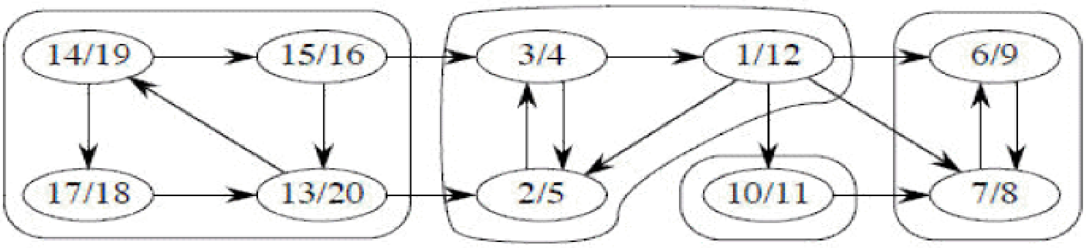
STRONGLY CONNECTED COMPONENTS

- A directed graph is strongly connected if there is a path between all pairs of vertices. A strongly connected component (**SCC**) of a directed graph is a maximal strongly connected subgraph.
- Decomposing a directed graph into its strongly connected components is a classic application of depth-first search.
- Given digraph or directed graph G = (V, E), a strongly connected component (SCC) of G is a maximal set of vertices C subset of V, such that for all *u*, *v* in C, both *u* ->*v* and *v*-> *u*; that is, both *u* and *v* are reachable from each other. In other words, two vertices of directed graph are in the same component if and only if they are reachable from each other.



C_1 C_2 C_3 C_4

- The above directed graph has 4 strongly connected components: C_1, C_2, C_3 and C_4.
- If G has an edge from some vertex in C_i to some vertex in C_j where $i \neq j$, then one can reach any vertex in C_j from any vertex in C_i but not return.
- In the example, one can reach any vertex in C_2 from any vertex in C_1 but cannot return to C_1 from C_2.

ALGORITHM

- A DFS(G) produces a forest of DFS-trees.
- Let C be any strongly connected component of G,
- let *v* be the first vertex on C discovered by the DFS and
- let T be the DFS-tree containing *v* when DFS-visit(*v*) is called all vertices in C are reachable from *v* along paths containing visible vertices;
- DFS-visit(*v*) will visit every vertex in C, add it to T as a descendant of *v*.

STRONGLY-CONNECTED-COMPONENTS (G)

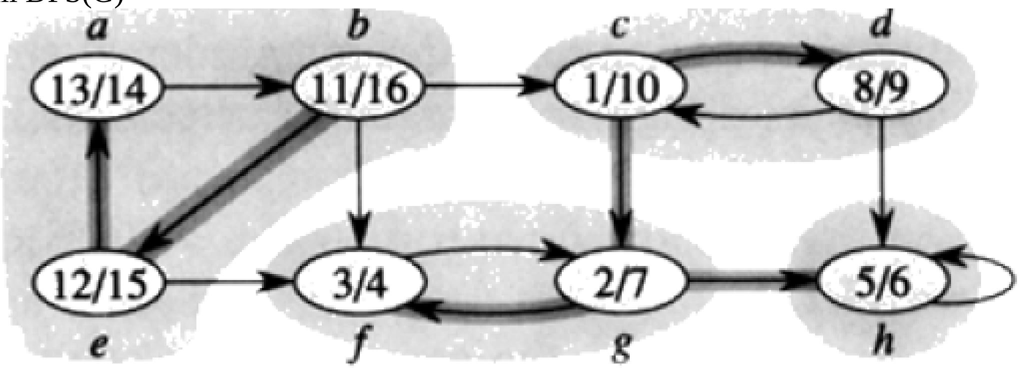
1. ***Call*** *DFS(G) to compute finishing times f[u] for all u.*
2. ***Compute*** G^T
3. ***Call*** *DFS(*G^T*), but in the main loop, consider vertices in order of decreasing f[u] (as computed in first DFS)*
4. ***Output*** *the vertices of each tree of the depth-first forest formed in second DFS as a separate SCC.*

Time complexity of strongly connected component - O(V+E) , since depth-first search takes O(V+E) time

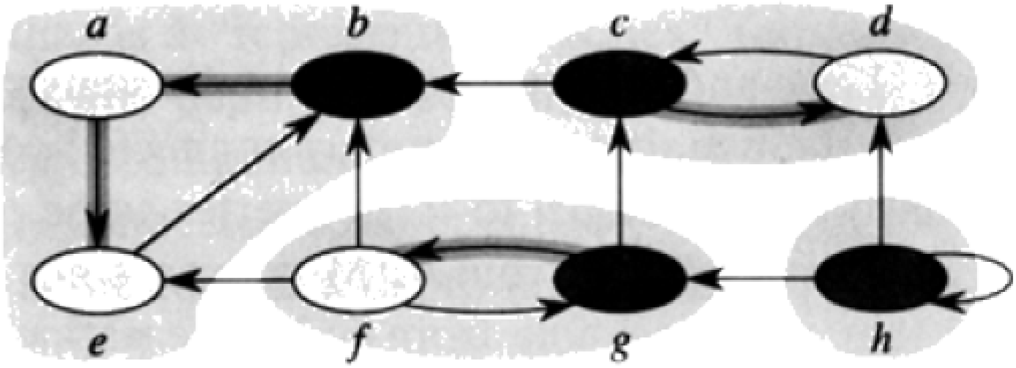
Example:

Consider a graph G = (V, E).

