Rajiv Gandhi University of Knowledge Technologies

Catering to the Educational Needs of Gifted Rural Youth of Andhra Pradesh

(Established by the Govt. of Andhra Pradesh and recognized as per Section 2(f) of UGC Act, 1956)

Rajiv Knowledge Valley Campus

Department of Computer Science and Engineering

Artificial Intelligence

Day-5

Presented by

R Sreenivas

Assistant Professor

RGUKT RK Valley

Agenda

Search Techniques

Heuristic Search Techniques

Quote of the Day

Search Techniques

In Artificial Intelligence (AI), search techniques are strategies used to navigate through problem spaces to find solutions. These techniques are broadly classified into two categories as follows

Search Techniques

Uninformed (Blind) Search

- Breadth First Search
- Depth First Search
- Depth – Limited Search
- Iterative Deepening Search
- Uniform Cost Search

Informed (Heuristic) Search

- Best First Search
- A* Search
- Iterative Deepening A*

Uninformed Search

Uninformed search algorithms explore the search space blindly, using only the problem's structure without any domain-specific knowledge or heuristics. They operate without guidance, systematically examining all possibilities.

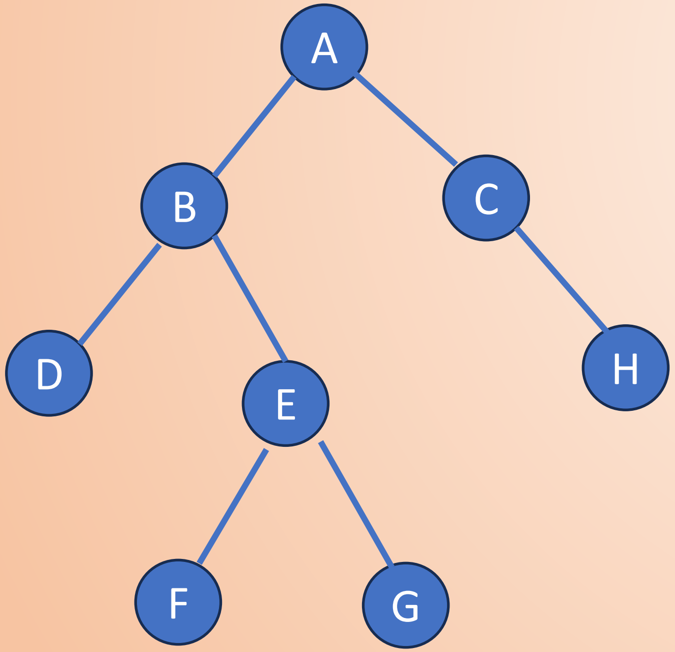
Uninformed (Blind) Search

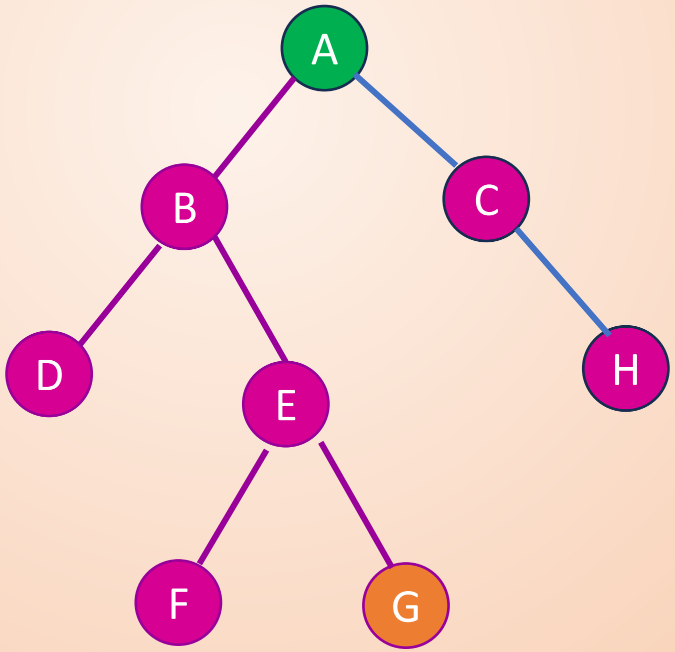
- Breadth First Search
- Depth First Search
- Depth – Limited Search
- Iterative Deepening Search
- Uniform Cost Search

