Bidirectional Search Algorithm

Breadth first Search (BFS)

→ It is the most common Search Strategy to traversing a tree/graph.

→ This algorithm searches breadthwise in a tree & graph, so it is called breadth-first search.

→ BFS algorithm starts searching from the root node of the tree and expands all success node at the current level before

→ BFS algorithm is an example of general-graph search algorithm

→ BFS implemented using FIFO queue data structure.

Advantages:

→ BFS will provide a solution if any solution exists.

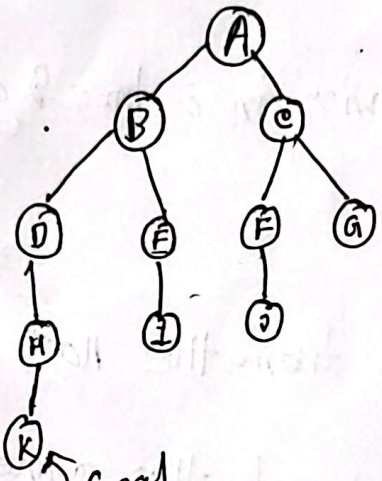
→ If there are more than one solution in a given problem, the BFS will provide minimal solution which requires the least no of steps

Disadvantages:

→ It requires lot of memory since each level of the tree must be saved into memory to expand the next level.

→ BFS needs lots of time. if the soln is far away from the root nodes.

Example BFS



Time complexity

d = depth of shallow

b = node of every state

$T(b) = 1 + b^2 + b^3 + \dots + b$

$= O(b^d)$

$A \to B \to C \to D \to E \to F \to G \to H \to I \to J \to K$ Space $= O(b^d)$

* BFS is complete.

optimality: optimal

Bidirectional Search Algorithm

→ Two different searches are run simultaneously. Searching starts from both ends. (forward search and backward search)

S → G G → S

→ Hence single search Graph is replaced with two small graphs.

→ Any search technique can be used (BFS, DFS, ...)

→ when Graphs intersects with each other it stops searching

Advantages

→ Fast

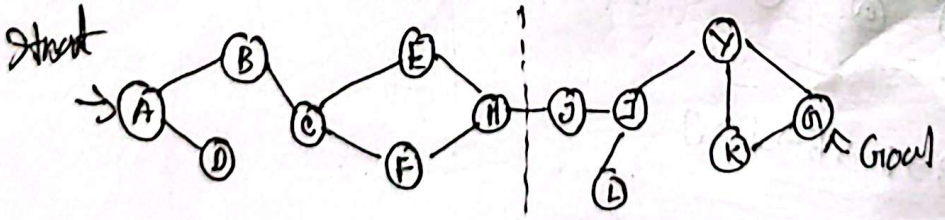
→ Less memory

Disadvantages:

→ Implementation is difficult

→ Goal state should be known in advance

Example!



(BFS)

Forward

ABDCEFH

Backward

GKXILJH

Path :→ A → B → D → C → E → F → H → J → L → I → Y → K → G

Depth-Limited Search Algorithm

→ Working is similar to DFS but with predetermined limit.

→ Helps in solving the problem of DFS (Infinite Path)

Time complexity $O(b^d)$
Space complexity $O(bd)$

Termination Conditions:

i) Failure value: There is no solution

ii) Cutoff Failure: Terminates on reaching predetermined depth.

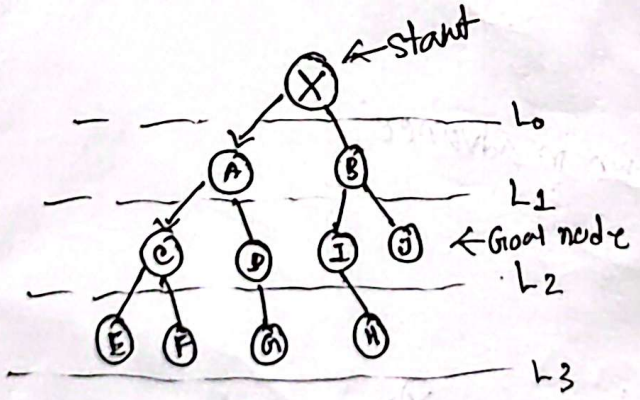
Advantages:

→ memory efficient

Disadvantages:

→ Can be terminated without finding solution (incompleteness)

→ not optimal.



depth, d=2

Path → X → A → C → D → B → I → J

Uniform Cost Search (UCS)

→ It is used for weighted Tree/Graph traversal

→ Goal is to path finding to goal node with lowest cumulative cost

→ Node expansion is based on path costs.

