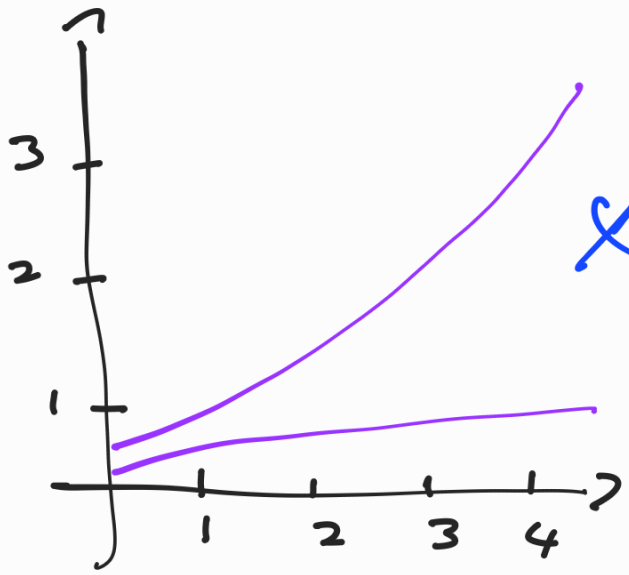
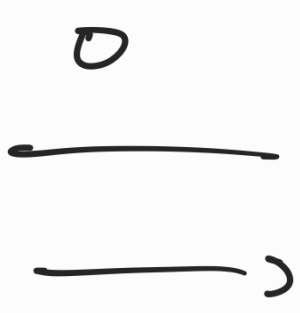
① what is graph ?



✗ not dealing with this graph

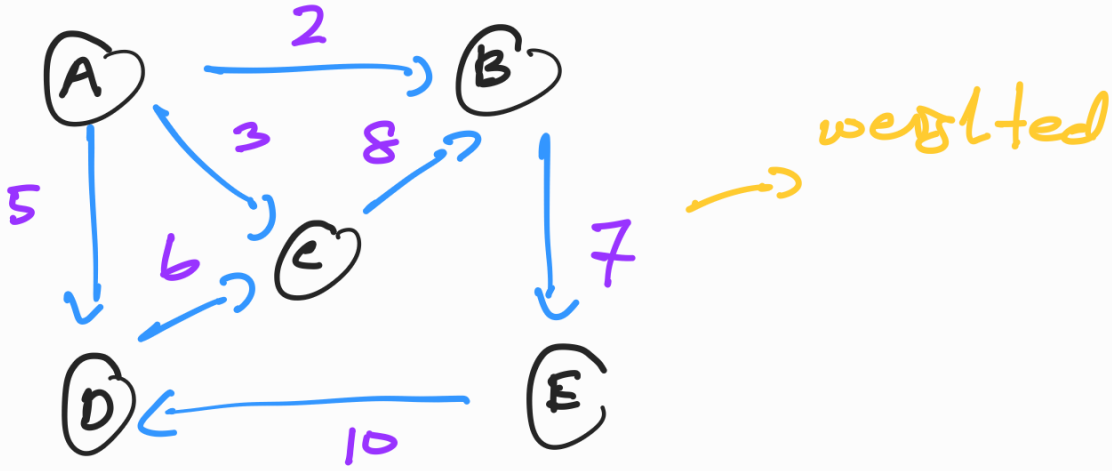
our graph → graph theory.



→ vertex

→ Edge

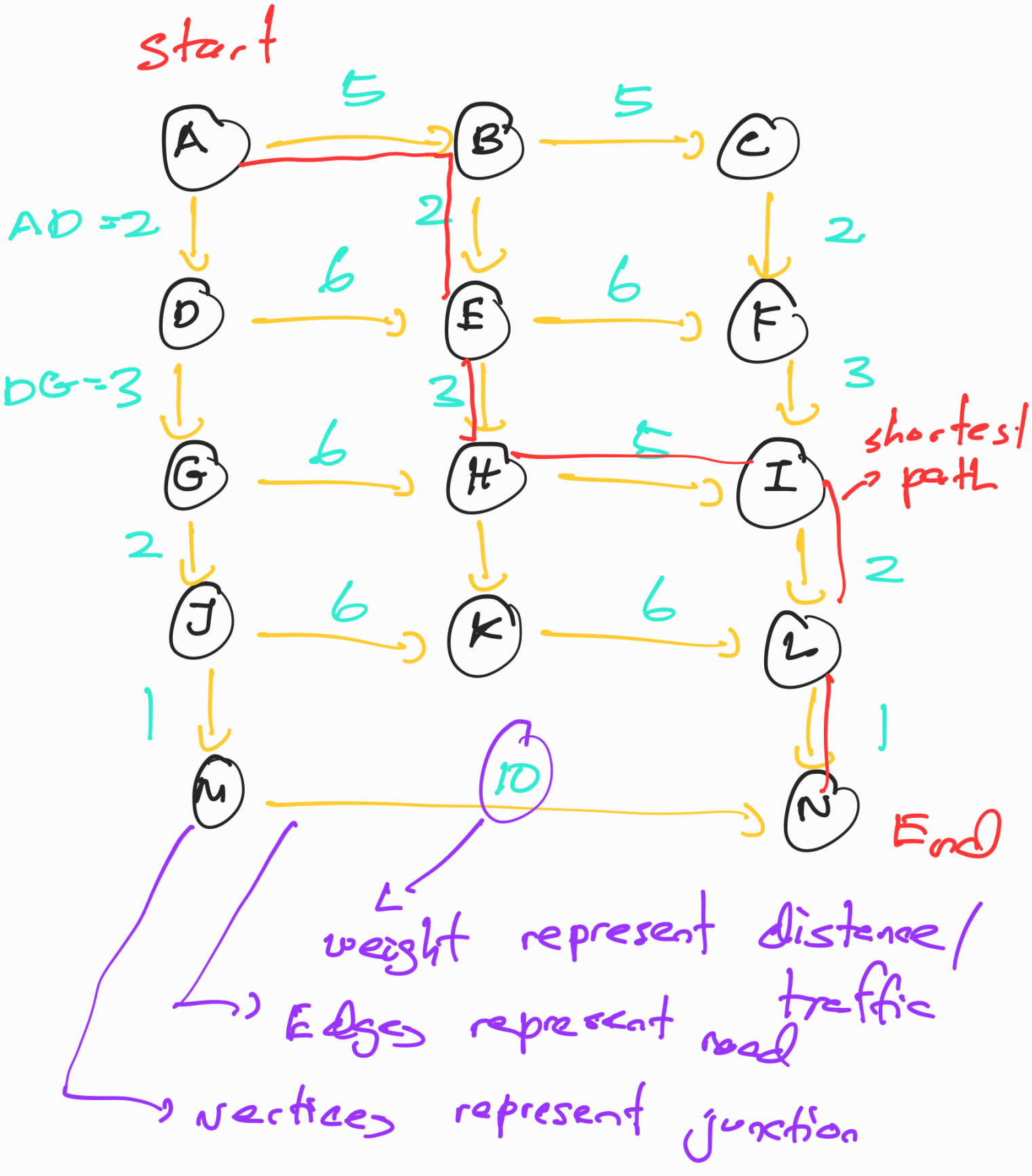
→ Edge with arrow / notation



→ weighted

(2) Graph in real applications?

Google Maps < path finding?
fastest route?



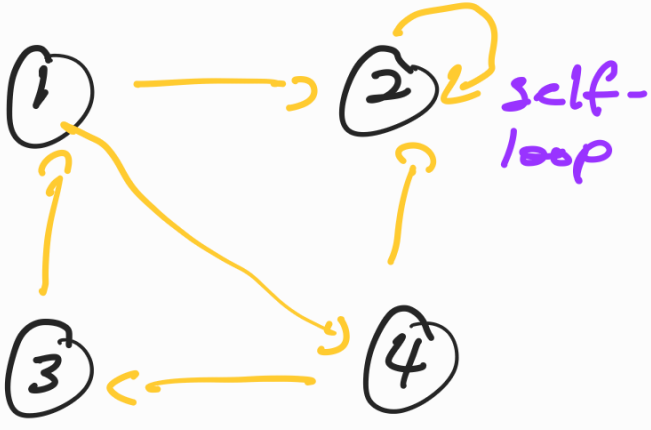
other applications?