Breadth First Search

Uninformed Search

- Breadth-First Search (BFS) is a uninformed search technique used to explore all possible nodes in a search space systematically.
- it explores all nodes level by level, ensuring that all nodes at the current depth are explored before moving on to nodes at the next depth.





Time Complexity : O(b^d)

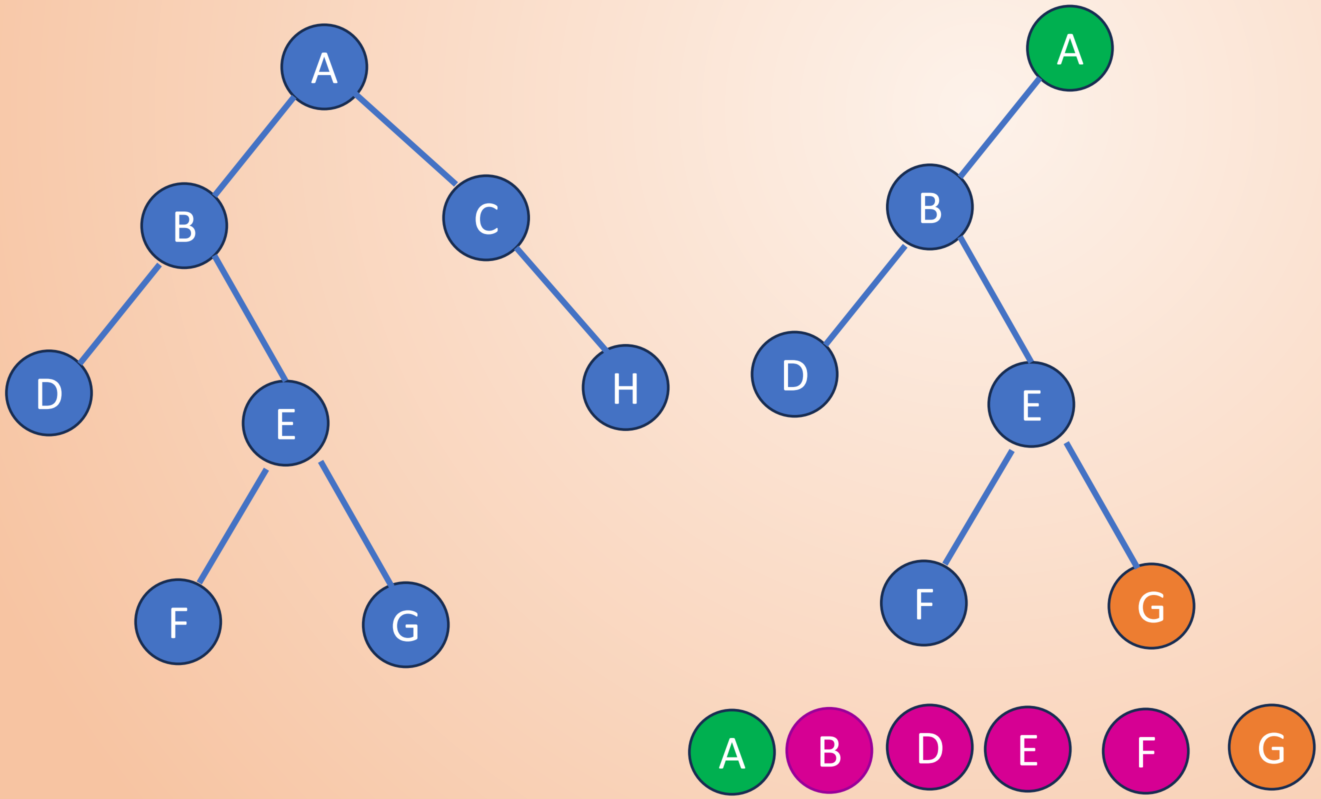
Space Complexity : O(b^d)

Completeness : Yes

Optimality : Yes

Depth First Search

- Depth-First Search (DFS) is a uninformed search technique used to explore all possible nodes in a search space systematically.
- It starts from the root node and follows each path to its greatest depth node before moving to the next path.



