第八次算法作业

1. 假定$\mathbf{G}=(\mathbf{V},\mathbf{E})$为一带权重的有向图，并且图中存在一个权重为负值的环路。给出一个有效的算法来列出所有属于该环路上的结点。请证明该算法的正确性。

算法伪码:

```
NEGATIVE-BELLMAN-FORD(G, w, s)
    INITIALIZE-SINGLE-SOURCE(G,s)
    for i = 1 to |G, V| - 1
        for each edge (u,v) in G.E
            RELAX(u, v, w)
    for each edge(u,v) in G.E
        if v.d ＞ u.d + w(u,v)
            v.d = −∞
    for each vertex v in G.V
        if v.d = −∞
            MARK-NEGATIVE(v)

MARK-NEGATIVE(v)
    if v.π != NIL and v.π.d != −∞
        v.π.d = −∞
        MARK-NEGATIVE(v)
    else
        return
```

算法正确性:

在使用 Bellman-Ford 算法进行松弛操作，最后对每条边进行检查，如果仍有$\mathrm{v.d}>\mathrm{u.d}+\mathrm{w(u,v)}$，说明图中存在权重为负值的环路，将$\mathrm{v.d}$标记为负无穷。遍历 d 为负无穷的点，以此为起点进行 DFS，找到前驱节点并标记为负无穷，如果这条搜索路径的权重为负值并且下一个点已经访问过，那么这条搜索路径的权重就为负值。

2. Yen 对 Bellman-Ford 算法的改进

a. 证明：$\mathbf{G}_f$是无环的，且其拓扑排序为$\langle \boldsymbol{v}_1,\boldsymbol{v}_2,\cdots,\boldsymbol{v}_{|V|}\rangle$；$\mathbf{G}_b$是无环的，且其拓扑排序为$\langle \boldsymbol{v}_{|V|},\boldsymbol{v}_{|v|-1},\cdots,\boldsymbol{v}_1\rangle$；

证明:

假设$\mathrm{G_f}$是有环的，那么$\mathrm{G_f}$至少存在一条边$(\mathrm{v}_i,\mathrm{v}_j)$使得$\mathrm{i}>\mathrm{j}$，因此这条边$(\mathrm{v}_i,\mathrm{v}_j)\notin \mathrm{E}_f$，所以这条边不在$\mathrm{G_f}$中，这是一个矛盾，因此$\mathrm{G_f}$是无环的。

由于v中的边$(\mathrm{v}_i,\mathrm{v}_j)$满足$\mathrm{i}<\mathrm{j}$, 因此$\mathrm{G_f}$中的边只要有$\mathrm{i}<\mathrm{j}$, 那么拓扑排序时必然有$\mathrm{v}_i<\mathrm{v}_j$，所以拓扑排序的序列是$\langle v_1,v_2,\cdots,v_{|V|}\rangle$。

图G_b的证明类似。

b. 证明：在上述操作方式下，如果图 G 不包含从源结点 s 可以到达的权重为负值的环路，则在$\left\lceil\frac{|V|}{2}\right\rceil$遍松弛操作后，对于所有的结点$\mathbf{v}\in\mathbf{V}$，有$\mathbf{v.d}=\boldsymbol{\delta}(\mathbf{s},\mathbf{v})$

证明:

对于 V 中的所有点对，我们只需考虑$\delta(s,v)$有限的情况，即(s,v)存在最短路径。假设这条路径为$p=\langle s,v_1,v_2,\cdots,v_{k-1},v\rangle$，考虑边方向的变化，即$(v_{i-1},v_i)\in E_f$，$(v_i,v_{i+1})\in E_b$（$(v_{i-1},v_i)\in E_b$，$(v_i,v_{i+1})\in E_f$相似）的情况。因为p中最多有$|V|-1$条边，所以这样的变化最多有$|V|-2$次。

由拓扑排序的特点可知，图G_f与G_b都可以生成各自的最短生成路径，在进行 Bellman-Ford 算法时需要进行松弛操作的次数与$|V|-1$的关系为:

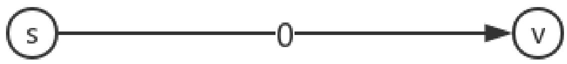
$\mathbf{\lvert V\rvert-1}$	第一条边的方向	松弛操作次数
偶数	前向	$(\lvert V\rvert-1)/2$
偶数	反向	$(\lvert V\rvert-1)/2+1$
奇数	前向	$\lvert V\rvert/2$
奇数	反向	$\lvert V\rvert/2$

综上，在$\left\lceil\frac{|V|}{2}\right\rceil$遍松弛操作后，对于所有的结点$v\in V$，有$v.d=\delta(s,v)$

c. 上述算法是否改善了 Bellman-ford 算法的渐进运行时间?

