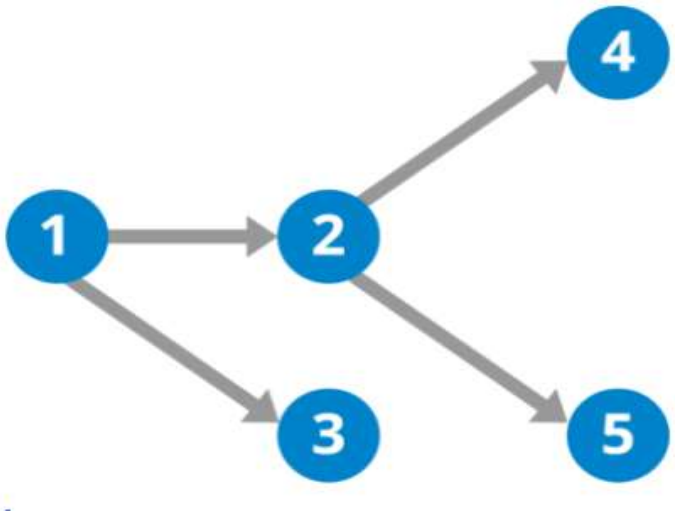
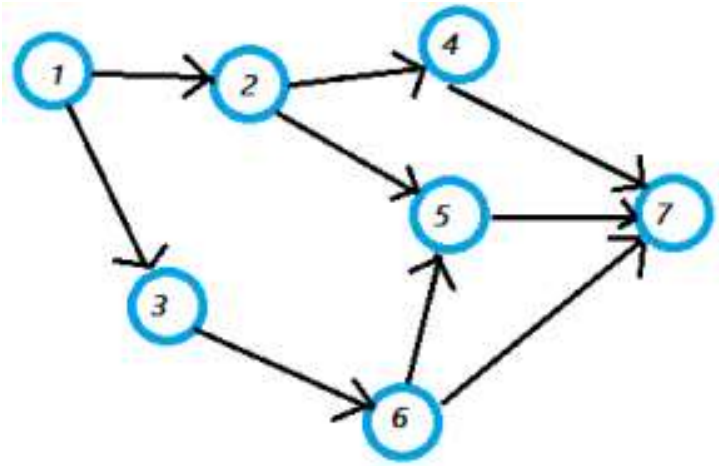
Topological Sorting

Directed Acyclic Graph

- DAG – Directed graph with no cycles.



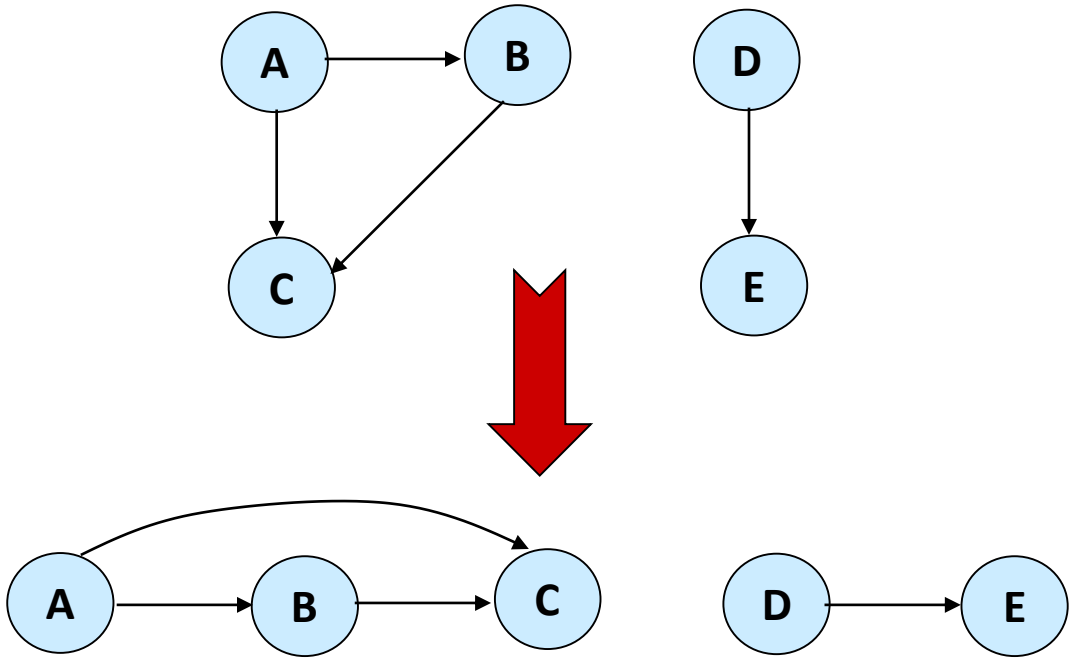


Topological Sorting

- Topological sorting for Directed Acyclic Graph (DAG) is a linear ordering of vertices such that for every directed edge u v, vertex u comes before v in the ordering.
- Topological Sorting for a graph is not possible if the graph is not a DAG.