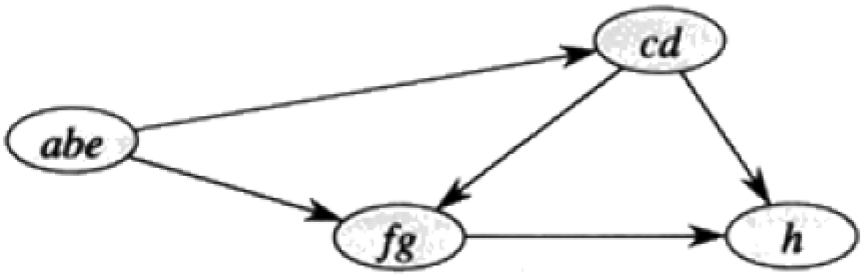
1. Call DFS(G)



2. Compute G^T



3. Call DFS(G^T) but this time consider the vertices in order to decreasing finish time.



4. Output the vertices of each tree in the DFS-forest as a separate strongly connected component.

{*a*, *b*, *e*}, {*c*, *d*}, {*f*, *g*}, and {*h*}

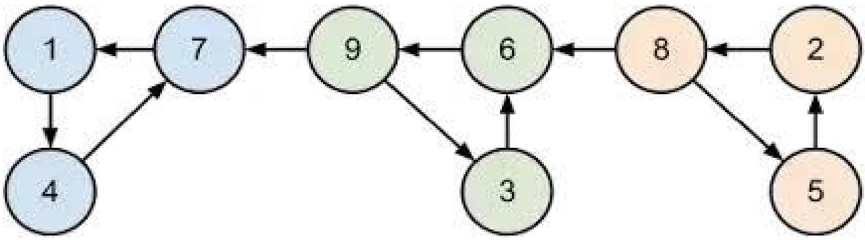
Qn.How can the number of strongly connected components of a graph change if a new edge is added?

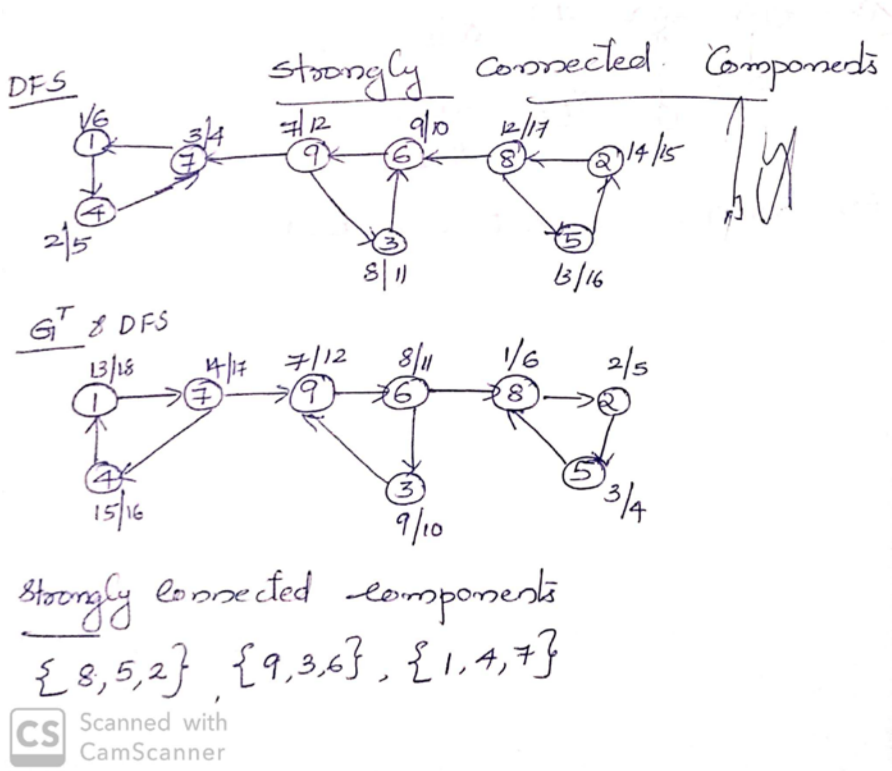
It can either stay the same or decrease. To see that it is possible to stay the same, just suppose you add some edge to a cycle. To see that it is possible to decrease, suppose that your original graph is on three vertices, and is just a path passing through all of them, and the edge added completes this path to a cycle. To see that it cannot increase, notice that adding an edge cannot remove any path that existed before.

So, if and are in the same connected component in the original graph, then there are a path from one to the other, in both directions. Adding an edge won't disturb these two paths, so we know that and will still be in the same in the graph after adding the edge. Since no SCC components can be split apart, this means that the number of them cannot increase since they form a partition of the set of vertices.