Time Complexity : O(b^m)

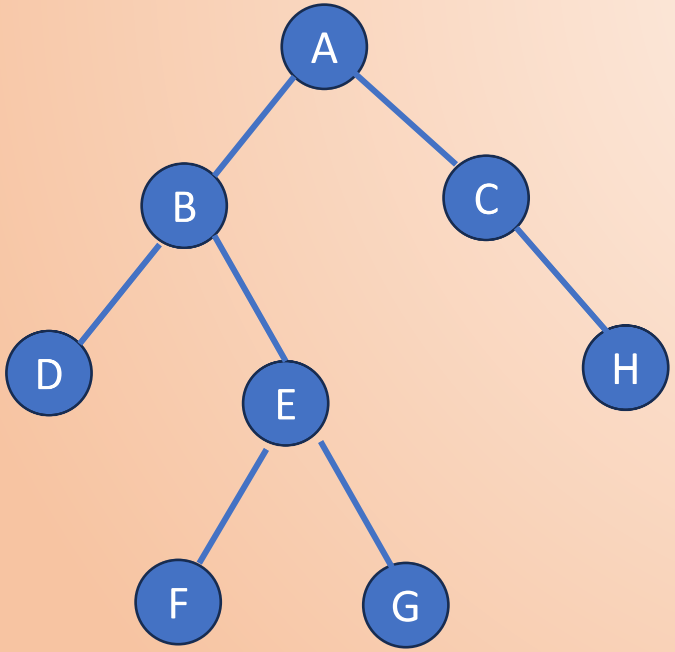
Space Complexity : O(b.d)

Completeness : Yes

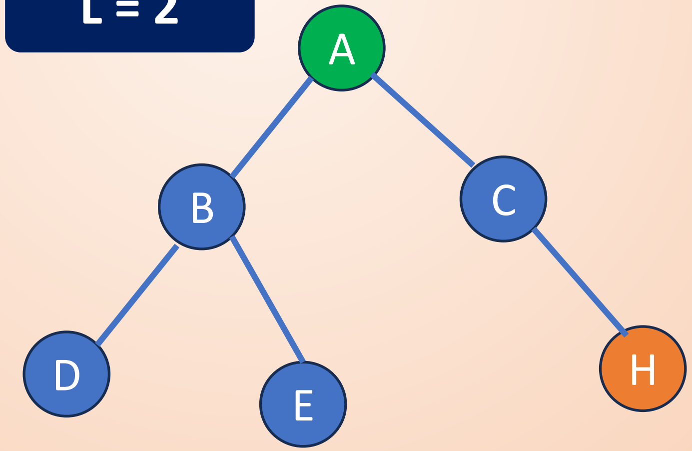
Optimality : No

Depth – Limited Search

- Depth-Limited Search (DLS) is a uninformed search technique used to explore all possible nodes in a search space systematically.
- A depth-limited search algorithm is similar to depth-first search with a predetermined limit.
- Depth-limited search can solve the drawback of the infinite path in the Depth-first search.



