Artificial & Computational Intelligence

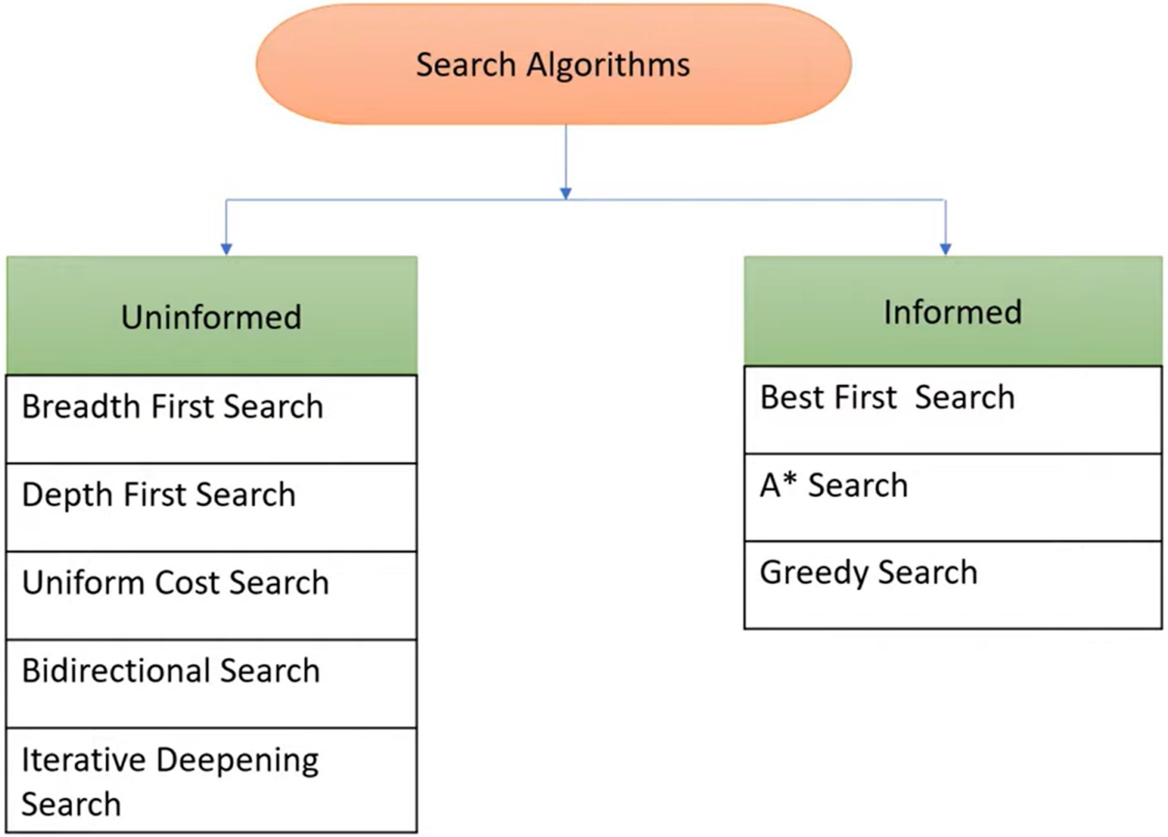
DSE **ZG557

Uninformed Search

BITS Pilani
Pilani Campus

Uninformed Search

Types of Search Algorithms



Search Algorithms

Uninformed

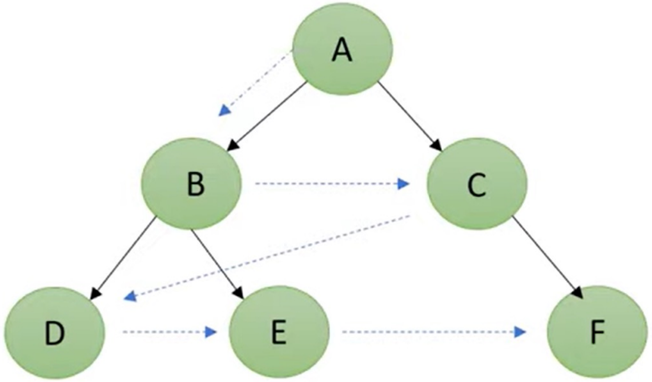
- Breadth First Search
- Depth First Search
- Uniform Cost Search
- Bidirectional Search
- Iterative Deepening Search

Informed

- Best First Search
- A* Search
- Greedy Search

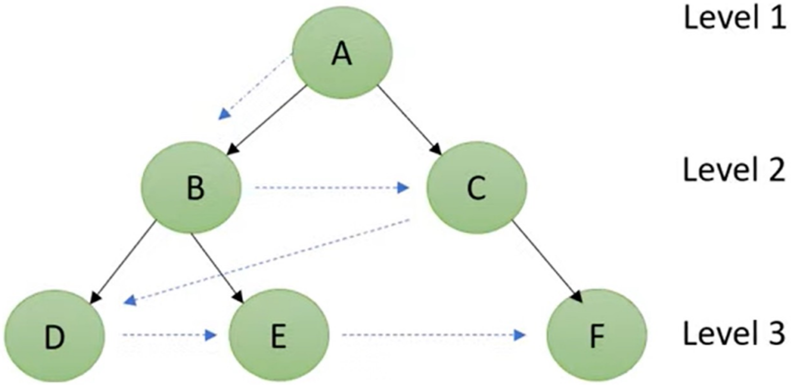
Uninformed Search

- Uninformed search algorithms have no additional information on the goal node other than the one provided in the problem definition.
- Uninformed search is also called blind search



