199

Algorithm Final (2016 Spring)

ID: B7033203o Name: 曾理硯

(Total: 235)

1. For the following statements, answer "×" and <u>correct the statement</u> if you think it is wrong; otherwise, answer "O": (9%)

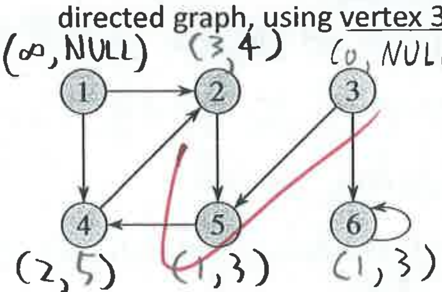
(X) (a) Given an undirected graph, the corresponding minimum spanning tree must be unique. 不一定，例如： 有 跟 兩種解

(O) (b) Suppose that vertices v_i and v_j are enqueued during the execution of BFS, and that v_i is enqueued before v_j. Then $v_i.d \le v_j.d$ at the time that v_j is enqueued.

(X) (c) In an AOE network, we could always reduce project length by speeding a critical activity. 有可能有兩條分開的critical控制全局，最好是找"共同必經的critical"才能減少花費時間

2. Show the d and π values that result from running breadth-first search on the following directed graph, using <u>vertex 3 as the source</u>. (6%)

我以(d, π)表示

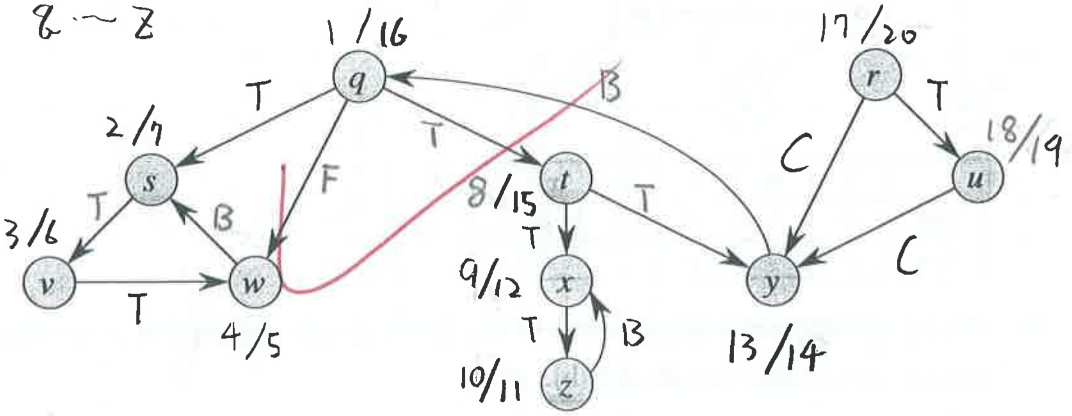


3. (a) Please enumerate four possible edge types in terms of the depth-first forest G_π produced by a depth-first search on a directed graph G. (4%)

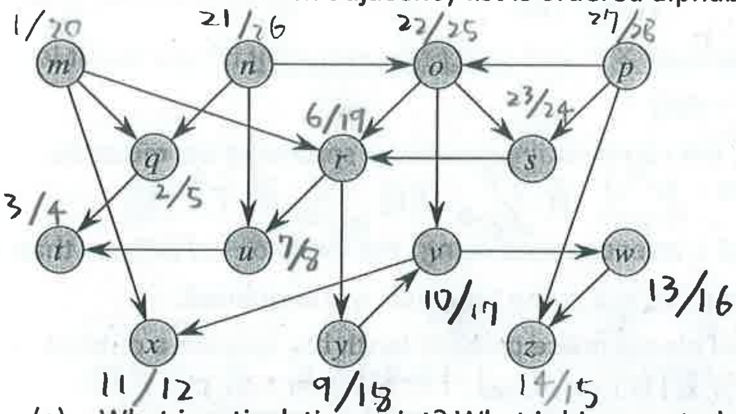
Tree edge
Forward edge
Back edge
Cross edge

(b) Show how depth-first search works on the graph below. Assume that the DFS procedure considers the vertices in <u>alphabetical order</u>, and assume that each adjacency list is ordered alphabetically. Show the discovery and finishing times for each vertex, and show the classification of each edge. (10%)

q ~ z



4. Show the ordering of vertices produced by TOPOLOGICAL-SORT when it is run on the following dag. Assume that the procedure considers the vertices in alphabetical order, and assume that each adjacency list is ordered alphabetically. (6%)



