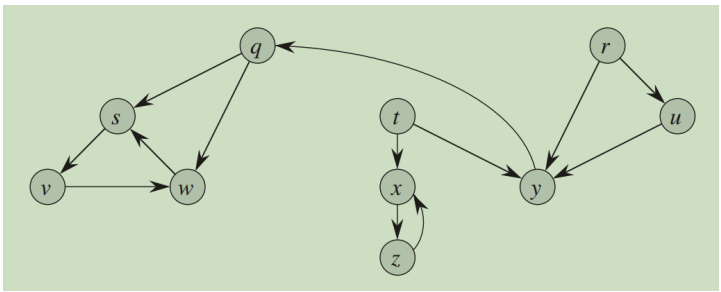
PS9--201300086 史浩男

下面用四种字母简化代表四种边

```
TREE:T
Back:B
Forward:F
Cross:C
```

1、画图

> 全错了
> 原题的意思按字母顺序，不管你是不是源点
> 所以应该从q开始



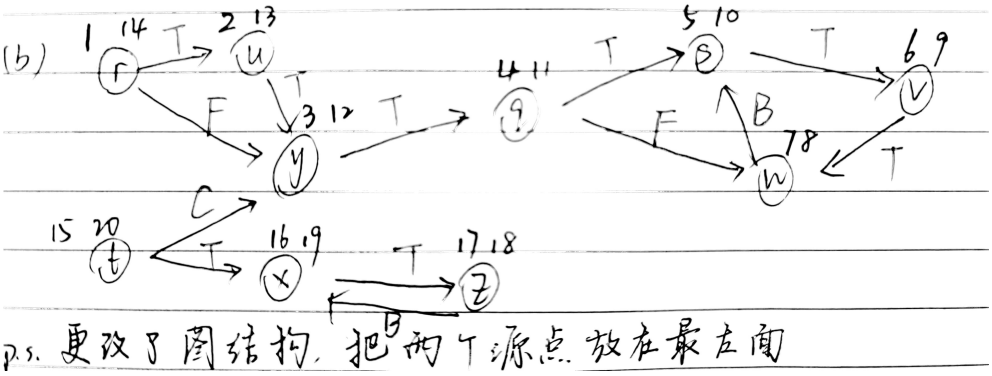
(a) BFS

```
(r,0,NIL)->(u,1,r)->(y,1,r)->(q,2,y)->(s,3,q)->(w,3,q)->(v,4,s)
(t,0,NIL)->(x,1,t)->(z,2,x)
```

(b) DFS

本图更改了图结构，把两个源点r和t放在了最左面

> (y,t)应该是C，因为从q开始了

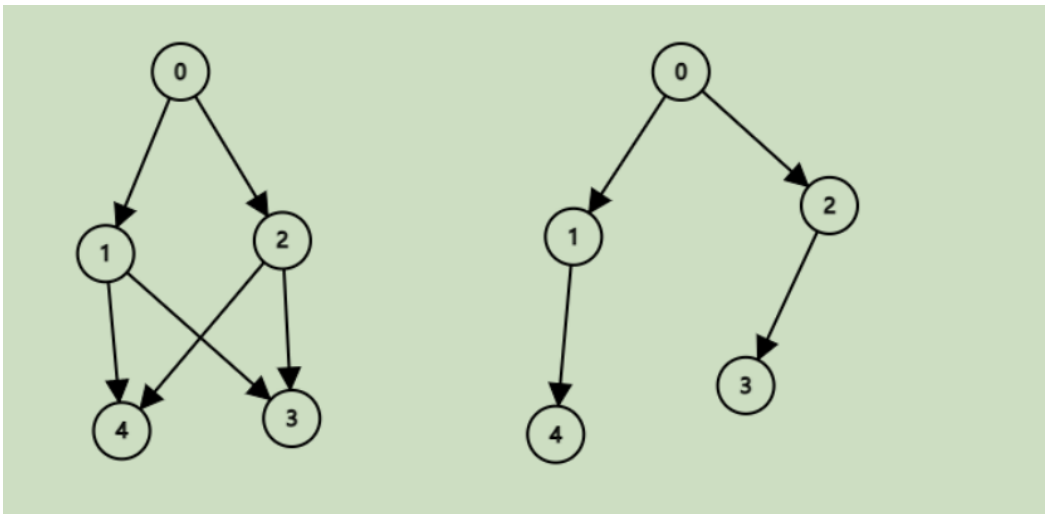


2、

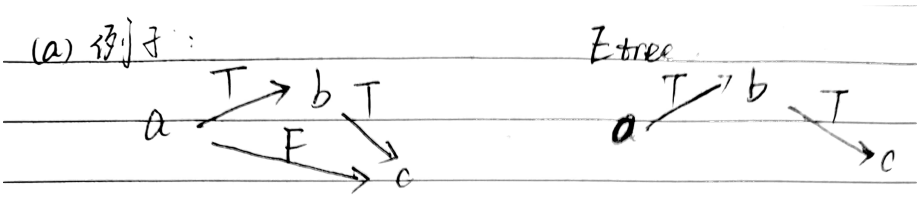
（a）BFS得不到DFS产生的Etree

例子如图

> DFS不走的是这种交叉边



> 错



解析：

- 取Etree包含（a,b）(b,c)
- 则BFS只会经过（a,b）(a,c)两条边，永远不会经过（b,c）

（b）边的类型

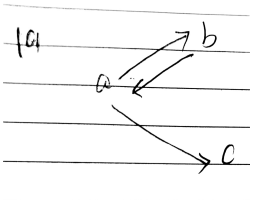
from/to	WHITE	GREY	BLACK
WHITE	TBFC	BC	C
GREY	TF	TBF	TFC
BLACK	无	B	TBFC

