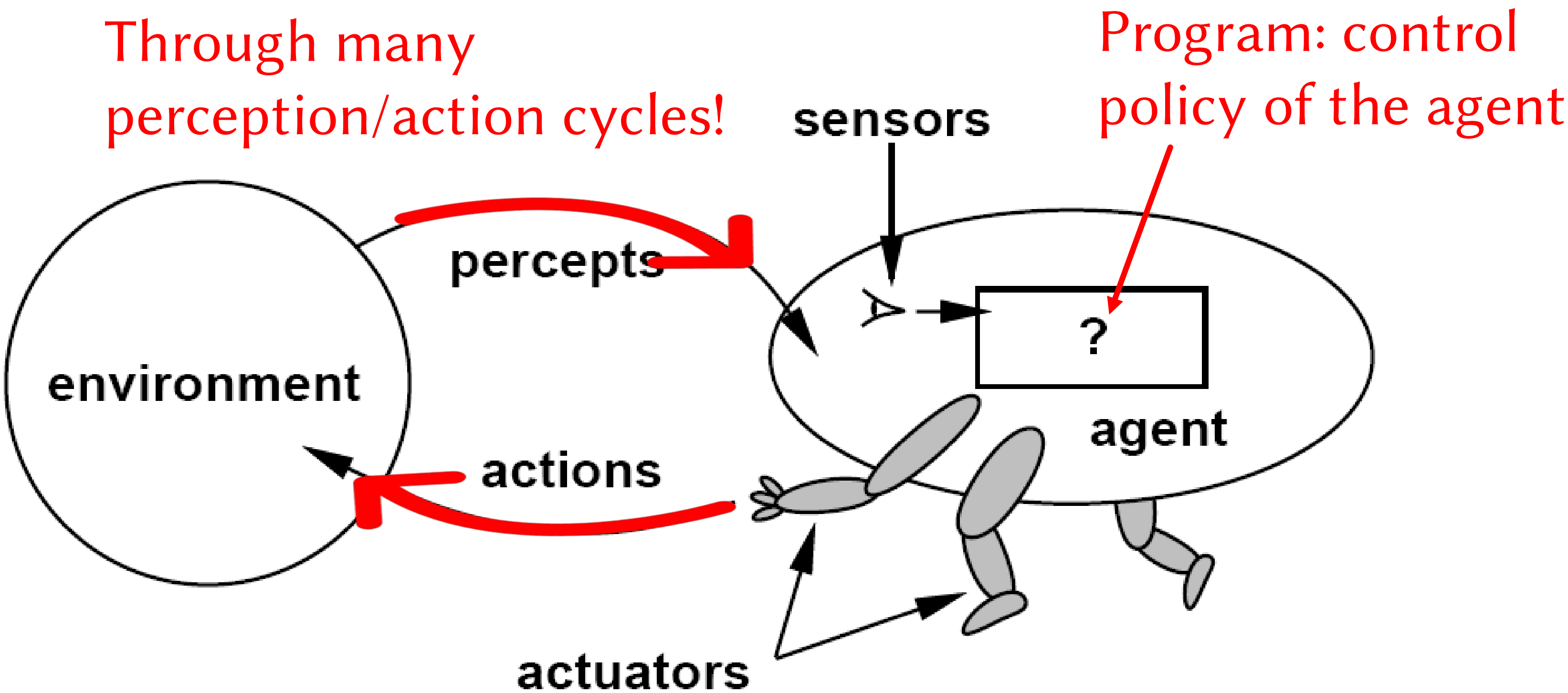
Search (Chapter 3)

Dr. Shengquan Wang

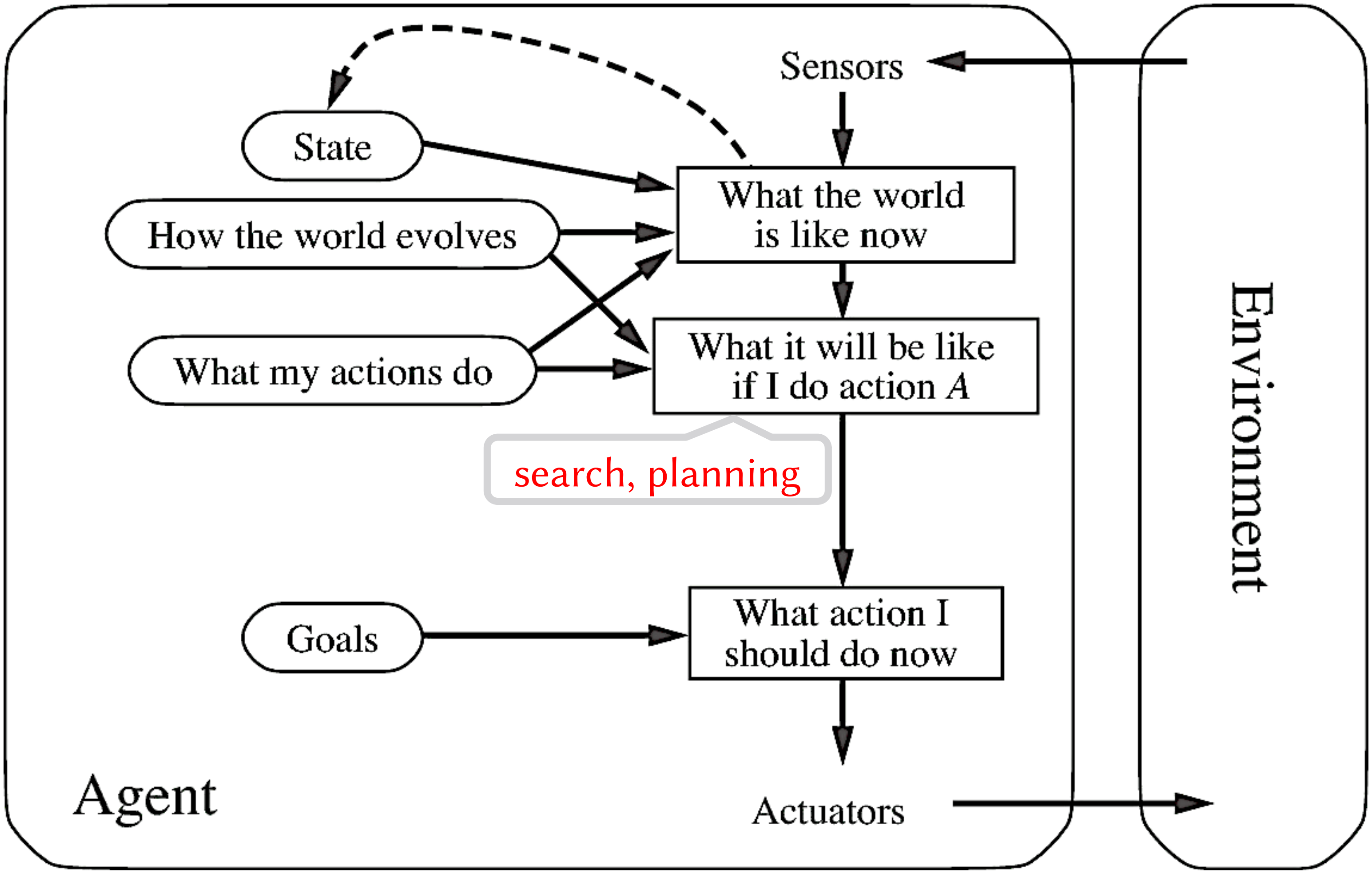
Most slides are adopted from

- Artificial Intelligence: A Modern Approach, 3rd ed. by Stuart Russell (UC Berkeley) and Peter Norvig (Google).
- Peter Norvig and Sebastian Thrun for Intro to Artificial Intelligence at Udacity.
- Dan Klein and Pieter Abbeel for CS188 Intro to AI at UC Berkeley.