- scheduling < Job scheduling
  - timetable
- sorting — web crawler
- computer network
- optimization
- etc. etc.

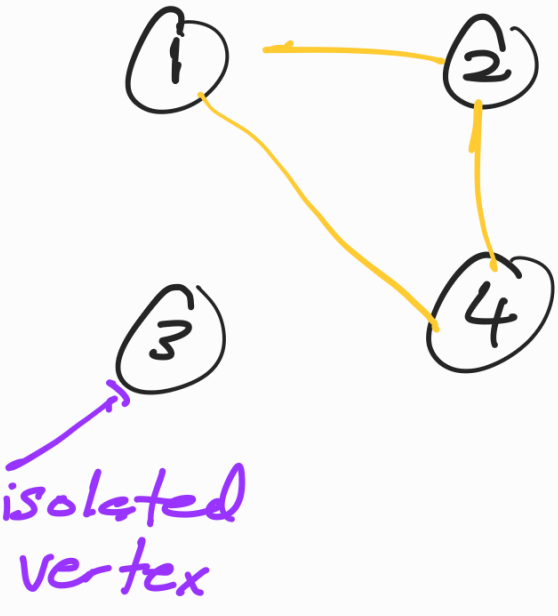
(3) Type of Graph ?

- directed vs. undirected
- completed vs. incomplete
- weighted vs. non-weighted
- self-loop

Directed



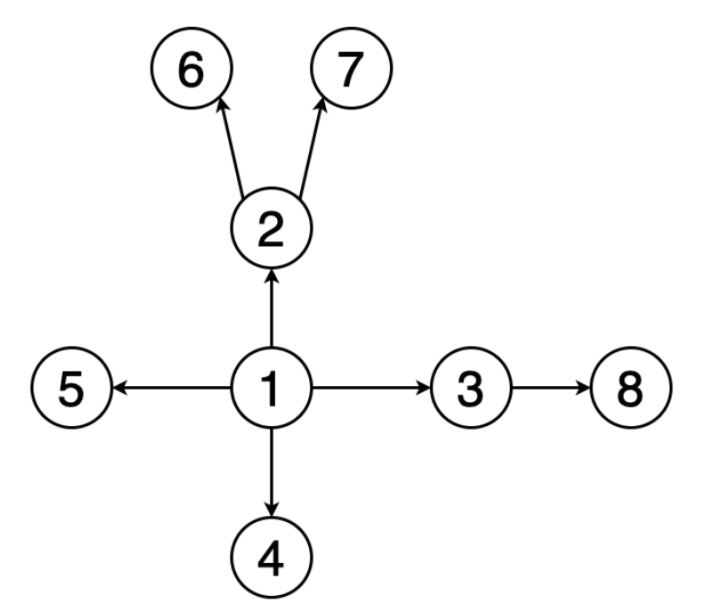
undirected



(4) Adjacent Matrix
(5) Adjacent List
} refer to sides

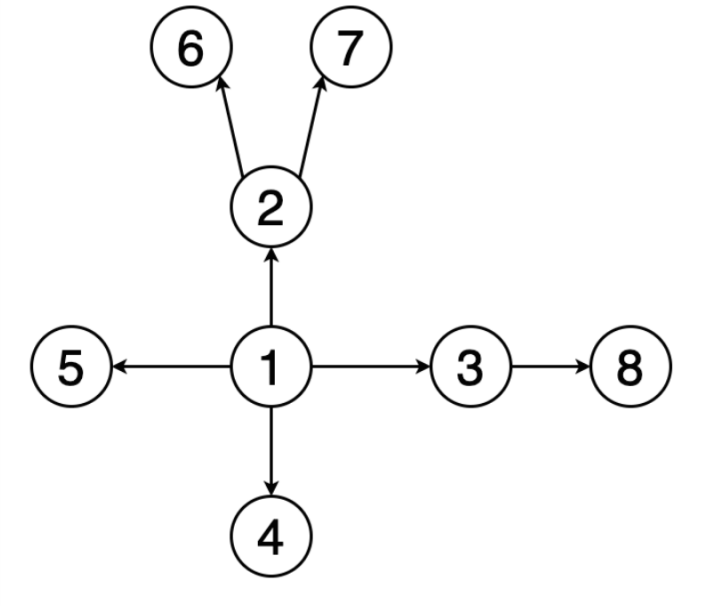
Brute Force (Exhaustive search)

- BFS
- DFS.



