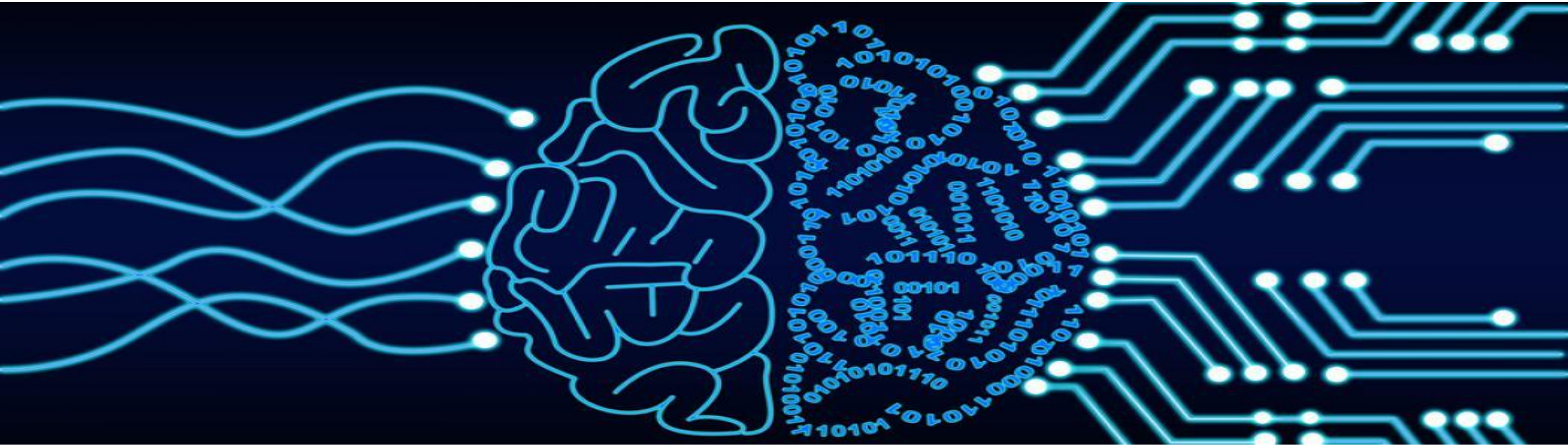


Artificial Intelligence (CS-401)

Chapter 3: Problem Solving Agent

Objectives of the Chapter

- Problem Solving Agents
- Problem Formulation
- Search Strategies
- Informed & Un-informed

Problem Solving Agent

Searching is one of the classic areas of AI

Problem Solving agents are one kind of goal-based agent that acts Rationally.

Problem Solving Agent

An agent that tries to come up with a sequence of actions by performing some kind of search algorithm in the background that will bring the environment into a desired state/goal.

Search

The use of search requires an abstract formulation of the problem and the available steps to construct solutions.

Problem Formulation

1. State
2. Initial State
3. Actions
4. Goal Test
5. Path Cost
6. Solution

A State Space
Set of all possible states where you can be.

A Start State. The state from where the search begins.

A Goal Test. A function that looks at the current state returns whether or not it is the goal state.

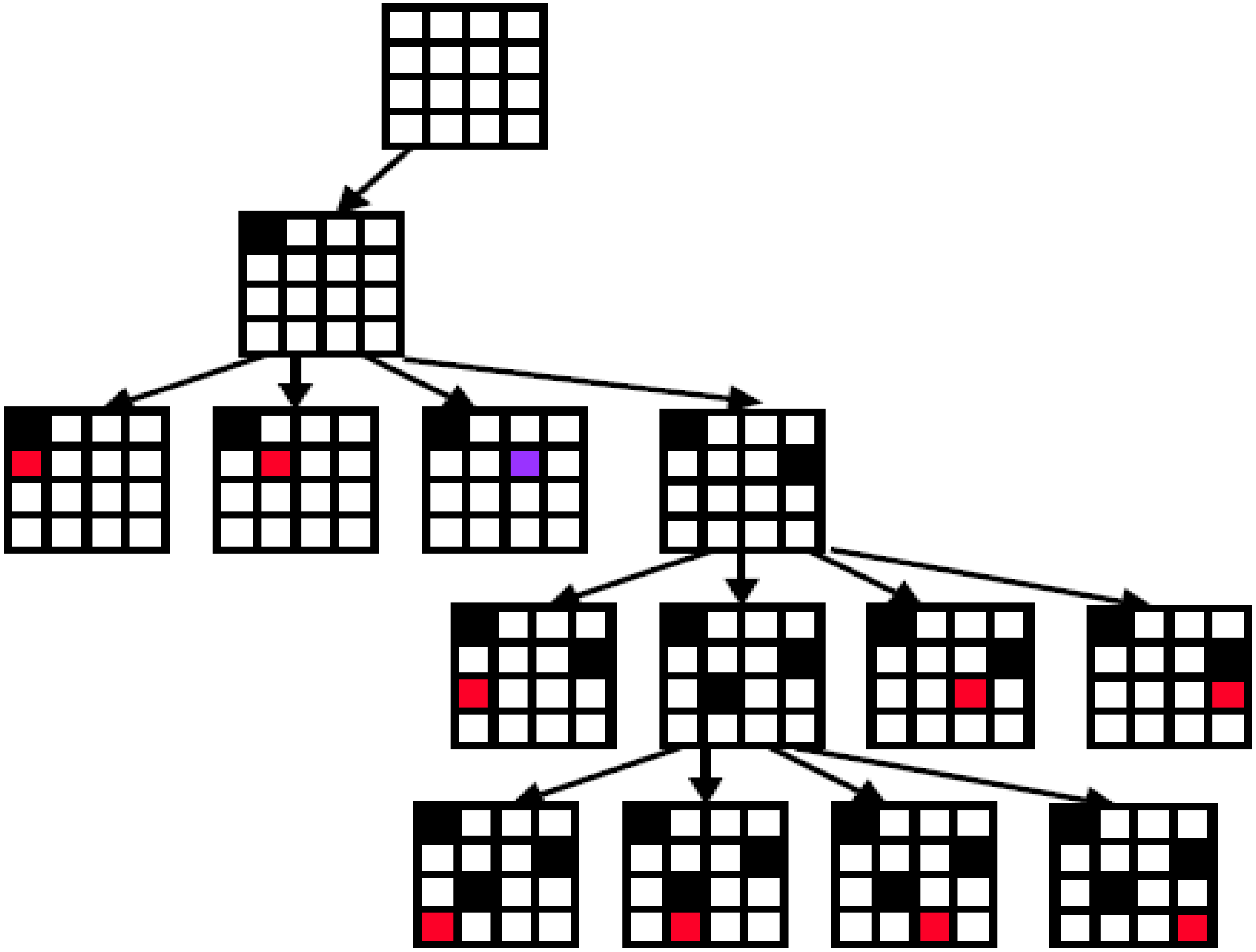
Solution to a search problem is a sequence of actions, called the **plan** that transforms the start state to the goal state.