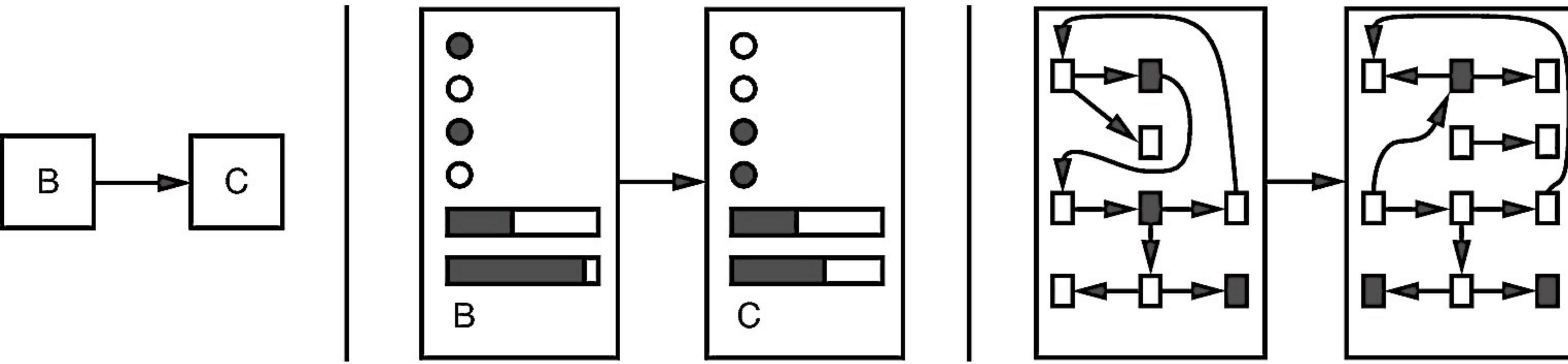
Review: Intelligent Agent



Review: Goal-based Agents



Review: Environment Representation



Atomic Factored Structured

Two Types of Goal-based Agents

Goal-based agents

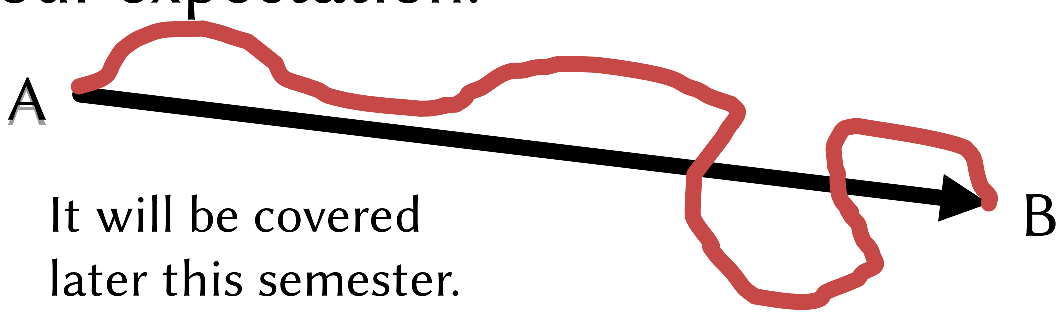
Problem-solving agents | Planning agents

- Problem-solving agents use atomic representations.
- It can be solved by generous-purpose search algorithms.

- Planning agents use more advanced factored or structured representations.

Planning Agents: Uncertainty

- The environment is very complex.
  - Roads are not straight, not flat.
  - Maybe due to road closure, a detour is needed.
- The sensors of the agent come with some measurement errors.
  - Even GPS's error range is 10m.
- The outcome of executing the plan could be different from your expectation.

