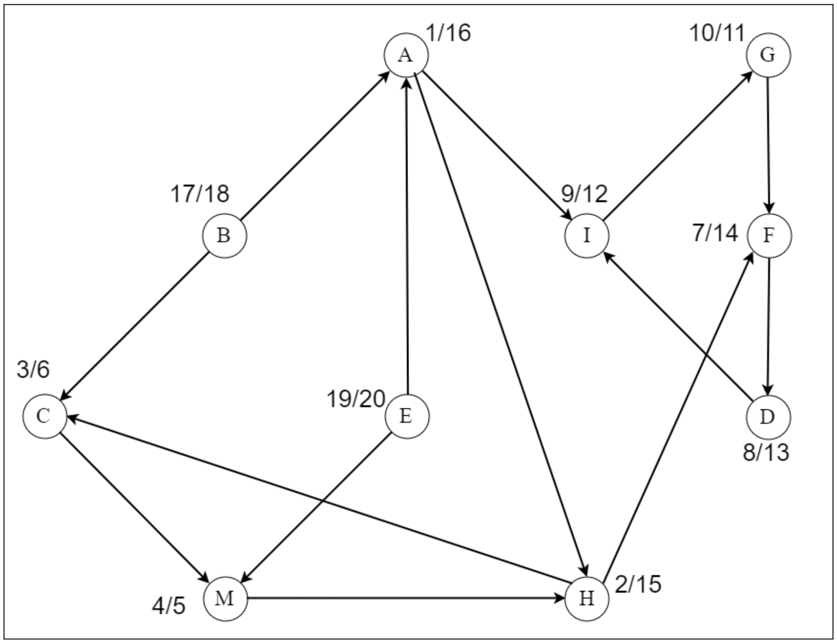
Solution - 9

Question 1.

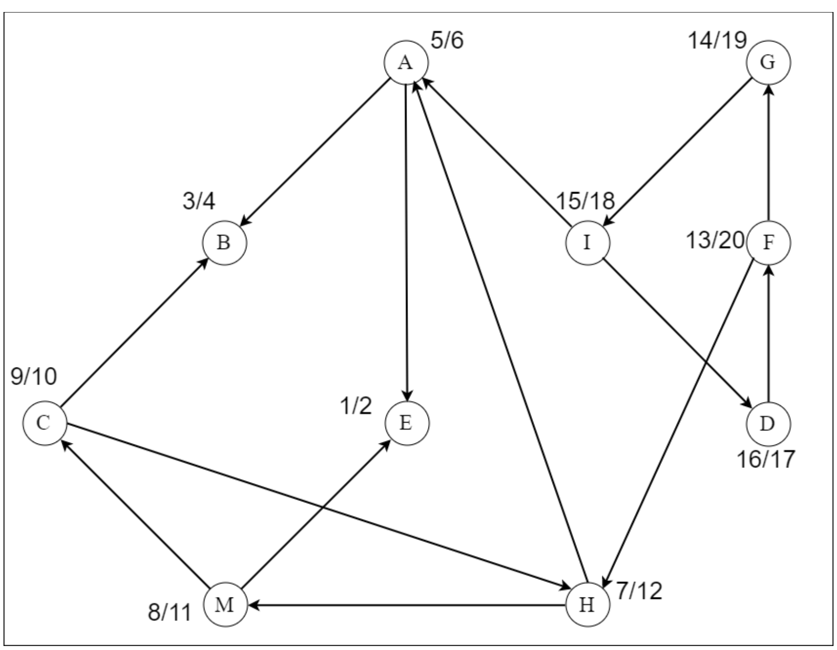
As seen in the lecture slides, the SCC algorithm has 4 main steps.

Step 1: Call DFS(G) to compute the finishing times u.f for all u.

