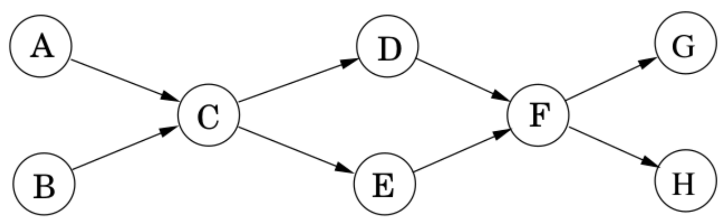
Homework 4 Solutions

October 23, 2019

3.3

Run the DFS-based topological ordering algorithm on the following graph. Whenever you have a choice of vertices to explore, always pick the one that is alphabetically first.



(a) Indicate the pre and post numbers of the nodes.

Solution

Node	Pre	Post
A	1	14
B	15	16
C	2	13
D	3	10
E	11	12
F	4	9
G	5	6
H	7	8

(b) What are the sources and sinks of the graph?

Solution

Source nodes: A, B
Sink nodes: G, H

(c) What topological ordering is found by the algorithm?

Solution

A topological ordering can be obtained by the decreasing ordering of post numbers from the DFS:
B, A, C, E, D, F, H, G

(d) How many topological orderings does this graph have?

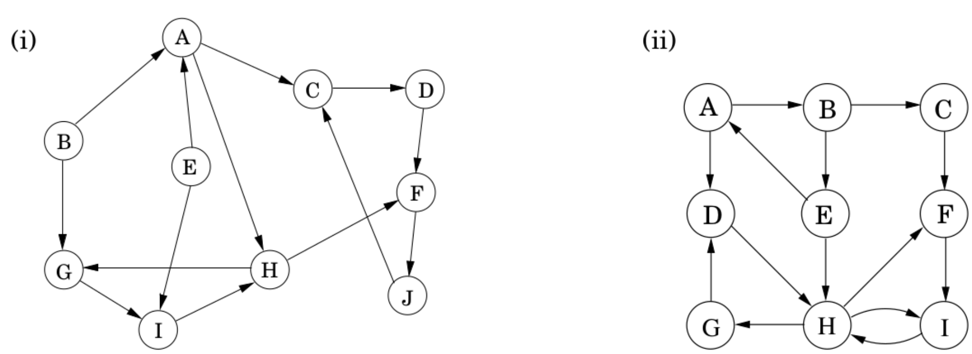
Solution

8

The topological ordering of this graph will be {A, B or B, A}, {C}, {D, E or E, D}, {F} {G, H or H, G}. So, overall we get $2 * 2 * 2 = 8$ topological orderings.

3.4

Run the strongly connected components algorithm on the following directed graphs G. When doing DFS on G^R: whenever there is a choice of vertices to explore, always pick the one that is alphabetically first.



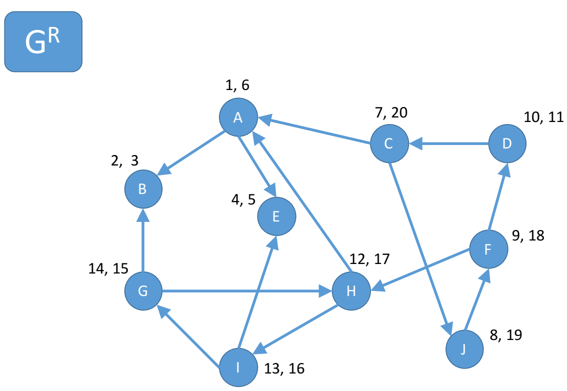
In each case answer the following questions.

(a) In what order are the strongly connected components (SCCs) found?
(b) Which are source SCCs and which are sink SCCs?
(c) Draw the "metagraph" (each meta-node is an SCC of G).
(d) What is the minimum number of edges you must add to this graph to make it strongly connected?

Solution on graph (i):

(a)

After running DFS on the edge reversed graph G^R, we get the following pre and post numbers for each node:



After ordering nodes by post number from high to low, we get:
C, J, F, H, I, G, D, A, E, B

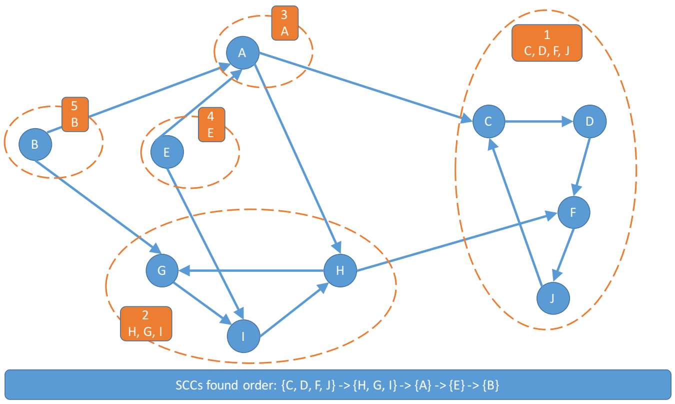
Run DFS upon the original graph, but choose nodes based of previous order:

Start from C, run DFS and go through C, D, F, J then back to C. So we get SSC {C, D, F, J}.

