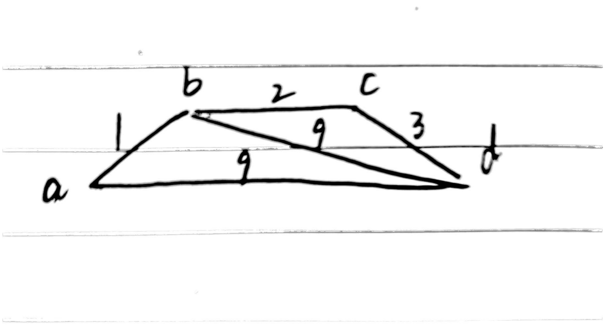
> 反证法，假设已有的MST不包含权最小的边
> 不妨假设w(a,b)最小，且不在MST中，即a，b不相邻
> 虽然a,b不相邻，但是a,b是联通的，所以存在路径a->c->.....->d->b
> （c，d可以是同一个点）
> 那么w(a,b)<=w(a,c),w(a,b)<=w(b,d)
> 所以我们可以删除（a,c），替换成(a,b)
> 这样一来，原路径变成c->.....->d->b->a，原MST的其他边都不变，联通性保持
> 所以新树仍然是一个最小生成树，但权和更小，与假设矛盾

归纳证明：

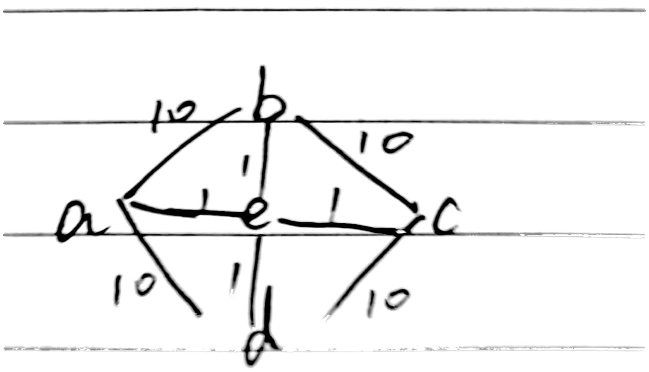
> 记|V|=n，当n<=3时，显然成立
> 假设n<=k时成立
> 当n=k+1时，采用反证法，假设有两个不同的MST：L和R
> 两个MST都必然包含权最小的那条边（a,b）
> 在两个MST中同时删除（a,b），则每个MST都被分解成了两个规模小于等于n的MST：记包含a的小MST为La，Ra，记包含b的小MST为Lb，Rb

> 则La与Ra，Lb与Rb包含的点完全相同，由归纳假设，La与Ra的MST相同，Lb与Rb的MST相同
> 由于L即为La,（a,b）,Lb合并；R即为Ra,（a,b）,Rb合并；所以L与R完全相同，与假设矛盾，所以n=k+1时也成立
> 综上，任意n均成立

（b）权相等MST仍唯一



（c）反例：MST必然包含每个圈中权最小那条边



6、MST反用--去圈

思路

- 寻找最大生成树即可，所有不在最大生成树中的边权和即为结果

算法：Prim修改

```
Prim(G,w):
Pick an arbitrary node x
for (each node u)
  u.dist = INF,  u.in = false
x.dist = 0
Build a priority queue Q based on “dist” values
while (Q is not empty)
  u = Q.ExtractMax()
  u.in = true
  for (each edge (u,v))
    if (v.in==false and w(u,v)>v.dist)
      v.dist = w(u,v)
      Q.Update(v,w(u,v))
```

最后再把不在最大生成树中的边的权和加在一起并返回即可

7、修改MST

（a）边权变小

若e在E'中

不做任何修改

若e不在E'中

> 不妨假设e=(a,b)，不在MST中，即a，b不相邻
>
> 虽然a,b不相邻，但是a,b仍在MST中，所以存在路径a->c->d->.....e->f->b（c，d可以是同一个点）
>
> 需要依次计算w(a,c)、w(c,d)、.....w(e,f)、w(f,b)，取最大的记为w（m，n）
>
> 如果w（a,b）>w(m,n)，则不用改变
>
> 否则，删除（m,n），替换成(a,b)
>
> 这样一来，原路径变成n->.....->e->f->b->a->c->d，原MST的其他边都不变，联通性保持
>
> 所以新树仍然是一个最小生成树，但权和更小

时间：最坏情况下1计算所有边的权，O（|E|）

（b）边权变大

若e不在E'中

不做任何修改

若e在E'中

> 不妨假设e=(a,b)，先删去e，使a，b不相邻
> 此时MST被分成两个部分，记包含a的为A，包含b的为B，则（A，B）为cut
> 下面找出A与B的light edge，再将这个edge替换原来的e即可得到新的MST
> （这个新的edge很有可能就是原来的e，但这并不影响我们需要进行的这些操作）

时间：

- 将MST分成两个部分：O（|V|）
- 寻找light edge：所有除了A与B内部的边都需要比较，O（|E|-|V|）
- 总时间：O（|E|）