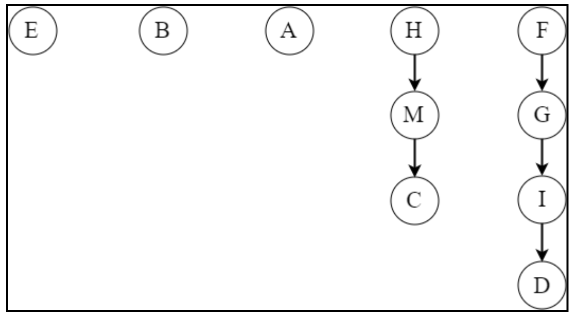


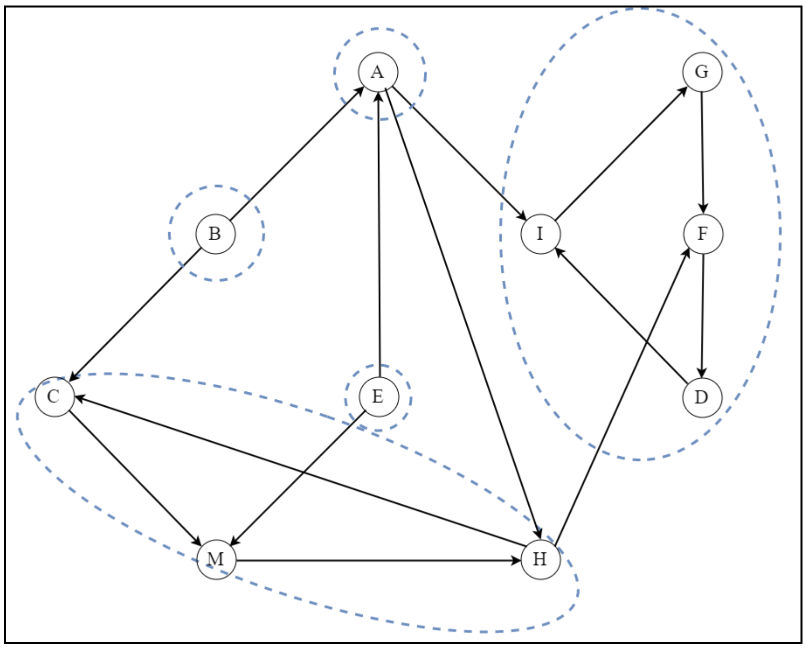
Step 2: compute G^T

Step 3: call DFS(G^T), but in the main loop, consider the main loop consider the vertices in order of decreasing u.f (as computed in the first DFS)



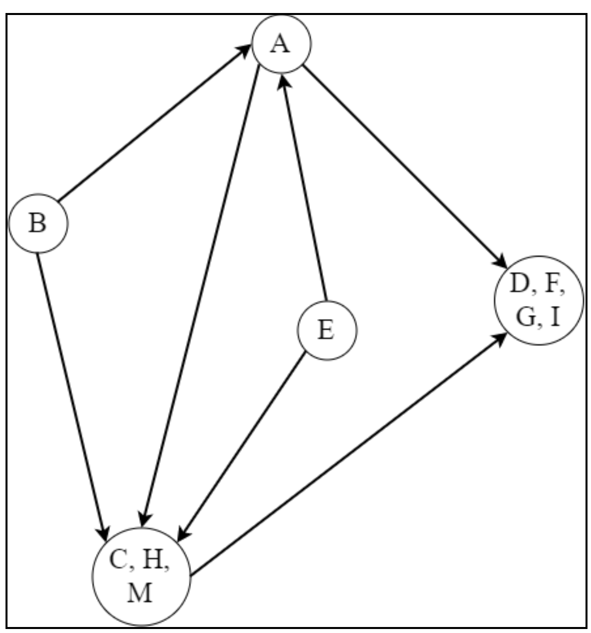
Step 4: output the vertices in each tree of the depth first forest formed in second DFS as a separate SCC.





Order in which the SCC are found: E, B, A, HMC, FGID

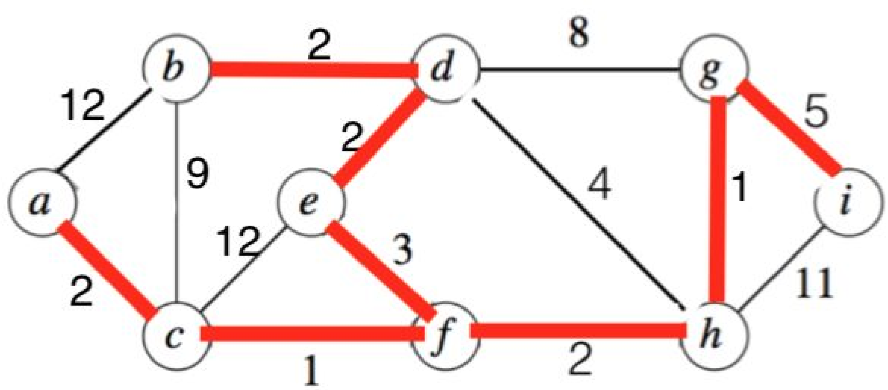
GSCC Graph (note: the above figure is **not** a GSCC. GSCC are DAGs, like below fig.):



Question 2.

The order of which edges are added:

Edge	Weight
cf	1
gh	1
ac	2
bd	2
de	2
fh	2
ef	3
gi	5



Question 3.

You should get O(1+VlogV+ElogE+ElogV+VlogV) before your final answer

KRUSKAL(G,w)

A = ∅

for each vertex $v \in G.V(v)$

MAKE-SET (v)

sort the edges of G.E into nondecreasing order by weight w

for each (u,v) taken from the sorted list

if FIND-SET(u) ≠ FIND-SET(v)

A = A ∪ {(u,v)}

UNION(u, v)

return A

Analysis:

Initialize X: O(1)

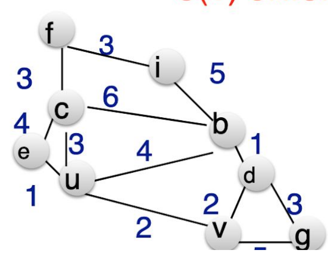
First **for** loop: |V| MAKE-SET

Sort E: O(E lg E)

Second **for** loop:

O(E) FIND-SET and

O(V) UNION



Running time O(E lg E + E lg* V)

Running time O(E lg V)

Question 4.

No, The algorithm mentioned does not always produce correct results. It is important for any connected component algorithm that the second dfs starts in a sink component, This algorithm does not provide that guarentee.

Counter-example: a⟷ b → c

In this case if the first dfs starting at b goes to c before a, the finishing time order would be c,a,b and the algorithm would produce correct results of (a,b) and (c)

However, if the first dfs goes to a before c and the finishing order is a,c,b then the second dfs would produce a single connected component (a,b,c) which is wrong

Question 5.

We can just iterate all the linked list of every ship. Then get the smallest distance between the ship and the closest polity.

The time complexity is O (n*k), where n is the number of ship and k is the number of polities.