→ Priority Queue is used for implementation. (Higher Priority to minimum cost)

→ Supports backtracking, complete.

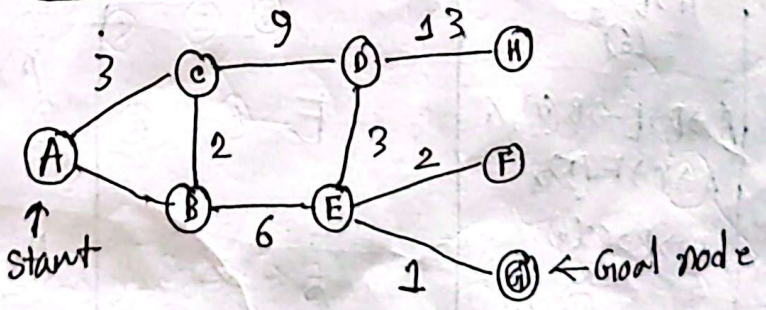
Advantages

→ optimal Solution

Disadvantages

→ Struck in infinite loop.

Example Path: A → C → B → E → G Cost: 3+2+6+1
12

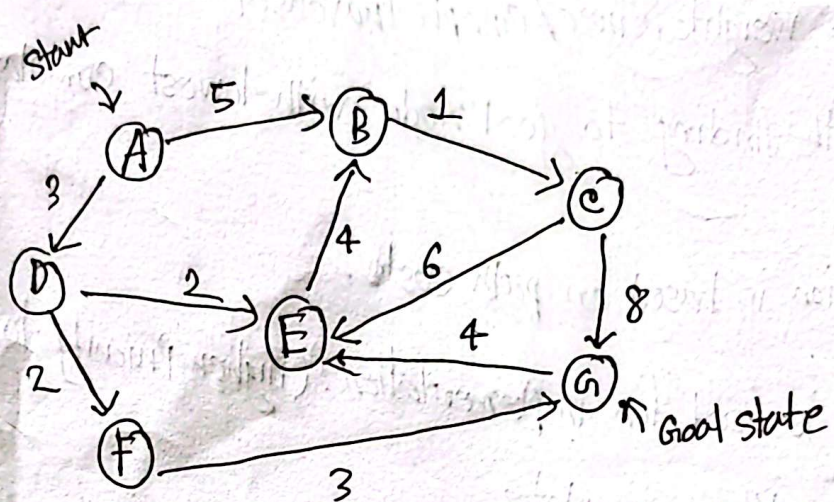


For Goal node 'H'

Path: A → C → B → E → G → E → F → E → D → H

Cost: 3+2+6+1+1+2+2+3+13 = 33

Consider the following graph. Let the starting node be A and the Goal node be G



	Frontier List	Expand List	Explored List
1.	A	A	–
2.	A→D (3), A→B (5)	D	A
3.	A→D→E (5), A→D→F (5), A→B (5)	B	A, D
4.	A→D→F (5), A→D→F (5), A→B→C (6)	E	A, D, B
5.	A→D→E→B (9), X, A→D→F (5), A→B→C (6),	F	A, D, B, E
6.	A→D→F→G (8), A→B→C (6)	C	A, D, B, E, F
7.	A→D→F→G (8), A→B→C→E (12) X, A→B→C→G (14)	G	A, D, B, E, F, C
8	A→D→F→G (8) (goal)	NULL	A, D, B, E, F, C, G traversed

Iterative Deepening Depth-first search (IDFS/IDS)

→ It is the combination of both DFS and BFS.

→ Best Depth limit is found out by gradually increasing limit.