THIS PLAN IS ACHIEVED THROUGH SEARCH ALGORITHMS

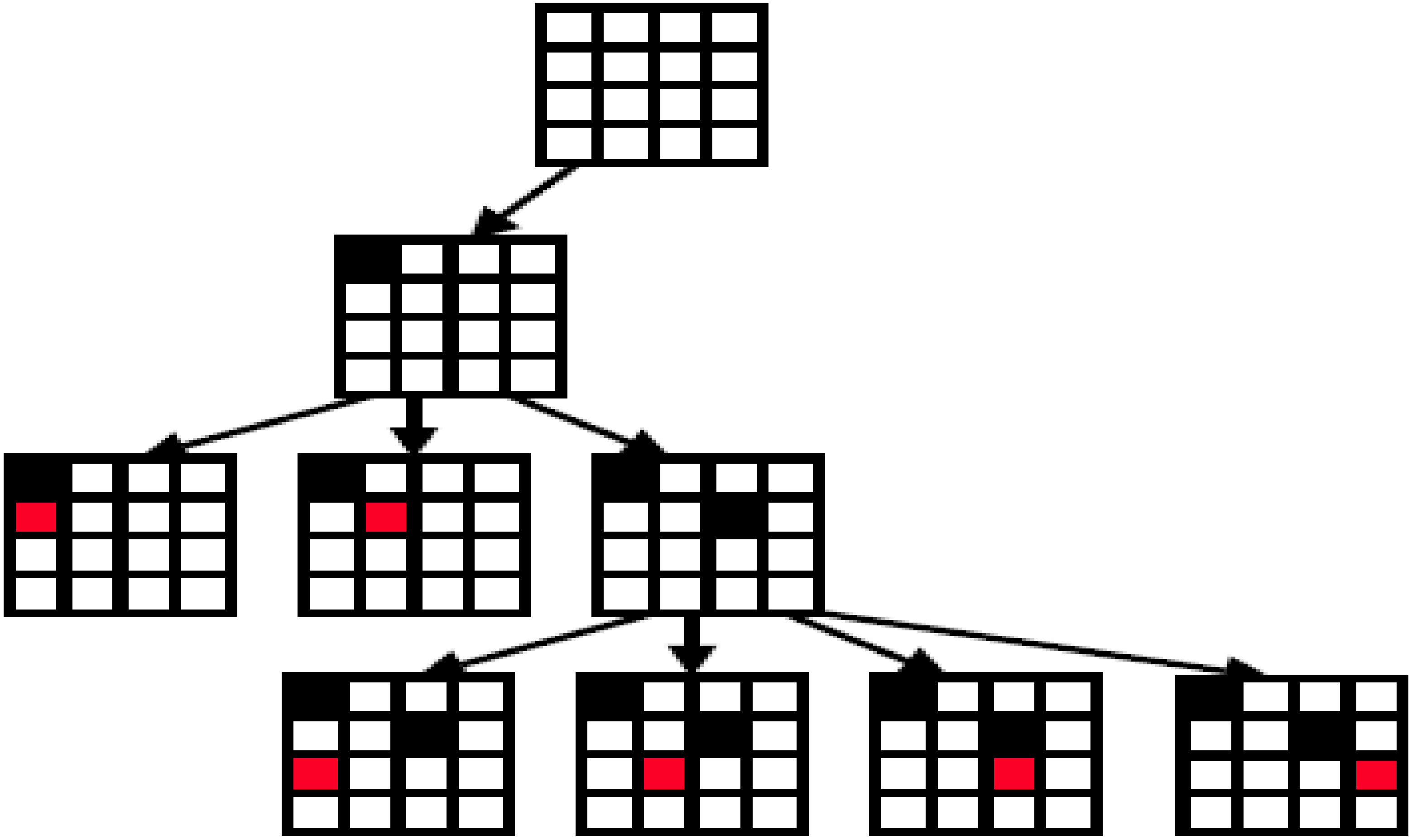
NOTE

Solution Quality is measured by the Path cost function, and an Optimal solution has the lowest path cost among all solutions