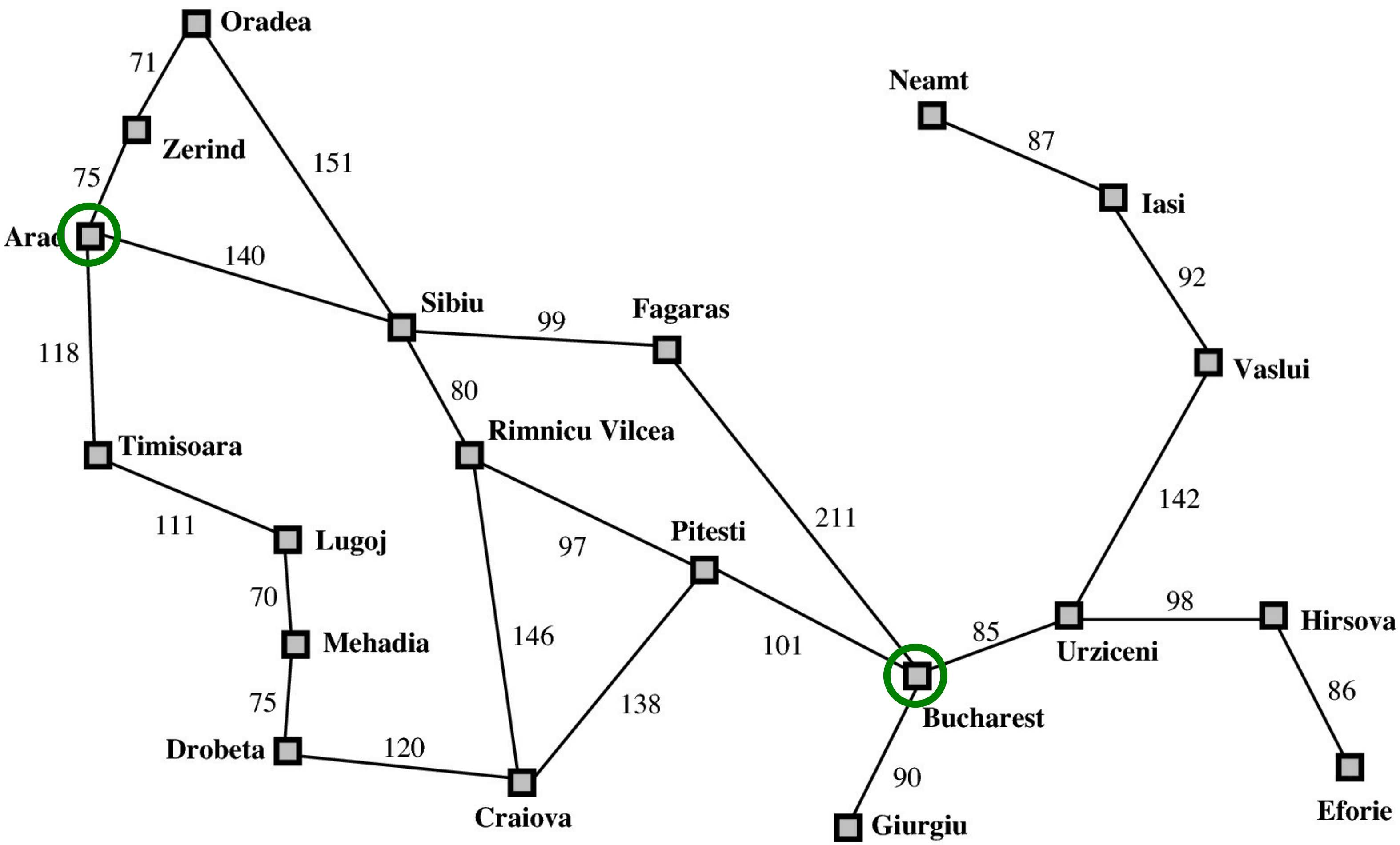


It will be covered
later this semester.

Outline

- Problem-Solving Agents

Example: Traveling in Romania



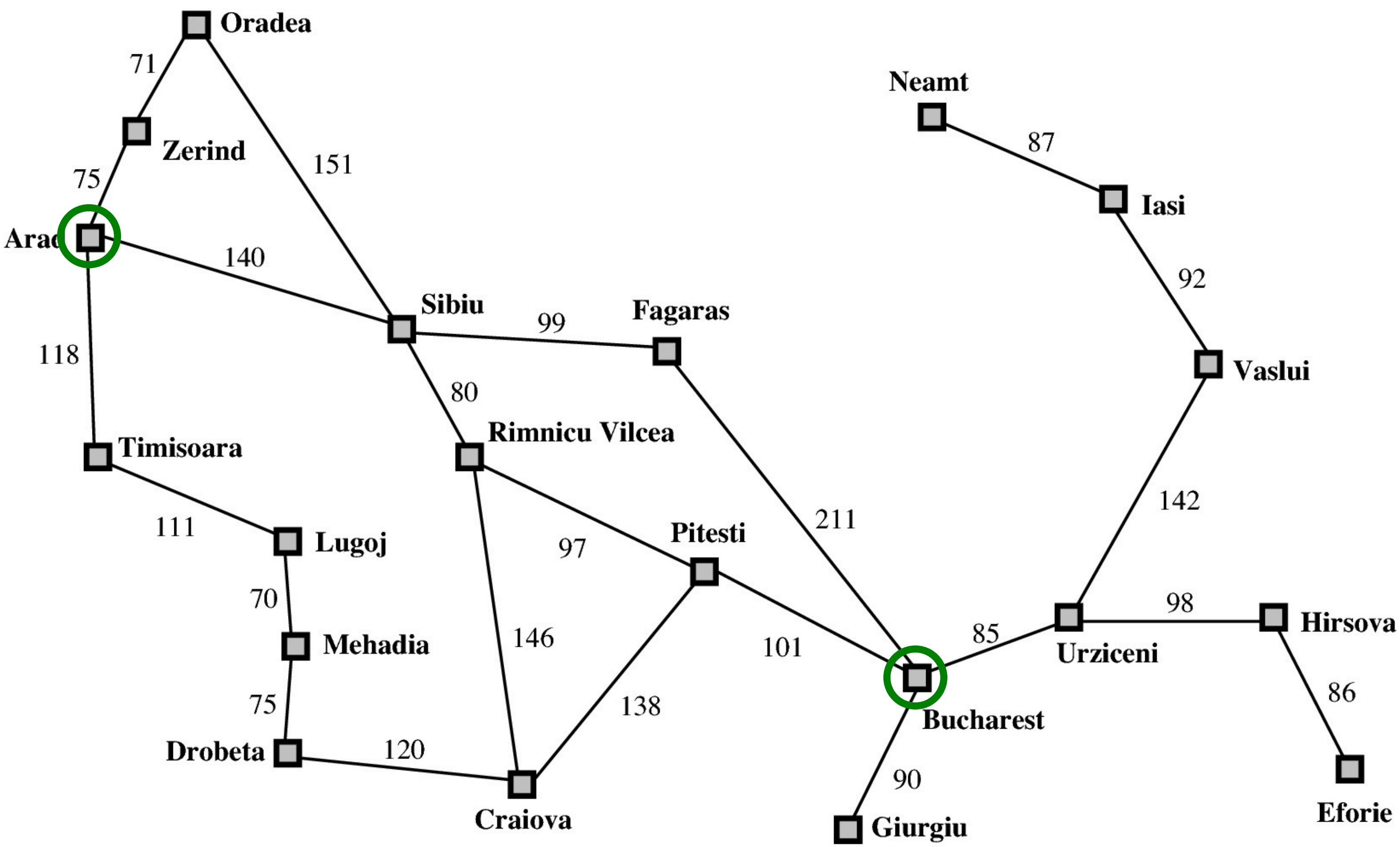
Problem-Solving Agents

A problem can be defined by:

- State space
  - Possible states including initial state and goal state
  - Possible actions
  - Transition model (what each action does)
- Goal test:
  - To determine if a given state is a goal state
- Path cost:
  - Summation of step cost

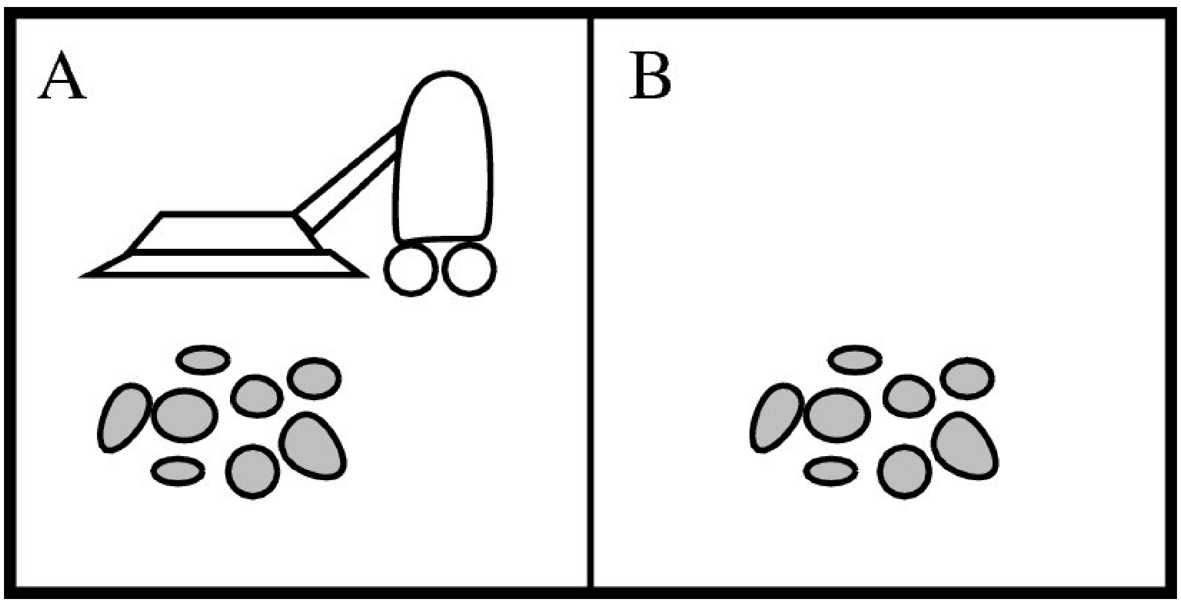
Example: Traveling in Romania

States	Actions	Goal test	Path costs
Cities	Drive between cities	?= Bucharest	Link costs