⇒由結束時間大的排序

⇒ p → n → o → s → m → r → y → v → w → z → x → u → q → t

−6

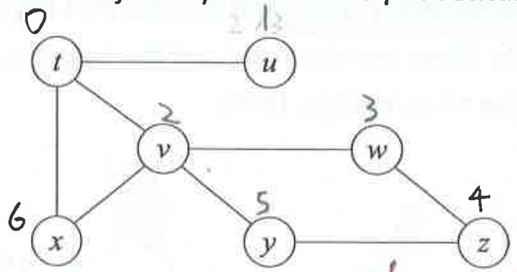
5. (a) What is articulation point? What is biconnected graph? What is biconnected component? (4%, 2%, 4%) 名詞解

articulation point：一張undirected graph 刪除 node n 及其相連 edge 會使 graph 分成數塊，node n 為 argulation point

biconnected graph：沒有 arculation point 的 graph

biconnected component：一張undirected graph 中的子集合，該子集合不含 articulation point 且彼此有路 且無法再加入任何 node 而不含 articulation point.

(b) Given an undirected graph below with $dfn(t) = 0$, please complete the following table. Assume that the procedure considers vertices in alphabetical order and that the adjacency lists are in alphabetical order. (10%)



	t	*u*	*v*	*w*	*x*	*y*	*z*
dfn	0	1	2	3	6	5	4
low	0	1	0	2	0	2	2

(c) Please identify articulation points of the above graph and explain your reasons based on the table obtained in (b). (6%)

−1

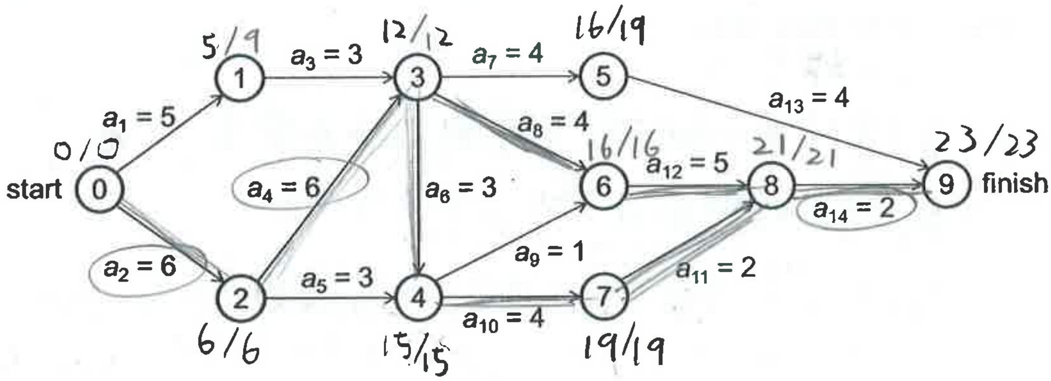
t: is root and have two child

v: is not root and have one child x that x.low < v.dfn

(d) Please identify the smallest biconnected component (i.e., the one with the least number of vertices) of the graph in (b). (2%)

(t)—(u)

6. Given the AOE network below, please answer the following questions:



(a) Use the forward-backward approach to obtain the early and late starting times for each activity. (14%)

	a_1	a_2	a_3	a_4	a_5	a_6	a_7	a_8	a_9	a_{10}	a_{11}	a_{12}	a_{13}	a_{14}
$e(i)$	0	0	5	6	6	12	12	12	15	15	19	16	16	21
$l(i)$	4	0	9	6	12	12	15	12	15	15	19	16	19	21

(b) What is the earliest time the project can finish? (3%)

23

(c) Which activities are critical? (4%)