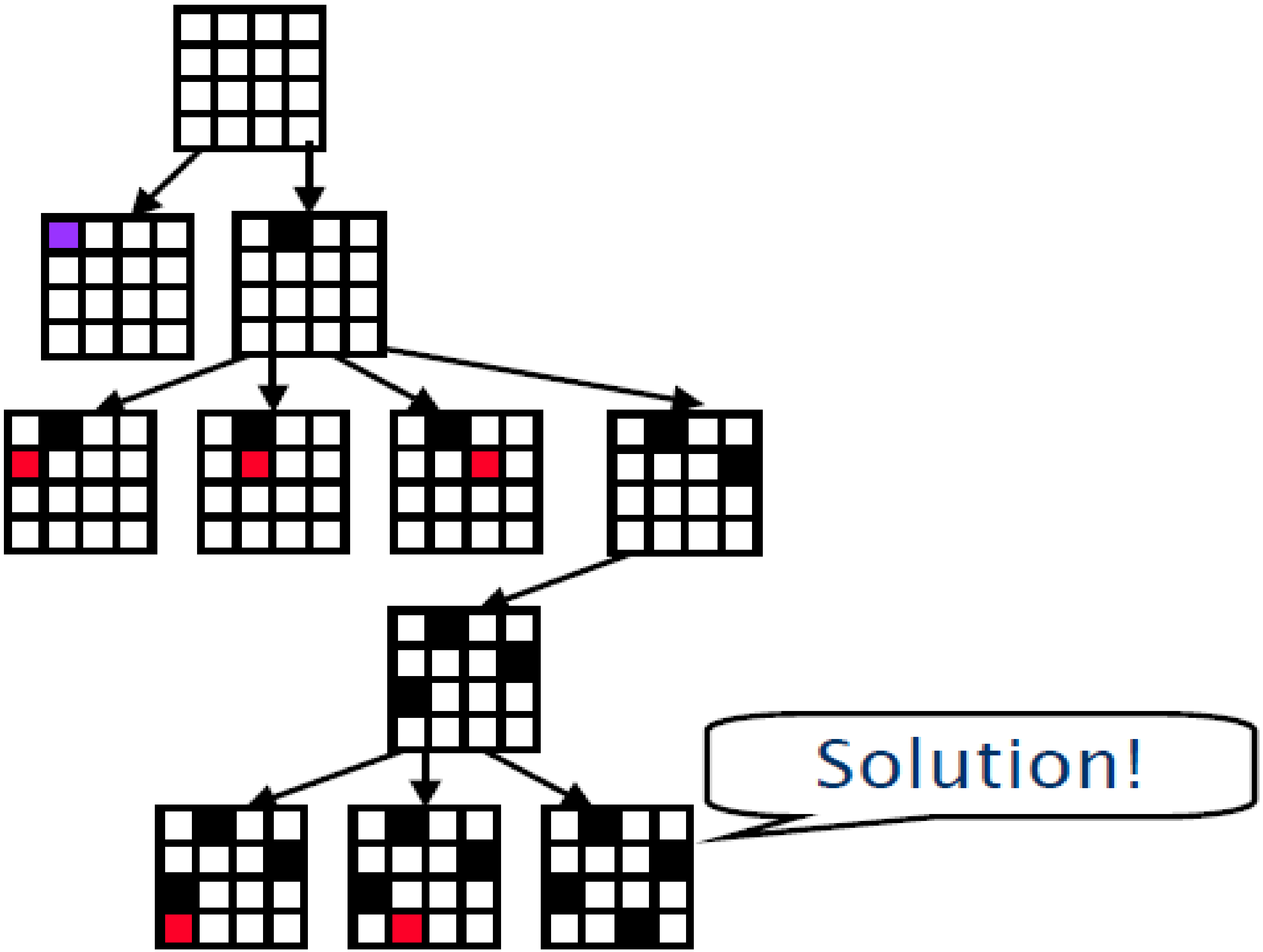
4 Queens Problem Example



4 Queens Problem Example........



4 Queens Problem Example........



The 8- Puzzle Example

State: The location of the eight tiles, and the blank

Initial State: {(2,0), (8,1), (3,2), (1,3), (6,4), (4,5), (7,6), (-,7), (5,8)}

2	8	3
1	6	4
7	-	5

Initial State

1	2	3
8	-	4
7	6	5

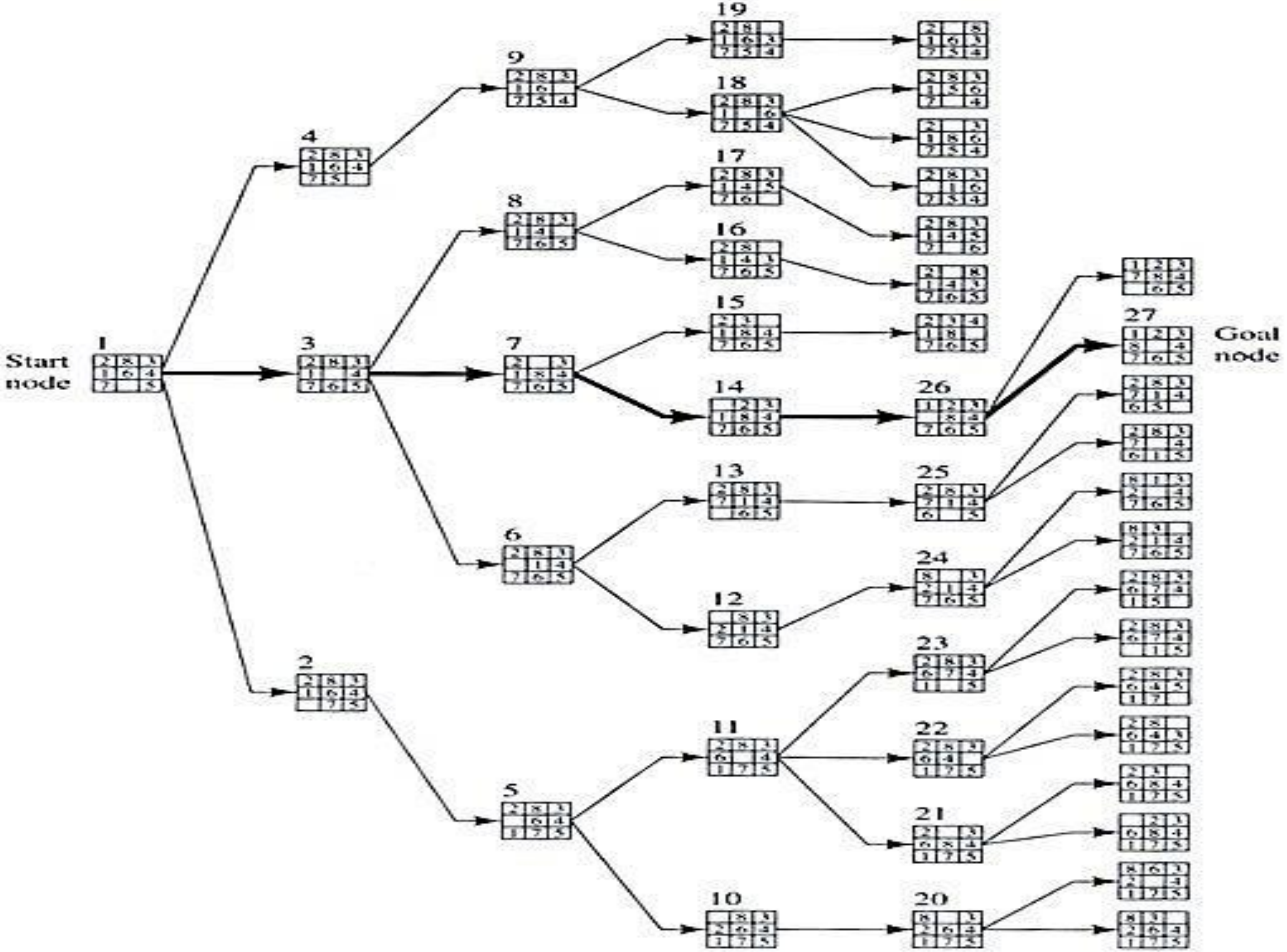
Goal State

Successor Function: four actions (blank moves Left, Right, Up, Down).

Goal Test: determine a given state is a goal state.

