Algorithm Topological Sort using Kahn's algorithm [**1**]

Ensure: The vertices are indexed from 1 to V
Require: The Graph is a Directed Acyclic Graph

```
1: function Topological_Sort(Adjacency_List, Vertex_Set)
2:     sorted ← Empty List that will contain the Topological Ordering
3:     queue ← Empty queue to store elements with 0 in-degree
4:     V ← Vertex_Set.size

5:     for node = 1 to V do
6:         in_degree[node] ← 0                          ▷ Intialize in-degree

7:     for node = 1 to V do
8:         for each child ∈ adjacency[node] do
9:             in_degree[child] ← in_degree[child] + 1  ▷ Update in-degree

10:    for node = 1 to V do
11:        if in_degree[node] is 0 then
12:            queue.push(node)                         ▷ Collect the source vertices

13:    while queue is not empty, do
14:        current ← queue.front
15:        queue.pop
16:        sorted.append(current)
17:        for each child ∈ adjacency[current] do
18:            in_degree[child] ← in_degree[child] − 1  ▷ Delete the current vertex virtually
19:            if in_degree[child] is 0 then
20:                queue.push(child)                    ▷ Capture the new source vertex

21:    if sorted.size ≠ Vertex_Set.size then
22:        return Error                                 ▷ Graph has atleast one cycle
23:    else
24:        return sorted

25: end function
```

Example:

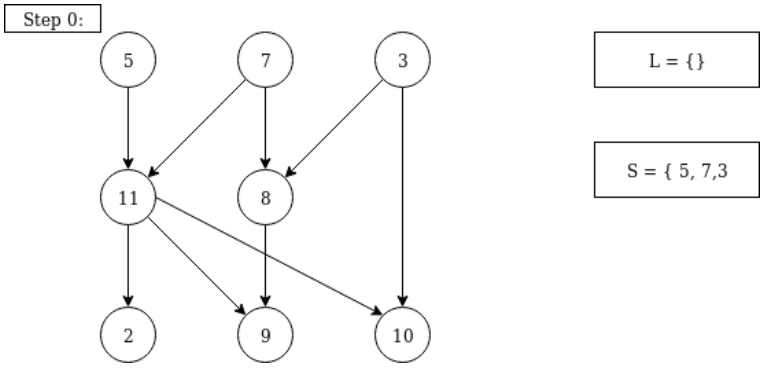

Figure 1: init state

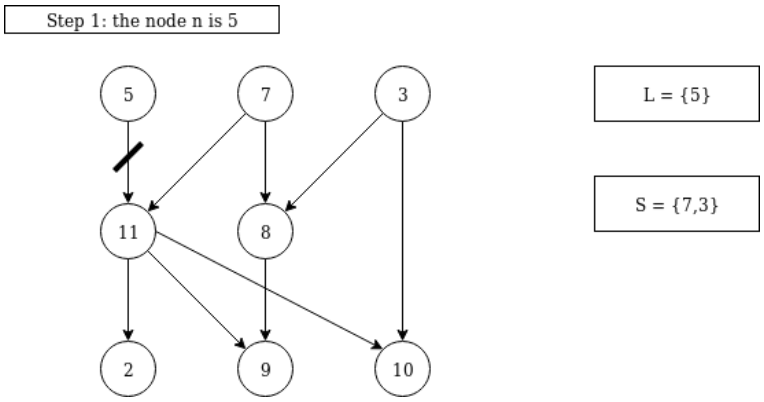

Figure 2: state 1

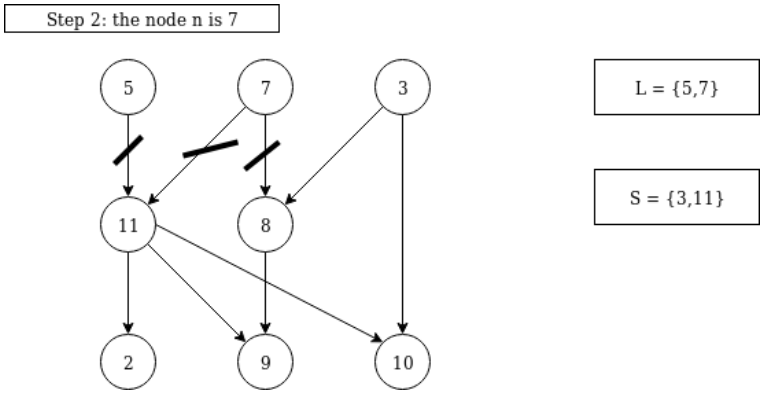

Figure 3: state 2

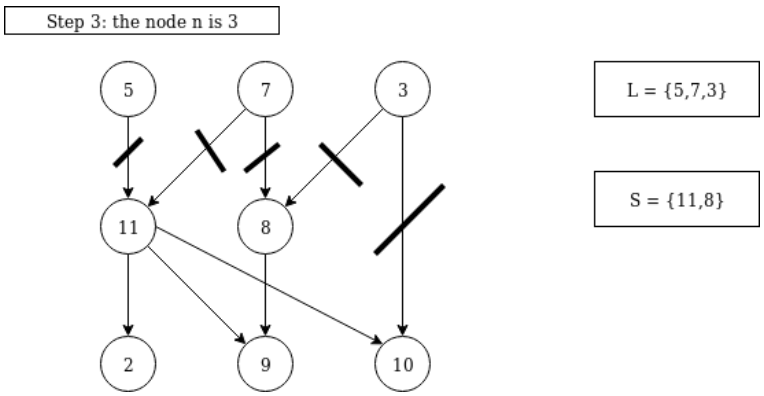

Figure 4: state 3

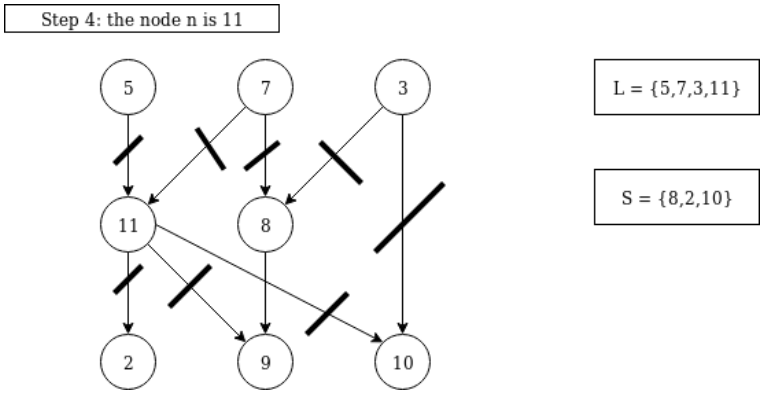

Figure 5: state 4

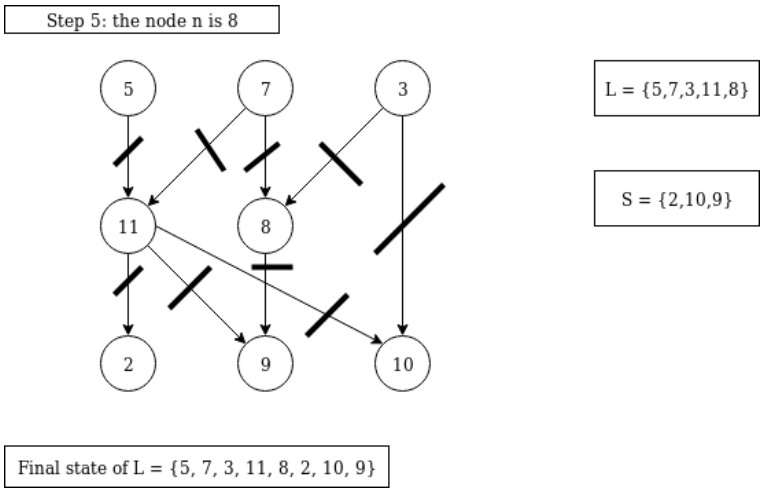

Figure 6: state 5

References

[1] Topological sort: Kahn's algorithm
Link