# 问题求解（三）第 1 周作业

黄奕诚 161220049

2017 年 9 月 6 日

## TC Chapter 24

### Exercise 24.1-2

*Proof.* 由条件知 “图 $G$ 不包含从 $s$ 可以到达的权重为负值的环路”，故对于 $s$ 与 $V$ 中除去 $s$ 外的任意一顶点 $v$，都存在 $\delta(s,v)$。在初始化后 $v.d=\infty$ 随后通过 RELAX 操作，只有当 $v.d > u.d+\omega(u,v)$ 时，$v.d$ 会减少 (不再为 $\infty$)，与此同时有 $v.\pi = u$，于是 $v$ 具有前驱结点. 因此若存在 $s$ 到 $v$ 的路径，则 $v$ 必然执行过 RELAX，因此有 $v.d < \infty$. □

### Exercise 24.1-3

为方便起见，在改变 BELLMAN-FORD 算法的同时，微改 RELAX 算法，使其可以返回一个 “$v.d$ 是否改变” 的 *bool* 值.

**Algorithm 1** RELAX$(u,v,w)$

```
if v.d > u.d + ω(u,v) then
    v.d = u.d + ω(u,v)
    v.π = u
    return true
end if
return false
```

**Algorithm 2** BELLMAN-FORDE$(G,\omega,s)$

```
INITIALIZE-SINGLE-SOURCE(G, s)
for i = 1 to |G.V| − 1 do
    flag=false
    for each edge(u, v) ∈ G.E do
        if RELAX(u, v, w)==true and flag==false then
            flag=true
        end if
    end for
    if flag==false then
        return true
    end if
end for
for each edge(u, v) ∈ G.E do
    if v.d > u.d + ω(u, v) then
        return false
    end if
end for
```

## Exercise 24.1-4

只需要将第 7 行换为 $v.d = -\infty$ 即可，并且算法返回的值仍符合原先的要求。算法如下：

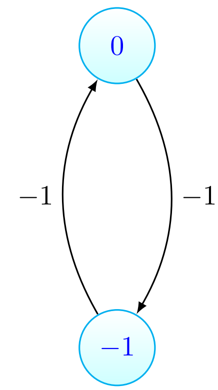
**Algorithm 3** BELLMAN-FORD$(G, \omega, s)$

1: INITIALIZE0SINGLE-SOURCE$(G, s)$
2: **for** $i = 1$ **to** $|G.V| - 1$ **do**
3:     **for** each edge $(u, v) \in G.E$ **do**
4:         RELAX$(u, v, \omega)$
5:     **end for**
6: **end for**
7: $flag$=**true**
8: **for** each edge$(u, v) \in G.E$ **do**
9:     **if** $v.d > u.d + \omega(u, v)$ **then**
10:         $v.d = -\infty$
11:         $flag$=**false**
12:     **end if**
13: **end for**
14: **return** $flag$

## Exercise 24.2-2

*Proof.* 对于拓扑排序中最后一个结点 $u$，假设其有后继结点，则与拓扑排序矛盾，因此不存在 $v \in G.Adj[u]$，因此算法第 4、5 行不会被执行。因此，改变后的算法仍然正确，与原算法等效。 □

## Exercise 24.3-2




根据 DIJKSTRA 算法，得到的下方结点的 $d$ 值为 -1，而实际上应该为 $-\infty$，因此得到了不正确的结果。

*Proof.* 定理 24.6 的证明不成立的关键在于：不能满足 $\delta(s, y) \leq \delta(s, u)$ □

## Exercise 24.3-4

**Algorithm 4** CHECK-OUTPUT$(G, \omega, s)$

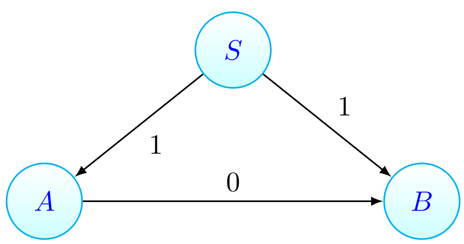
$G' = G$
DAG-SHORTEST-PATHS$(G', \omega, s)$
topologically sort the vertices of $G$
**for** each vertex $v_1$ in $G$, $v_2$ in $G'$, taken in topologically sorted order **do**
    **if** $v_1.\pi \neq v_2.\pi$ or $v_1.d \neq v_2.d$ **then**
        **return** **false**
    **end if**
**end for**
**return** **true**

### Exercise 24.3-7

该图将每条边 $(u,v)\in E$ 替换成 $\omega(u,v)$ 条具有单位权重的边，于是边数 $|E'| = \sum_{(u,v)\in E}\omega(u,v)$，因此结点个数变为 $|V'| = \sum_{(u,v)\in E}\omega(u,v)-1$.