上述算法没有改善 Bellman-ford 算法的渐进运行时间。尽管松弛操作的次数从$|V|-1$次变为$\left\lceil\frac{|V|}{2}\right\rceil$次，但算法复杂度仍然为$O(V)$次松弛操作，也就是$O(VE)$的时间复杂度。

3. JOHNSON 算法是否有必要创建新的结点



如上图所示，$h(v)=\delta(v,v)=0, h(s)=\delta(v,s)=\infty$

因此$\widehat{w}(s,v)=0+\infty-0=\infty$，$\hat{\delta}(s,v)=\ \infty$，因此可以计算$d_{sv}=\infty+0-\infty\neq 0$

而$\delta(s,v)=0$，因此出现矛盾。

在强连接图中，对于任意的边(s,v)，都有$\delta(s,v)<\infty$，根据三角不等式，可以推出

$$\widehat{w}(s,v)=w(s,v)+h(s)-h(v)\geq 0$$

由于边的权重都是有限的，所以通过权重函数映射后重新赋予的权重并没有改变最短路径。所以强连接图满足（1）使用 Johnson 算法没有改变最短路径（2）对于任意的边(s,v)，都有$\hat{\delta}(s,v)<\infty$，即$d_{sv}=\hat{\delta}(s,v)+h(v)-h(s)$是有限的。所以在强连接图中该算法是正确的。

4. 动态图的传递闭包

a. 说明在加入一条新边到图 G 时，如何在$\mathbf{O(V^2)}$时间内更新图$\mathbf{G=(V,E)}$的传递闭包$\mathbf{G'=(V,E')}$。

用$|V|\times|V|$的矩阵T表示传递闭包，如果点 i 到 j 有一条路径，则$t_{ij}=1$。初始化矩阵时，若$i=j$，则$t_{ij}=1$，否则初始化为 0。

因此在边(u,v)加入时，更新图的算法为:

```
UPDATE-TRANSITIVE-CLOSURE(T, u, v):
    for i = 1 to |V|
        for j = 1 to |V|
            if t[i,u] == 1 and t[v,j]==1
                t[i,j] = 1
```

算法复杂度为$\mathrm{O}(\mathrm{V}^2)$

b. 给出一个图 G 和一条边 e，使得在将边 e 插入到图 G 后，更新传递闭包的时间复杂度为$\boldsymbol{\Omega}(\mathbf{V}^2)$，而不管使用的是那种算法。

考虑直线图$\mathrm{G} = (\mathrm{V}, \mathrm{E})$, 点集$\mathrm{V} = \{\mathrm{v}_1, \mathrm{v}_2, \cdots, \mathrm{v}_n\}$, 边集E满足在点$(\mathrm{v}_i, \mathrm{v}_{i+1})$间存在一条边，在边 e 插入之前，传递闭包矩阵上对角线和对角线以上的部分为 1，即有$|\mathrm{V}|(|\mathrm{V}|+1)/2$的数为 1。在插入边 e 之后，该图成为一个环，则传递闭包矩阵的所有位置都为 1，那么至少需要更新$|\mathrm{V}|^2 - \frac{|\mathrm{V}|(|\mathrm{V}|+1)}{2} = \Theta(\mathrm{V}^2)$个位置的数据。因此算法复杂度为$\Omega(\mathrm{V}^2)$。

c. 描述一个有效的算法，使得在将边加入到图 G 中时更新传递闭包。对于任意 n 次插入的序列，算法运行的总时间应该是$\sum_{i=1}^{n} t_i = O(V^3)$

更新的算法为：

```
EFFICIENT-UPDATE-TRANSTIVE-CLOSURE(T,u,v)
for i = 1 to |V|
    if t[i,u]==1 and t[i,v]==0
        for j=1 to |V|
            if t[v,j]==1
                t[i,j] = 1
```

证明：

在 n 次插入后，外层 for 语句的执行次数为$\mathrm{O}(\mathrm{nV}) = \mathrm{O}(\mathrm{V}^3)$，因为$\mathrm{n} \le |\mathrm{V}|^2$

只有在$\mathrm{t}[\mathrm{i}, \mathrm{v}] = 0$时，内层 for 循环才会执行，因为 T 中最多只有$|\mathrm{V}|^2$个元素，所以内层 for 循环执行的次数最多为$|\mathrm{V}|^2$次。

所以算法复杂度为$\mathrm{O}(\mathrm{V}^3)$