BFS (Breadth First Search)

1) It finds the sharest path (in terms of edges) in an unweight graph.

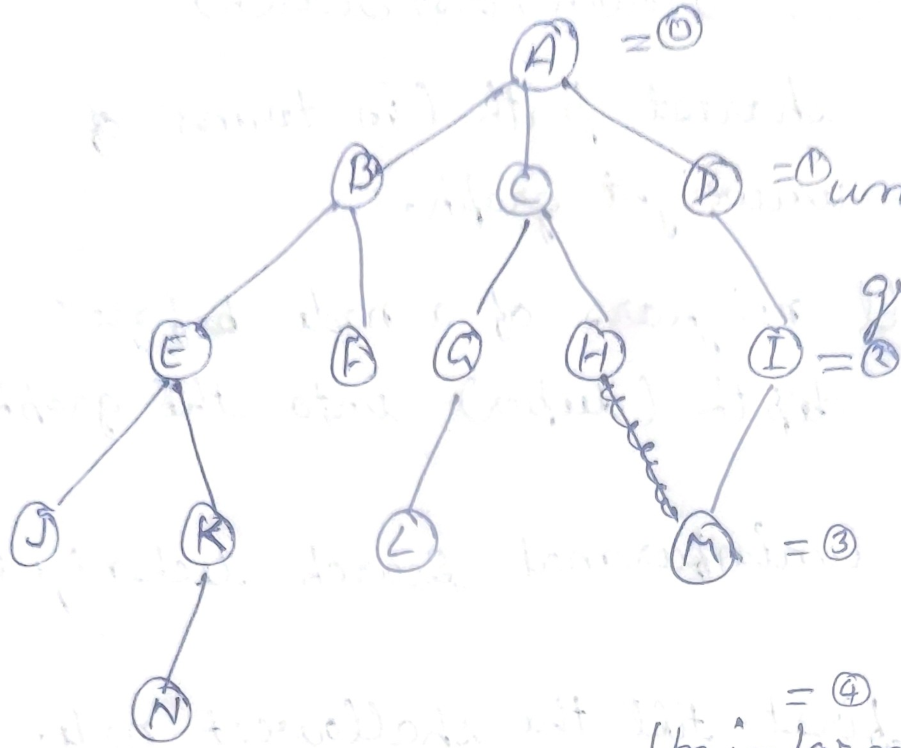
2) It explores all neighars of a node before going to the depth (deeper) into the graph.

Note: → It is uninformed search technique.
→ FIFO

→ Traversed first till the shallowest node.

→ Complete (It do the complete traversal on the same depth) It gurantee to give the answer so it is called complete.

Ex
- Finding the shartests path in the unweighted graph.
- Social Network [Finding degree of seperation between people]
- Web crawling (explaring a website by following links.



This is an unweighted graph.

Uninformed search means

A → B → G

Node A knows about B but does not know about G

Means A is uniformed about the Node G

* Traversal of above tree using BFS

← [A] ←

[B C D] ⇒ [~~B~~ C D] ⇒ [C D E F]

⇒ [~~C~~ D E F G H] ⇒ [~~D~~ E F G H] ⇒ [E F G H I]

[~~E~~ F G H I] ⇒ [~~F~~ G H I J K] ⇒ [G H I J K]

[~~G~~ H I J K] ⇒ [~~H~~ I J K L] ⇒ [I J K L]

$\boxed{\cancel{I}\ J\ K\ L} \Rightarrow \boxed{\cancel{J}\ K\ L\ M} \Rightarrow \boxed{K\ L\ M}$

$\boxed{\cancel{K}\ L\ M} \Rightarrow \boxed{\cancel{L}\ M\ N} \Rightarrow \boxed{M\ N}$

$\boxed{\cancel{M}\ N} \Rightarrow \boxed{N}$

In this ~~above~~ example we can see that it traversing level by level or depth by depth.