DFS

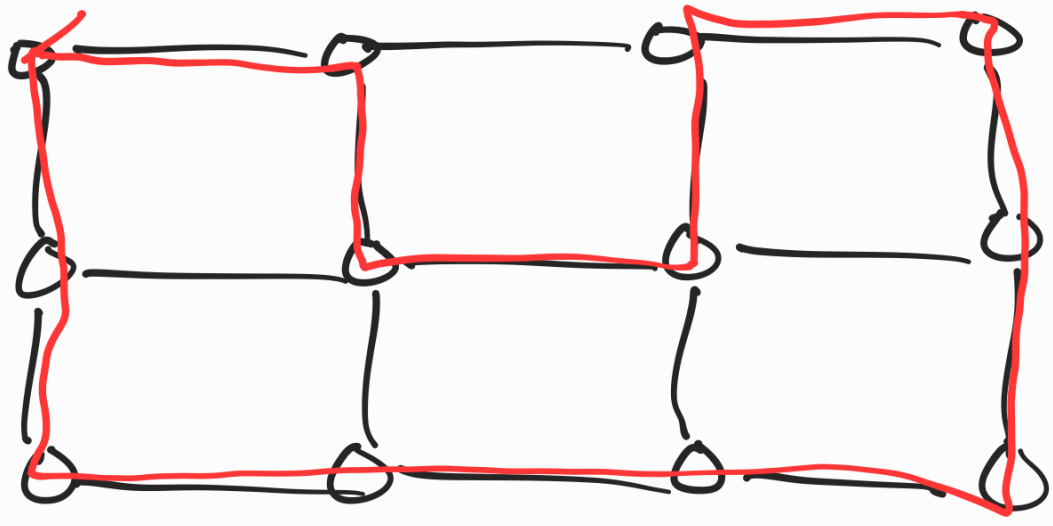
BFS

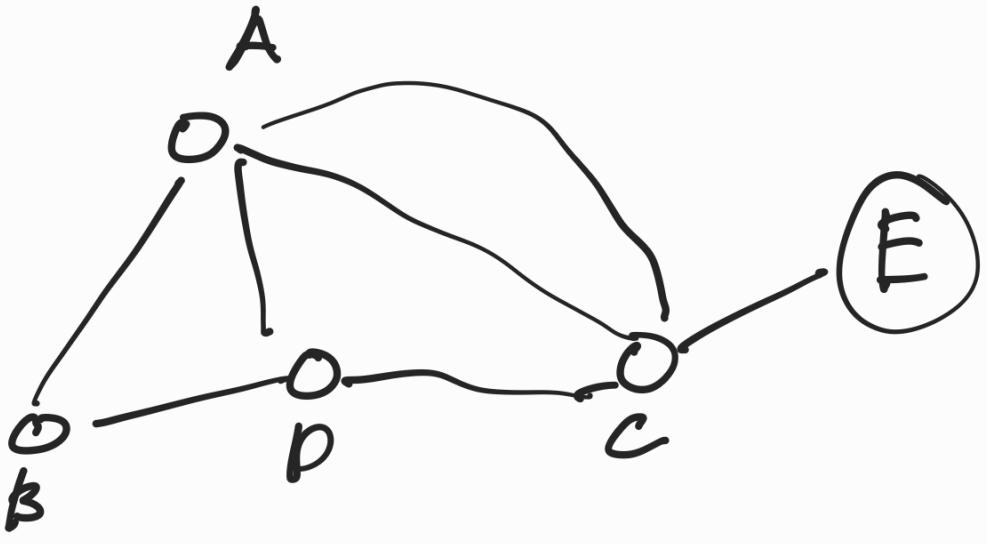


Hamilton circuit problem

circuit - return to starting point
visit every vertex once, no repeats

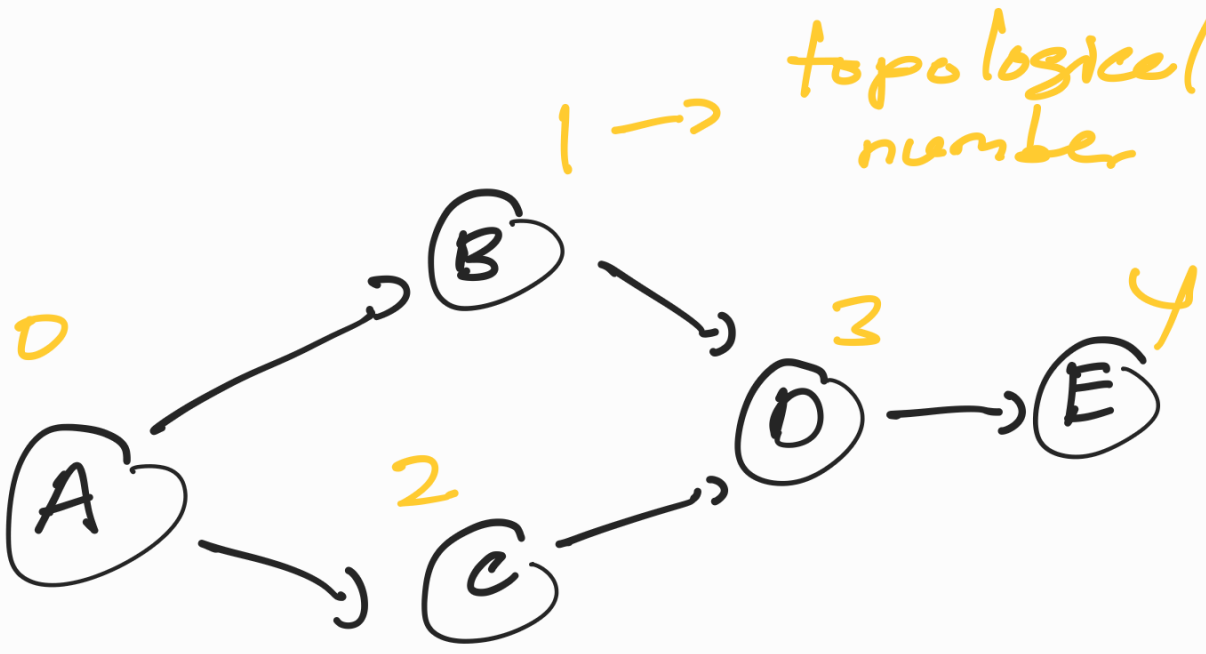
start & End





* No HC in this graph!