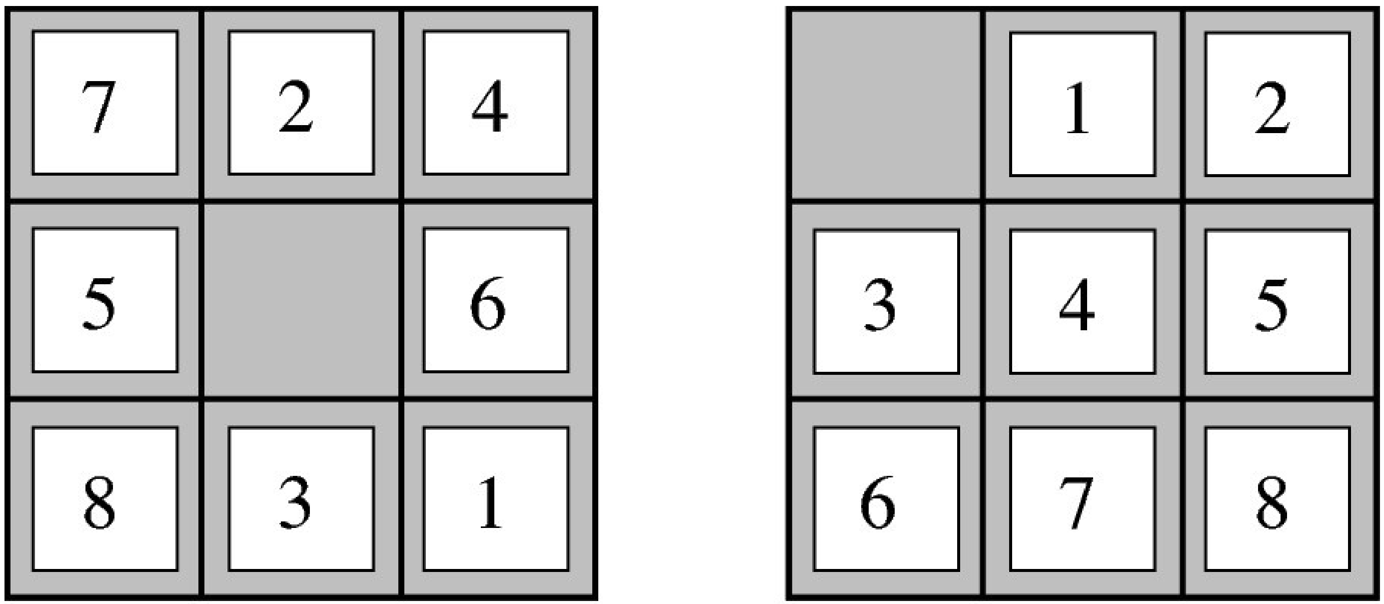
Example: Vacuum World

States	Actions	Goal test	Path costs
Dirt and robot locations	Left, Right, Suck, NoOp	No dirt	1 per action (0 for NoOp)



Example: 8-Puzzle

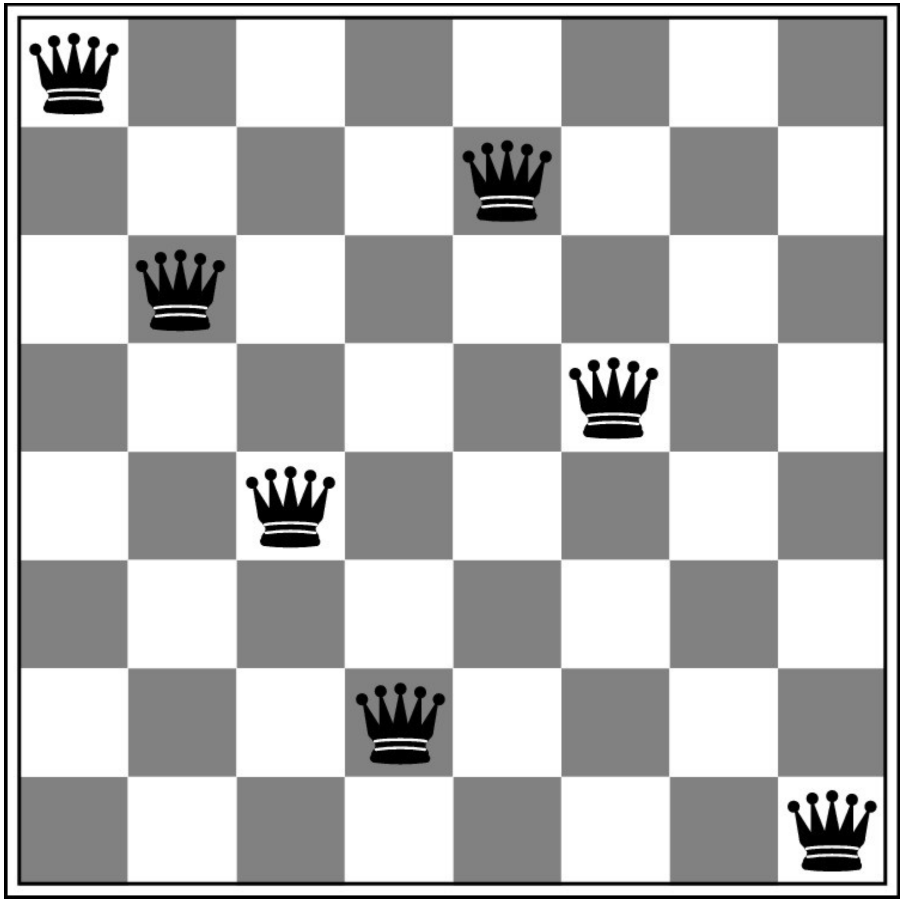
States	Actions	Goal test	Path costs
Locations of tiles	Move blank left, right, up, down	Given	1 per move



