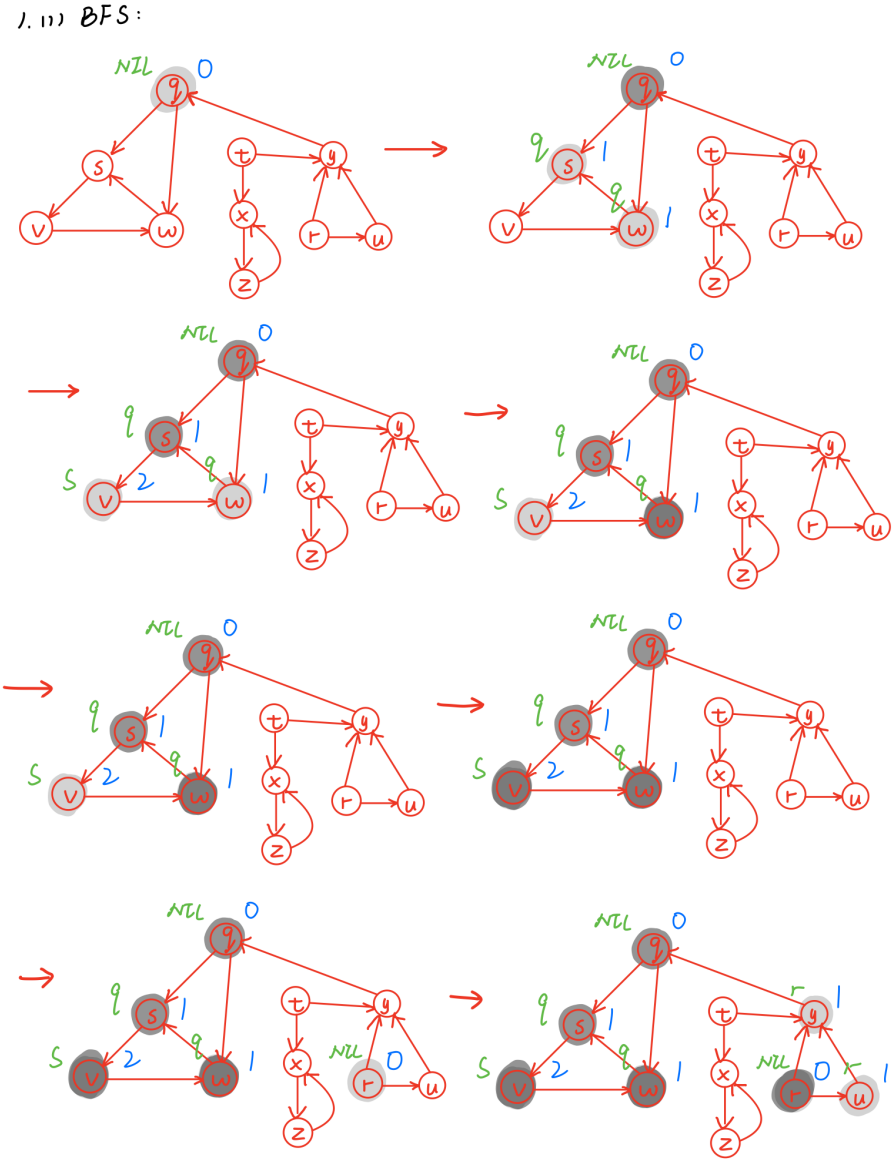
Problem Set 9

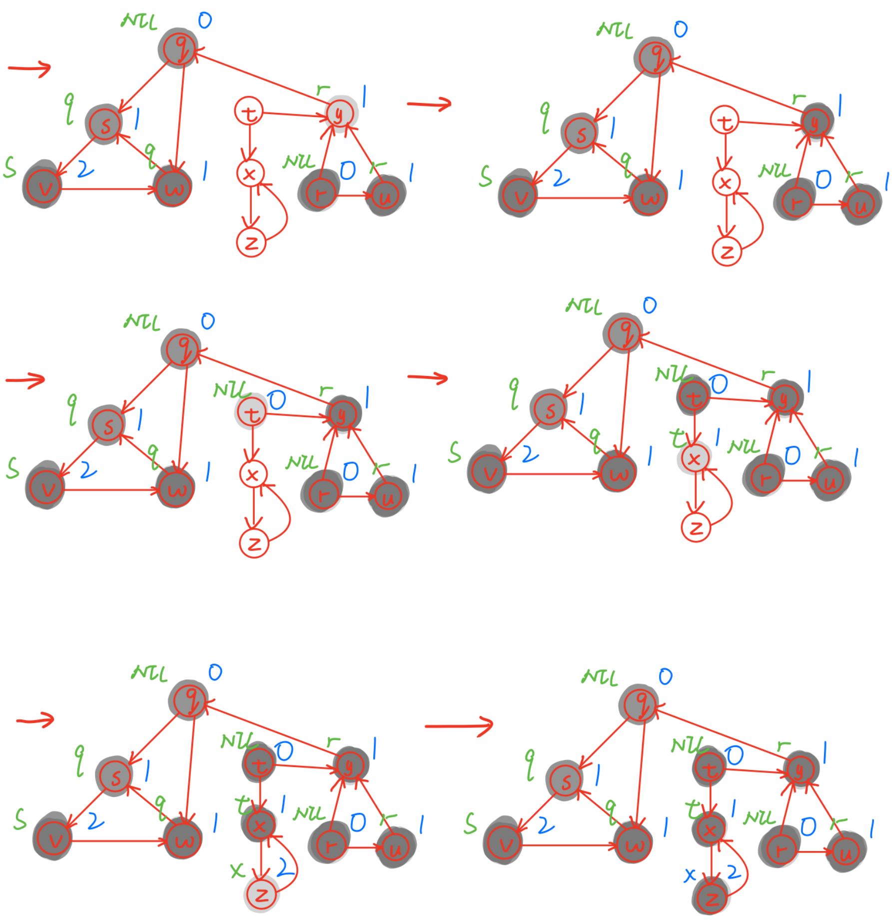
麻超　201300066

南京大学人工智能学院

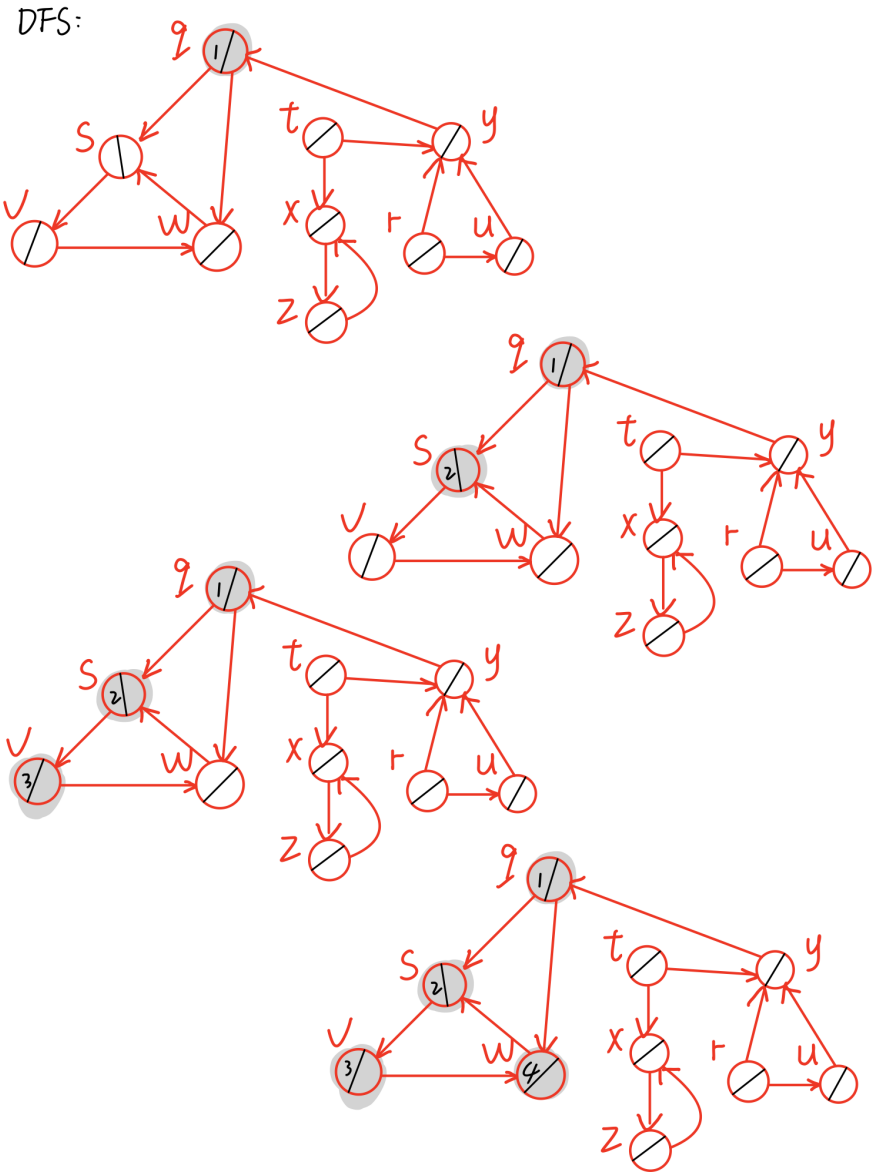
1 Problem 1

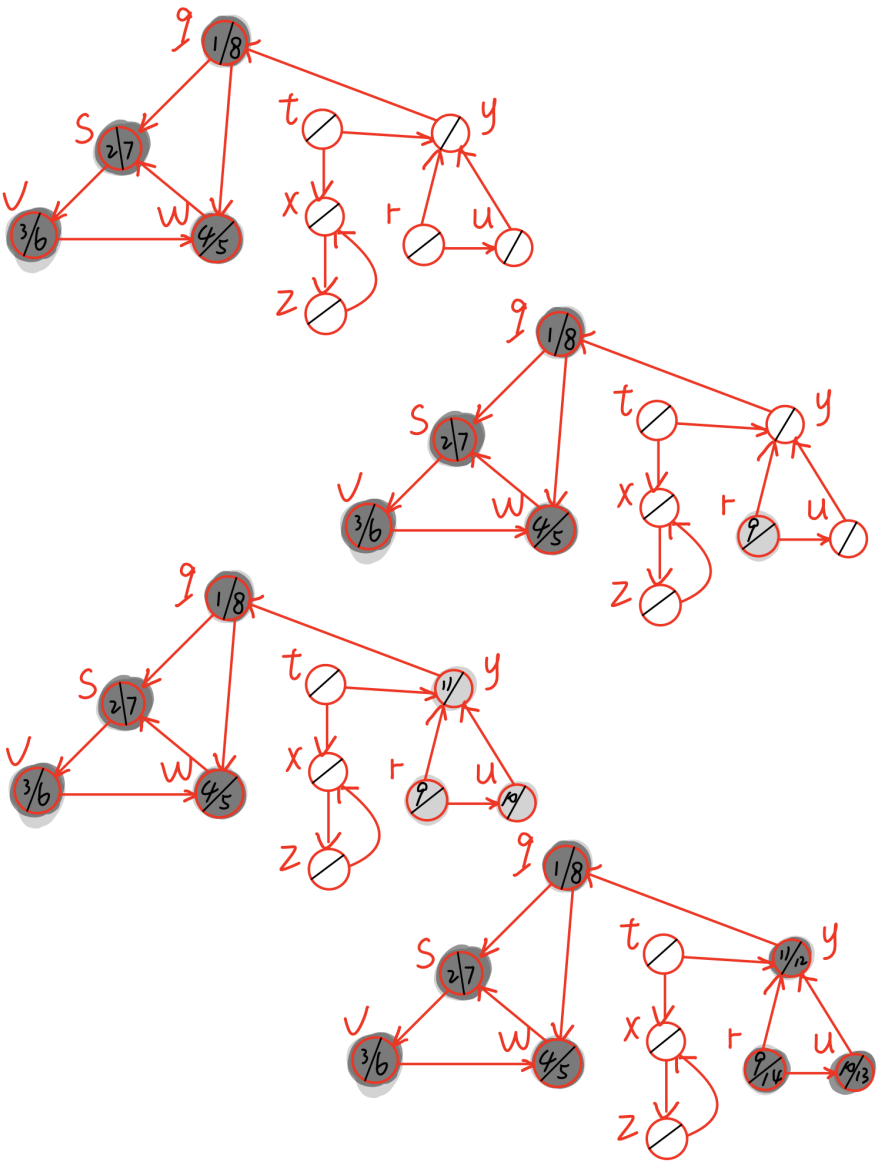
1.1 a

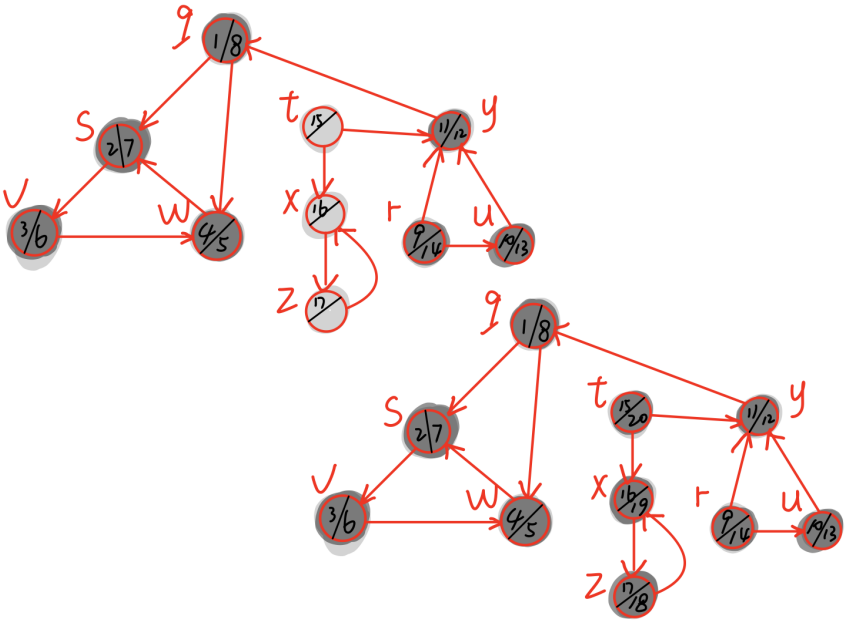




1.2 b




q
1/8
s
2/7
v
3/6
w
4/5
t
y
x
r
u
z
q
1/8
s
2/7
v
3/6
w
4/5
t
y
x
r
9/
u
z
q
1/8
s
2/7
v
3/6
w
4/5
t
11/
y
x
r
9/
u
10/
z
q
1/8
s
2/7
v
3/6
w
4/5
t
11/12
y
x
r
9/14
u
10/13
z



树边：(q,s), (s,v), (v,w), (r,u), (u,y), (t,x), (x,z)