-1

$a_2, a_4, a_6, a_8, a_{10}, a_{11}, a_{12}, a_{14}$

a_9

(d) Is there a single activity whose speed up would result in a reduction of the project length? If such an activity does not exist, please answer "No". Otherwise, please point out the activity. If there is more than one such activity, list them all. (3%)

a_2, a_4, a_{14}

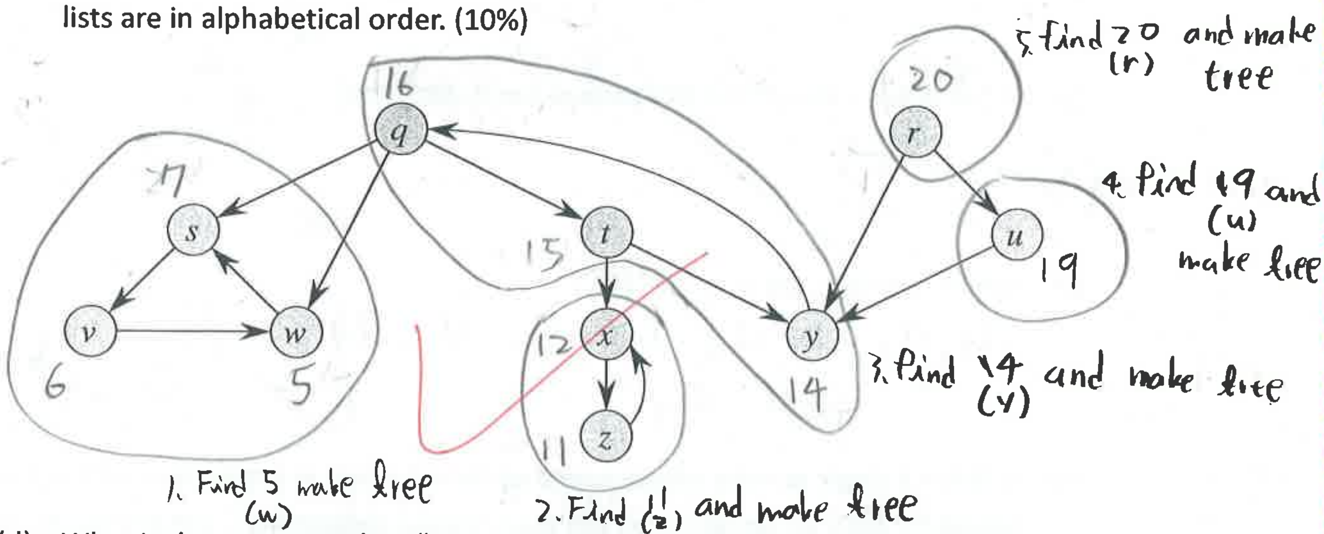
7. (a) Please define "strongly connected component". (4%)

一張graph中的子集合，該子集合各node皆可找到至少一條路通往集合中的任意node，且無法從graph中再找到一個node加入該集合而能維持以上性質，則此子集合為strongly connected component

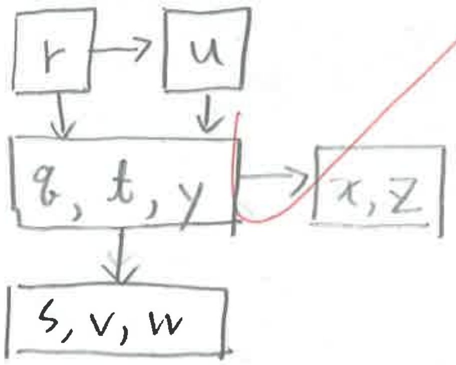
(b) Please explain how to compute the strongly connected component of a directed graph in four steps. (8%)

1. 做DFS
2. 反轉所有edge方向，時間mark不要變
3. 依結束時間由小到大做DFS
4. 輸出(每棵tree的所有成員就是strongly connected component)

(c) Show how the procedure STRONGLY-CONNECTED-COMPONENTS works on the graph below. Specifically, show the finishing times computed in line 1 (you could reference what you answered in 3.(b)) and the forest produced in line 3. Assume that the procedure considers vertices in alphabetical order and that the adjacency lists are in alphabetical order. (10%)



(d) What is the corresponding "component graph" of the strongly connected components in (c)? (5%)



8. We can interpret systems of difference constraints from a graph-theoretic point of view. For the following system of difference constraints:

$$
\begin{aligned}
x_1 - x_2 &\le 0\,, \\
x_1 - x_5 &\le -1\,, \\
x_2 - x_5 &\le 1\,, \\
x_3 - x_1 &\le 5\,, \\
x_4 - x_1 &\le 4\,, \\
x_4 - x_3 &\le -1\,, \\
x_5 - x_3 &\le -3\,, \\
x_5 - x_4 &\le -3\,.
\end{aligned}
$$

