# BFS & DFS



## Graph Traversal Methods

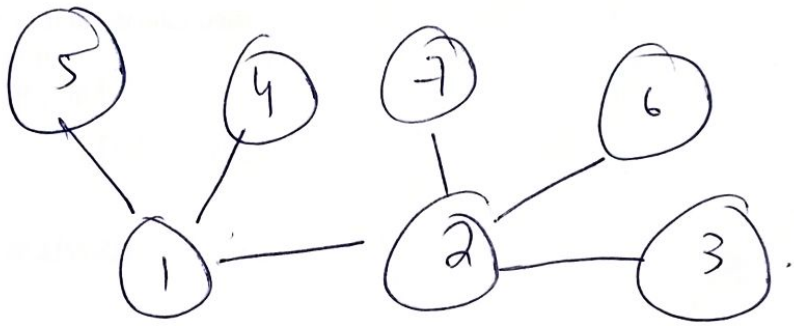
We need to know two things.

(1) → Visiting a Vertex ⇒ Going on some vertex

(2) → Exploration of vertex ⇒ visiting all adjacent vertex

Both complexities are $O(n)$

Start with BFS. No of vertex.




→ We can start with any vertex, lets start with (1),

We will visit all ajacent vertice

1, 2, 4, 5 (any order)

Now explore 1, 2, 4, 5, 7, 3, 6

All are explored

DFS

For some graph, start from ①.

So take ①.

1, 2. (we can take any)

Now visit ② & explore it.

1, 2, 3

Now explore ③

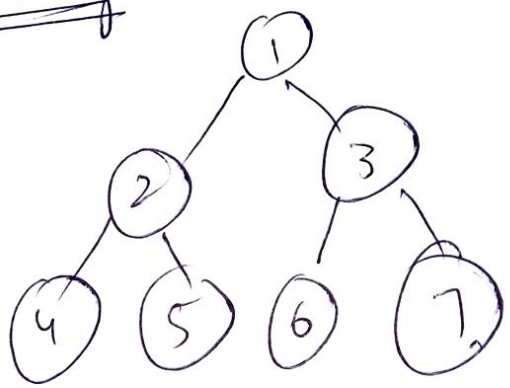
As nothing there then explore ①

again

1, 2, 3, 6, 7, 4, 5.

Results are different.

Another eg:

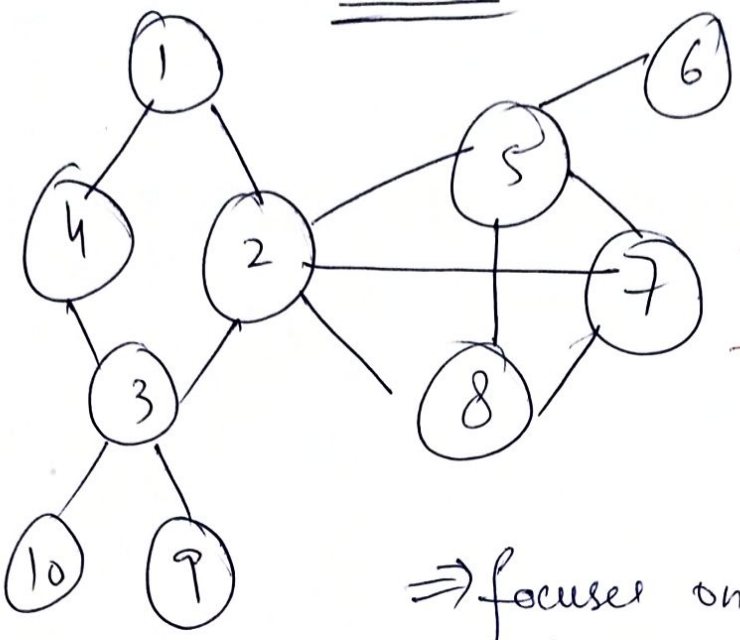



BFS 1, 2, 3, 4, 5, 6, 7 level order

DFS 1, 2, 4, 5, 3, 6, 7 pre order traversal

# BFS






Applicatin
→ Garbage Collection
→ Shortest path b/w nodes

⇒ focuses on finding shortest distance b/w nodes

Take a queue.

initial.

→ Start with any vertex (take 1)

Q = | 1 | (1)

Take out 1 & explore them.

| ✗ | 4 | 2 |


↓

1 explored

Now take 4

| ✗ | 4 | 2 | 3 | 5 | 8 | 7 | 10 | 9 | 6 |


BFS spanning tree

Already visited

spanning Tree
subset of Graph with
all vertices & minimum number
of edge. (All vertices connected)

..... => cross adjust

## Exercise

visit from 5

5 2 8 7 6 3 1 9 10 4

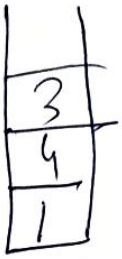
## DFS

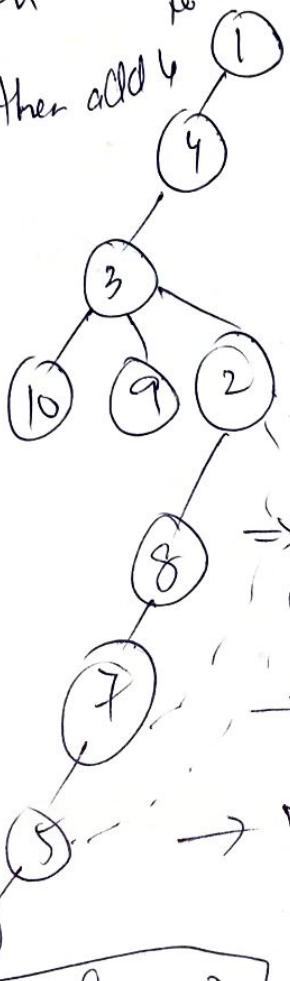
use stack

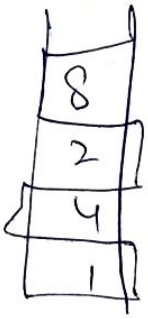
start from any point.

add 1.

visit 4 then 10 then add 3 then add 4




→ back adjust

→ DFS spanning tree

Applications
Scheduling, code defection
solving puzzles e.g. maze
topological sorting / course prerequisite

Exercise start from 3 3 4 1, 2 5 6 7, 10 9