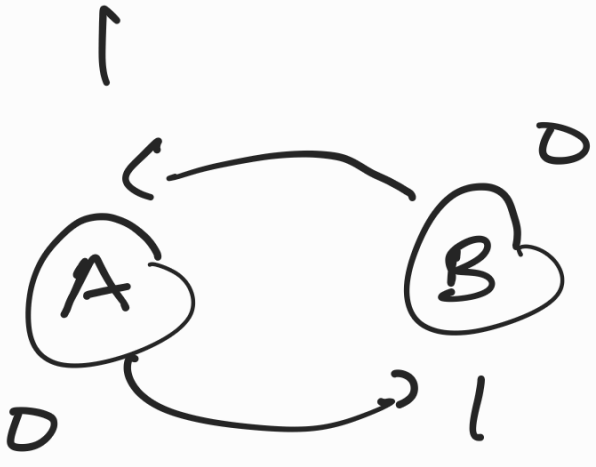
Topological sort problem



A, B, C, D, E

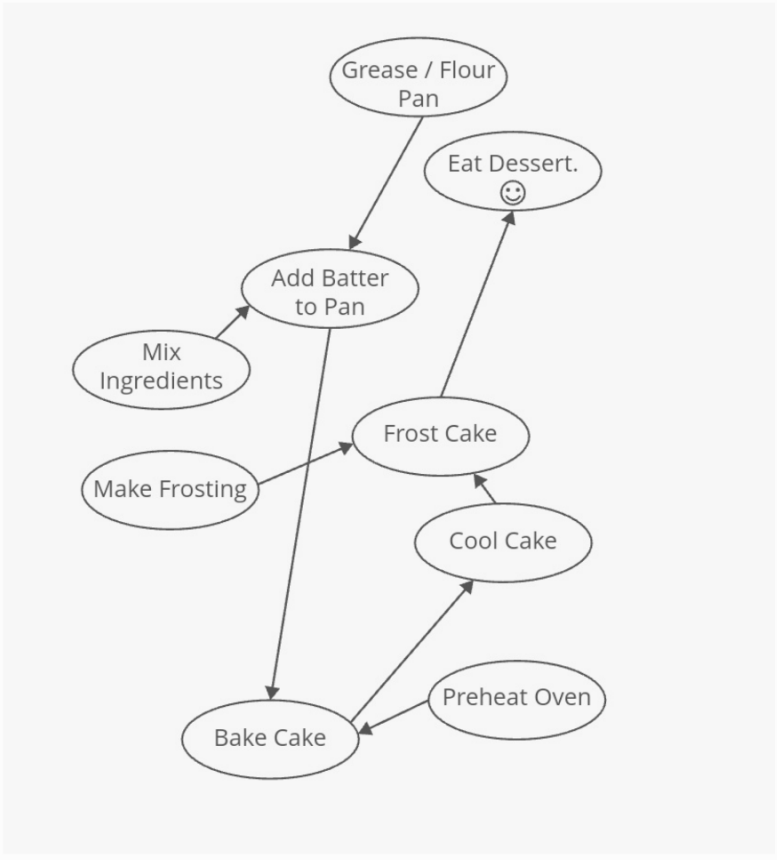
A, C, B, D, E

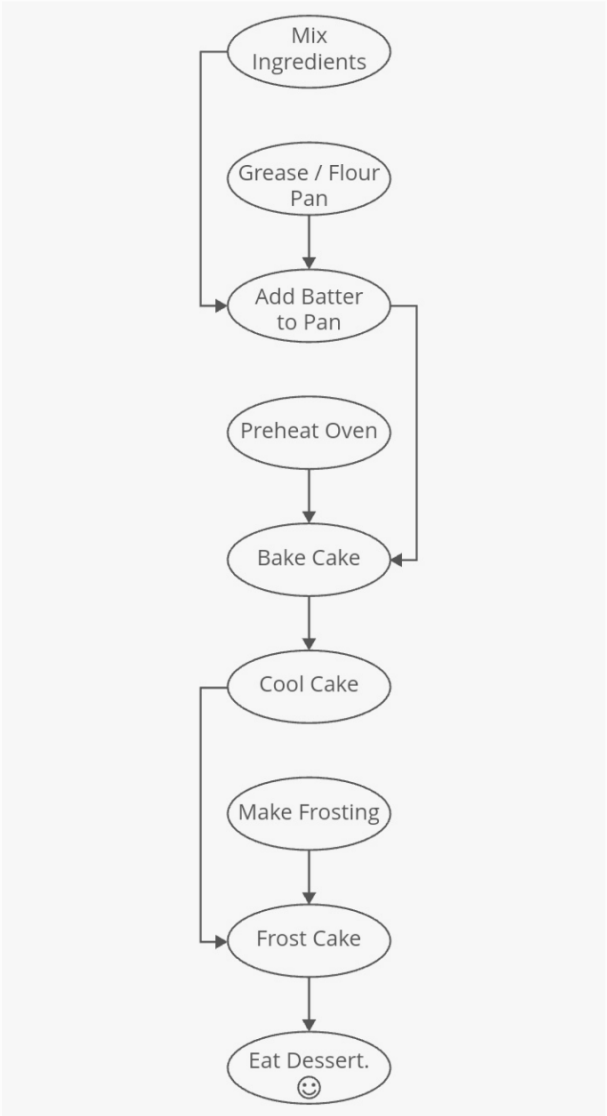
Topological order

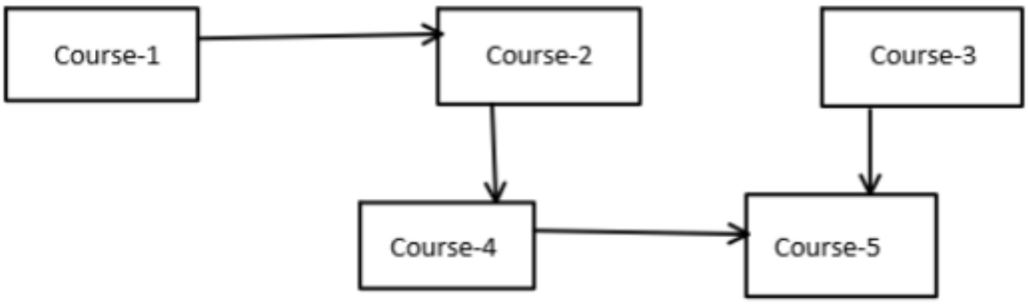


X topological sorted

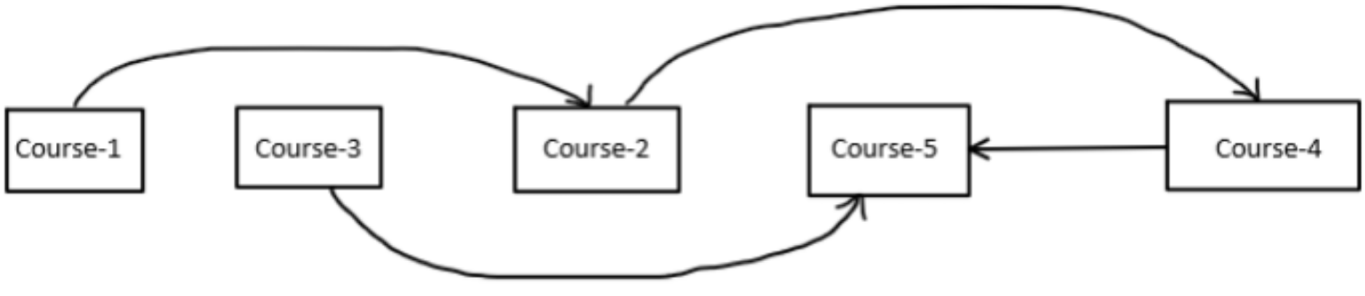
TS → directed graph.
Directed Acyclic Graph (DAG)



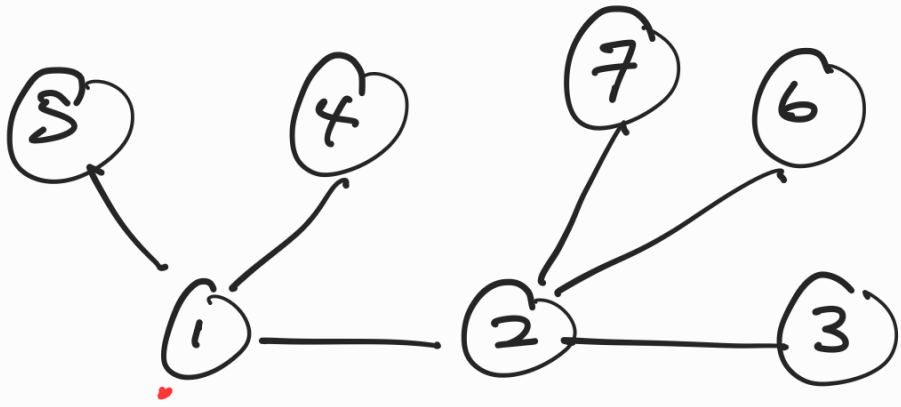




A possible topological ordering:



BFS and DFS

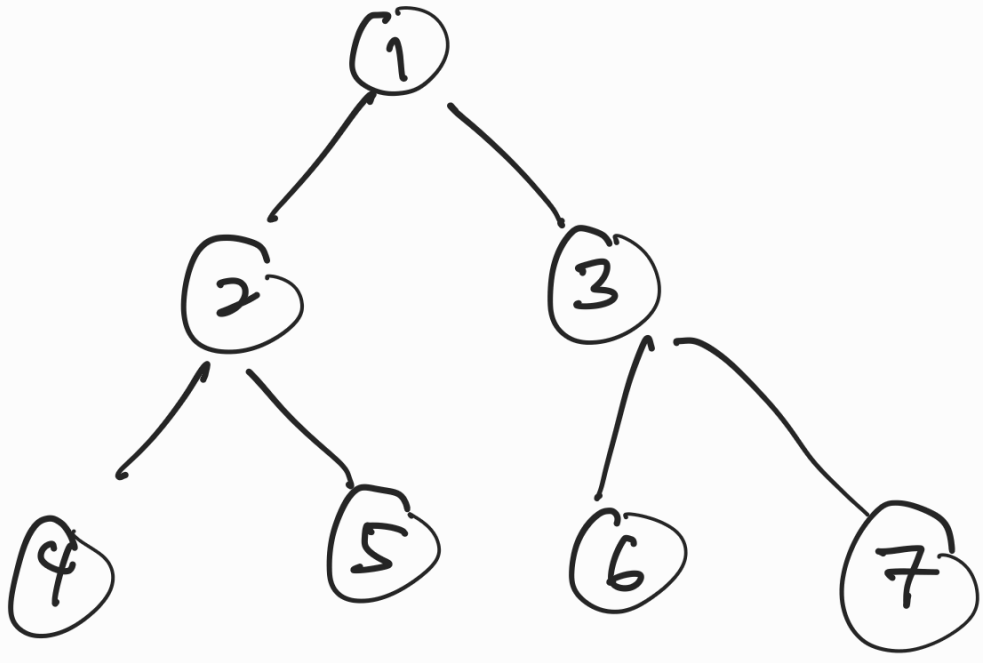


1. Visiting a vertex
2. Exploration of vertex

BFS : 1, 2, 4, 5, 7, 6, 3

DFS : 1, 2, 3, 6, 7, 4, 5

come back to 2 | come back to 2 | come back to 2 | come back to 1

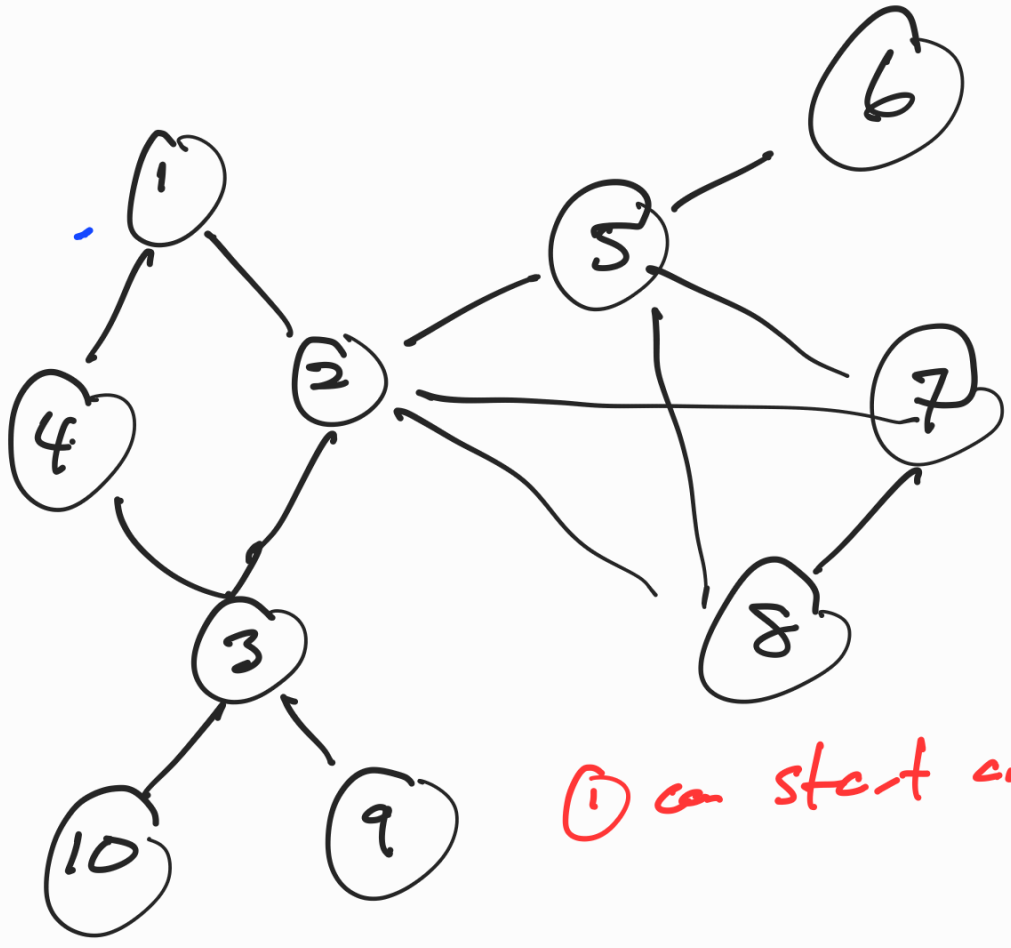


level order

BFS : 1, 2, 3, 4, 5, 6, 7

dfs : 1, 2, 4, 5, 3, 6, 7

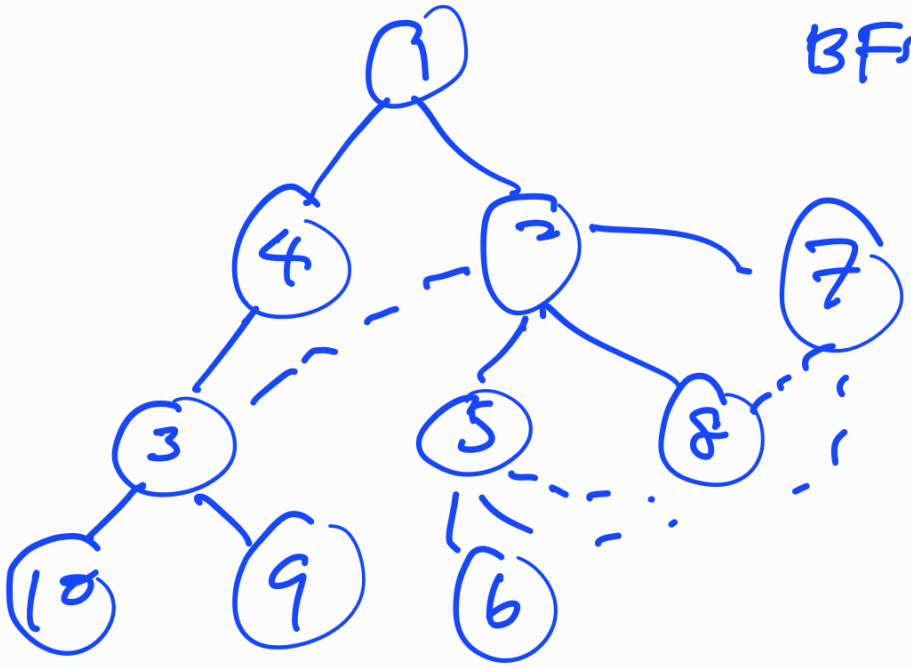
pre-order



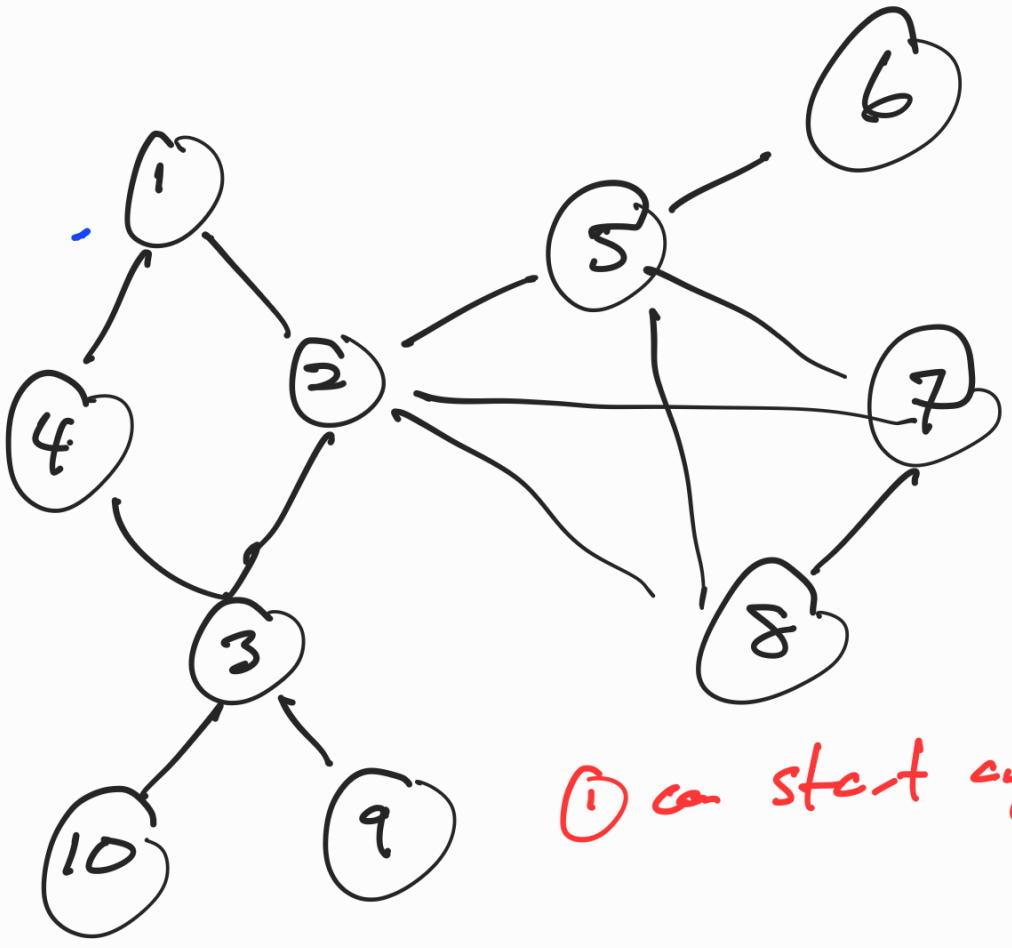
① can start any vertex

BFS : 1, 4, 2, 3, 5, 7, 10, 9, 6

Q 



BFS spanning tree



