Chapter V Shortest Paths: Label-Correcting Algorithms

- Label-setting与Label-correcting的区别与联系

  基本出发点是相同的：在每次迭代时为每个非源节点分配一个临时距离标签，作为源节点到节点最短路径的估计值。

  不同的是它们如何更新临时距离标签：
  Label-Setting Algorithm在每次迭代时将当前临时距离标签最小的更新为永久距离标签，直到所有的临时距离标签都更新为永久距离标签；

  Label-Correcting Algorithm在每次迭代时都有可能更新临时距离标签的值，直到最后一次迭代时所有的临时距离标签才成为永久距离标签。

算法	适用条件
（标准）Label Setting Algorithm	无环网络（Dijkstra Algorithm 无法求解含负权的网络最短路径问题）
Label Correcting Algorithm	所有类型（含有负弧长、负环）

1 最优性条件

定理1：（最短路径最优条件）
对于N中的每个点j，令d(j)代表从源点到j的有向路径长度，则d(j)代表的是最短路径长度，当且仅当它满足如下条件：

$$d(j) \le d(i) + c_{ij} \quad \text{for all } (i, j) \in A.$$

性质2：
首先，定义弧(i,j)相对于d(•) 的缩减弧长：$c_{ij}^d = c_{ij} + d(i) - d(j)$.
缩减弧长相关的三条性质：

(a) *For any directed cycle W,* $\sum_{(i,j)\in W} c_{ij}^d = \sum_{(i,j)\in W} c_{ij}$.
(b) *For any directed path P from node k to node l,* $\sum_{(i,j)\in P} c_{ij}^d = \sum_{(i,j)\in P} c_{ij} + d(k) - d(l)$.
(c) *If* $d(\cdot)$ *represent shortest path distances,* $c_{ij}^d \ge 0$ *for every arc* $(i, j) \in A$.

(a)对于所有有向环，缩减弧长之和等于所有弧长之和.
(b)对于k到l间的有向路径P，区间的所有缩减弧长之和等于区间弧长之和，再加上d(k)-d(l).
(c)如果d(•)代表的是最短距离，那么所有缩减弧长都大于0.
（由定理1直接推来）

如果一个网络中存在负环，那将不存在满足性质2的d(•)，此性质用来限制网络中不能包含负环。

2 通用标号校正算法

前提：网络中不包含负环。

通用标号校正算法在每个阶段维护一组距离标号d(•)，标号d(j)要么是∞，表示我们还没有发现从源到节点j的有向路径，要么是某个有向路径的长度对于每个节点j，我们也保持一个前置索引pred(j)，它记录了当前长度为d(j)的有向路径中在节点j之前的节点。在结束时，前置索引允许我们从源节点追踪到节点j的最短路径。

通用标号校正算法通过连续更新距离标签直到它们都满足最短路径最优性条件(定理1)的一般过程。

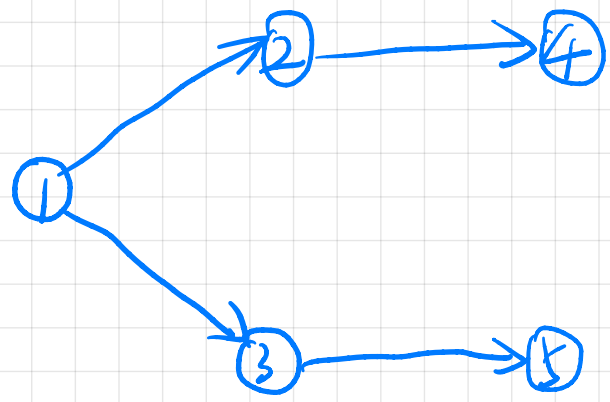
```
algorithm label-correcting;
begin
    d(s) := 0 and pred(s) := 0;
    d(j) := ∞ for each j ∈ N − {s};
    while some arc (i, j) satisfies d(j) > d(i) + cij do
    begin
        d(j) := d(i) + cij;
        pred(j) := i;
    end;
end;
```

Figure 5.1 Generic label-correcting algorithm.

