ARTIFICIAL INTELLIGENCE
&
MACHINE LEARNING

SESSION NO : 06

Uninformed Search

- It is a search algorithm that explores a problem space without any specific knowledge or information about the problem other than the initial state and the possible actions to take.
- It lacks domain-specific heuristics or prior knowledge about the problem.

1. Uniform Cost Search
2. Iterative Deepening Search

1. Uniform Cost Search

- Uniform Cost Search (UCS) is an uninformed search algorithm in Artificial Intelligence used to find the least-cost path from a starting node to a goal node in a weighted graph.
- It explores nodes based on their cumulative cost from the start, using a priority queue to prioritize nodes with lower costs.
- Uniform-Cost Search is a variation of Dijikstra's algorithm.

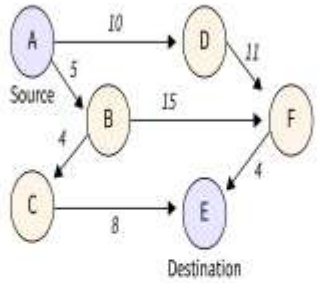
Example

- We will have an empty priority queue and a boolean visited array. We will insert A with cost 0 into the queue.

A - 0					

visited:

A	B	C	D	E	F
False	False	False	False	False	False



SCALER Topics

Example

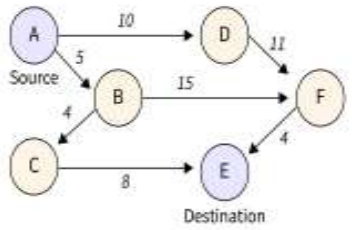
Step 1:

Pop A from the queue.A is **not** destination node.

Cost of A = 0, cst = 0.

Append B and D to the queue with the costs cst + 10 and cst + 5.

And mark A as visited.



SCALER Topics

Queue:

B - 5	D - 10				

visited:

A	B	C	D	E	F
True	False	False	False	False	False

Example

Step 2:

Pop B from the Queue. As the cost of B is less than D, it appears first in the queue.B is **not** the destination node.

Cost of B = 5, cst = 5.We will append C with cost 4 + cst and F with cost 15 + cst to the queue.Mark B as visited.

Queue:

C - 9	D - 10	F - 20			

A	B	C	D	E	F
True	True	False	False	False	False

Example

Step 2:

Pop B from the Queue. As the cost of B is less than D, it appears first in the queue.B is **not** the destination node.

Cost of B = 5, cst = 5.We will append C with cost 4 + cst and F with cost 15 + cst to the queue.Mark B as visited.

Queue:

C - 9	D - 10	F - 20			

A	B	C	D	E	F
True	True	False	False	False	False

Example

Step 3:

Similarly, C is popped, and E is pushed in the stack.

Similarly, D is popped, and F is pushed into the stack as it is still not visited.

Queue:

E - 17	F - 20	F - 21			

visited:

A	B	C	D	E	F
True	True	True	True	False	False

Example

Step 4:

Pop E from the Queue. As the cost of E is least it appears first in the queue.E is the destination node.

Cost of E = 17, cst = 17.

The initial value of min_cost is Infinity or Integer.MAX_VALUE. Thus, min_cost = cst = 17 is updated.

Queue:

F - 20	F - 21				

visited:

A	B	C	D	E	F
True	True	True	True	True	False

Example

Step 5:

Pop F from the Queue.F is **not** the destination node.

Cost of F = 20, cst = 20.

F has the outgoing edge to E. We will push it to the queue with the cost 4 + cst.Mark F as visited.

Queue:

F - 21	E - 24				

visited:

A	B	C	D	E	F
True	True	True	True	False	True

Example

Step 6:

Pop F from the Queue.F is **not** the destination node.

Cost of F = 21, cst = 21.

F has the outgoing edge to E. We will push it to the queue with the cost 4 + cst.Mark F as visited.

Queue:

E - 24	E - 25				

visited:

A	B	C	D	E	F
True	True	True	True	False	True

Example

Step 7:

Pop E from the Queue.

E is the destination node.

Cost of E = 24 cst = 24.

The min_cost = 17 and the current cost is 24. Thus, we are not required to update any values.

Again Pop E from the Queue.

The min_cost = 17 and the current cost is 25. Thus, we are not required to update any values.

Algorithm

- **Initialization:** A priority queue (frontier) is created, initialized with the starting node and its cost (usually 0).
- **Node Expansion:** The node with the lowest cost from the frontier is removed and expanded.
- **Neighbor Exploration:** The algorithm examines the neighboring nodes of the expanded node.
- **Cost Calculation:** For each neighbor, the cost to reach it from the starting node is calculated.

Cont’d

Frontier Update: If a neighbor's cost is lower than its previously recorded cost, or if it's not yet in the frontier, it's added to the frontier with the new cost.

Goal Check: If the expanded node is the goal node, the algorithm terminates and returns the path.

Iteration: Steps 2-6 are repeated until the goal node is reached or the frontier is empty.