Path Cost: each step costs 1

Solution: {(1,0),(2,1),(3,2) ,(8,3) ,(-,4) ,(4,5) ,(7,6) ,(6,7) ,(5,8)}


Start node
Goal node
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27

2	8	3
1	6	4
7	-	5

2	8	3
1	-	4
7	6	5

2	-	3
1	8	4
7	6	5

-	2	3
1	8	4
7	6	5

1	2	3
-	8	4
7	6	5

1	2	3
8	-	4
7	6	5

1	2	3
8	-	4
7	6	5

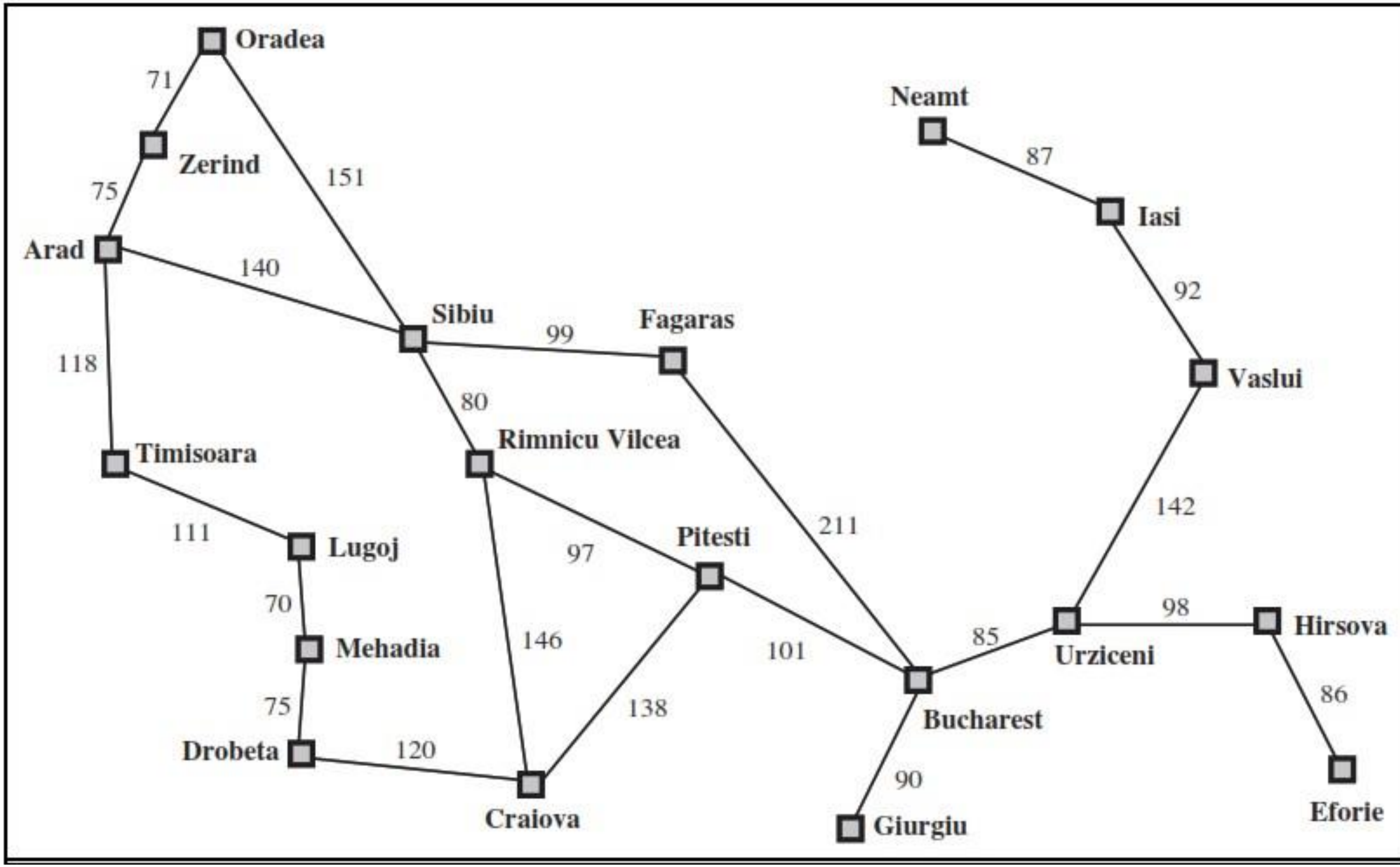
Goal

A Touring Agent Problem

(SCENARIO)

Imagine an agent in the city of Arad, Romania, enjoying a touring holiday. now, suppose the agent has a non-refundable ticket to fly out of Bucharest the following day. In that case, it makes sense for the agent to adopt the **Goal** of getting to Bucharest

Problem Formulation (The Romania Example)



Solution: a sequence of actions leading from the initial state to a goal state

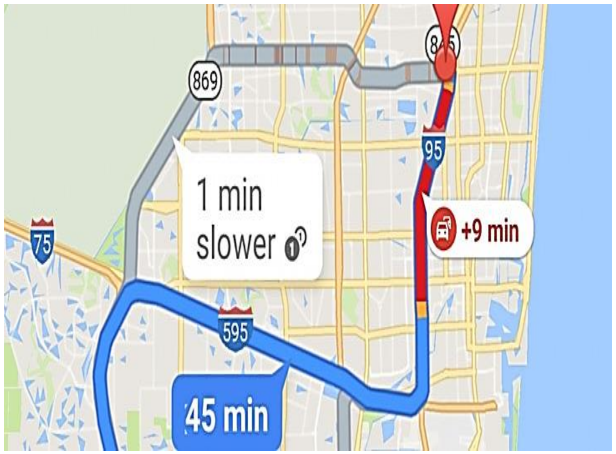
{Arad →Sibiu → Rimnicu Vilcea → Pitesti → Bucharest}

Problems

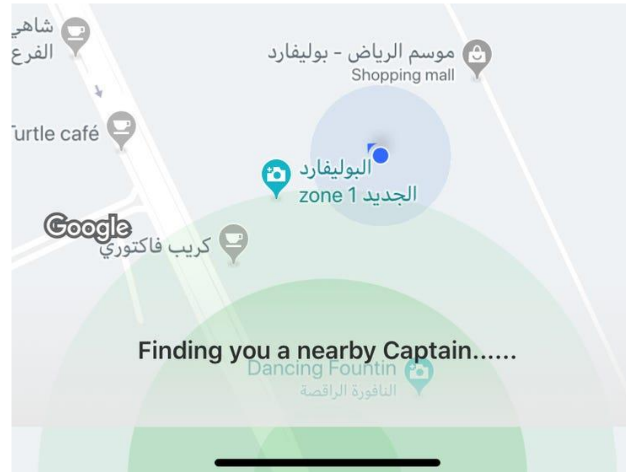
- Vacuum Cleaner
- 8-Puzzle
- 8-Queens Problem
- Robotic Assembly
- VLSI Layout

Problem Formulation (Real-life Applications)

Car Navigation



Route Finding Problem



Nearby Captain

- States
  - locations
- Initial state
  - starting point
- Successor function (operators)
  - move from one location to another
- Goal test
  - arrive at a certain location
- Path cost
  - may be quite complex
    - money, time, travel comfort, scenery,

Examples:

- Military Operation Planning
- Train Travel Planning
- Airline Travel Planning
- Routing in Computer Networks

Search Algorithms

Uninformed Search

Informed Search

Uninformed Search

– breadth-first
– uniform-cost search
– depth-first
– depth-limited search
– iterative deepening
– bi-directional search

(blind search)

- number of steps, path cost unknown
- agent knows when it reaches a goal

Informed Search

– Greedy best-first
– A*

(heuristic search)

- agent has background information about the problem

Uninformed (Blind Search)

- The Uninformed Search does not contain any domain knowledge such as closeness or location of goal.

- It operates in a brute force way, as it only includes information about how to traverse the tree and how to identify leaf and goal nodes.

- Uninformed Search applies a way in which search tree is searched without any information, so it is called blind search.

- It examines each node until it achieves the goal node.

Evaluation of Search Strategies

A search strategy is defined by picking the order of node expansion

Strategies are evaluated along the following dimensions:

- Completeness: if there is a solution, will it be found
- Time complexity: How long does it takes to find the solution
- Space complexity: memory required for the search
- Optimality: will the best solution be found

Time and space complexity are measured in terms of

- *b:* maximum branching factor of the search tree
- *d:* depth of the least-cost solution
- *m*: maximum depth of the state space (may be ∞)

1. Breadth-First Search (BFS) Algorithm

Breadth-First Search

- It is the most common search strategy for traversing a tree or graph.