Question-answers

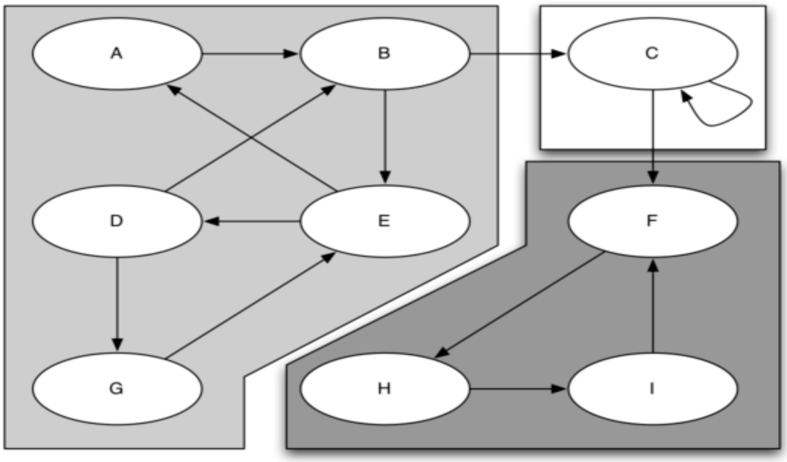
Find strongly connected components of the digraph using the algorithm showing each step [December 2019- 3 marks]

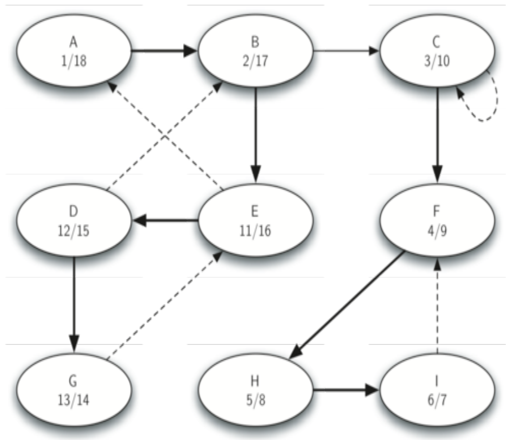




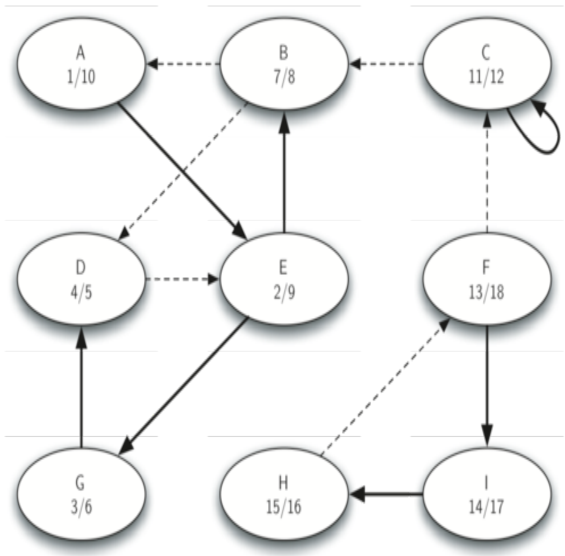
1-4-7,9-3-6 and 8-5-2

Example

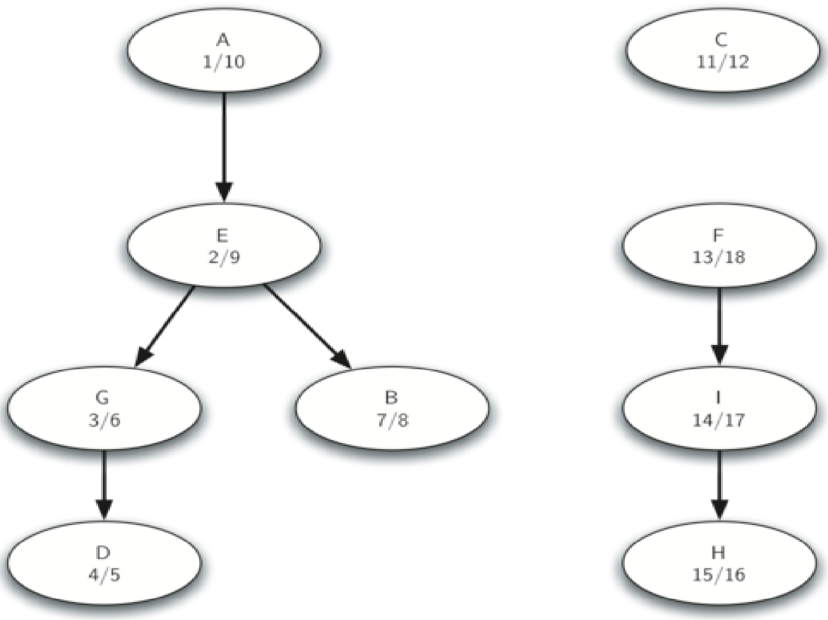




DFS(G)



DFS(G^T)



strongly connected components