Suppose our goal state is (L), How will find the our goal state (L).

→ Traverse using BFS level by level

→ Check the parent of (L), which (G)

→ Check the parent of (G) which is (C)

→ Check the parent of (C) which is (A)

→ (A) is our starting point, so now we can say that $\boxed{A\ C\ G\ L}$ is our path to reach the goal state.

Result :- It always gives the optimal result in both cases if weight is given to the edge. 1 or 2.

Time complexity :- $O(b^d)$

Branch factor = number of branches of a node.

d = depth.

DFS (Depth First Search)

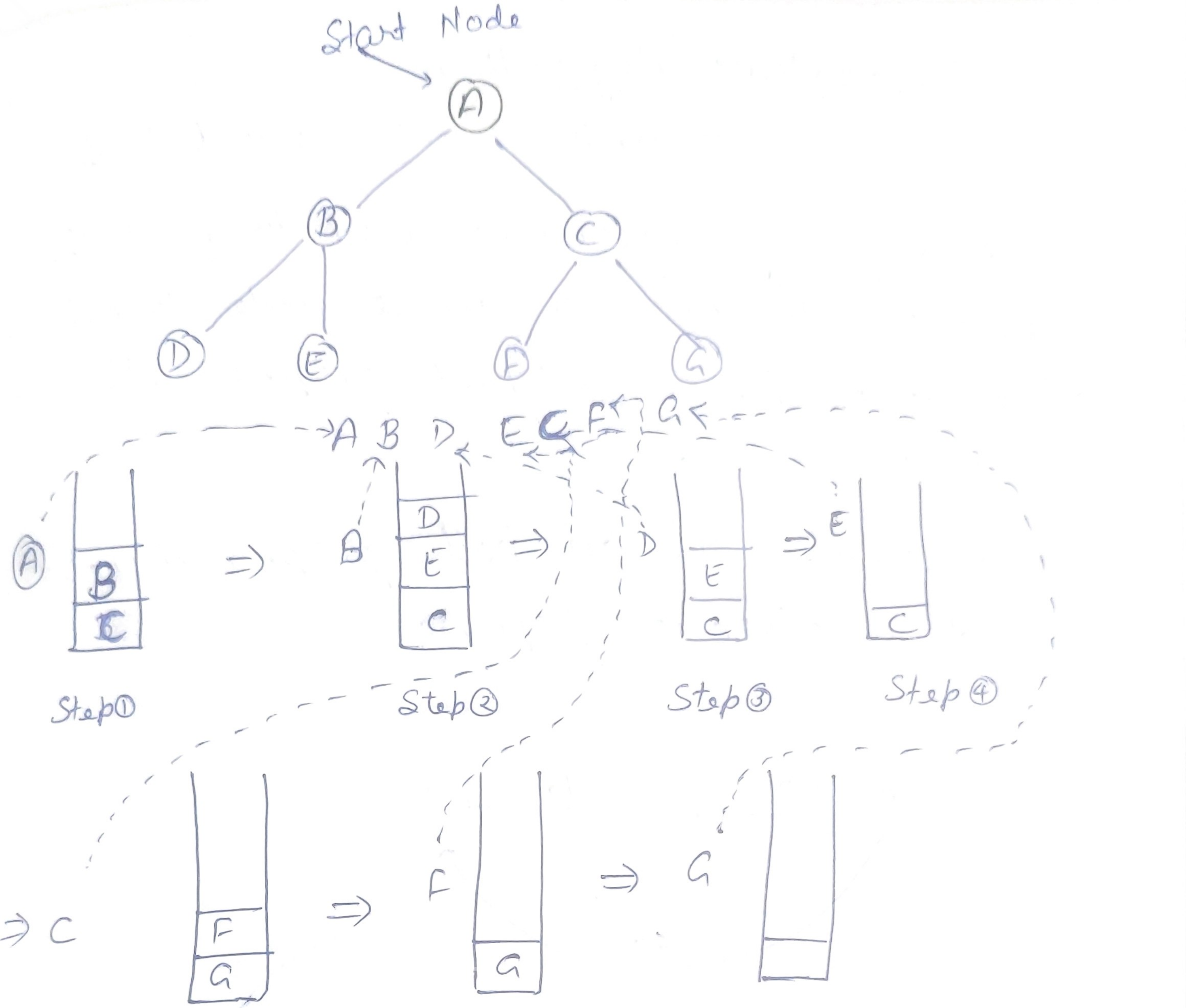
It is a graph traversal algorithm used to explore all nodes of a graph or tree by going as deep as possible before backtracking.