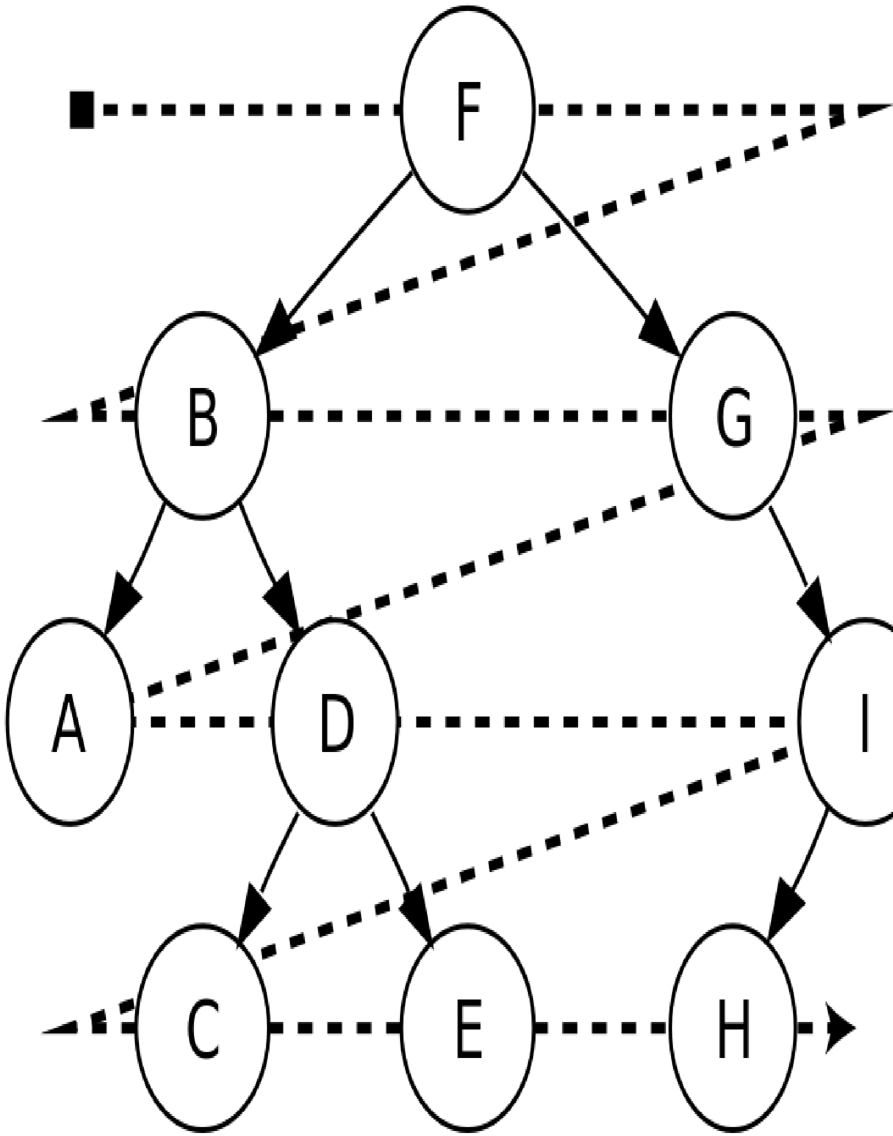
- This algorithm searches breadthwise in a tree or graph, so it is called breadth-first search.

- BFS algorithm starts searching from the root node of the tree and further are traversed in *level-order*, where we visit every node on a level before going to a lower level.

- It starts at the tree root (or some arbitrary node of a graph)

- BFS is implemented using FIFO Queue data structure.

Example: BFS

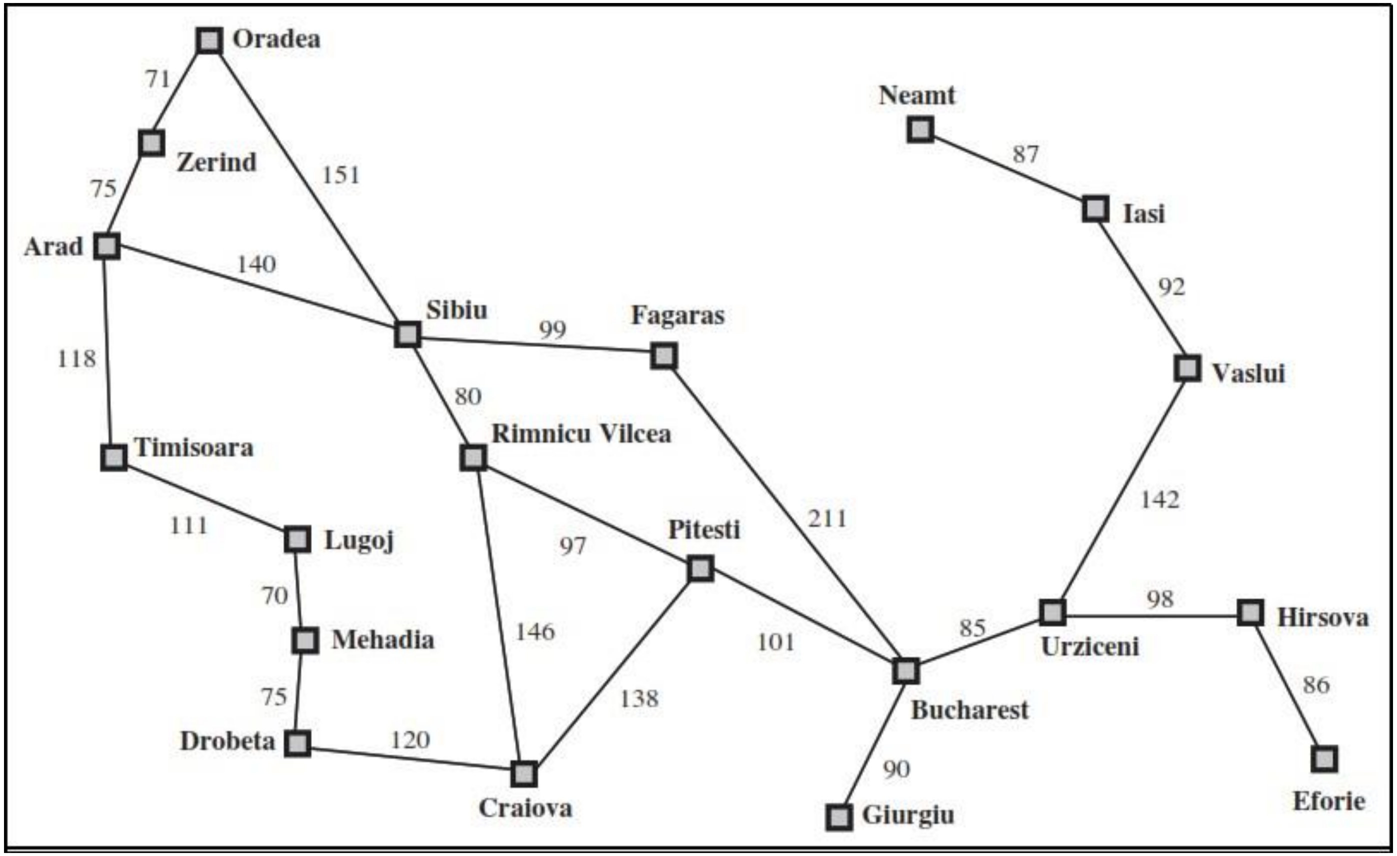


For example, after searching F, then B, then G, the search proceeds with A,D, I, C, E,H

Node are explored in the Level order F,B,G,A,D,I,C,E,H

Example: Romania

You are in Arad and want to go to Bucharest



➔ How to design an intelligent agent to find the way between 2 cities?