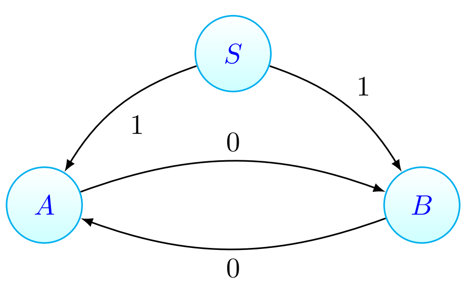
*Proof.* 由于广度优先搜索不计权重，每次从灰色结点搜与其相邻的白色结点（可以将所有权重视为 1），因此对于 $V$ 中一结点 $v_0$，其相邻的 $V$ 中其他结点与其距离越近，便越早搜索到，便越早涂上黑色；这与 Dijkstra 算法从优先队列中抽取结点的次序是一致的。 □

### Exercise 24.5-2




如图所示，共有 3 棵最短路径树，且三条边 $(S,A)$,$(S,B)$,$(A,B)$ 都存在最短路径树包含它，也存在最短路径树不包含它，满足题意。

### Exercise 24.5-5




与此同时，设定 $A.\pi = B$，且 $B.\pi = A$，于是 $G_\pi$ 中形成了一条环路。

### Exercise 24-2

**a.**

*Proof.* 假设盒子 $x$ 嵌套于盒子 $y$ 中，盒子 $y$ 嵌套于盒子 $z$ 中，也即存在排列 $\pi$，使得 $x_{\pi(1)} < y_1$，$x_{\pi(2)} < y_2$，...，$x_{\pi(d)} < y_d$，同时存在排列 $\tau$，使得 $y_{\tau(1)} < z_1$，$y_{\tau(2)} < z_2$，...，$y_{\tau(d)} < z_d$，因此存在一种排列使得 $x_{\pi(\tau(1))} < z_1$，$x_{\pi(\tau(2))} < z_2$，...，$x_{\pi(\tau(d))} < z_d$. 所以嵌套关系是传递的。 □

**b.**

盒子 $x$ 嵌套于盒子 $y$ $(*)$ 的充要条件为：按非递减序排列的 $x$ 各元素在各个位上都比 f 非递减序排列的 $y$ 各元素小。$(**)$

*Proof.* 若满足 $(**)$，则将非递减序中的 $x$ 与 $y$ 各元素两两配对，并使 $y$ 恢复先前次序，同时根据配对改变 $x$ 的次序，便存在使之成立的一种排列；若满足 $(*)$，则存在一个排列 $\pi$，使得 $x_{\pi(1)} < y_1$，$x_{\pi(2)} < y_2$，...，$x_{\pi(d)} < y_d$，则 $x$ 中的最小值必小于 $y$ 中的最小值（否则与 $(*)$ 矛盾），同理 $x$ 中第 $n$ 小值小于 $y$ 中第 $n$ 小值，于是得到了 $(**)$； □

**Algorithm 5** JUDGE-NESTED$(x,y,d)$

```
sort x, y in increasing sequence
for i = 1 to d do
    if x_i > y_i then
        return false
    end if
end for
return true
```

这个算法总时间为 $O(d\lg d)$.

c.

**Algorithm 6** LONGEST-SEQUENCE$(G, n, d)$

**for** each pair$(B_i, B_j)$ in $G$ **do**
  **if** JUDGE-NESTED$(B_i, B_j, d)$ **then**
    add $B_j$ to $G.Adj[B_i]$
  **end if**
**end for**
$H = \emptyset$，$dis = -\infty$
**for** each vertex $s$ in $G$ **do**
  $H_1 = \emptyset$
  **for** each vertex $u \in G.V - \{s\}$ **do**
    $u.color$=WHITE,$u.d = \infty$,$u.\pi = NIL$
  **end for**
  $s.color$=GRAY,$s.d = 0$,$s.\pi = NIL$,$Q = \emptyset$
  ENQUEUE$(Q, s)$
  **while** $Q \neq \infty$ **do**
    $u$=DEQUEUE$(Q)$
    **for** each $v \in G.Adj[u]$ **do**
      **if** $v.color == WHITE$ **then**
        $v.d = u.d + 1$，$v.\pi = u$
        ENQUEUE$(Q, v)$
      **end if**
    **end for**
  **end while**
  record $u.\pi$ until $s$ into $H_1$
  **if** $u.d > dis$ **then**
    $dis = u.d$，$H = H_1$
  **end if**
**end for**
**return** $H$

## Exercise 24-3

**a.**

可作出如下转化：

$$R[i_1, i_2] \bullet R[i_2, i_3] \bullet \cdots \bullet R[i_{k-1}, i_k] \bullet R[i_k, i_1] > 1$$

$$\ln(R[i_1, i_2]) + \ln(R[i_2, i_3]) + \cdots + \ln(R[i_{k-1}, i_k]) + \ln(R[i_k, i_1]) > 0$$

$$-\ln(R[i_1, i_2]) - \ln(R[i_2, i_3]) - \cdots - \ln(R[i_{k-1}, i_k]) - \ln(R[i_k, i_1]) < 0$$

我们可以通过 BELLMAN-FORD 算法来检验是否存在负权重的环，因为，从这个不等式可以看出：图中有负权重的环等价于存在不等式对应的这种套利情况。
由于 BELLMAN-FORD 算法的运行时间为 $O(VE)$，所以该算法的运行时间为 $O(VE)$。

**b.**

首先用 $a$ 中的算法来判断是否存在这种的一种序列，如果不存在，无需打印；如果存在，则需要进一步打印其中的一种序列。关键在于找出图中的负权重的环。首先调用 BELLMAN-FORD 算法，同时记录每一个结点的 $d$ 值；然后再 relax 每个结点足够多次，若某结点的 $d$ 值仍然在减少，则其属于某一个负权重的环。将这些 $d$ 值为 $-\infty$ 的结点标记，找出它们其中的一个连通图，便是这样的一个序列.