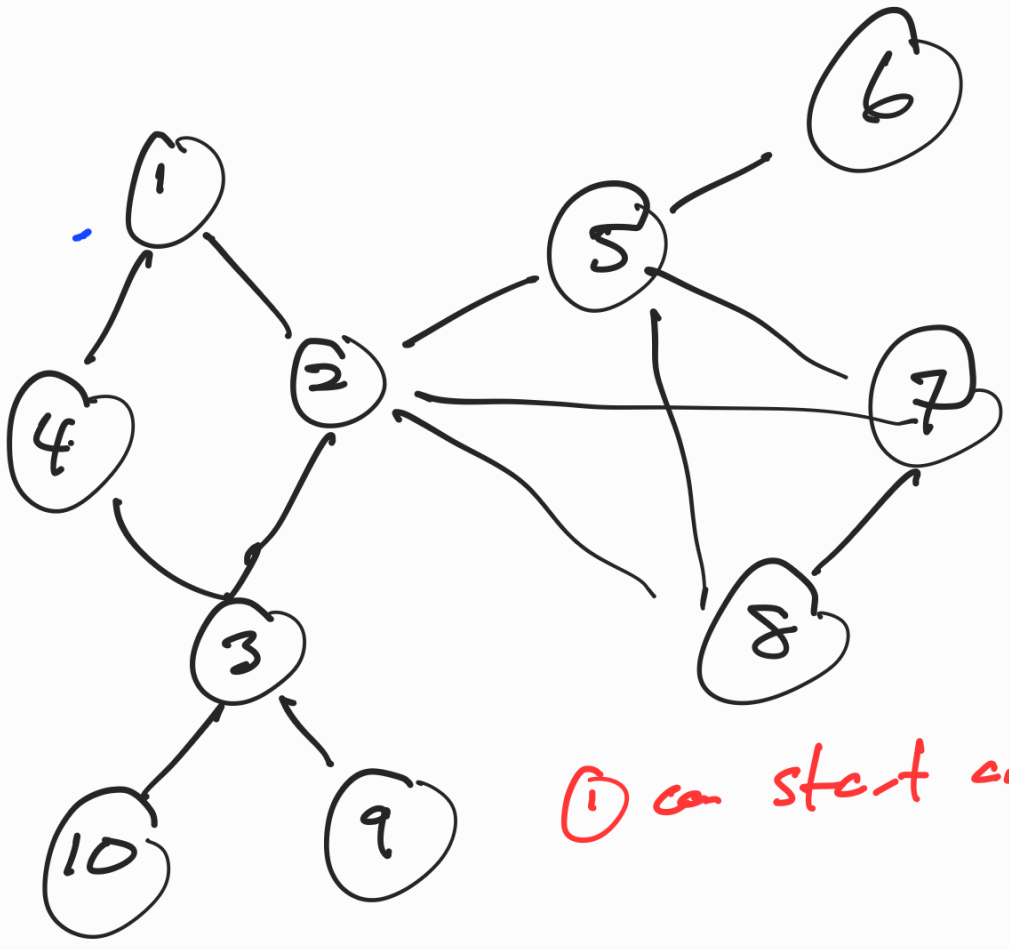
① can start any vertex

BFS : 1, 2, 4, 8, 5, 7, 3, 6, 10, 9

BFS : 5, 2, 8, 7, 6, 3, 9, 10, 4

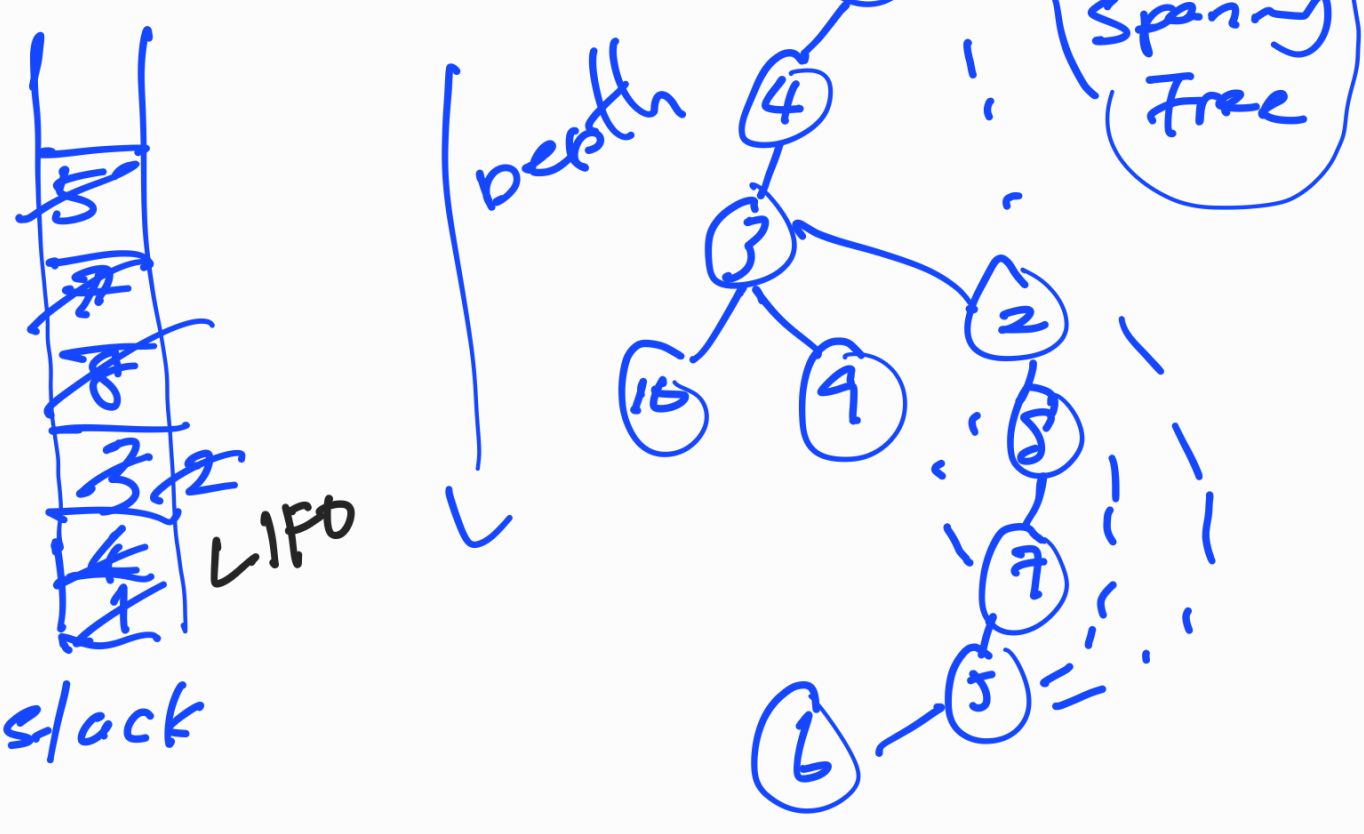
* All valid *

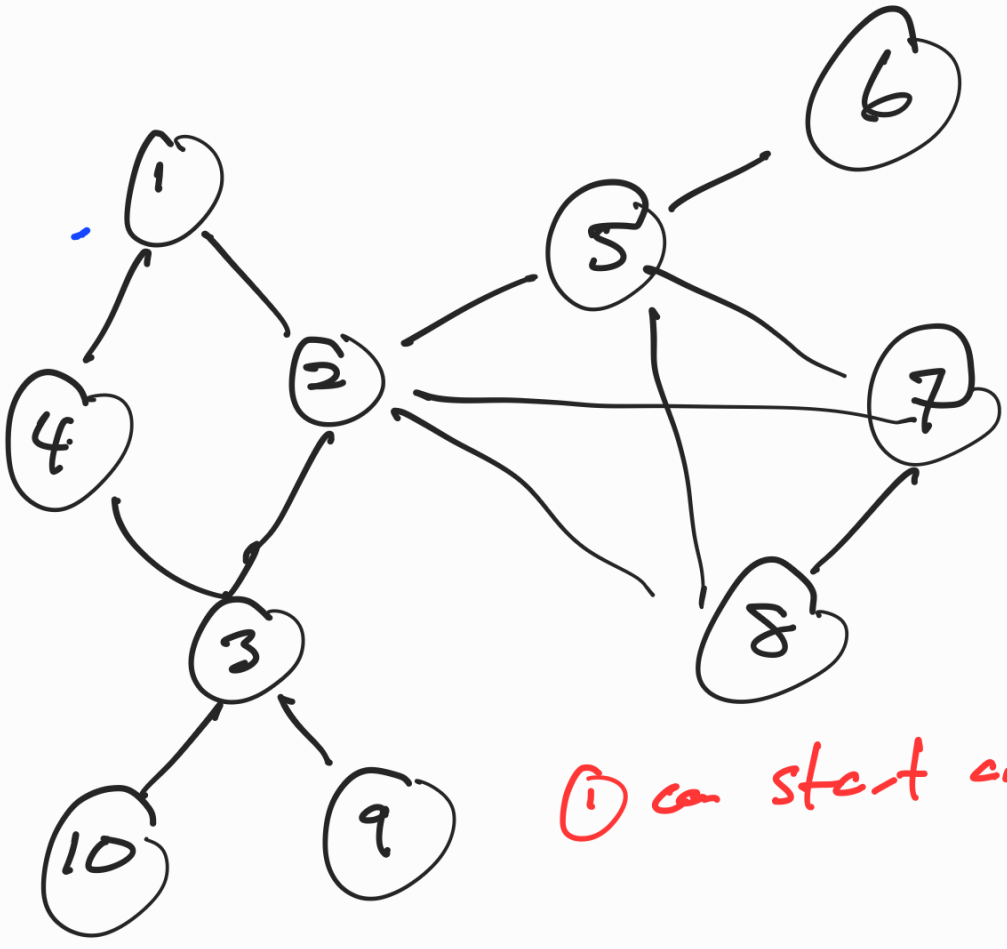
DFS



① can start any vertex

DFS : 1, 4, 3, 10, 9, 2, 8, 7, 5





① can start any vertex

1) 1, 2, 8, 7, 5, 6, 3, 9, 10, 4

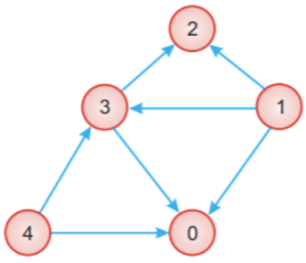
2) 3, 4, 1, 2, 5, 6, 7, 8, 10, 9

* All valid *

Topological sort

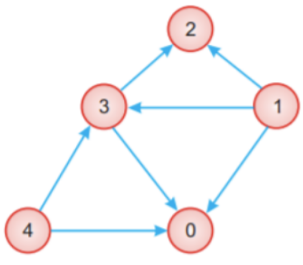
- In-degree

1