Properties of Breadth-First Search (BFS)

Completeness: Yes (if b is finite), a solution will be found if exists.

Time Complexity: (nodes until the solution)

Optimality: Yes

Criterion	Breadth-First
Complete?	Yes
Time	$O(b^{d+1})$
Space	$O(b^{d+1})$
Optimal?	Yes

b	Branching Factor
d	The depth of the goal

Suppose the branching factor b=3, and the goal is at depth d=20:

– Then we need 3^{20} time to finish.

➔Not suitable for searching large graphs

2- Uniform-Cost -First

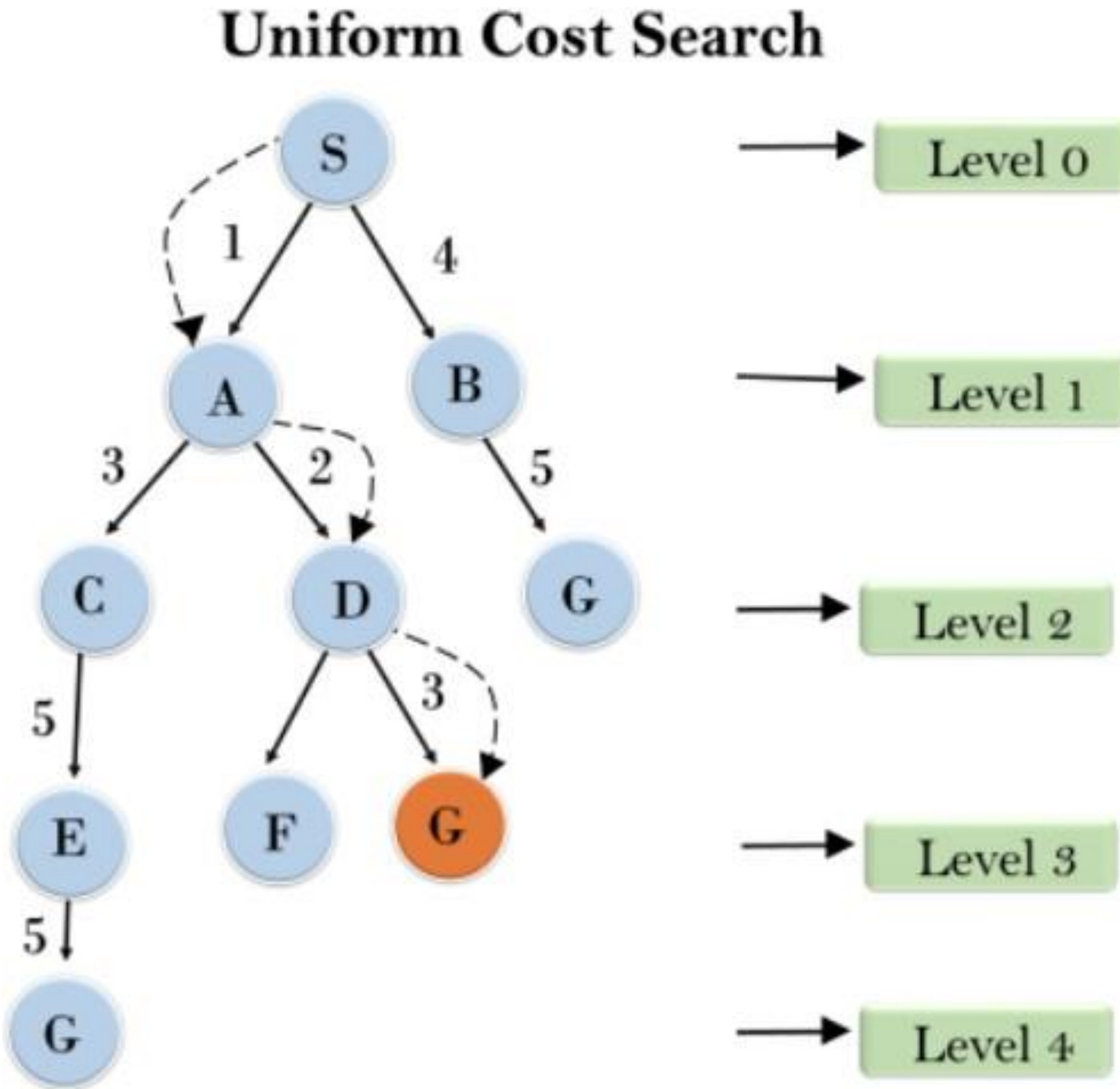
Uniform-Cost -First

- The primary goal of the uniform-cost search is to find a path to the goal node which has the lowest cumulative cost.

  Uniform-cost search expands nodes according to their path costs form the root node.

  It can be used to solve any graph/tree where the optimal cost is in demand.

  Uniform cost search is equivalent to BFS algorithm if the path cost of all edges is the same.


Uniform Cost Search
S
1
4
A
B
3
2
5
C
D
G
5
3
E
F
G
5
G
Level 0
Level 1
Level 2
Level 3
Level 4

Properties of Uniform-cost Search (UCS)

Completeness Uniform-cost search is complete, such as if there is a solution, UCS will find it.

Time Complexity much larger than $\mathbf{b^d}$, and just $\mathbf{b^d}$ if all steps have the same cost.

Space Complexity: as above

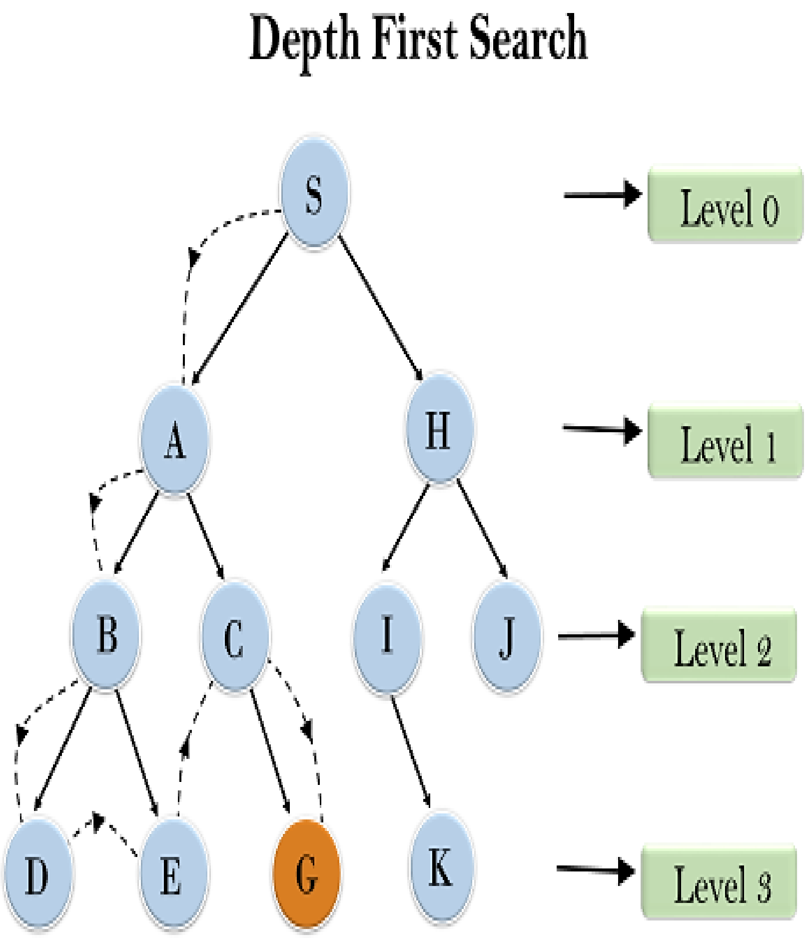
Optimality: Uniform-cost search is always optimal as it only selects a path with the lowest path cost.

