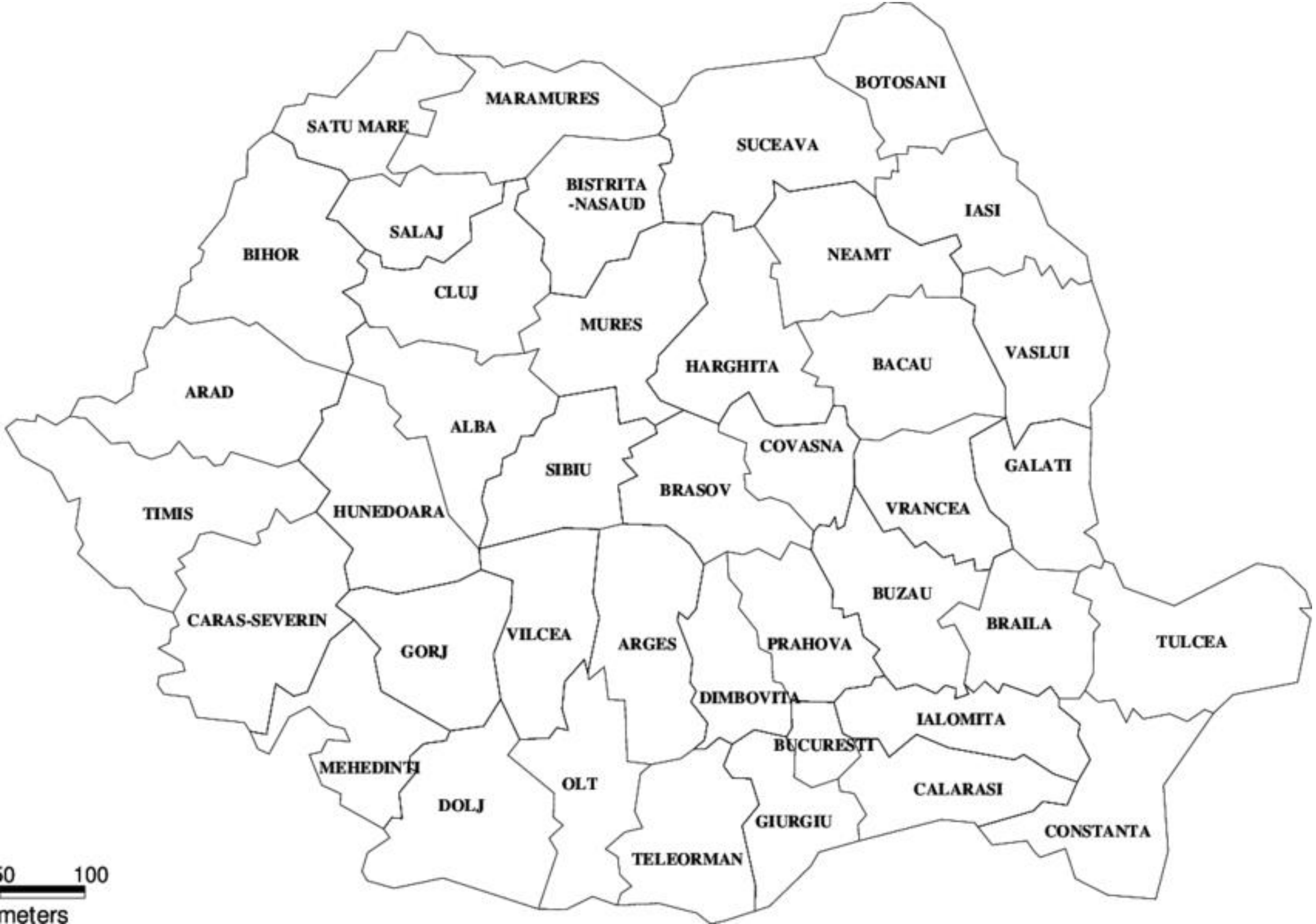
Artificial Intelligence

Lecture 02 - Search

Dr. Rajiv Misra, Professor
Dept. of Computer Science & Engg.
Indian Institute of Technology, Patna
rajivm@iitp.ac.in

Searching – Problems and Solutions



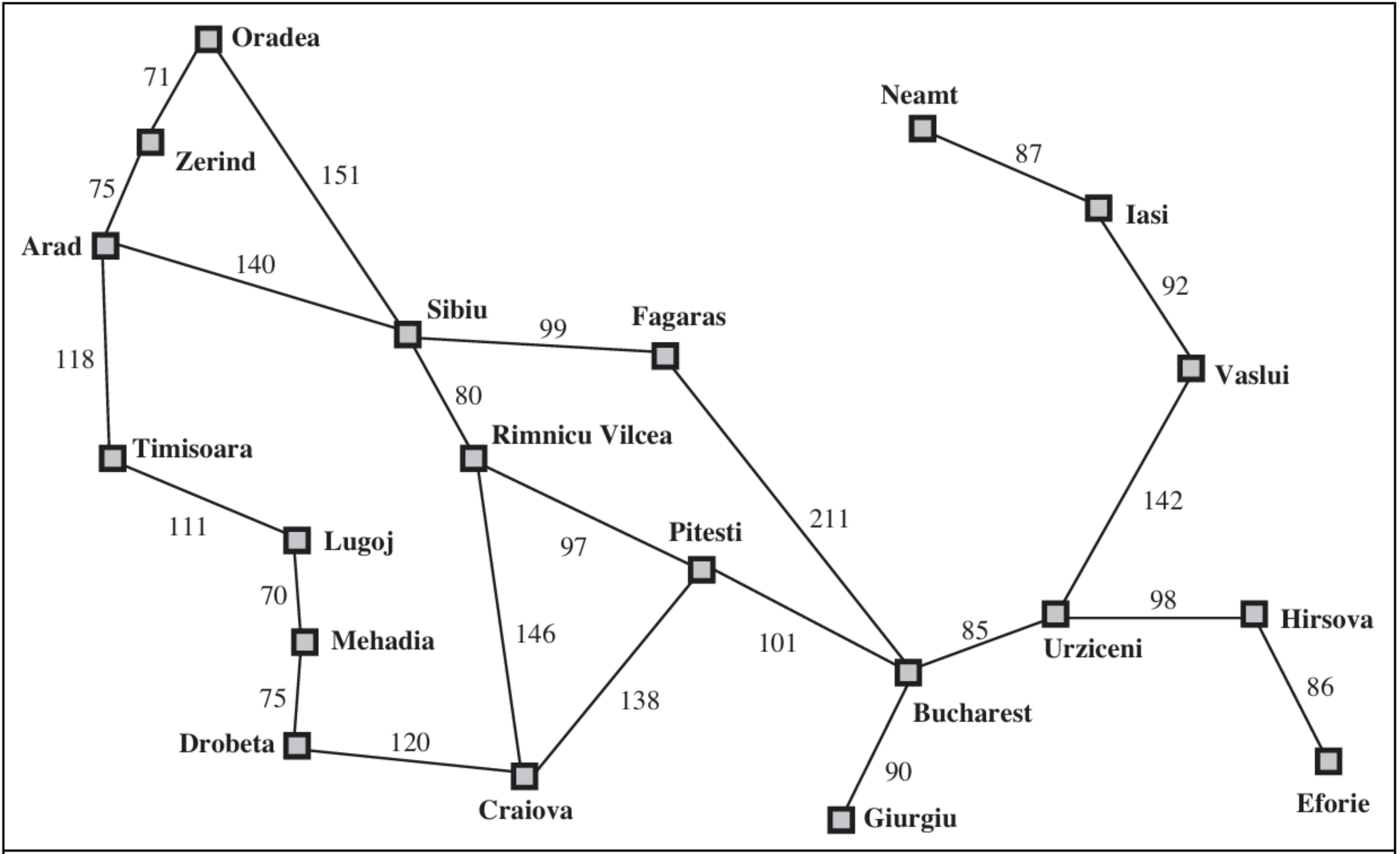
Searching – Problems and Solutions

BOTOSANI
MARAMURES
SATU MARE
SUCEAVA
BISTRITA
-NASAUD
IASI
SALAJ
BIHOR
NEAMT
CLUJ
MURES
VASLUI
HARGHITA
BACAU
ARAD
ALBA
COVASNA
SIBIU
GALATI
BRASOV
TIMIS
HUNEDOARA
VRANCEA
BUZAU
BRAILA
CARAS-SEVERIN
VILCEA
ARGES
PRAHOVA
TULCEA
GORJ
DIMBOVITA
IALOMITA
BUCURESTI
MEHEDINTI
OLT
CALARASI
DOLJ
GIURGIU
CONSTANTA
TELEORMAN
50
100
meters