Start State Goal State

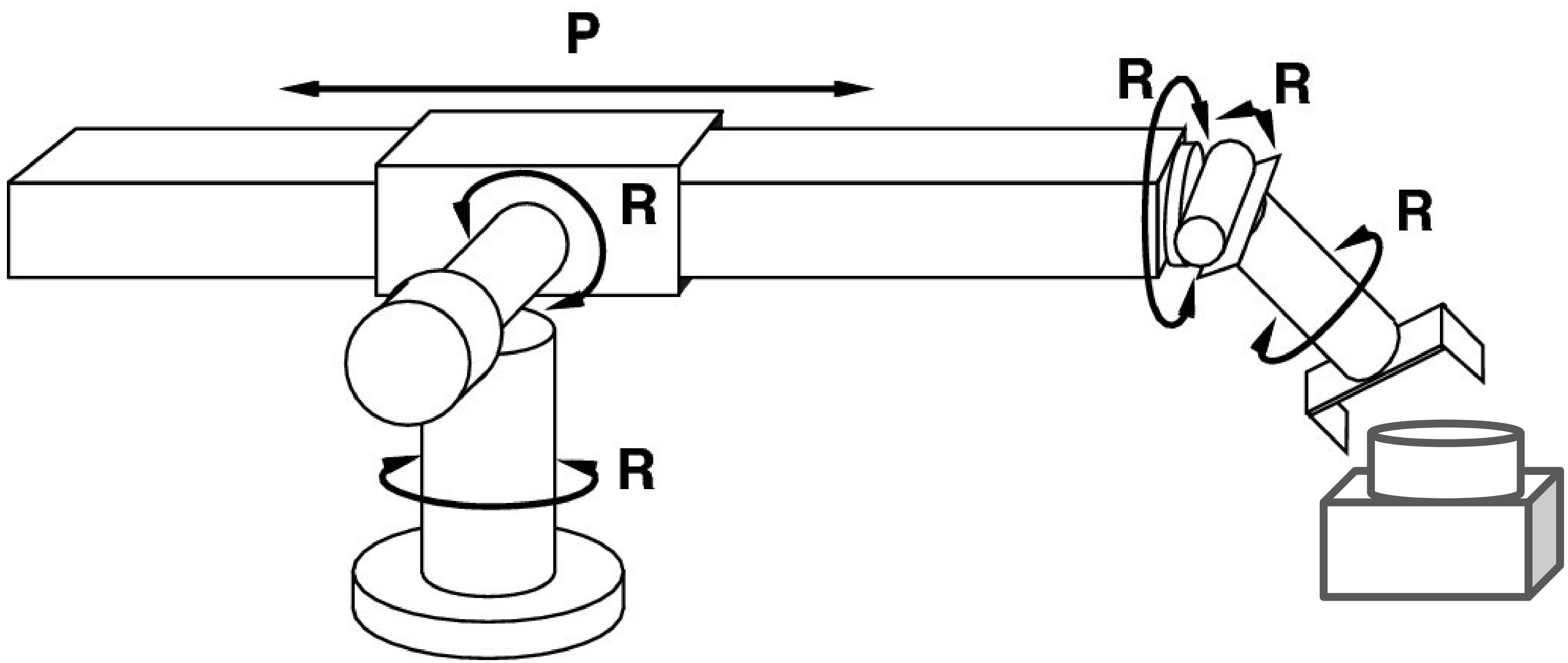
Example: 8-Queen

States	Actions	Goal test	Path costs
Any arrangement of 0 to 8 queens on the board	Add a queen to any empty square	8 queens are on the board, none attacked	N/A



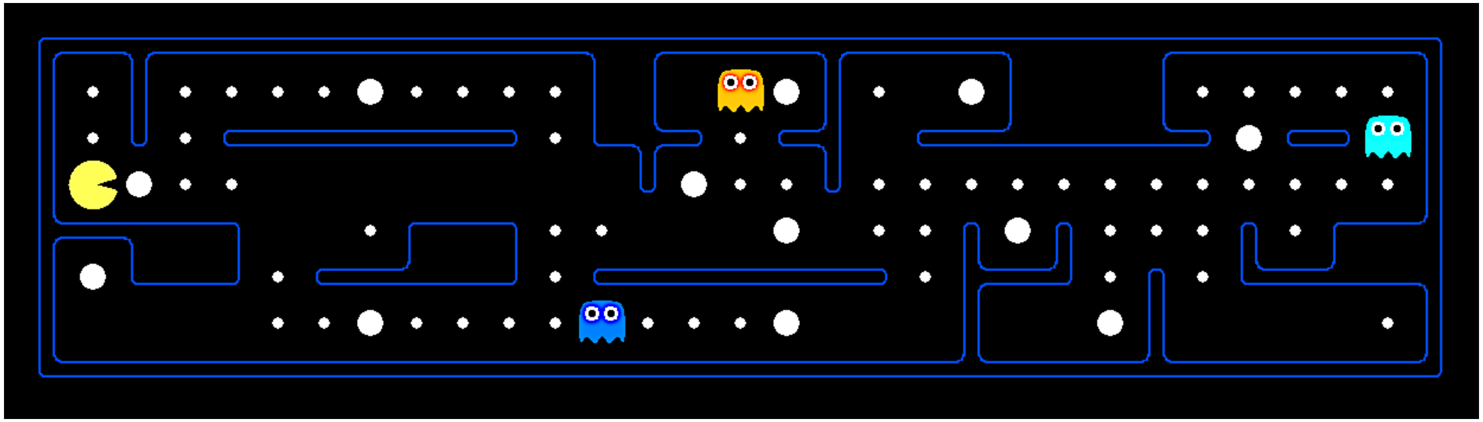
Example: Robotic Assembly

States	Actions	Goal test	Path costs
Real-valued coordinates of robot joint angles; Parts of the object to be assembled	Continuous motions of robot joints	Complete assembly of all parts of the object!	Time to execute



Example: Pac-Man

States	Actions	Goal test	Path costs
Position of Pac-Man, Boolean dots, ghost position	Left, right, up, down	All dots are eaten	1 per eating a dot, dead after eaten by a ghost



