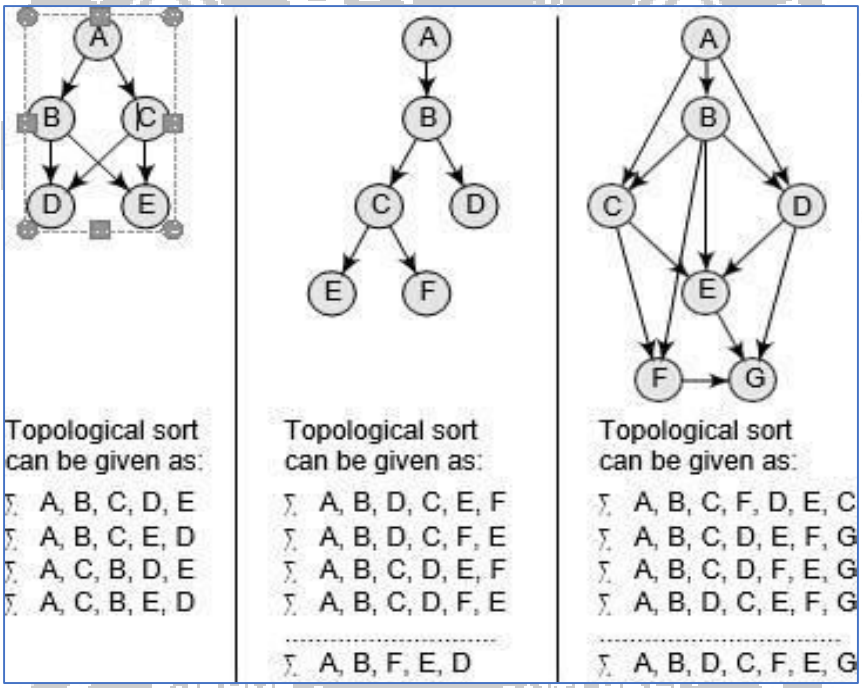
TOPOLOGICAL SORTING

- Topological sort of a directed acyclic graph (DAG) G is defined as a linear ordering of its nodes in which each node comes before all nodes to which it has outbound edges. Every DAG has one or more number of topological sorts.
- A topological sort of a DAG G is an ordering of the vertices of G such that if G contains an edge (u, v), then u appears before v in the ordering. Note that topological sort is possible only on directed acyclic graphs that do not have any cycles. For a DAG that contains cycles, no linear ordering of its vertices is possible.
- In simple words, a topological ordering of a DAG G is an ordering of its vertices such that any directed path in G traverses the vertices in increasing order.



Topological Sorting

Algorithm

The two main steps involved in the topological sort algorithm include:

- Selecting a node with zero in-degree
- Deleting N from the graph along with its edges

Step 1: Find the in-degree INDEG(N) of every node in the graph

Step 2: Enqueue all the nodes with a zero in-degree

Step 3: Repeat Steps 4 and 5 until the QUEUE is empty

Step 4: Remove the front node N of the QUEUE by setting FRONT = FRONT + 1

Step 5: Repeat for each neighbour M of node N:

a) Delete the edge from N to M by setting INDEG(M) = INDEG(M) – 1

b) IF INDEG(M) = , then Enqueue M, that is, add M to the rear of the queue

[END OF INNER LOOP]

[END OF LOOP]

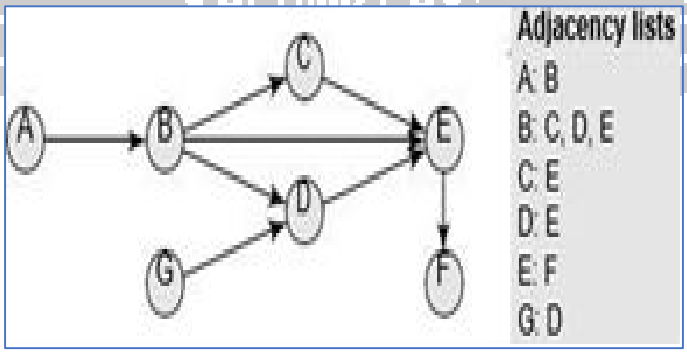
Step 6: Exit

We will use a QUEUE to hold the nodes with zero in-degree. The order in which the nodes will be deleted from the graph will depend on the sequence in which the nodes are inserted in the QUEUE. Then, we will use a variable INDEG, where INDEG(N) will represent the in-degree of node N.

Example

Consider a directed acyclic graph G given below. We use the algorithm given above to find a topological sort T of G.

The steps are given as below



Step 1: Find the in-degree INDEG(N) of every node in the graph

INDEG(A) = 0 INDEG(B) = 1 INDEG(C) = 1 INDEG(D) = 2

INDEG(E) = 3 INDEG(F) = 1 INDEG(G) = 0

Step 2: Enqueue all the nodes with a zero in-degree

FRONT = 1 REAR = 2 QUEUE = A, G

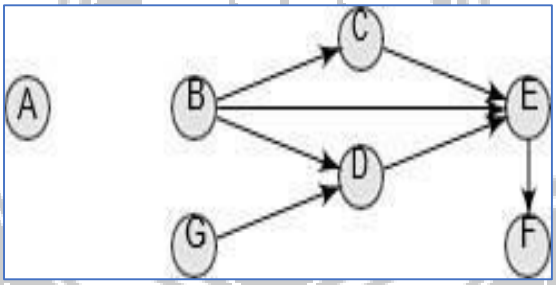
Step 3: Remove the front element A from the queue by setting FRONT = FRONT + 1, so