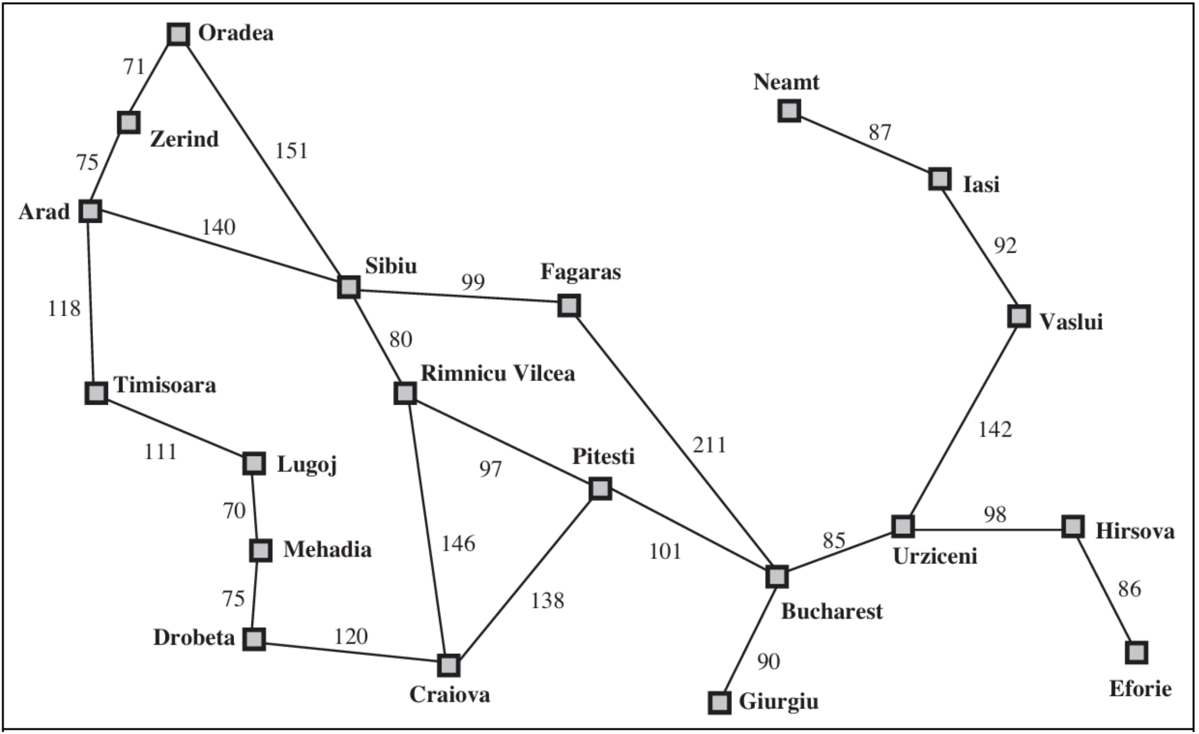
- Initial State
- Possible Actions
- Transition Model

This forms the state space which is a graph.