后向边：(w,s), (z,x)

前向边：(q,w), (r,y)

横向边：(y,q), (t,y)

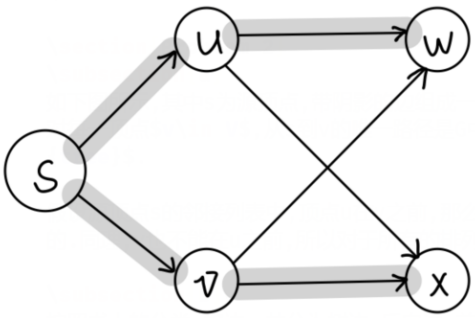
2 Problem 2

2.1 a

如下图所示, 其中 s 为源顶点, 带阴影的边组成一个边的集合 E_{tree}, 自然 E_{tree} 属于 E. 显然 e 满足对每一顶点 $v \in V$, 从 s 到 v 的唯一路径是 G 中的一条最短路径. 但

是对该图做 BFS, 无法产生边的集合 E_{tree}.

假设源顶点 s 的邻接列表中, 顶点 u 在 v 之前, 那么需要在 v 之前列出 u, 自然 $w.\pi = u, x.\pi = u$, 但这是不成立的. 同理,v 也不能在 u 之前, 所以对于所有的排列, 都不能通过在该图上运行 BFS 而产生边集 $E.tree$



2.2 b

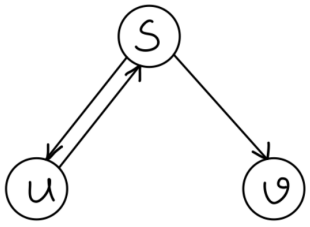