Topological Sort

Want to “sort” a directed acyclic graph (DAG).



At the beginning, always try to pick nodes with no dependencies i.e. incoming degree of node is 0.

Topological Sort

- Performed on a DAG.
- Linear ordering of the vertices of G such that if $(u, v) \in E$, then u appears somewhere before v.

Topological-Sort (G)

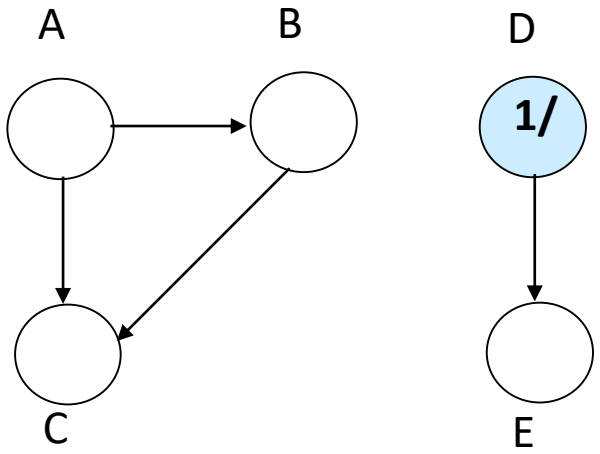
1. call DFS(G) to compute finishing times $f[v]$ for all $v \in V$
2. as each vertex is finished, insert it onto the front of a linked list
3. **return** the linked list of vertices

Time: $\Theta(V + E)$.

Algorithm

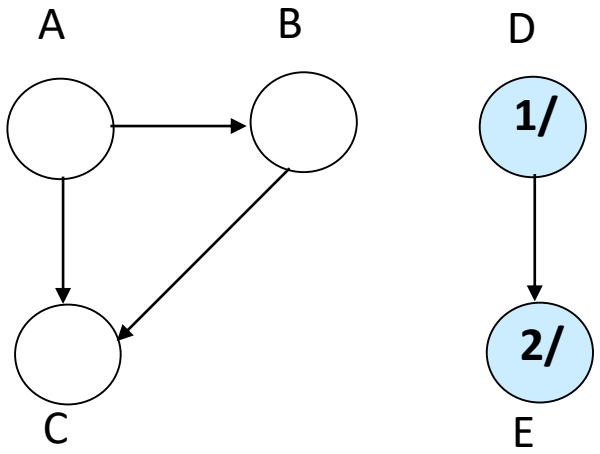
- Call dfs(g) for some graph g. The main reason we want to call depth first search is to compute the finish times for each of the vertices.
- Store the vertices in a list in decreasing order of finish time.
- Return the ordered list as the result of the topological sort.

Example



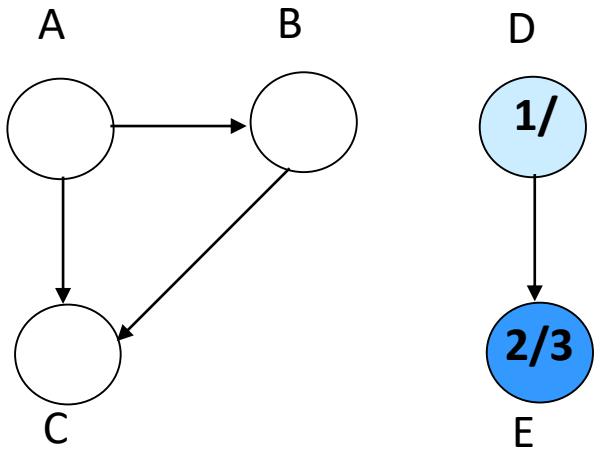
Linked List:

Example

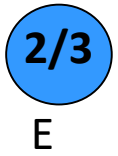


Linked List:

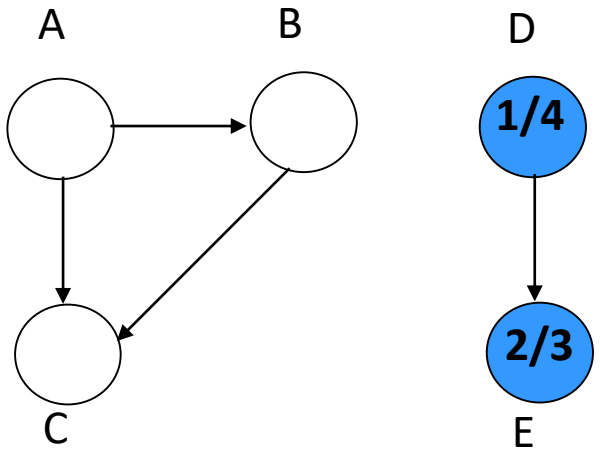
Example



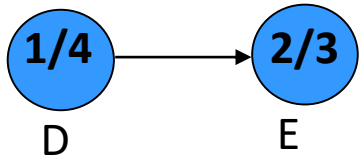
Linked List:



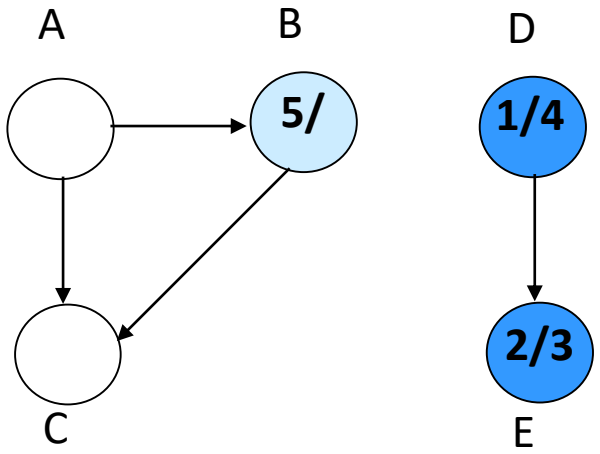
Example



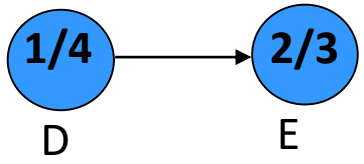
Linked List:



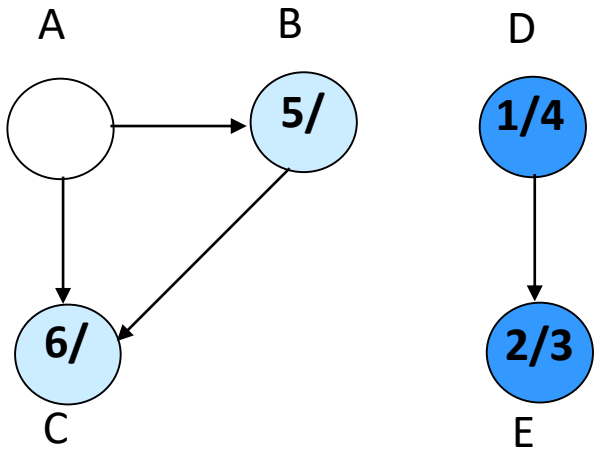
Example



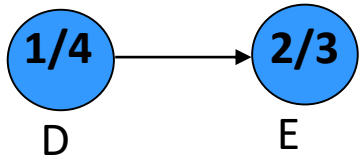
Linked List:



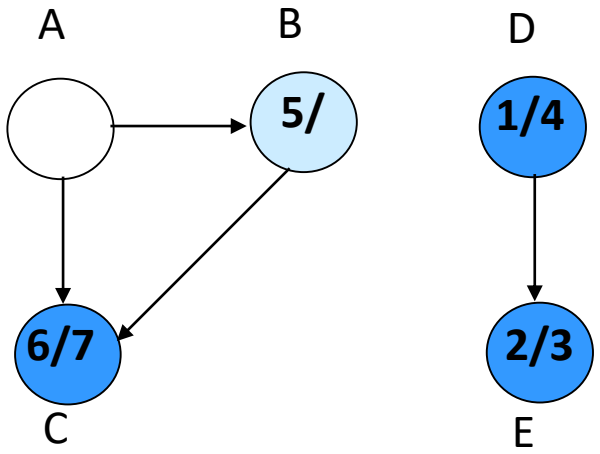
Example



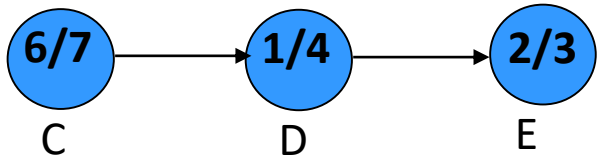
Linked List:



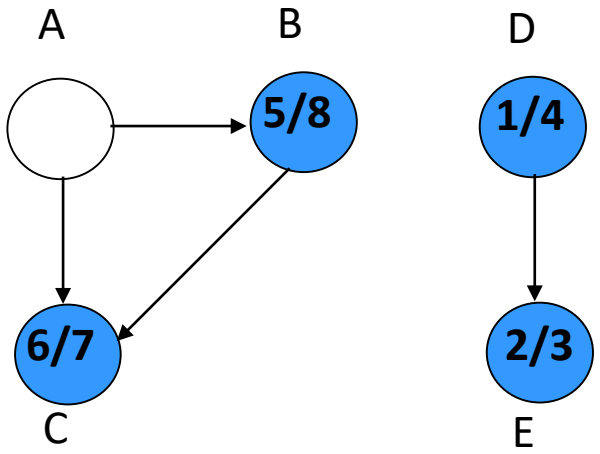
Example



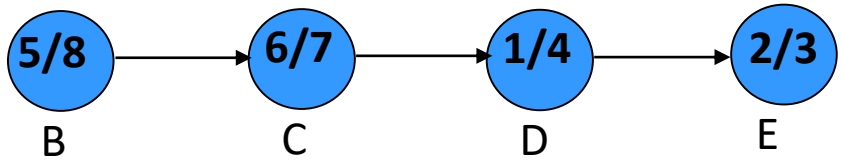
Linked List:



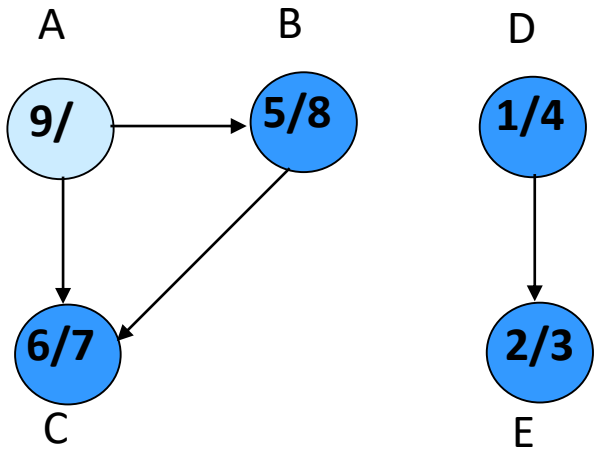
Example



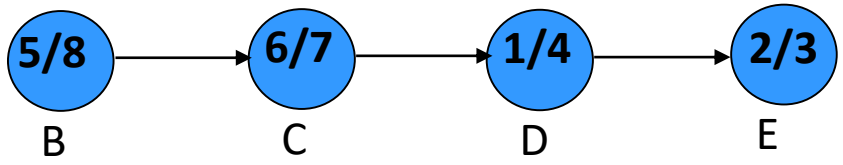
Linked List:



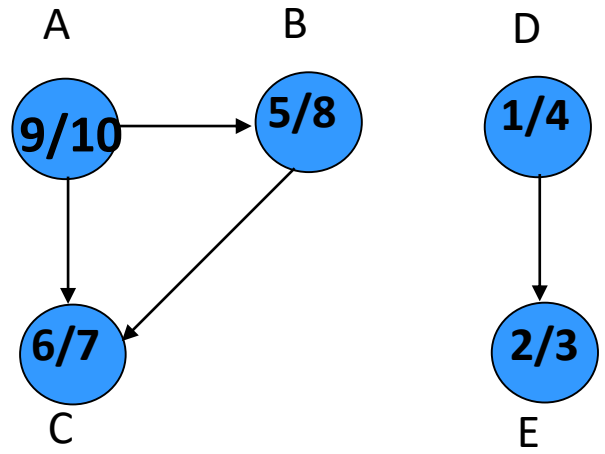
Example



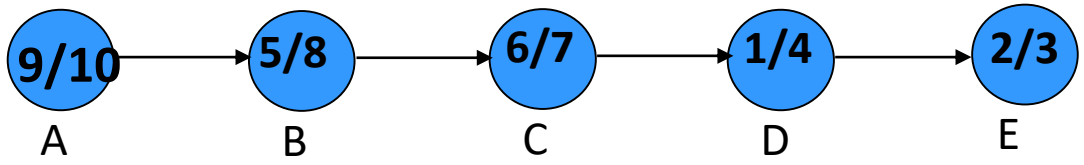
Linked List:



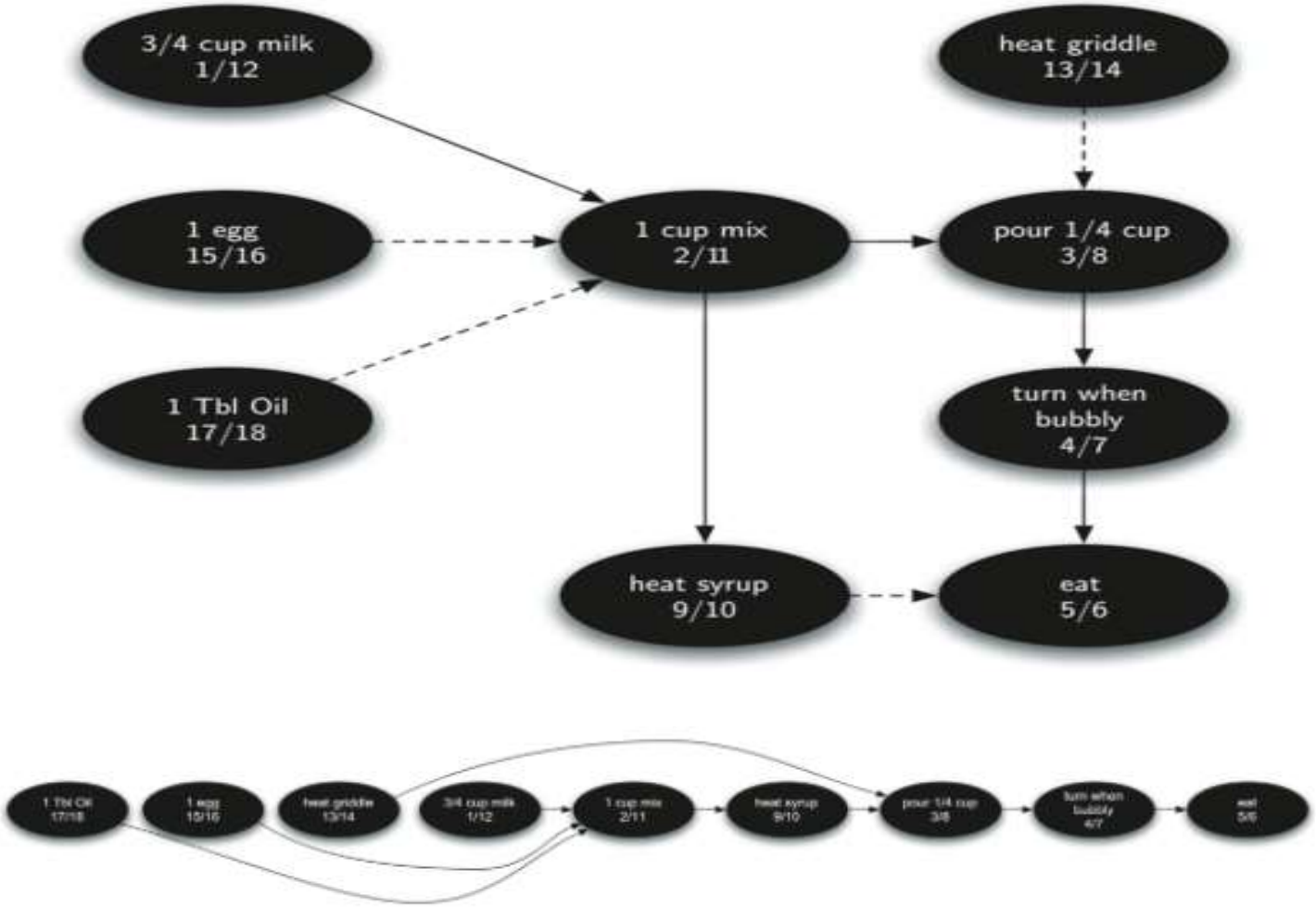
Example



Linked List:

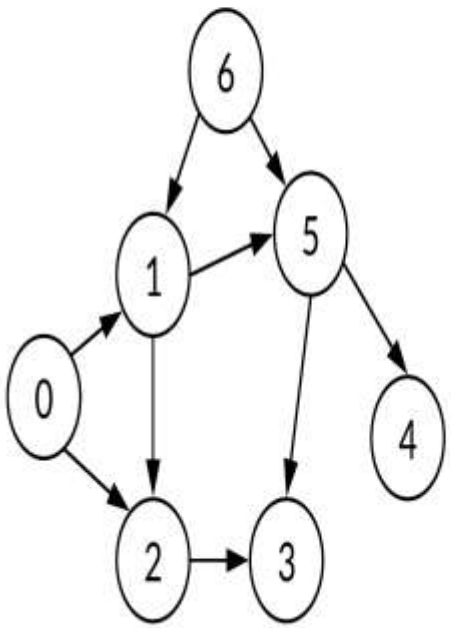


Example-2

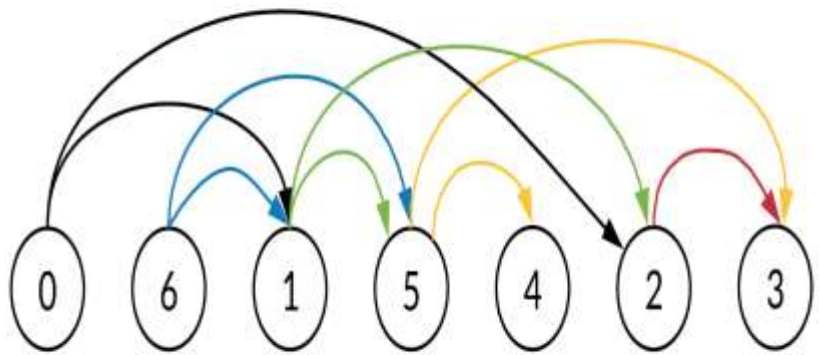

3/4 cup milk
1/12
heat griddle
13/14
1 egg
15/16
1 cup mix
2/11
pour 1/4 cup
3/8
1 Tbl Oil
17/18
turn when
bubbly
4/7
heat syrup
9/10
eat
5/6