Searching – Romania Map

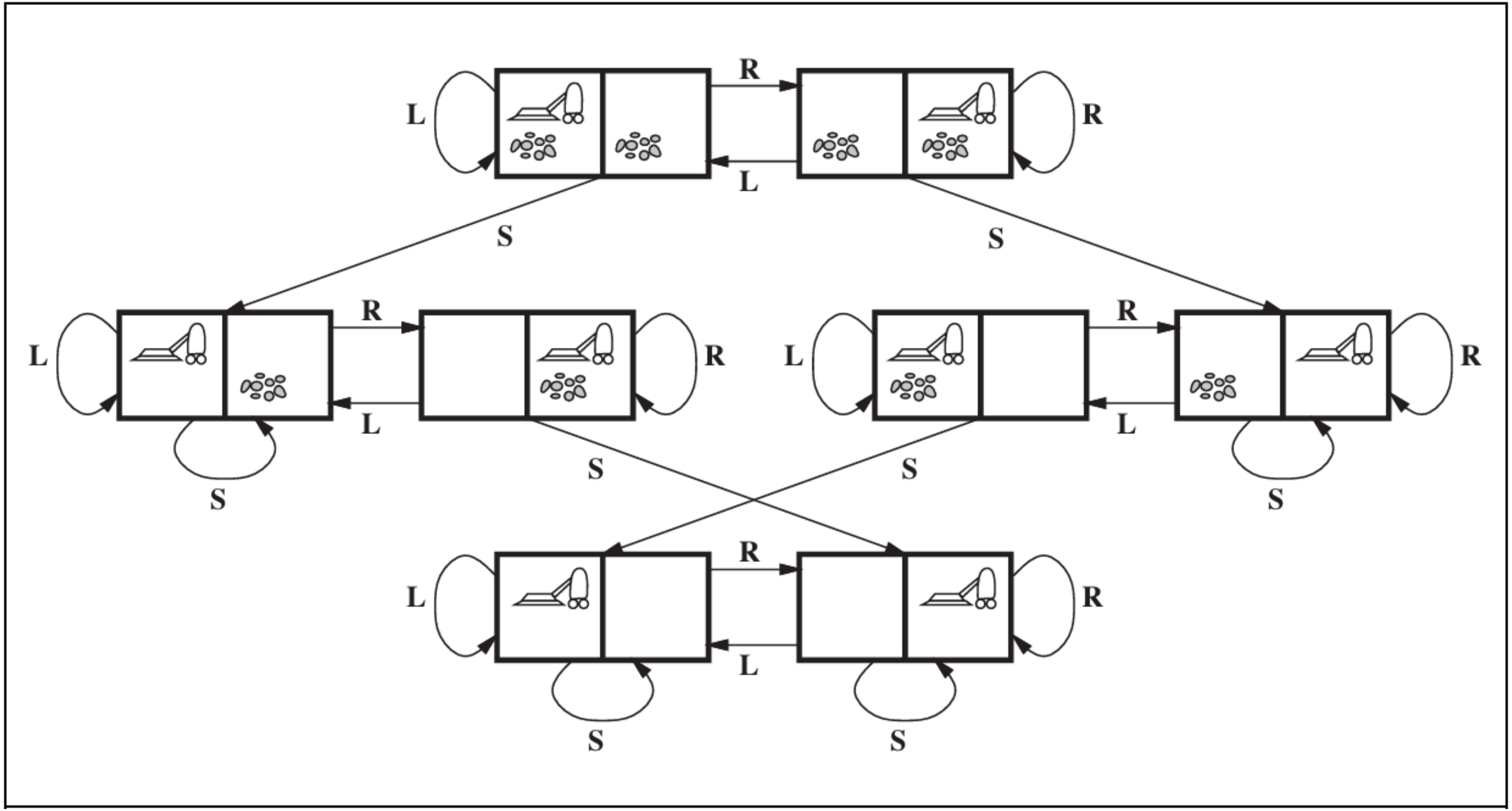


Searching – Romania Map



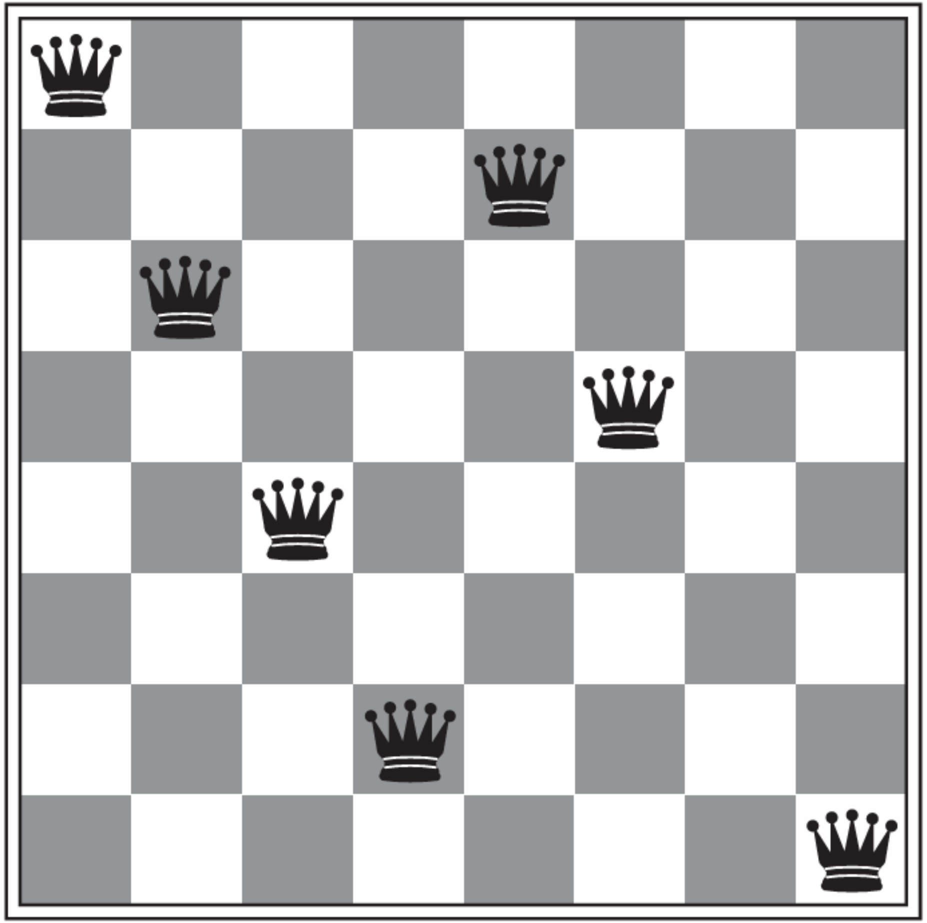
- Initial State: **in(A)**
- Possible Actions: **in(A)** has actions **{go(S), go(T), go(Z)}**
- Transition Model: **Result (in(A), go(Z)) = in(Z)**

Searching – States/Graph



- States: Consider Agent and Dirt Locations. $n2^n$ states.

Searching – States/Graph

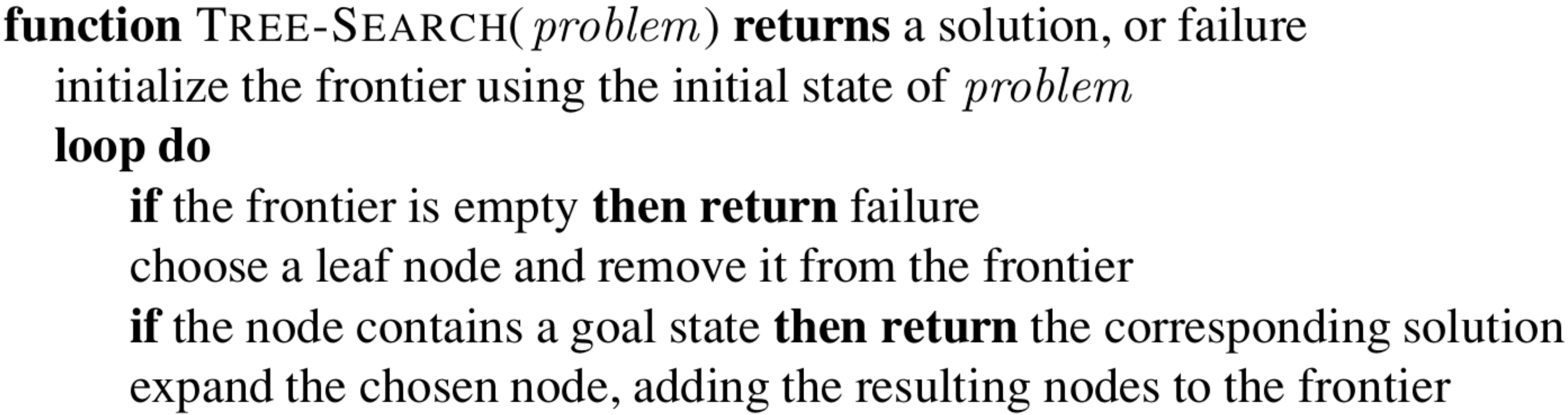


- Is this a goal state?

Searching – Problems in Real World

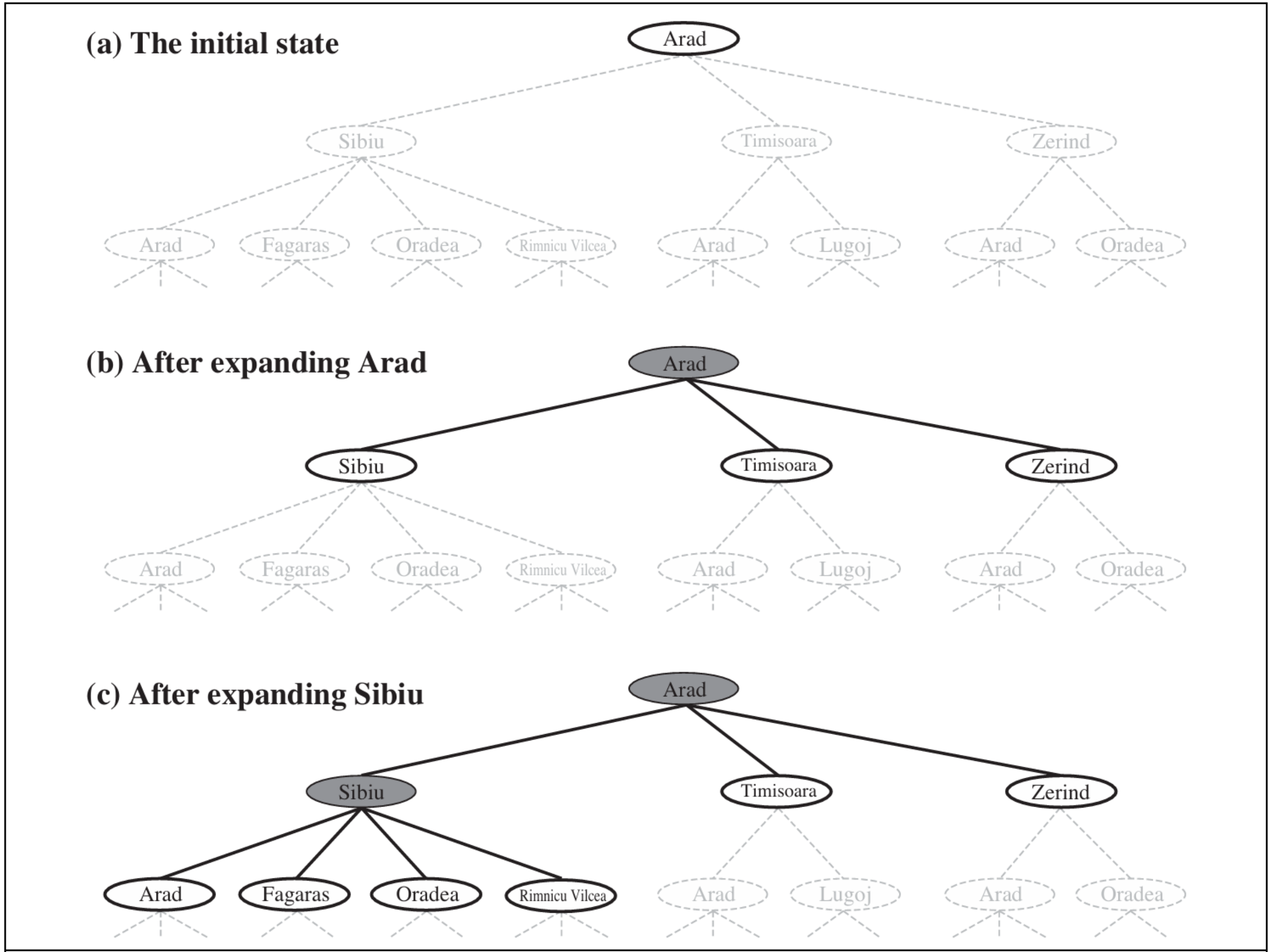
- States are much more complex
- Search Spaces can be huge
- NP-Hard (TSP)
- 2D/3D navigation

