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

procedure TRANSPOSE(M)
    $V \leftarrow$ VERTICES(M)
    $T \leftarrow$ EMPTYMATRIX(V)
    for $i = 1 \to |V|$ **do**
        for $j = 1 \to |V|$ **do**
            $T[i, j] \leftarrow M[j, i]$
    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。

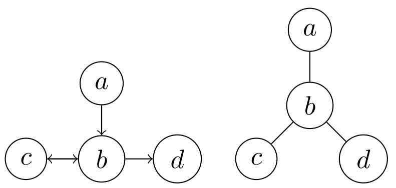
算法 2 转置邻接表

Input: 邻接表G

procedure TRANSPOSE(G)
    $GT.Vertices \leftarrow G.Vertices$
    for $v \in G.Vertices$ **do**
        $GT.Adj[v] \leftarrow$ EMPTYLIST
        for $e \in G.Adj[v]$ **do**
            $GT.Adj[e.V2] \leftarrow (e.V2, e.V1)$

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

```
procedure TARJAN(G, r)
    low[r] ← dfn[r] ← ++dfn_cnt
    vis[r] ← true
    stack ← r
    for each (r, v) ∈ E do
        if !vis[v] then TARJAN(G, v)
            low[r] ← min(low[r], low[v])
        else if dfn[v]¡dfn[r] then
            low[r] ← min(low[r], dfn[v])
    if low[r]==dfn[r] then
        scc ← scc_cnt++
        while stack ≠ ∅ do
            v ← stack.pop()
            scc[v] ← scc_cnt
            vis[r] ← false
```

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。具体而言，按(1)初始化矩阵

$$G[i,j]=\begin{cases}1 & i=j\\ 0 & otherwise\end{cases} \tag{1}$$

并(2)按更新

$$G[i,j]=\begin{cases}true & \exists k, G[i,k]=true\,and\,G[k,j]=true\\ false & otherwise\end{cases} \tag{2}$$

算法 4 Floyd算法求传递闭包

```
procedure FLOYD(G)
    for i ∈ V do
        for j ∈ V do
            if i==j then
                G[i, j] ← 1
            else
                G[i, j] ← 0
    for k ∈ V do
        for i ∈ V do
            for j ∈ V do
                G[i, j] ← G[i, j] or G[i, k] and G[k, j]
```

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。

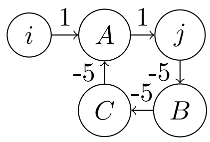


图 2: 反例

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

```
procedure BELLMANFORD(G, w, s)
    for i ← 1 → ‖V‖ − 1 do
        for each edge(u, v) ∈ E do
            RELAX(u,v,w)
    for each edge(u, v) ∈ E do
        if d[v] > d[u] + w(u, v) then
            d[v] ← ∞
            return 0
    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(3)由逆拓扑序递推可以求出任意两点s,t间的路径数量。

$$f[i] = \begin{cases} 1 & i = t \\ f[i] + \sum_{(i,v)\in E} f[v] & i \neq t \end{cases} \tag{3}$$

然后对起点s和任意点求路径数并求和，就可以得到路径数，如算法 6。

算法 6 求路径数

```
procedure PATHCOUNT(G, s, t)
    topo ← TOPOSORT(G)
    f[t] ← 1
    for i ← 1 → ‖V‖ − 1 do
        for each v ∈ topo[i] do
            f[v] ← f[v] + f[i]
    return f[s]
```

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

```
procedure BELLMANFORD(G, s)
    INITIALIZESINGLESOURCE(G, s)
    updated ← true
    while updated do
        updated ← false
        for each edge(u, v) ∈ E do
            if d[v] > d[u] + w(u, v) then
                RELAX(u,v,w)
                updated ← true
```

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

```
procedure FLOYD(G)
    for i ∈ V do
        for j ∈ V do
            if i == j then
                d[i, j] ← 1
            else if G[i, j] then
                d[i, j] ← G[i, j]
            else
                d[i, j] ← ∞
    for i ← 1 → ‖V‖ do
        for j ← 1 → ‖V‖ do
            for k ← 1 → ‖V‖ do
                if d[i][j] > d[i][k] + d[k][j] then
                    d[i][j] ← d[i][k] + d[k][j]
                    p[i][j] ← k
    return d, p
```

然后可以递归地根据p输出具体路径。

12 题8.25

每次迭代生成的矩阵如表：

表格 2: $k = 1$

0	∞	-1	∞	∞	∞
-4	0	3	∞	∞	∞
∞	∞	0	7	∞	∞
1	2	∞	0	∞	∞
∞	∞	∞	2	0	-8
∞	∞	∞	5	10	0

表格 3: $k=2$

0	∞	-1	∞	∞	∞
-4	0	-5	∞	∞	∞
∞	∞	0	7	∞	∞
-2	2	-3	0	∞	∞
∞	∞	∞	2	0	-8
∞	∞	∞	5	10	0

表格 4: $k=3$

0	∞	-1	6	∞	∞
-4	0	-5	2	∞	∞
∞	∞	0	7	∞	∞
-2	2	-3	0	∞	∞
∞	∞	∞	2	0	-8
∞	∞	∞	5	10	0

表格 5: $k=4$

0	8	-1	6	∞	∞
-4	0	-5	2	∞	∞
5	9	0	7	∞	∞
-2	2	-3	0	∞	∞
0	4	-1	2	0	-8
3	7	2	5	10	0

表格 6: $k=5$

0	8	-1	6	∞	∞
-4	0	-5	2	∞	∞
5	9	0	7	∞	∞
-2	2	-3	0	∞	∞
0	4	-1	2	0	-8
3	7	2	5	10	0

表格 7: $k=6$

0	8	-1	6	∞	∞
-4	0	-5	2	∞	∞
5	9	0	7	∞	∞
-2	2	-3	0	∞	∞
-5	-1	-6	-3	0	-8
3	7	2	5	10	0

13 题8.26

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

该算法有循环不变式(4)

$$\textbf{在第}k\textbf{次迭代之前，}d[i,j]\textbf{包含了}i\textbf{到}j\textbf{的最短路径}Q\textbf{，该路径仅包含顶点}\{1,\ldots,k-1\} \tag{4}$$

Proof. 在第k次迭代中，Q的长度与路径$R = R_1 + R_2$进行比较，R_1为从i到k的路径，R_2为从k到j的路径。然后选择较短的路径，这条路径的顶点在$\{1,\ldots,k\}$中。此时有两种情况

1. 没有进行松弛，不包含k，则Q在第$k-1$次迭代中被确定。
2. 包含k，则Q可以分解为R_1和R_2，它们在$k-1$次迭代中被确定。

因此在第k次迭代结束后，循环不变式依然满足。 □

14 题8.27

由松弛方法(5)可知，当有负环存在，该松弛会导致每次调用Floyd算法都会减少某些路径长度。所以可以调用两次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。

$$d[i,j] = \min_{k\in V}(d[i,k] + d[k,j]) \tag{5}$$