核心思想：逐个检查不满最优性条件的距离标签，并根据 $d(j) := d(i) + c_{ij}$ 更新距离标签，同时前向节点也随之更新，直到所有的距离标签都满足最优性条件。复杂度为：$O(n^2C)$

案例演示：

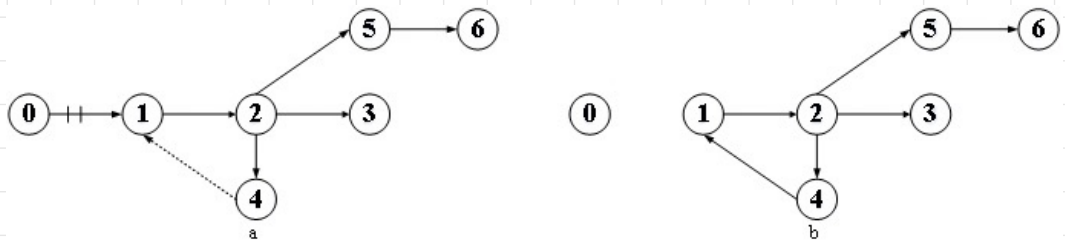
算法完毕后，所有弧都满足最优性条件，我们可以通过前向节点集合来生成节点1到其他节点的最短路径，例如，节点5的前向节点为3，节点3的前向节点为1，因此节点1到节点5的最短路径为1-3-5。通过这种方法我们可以得到一颗以节点1为根的前向节点树，此前向节树记录了根节点到其他子节点的最短路径。



前向节点树

前向节点树上的所有弧的Cdij均小于等于0.

前向节点集合并不一定会形成一棵以源节点为根的树，造成这种情况的原因是这个网路之中存在负环。



3　改进标号改正算法

通用标号改正算法默认对所有弧进行遍历，简单却低效。

改进算法通过引入一个可扫描列表SE_LIST，用来记录可能违反最优性条件的所有弧，每次向列表添加弧时，都会添加从单个节点发出的所有弧。如果SE_LIST为空，则表明所有弧都满足最优性条件，已找到最短路径，否则就从SE_LIST中选择一条弧，判定其是否违反最优性条件，并将其从SE_LIST中移除。如果弧违反了最优性条件，就用它更新节点的距离标签，同时更新节点的前向节点。

4　标号改正算法的O(nm)实现——FIFO

通过维护一个列表，当弧(i,j)被判定为违规弧时，将点j添加到列表中，按照先进先出的顺序，每次取出一个点，扫描从这个点出发的所有弧，直至列表为空，算法结束。

优点：1.时间效率高，在扫描过一个点上的所有弧后，就能保证从这个点出发的所有弧都满足最优条件，从而不再扫描这个点上的弧。

2.能够识别负环。如果该方法检查某个点的次数多于n-1，则网络中存在负环。

5　dequeue实现

将FIFO实现中的list改为dequeue，当添加点j时，如果j在list中出现过，则把j添加到list的最前面，否则添加到最后面。

6　全对最短路径问题

对于网络中每个点对$[i,j]$，$d[i,j]$是它们之间的最短距离，当且仅当它满足全对最短路径最优条件：$d[i, j] \le d[i, k] + d[k, j]$　　*for all nodes i, j, and k.*

标号算法在全路最短路问题的实现：

algorithm *all-pairs label-correcting*;
begin
　set $d[i, j] := \infty$ for all $[i, j] \in N \times N$;
　set $d[i, i] := 0$ for all $i \in N$;
　for each $(i, j) \in A$ **do** $d[i, j] := c_{ij}$;
　while the network contains three nodes i, j, and k
　　satisfying $d[i, j] > d[i, k] + d[k, j]$ **do** $d[i, j] := d[i, k] + d[k, j]$;
end;

Figure 5.6　Generic all-pairs label-correcting algorithm.

时间复杂度为O(n^3*C)

7 Floyd算法

时间复杂度为O(n^3)

定义dk[i,j]为：i和j之间只经过点1,2,...,k-1情况下的最短距离，
按照 $d^{k+1}[i, j] = min\{d^k[i, j], d^k[i, k] + d^k[k, i]\}$. 通过循环k确定两点之间的最短路。

map（i，j）表示节点i到j最短路径的距离，对于每一个节点k，检查
map（i，k）+map（k，j）小于map(i，j),如果成立，
map（i，j）=map（i，k）+map（k，j）；遍历每个k，每次更新的是除第k行和第k列的数。

```
algorithm Floyd–Warshall;
begin
    for all node pairs [i, j] ∈ N × N do
            d[i, j] : = ∞ and pred[i, j] : = 0;
    for all nodes i ∈ N do d[i, i] : = 0;
    for each arc (i, j) ∈ A do d[i, j] : = cij and pred[i, j] : = i;
    for each k : = 1 to n do
        for each [i, j] ∈ N × N do
            if d[i, j] > d[i, k] + d[k, j] then
            begin
                d[i, j] : = d[i, k] + d[k, j];
                pred[i, j] : = pred[k, j];
            end;
end;
```

效率较高，但数据量大的情况时间花费较长