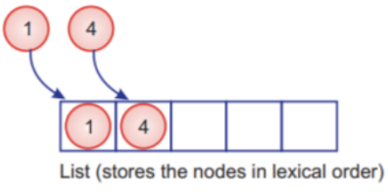


Node	Incoming edges Count
0	3
1	0
2	2
3	2
4	0

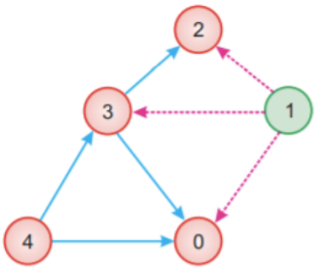
2



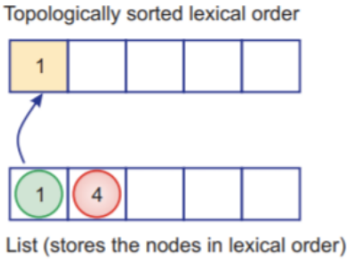
Node	Incoming edges Count
0	3
1	0
2	2
3	2
4	0



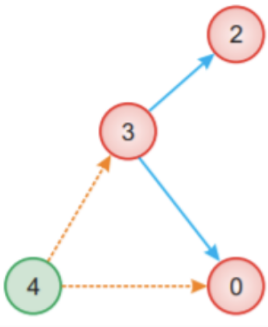
3



Node	Incoming edges Count
0	3 –> 2
1	0
2	2 –> 1
3	2 –> 1
4	0

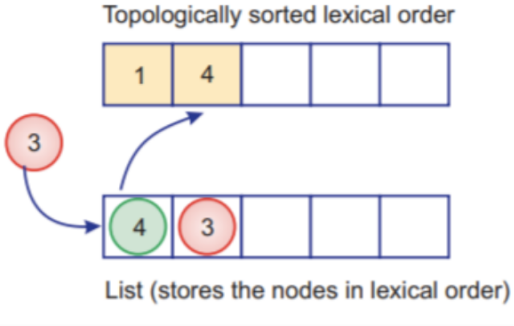


4

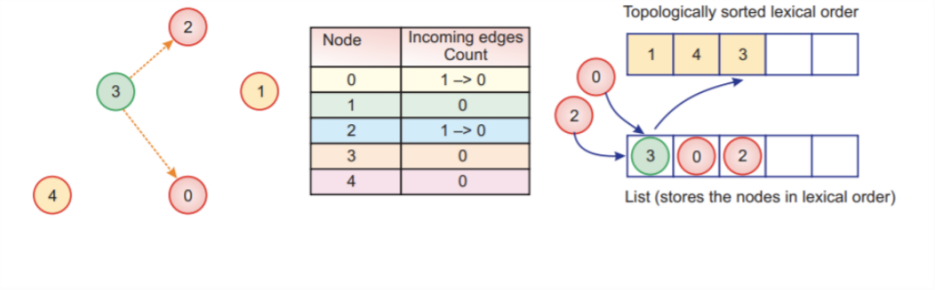


1

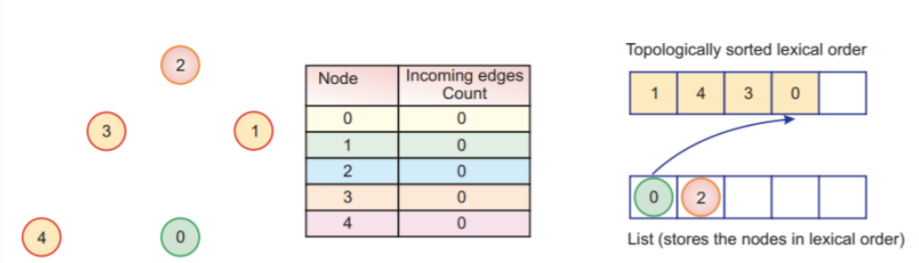
Node	Incoming edges Count
0	2 –> 1
1	0
2	1
3	1 –> 0
4	0



