Artificial & Computational Intelligence

DSE CLZG557

M2 : Problem Solving Agent using Search

Raja vadhana P

Assistant Professor,

BITS - CSIS

BITS Pilani

Pilani Campus

Course Plan

M1 Introduction to AI

M2 Problem Solving Agent using Search

M3 Game Playing, Constraint Satisfaction Problem

M4 Knowledge Representation using Logics

M5 Probabilistic Representation and Reasoning

M6 Reasoning over time, Reinforcement Learning

M7 AI Trends and Applications, Philosophical foundations

Module 2 : Problem Solving Agent using Search

A. Uninformed Search

B. Informed Search

C. Heuristic Functions

D. Local Search Algorithms & Optimization Problems

Problem Formulation

Problem Solving Agents

Goal based decision making agents which finds sequence of actions that leads to the desirable stated.

Phases of Solution Search by PSA

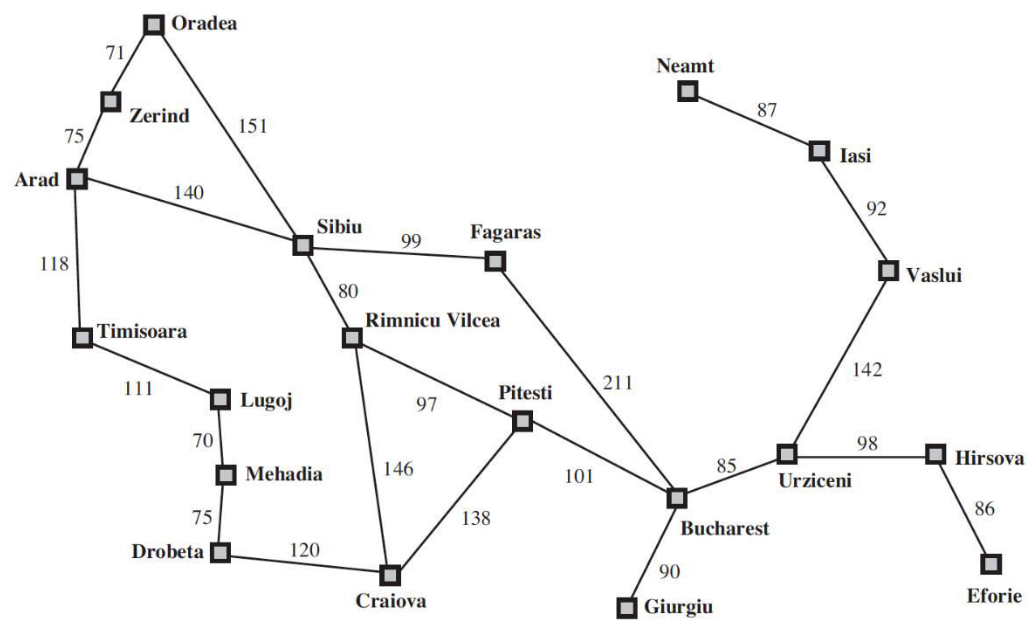
Goal Formulation

Optimizes the Objective (Local | Global)
Limits the Actions

Problem Formulation

Search Phase

Execution Phase



Problem Solving Agents

Goal Formulation

Problem Formulation

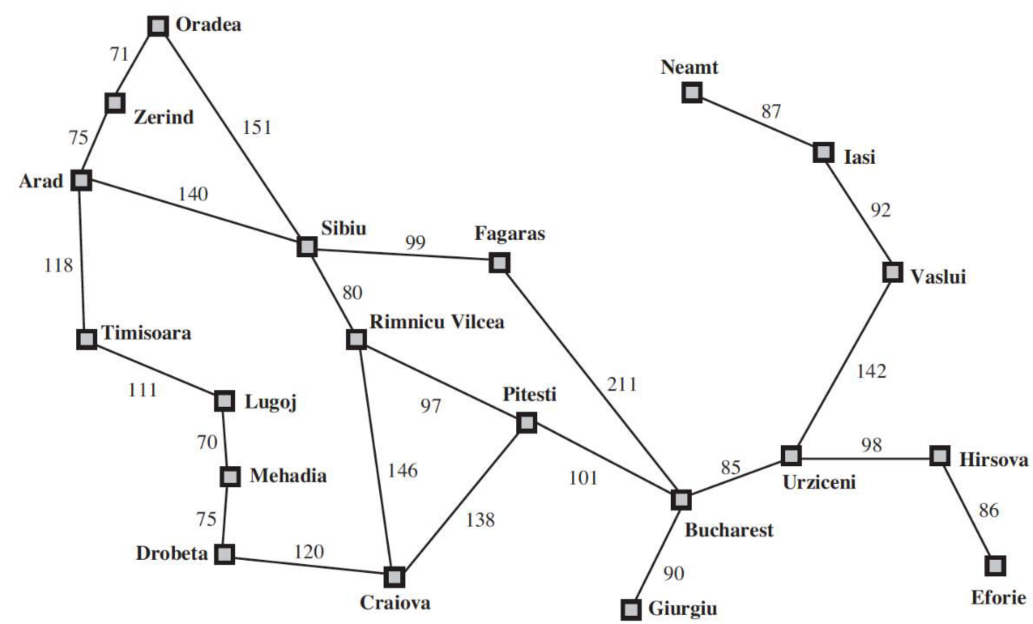
Search Phase

Execution Phase

Phases of Solution Search by PSA

State Space Creations [in the path of Goal]

