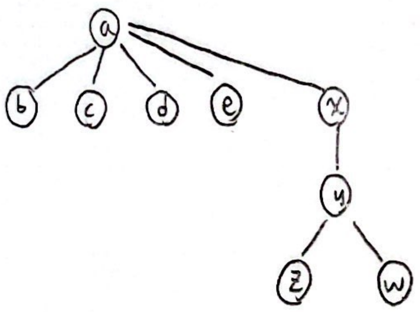
1.

110062619

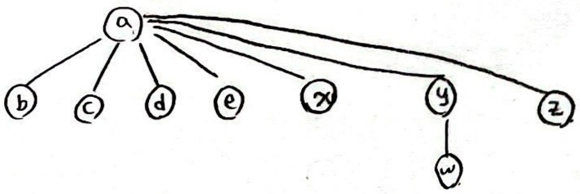
楊淨富.

(i) Union (a, x).

⇒. Union-by-size

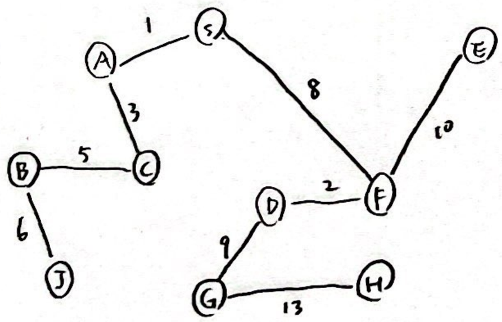


(ii). find (Z) ⇒. return a



2.

(a). 1 → 2 → 3 → ~~4~~ → 5 → 6 → ~~7~~ → 8 → 9 → 10 → ~~11~~ → ~~12~~ → 13



10個 nodes 所以挑完 9個 edges

即 done.

(b) Unique.

因為每條 edge 的 weight 皆 unique

3.

(a).

Iteration		selected	dist [A]	[B]	[C]	[D]	[E]	[F]	[G]	[H]	[J]	[S]
0.			1{s}.	∞	4{s}.	∞	11{s}	8{s}	∞	∞	∞	0.
1.	{s.}.	{A}.	1{s}.	8{A}.	4{s}.	∞	11{s}	8{s}	∞	∞	∞	
2.	{s, A}.	{C}.		8{A}.	4{s}	16{C}	11{s}	8{s}	∞.	∞	∞	
3.	{s, A, C}.	{B.}.		8{A}.		16{C}.	11{s}	8{s}	∞.	∞	14.	
4.	{s, A, C, B}	{F}.				10{F}.	11{s}	8{s}	∞.	∞.	14	
5.	{s, A, C, B, F}.	{D}.				10	11.		19	24	14.	
6.	{S, A, C, B, F, D}	{E}.					11.		19	24	14.	
7.	{S, A, C, B, F, D, E}	{J.}							19.	24	14	
8.	{S, A, C, B, F, D, E, J}	{G.}.							19.	24		
9.	{S, A, C, B, F, D, E, J, G}	{H.}.								24.		

order: $S \to A \to C \to B \to F \to D \to E \to J \to G \to H$.

(b).

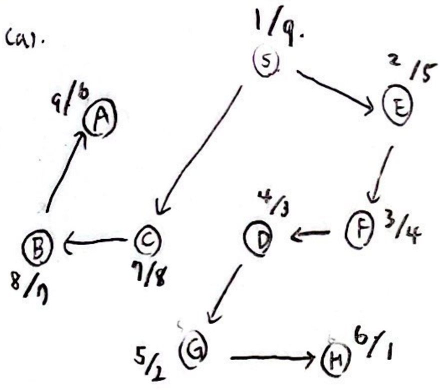
Not unique.

因為在 iteration 3 時, 可發現此時 (S 到 B) 和 (S 到 F) 之距離皆是 8,

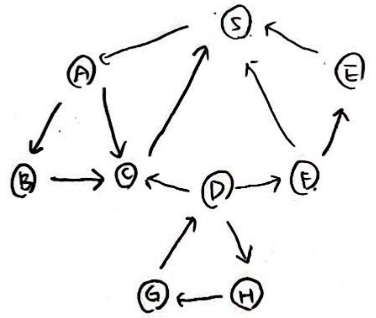
故可先挑 B 或 F, ⇒ 非唯一 order.

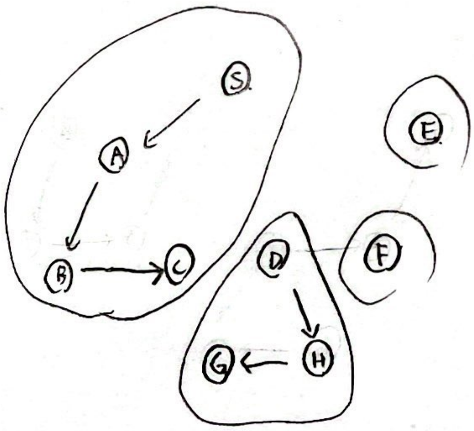
4.