Criterion	Uniform-Cost
Complete?	Yes
Time	$O(b^{\lceil C^*/\epsilon \rceil})$
Space	$O(b^{\lceil C^*/\epsilon \rceil})$
Optimal?	Yes

3- Depth-First Search

Depth-First Search

Based on[4]



It will start searching from root node S, and traverse A, then B, then D and E, after traversing E, it will backtrack the tree as E has no other successor and still goal node is not found.

After backtracking it will traverse node C and then G, and here it will terminate as it found goal node.

Properties of Depth-First Search

Complete: No: fails in infinite-depth spaces, spaces with loops
– Yes, complete in finite spaces

Time: $O(b^m)$: terrible if m is much larger than d

– but if solutions are dense, may be much faster than breadth-first

Space: $O(bm)$

Optimal: No

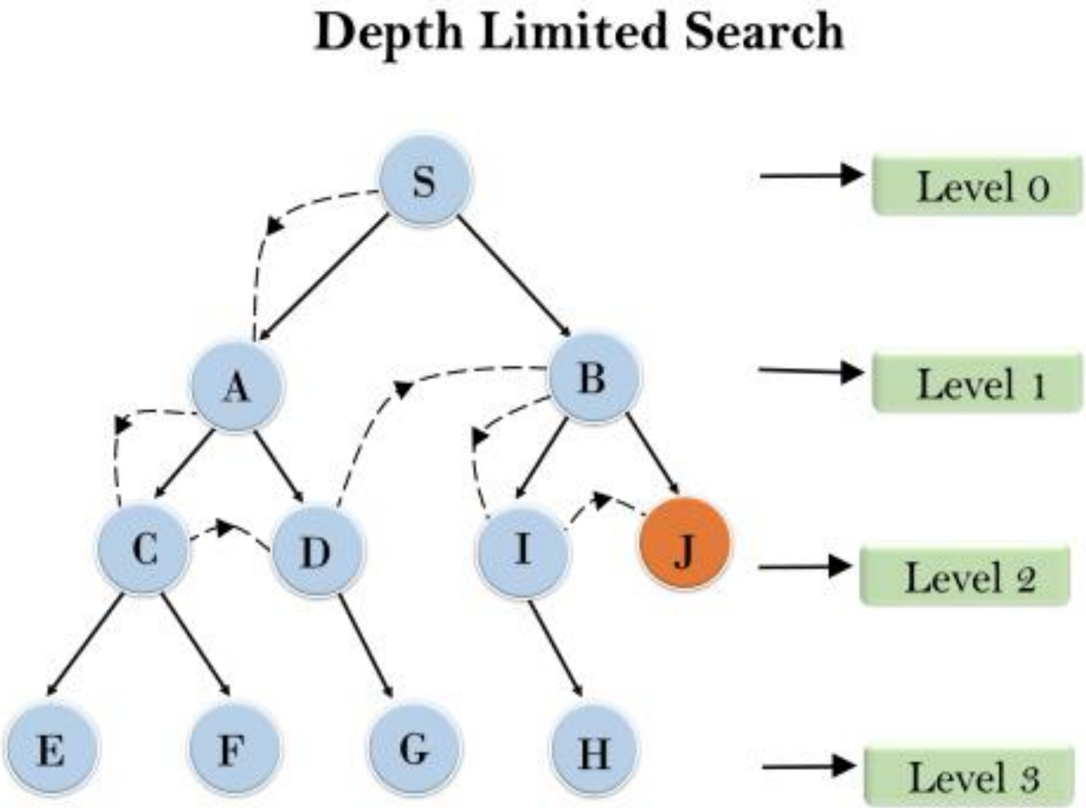
Criterion	Depth-First
Complete?	No
Time	$O(b^m)$
Space	$O(bm)$
Optimal?	No

4- Depth-Limited Search

4- Depth-Limited Search

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.
In this algorithm, the node at the depth limit will treat as it has no successor nodes further.
Depth-limited search can be terminated with two Conditions of failure:
Standard failure value: It indicates that problem does not have any solution.
Cutoff failure value: It defines no solution for the problem within a given depth limit.

Example


Depth Limited Search
S
A
B
C
D
I
J
E
F
G
H
Level 0
Level 1
Level 2
Level 3

4- Depth-Limited Search

- Complete: no (if goal beyond l (l<d), or infinite branch length)
- **Time:** $O(b^l)$
- **Space:** $O(bl)$
- **Optimal:** No (if l < d)

Criterion	Depth-Limited
Complete?	No
Time	$O(b^l)$
Space	$O(bl)$
Optimal?	No

5- Iterative Deepening Depth-First Search

5- Iterative Deepening Depth-First Search

The iterative deepening algorithm is a combination of DFS and BFS algorithms.