Lists the Actions



Problem Solving Agents

Goal Formulation

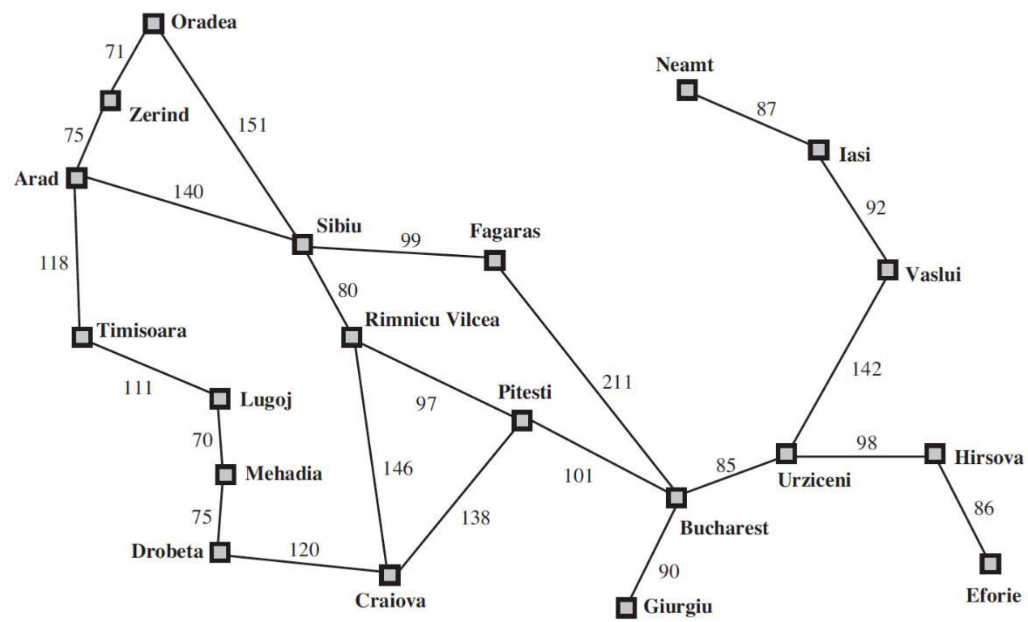
Problem Formulation

Search Phase

Execution Phase

Phases of Solution Search by PSA

Assumptions – Environment :
Static
Observable
Discrete
Deterministic



Problem Solving Agents

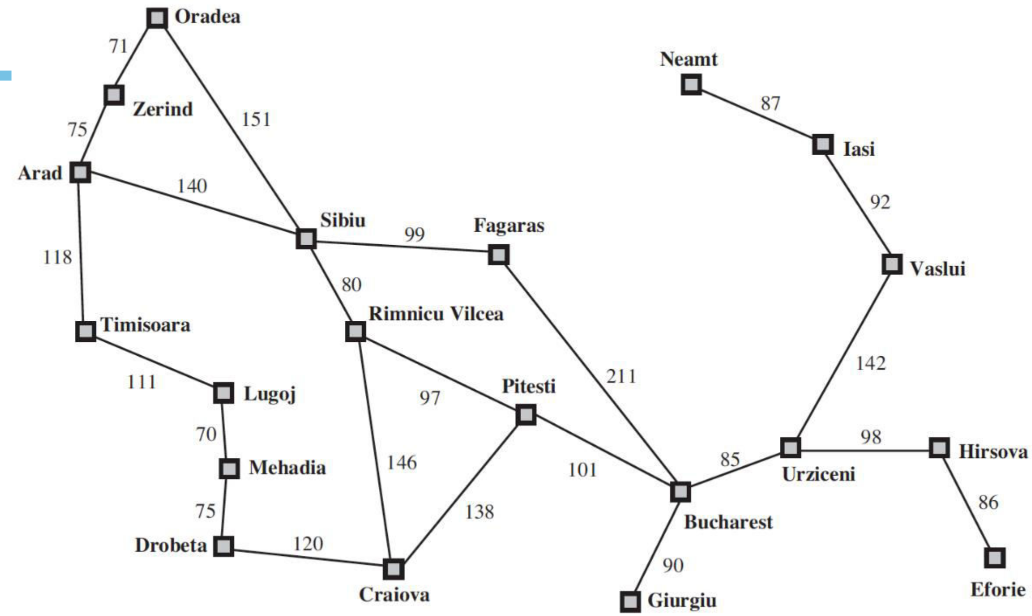
Phases of Solution Search

Goal Formulation

Problem Formulation

Search Phase

Execution Phase



Examine all sequence
Choose best | Optimal

Problem Solving Agents – Problem Formulation

Abstraction Representation
Decide what actions under states to take to achieve a goal

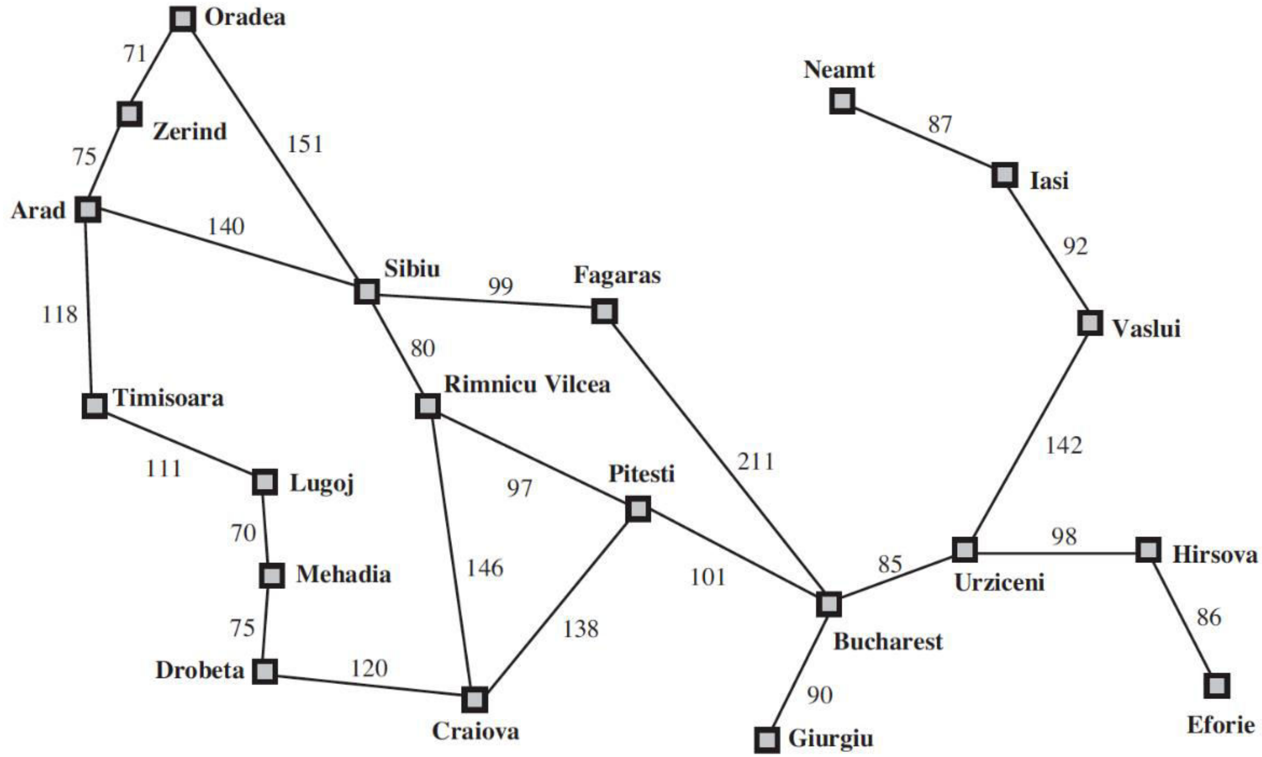
5 Components

- Initial State
- Possible Action | Operators
- Successor Function |Transition Model
- Goal Test
- Path Cost

A function that assigns a numeric cost to each path. A path is a series of actions. Each action is given a cost depending on the problem.

