问题求解（三）第1周作业

黄奕诚161220049

September 11, 2017

TC Chapter 25

25-1.4

Proof. 欲证该算法中所定义的矩阵乘法是相关的，可证明$(A \cdot B) \cdot C = A \cdot (B \cdot C)$，根据EXTEND-SHORTEST-PATHS的矩阵乘法，可令$S = (A \cdot B) \cdot C, S' = A \cdot (B \cdot C)$.
$s_{i,j} = \sum_{t=1}^{n}(\sum_{k=1}^{n} a_{ik}bkt)c_{tj}$
$= \sum_{t=1}^{n}\sum_{k=1}^{n} a_{ik}bktc_{tj}$
$= \sum_{t=1}^{n}(a \cdot b)_{it} \cdot c_{tj}$
$= (a \cdot b \cdot c)_{ij}$
同理可证$s'_{ij} = (a \cdot b \cdot c)_{ij}$，由此可知其相关性. □

25-1.5

不妨联想一下多源最短路径算法的矩阵，这是一个$|V| \times |V|$的矩阵，而对于单源最短路径问题，只需要从中抽取出一行从结点s出发的行即可.可以定义一个向量与矩阵相乘的运算：$v_{i+1} = v_i W$，当计算v_{n-1}之后停止，则能够计算出从s出发的最短路径.与Bellman-Ford算法的关系：实际上，本节中讨论的算法关键是v_i表示最多经过i条边，在单源最短路径问题上，这与Bellman-Ford算法逐步进行RELAX操作的循环不变式是吻合的.

25-1.6

Algorithm 1 PREDECESSOR-MATRIX(L,W)

```
n = L.rows
for i = 1 to n do
    for j = 1 to n do
        for k = 1 to n do
            if L[i][j] = L[i][k] + W[k][j] then
                π[i][j] = k
            end if
        end for
    end for
end for
```

如此可在$O(n^3)$的时间内计算出前驱矩阵.

25-1.9

Algorithm 2 FASTER-ALL-PAIRS-SHORTEST-PATHS(W)

```
n = W.rows
L^(1) = W
m = 1
while m < n − 1 do
    let L^(2m) be a new n × n matrix
    L^(2m) = EXTEND-SHORTEST-PATHS(L^(m), W)
    m = 2m
end while
L' = EXTEND-SHORTEST-PATHS(L^(m), W)
if L' = L then
    print(No negative circle)
else
    print(Have negative circle!)
end if
return L^(m)
```

思路：在循环执行结束后，再多计算一步$L^{(2m)}$即可，若其与$L^{(2m)}$比较.若相等，则无负权重回路，反之则存在负权重回路.

25-1.10

可　以　先　在FASTER-ALL-PAIRS-SHORTEST-PATHS(W)中记录下所有存在于负权重的环中的结点，方法为：如果一个结点v存在于一个负权重的环中，则会出现在某一个m时，有$W_{ii} < 0$，每次遍历一遍所有结点，一旦出现上述情况，则记录进集合.在该算法结束后，通过判断各结点之间的连接情况，来计算负权重环路的长度.

25-2.2

可以模仿CLRS P407的定义，将25.1中的EXTEND-SHORTEST-PATHS中的$l'_{ij} = min(lij', l_{ik} + \omega_{kj})$改为$l'_{ij} = l'_{ij} \vee (v_{ik} \wedge \omega_{kj})$.同时定义：

$$\omega_{ij} = \{\begin{matrix} 0 & \quad i \neq j\text{且}(i,j) \notin E \\ 1 & \quad i = j\text{或}(i,j) \in E \end{matrix}$$

然后可以运行FASTER-ALL-PAIRS-SHORTEST-PATHS算法.

25-2.4

Proof. 该算法相对于FLOYD-WARSHALL算法，去掉了所有上标，以前的版本为$d_{ij}^{(k)} = min(d_{ij}^{(k-1)}, d_{ik}^{(k-1)} + d_{kj}^{(k-1)})$.即证明这个递归式执行后不改变右侧各个表达式的值.对于$d_{ij}^{(k)}$来说，由于其不经过k，与上标无关，故不改变值.而对于$d_{ik}^{(k-1)}$来说，从i到k的最短路径相当于从i到$k-1$的最短路径，两者没有区别，因而不会改变.所以去掉上标后的算法仍然正确. □

25-2.6

此题与25.1-10想法类似，主要在算法主体执行完毕后，检查所有的D_{ii}(也即对角线上的值)，若存在负数，则存在负权重的环.

25-2.8

初始化一个所有元素都为0的矩阵，如果一个结点i可以到达j，则将(i,j)填为1.目的是找到一个$O(VE)$的算法，来完成这个任务.
将所有边的权重赋值为1，有$\delta(i,j) < |E|$，对于一个给定的结点i，计算$\delta(i,j)$最多需要$|E|$的时间.再对每一个结点如此遍历，则该算法可以在$O(VE)$的时间内结束.

25-3.2

目的：用来解决非连通图的情况，通过文中给出的新结点s，可以顺利地计算出每一个结点的情况(包括两种无穷大).而如果使用图中的结点，则对于无法达到的点无法判断它的情况.

25-3.3

总有$\omega = \widehat{\omega}$.由于所有边的权重全为非负值，则有

$$d_{uv} = \delta(u,v) = \widehat{\delta}(u,v)$$

又有

$$\delta(u,v) = \widehat{\delta}(u,v) + h(v) - h(v)$$

因此总有$h(v) = h(u)$.
由此可推导出$\omega = \widehat{\omega}$.

25-2

a.

类　比　二　叉　堆　，　并　相　应　改　变　三　个　操　作　的　算　法　，　容　易　得　出INSERT,EXTRACT-MIN,DECREASE-KEY的渐进运行时间分别为$O(\log_d n), O(d \log_d n), O(\log_d n)$.当$d = \Theta(n^\alpha)$时，代入式子中，得$O(\frac{1}{\alpha}), O(\frac{n^\alpha}{\alpha}), O(\frac{1}{\alpha})$.摊还代价为$O(1), O(\lg n), O(1)$.

b.

$d = n^\epsilon$，在DIJKSTRA算法中使用d叉堆数据结构，运行时间为$O(\frac{n}{\epsilon} + \frac{2n^{\epsilon+1}}{\epsilon})$
也即$O(V^{1+\epsilon}) = O(E)$.

c.

对每一个结点都执行b中的算法，即可计算出所有结点对之间的最短路径.

d.

此时包含了权重为负值的边，不可使用DIJKSTRA算法，故不能沿用b,c的算法.这里可以使用JOHNSON算法，由于算法复杂度为$O(VE + V^2 \lg V)$，将$|E| = \Theta(V^{1+\epsilon})$代入，得$O(V^{2+\epsilon} + V^{2+p})$，其中$0 < \epsilon \leq 1$，只要满足$\epsilon > p$即可在$O(VE)$时间内计算结束.