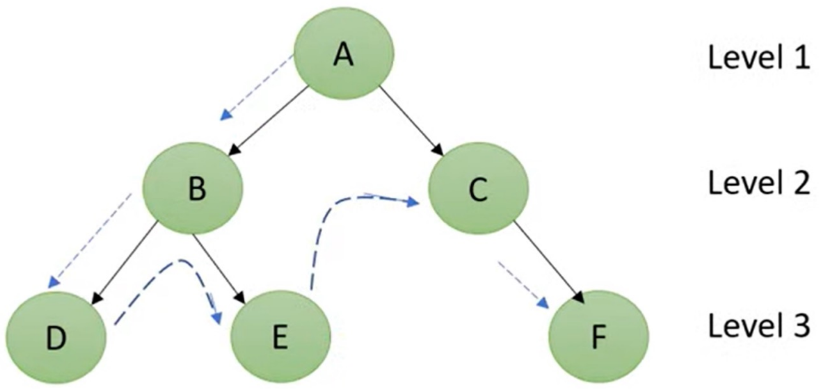
Breadth First Search

- In breadth-first search, the tree or the graph is traversed breadthwise
- It starts from the root node, explores the neighboring nodes first and moves towards the next level neighbors.
- It is implemented using the queue data structure that works on the concept of first in first out (FIFO).



Depth First Search

- Depth first search (DFS) algorithm starts with the initial node and then goes to deeper and deeper until we find the goal node or the node which has no children.
- The algorithm, then backtracks from the dead end towards the most recent node that is yet to be completely unexplored.
- DFS uses a stack data structure for its implementation.



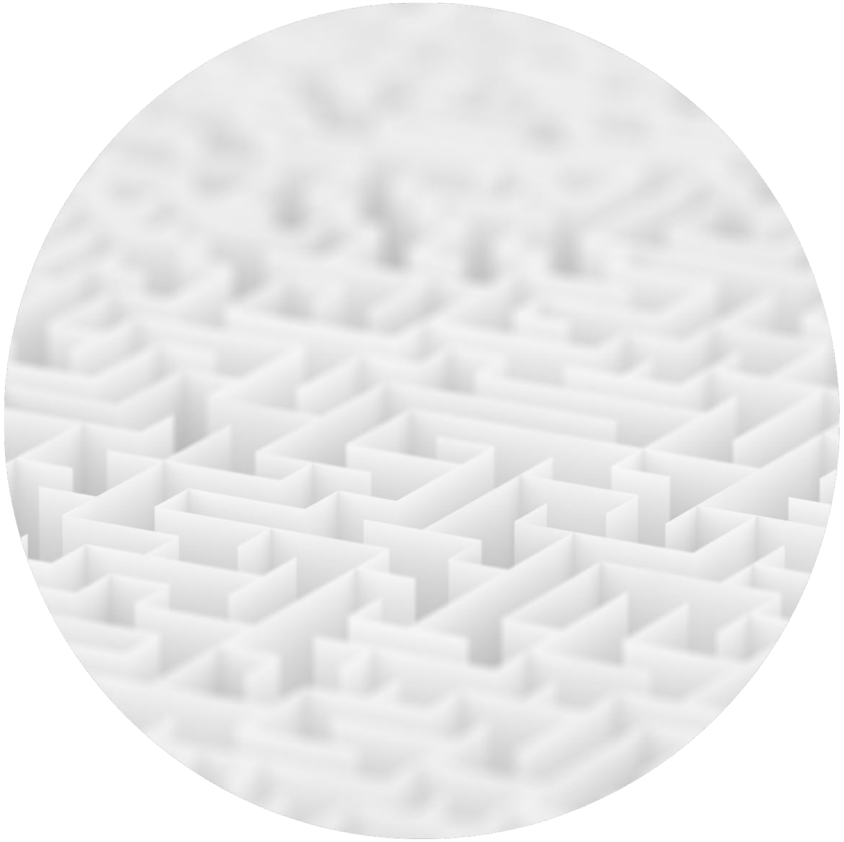
Problem Formulation

A binary maze is an n*m matrix where,

maze[i][j] = 1 represent traversable blocks
maze[i][j] = 0 represent obstacles.

Given the binary maze with obstacles and traversable blocks as represented in the next slide, find the optimal path between a source cell and destination cell.