Then the remaining nodes H, I, G, A, E, B. Start from H, run DFS through H, G, I and go back to I. We get SSC {H, I, G}.

The remaining nodes were A, E, B. The same, start from A, get SSC {A}. Start from E, get SSC {E}. Start from B. We get SSC {B}.

Finally, we have:

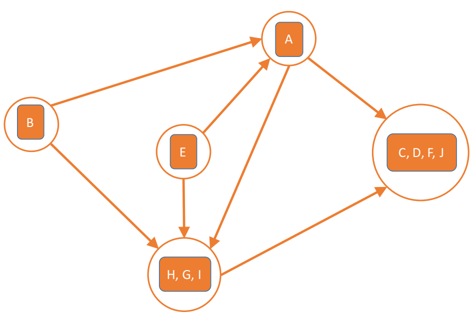


(b)

Source SCC: {B,} and {E}

Sink SCC: {C, D, F, J}

(c) The meta-graph:



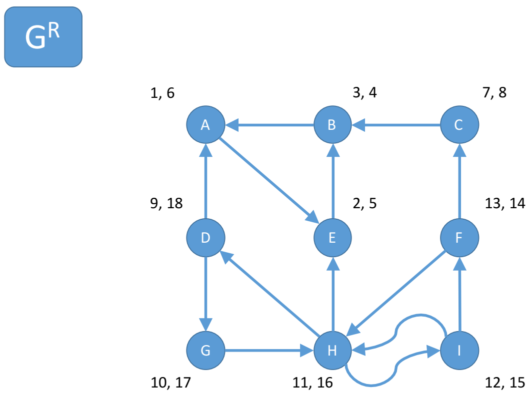
(d)

2 edges:
From SCC {HGI} to SCC {B}
From SCC {C, D, F, J} to SCC {E}

Solution on graph (ii):

(a)

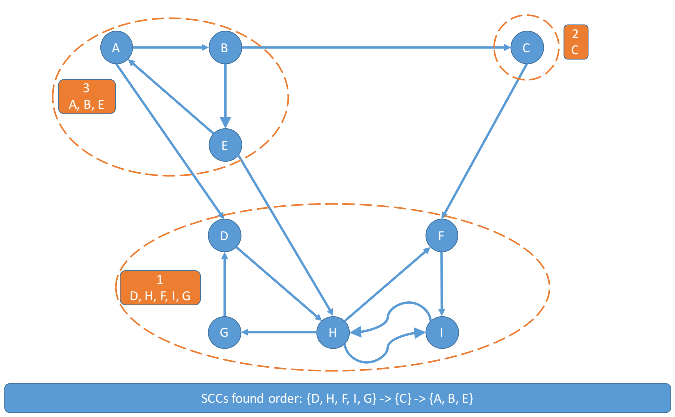
Run DFS unon G^R get:



Order nodes by its post numbers (High to low), get:
D, G, H, I, F, C, A, E, B
Run undirected connected components algorithm as what question (i) did:

Start from node D, run DFS through D, H, F, I, G and go back D, get SSC {D, H, F, I, G}.
Then, the remaining nodes are: C, A, E, B. Run DFS start from C and back to C, get SSC {C}.
Last, remains nodes are A, E, B. Started from A, run DFS through A, B, E, back to A. Then SSC {A, B, E}.

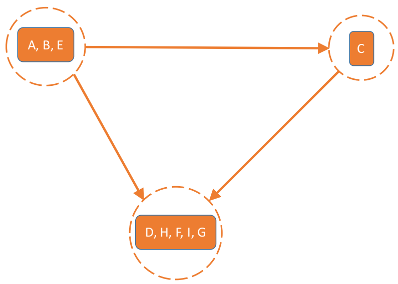
Finally, have:



(b)

Source SCC: {A, E, B}
Sink SCC: {D, H, F, I, G}

(c) The meta-graph:



(d)

1 edge:
From SCC {D, H, F, I, G} to SCC {A, E, B}

3.5

The *reverse* of a directed graph $G = (V, E)$ is another directed graph $G^R = (V, E^R)$ on the same vertex set, but with all edges reversed; that is, $E^R = (v, u) : (u, v) \in E$.
Give a linear-time algorithm for computing the reverse of a graph in adjacency list format.

Solution

Pseudocode:

Adj(G, v)
return the adjacency list of node v in graph G

Reverse(G)

Input: Graph $G = (V, E)$
Output: Graph $G^R = (V, E^R)$
for each node $v \in V$:
    Create an empty adjacency list for v in G^R: $\textsc{Adj}(G^R, v) = \{\}$
for each node $s \in V$:
    for each adjacent node $k \in \textsc{Adj}(G, s)$:
        insert s into the adjacency list $\textsc{Adj}(G^R, k)$
return G^R

Locating or inserting into an adjacency list can be done in constant time if the graph has a node index corresponding to its position, or the graph has a hash table.

Overall, this algorithm processes each node once, which takes $O(V)$. Then it visits all nodes on each from adjacency list once, which takes $O(E)$. Reversing each edge and locating its position in an adjacency list in G^R is $O(E)$.

Overall the running time is $O(V + E)$.