规律：

- 白白，黑黑：等价，都有可能
- 白->?：都可以是C
- 灰->?：都可以是TF
- ？->灰：都可以是B
- ？->黑：都可以是C

（c）存在路径，不代表可确定访问顺序

反例如图



解析：

- 存在b到c的路径
- 但对于深搜（a,b）（a,c），b.f=3，c.d=4
- c.d>b.f，找到反例

3、无向图深搜标记连通分支

```
DFS(G,s,k):
s.visited = true
s.cc=k
for (each edge (s,v) in E)
  if (!v.visited)
    DFS(G,v,k)
```

```
DFSAll(G):
k=1
for (each node u)
    u.visited = false
for (each node u)
    if (u.visited == false)
      k++
      DFS(G,u,k)
```

时间分析

- DFSALL的第一个for：|V|
- DFSALL的第一个for：已访问过的点不会再耗时，所以|V|
- DFSALL的第一个for中的DFS：已访问过的联通分治不会重复，因此O（|E|）
- 总共O（|V|+|E|）

4、二部图无奇圈

（a）树都是二部图

构造证明：

将树的奇数层结点加入L，将偶数层结点加入R，则满足要求

（b）二部图《==》无奇圈

==》：

反设有奇圈，不妨设奇圈长度为2k+1

在奇圈中随机取一个结点，假设为第一个结点，沿这个圈顺时针依次是第2，3，4.....2k+1个结点

不妨设第一个结点在L中，则第二个一定在R中.....第2k+1个一定在L中

矛盾！因为第一个结点和第2k+1个结点是相邻的，且都在L中

《==：

如果无圈，则每个联通分支都是树，已证

如果只有偶圈：

- 偶圈内的任意相邻顶点一定不属于同一个subset，由于偶圈有偶数个点，可以做到这一点。
- 可以删去这个偶圈内的所有边，不影响二部图的性质
- 对所有偶圈执行上述操作，直到没有偶圈

此时，无圈，已证

（c）二部图判定算法

算法：广搜改进

```
is_bipartite(G):
for (each u in V).....O(|V|)
  u.c = WHITE, u.d = INF//根据奇偶性记录属于哪个subset
for (each u in V).....O(|V|)
  if (u.c == WHITE)//此处循环判断新连通分支
    u.c = GRAY, u.d = 0
    Q.enque(u)
    while (!Q.empty())
       v = Q.dequeue()
      v.c = BLACK
      for (each edge (v,w) in E).....所有循环一共O（|E|）
        if((w.d+v.d)%2==0)
            return false
        else if (w.c == WHITE)
          w.c = GRAY
          w.d = v.d+1
          Q.enque(w)
return true
```

时间分析

O（|V|+|E|），详见注释

五、广搜应用

问题转化--一些说明

```
u    点，代表一个小方格
u.l  u.r  分别代表点u所在的横纵坐标
MAZE点集，共n^2个点
u.num格子内的数值
Reachable(u)点集：包含所有从u开始，上下左右移动u.num格能抵达且在maze中的点
edge(u,v)  边，表示可以从u抵达v
```

```
Find_path(G):
for (each u in MAZE)
  u.c=WHITE, u.d=INF
s.c=GRAY, s.d=0,s.l=s.r=1
Q.enque(s)
while (!Q.empty())
  u = Q.dequeue()
  u.c = BLACK
  for (each v in Reachable(u))
    if (v.c == WHITE)
      v.c = GRAY
      v.d = u.d+1
      Q.enque(v)
    if(v.l==n and v.r==n)
        return v.d
return false
```

时间分析

共有n^2个点，至多有4*n^2条边

总时间为O（|V|+|E|）=O（n^2）

六、深搜应用

序号转化颜色

```
color(n):
    switch(n%3)
        case 0: return RED
        case 1: return WHITE
        case 2: return BLUE
```

用 (s,v).color表示这条边的颜色

```
French(G,s):
s.visited = true
Q.enque(s)//记录French walk可抵达的所有结点，包含了源点s
for (each edge (s,v) in E)
  if (!v.visited and (s,v).color==color(s.i))
    v.i=s.i+1
    French(G,v)
```

主程序：

```
v.i=0
French(G,v)
Q.dequeue(v)//由题意，源点不应该在结果中，最后应该剔除
```

时间分析：

最坏情况下可以遍历所有点和边，即O（|V|+|E|）