Searching – Finding a solution (Tree-Search)

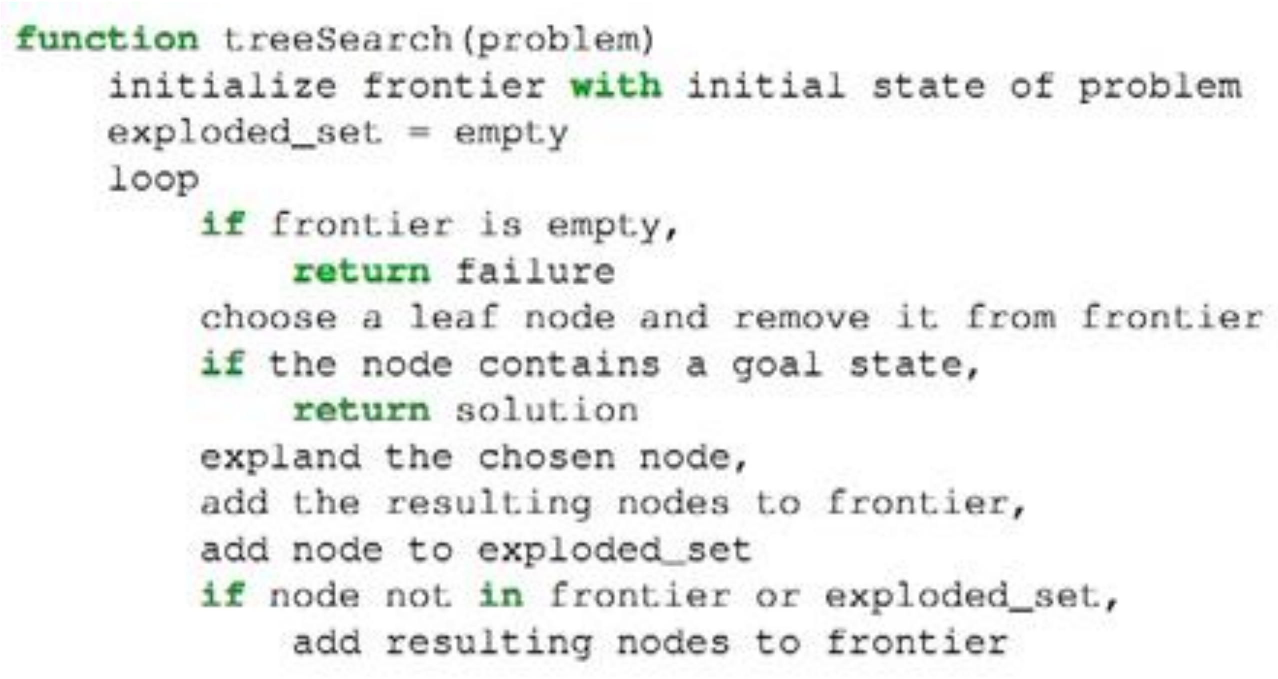


```
function TREE-SEARCH(problem) returns a solution, or failure
    initialize the frontier using the initial state of problem
    loop do
        if the frontier is empty then return failure
        choose a leaf node and remove it from the frontier
        if the node contains a goal state then return the corresponding solution
        expand the chosen node, adding the resulting nodes to the frontier
```

Searching – Expanding Trees

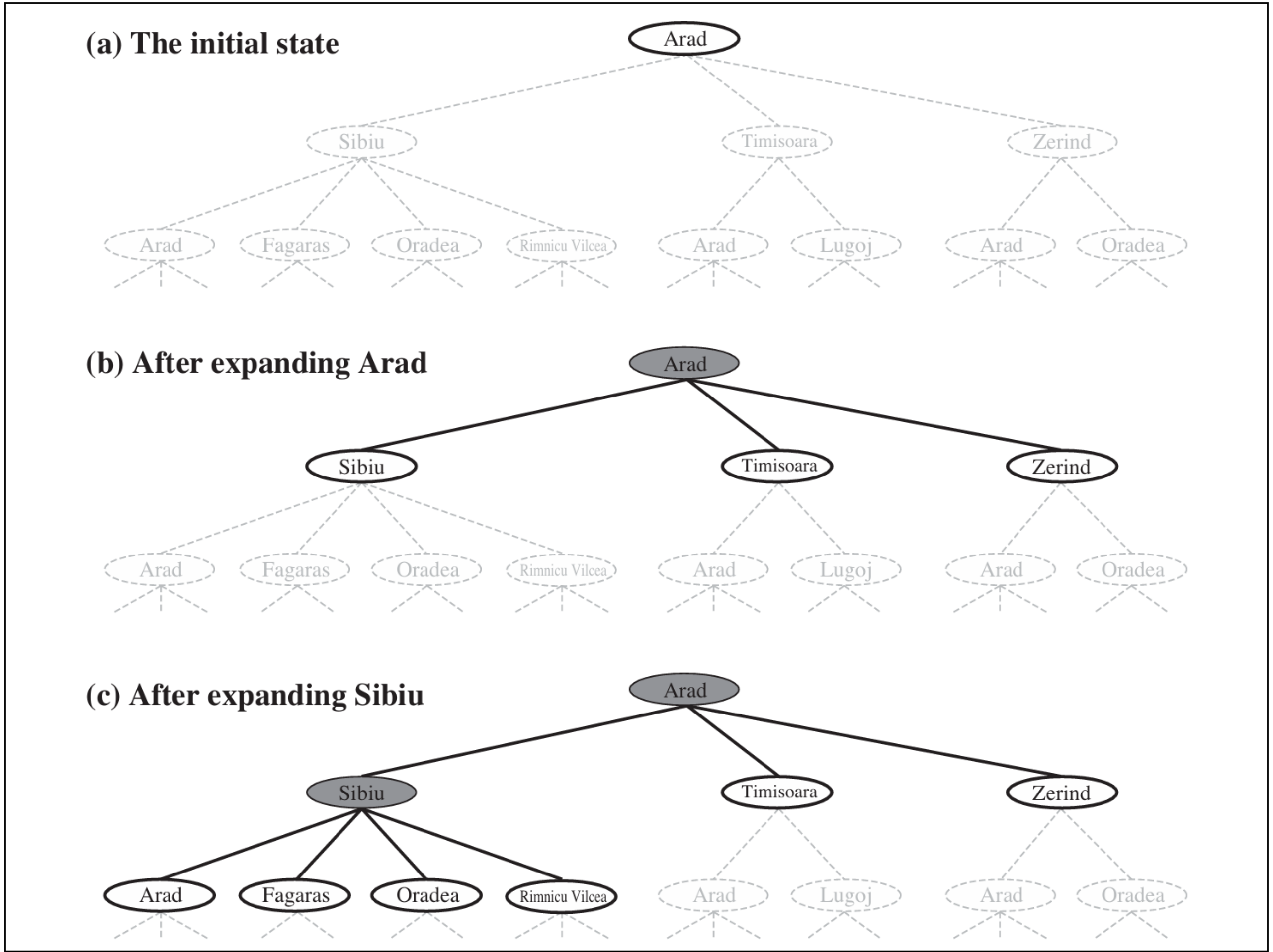


Searching – Expand but ignore loops

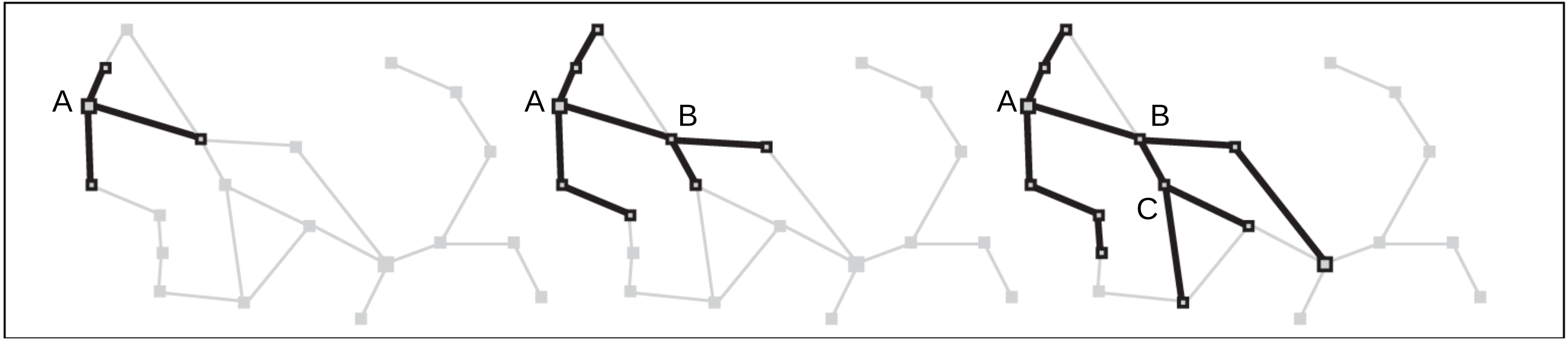


```
function treeSearch(problem)
    initialize frontier with initial state of problem
    exploded_set = empty
    loop
        if frontier is empty,
            return failure
        choose a leaf node and remove it from frontier
        if the node contains a goal state,
            return solution
        expland the chosen node,
        add the resulting nodes to frontier,
        add node to exploded_set
        if node not in frontier or exploded_set,
            add resulting nodes to frontier
```

