209

Algorithm Final (2016 Spring)

ID: B10315001 Name: [illegible]

(Total: 235)

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

(×) (a) Given an undirected graph, the corresponding minimum spanning tree must be ~~unique~~. can be more than one

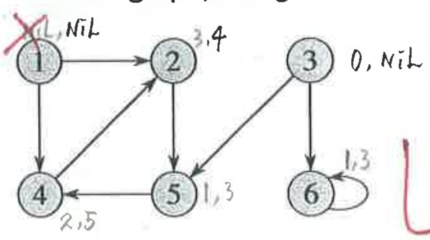
(×) (b) Suppose that vertices v_i and v_j are enqueued during the execution of BFS, and that v_i is enqueued before v_j. Then ~~$v_i.d \le v_j.d$~~ $v_i.d > v_j.d$ at the time that v_j is enqueued.

−4

(×) (c) In an AOE network, we could always reduce project length by speeding a critical activity. 必需要在所有critical path上交集的活動才會加速

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

−1

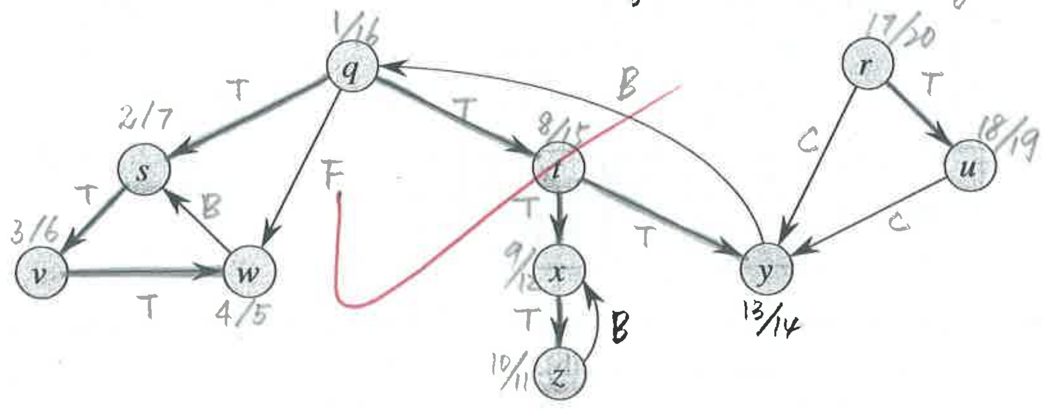


3. (a) Please enumerate four possible edge types in terms of the depth-first forest G_π produced by a depth-first search on a directed graph G. (4%)

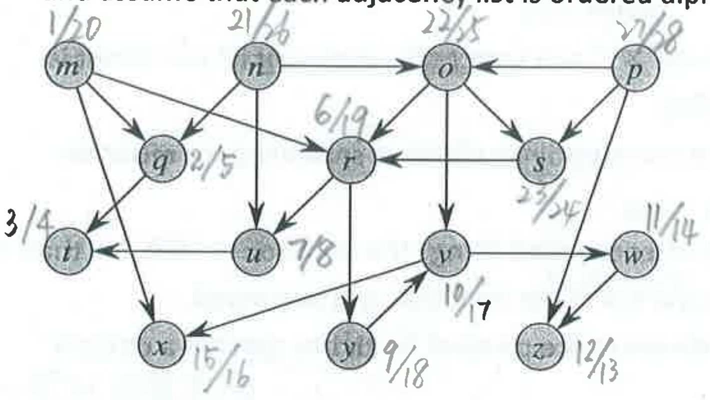
tree edge, forward edge, back edge, cross edge

(b) Show how depth-first search works on the graph below. Assume that the DFS procedure considers the vertices in alphabetical order, 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, r, s, t, u, v, w, x, y, 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%) dfs



t, q, u, z, w, x, v, y, r, m, s, o, n, p #

-6

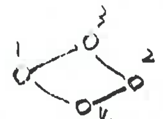
5. (a) What is articulation point? What is biconnected graph? What is biconnected component? (4%, 2%, 4%)

articulation point: If delete articulation point, and all edges incide with that point, the graph will break into two (or more) connected component.