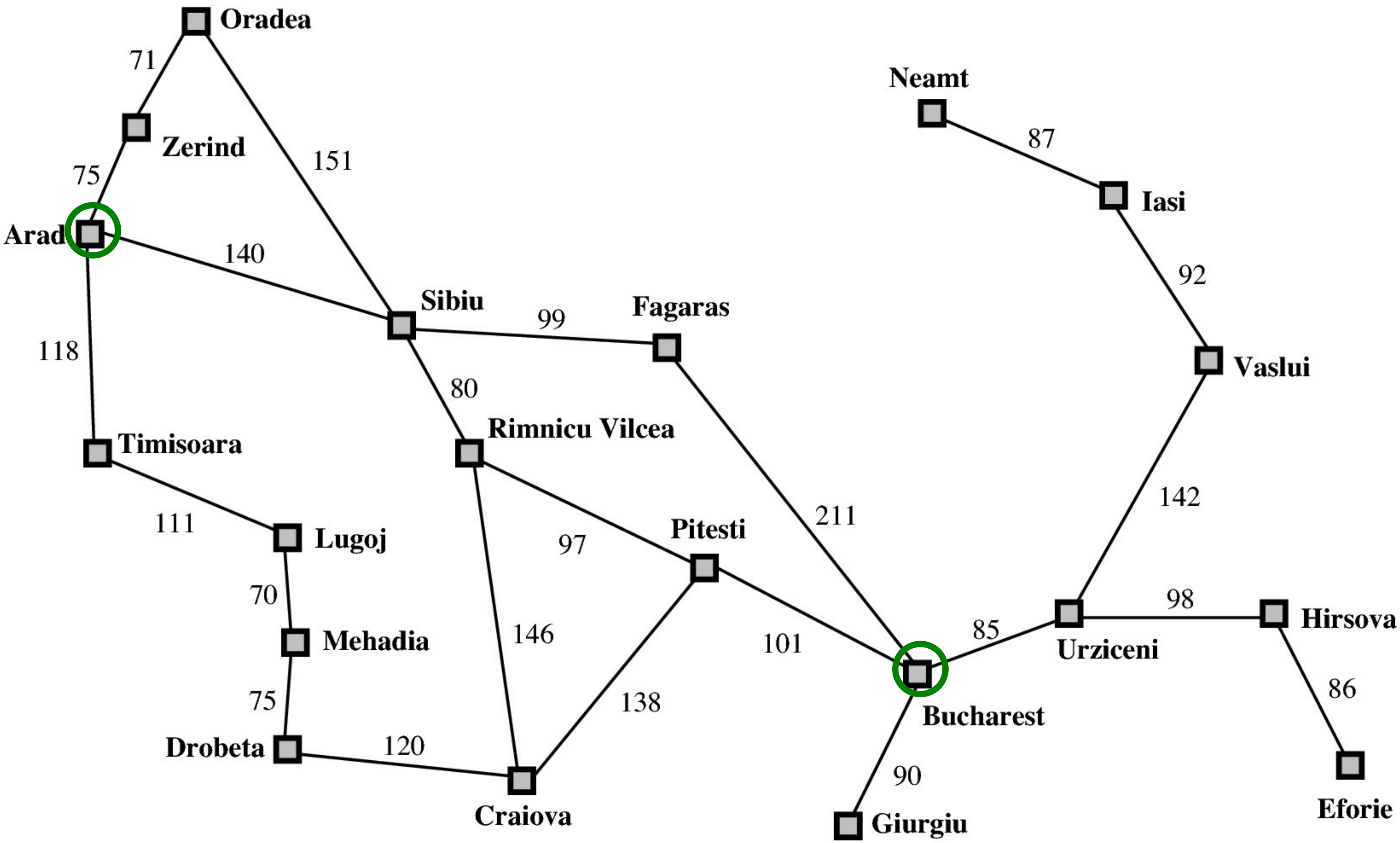
Outline

- Problem-Solving Agents
- Searching for Solution

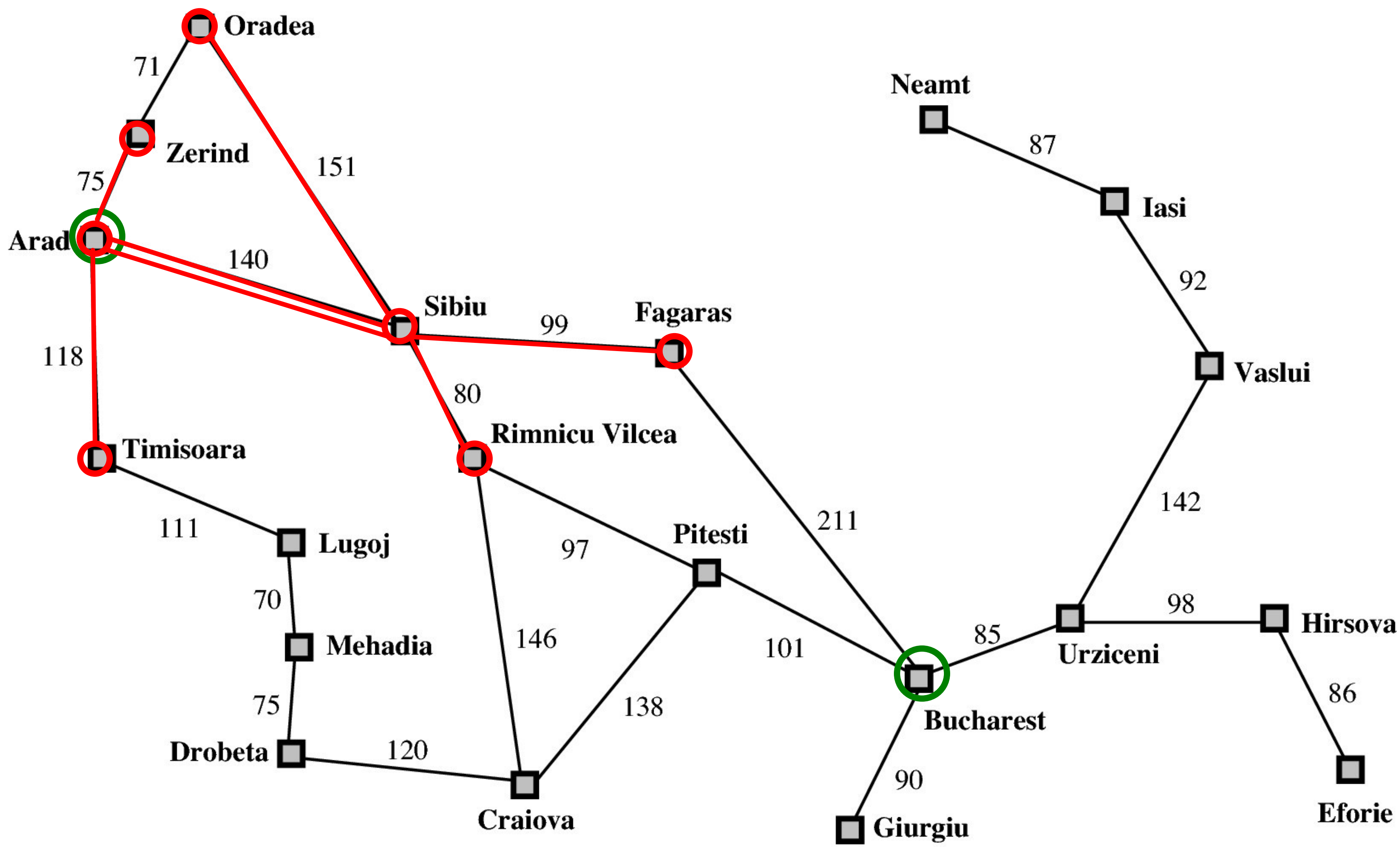
Solution

- A solution is an action sequence.
- So search algorithms work by considering various possible action sequences.
- The possible action sequences starting at the initial state form a search tree:
  - the initial state is at the root;
  - the branches are actions;
  - the nodes correspond to states in the state space of the problem.