Searching – Expanding Trees

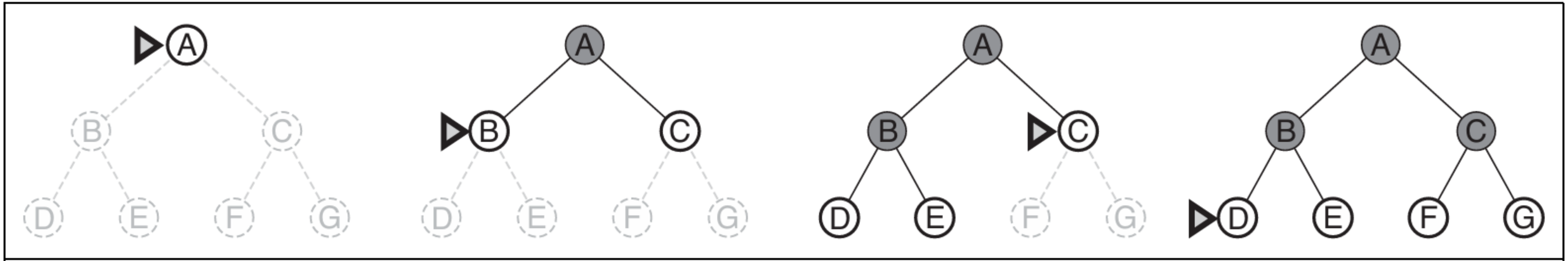


Searching – Expanding Trees

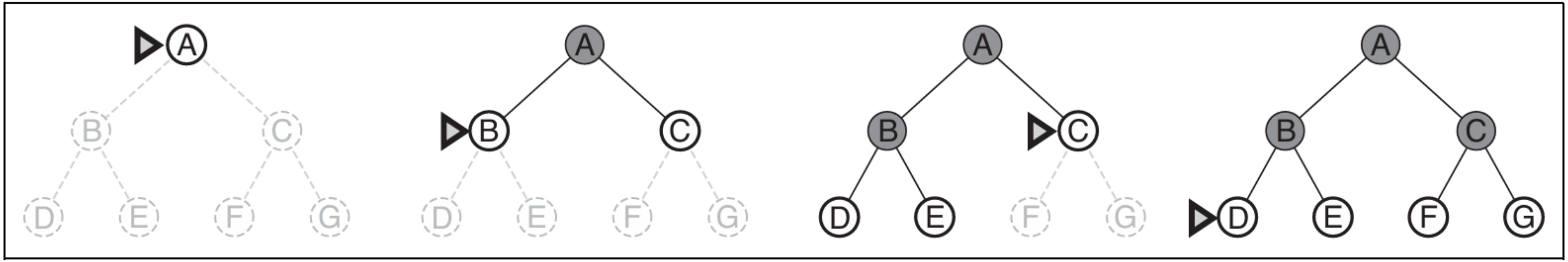


Uninformed Search – Breadth First Search (BFS)

```
function BREADTH-FIRST-SEARCH(problem) returns a solution, or failure
  node ← a node with STATE = problem.INITIAL-STATE, PATH-COST = 0
  if problem.GOAL-TEST(node.STATE) then return SOLUTION(node)
  frontier ← a FIFO queue with node as the only element
  explored ← an empty set
  loop do
     if EMPTY?(frontier) then return failure
     node ← POP(frontier)   /* chooses the shallowest node in frontier */
     add node.STATE to explored
     for each action in problem.ACTIONS(node.STATE) do
        child ← CHILD-NODE(problem, node, action)
        if child.STATE is not in explored or frontier then
           if problem.GOAL-TEST(child.STATE) then return SOLUTION(child)
           frontier ← INSERT(child, frontier)
```

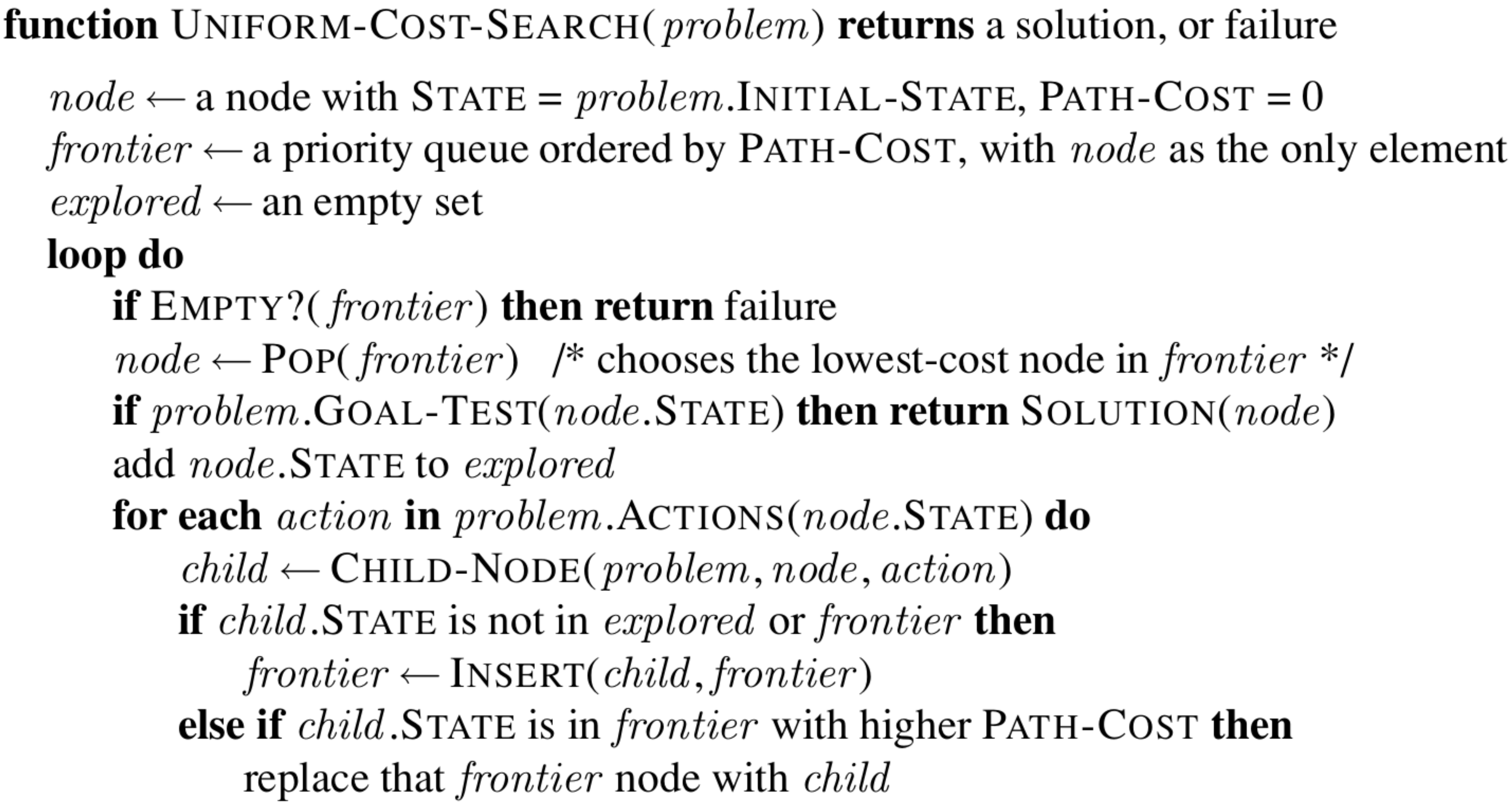


Breadth First Search



- If $\boldsymbol{b}$ is the branching factor
- $\boldsymbol{d}$ is the number of levels (depth) of the tree
- Time Complexity $\boldsymbol{O}(\boldsymbol{b}^{\boldsymbol{d}})$