* connected component: A graph that all vertex is connected by edge(s).

biconnected graph: A graph that has no articulation point.

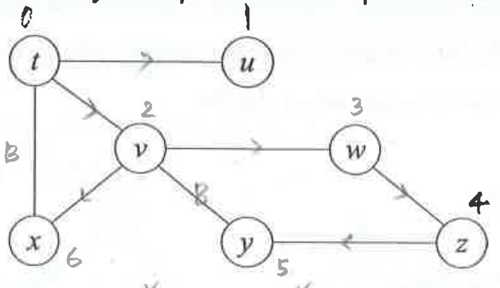
biconnected component: A graph that from a vertex to the other, has more than one path.

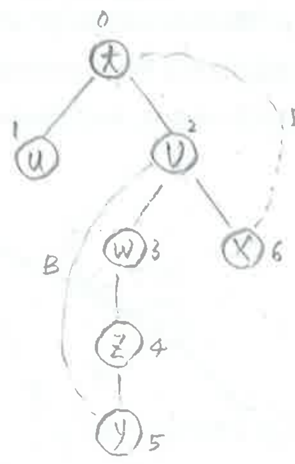


1→2 : 1→3→2 or 1→4→2

-4

(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%)





low(u) = min{dfn(u), min{low(w)}, dfn(w) (u,w) is B} w: descendant of u

x: 6, x, dfn(t)

y: 5, x, 2.

z: 4, 2.

	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%)

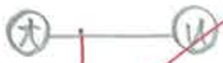
-1

Child	u,v	NiL	w,x	z	NiL	NiL	y
Childlow	1,0	NiL	2,0	2	NiL	NiL	2

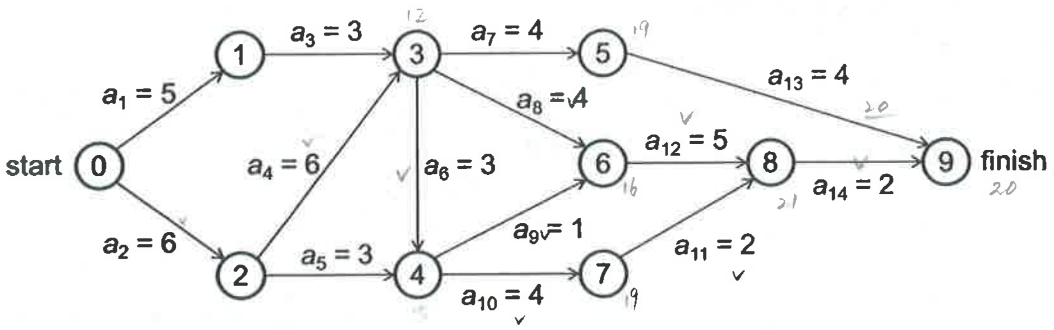
Childlow ≥ dfn → articulation point

∴ t and v #

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



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%)

$21+2=23$ #

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

$l(i)-e(i)=0 \Rightarrow$ critical

$\therefore$ $a_2, a_4, a_6, a_8, a_9, a_{10}, a_{11}, a_{12}, a_{14}$ #

(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, please list them all. (3%)

a_2, a_4, a_{14} #

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

-1

A graph that every pair of vertex can reach each other. In other word, if $u \overset{p}{\rightsquigarrow} v$ exist, then $v \overset{p'}{\rightsquigarrow} u$ also exists #

