算法设计与分析

作业（七）

姓　　名	熊恪峥
学　　号	22920202204622
日　　期	2022年4月13日
学　　院	信息学院
课程名称	算法设计与分析

作业（七）

目录

1 题8.3

使用邻接矩阵存图，图的转置就是矩阵的转置，因此可以直接使用矩阵的转置来求解。这种方法的时间复杂度是$\mathcal{O}(\|V\|^2)$，如算法 1。

算法 1 转置邻接矩阵

Input: 邻接矩阵M

```
1: procedure TRANSPOSE(M)
2:     V ← VERTICES(M)
3:     T ← EMPTYMATRIX(V)
4:     for i = 1 → |V| do
5:         for j = 1 → |V| do
6:             T[i, j] ← M[j, i]
7:     return T
```

使用邻接表存图，可以通过遍历每个顶点的邻接表并将反向边插入新图中实现，这种方法的时间复杂度是$\mathcal{O}(\|V\| + \|E\|)$，如算法 2。

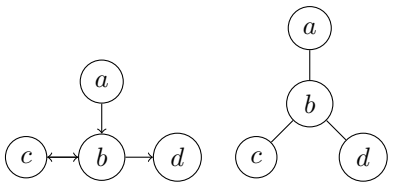
算法 2 转置邻接表

Input: 邻接表G

```
1: procedure TRANSPOSE(G)
2:     GT.Vertices ← G.Vertices
3:     for v ∈ G.Vertices do
4:         GT.Adj[v] ← EMPTYLIST
5:         for e ∈ G.Adj[v] do
6:             GT.Adj[e.V2] ← (e.V2, e.V1)
```

2 题8.4

如图 2，以a为源点进行深度优先遍历，得到的$d(u)$如表 2



(a) 无向图G (b) 产生的搜索树

图 1: 反例

u	a	b	c	d
$d(u)$	1	2	3	4

表格 1: 深度优先遍历得到的$d(u)$

可见$d(c) > d(d)$，且c到d存在路径，但d不是c的子顶点。

3 题8.6

Tarjan算法可以在$\mathcal{O}(\|V\| + \|E\|)$的时间里求出强连通分量。对无向图中的每一个顶点使用算法 3，可以求出强连通分量。

算法 3 Tarjans算法

```
1: procedure TARJAN(G)
2:     GT.Vertices ← G.Vertices
```

4 题8.7

Floyd算法的修改版可以用来计算传递闭包，如算法4。

算法 4 Floyd算法求传递闭包

```
1: procedure FLOYD(G)
2:     GT.Vertices ← G.Vertices
```

5 题8.11

存在一些图使得$Prim$算法慢于$Kruskal$算法。当使用的排序算法足够好时，$Kruskal$算法的时间复杂度时$\mathcal{O}(\|E\| \log \|E\|)$，而$Prim$算法的时间复杂度是$\mathcal{O}(\|V\|^2)$。因此，对于顶点多而边少的图，$Prim$算法比$Kruskal$算法慢。

6 题8.12

当图中存在负环时i到j无最短路径。如图 2。

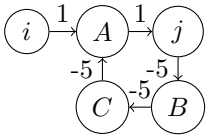


图 2: 反例

7 题8.13

实现该功能，只需要遍历所有的边，并比较是否有不满足三角形不等式的距离，即 $d[v] > d[u] + w(u, v)$。如算法5。

算法 5 Bellmanford算法

```
1: procedure BELLMANFORD(G, w, s)
2:     for i ← 1 → ‖V‖ − 1 do
3:         for each edge(u, v) ∈ E do
4:             RELAX(u,v,w)
5:     for each edge(u, v) ∈ E do
6:         if d[v] > d[u] + w(u, v) then
7:             d[v] ← ∞
8:             return 0
9:     return 1
```

8 题8.14

为了求出DAG中的路径数量，首先进行拓扑排序然后按由逆拓扑序递推可以求出任意两点s,t间的路径数量。

$$f[i] = \begin{cases} 1 & i = t \\ f[i] + \sum_{(i,v)\in E} f[v] & i \neq t \end{cases} \tag{1}$$

然后对起点s和任意点求路径数并求和，就可以得到路径树，如算法 6。

算法 6 计算路径数

```
1: procedure PathCount(G, s)
2:     GT.Vertices ← G.Vertices
```

9 题8.16

当第一次$Relax$操作未能改变$d[v]$时停止，则可以得到最小边数的最大值。如算法 7

算法 7 计算最小边数最大值

```
1: procedure BellmanFord(G, s)
2:     GT.Vertices ← G.Vertices
```

10 题8.20

算法仍然正确。

Proof. 设最后一个顶点为u，如果u到s不可达，则$d[u] \leftarrow \infty = \delta(s,u)$

如果u到s可达，则存在路径p，从s经过点m到达u，由于$d[x] = \delta(s,x)$，则按路径松弛性质，$d[u] = \delta(s,u)$

则算法正确 □

11 题8.24

为了构造最优解，需要在每次成功松弛的时候记录松弛的点。如算法 8

算法 8 计算最小边数最大值

```
1: procedure Floyd(G)
2:     GT.Vertices ← G.Vertices
```

12 题8.25

每次迭代生成的矩阵如表：

13 题8.26

该算法需要存储d_{ij}，由于$i \leq \|V\|, j \leq \|V\|$，则所需空间为$\mathcal{O}(\|V\|^2)$。

该算法的正确性证明如下

Proof. □

14 题8.27

由松弛方法可知，可以调用两次Floyd算法，如果调用完第二次，有路径长度被再一次更新，则说明存在负环。