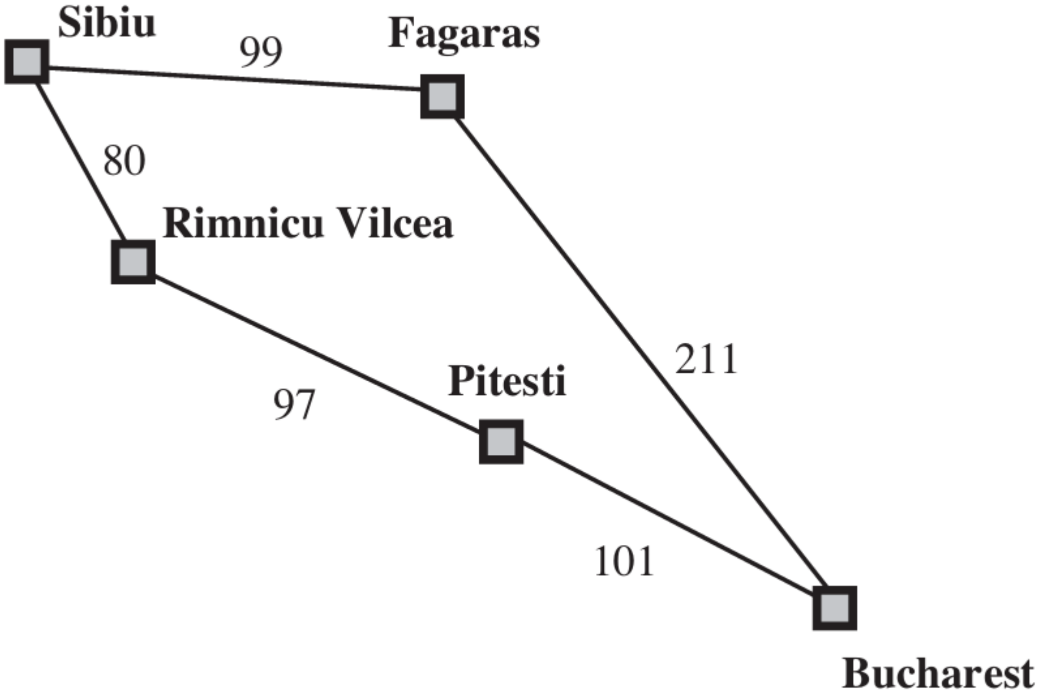
Searching – Uniform Cost Search (UCS)



```
function UNIFORM-COST-SEARCH(problem) returns a solution, or failure

  node ← a node with STATE = problem.INITIAL-STATE, PATH-COST = 0
  frontier ← a priority queue ordered by PATH-COST, with node as the only element
  explored ← an empty set
  loop do
      if EMPTY?(frontier) then return failure
      node ← POP(frontier)   /* chooses the lowest-cost node in frontier */
      if problem.GOAL-TEST(node.STATE) then return SOLUTION(node)
      add node.STATE to explored
      for each action in problem.ACTIONS(node.STATE) do
          child ← CHILD-NODE(problem, node, action)
          if child.STATE is not in explored or frontier then
              frontier ← INSERT(child, frontier)
          else if child.STATE is in frontier with higher PATH-COST then
              replace that frontier node with child
```

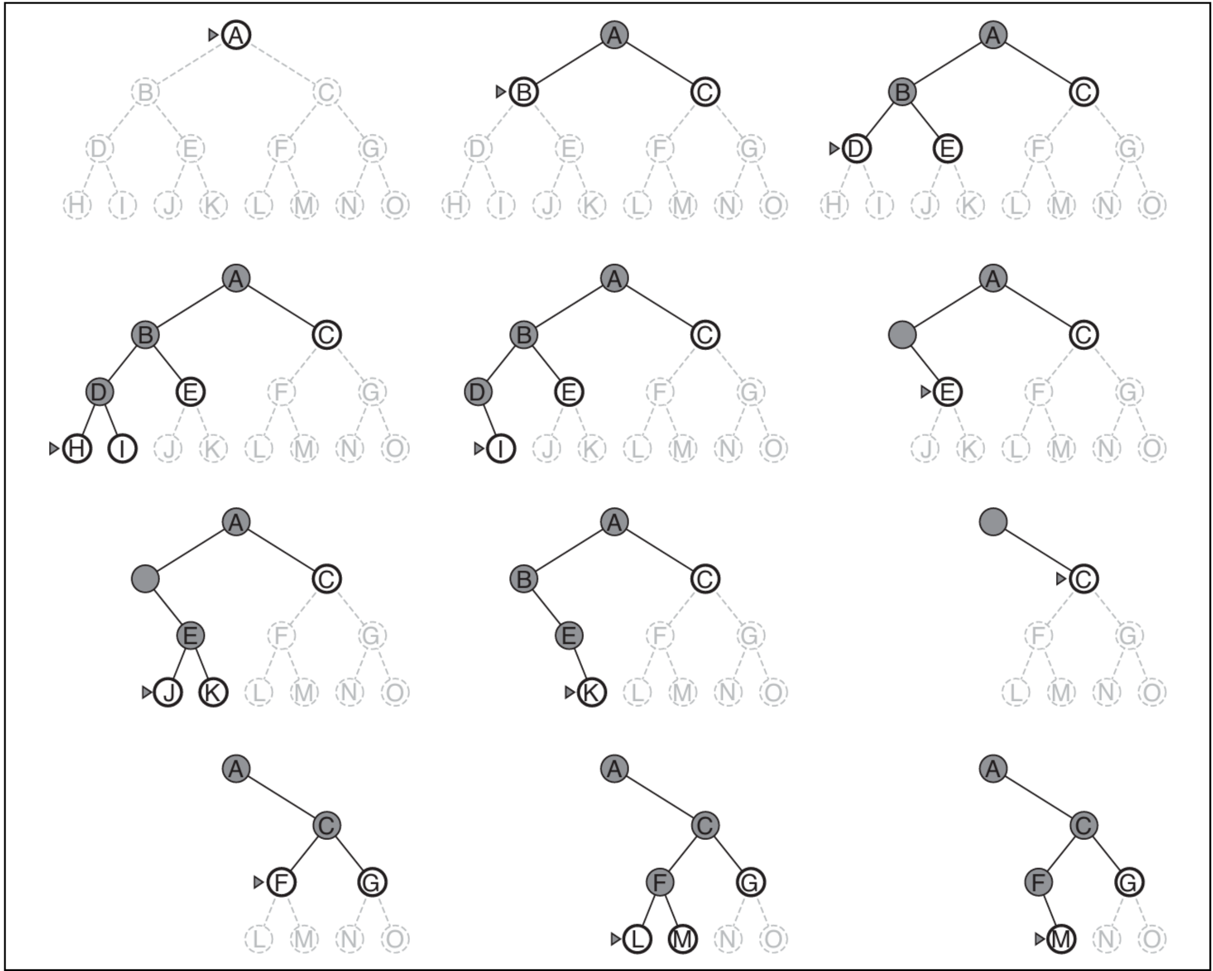
Searching – Romania Map



Depth First Search (DFS)

- DFS explores one branch completely before backtracking to the next branch.

- It lacks BFS's guarantees of finding the shortest path.