按照书上的分类, 将边一共分为树边, 后向边, 前向边, 横向边

对于有向图:

-	WHITE	GRAY	BLACK
WHITE	所有可能	后, 横	横
GRAY	前, 横	树, 后, 前	树, 前, 横
BLACK	横	横, 后	所有可能

2.3 c

如下图所示, 即为反例.



	d	f
s	1	6
u	2	3
v	4	5

3 Problem 3

当执行 DFS 的第 5,6 行时, 每跳入一次第 7 行, 连通分量数加 1, 在 DFS-VISIT 里遇到的结点的联通分量数相同.

Algorithm 1 RE-Dijkstra(G,r,x,y)

```
INITIALIZE-SINGLE-SOURCE(G,x)
S = ∅
while Q ≠ ∅ do
    u =EXTRACT(Q)
    S = S ∪ {u}
    for each vertex v if v ∈ G.Adj[u] do
        if v.d < u.d × r(u,v) then
            v.d = u.d × r(u,v)
            v.π = u
        end if
    end for
end while
while y ≠ x do
    Output y
    y = y.π
end while
Output x
```

Algorithm 2 DFS-VISIT(G,u,ceil)

```
time = time + 1
u.d = time
u.color=GRAY
for each vertex u ∈ G : Adj[u] do
    if v.color==WHITE then
        v.π = u
        DFS-VISIT(G,u,ceil)
    end if
end for
u.color=BLACK
u.ceil = ceil
time = time + 1
u.f = time
```

4 Problem 4

4.1 a

对于每一个树, 先找其顶点 (根节点), 然后与根节点距离为偶数的结点 (包含根节点) 归为一个点集合, 与根节点距离为奇数的结点归为另外一个点集合, 那么这两个点集合就构成了图中所有顶点集合的划分.