Initially d=0

→ every iteration increase by 1

→ Complete if the branching factor is finite. [It's use stack]

Advantage:

→ Incorporates benefits of both DFS and BFS

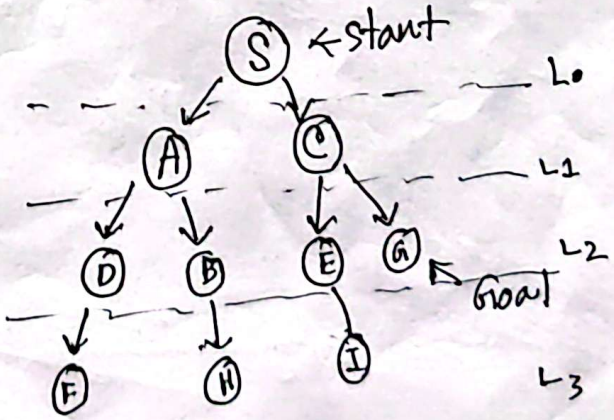
fast and less memory required

Disadvantage

→ Repeat the work/process.

{ Time complexity $O(b^d)$
Space complexity $O(bd)$

working of IDFS/IDS



1st Iteration, d=0 [S]

2nd Iteration, d = 0+1 = 1 [S→A→C]

3rd Iteration d = 1+1 = 2

[S→A→D→B→C→E→G]

4th Iteration d = 2+1 = 3

Heuristic Search [Informed Search]

Heuristic Search: Tries to optimize a problem using heuristic function.
→ tries to solve problem in minimum steps/cost

Heuristic function: It is a function $f(n)$ that gives an estimation on the cost of getting from node 'n' to the Goal state.
→ (estimated value)

Types of Heuristic

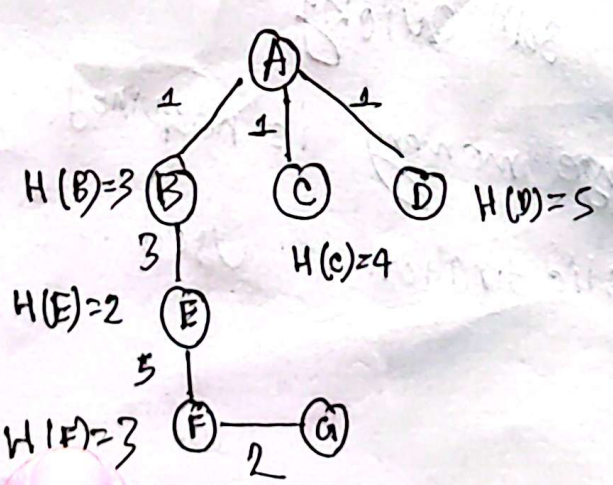
(I) **Admissible:** In this Heuristic function, never overestimates the cost of reaching the Goal.

$$H'(n) \leq H(n)$$

$h'(n)$ is always less than or equal to actual cost of lowest-cost path from node n to goal.

(II) **Non-Admissible:** - Overestimate

$$H(n) > H'(n)$$



Blind Search: It is also known as unknown/ Uninformed Search.

→ There is no info about the Searching
→ No knowledge of where the Goal.
→ Eg:- Depth first, Breadth first Search
→ Efficiency is low
→ Slower than Heuristic
→ Large memory is used

Heuristic Search: It is a method of solving problem more easily and fast. They have knowledge of where goal or finish of the graph. (Informed Search)

Eg: Hill Climbing, A*, AO*

→ Highly efficient < less time / less cost
→ finds soln quicky
→ no large memory is required.
→ Heuristic function is used.

Beam Search:

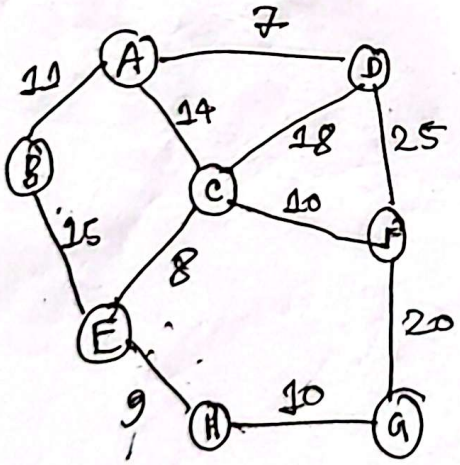
optimized version of Best first Search

→ Heuristic Search Algorithm

→ Explores a Graph by expanding the most promising node in a limited set.

→ Reduces Memory Requirement. (Greedy Algorithm)

Here only Predetermined no of Best Partical solutions are kept as candidates.



A → G = 40
B → G = 32
C → G = 25
D → G = 35
E → G = 19
F → G = 17
H → G = 10
G → G = 0

Beam Search
(β = 2)