Depth First Search (DFS)

- To address DFS limitations:

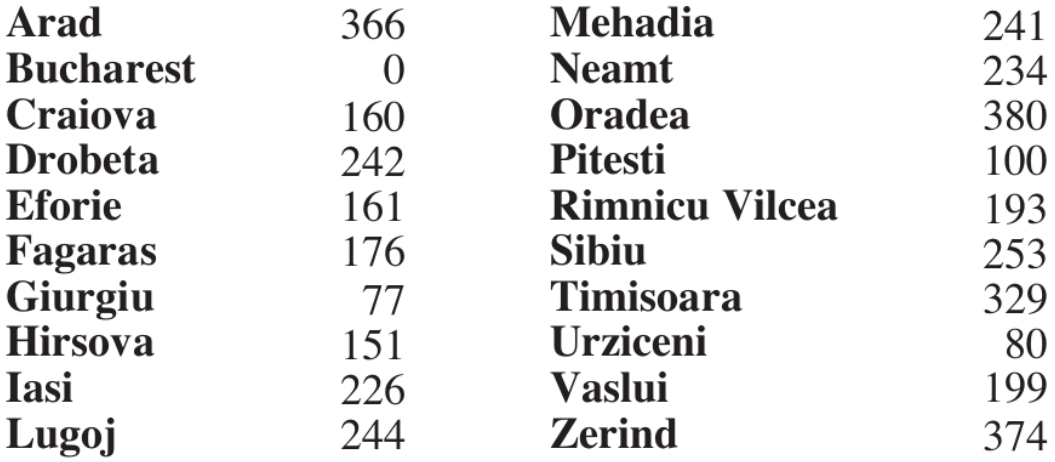
- **Limited Depth-First Search**: Stops expansion beyond a predefined depth, reducing risk of infinite loops.

- **Iterative Deepening DFS**: Gradually increases the search depth, combining the thoroughness of DFS with BFS's optimality.

- **Bidirectional Search**: Runs two simultaneous searches (one from the start and one from the goal) until they meet. This reduces the effective depth to half, with time complexity $\boldsymbol{O}(\boldsymbol{b}^{\boldsymbol{d}/\boldsymbol{2}})$

Informed Search

- **Best First Search:** node is selected for expansion based on an evaluation function, **f(n)**
- **Greedy Best First Search:** Solely based on heuristic function **denoted h(n) = estimated cost of the cheapest path from the state at node n to a goal state.**
- e.g., h(n) = *straight line distance to the goal state*



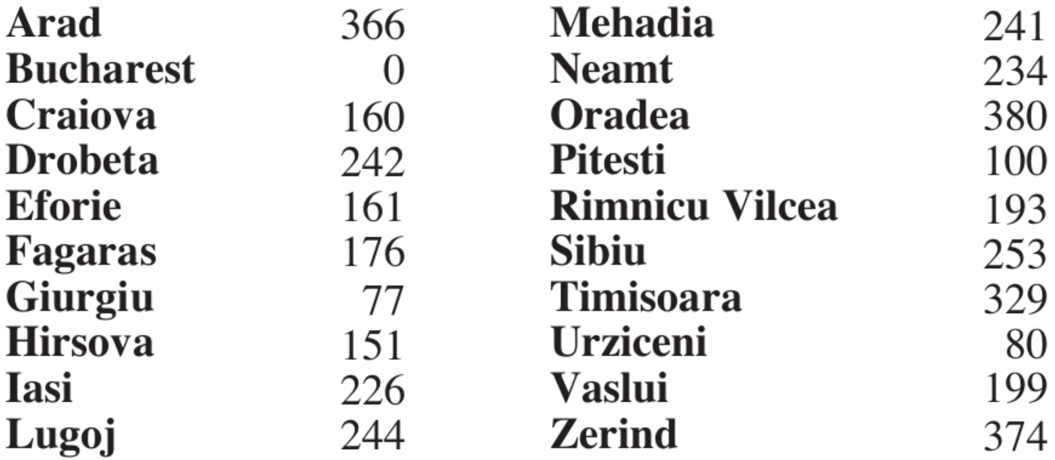
Arad	366	**Mehadia**	241
Bucharest	0	**Neamt**	234
Craiova	160	**Oradea**	380
Drobeta	242	**Pitesti**	100
Eforie	161	**Rimnicu Vilcea**	193
Fagaras	176	**Sibiu**	253
Giurgiu	77	**Timisoara**	329
Hirsova	151	**Urziceni**	80
Iasi	226	**Vaslui**	199
Lugoj	244	**Zerind**	374

A* Search – Minimizing Cost

- Combines both cost to reach the node $g(n)$ and the cost to get from the node to the goal $h(n)$

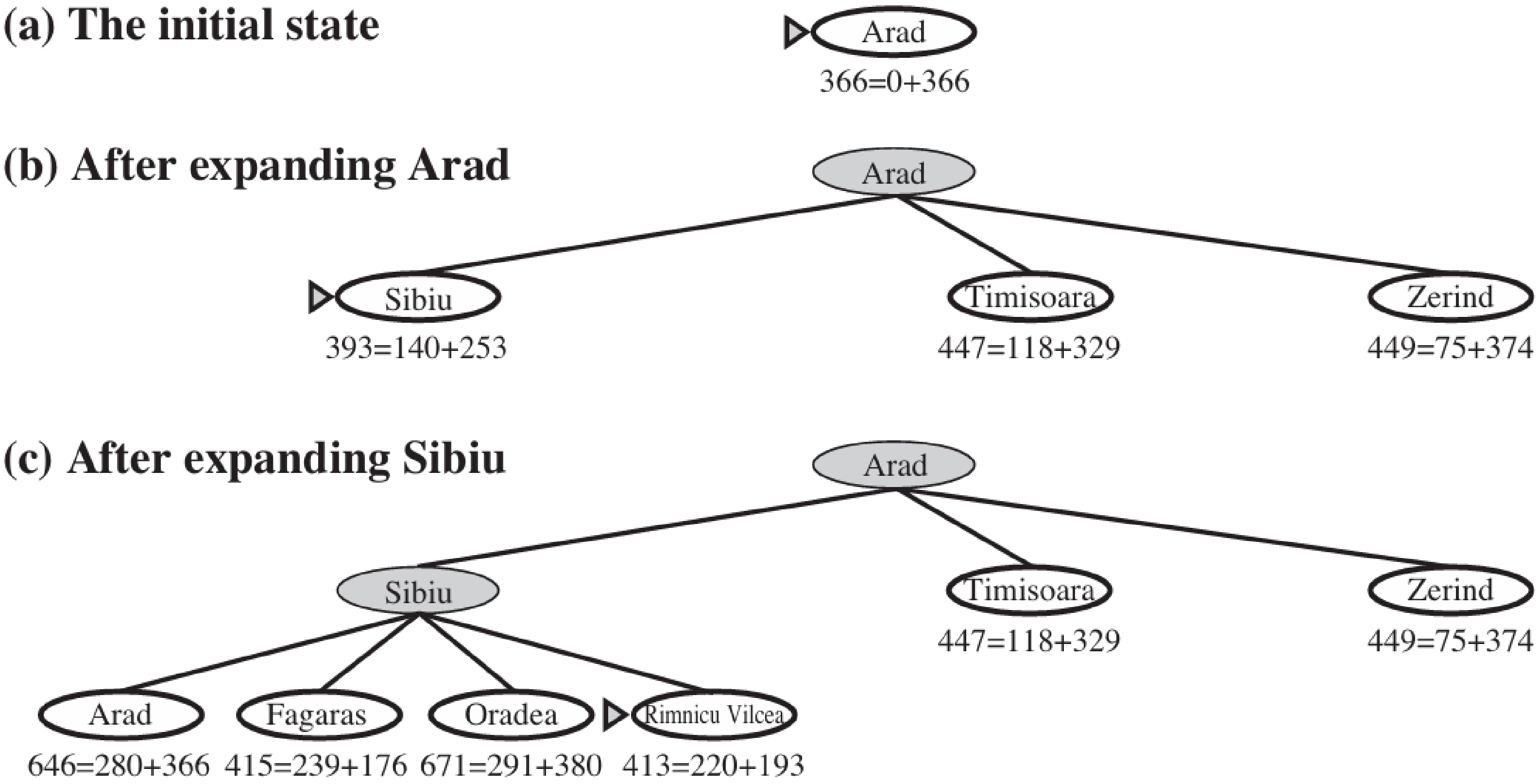
$$f(n) = g(n) + h(n)$$

- Algorithm is identical to uniform cost search but path cost is now f(n)