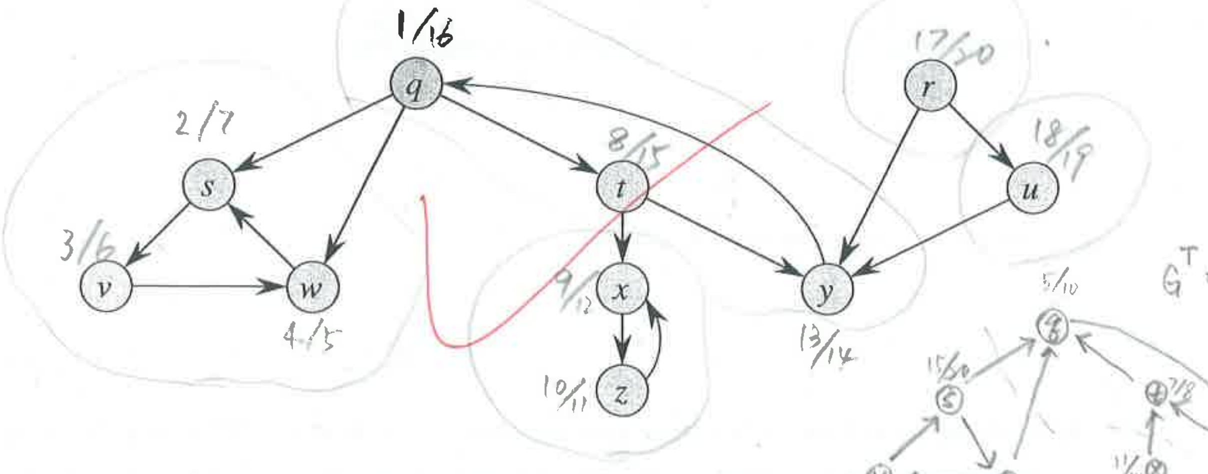
$u, v \in G.V$

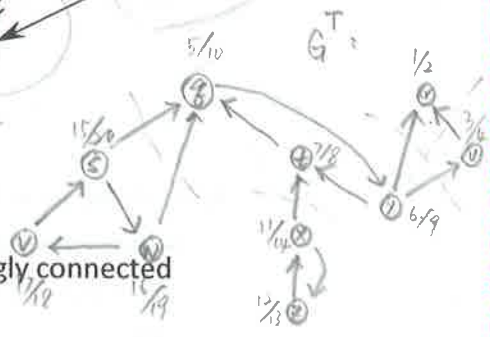
(b) Please explain how to compute the strongly connected component of a directed graph in four steps. (8%)

-1

1. Run DFS, mark the tree edges
2. Reverse all the graph edges
3. Run again DFS in reverse time order
4. Then we can find strongly connected component by discover and finish time.

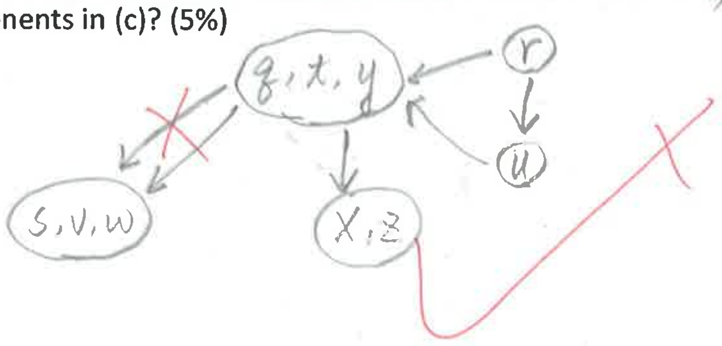
(c) Show how the procedure STRONGLY-CONNECTED-COMPONENTS works on the graph below. Specifically, show the finishing times computed in line 1 (you could reference what you answered in 3.(b)) and the forest produced in line 3. Assume that the procedure considers vertices in alphabetical order and that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





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

-1



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 $x = (\delta(v_0, v_1), \delta(v_0, v_2), \delta(v_0, v_3), \dots, \delta(v_0, v_n))$ is a feasible solution for the system. If G contains a negative-weight cycle, then there is no feasible solution for the system. (10%) I II

$w(c) < 0$

(I):

$x_j - x_i \le b_k \rightarrow$ feasible sol

x_j: $\delta(v_0, v_j)$

x_i: $\delta(v_0, v_i)$

b_k: $w(i, j)$

By triangular inequality:

$\delta(v_0, v_j) \le \delta(v_0, v_i) + w(i, j)$

$\Rightarrow \delta(v_0, v_j) - \delta(v_0, v_i) \le w(i, j)$

$\Rightarrow x_j - x_i \le b_k$, we prove #

(II) suppose $c = \langle v_1, v_2, \dots, v_k \rangle$ is in G, and $v_1 = v_k$ (cycle)

$~~x_2~~ - x_1 \le w(v_1, v_2)$

$~~x_3 - x_2~~ \le w(v_2, v_3)$

$+)\ x_k - ~~x_{k-1}~~ \le w(v_{k-1}, v_k)$

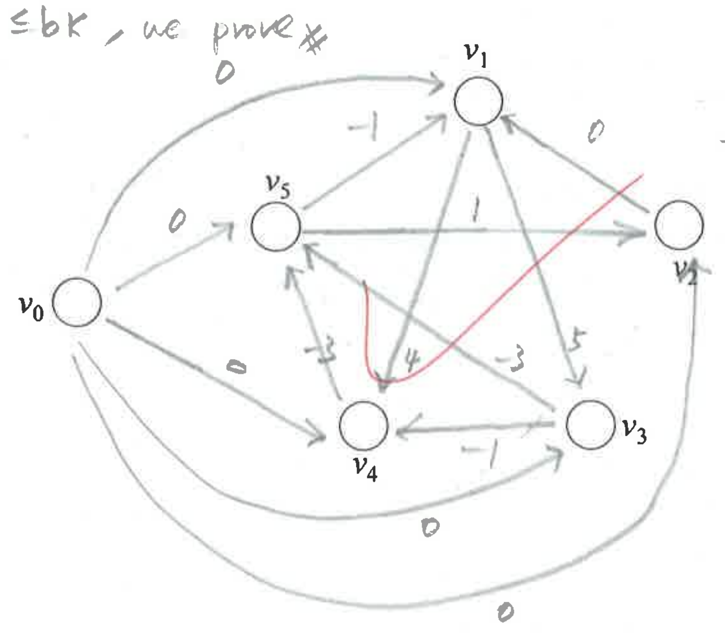
$x_k - x_1 \le w(c)$

$\because (v_1 = v_k) \Rightarrow x_k - x_1 = 0 \Rightarrow 0 \le w(c)$

contradicts $w(c) < 0$

$\Rightarrow$ no feasible sol. #

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



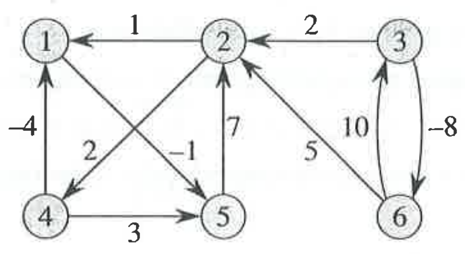
5次 Bellman-Ford

v_1,	v_2,	v_3,	v_4,	v_5
0	0	0	0	0
-1	0	0	-1	-3
-4	-2	0	-1	-4
-5	-3	0	-1	-4
〃	〃	〃	〃	〃

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

$(x_1, x_2, x_3, x_4, x_5) = (-5, -3, 0, -1, -4)$ #

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%)



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

-1

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

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

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) (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

-2

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

path from vertex i to vertex j, can go through vertex 1, 2, ..., k

Ex. $d_{13}^{(2)}$, from 1 → 3, can go through 1, 2, ~~3~~, ~~4~~ ...

(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%)

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

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

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

$$t_{ij}^{(k)} = \begin{cases} 1, & (i,j) \in E \\ 0, & \text{otherwise} \end{cases} \qquad t_{ij}^{(0)} = \begin{cases} 0, & \text{if } i\neq j \text{ and } (i,j)\notin E \\ 1, & \text{if } i=j \text{ or } (i,j)\in E \end{cases}$$