Solution = Path Cost Function + Optimal Solution

Problem Solving Agents – Problem Formulation



Initial State –E.g., *In(Arad)*

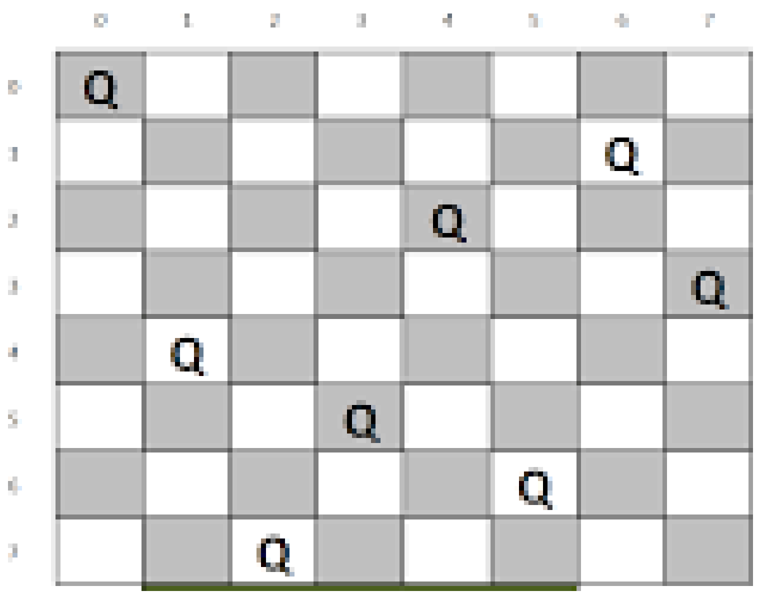
Possible Actions – *ACTIONS(s)* ➔ *{Go(Sibiu), Go(Timisoara), Go(Zerind)}*

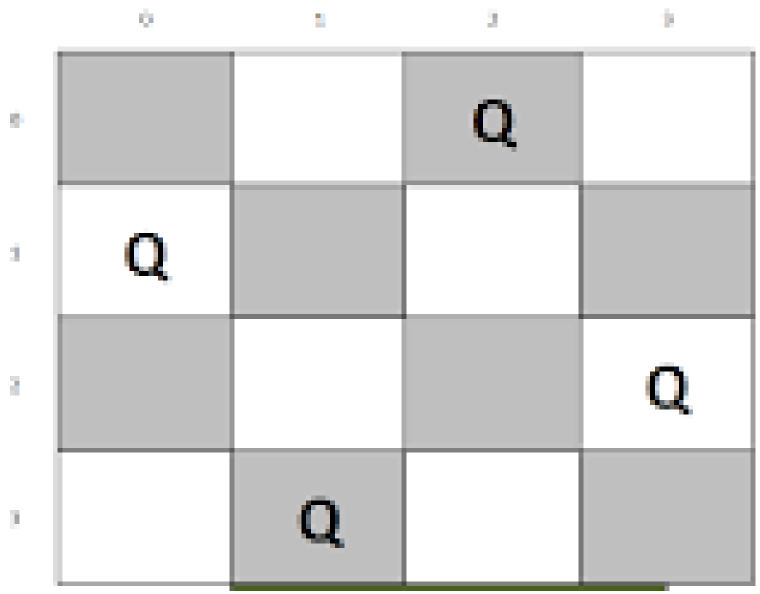
Transition Model – *RESULT(In(Arad), Go(Sibiu)) = In(Sibiu)*

Goal Test – *IsGoal(In(Bucharest)) = Yes*

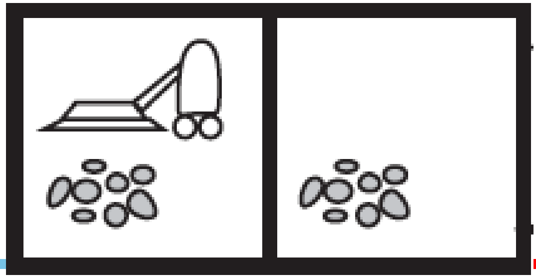
Path Cost – *cost(In(Arad), go(Sibiu)) = 140 kms*