FRONT = 2 REAR = 2 QUEUE = A, G

Step 4: Set INDEG(B) = INDEG(B) – 1, since B is the neighbour of A. Note that INDEG(B) is 0, so add it on the queue. The queue now becomes

FRONT = 2 REAR = 3 QUEUE = A, G, B

Delete the edge from A to B. The graph now becomes as shown in the figure below



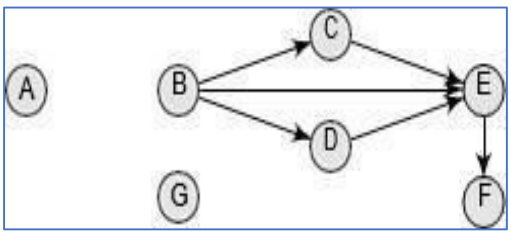
Step 5: Remove the front element B from the queue by setting FRONT = FRONT + 1, so

FRONT = 2 REAR = 3 QUEUE = A, G, B

Step 6: Set INDEG(D) = INDEG(D) – 1, since D is the neighbour of G. Now, INDEG(C) = 1

INDEG(D) = 1 INDEG(E) = 3 INDEG(F) = 1

Delete the edge from G to D. The graph now becomes as shown in the figure below



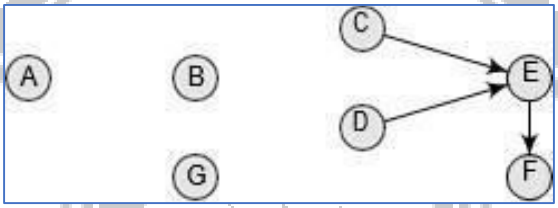
Step 7: Remove the front element B from the queue by setting FRONT = FRONT + 1, so

FRONT = 4 REAR = 3 QUEUE = A, G, B

Step 8: Set INDEG(C) = INDEG(C) – 1, INDEG(D) = INDEG(D) – 1, INDEG(E) = INDEG(E) – 1, since C, D, and E are the neighbours of B. Now, INDEG(C) = 0, INDEG(D) = 1 and INDEG(E) = 2

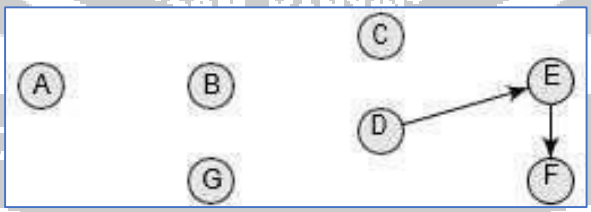
Step 9: Since the in-degree of node c and D is zero, add C and D at the rear of the queue. The queue can be given as below:

FRONT = 4 REAR = 5 QUEUE = A, G, B, C, D The graph now becomes as shown in the figure below



Step 10: Remove the front element C from the queue by setting FRONT = FRONT + 1, so FRONT = 5 REAR = 5 QUEUE = A, G, B, C, D

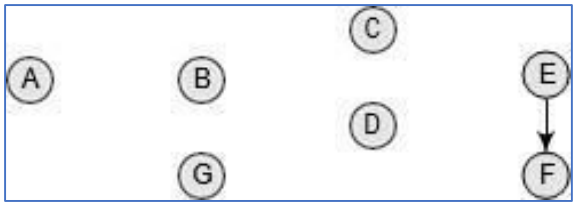
Step 11: Set INDEG(E) = INDEG(E) – 1, since E is the neighbour of C. Now, INDEG(E) = 1 The graph now becomes as shown in the figure below



Step 12: Remove the front element D from the queue by setting FRONT = FRONT + 1, so FRONT = 6 REAR = 5 QUEUE = A, B, G, C, D

Step 13: Set INDEG(E) = INDEG(E) – 1, since E is the neighbour of D. Now, INDEG(E) = 0, so add E to the queue. The queue now becomes. FRONT = 6 REAR = 6 QUEUE = A, G, B, C, D, E

Step 14: Delete the edge between D an E. The graph now becomes as shown in the figure below

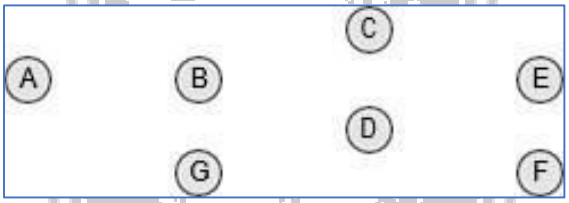


Step 15: Remove the front element D from the queue by setting FRONT = FRONT + 1, so FRONT = 7 REAR = 6 QUEUE = A, G, B, C, D, E

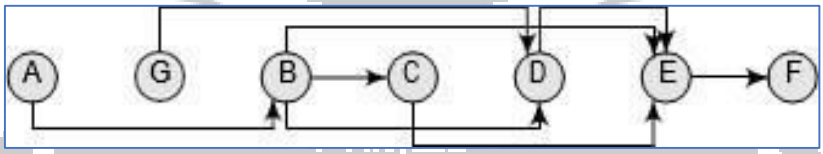
Step 16: Set INDEG(F) = INDEG(F) – 1, since F is the neighbour of E. Now INDEG(F) = 0, so add F to the queue. The queue now becomes,

FRONT = 7 REAR = 7 QUEUE = A, G, B, C, D, E, F

Step 17: Delete the edge between E and F. The graph now becomes as shown in the figure below



There are no more edges in the graph and all the nodes have been added to the queue, so the topological sort T of G can be given as: A, G, B, C, D, E, F. When we arrange these nodes in a sequence, we find that if there is an edge from u to v, then u appears before v.



Topological Sort of G