Example: Traveling in Romania



Example: Traveling in Romania

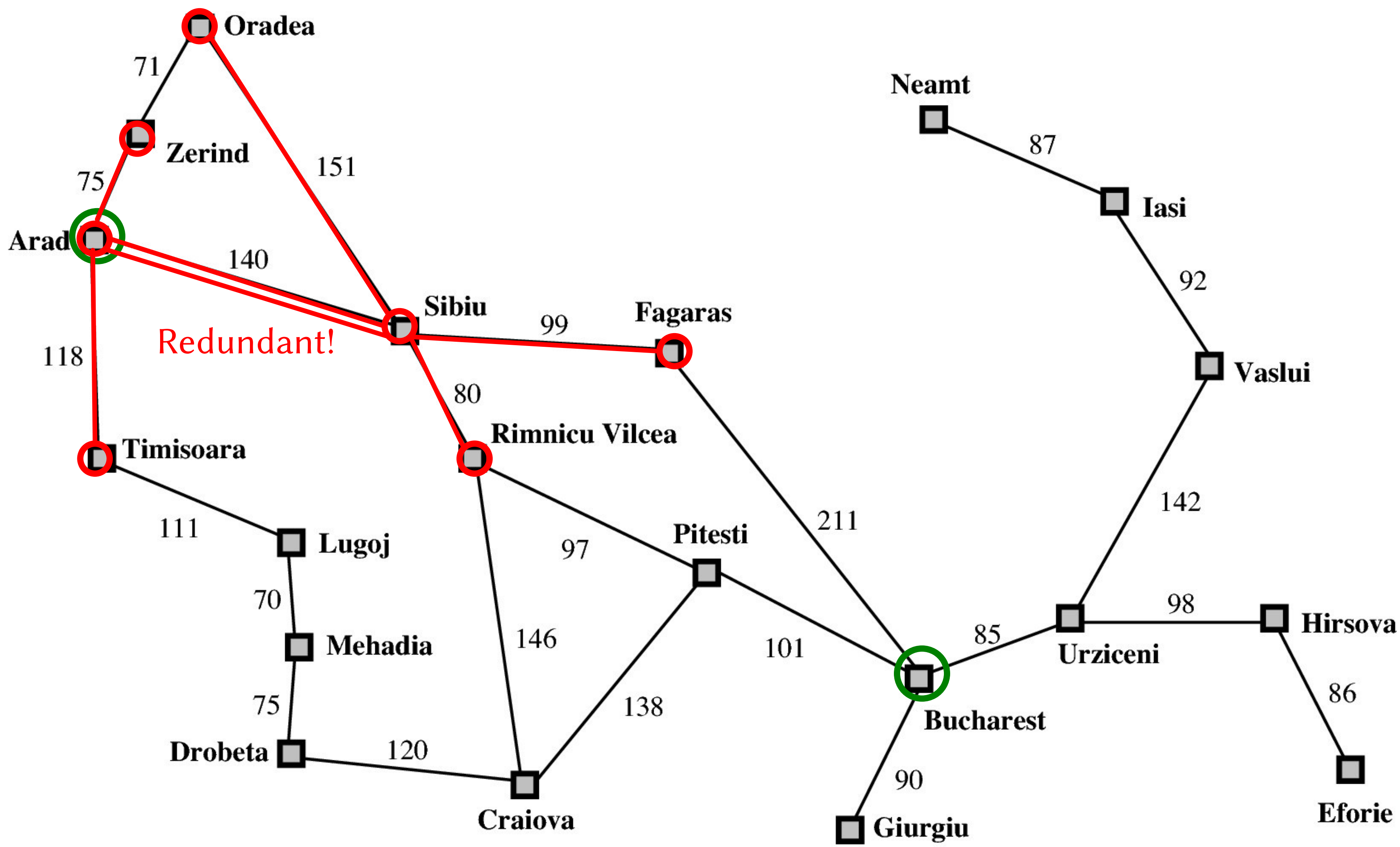


Tree Search Algorithms

```
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
```

- Frontier: the set of all leaf nodes available for expansion at any given point.

Example: Traveling in Romania

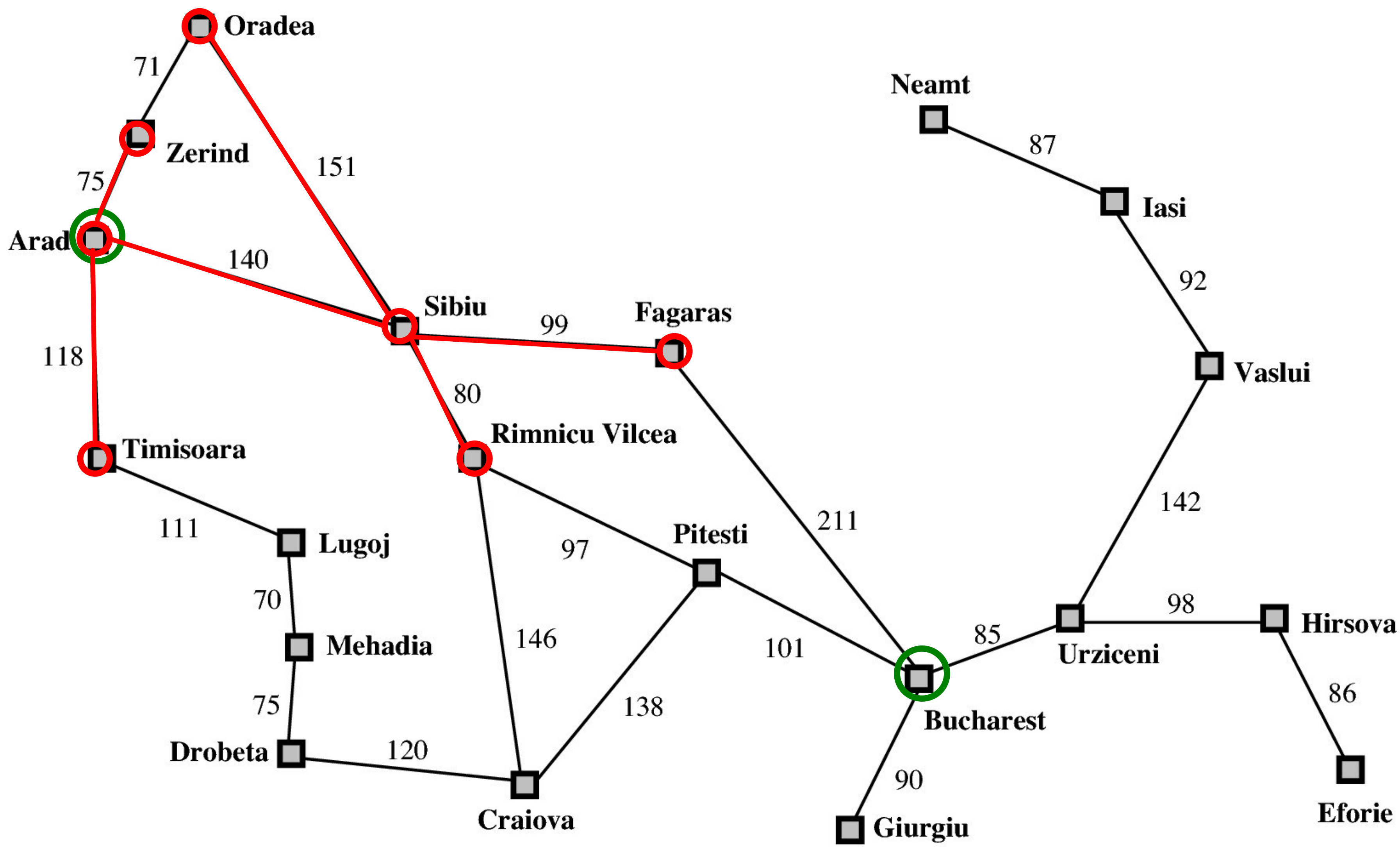


Graph Search Algorithms

function GRAPH-SEARCH(*problem*) **returns** a solution, or failure
  initialize the frontier using the initial state of *problem*
  initialize the explored set to be empty
  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
    add the node to the explored set
    expand the chosen node, adding the resulting nodes to the frontier
      only if not in the frontier or explored set