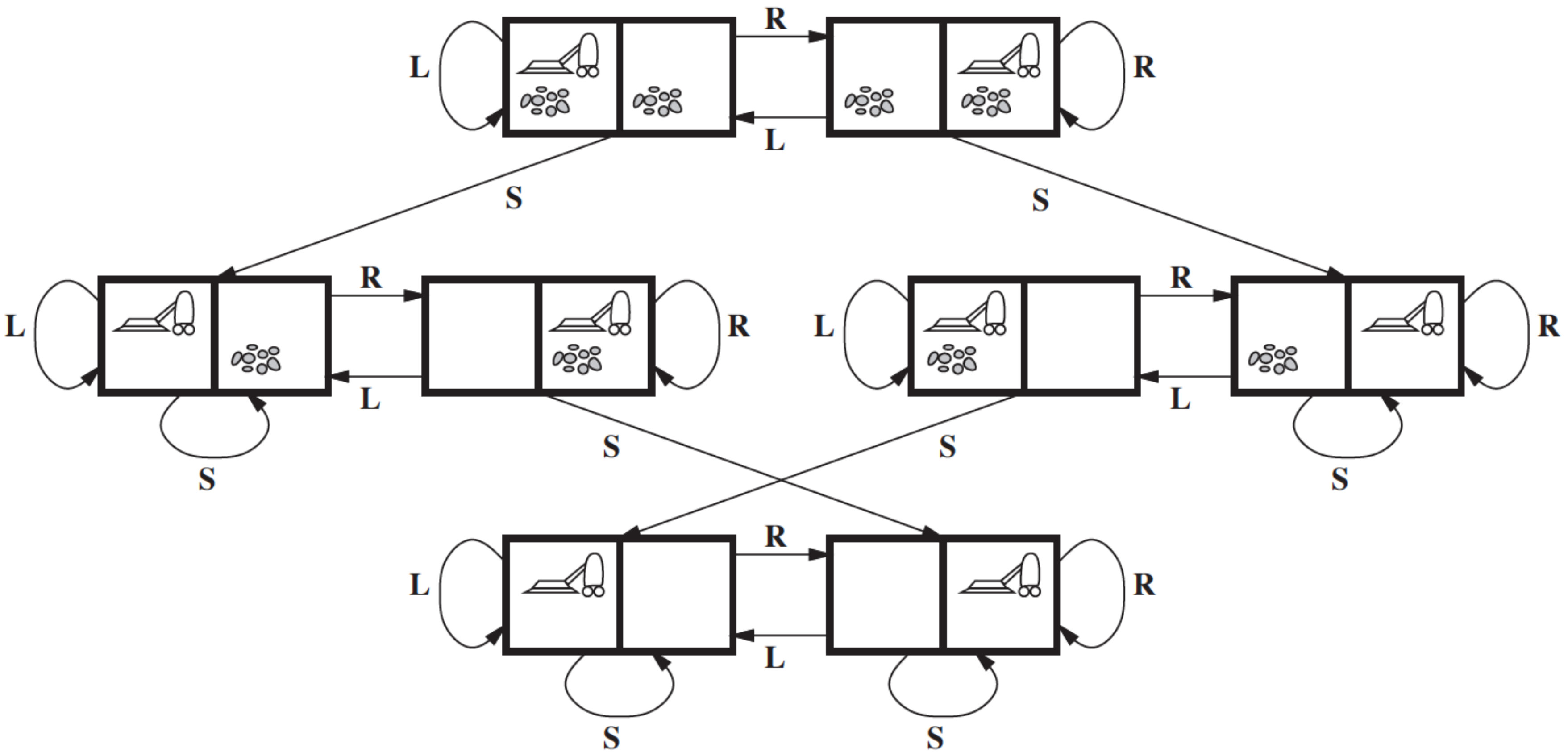
N - Queen





Vacuum World Problem





Example Problem Formulation

	Vacuum World	8 – Queen Problem	Travelling Problem
Initial State	Any	No Queen on board	Based on the problem
Possible Actions	[Move Left, Move Right, Suck]	Add a Queen to any empty square	Take a flight | Train | Shop
Transition Model	[A, ML] = [B , Dirty] [A, ML] = [B, Clean]	[A1, B2] = [FAIL] [A1, B3] = [SAFE]	[A, Go(A->S)] = [S]
Goal Test	Is all room clean? [A, Clean] [B, Clean]	All Queen Safe	Is current = B (destination)
Path Cost	No of steps in path	No of Moves done, backtracking	Cost + Time + Quality

Searching for Solutions

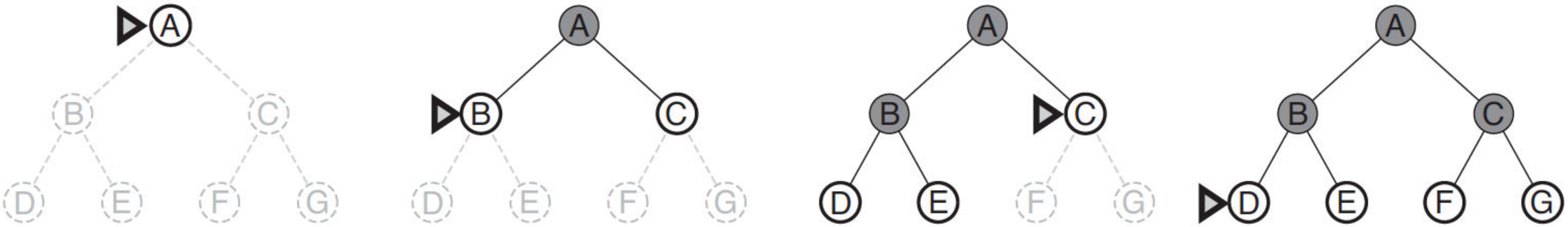
Choosing the current state, testing possible successor function, expanding current state to generate new state is called Traversal. Choice of which state to expand – Search Strategy

Search Strategy (under certainty)

- **Uninformed**
  - BFS, DFS, UCS
  - IDS, DLS
  - Bi-Directional
- **Informed**
  - Best First Search
  - A*
  - AO*

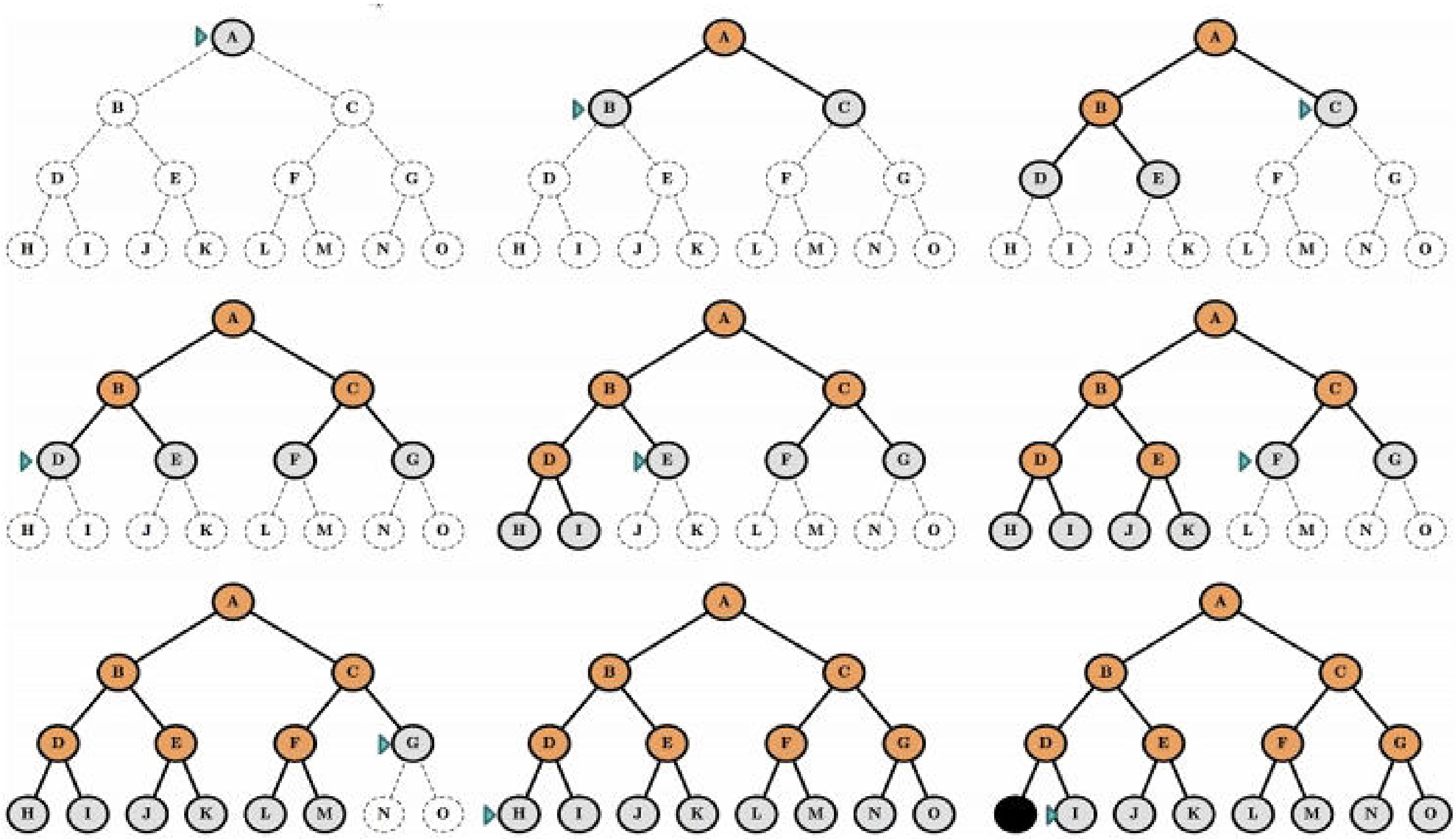
Uninformed Search
BFS & its Variants

Breadth First Search (BFS)