L = 2



Time Complexity : O(b^L)

Space Complexity : O(L)

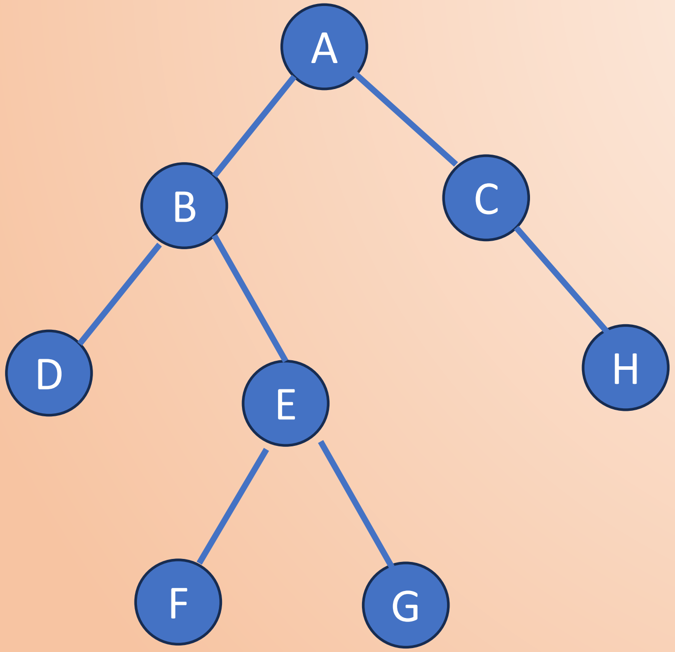
Completeness : No

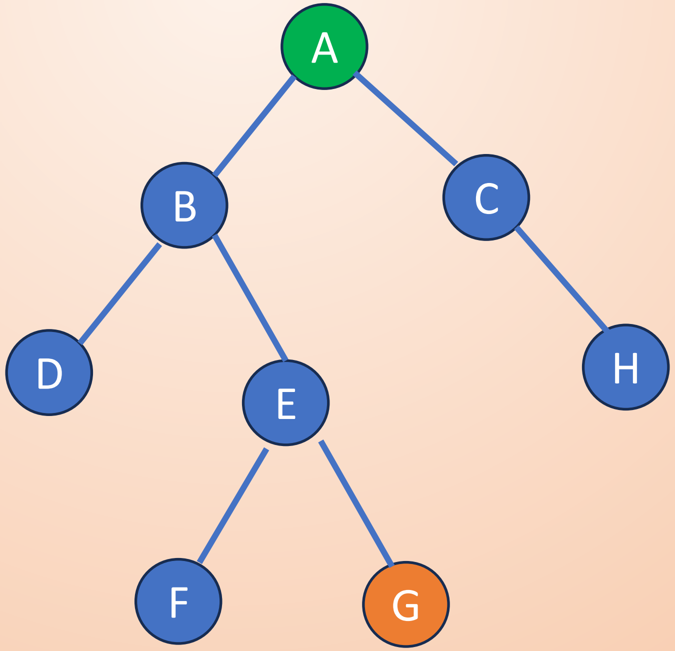
Optimality : No

Uninformed Search

Iterative Deepening Search

- Iterative Deepening Search(IDS) is a uninformed search technique used to explore all possible nodes in a search space systematically.
- The iterative deepening algorithm is a combination of DFS and BFS algorithms.
- This search algorithm finds out the best depth limit and does it by gradually increasing the limit until a goal is found.





Time Complexity : O(b^d)

Space Complexity : O(b.d)