Application:- → Pathfinding algorithms (Solving mazes
→ Topological Sorting in Directed Acyclic Graphs (DAGs)
→ Detecting cycles in graphs.
→ Finding connected components in an undirected graph.
→ Solving puzzles (like Sudoku, Knight's tour

Note:- • It is uniformed Search technique

- Stack (LIFO)
- Deepest Node
- Incomplete

Uniformed ⇒ Means we have the current knowledge, not the complete domain knowledge.



It may possible that tree has infinite number of nodes than there is a possibility that it may go in cyclic loop so in this case it may not find the goal state.

That's the reason it is called the "Incomplete search".

Result :- It can give the non optimal sol?

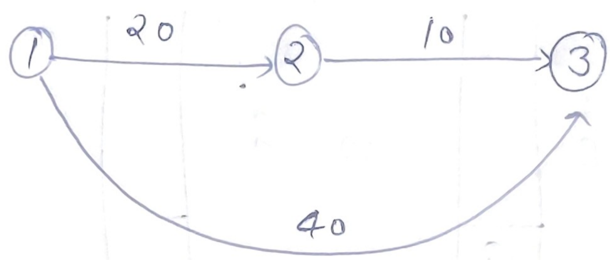
Time Complexity :- $O(b^d)$ b = branching factor, d = depth.

Dijkastra's Algorithm

Single ~~path~~ source Shortest path.

It is a greedy algorithm used to find the shortest path from a source node to all other nodes in a weighted graph (with no negative weights).

It works well on both graphs directed and undirected graphs:



Relaxation :-

if $d(u) + c(u, v) < d(v)$

$$d(v) = d(u) + c(u, v)$$

Assumption :-

Initialize the distance of all node or vertex as infinite (∞), except source node, Distance of source node will be always zero (0) because it is at same place.

1) Find the distance of node (2) from node (1)
Here node (1) is the source node.

distance of node (1) is 0

By applying the relaxation

node (1) is u
node (2) is v

$d(u) = 0$, $d(v) = \infty$

$c(u,v) = 20$

Here $d(u) + c(u,v) = 0 + 20 < d(v)$

so $d(v) = d(u) + c(u,v)$

$= 0 + 20$

$d(v) = 20$

(2) Find the distance of node (3) from source node (1),

There are two ways

1) $(1) \longrightarrow (2) — (3)$

2) $(1) \longrightarrow (3)$

* In above we already find out the distance of node (2), Now node (2) will become the source node.

$d(u) = 20$ $c(u,v) = 10$

$d(v) = \infty$

$d(u) + \cancel{d}\, c(u,v) \Rightarrow 20 + 10 = 30$

which is less than ∞

$d(v) = d(u) + c(u,v) = 30$

$d(v) = 30$

* Distance from source node ① to node ③

$d(u) = 0$ $\quad$ $d(v) = 30$ (It is already calculated)

$c(u, v) = 40$

$\Rightarrow$ if $d(u) + c(u, v) < d(v)$

$\cancel{d(v)} = d(u) + c(u, v)$

$= 0 + 40$

$= 40$

$d(u) + c(u, v) \nless d(v)$

$40 \nless 30$

so $\underline{\underline{d(v) = 30}}$