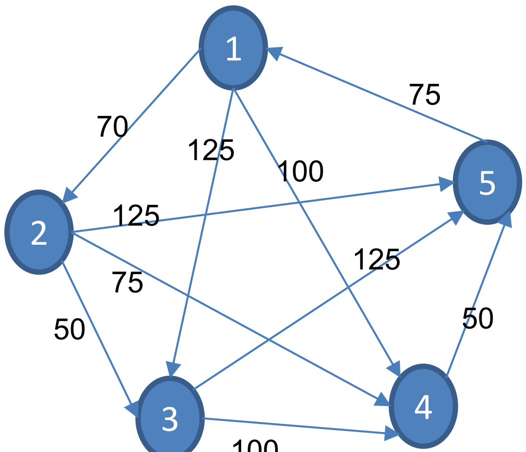


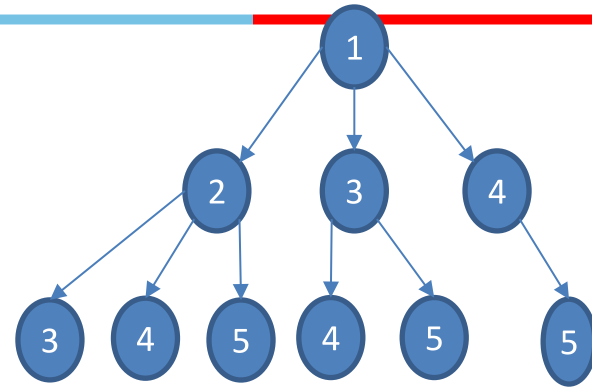
Generate all nodes at a given depth
before proceeding to deeper nodes

Breadth First Search (BFS)



BFS – Uninformed





Each NODE in in the search tree denotes an entire PATH through the state space graph.

(1)
(1 2) (1 3) (1 4)
TEST FAILED

(1 3) (1 4) (1 2 3) (1 2 4) (1 2 5)
(1 2 3) (1 2 4) (1 2 5) (1 3 4) (1 3 5) (1 4 5)
TEST PASSED

C(1-2-5) = 70 + 125 =195
Expanded : 4
Generated : 10
Max Queue Length : 6

Breadth First Search – Evaluation

Depth	Nodes	Time	Memory
2	110	.11 milliseconds	107 kilobytes
4	11,110	11 milliseconds	10.6 megabytes
6	10^6	1.1 seconds	1 gigabyte
8	10^8	2 minutes	103 gigabytes
10	10^{10}	3 hours	10 terabytes
12	10^{12}	13 days	1 petabyte
14	10^{14}	3.5 years	99 petabytes
16	10^{16}	350 years	10 exabytes

Why is Space Complexity a big problem? Imagine a problem with

- branching factor b = 10
- generates 1 million nodes/sec
- Each node requires 1KB

Breadth First Search – Evaluation

Complete – If the shallowest goal node is at a depth d, BFS will eventually find it by generating all shallower nodes

Optimal – Not necessarily. Optimal if path cost is non-decreasing function of depth of node. E.g., all actions have same cost

Time Complexity – $\mathcal{O}(b^d)$ b - branching factor, d – depth

- Nodes expanded at depth 1 = b
- Nodes expanded at depth 2 = b^2
- Nodes expanded at depth d = b^d
- Goal test is applied during generation, time complexity would be $\mathcal{O}(b^{d+1})$

Space Complexity – $\mathcal{O}(b^d)$

- $\mathcal{O}(b^{d-1})$ in explored set
- $\mathcal{O}(b^d)$ in frontier set

Uniform Cost Search

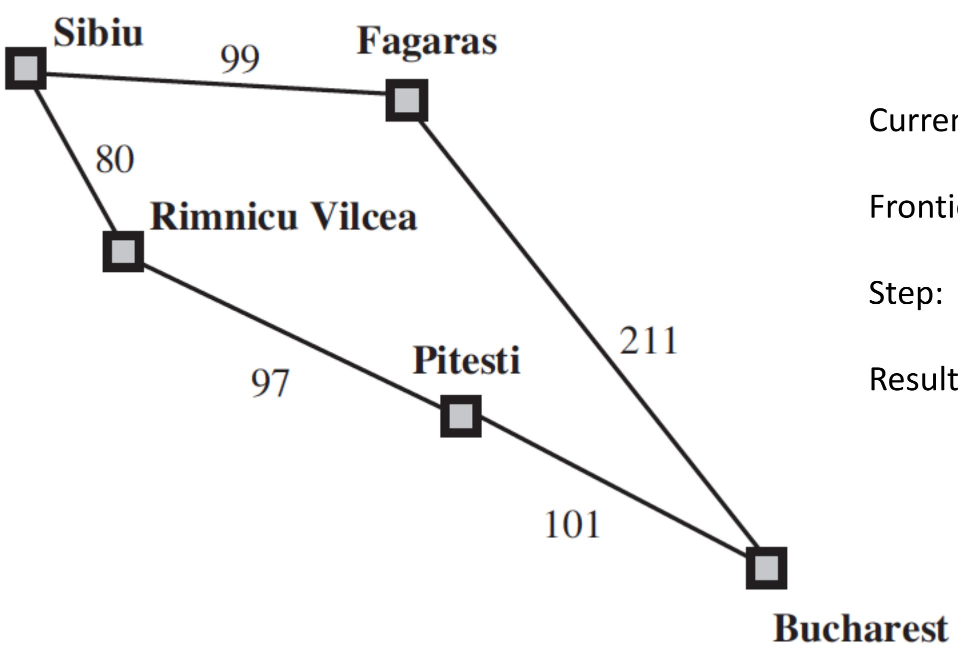
Instead of expanding the shallowest node, Uniform-Cost search expands the node n with the lowest path cost g(n)

Sorting the Frontier as a priority queue ordered by g(n)

Goal test is applied during expansion

- The goal node if generated may not be on the optimal path
- Find a better path to a node on the Frontier

Uniform Cost Search



Current State: Sibiu

Frontier: []