Example-3

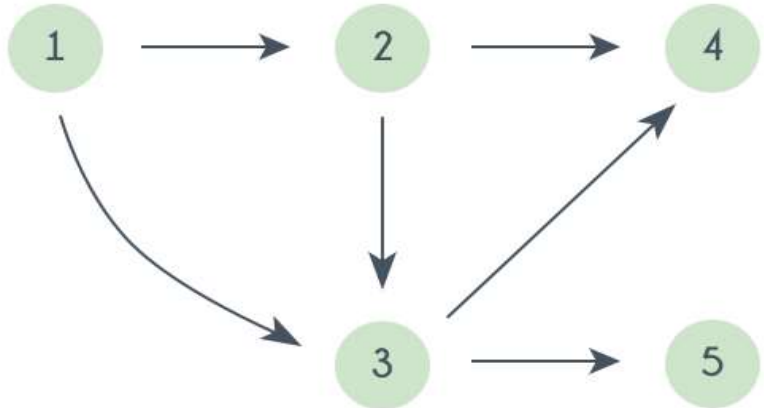
Unsorted graph

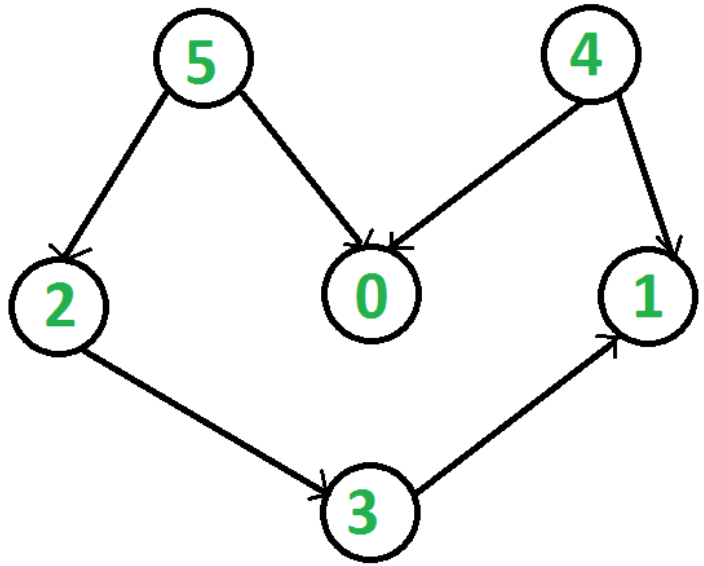


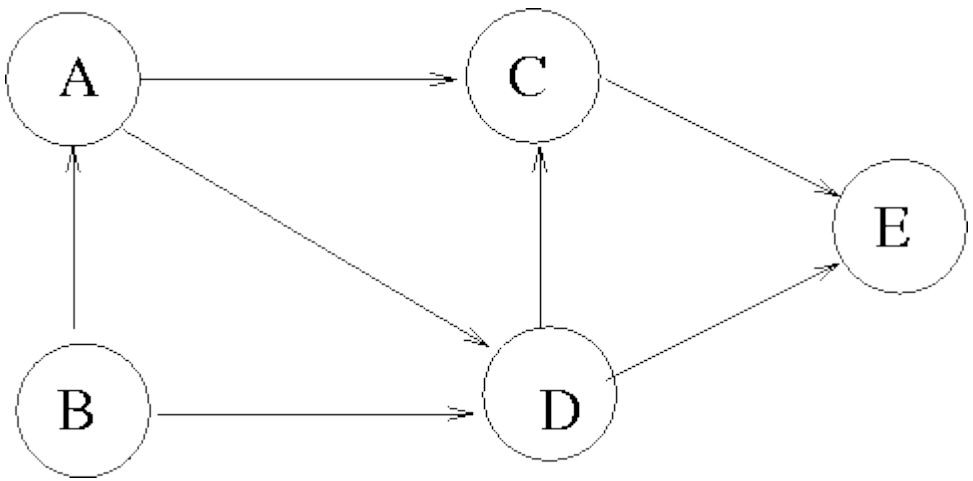
Topologically sorted graph



Exercises





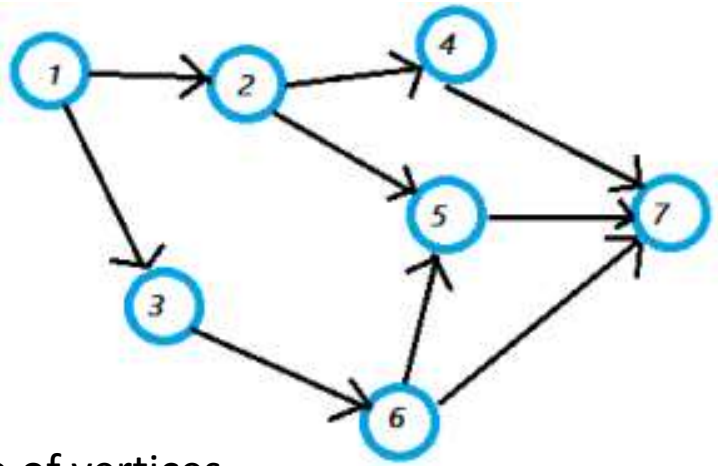


Kahn’s algorithm for topological sort

- This algorithm can find a topological ordering in O(V+E) time.
- Add all vertices with no dependencies into a queue (An array stores incoming degree of vertices).
- Choose the first vertex in queue. Remove all outgoing edges of the vertex and add it to topological sorting.
- Repeat the process until all vertices in the graph are visited or a cycle is detected.
- Note: topological orderings are not unique

Example

Topological Sorting Array



Incoming vertex degree of vertices

1	2	3	4	5	6	7
0	1	1	1	2	1	3

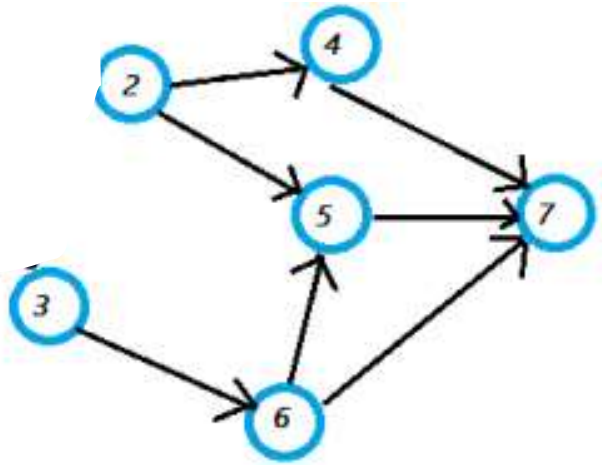
Queue

1						

Example

Topological Sorting Array

1						



Incoming vertex degree of vertices

1	2	3	4	5	6	7
0	0	0	1	2	1	3

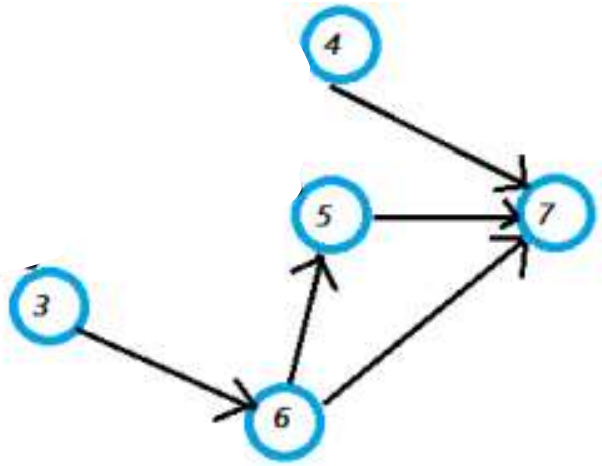
Queue (Incoming vertex degree=0)