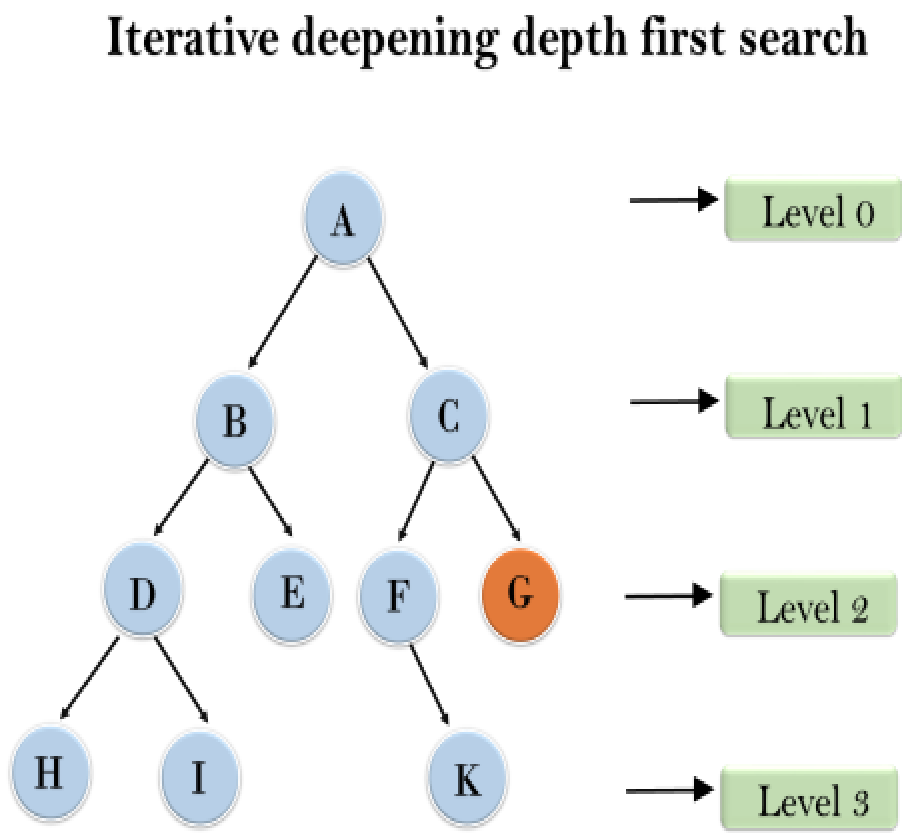
This search algorithm finds out the best depth limit and does it by gradually increasing the limit until a goal is found.

This algorithm performs depth-first search up to a certain "depth limit", and it keeps increasing the depth limit after each iteration until the goal node is found.

This Search algorithm combines the benefits of Breadth-first search's fast search and depth-first search's memory efficiency.

The iterative search algorithm is useful uninformed search when search space is large, and depth of goal node is unknown.

5- Iterative Deepening Depth-First Search



1'st Iteration-----> A

2'nd Iteration----> A, B, C

3'rd Iteration------>A, B, D, E, C, F, G

4'th Iteration------>A, B, D, H, I, E, C, F, K, G
In the fourth iteration, the algorithm will find the goal node.

Properties of Iterative Deepening Search

Complete: Yes (if the b is finite)

Time: $O(b^d)$

Space: $O(bd)$

Optimal: Yes, if step cost = 1

Criterion	Iterative Deepening
Complete?	Yes
Time	$O(b^d)$
Space	$O(bd)$
Optimal?	Yes

6- Bi-directional Search

6- Bi-directional Search

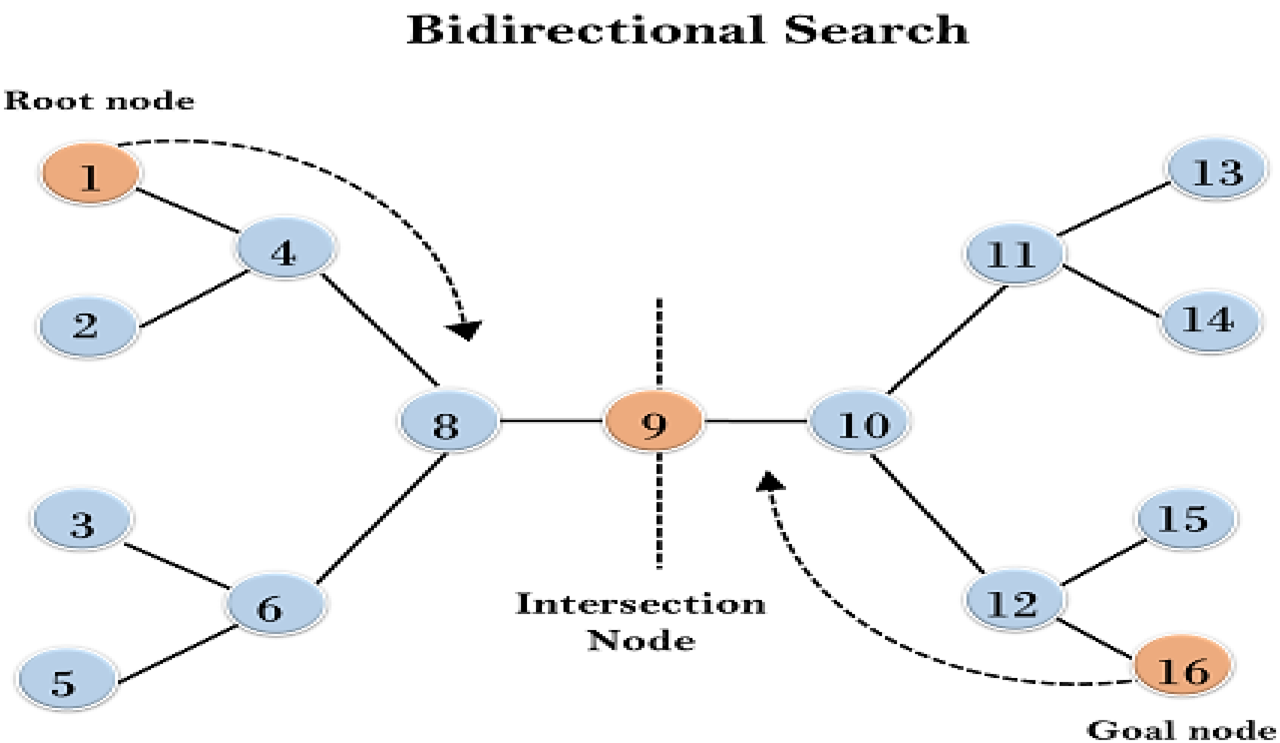
Bidirectional search algorithm runs two simultaneous searches, one form initial state called as forward-search and other from goal node called as backward-search, to find the goal node.

Bidirectional search replaces one single search graph with two small subgraphs in which one starts the search from an initial vertex and other starts from goal vertex.
The search stops when these two graphs intersect each other.

Bidirectional search can use search techniques such as BFS, DFS, etc.

Bi-directional Search Example

In the below search tree, bidirectional search algorithm is applied. This algorithm divides one graph/tree into two sub-graphs. It starts traversing from node 1 in the forward direction and starts from goal node 16 in the backward direction.
The algorithm terminates at node 9 where two searches meet.



6- Bi-directional Search

Completeness: Bidirectional Search is complete if we use BFS in both searches.
Time Complexity: Time complexity of bidirectional search using BFS is $\mathbf{O(b^d)}$.
Space Complexity: Space complexity of bidirectional search is $\mathbf{O(b^d)}$.
Optimal: Bidirectional search is Optimal.

Summary

- Problem formulation

- Variety of uninformed search strategies

Criterion	Breadth-First	Uniform-Cost	Depth-First	Depth-Limited	Iterative Deepening
Complete?	Yes	Yes	No	No	Yes
Time	$O(b^{d+1})$	$O(b^{\lceil C^*/\epsilon \rceil})$	$O(b^m)$	$O(b^l)$	$O(b^d)$
Space	$O(b^{d+1})$	$O(b^{\lceil C^*/\epsilon \rceil})$	$O(bm)$	$O(bl)$	$O(bd)$
Optimal?	Yes	Yes	No	No	Yes