首先, 对于每个树, 其节点到根节点的距离是一定的一个整数, 则所有的顶点都分属于这两个集合. 另外, 对于树中的每一条边, 其连接的都是分属于两个集合中的顶点, 不会连接每个集合内部, 则树中所有的边两端的顶点分别属于这两个集合, 所以每一个树都是一个二部图.

4.2 b

首先证明必要性:

设 G 为二部图 <X,E,Y>.X,Y 非空.

设 C 为 G 中的任一回路, 令 C=($v_0, v_1, v_2, ...v_{l-1}, v_l = v_0$)

由二部图的性质,v_i 必定相间出现在 X 和 Y 中.

设 ($v_0, v_2, ...v_l = v_0$) 属于 X,($v_1, v-3, ...v_{l-1}$) 属于 Y. 则 l 必然为偶数, 即 C 中有偶数条边.

接下来证明充分性:

设 G 的所有回路具有偶数长度, 并设 G 为连通图 (若 G 不连通, 则可对 G 的各连通分支作下述讨论):

令 G 的顶点集为 V, 边集为 E, 现构作 X,Y, 使 <X,E,Y> =G. 取 v_0 属于 V, 则令 X={v|$v = v_0$ 或到 v_0 有偶数长度的通路},Y=V-X.

显然,X 非空, 由于 G 为一连通图, 则 v_0 必然有相邻顶点, 则 Y 亦非空.

假设有一条边 (u,v), 且 u,v 都属于 X, 那么 v_0 有到 u 偶数长度通路或者 $u = v_0$,v 同理. 无论是哪种情况, 均有一条 v_0 到 v_0 的奇数长度的回路, 与题设矛盾, 所以不可能有这么一条边.Y 同理, 则没有任何边的两个端点都在 X 中或都在 Y 中, 因此充分性得证.

4.3 c

首先 for 循环迭代每个点, 如果这个点还没有染色, 那么就将这个点染色, 然后 dfs 它的所有临点, 并且将没有染过色的临点染色成另外一种颜色, 如果临点已经染过色了那么判断相邻两点颜色是否一致, 一致的话就不是二部图. DFS 的时间复杂

Algorithm 3 Judge

```
bool flag=true
for i = 1 → n do
    if !color[i] then
        if (!dfs(i,1)) then
            return False
        end if
    end if
end for
function DFS(u,c)
    color[u]=c
    for (i=h[u];ĩ;i=ne[i]) do
        j=e[i]
        if !color[j] then
            if !dfs(j,3-c) then return false
            end if
        end if
        if color[j]==c then return false
        end if
    end for
    return true
end function
```

度为 $\Theta(n+m)$, 故判断其是不是二部图的时间复杂度为 $O(n+m)$.

5 Problem 5

选择从迷宫的起点入手推导, 利用 DFS, 当遇到之前已经访问过的节点或者无论这一次怎么走都会出界的节点时就换一条路, 如果遍历了所有路径都无法到达终点则说明这个迷宫是无解的, 否则则选取所有可到达终点的路径中最短的一条的长度返回.

伪代码如下 (下一页):

由于其边和点的总个数不会超过 $2n^2$, 所以时间复杂度为 $O(n^2)$.

Algorithm 4 Maze(G,n,point=(1,1))

```
for every v ∈ G do
    visited[v]=false
end for
visited[point]=true
nextpoint[]=point.change(point.value)
for every exist point in nextpoint[] do
    if point==goal then
        return pathlength
    else
        if point!=visited then
            Maze(G,n,point)
        end if
    end if
end for
```

6 Problem 6

只需要对原图进行一次 DFS 并注意将符合条件的点纳入即可完成.

伪代码如下:

Algorithm 5 French(G,v)

```
for every v ∈ G do
    visited[v]=false
    result[v]=false
end for
function DFS(G,v)
    visited[v]=true
    result[v]=true
    for each w adjecent to v
    if site (w,v) meet the criteria then
        result[w]=true
        if !visited[w] then
            DFS(G,w)
        end if
    end if
end function
```

由于 DFS 的时间复杂度为 $\Theta(n+m)$, 所以该算法的时间复杂度为 $O(n+m)$