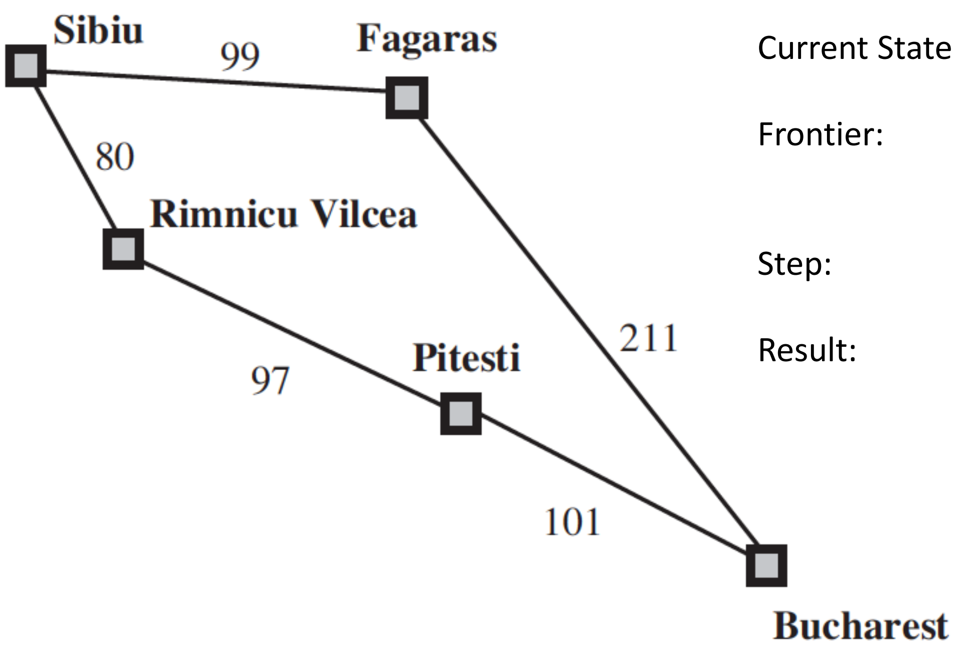
Step: Expand Sibiu

Result: Generates (“Rimnicu Vilcea” 80) (“Fagaras”, 99)

Add to Frontier

Initial State: Sibiu
Goal State: Bucharest

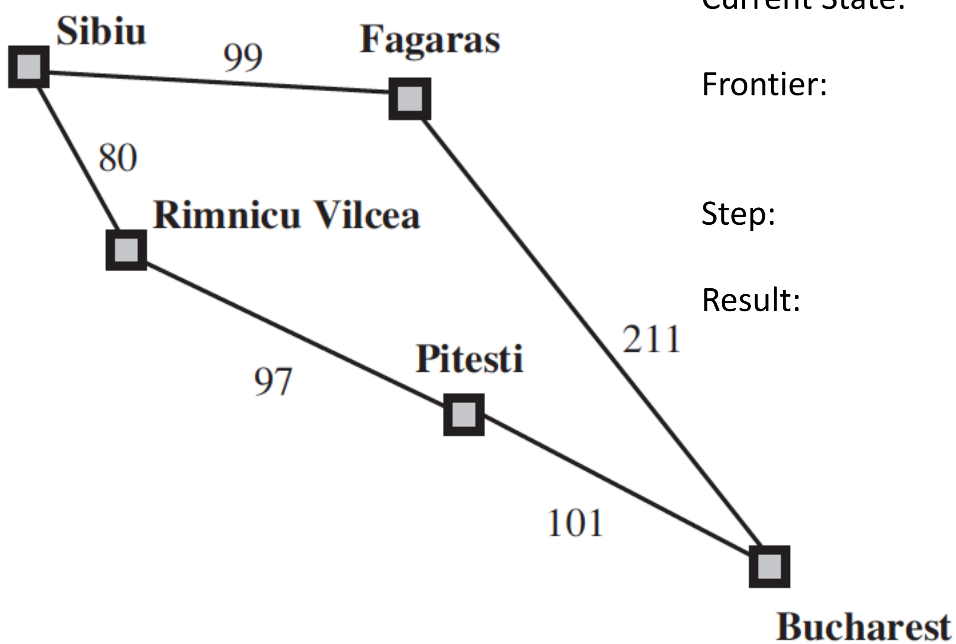
Uniform Cost Search



Current State:	Sibiu
Frontier:	[(“Rimnicu Vilcea” 80) (“Fagaras”, 99)]
Step:	Expand “Rimnicu Vilcea” (least cost)
Result:	Generates (“Pitesti”, 177) Add to Frontier

Initial State:	Sibiu
Goal State:	Bucharest

Uniform Cost Search



Current State: Rimnicu Vilcea (not a Goal state)

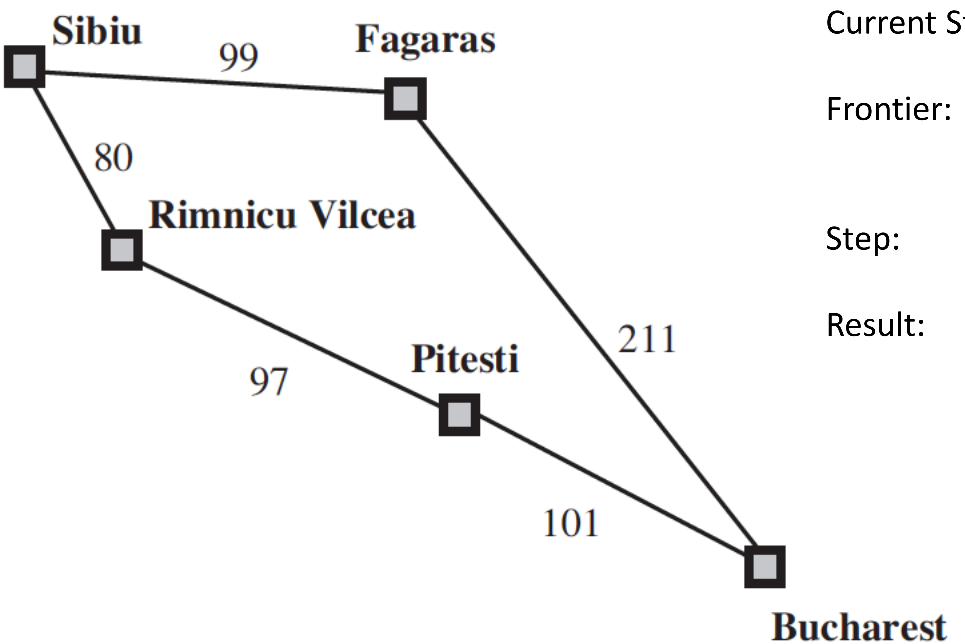
Frontier: [(“Fagaras”, 99)
(“Pitesti”, 177)]

Step: Expand “Fagaras” (least cost)

Result: Generates (“Bucharest”, 310)
Add to Frontier
(It’s a Goal State but we won’t test during generation)