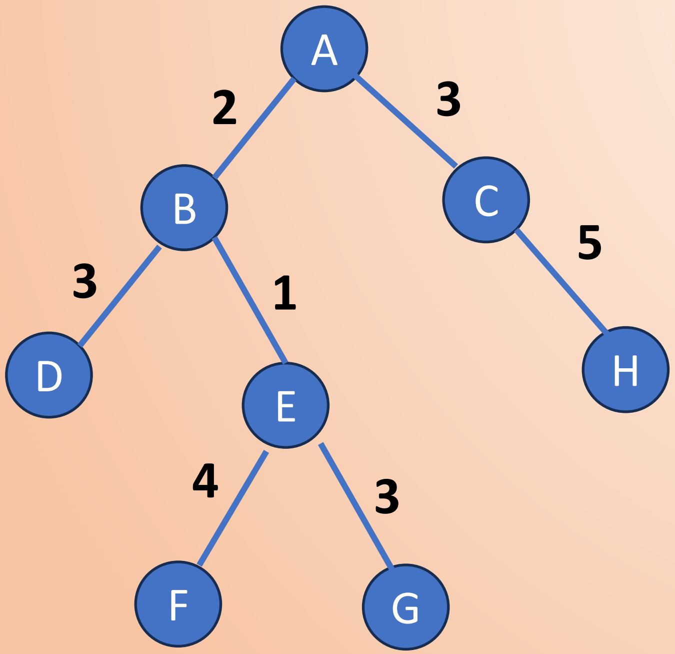
Completeness : Yes

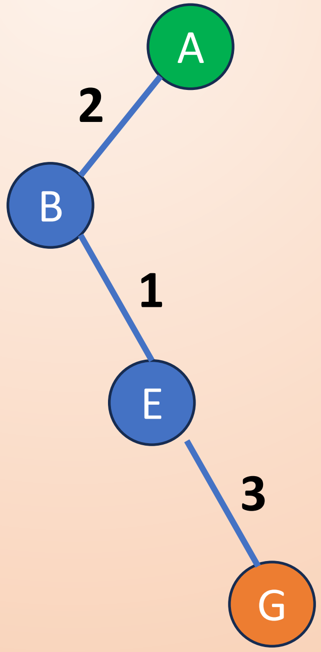
Optimality : Yes

Uninformed Search

Uniform Cost Search

- Uniform Cost Search(UCS) is a uninformed search technique used to explore all possible nodes in a search space systematically.
- Uniform-cost search is a searching algorithm used for traversing a weighted tree or graph.
- The primary goal of the uniform-cost search is to find a path to the goal node which has the lowest cumulative cost.





Time Complexity : $O(b^{1 + [C^*/\varepsilon]}$

Space Complexity : $O(b^{1 + [C^*/\varepsilon}$

Completeness : Yes

Optimality : Yes

Informed Search

Informeed Search

Informed search algorithms use heuristics or domain-specific knowledge to efficiently guide the search process toward the goal, making them faster and more focused than uninformed search.

Informed (Heuristic) Search

- Best First Search
- A* Search
- Iterative Deepening A*

Best First Search

Informed Search

- Best-First Search is a search algorithm that selects the node to expand based on a heuristic function, f(n) , which estimates the cost to reach the goal from node n.
- The node with the smallest heuristic value is expanded first. This can be implemented using a priority queue.

- Initialize the frontier with the start node.
- At each step, choose the node with the lowest heuristic value (h(n)) expand it, and add its neighbors to the frontier.
- Repeat until the goal is found or the frontier is empty

Time Complexity : O(b^d)

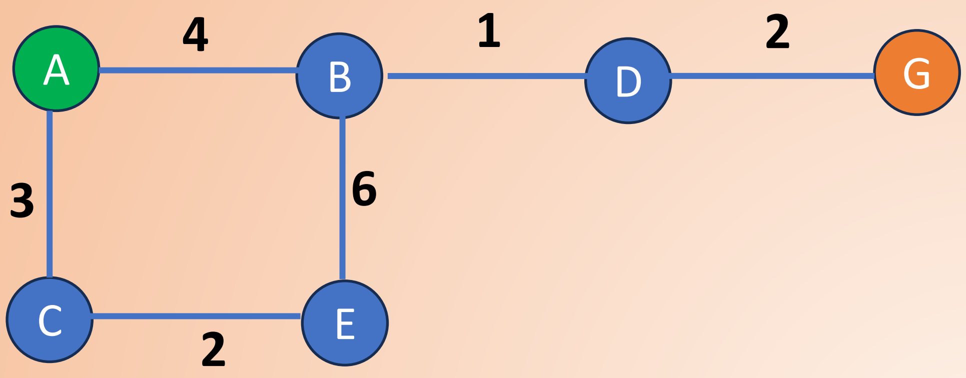
Space Complexity : O(b^d)