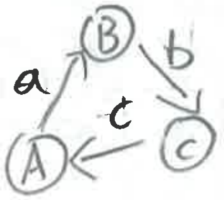
(a) Please prove the following theorem: Given a system $Ax \le b$ of difference constraints, let $G = (V, E)$ be the corresponding constraint graph. If G contains no negative-weight cycles, then $x = (\delta(v_0, v_1), \delta(v_0, v_2), \delta(v_0, v_3), \ldots, \delta(v_0, v_n))$ is a feasible solution for the system. If G contains a negative-weight cycle, then there is no feasible solution for the system. (10%)

−6

−2

假設存在有negative cycle 的 G 且存在 feasible solution

⇒ 存在



且 $a+b+c<0$

$$\Rightarrow \begin{cases} B - A \le a \\ C - B \le b \\ A - C \le c \end{cases} \Rightarrow 0 \le a+b+c$$

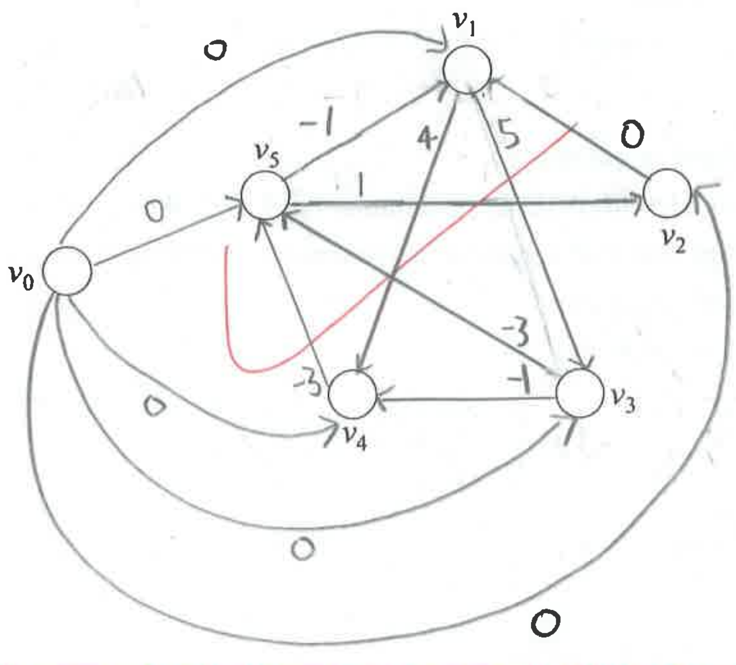
三式相加

⇒ 與題目假設 $a+b+c<0$ 不合

反證得證

(b) Please complete the corresponding constraint graph. (8%)

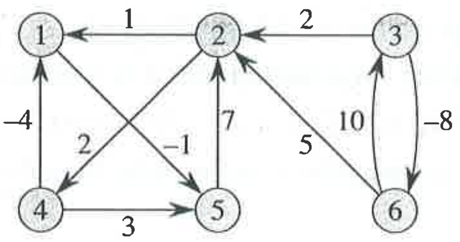
畫出



(c) Please find a feasible solution (with $x_3 = 0$) or determine that no feasible solution exists. (4%)

−4 一解

9. Let the matrix $L^{(m)} = (l_{ij}^{(m)})$, where $l_{ij}^{(m)}$ is the minimum weight of any path from vertex i to vertex j that contains at most m edges. Given the following weighted, directed graph, what is the corresponding $L^{(1)}$, $L^{(2)}$, and $L^{(8)}$? (3%, 5%, 8%)



−1
$$L^{(2)} = \begin{bmatrix} 0 & 6 & \infty & \infty & -1 & \infty \\ -2 & 0 & \infty & \cancel{\infty} & 0 & \infty \\ 3 & -3 & 0 & 4 & \infty & -8 \\ -4 & 10 & \infty & 0 & \cancel{-5} & \infty \\ 8 & 7 & \infty & 9 & 0 & \infty \\ 6 & 5 & 10 & 7 & \infty & 0 \end{bmatrix}$$

−1
$$L^{(1)} = \begin{bmatrix} 0 & \infty & \infty & \infty & -1 & \infty \\ 1 & 0 & \infty & 2 & \infty & \infty \\ \infty & 2 & 0 & \infty & \infty & -8 \\ -4 & \infty & \infty & 0 & 3 & \infty \\ \infty & 7 & \infty & \infty & 0 & \infty \\ \infty & 5 & 10 & \infty & \infty & 0 \end{bmatrix}$$