Initial State: Sibiu
Goal State: Bucharest

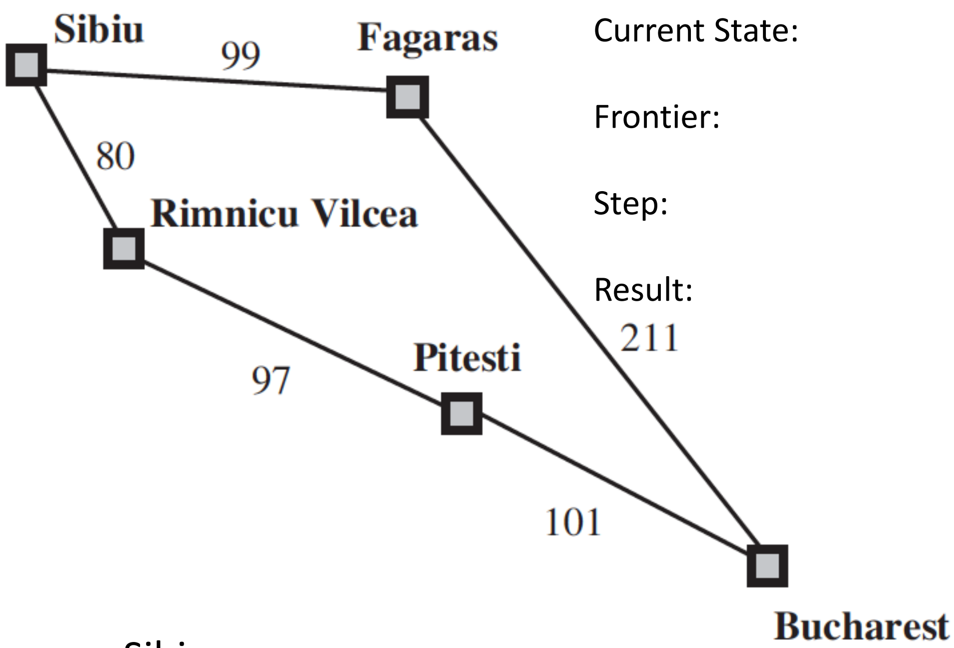
Uniform Cost Search



Current State:	Fagaras (not a goal state)
Frontier:	[("Pitesti", 177) ("Bucharest", 310)]
Step:	Expand "Pitesti" (least cost)
Result:	Generates ("Bucharest", 278) Replace in Frontier (It's a Goal State but we won't test during generation)

Initial State:	Sibiu
Goal State:	Bucharest

Uniform Cost Search



Current State: Pitesti (not a goal state)

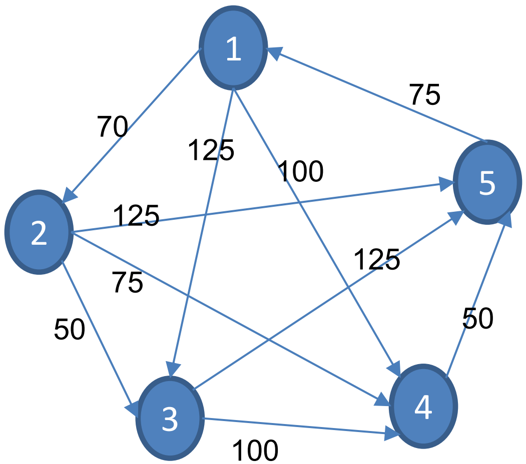
Frontier: [(“Bucharest”, 278)]

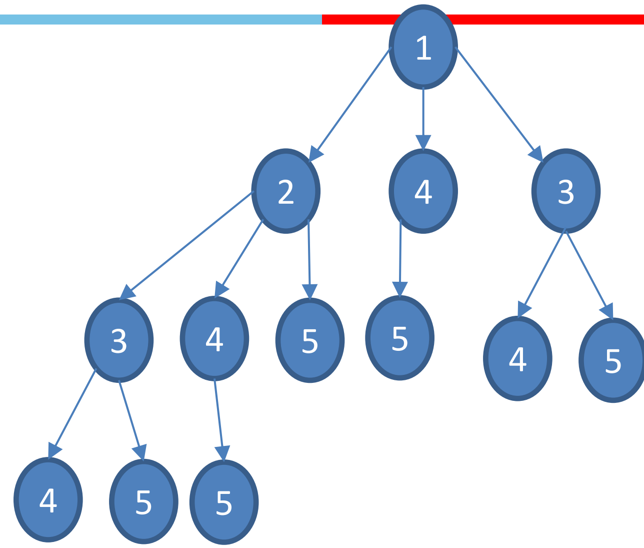
Step: Expand “Bucharest”

Result: No further generation as Goal Test satisfied Return Solution

Initial State: Sibiu
Goal State: Bucharest

UCS





(1)
(1 2 : 70) (1 4 : 100) (1 3 : 125)
TEST-F
(1 4 : 100) (1 2 3 :120)(1 3 : 125) (1 2 4 : 145) (1 2 5 : 195)
TEST-F
(1 2 3 :120)(1 3 : 125) (1 2 4 : 145) (1 4 5: 150)(1 2 5 : 195)
TEST-F
(1 3 : 125) (1 2 4 : 145) (1 4 5: 150) (1 2 3 4:170) (1 2 5 : 195) (1 2 3 5 : 245)
TEST-F
(1 2 4 : 145) (1 4 5: 150) (1 2 3 4:170) (1 2 5 : 195) (1 3 4 : 225) (1 2 3 5 : 245) (1 3 5 : 250)
TEST-F
(1 4 5: 150) (1 2 3 4:170) (1 2 4 5 : 195) (1 2 5 : 195) (1 3 4 : 225) (1 2 3 5 : 245) (1 3 5 : 250)
TEST - P

Uniform Cost Search – Evaluation

Completeness – It is complete if the cost of every step > small +ve constant ∈

- It will stuck in infinite loop if there is a path with infinite sequence of zero cost actions

Optimal – It is Optimal. Whenever it selects a node, it is an optimal path to that node.

Time and Space complexity – Uniform cost search is guided by path costs not depth or branching factor.

- If C* is the cost of optimal solution and ∈ is the min. action cost
- Worst case complexity = $\mathcal{O}(b^{1+\frac{C^*}{\epsilon}})$,
- When all action costs are equal → $\mathcal{O}(b^{d+1})$, the BFS would perform better
  - As Goal test is applied during expansion, Uniform Cost search would do extra work