Permissible moves are north, south, east and west (up, down, right, left)



Problem Formulation

Input : Binary matrix, source indices, destination indices

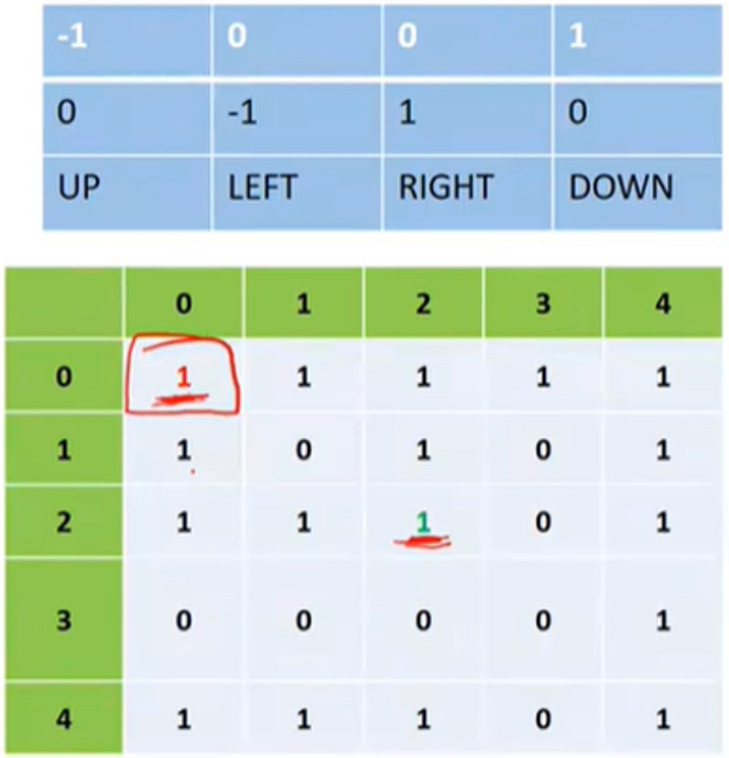
Input

Binary Matrix:

```
inputMaze=      [[1, 1, 1, 1, 1, 1, 1, 1],
                  [1, 1, 1, 1, 1, 1, 1, 1],
                  [1, 1, 1, 1, 1, 1, 1, 1],
                  [1, 1, 0, 0, 0, 0, 0, 1],
                  [1, 0, 1, 1, 1, 1, 1, 1],
                  [1, 1, 0, 1, 1, 1, 0, 1],
                  [1, 1, 1, 1, 1, 1, 1, 1]]
src = [0, 0]
dest = [2, 2]
```

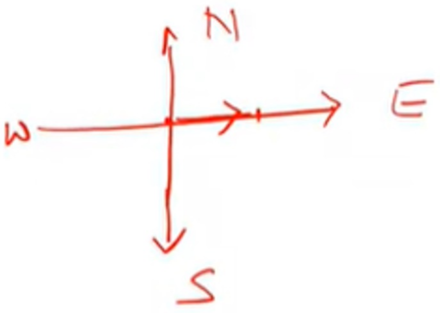
Logic / Search Technique: DFS/BFS

Output : List of tuples representing path from source from destination



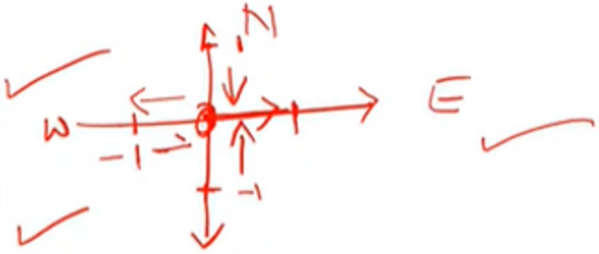
-1	0	0	1
0	-1	1	0
UP	LEFT	RIGHT	DOWN

	0	1	2	3	4
0	1	1	1	1	1
1	1	0	1	0	1
2	1	1	1	0	1
3	0	0	0	0	1
4	1	1	1	0	1



0,0
1 0
2,0
2,1
2,2

-1	0	0	1
0	-1	1	0
UP	LEFT	RIGHT	DOWN

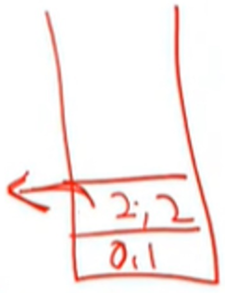


[F F F]

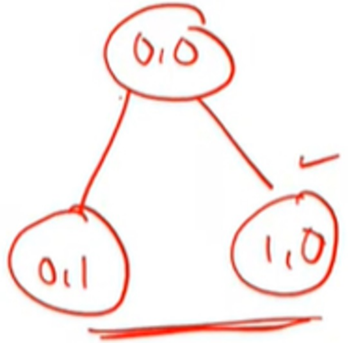
	0	1	2	3	4
0	1 X	1	1	1	1
1	1	0 X	1	0	1
2	1	1	1	0	1
3	0 X	0 X	0	0	1
4	1	1	1	0	1

Stack S (2,2)

Visited
(0,0) ✓
(1,0) ✓
(2,0) ✓
(2,1) ✓
2,2)



Goal



1 → path
0 — Block

-1	0	0	1
0	-1	1	0
UP	LEFT	RIGHT	DOWN

	0	1	2	3	4
0	1	1	1	1	1
1	1	0	1	0	1
2	1	1	1	0	1
3	0	0	0	0	1
4	1	1	1	0	1