−1
$$L^{(8)} = \begin{bmatrix} 0 & 6 & \infty & 8 & -1 & \infty \\ -2 & 0 & \infty & 4 & -3 & \infty \\ -5 & -3 & 0 & -1 & -6 & -8 \\ -4 & 2 & \infty & 0 & -5 & \infty \\ 5 & 7 & \infty & 9 & 0 & \infty \\ 3 & 5 & 10 & 7 & 2 & 0 \end{bmatrix}$$

10. Given the code segment below (W is the input parameter):

```
n = W.rows
D^(0) = W
for k = 1 to n
    let D^(k) = (d_ij^(k)) be a new n × n matrix
    for i = 1 to n
        for j = 1 to n
            d_ij^(k) = min(d_ij^(k-1), d_ik^(k-1) + d_kj^(k-1))
return D^(n)
```

Please answer the following questions:

3 (3) (a) What is the above algorithm? (3%) (1) Bellman-Ford algorithm (2) Dijkstra's algorithm (3) Floyd-Warshall algorithm (4) Johnson's algorithm (5) Kruskal's algorithm (6) Prim's algorithm

−4 (b) What does $d_{ij}^{(k)}$ mean? (4%)

由 node i ~~到~~ node j 可使用 { node i 指出去的edge / node 0~node k 指出去的edge } 時所花費的 cost

(c) We can compute the predecessor matrix Π while the algorithm computes the matrices $D^{(k)}$. Please complete the following definition of $\pi_{ij}^{(k)}$. (14%)

$$\pi_{ij}^{(0)} = \begin{cases} \text{NIL} & \text{if } i=j \text{ or } d_{ij}=\infty \\ i & \text{if } i\neq j \text{ and } d_{ij}\neq\infty . \end{cases}$$

−4

$$\pi_{ij}^{(k)} = \begin{cases} \pi_{ij}^{(k-1)} & \text{if } d_{ij}^{(k-1)} \le d_{ik}^{(k-1)} + d_{kj}^{(k-1)} , \\ \pi_{kj}^{(k-1)} & \text{if } d_{ij}^{(k-1)} > d_{ik}^{(k-1)} + d_{kj}^{(k-1)} . \end{cases}$$

(d) Given a directed graph $G = (V, E)$, please define the transitive closure of G. (4%)

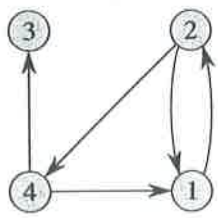
將一張graph取出任意二點a,b
若a有路通往b，則G上有a→b的edge
(G表示原graph node間通達情形)

(e) We can modify the given algorithm as follows to compute the transitive closure of a graph. Please fill in the blank in line 12 to complete the modified algorithm. (4%)

TRANSITIVE-CLOSURE(G)

```
1  n = |G.V|
2  let T^(0) = (t_ij^(0)) be a new n × n matrix
3  for i = 1 to n
4      for j = 1 to n
5          if i == j or (i, j) ∈ G.E
6              t_ij^(0) = 1
7          else t_ij^(0) = 0
8  for k = 1 to n
9      let T^(k) = (t_ij^(k)) be a new n × n matrix
10     for i = 1 to n
11         for j = 1 to n
12             t_ij^(k) = t_ij^(k-1) ∨ (t_ik^(k-1) ∧ t_kj^(k-1))
13 return T^(n)
```

(f) Given the following graph, please compute the matrices $T^{(0)}$, $T^{(1)}$, $T^{(2)}$, $T^{(3)}$, and $T^{(4)}$ based on the modified algorithm in (e). (15%)



$T^{(0)} = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 1 \end{bmatrix}$ $T^{(1)} \begin{bmatrix} 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 1 & 1 & 1 & 1 \end{bmatrix}$

$T^{(2)} = \begin{bmatrix} 1 & 1 & 0 & 1 \\ 1 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 1 & 1 & 1 & 1 \end{bmatrix}$ $T^{(3)} = \begin{bmatrix} 1 & 1 & 0 & 1 \\ 1 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 1 & 1 & 1 & 1 \end{bmatrix}$ $T^{(4)} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 \\ 1 & 1 & 1 & 1 \end{bmatrix}$