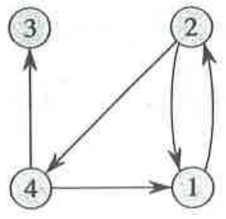
-3

$$t_{ij}^{(k)} = t_{ij}^{(k-1)} \vee (t_{ik}^{(k-1)} \wedge t_{kj}^{(k-1)})$$

(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 $n \times n$ matrix
3  for $i = 1$ to $n$
4      for $j = 1$ to $n$
5          if $i == j$ or $(i,j) \in G.E$
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 $n \times n$ matrix
10     for $i = 1$ to $n$
11         for $j = 1$ to $n$
12             $t_{ij}^{(k)} = \underline{t_{ij}^{(k-1)} \vee (t_{ik}^{(k-1)} \wedge t_{kj}^{(k-1)})}$
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%)



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

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

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

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

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

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%)

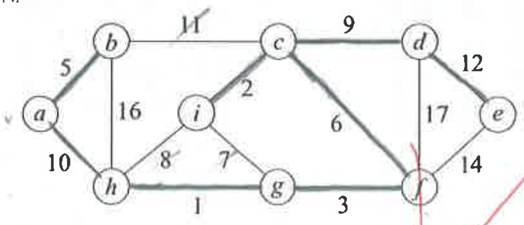
initialize vertex r, for later comparison.

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

scan every vertex adjacent to u
if v is in queue and edge (u,v)'s weight is smaller than v's key (Greedy choice)
set v's parent to u, also renew v's key to (u,v)'s weight.

(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%) Greedy choice

請寫答(選順序)



/ : discard (form cycle)

依序: (a,b), (a,h), (h,g), (g,f)
(c,f), (c,i), (c,d), (d,e)
#

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:

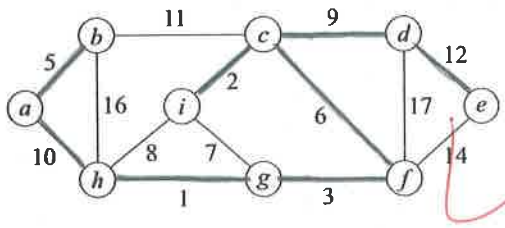
(5) (a) What is the above algorithm? (3%) (1) Bellman-Ford algorithm (2) Dijkstra's algorithm (3) Floyd-Warshall algorithm (4) Johnson's algorithm (5) Kruskal's algorithm (6) Prim's algorithm

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

-1

pick weighted edge in increasing order. if the picked-up edge does not form a cycle, we add it to A; otherwise, we discard the edge.

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



依序：(h, g)、(c, i)、(g, f)、(a, b) (c, f)、(c, d)、(a, h)、(d, e) #

* discard (form cycle): (i, g), (i, h) (b, c)

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

```
1  INITIALIZE-SINGLE-SOURCE(G, s)
2  for i = 1 to |G.V| - 1
3      for each edge (u, v) ∈ G.E
4          RELAX(u, v, w)
5  for each edge (u, v) ∈ G.E
6      if v.d > u.d + w(u, v)
7          return FALSE
8  return TRUE        no negative
```

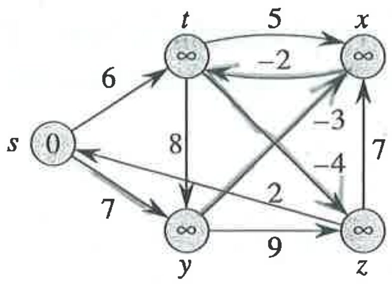
Please answer the following questions:

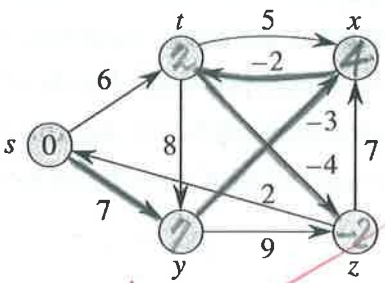
(1) (a) What is the above algorithm? (3%) (1) Bellman-Ford algorithm (2) Dijkstra's algorithm (3) Floyd-Warshall algorithm (4) Johnson's algorithm (5) Kruskal's algorithm (6) Prim's algorithm

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

TRUE ⇒ no negative weight 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%)





t ,	x ,	y ,	z
6	∞	7	∞
6	4	7	2
2	4	7	2
2	4	7	-2