(a).



G^T:





# of SCCs = 4.

(b).

True.

Prove by contradiction

Assume to the contrary, 某个 SCC 內 2 点, A, B, 沒有一个 simple cycle 可包含它們.

則 WLOG, 假設 G 中, A $\rightsquigarrow$ B (A 有 path 到 B).

但 G^T 中 A $\not\rightsquigarrow$ B (A 無 path 到 B).

矛盾, 因為 A 和 B 在同一个 SCC.

5.

用 BFS Algo.

1. Mark ~~s~~ u as discovered in round 0.

2. For round $k = 1, 2, 3, \dots$

For (each ~~x~~ s discovered in round $k-1$).

{

mark ~~x~~ s as visited;

visit each neighbor v of ~~x~~ s.

if (v is not visited and not discovered).

mark v as discovered in round k;

}.

Stop if no vertices were discovered in round $k-1$.

Correctness: A vertex v is discovered in round k if and only if shortest distance of v from $\boxed{\text{source } s}$ is k.

$= u.$

(can be proved by induction.).

shortest path length from v to u.

i=1. v 到 u 的 sp = 1. 成立. u can be visited by v in round 1 ✓

令 $i = k-1$. 時成立.

則 $i = k$ 時. v 可在 round $k-1$ 到集合 s,

且 u can be visited by (s 再走一條边

理论: $i = k$.).

∴ 成立

Time: Since no vertex is discovered twice, and each edge is visited at most twice.

total time: $O(|V| + |E|)$.

6.

```
Bellman-Ford ( G, w, s ).      (w: weight, s: source)
{
  n = |V|.
  for (i = 1 to n-1).
      for each (u,v) ∈ G.E
          Relax (u,v,w).

  for each (u,v) ∈ G.E          多做一次 relax
      if v.d > u.d + w(u,v)
          return "有負 cycle"
  else
          return "沒有負 cycle"
}.
```

```
Relax (u,v,w)        distance.
{
  if v.d > u.d + w(u,v)
      v.d = u.d + w(u,v)
}.
```

Time: 第一了 for-loop (n-1)次的 relax, 每次 relax 要 check O(|E|) 條邊.

∴ O(|V||E|).

Correctness: 如果 source s. 可到達某個負 "cycle", 那若再多一次 relax. 必可使 s"到負 cycle 上"各点距離值更小", 在此情況下, shortest path 問題也不 well-defined.

7.

By DFS algorithm (listed below).

An edge (u, v) is a bridge $\Leftrightarrow$ None of the vertices v and its descendants in the DFS traversal tree has a back-edge to vertex u or any of its ancestors,

i.e., there is no other way from u to v except for (u, v)

let $s[u]$: entry time for u.

$$low[u] = \begin{cases} s[u] \\ s[p] & \text{for all } p \text{ for which } (u, p) \text{ is a back edge} \\ s[v] & \text{for all } v \text{ for which } (u, v) \text{ is a tree edge.} \end{cases}$$

Correctness :

- There is a back edge from vertex u or one of its descendants to one of its ancestors if and only if vertex u has a child v for which $low[v] \le s[u]$
- If $low[v] = s[u]$, the back edge comes directly to u, otherwise, it comes to one of the ancestors of u.
- Thus, the edge (u, v) in the DFS tree is a bridge $\Leftrightarrow$ $low[v] > s[u]$

Algo. of DFS and Time complexity :

```
DFS (G).
{
  for each vertex u ∈ G.V
      u.color = white;
      u.π = NULL;   // π : parent.
  t = 0;
  for each u ∈ G.V
  {
      if (u.color == white).
      {
          DFS-visit (G, u)
      }.
  }
}.
```

```
DFS-visit (G, u)
{
   t++;
   u.d = t;   // d : discover time
   u.color = gray
   for each v ∈ G.adj[u].   // adj: adjancency list
   {
      if (v.color == white)
      {
          v.π = u
          DFS-visit (G, v).
      }.
   }.
   u.color = black;
   t++;
   u.f = t;   // f: finish time.
}.
```

Time: 以 adjancency list 存 graph，因為 G 中每点会 run DFS-visit (G, u) 一次. $\therefore O(|V|)$.

另外，DFS-visit (G, u) 中的 for loop，因為每條边最多也只进迴圈二次 $\therefore O(|E|)$.

因此，$O(|V| + |E|)$.

8.

用 DFS algorithm 找 s 到 v 的所有 paths.

且過程中 若當前 u 的 node 給予 weight，該 weight 為從哪條邊上過來的.

所以之後往下 DFS 時，需加入條件，即 u 的 node weight 必須 ≤ (u, v)

current neighbor.

otherwise, 放棄該 v. 改看其它 u 的 neighbors.

此外，在過程中，還會記錄每條 path 的 weight 和，並 keep min value.

keep path 的.

e.g. (A → B → C → D)

Correctness and time:

Time: DFS 的 time 於 7. 分析為 O(|V|+|E|)