11. Given the code segment below (G, w, and r are input parameters):

1 **for** each $u \in G.V$
2 $\quad u.key = \infty$
3 $\quad u.\pi = \text{NIL}$
4 $r.key = 0$
5 $Q = G.V$
6 **while** $Q \neq \emptyset$
7 $\quad u = \text{EXTRACT-MIN}(Q)$
8 $\quad$ **for** each $v \in G.Adj[u]$
9 $\quad\quad$ **if** $v \in Q$ and $w(u, v) < v.key$
10 $\quad\quad\quad v.\pi = u$
11 $\quad\quad\quad v.key = w(u, v)$

Please answer the following questions:

(6) (a) What is the above algorithm? (3%) ~~(1)~~ Bellman-Ford algorithm ~~(2)~~ Dijkstra's algorithm ~~(3)~~ Floyd-Warshall algorithm ~~(4)~~ Johnson's algorithm (5) Kruskal's algorithm (6) Prim's algorithm

(b) Please explain why we set *r.key* to 0 in line 4. (3%)

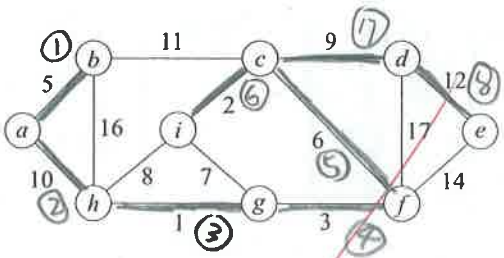
必須設初始點才會跑進迴圈中

(c) Please explain what do lines 8-11 do. (4%)

找到所有cross edge (在queue中) 中最小的
便找到要新加入的node是誰，把它拉進子集合中
更新queue，直到queue.size==0前 反覆步驟

-2

(d) Given the graph below, and the root vertex is a. Please highlight edges selected by the algorithm and number these edges based on the selection order. (5%)



12. Given the code segment below (G and w are input parameters):

```
1  A = ∅
2  for each vertex v ∈ G.V
3      MAKE-SET(v)
4  sort the edges of G.E into nondecreasing order by weight w
5  for each edge (u, v) ∈ G.E, taken in nondecreasing order by weight
6      if FIND-SET(u) ≠ FIND-SET(v)
7          A = A ∪ {(u, v)}
8          UNION(u, v)
9  return A
```

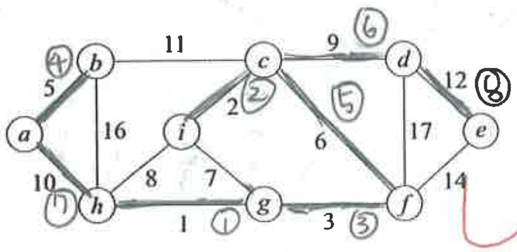
Please answer the following questions:

(5) (a) What is the above algorithm? (3%) ~~(1)~~ Bellman-Ford algorithm ~~(2)~~ Dijkstra's algorithm ~~(3)~~ Floyd-Warshall algorithm ~~(4)~~ Johnson's algorithm (5) Kruskal's algorithm (6) Prim's algorithm

(b) Please explain what do lines 5-8 do. (4%)

由cost小的edge一一檢查
如果該edge的兩個node在同一個group，不使用
反之，加入該edge並整理group

(c) Given the graph below, please highlight edges selected by the algorithm and number these edges based on the selection order. (5%)



13. Given the code segment below (G, w, and s are input parameters):

```
INITIALIZE-SINGLE-SOURCE(G, s)
for i = 1 to |G.V| − 1
    for each edge (u, v) ∈ G.E
        RELAX(u, v, w)
for each edge (u, v) ∈ G.E
    if v.d > u.d + w(u, v)
        return FALSE
return TRUE
```

Please answer the following questions:

(1) (a) What is the above algorithm? (3%) (1) Bellman-Ford algorithm (2) Dijkstra's algorithm (3) Floyd-Warshall algorithm (4) Johnson's algorithm (5) Kruskal's algorithm (6) Prim's algorithm

-3

(b) If the procedure returns TRUE, what does it mean? (3%)

存在 negative cycle

(c) Given the input graph on the left, where the source is vertex s and the d values appear within the vertices. After executed by the given algorithm, please give the resulting graph on the right. *__Note: Please highlight edges to indicate predecessor values.__* (6%)