# DSA_13

邵宁录　2018202195

2020 年 12 月 17 日

## 1　24.1-3

在每一次循环所有点之前，先把所有的 $v.d$ 给记录下来，然后进行 $RELAX$ 操作。若本轮循环结束时，$v.d$ 没有发生变化，那么就不需要再更新，即此次循环为第 $m+1$ 次。

伪代码如下：

```
BELLMAN-FORD(G, w, s)
    INITIALIZE-SINGLE-SOURCE(G, s)
    for i = 1 to |G.V| - 1:
        for each v in G.V:
            save v.d to T
        for each edgr(u, v) in G.E:
            RELAX(u, v, w)
        for each v in G.V:
            if v.d != T[v].d:
                goto row 11
    for each edge(u, v) in G.E:
        if v.d > u.d + w(u, v):
            return FALSE
    return TRUE
```

## 2　24.3-8

由于 $Dijkstra$ 算法的时间复杂度依赖于最小优先队列队列的实现，所以想要改变算法的时间复杂度应当从优先队列的实现角度来思考。

想要实现 $O(WV+E)$ 的时间复杂度，我们只需要用一个二维数组来实现优先队列。

令 $A$ 为一个 $W \times A$ 的二维数组，其中，具有长度为 $d$ 的节点在 $A[d]$ 的 $list$ 中。

$EXTRACT-MIN$ 操作的时间复杂度为 $O(W)$，因为需要找一遍长度为 $d$ 的节点中最小的 $d$。因此所有的该操作加起来的时间复杂度为 $O(WV)$。

$DECREASE-KEY$ 操作的时间复杂度为 $O(1)$ 的时间，因为检查一条边只需要随机访问就能完成。因此所有的该操作的时间复杂度为 $O(E)$。

综上，总的时间复杂度为 $O(WV+E)$。

# 3 25.2-7

**递归式如下：**

$$\phi_{ij}^{(k)} = \begin{cases} \phi_{ij}^{(k-1)} & if\ d_{ik}^{(k)} + d_{ij}^{(k)} \geq d_{ij}^{(k-1)} \\ k & otherwise \end{cases}$$

**理由如下：**

如果 $d_{ik}^{(k)} + d_{ij}^{(k)} \geq d_{ij}^{(k-1)}$，说明此时在 $Floyd - Warshall$ 算法中，应当取 $d_{ij}^{(k)} = d_{ij}^{(k-1)}$。这说明取 $k$ 个点与取 $k-1$ 个点所得出的最短路径是相同的，因此 $\phi_{ij}^{(k)} = \phi_{ij}^{(k-1)}$。否则，说明新加入的结点 $k$ 跟 $1, 2, \ldots, k-1$ 中的点构成了一条新的最短路径，因此结点 $k$ 是其中最大的节点，即 $\phi_{ij}^{(k)} = k$

**修改后的代码：**

```
FLOYD-WARSHALL(W)
    n = W.rows
    Phi_0 = W
    for k = 1 to n:
        let Phi_k = Phi_0
        for i = 1 to n:
            for j = 1 to n:
                if d_(k)[i][k] + d_(k)[k][j] < d_(k-1)[i][j]:
                Phi_k[i][j] = k


PRINT-ALL-PAIRS-SHORTEST-PATH(Phi, i, j)
    if i == j:
        print i
    else if Phi[i][j] == NIL:
        print "no path from" i "to" j "exist"
    else:
        PRINT-ALL-PAIRS-SHORTEST-PATH(Phi, i, Phi[i][j])
        print j
```

**相似：**

$\Phi$ 与矩阵链式乘法中的 $s$ 表非常类似，因为都是用相似的递归式进行计算的。