2	3					

Example

Topological Sorting Array

1	2					



Incoming vertex degree of vertices

1	2	3	4	5	6	7
0	0	0	0	1	1	3

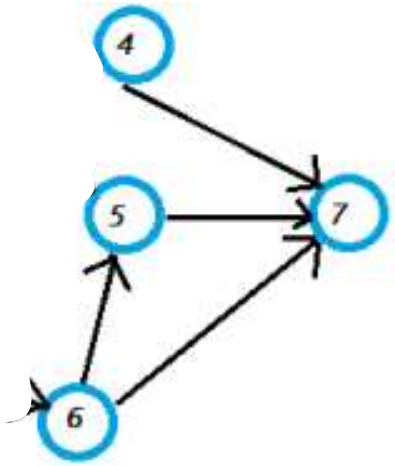
Queue (Incoming vertex degree=0)

3	4					

Example

Topological Sorting Array

1	2	3				



Incoming vertex degree of vertices

1	2	3	4	5	6	7
0	0	0	0	1	0	3

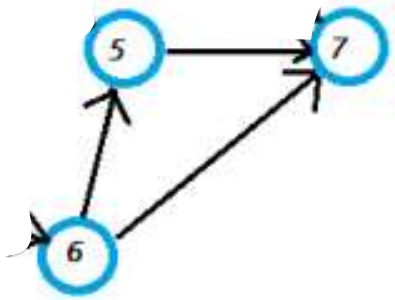
Queue (Incoming vertex degree=0)

4	6					

Example

Topological Sorting Array

1	2	3	4			



Incoming vertex degree of vertices

1	2	3	4	5	6	7
0	0	0	0	1	0	2

Queue (Incoming vertex degree=0)

6						

Example

Topological Sorting Array

1	2	3	4	6		

Incoming vertex degree of vertices

1	2	3	4	5	6	7
0	0	0	0	0	0	1

Queue (Incoming vertex degree=0)

5						

Example

Topological Sorting Array

1	2	3	4	6	5	

Incoming vertex degree of vertices

1	2	3	4	5	6	7
0	0	0	0	0	0	0

Queue (Incoming vertex degree=0)

7						

Example

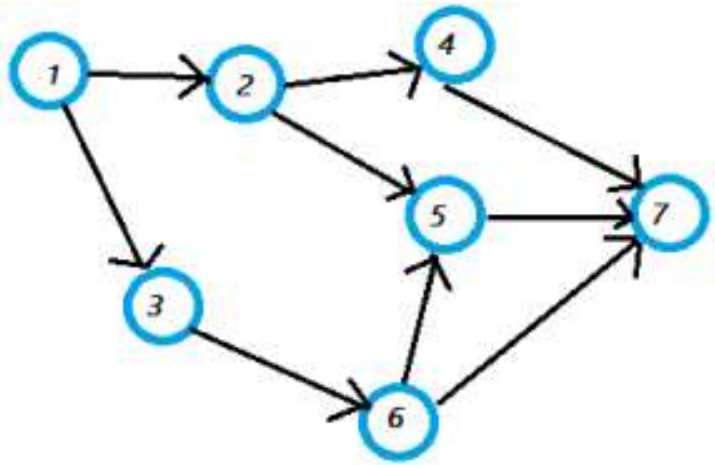
Topological Sorting Array

1	2	3	4	6	5	7

Incoming vertex degree of vertices

1	2	3	4	5	6	7
0	0	0	0	0	0	0

Queue (Incoming vertex degree=0)


1
2
4
5
7
3
6

Application of Topological Sort

- Topological Sorting is mainly used for scheduling jobs from the given dependencies among jobs.
- In computer science, applications of this type arise in Prerequisite courses, Program dependencies/Building systems, Event scheduling, instruction scheduling, ordering of formula cell evaluation when recomputing formula values in spreadsheets, logic synthesis, determining the order of compilation tasks to perform in make files, data serialization, and resolving symbol dependencies in linkers.
- A graph that contains a cycle (cyclic graph) cannot have valid sorting.
- All rooted trees have topological ordering since there are no cycles.