数据结构与算法 II 作业 (12.15)

中国人民大学　信息学院　崔冠宇　2018202147

25.2-4　如前所述，Floyd-Warshall 算法的空间需求为 $\Theta(n^3)$，因为要计算 $d_{ij}^{(k)}$，其中 $i, j, k = 1, 2, \ldots, n$。请证明下面所列出的去掉所有上标的算法是正确的，从而将 Floyd-Warshall 算法的空间需求降低到 $\Theta(n^2)$。

```
FLOYD-WARSHALL(W)
    n = W.rows
    D = W
    for k = 1 to n
        for i = 1 to n
            for j = 1 to n
                d[i][j] = min(d[i][j], d[i][k] + d[k][j])
    return D
```

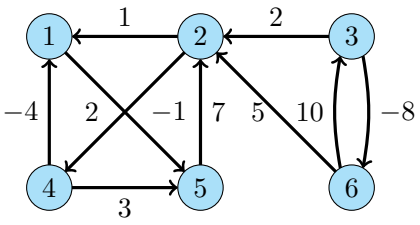
解: 算法正确性的证明：

原 Floyd-Warshall 算法的更新步骤是 $d_{ij}^{(k)} = \min(d_{ij}^{(k-1)}, d_{ik}^{(k-1)} + d_{kj}^{(k-1)})$，其中 d_{ij} 表示节点 i 到节点 j 的最短路径长度的上界。若最小值函数选后者，则表示经过中间节点 k 可使路径缩短（松弛）。改进后的算法只使用一个矩阵，当新的 d_{ij} 较小时，将会覆盖原来的矩阵。但是由于算法求的是路径长度的最小值，并不会因为原来的值被更小的值覆盖而引起错误。若 d_{ij} 被更小的值所覆盖，说明此时找到了更短的路，算法的正确性仍能得到保证。

空间需求：

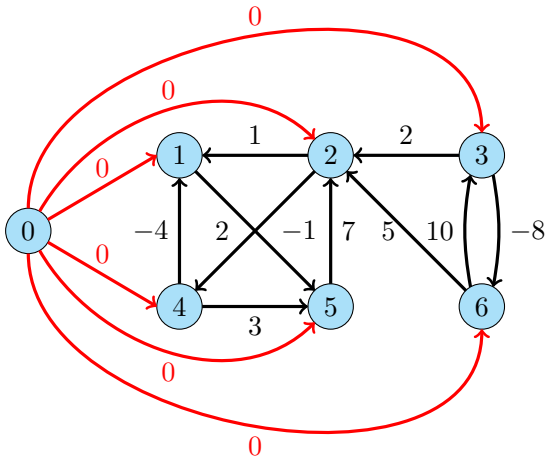
原 Floyd-Warshall 算法由于要保存 n 个 $n \times n$ 的矩阵 $D^{(k)}$，故空间需求为 $\Theta(n^3)$；而改进后的 Floyd-Warshall 算法只需要维护一个 $n \times n$ 的矩阵 D，于是空间需求降为了 $\Theta(n^2)$。

25.3-1　请在图 25-2 上使用 Johnson 算法来找到所有结点对之间的最短路径。给出算法计算出的 h 和 $\hat{w}$ 值。



解: 执行 Johnson 算法过程如下：

1. 增加源点 0 和源点指向各顶点的边，它们的距离为 0：



2.（假定边的遍历顺序为 $(0,1)$, $(0,2)$, $(0,3)$, $(0,4)$, $(0,5)$, $(0,6)$, $(1,5)$, $(2,1)$, $(2,4)$, $(3,2)$, $(3,6)$, $(4,1)$, $(4,5)$, $(5,2)$, $(6,2)$, $(6,3)$）运行 Bellman-Ford 算法：

