24.2-4

24.2-4 给出一个有效的算法来计算一个有向无环图中的路径总数。分析你自己的算法。

```
PATHS_COUNT
    topologically sort the vertices of G
    for each vertex u, take in topologically sorted order
        for each v ∈ G.Adj[u]
            v.npath = u.npath + 1 + v.npath
    sum = 0
    for each u ∈ V
        sum = sum + u.npath
    return sum.
```

拓扑排序的时间复杂度为 $O(V+E)$，for循环嵌套的时间复杂度为 $O(V+E)$，因此该算法的运行时间为 $O(V+E)$

24.3-6

24.3-6 给定有向图 $G=(V, E)$，每条边 $(u, v)\in E$ 有一个关联值 $r(u, v)$，该关联值是一个实数，其范围为 $0\leqslant r(u, v)\leqslant 1$，其代表的意思是从结点 u 到结点 v 之间的通信链路的可靠性。可以认为，$r(u, v)$ 代表的是从结点 u 到结点 v 的通信链路不失效的概率，并且假设这些概率之间相互独立。请给出一个有效的算法来找到任意两个结点之间最可靠的通信链路。

将 Dijkstra 算法中路径权重的累加改为乘法，同时反转松弛中不等号的方向，以寻找乘积最大的路径。

时间复杂度与 Dijkstra 相比保持不变

```
RELIABILITY-DIJKSTRA(G, r, s, t)
    for each vertex v ∈ G.V
        v.d = -∞
        v.π = NIL
    s.d = 0
    Q = G.V
    while Q ≠ ∅
        u = EXTRACT-MAX(Q)
        for each vertex v ∈ G.Adj[u]
            if v.d < u.d · r(u,v)
                v.d = u.d · r(u,v)
                v.π = u
```

与上一个 while 对齐

```
v = t
while v ≠ s
    PRINT(v)
    v = v.π
```

25.1-9

25.1-9 修改 FASTER-ALL-PAIRS-SHORTEST-PATHS，使其可以判断一个图是否包含一个权重为负值的环路。

多做一次乘法.

得到的结果矩阵不变 ⟶ 包含一个权重为负值的环路.

得到的结果矩阵变化 ⟶ 不包含一个权重为负值的环路

25.2-8

25.2-8 给出一个 $O(VE)$ 时间复杂度的算法来计算有向图 $G=(V, E)$ 的传递闭包。

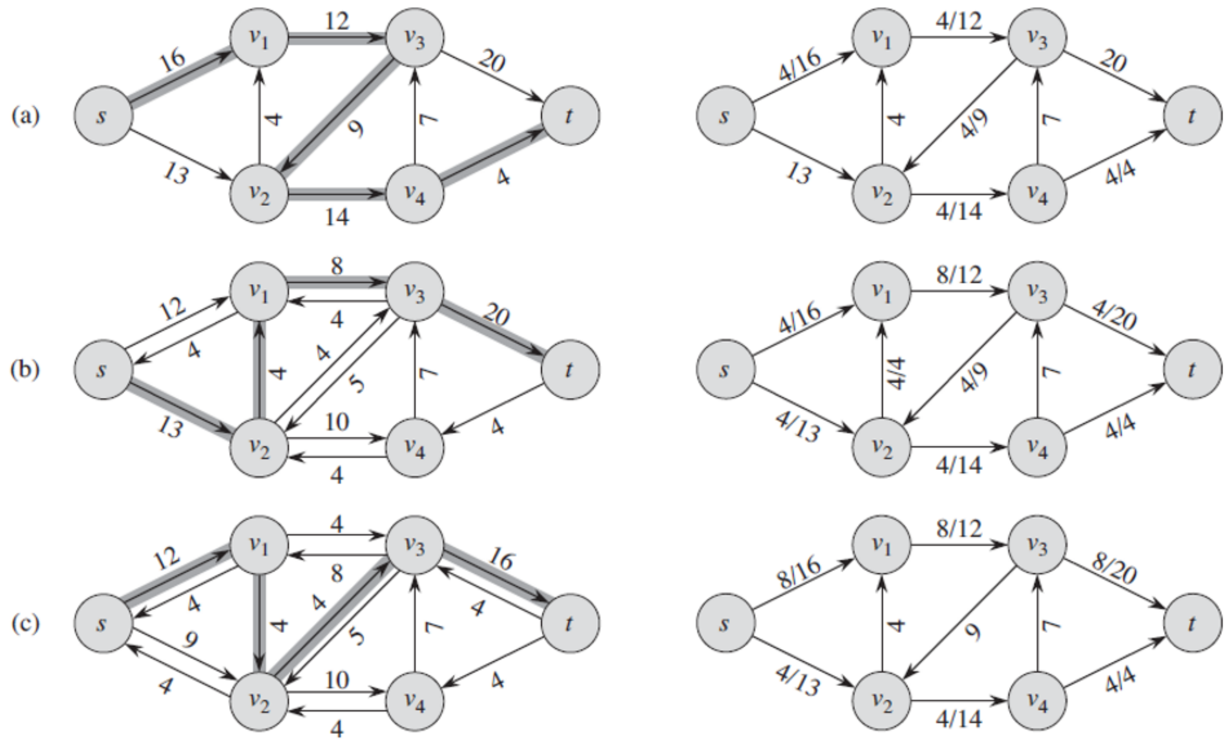
① 搜索所有度数为0的顶点 ⇒ $O(V+E)$

② 排除度数为0的顶点，对其他顶点执行 DFS 算法，将源顶点和搜索到的其他顶点相关联

③ 分析：每条边只能关联2个顶点，遍历到的顶点数不超过 $2E$，因此总耗时 $O(V)\cdot O(V+E)=O(V^2+VE)=O(VE)$，即时间复杂度为 $O(VE)$

26.2-4

In the example of Figure 26.6, what is the minimum cut corresponding to the maximum flow shown? Of the augmenting paths appearing in the example, which one cancels flow?



minimum cut : ({s, v_1, v_2, v_4}, {t, v_3})

In part (c), the edge (v_2, v_3) cancels flow on edge (v_3, v_2)

26.2-10. ①假设我们有一个最大流 f

②在一个新图 G 上我们将边 (u,v) 的容量 (capacity) 修改为 f(u,v)
(复制的一份)

③在 G 上运行 Ford-Fulkerson 算法，当一条边的流量 (flow) 等于其容量 (capacity) 时，删除这条边

26.2-10 ④此时遍历的增广路径即为所需的路径，

26.2-10 说明在流网络 $G=(V, E)$ 中，如何使用一个最多包含 $|E|$ 条增广路径的序列来找到一个最大流。（提示：找到最大流后再确定路径。）

最多为 O(E) 条