Cont’d

Advantages:

- Guarantees finding the optimal path (least-cost path).
- Suitable for cost-sensitive problems.
- Complete, meaning it will find a solution if one exists.

Disadvantages:

- Can be computationally expensive, especially in large state spaces, due to the need to explore nodes based on cost.
- Requires storing all generated nodes in the priority queue, potentially leading to high memory usage.

Cont’d

Uninformed:

It doesn't use any knowledge about the goal or the structure of the search space beyond the graph's edges and their costs.

Optimality:

If a solution exists, UCS guarantees finding the least-cost path (optimal path).

Completeness:

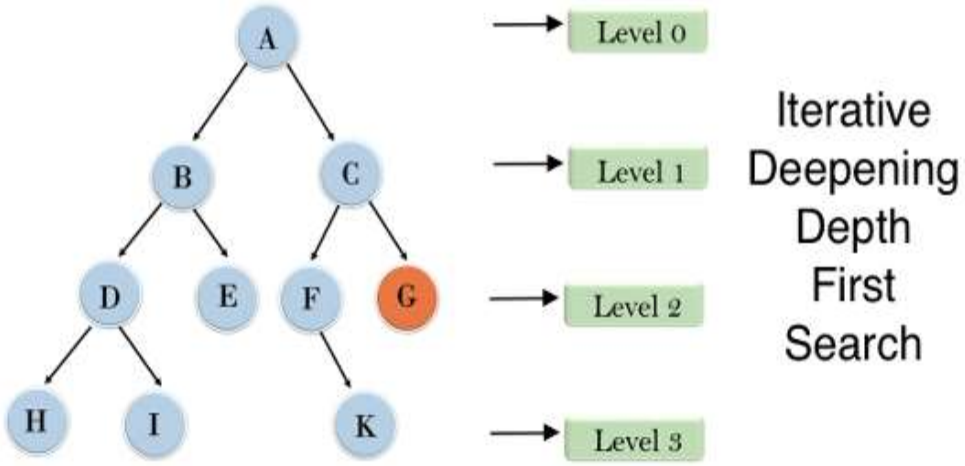
It will find a solution if one exists.

Iterative Deepening Depth First Search (IDDFS)

- It is a search algorithm that uses the combined power of the BFS and DFS algorithms.
- It is iterative in nature. Searches for the best depth in each iteration. It performs the Algorithm until it reaches the goal node.
- The algorithm is set to search until a certain depth and the depth keeps increasing at every iteration until it reaches the goal state.

Iterative Deepening Depth First Search (IDDFS)

- Iteration 1. Depth=0 , [A]
- Iteration 2. Depth=1 , [A,B,C]
- Iteration 3. Depth=2 , [A,B,C,D,E,F,G]



IDDFS-Algorithm

```
function IterativeDeepeningSearch(start, goal):
    for depth from 0 to ∞:
        result ← DepthLimitedSearch(start, goal, depth)
        if result ≠ failure:
            return result
```

Iterative Deepening Depth First Search

Advantages

It combines the benefits of BFS and DFS search algorithms in artificial intelligence in terms of fast search and memory efficiency.

Disadvantages

The main drawback of IDDFS is that it repeats all the work from the previous phase.

Time Complexity: $O(b^d)$

Space complexity: $O(b^d)$

Conclusion

- The term ‘uninformed’ means that they do not have any additional information about states or state space.
- “uninformed algorithm” is an algorithm that doesn’t use any prior knowledge or heuristics to solve a problem.

Objective Question

1.Which of the following uninformed search strategies always expands the shallowest unexpanded node first?

A) Depth-First Search

B) Breadth-First Search

C) Uniform Cost Search

D) Iterative Deepening Search

2.Which search strategy can get stuck in an infinite branch of the search tree?

A) Uniform Cost Search

B) Breadth-First Search

C) Depth-First Search

D) Iterative Deepening Search

3.Which uninformed search strategy combines the benefits of both depth-first and breadth-first search?

A) Best-First Search

B) Uniform Cost Search

C) Iterative Deepening Search

D) Random Walk

Objective Question

1.Which of the following uninformed search strategies always expands the shallowest unexpanded node first?

A) Depth-First Search

B) Breadth-First Search

C) Uniform Cost Search

D) Iterative Deepening Search

2.Which search strategy can get stuck in an infinite branch of the search tree?

A) Uniform Cost Search

B) Breadth-First Search

C) Depth-First Search

D) Iterative Deepening Search

3.Which uninformed search strategy combines the benefits of both depth-first and breadth-first search?

A) Best-First Search

B) Uniform Cost Search

C) Iterative Deepening Search

D) Random Walk

Question for Practice

1. What is Iterative Deepening Search, and why is it considered a good compromise between DFS and BFS?

2. Discuss the completeness, optimality, time complexity, and space complexity of any two uninformed search algorithms.

3. What are the limitations of uninformed search algorithms in problem-solving? Discuss with examples.