(a) 初始化，设置各节点的 d：

v	0	1	2	3	4	5	6
$d[v]$	0	∞	∞	∞	∞	∞	∞

(b) 第一轮松弛后，各节点的 d：

v	0	1	2	3	4	5	6
$d[v]$	0	-4	0	0	0	-1	-8

(c) 第二轮松弛后，各节点的 d：

v	0	1	2	3	4	5	6
$d[v]$	0	-4	-3	0	0	-5	-8

(d) 第三轮松弛后，各节点的 d：

v	0	1	2	3	4	5	6
$d[v]$	0	-5	-3	0	-1	-5	-8

(e) 第四轮松弛后，各节点的 d：

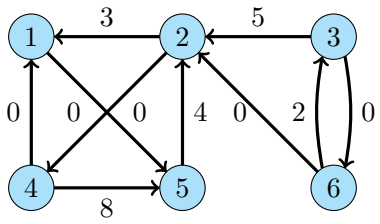
v	0	1	2	3	4	5	6
$d[v]$	0	-5	-3	0	-1	-6	-8

(f) 之后的松弛过程中，各节点的 d 不再变化，因此原图 G 没有负环。

3. 对每个节点 v，设置 $h(v)$ 为上面 Bellman-Ford 算法得到的 $d[v]$：

v	0	1	2	3	4	5	6
$h(v)$	0	-5	-3	0	-1	-6	-8

4. 对每条边 $(u,v) \in E(G)$，计算新的权重 $\hat{w}(u,v) = w(u,v) + h(u) - h(v)$：



5. 对得到的图 G_1，以每个顶点作为起点运行 Dijkstra 算法：

(a) `Dijkstra(G_1, 1)` 结果：

v	1	2	3	4	5	6
$d[v]$	0	4	∞	4	0	∞
$\pi[v]$	NIL	5	NIL	2	0	NIL

- $1 \to 2$ 最短路：$1 \to 5 \to 2$，权值 $w(1,2) = \hat{w}(1,2) - h(1) + h(2) = d[2] - h(1) + h(2) = 6$；
- $1 \to 4$ 最短路：$1 \to 5 \to 2 \to 4$，权值 $w(1,4) = \hat{w}(1,4) - h(1) + h(4) = d[4] - h(1) + h(4) = 8$；
- $1 \to 5$ 最短路：$1 \to 5$，权值 $w(1,5) = \hat{w}(1,5) - h(1) + h(5) = d[5] - h(1) + h(5) = -1$；
- 其他节点没有路径。

(b) `Dijkstra(G_1, 2)` $\sim$ `Dijkstra(G_1, 6)` 方法类似，略去。

26-3 (算法咨询) Gore 教授希望创办一家算法咨询公司。教授选出了算法中的 n 个重要子领域 (大概相当于本书的每个不同部分)，并用集合 $A = \{A_1, A_2, \cdots, A_n\}$ 予以表示。在每个子领域 A_k，教授可以用 c_k 美元聘请该领域的一位专家。咨询公司已经列出了一个潜在工作的集合 $J = \{J_1, J_2, \cdots, J_m\}$。为了完成工作 J_i，公司需要在子领域的子集合 $R_i \subseteq A$ 中聘请专家。每位专家可以同时从事多项工作。如果公司选择接受工作 J_i，则公司必须在集合 R_i 中的所有子领域中聘请专家，同时，公司从该项目中可以收入的营业额为 p_i 美元。

Gore 教授的工作是决定聘请哪些子领域的专家，接受哪些工作，以便使公司的净营业额达到最高，这里净营业额指的是从工作中获得的总收入减去聘请专家的总成本的差额。

考虑下面的流网络 G。该网络包含一个源结点 s，结点 $A_1, A_2, \cdots, A_n$，结点 $J_1, J_2, \cdots, J_m$，以及一个汇点 t。对于 $k = 1, 2, \cdots, n$，流网络包含一条边 (s, A_k)，其容量为 $c(s, A_k) = c_k$，而对于 $i = 1, 2, \cdots, m$，流网络包含一条边 (J_i, t)，容量为 $c(J_i, t) = p_i$。对于 $k = 1, 2, \cdots, n$ 和 $i = 1, 2, \cdots, m$，如果 $A_k \in R_i$，则图 G 包含边 (A_k, J_i)，其容量为 $c(A_k, J_i) = \infty$。