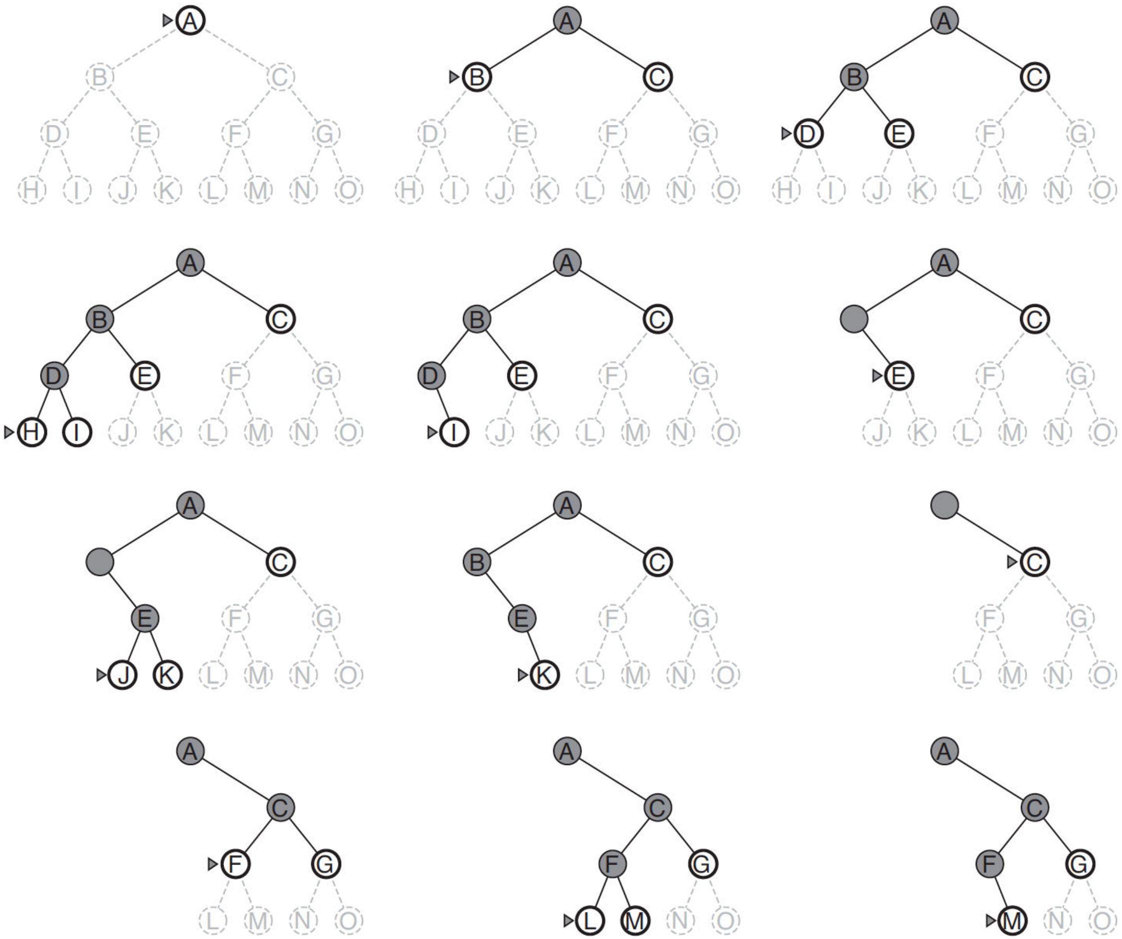
Uninformed Search

DFS & its Variants

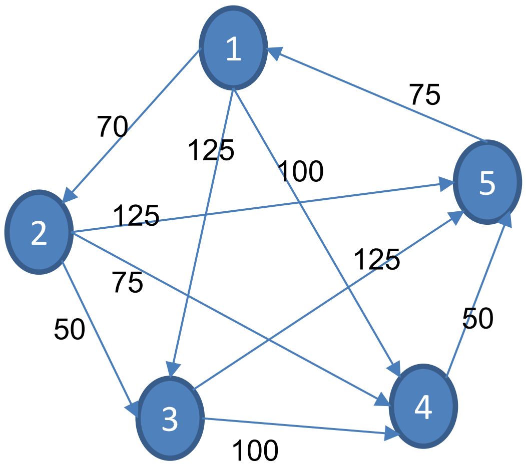
Depth First Search (DFS)

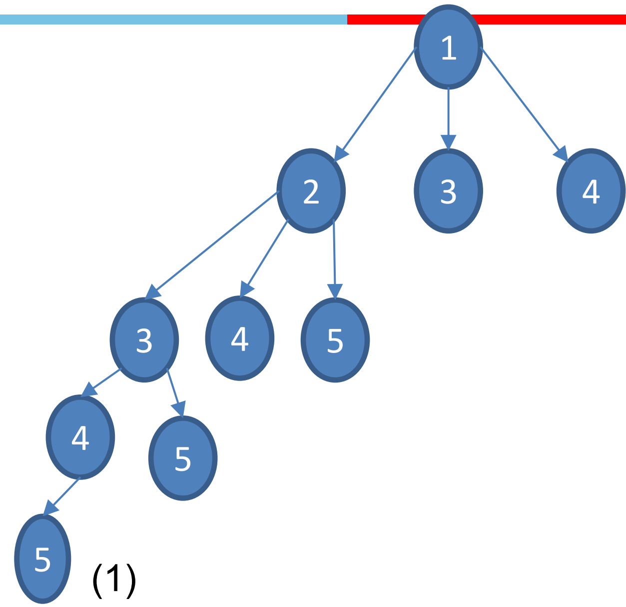
- Uses LIFO as the Frontier Strategy
- Expands deepest node in current Frontier
- Instance of Graph Search Algorithm

Depth First Search (DFS)



DFS – Uninformed





(1)
(1 2) (1 3) (1 4)
(1 2 3) (1 2 4) (1 2 5) (1 3) (1 4)
(1 2 3 4) (1 2 3 5) (1 2 4) (1 2 5) (1 3) (1 4)
(1 2 3 4 5) (1 2 3 5) (1 2 4) (1 2 5) (1 3) (1 4)

C(1-2-3-4-5) = 70 + 50 + 100 + 50 =270
Expanded : 4
Generated : 10
Max Queue Length : 6

Depth First Search (DFS)

Completeness – Complete in finite state spaces because it will eventually expand every node

Optimal – Not Optimal as it would stop when the goal node is reached without evaluating if there is a better path

Time Complexity - $\mathcal{O}(b^m)$ where m = maximum depth of any node

- Can be much larger than the size of state space
- m can be much larger than d (shallowest goal)

Space Complexity – Needs to store only one path and unexpanded siblings.

- Any node expanded with all its children can be removed from memory
- Requires storage of only $\mathcal{O}(bm)$, b – branching factor, m - max depth

Depth Limited Search (DLS)

- Running DFS with a predetermined depth limit l

- **Completeness**: No, cannot guarantee a goal if l < d
- **Optimal**: No
- **Time complexity:** $\mathcal{O}(b^l)$
- **Space Complexity:** $\mathcal{O}(bm)$

Application

Breadth First Search

- Finding path in a graph (many solutions)
- Finding the Bipartitions in a graph

Depth First Search

- Find the Connectedness in a graph
- Topological Sorting

Terminologies Learnt

- Nodes
- States
- Frontier | Fringes
- Search Strategy : LIFO | FIFO | Priority Queue
- Performance Metrics
  - Completeness
  - Optimality
  - Time Complexity
  - Space Complexity
- Algorithm Terminology

- d Depth of a node	- m – maximum
- b Branching factor	- C* - Optimal Cost
- n – nodes	- E – least Cost
- l – level of a node	- N –total node generated

Required Reading: AIMA - Chapter # 3.1, 3.2, 3.3, 3.4(Partial)

Thank You for all your Attention

Note : Some of the slides are adopted from AIMA TB materials