Arad	366	**Mehadia**	241
Bucharest	0	**Neamt**	234
Craiova	160	**Oradea**	380
Drobeta	242	**Pitesti**	100
Eforie	161	**Rimnicu Vilcea**	193
Fagaras	176	**Sibiu**	253
Giurgiu	77	**Timisoara**	329
Hirsova	151	**Urziceni**	80
Iasi	226	**Vaslui**	199
Lugoj	244	**Zerind**	374

A* Search – Example

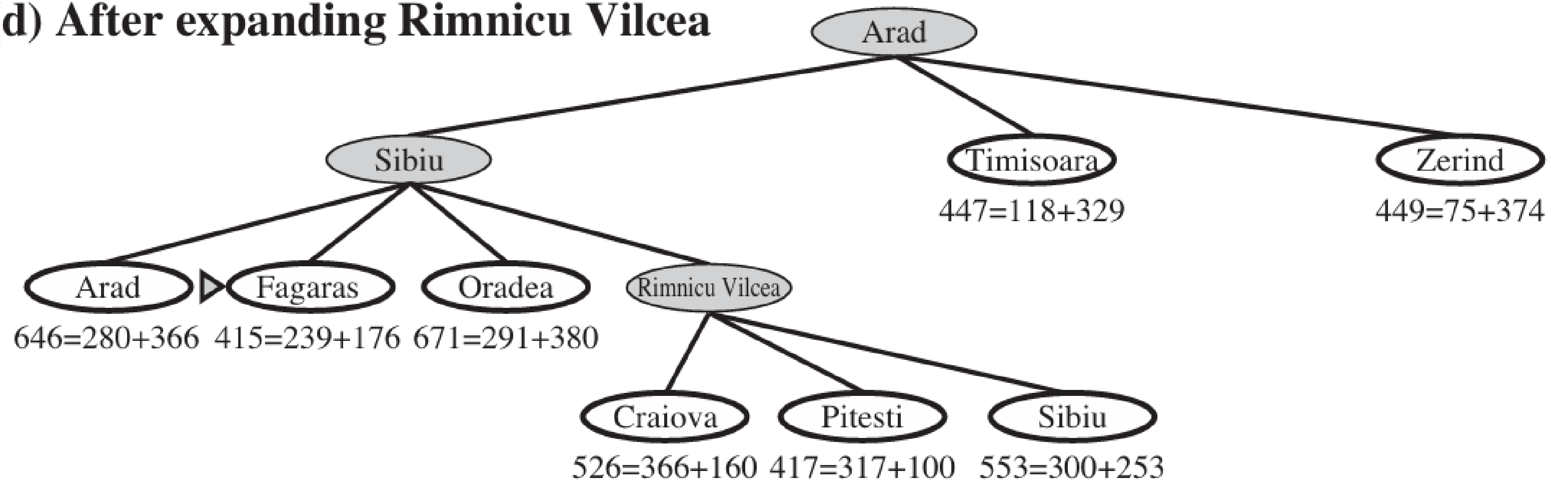


(a) The initial state

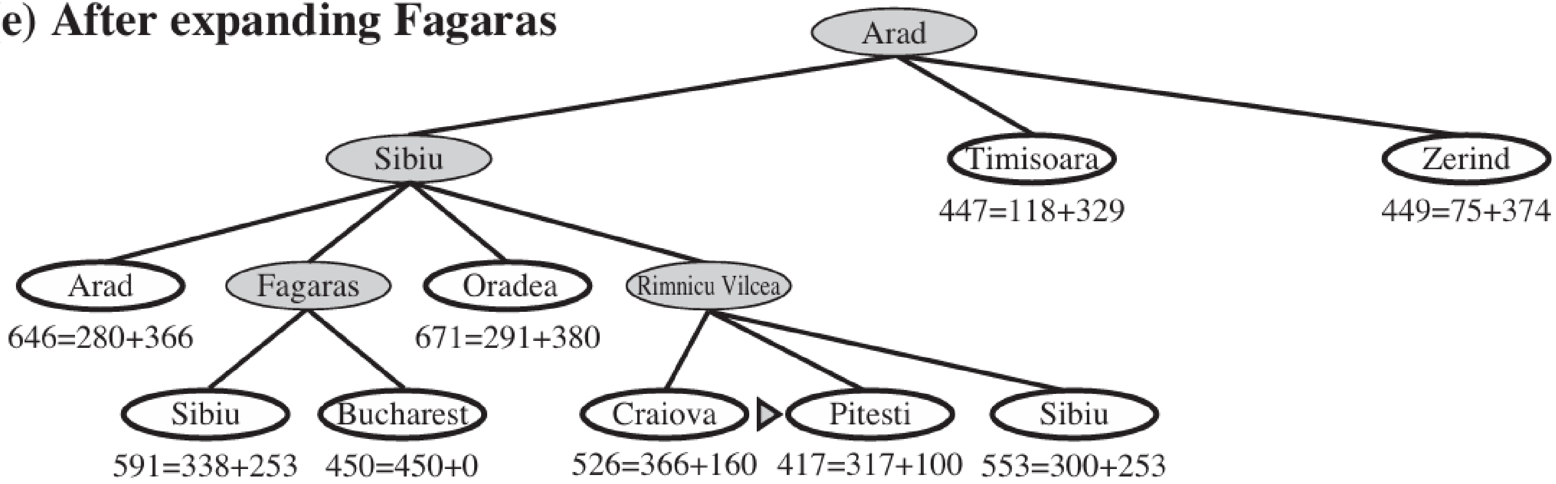
(b) After expanding Arad

(c) After expanding Sibiu

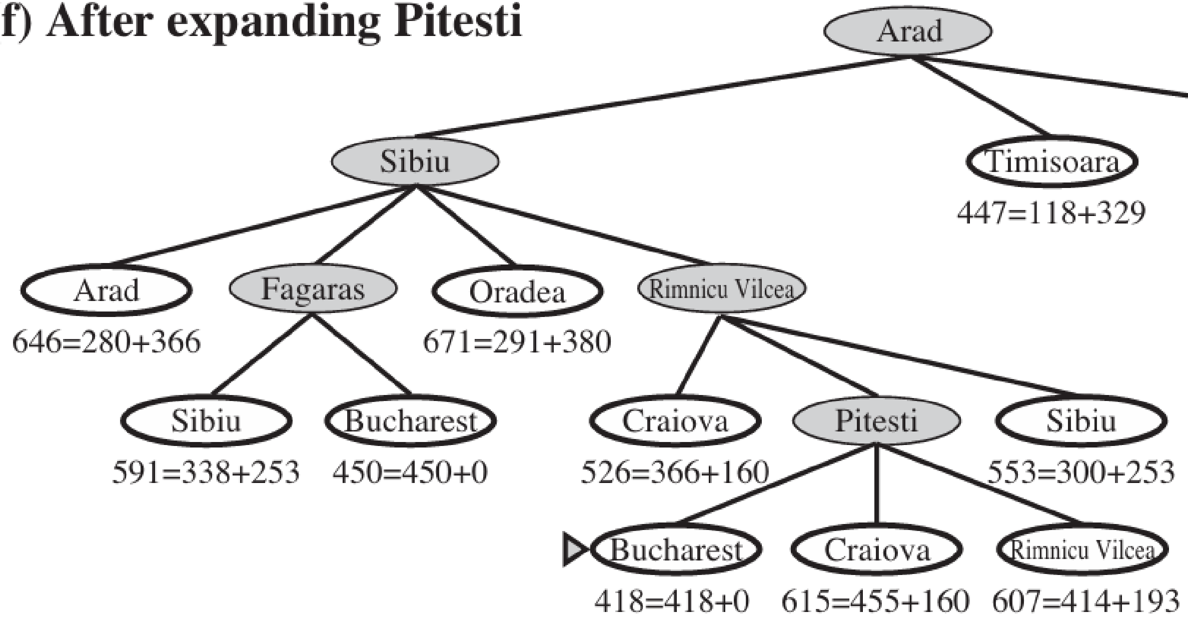
(d) After expanding Rimnicu Vilcea



(e) After expanding Fagaras



(f) After expanding Pitesti



Thank You

Dr. Rajiv Misra, Professor
Dept. of Computer Science & Engg.
Indian Institute of Technology, Patna
rajivm@iitp.ac.in