a. 证明：如果对于一个有限容量的切割 (S, T)，有 $J_i \in T$，则对于每个结点 $A_k \in R_i$，有 $A_k \in T$。

b. 请说明如何从图 G 的一个最小切割的容量和给定的 p_i 值中计算公司的最大净收入。

c. 请给出一个有效算法来判断哪些工作应该接受，哪些专家应该聘请。分析算法的运行时间，并以 m、n 和 $r = \sum_{i=1}^{m} |R_i|$ 来予以表示。

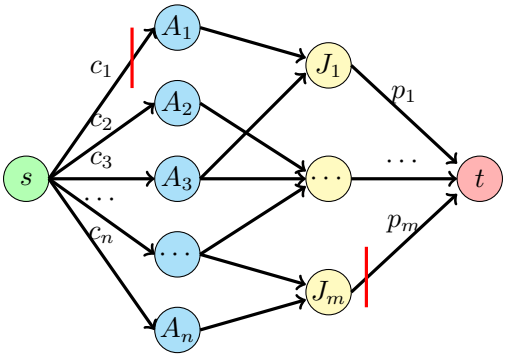


图 1: 流网络和切割的示意图

解： **a.** 证明：反证法。假设对于某 $J_i \in T$ 以及 $A_k \in R_i$ 有 $A_k \notin T$。根据切割的定义，必然有 $A_k \in S$。又根据题意，G 包含边 (A_k, J_i)，它通过了切割，且容量为 $c(A_k, J_i) = \infty$，这与切割的有限容量条件矛盾。

b. 显然图存在一个非无穷的切割（如 $S = \{s\}, T = V(G) - \{s\}$），因而必然存在一个非无穷的最小割。下面设 $A_S = \{A_k | A_k \in S\}$，$A_T = \{A_k | A_k \in T\}$，$J_S = \{J_i | J_i \in S\}$，$J_T = \{J_i | J_i \in T\}$。当 $A_i \in T$ 时，表示选择聘请领域 A_i 中的专家，当 $J_i \in T$ 时，表示选择接受工作 J_i。于是最小割的容量 $C = c(S,T) = \sum\limits_{A_i \in A_T} c_i + \sum\limits_{J_i \in J_S} p_i$。从而公司获得的利润为

$$\begin{aligned} P &= \sum_{J_i \in J_T} p_i - \sum_{A_i \in A_T} c_i \\ &= (\sum_{i=1}^{m} p_i - \sum_{J_i \in J_S} p_i) - \sum_{A_i \in A_T} c_i \\ &= \sum_{i=1}^{m} p_i - C \end{aligned}$$

c. 根据最大流最小割定理，最小割一定出现在最大流网络中。所以思路是先找到最大流，然后分出切割的两部分，并据此确定哪些工作应当接受，哪些领域的专家应当聘任。

1. 在流网络 G 上运行最大流算法，获取剩余网络 G_f。根据选择的算法不同，时间复杂度不同：
   - 若选择 Ford-Fulkerson 算法，时间复杂度是 $O(E|f^*|) = O((n+m+r)|f^*|)$；
   - 若选择 Edmonds-Karp 算法，时间复杂度是 $O(VE^2) = O((n+m+2)(n+m+r)) = O((n+m)(n+m+r))$。
2. 在剩余网络 G_f 上找寻源可达的节点，它们的集合构成了切割的 S，其余节点构成了切割的 T。使用 DFS，时间复杂度为 $O(V+E) = O(n+m+2+n+m+r) = O(n+m+r)$。
3. 遍历各 A_i 和各 J_i，若它属于 T，则聘请该领域的专家或接受该工作。使用数组，判定一个节点的是否属于集合 T 的时间复杂度可以达到 $O(1)$，因此该过程的时间复杂度为 $O(n+m)$。

综上，该算法的时间复杂度为 $O((n+m)(n+m+r))$。