Completeness : No

Optimality : No

Informed Search

Best First Search



Node	h(n)
A	10
B	6
C	9
D	4
E	5
G	0

Start with node A:
h(A) = 10
Open set : {A}
Iteration 1:
Current Node: A
Neighbors: B, C ; h(B)=6, h(C) = 9
Open Set: {B,C}
Minimum is B
Iteration 2:
Current Node: B
Neighbors: A, E, D; h(E)=5,h(D)=4
Open Set: {C, E, D}
Minimum is D
Iteration 3 :
Current Node: D
Neighbors : B, G; h(G) = 0
Open set: {G,C,E}
Minimum: G
Iteration 4: Current Node: G
Goal reached .

Shortest: A->B->D->G

Informed Search

A* Search

- A* Search is one of the most commonly used and powerful informed search algorithms.
- It combines Best-First Search with Uniform Cost Search to find the least-cost path to the goal.
- A* uses a combined heuristic hash function/cost function f(n) = g(n)+h(n), where, g(n) is cost from the start node to node n, h(n) is heuristic estimate of the cost to reach goal from node n

- Initialize the frontier with the start node, with f(start) = g(start) + h(start).
- At each step, choose the node with the lowest f(n) value and expand it.
- For each neighbor, calculate f(n) and add it to the frontier if not already visited or if a better path is found.
- Repeat until the goal is reached.

Time Complexity : O(b^d)

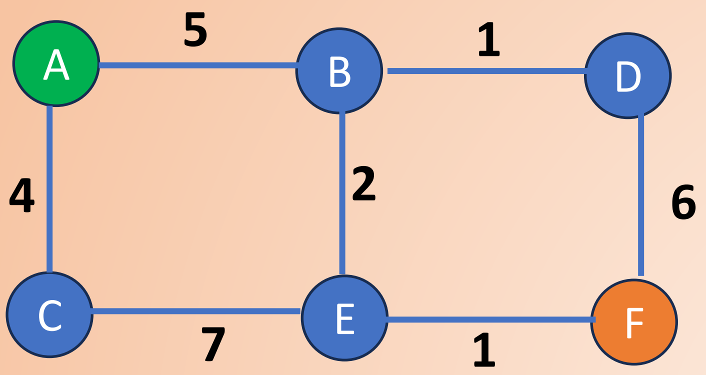
Space Complexity : O(b^d)

Completeness : Yes

Optimality : Yes

A* Search

Informed Search



Node	h(n)
A	7
B	4
C	6
D	3
E	1
F	0

Node	h(n)	g(n)	f(n)
A	7	0	7
A->B A->C	4 6	5 4	9 10
A->B->E A->B->D	1 3	7 6	8 9
A->B->E->F A->B->E->C	0 6	8 14	8 20

Cost function f(n) = g(n) + h (n)

Start with node A:

g(A) = 0 , h(A) = 7

f(A) = 0 + 7 = 7

Neighbors of A : B and C

For B:

g(B) = g (A) + c(A,B) = 0 + 5 = 5

f(B)= 5 + 4 = 9

For C:

g(c) = g(A) + c(A,C) = 0 + 4 = 4

f(C) = 4 + 6 = 10

f(C)>f(B) , So choose A->B path

Current Node B:

Neighbors : D, E

For D

g(D) = 6, h(D) = 3

f(D) = 6 + 3 = 9

For E

g(E) = 7, h(E) = 1

f(E) = 7 + 1 = 8

Shortest: A->B->E->F

Iterative Deepening A*

- Iterative Deepening A* is a variant of A* that uses depth-limited search and iteratively increases the depth limit, using the heuristic to prune the search.
- This technique is used to avoid memory limitations that A* might face in large search spaces.

- Start with a depth limit L = 0.
- Perform a depth-first search with A* within this limit.
- If no solution is found, increase the limit and repeat the process.

Time Complexity : O(b^d)

Space Complexity : O(bd)

Completeness : Yes

Optimality : Yes

Quote of the Day

Life is not a problem to be solved
Life is reality to be experienced

Thank You