算法作业 10

孟妍廷 2015202009

2017 年 12 月 3 日

24.1-3

分析：边的条数的最大值 m 不是事先知道的，我们需要修改 BELLMAN-FORD 算法，使其在计算过程中当没有边可以被松弛时停止循环，算法如下:

$BELLMAN-FORD(G, w, s)$
  $INITIALIZE-SINGLE-SOURCE(G, s)$
  $flag = true$
  $count = 0$
  $while\, flag\ =\ true$
    $for\, each\, edge(u, v)\ \in\ G.E$
      $flag = false$
      $RELAX(u, v, w, flag)$
      $count = count + 1$
      $if\ count\ >\ n$
        $return\ false$ //存在负环路
  $return\, true$

$RELAX(u, v, w, flag)$
  $if\ v.d > u.d + w(u, v)$
    $v.d = u.d + w(u, v)$
    $v.\pi = u$
    $flag = true$

24.3-8

分析：利用斐波那契堆来实现优先队列时，Dijkstra 算法的时间复杂度为 $O(VlgV + E)$, 是因为 EXTRACT-MIN 的时间复杂度为 O(lgV)，但由于该图每条边的权重都是正整数，所以可以直接用线性时间排序算法代替优先队列. 算法如下：

$DIJKSTRA(G, , w, s)$
  $INITIALIZE-SINGLE-SOURCE(G, s)$
  $S = \phi$
  $Q = G.V$
  $while\, Q \neq \phi$
    //对Q中各结点的d值进行计数排序，共W个数
    $u = Q[0]$//取出d值最小的结点
    $delete(Q[0])$ //在Q中删除$Q[0]$
    $S = S \cup \{u\}$
    $for\, each\, vertex\ v \in G.Adj[u]$
      $RELAX(u, v, w)$

算法时间变为 O(VW+E)

25.2-7

分析：$\phi_{ij}^{(k)}$ 是从结点 i 到 j 的一条中间结点都取自集合 {1,2,.....,k} 的最短路径上编号最大的结点。用 $\phi_{ij}^{(k)}$ 代替 $\Phi_{ij}^{(k)}$ 来修改 Floyed-Warshall 算法。$\phi_{ij}^{(k)}$ 的递归公式如下：

$$\phi_{ij}^{(0)} = NIL$$

$$\phi_{ij}^{(k)} = \begin{cases} \phi_{ij}^{(k-1)}, 若 d_{ij}^{(k-1)} \leq d_{ik}^{(k-1)} + d_{kj}^{(k-1)} \\ k, 若 d_{ij}^{(k-1)} \geq d_{ik}^{(k-1)} + d_{kj}^{(k-1)} \end{cases}$$

得到递归公式后修改 Floyed-Warshall 算法如下:

$Floyed - Warshall(W)$

$\quad n = W.rows$

$\quad D^{(0)} = W$

$\quad for\ k = 1\ to\ n$

$\quad\quad let\ D^{(k)} = (d_{ij}^{(k)}), \Phi^{(k)} = (\phi_{ij}^{(k)})\ be\ new\ n \times n\ matrixs$

$\quad\quad for\ i = 1\ to\ n$

$\quad\quad\quad for\ j = 1\ to\ n$

$\quad\quad\quad\quad if\ d_{ij}^{(k-1)} \leq d_{ik}^{(k-1)} + d_{kj}^{(k-1)}$

$\quad\quad\quad\quad\quad d_{ij}^{k} = d_{ij}^{k-1}$

$\quad\quad\quad\quad\quad \phi_{ij}^{(k)} = \phi_{ij}^{(k-1)}$

$\quad\quad\quad\quad else$

$\quad\quad\quad\quad\quad d_{ij}^{(k-1)} = d_{ik}^{(k-1)} + d_{kj}^{(k-1)}$

$\quad\quad\quad\quad\quad \phi_{ij}^{(k)} = k$

$\quad return\ D^{(n)}\ and\ \Phi^{(n)}$

修改 PRINT-ALL-PAIRS-SHORTEST-PATH 算法如下:

$PRINT - ALL - PAIRS - SHORTEST - PATH(\Phi, i, j)$

$\quad if\ \phi_{ij} = NIL$

$\quad\quad print\ i$

$\quad else$

$\quad\quad PRINT - ALL - PAIRS - SHORTEST - PATH(\Phi, i, \phi_{ij})$

$\quad\quad PRINT - ALL - PAIRS - SHORTEST - PATH(\Phi, \phi_{ij}, j)$

$\quad\quad print\ j$

Φ 类似于链式矩阵乘法的 s 表格, s 是对矩阵元素的划分, Φ 是对图中一条最短路径上结点的划分.