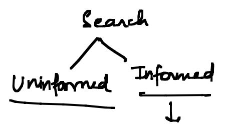
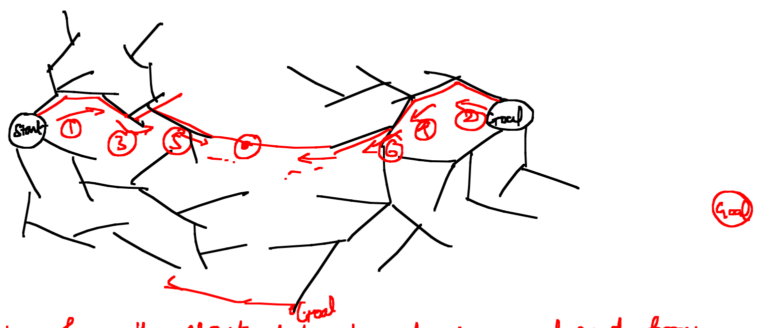
24/01/2022 AI.



Uninformed Search (Blind Search)

- → BFS
- → DFS
- → DLS (Depth Limit Search)
- → DFID (Depth First Iterative Deepening)
- → Bidirectional Search
- → Uniform Cost Search → (special version of BFS)

Bidirectional Search

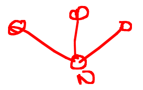


→ Alternating searching from the start state towards the goal and from the goal state towards the start state.

→ Stop when the frontiers intersect.

S
N1
N2
N4
N5
G

→ Works well only where there is only one goal state.



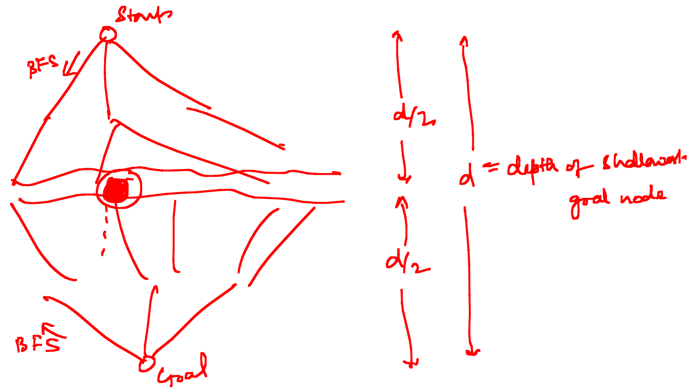
Problem:- How do we search backwards from goal?

↳ Requires the ability to generate predecessor states.

Predecessor of a node n = all the nodes that have n as successor.

BFS = $O(b^d)$

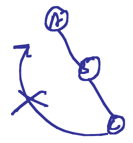
Bi-directional search = $2 \cdot b^{d/2}$ Better



	BFS	DFS	DFID	Bi-directional
Time	b^d	b^m	b^d	$b^{d/2}$
Space	b^d	$b \cdot m$	b^d	$b^{d/2}$
Completeness	✓	✗	✓	✓
Optimality	✓	✗	✓	✓

How to avoid Repeating states while searching ?

1. Do not return to the state we just came from.
2. Do not create path with cycles in them
3. Do not generate any state that was ever created before.



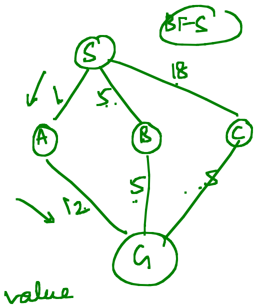
Graph Search Algo

```
Let fringe be a list containing the initial state.
Let closed be initially empty.
Loop
    If fringe is empty return fail.
    Node ← remove-first (fringe).
    if Node is goal then
            return path from initial state to Node.
    else
        put Node in closed.
        generate all the succesors of Node as S.
            for all nodes m in S.
                If m is not in closed.
                    merge m in fringe.
End Loop.
```

Uniform Cost Search

Queue

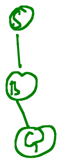
~~S~~ ~~A~~ B C G



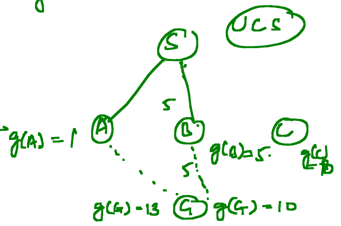
→ Enqueue nodes by path cost.   Priority Queue

Let $g(n)$ = cost of the path from the start node to the current node n.

Sort the nodes in the priority queue (fringe) by increasing value of $g(n)$.

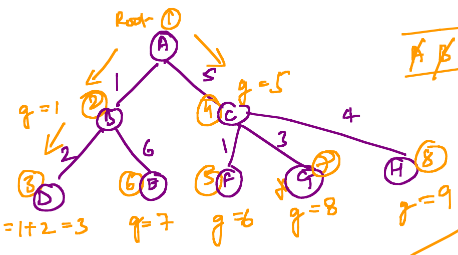


~~S~~ ~~A~~ ~~B~~ G C.



- Complete ✓
- Optimal ✓
- Time/Space complexity = $O(b^d)$ ✓

Uniform Cost Search



~~A~~ ~~B~~ ~~D~~ ~~C~~ ~~F~~ ~~E~~ ~~G~~ H

Order:- A B D C F E G | H

$O(b^{C^*/\epsilon})$

C^* :- cost to optimal solⁿ.

ϵ : every step cost.