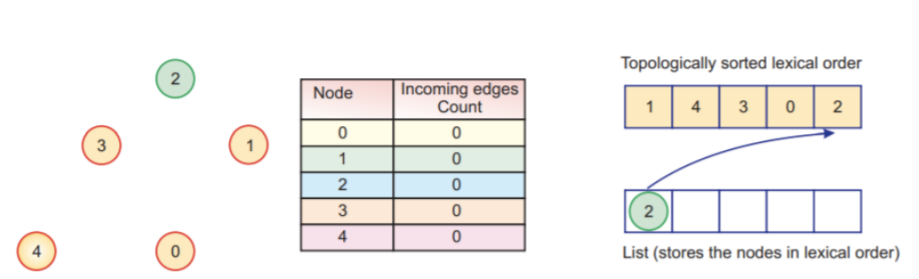
5



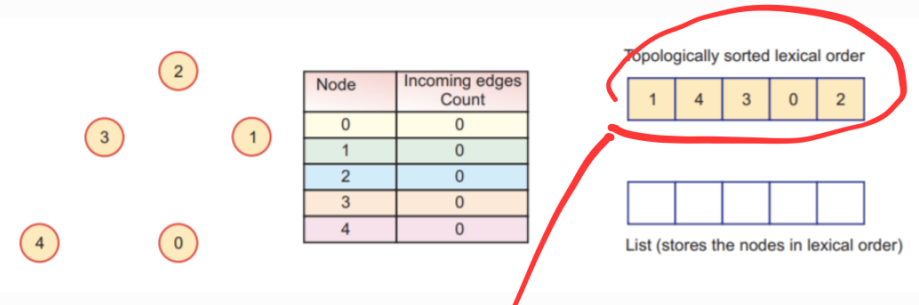
6



7

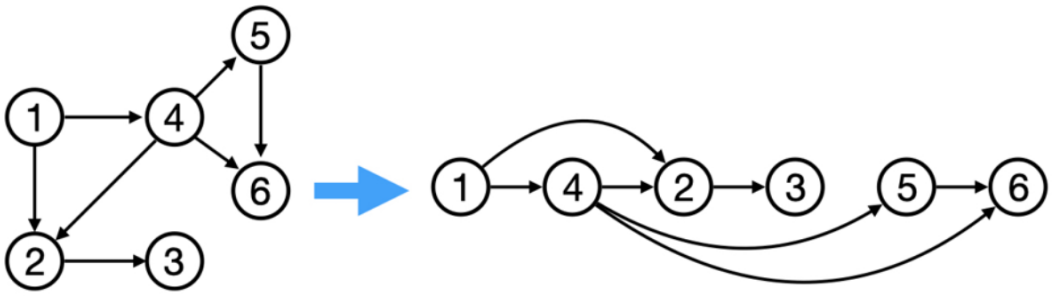


8

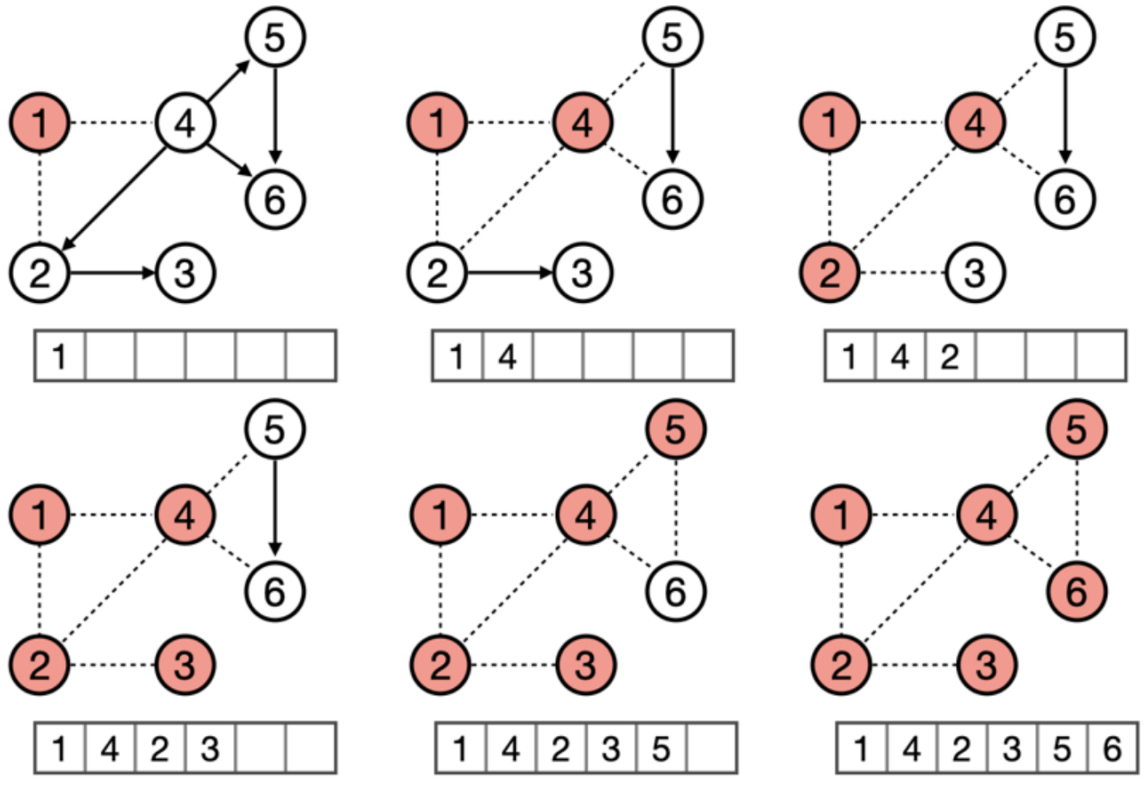


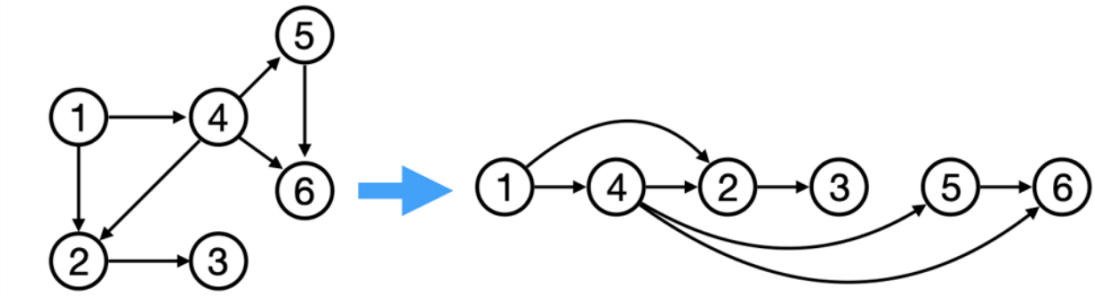
sorted topology

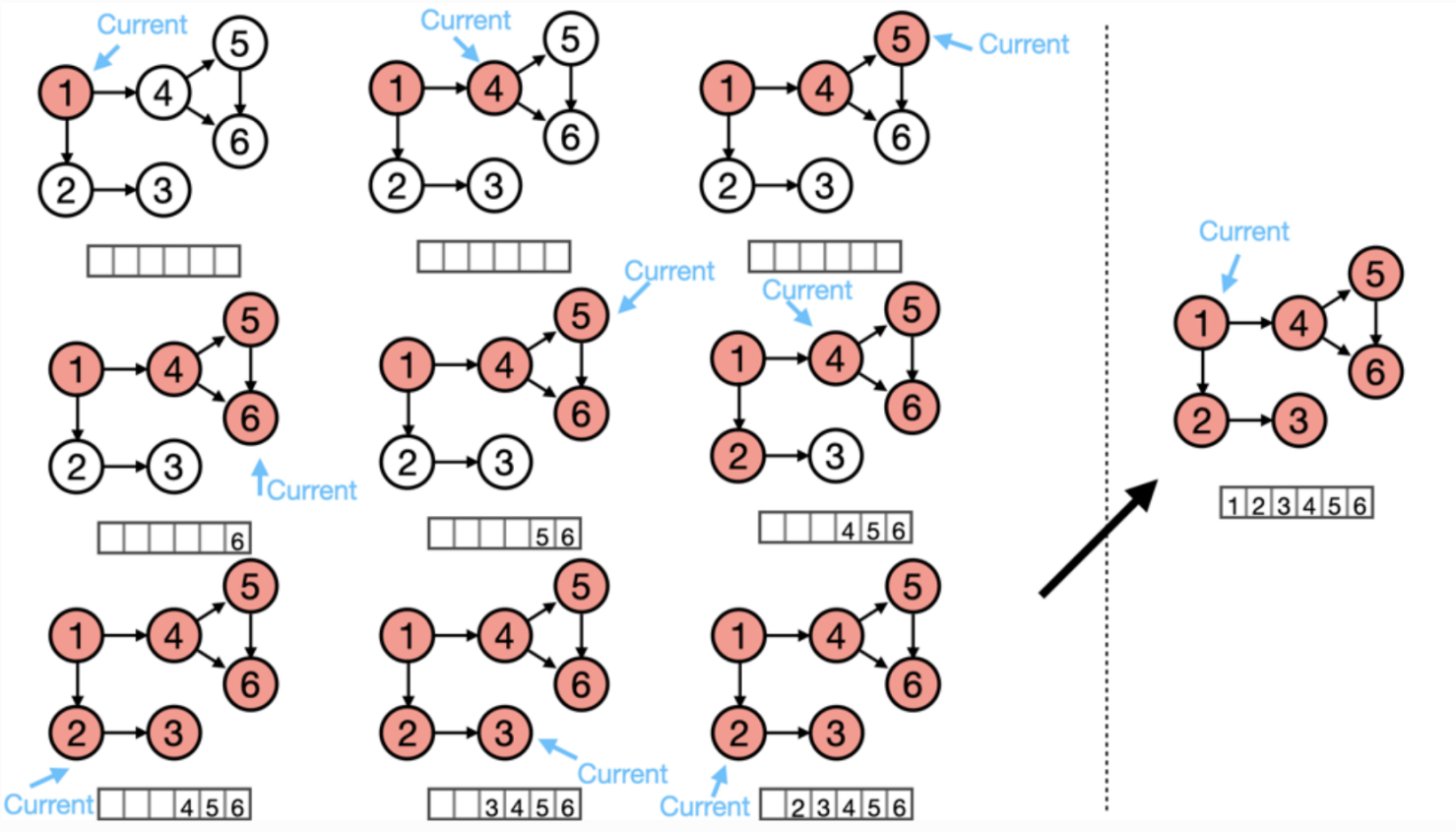
— DFS & BFS



According to Introduction to Algorithms, given a **directed acyclic graph (DAG)**, a topological sort is a linear ordering of all vertices such that for any edge `(u, v)`, `u` comes before `v`. Another way to describe it is that when you put all vertices horizontally on a line, all of the edges are pointing from left to right.




1
4
5
6
2
3
1
4
2
3
5
6


Current
Current
Current
Current
Current
Current
Current
Current
Current
Current
6
5 6
4 5 6
4 5 6
3 4 5 6
2 3 4 5 6
1 2 3 4 5 6