- Use explored set to remember every expanded node to avoid redundant paths.

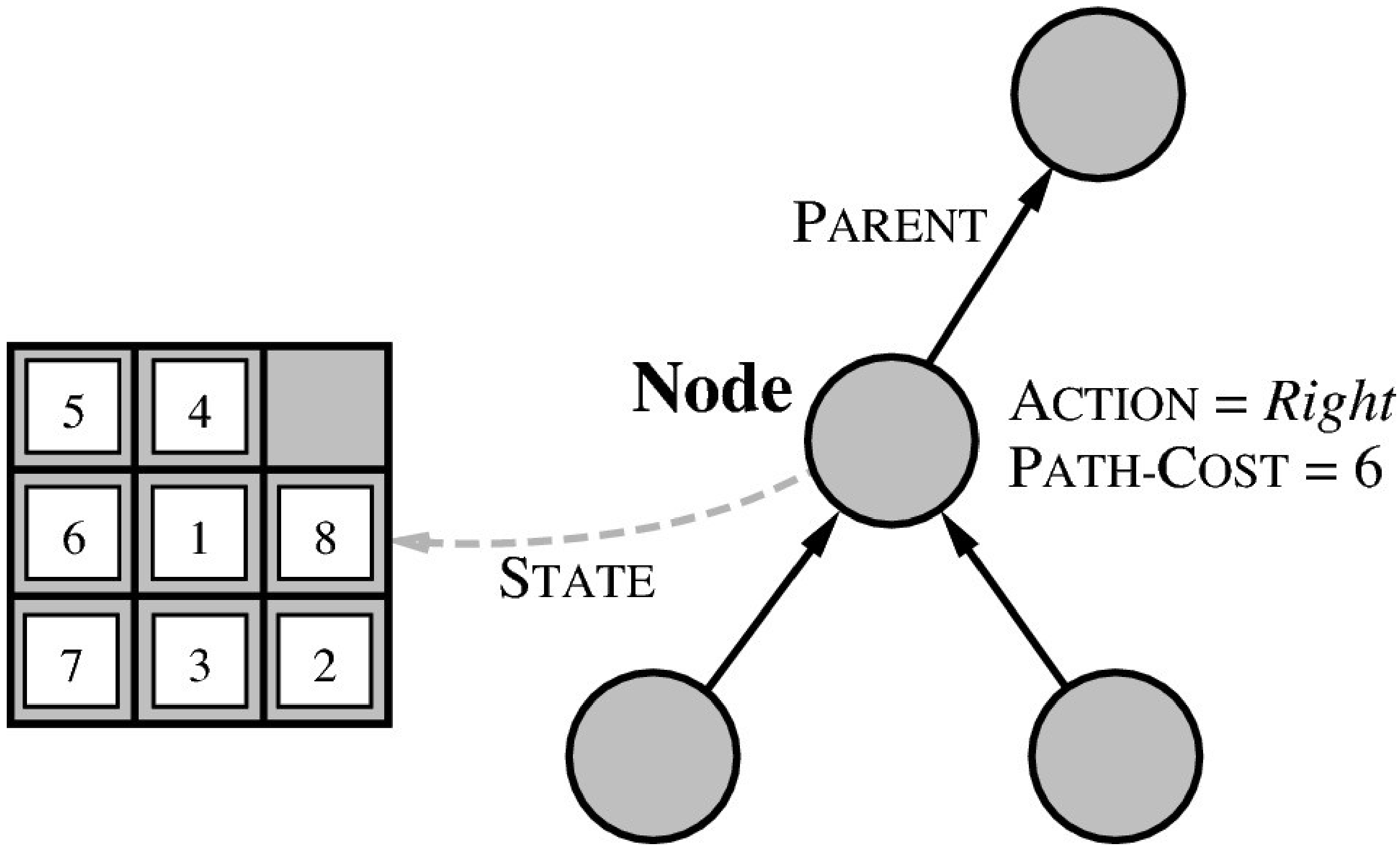
Example: Traveling in Romania



Implementation: Node

STATE	the state in the state space to which the node corresponds
PARENT	the node in the search tree that generated this node
ACTION	the action that was applied to the parent to generate the node
PATH-COST	the cost of the path from the initial state to the node, as indicated by the parent pointers

Example: 8-Puzzle

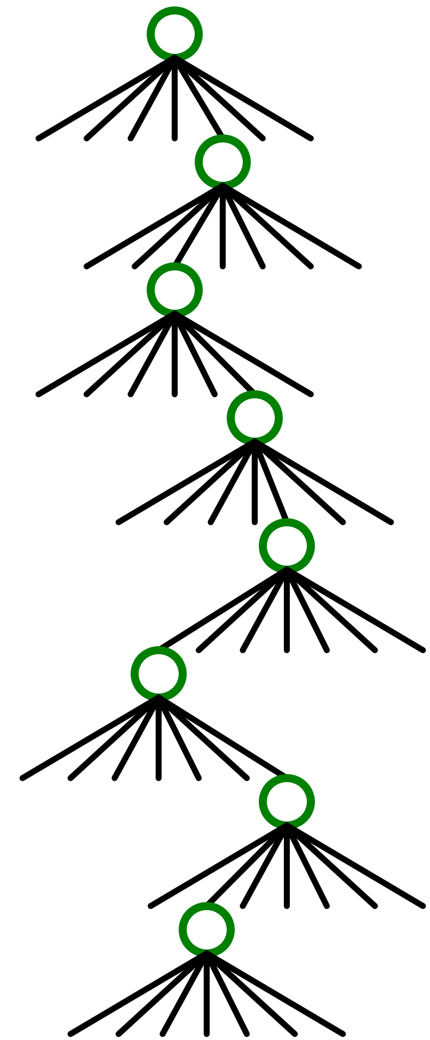


Implementation: Frontier/Explored Sets

- The frontier as a:
  - FIFO queue: it pops the oldest element;
  - LIFO stack: it pops the newest element;
  - Priority queue: it pops the highest-priority element according to some ordering function.
- The explored set as a hash table.

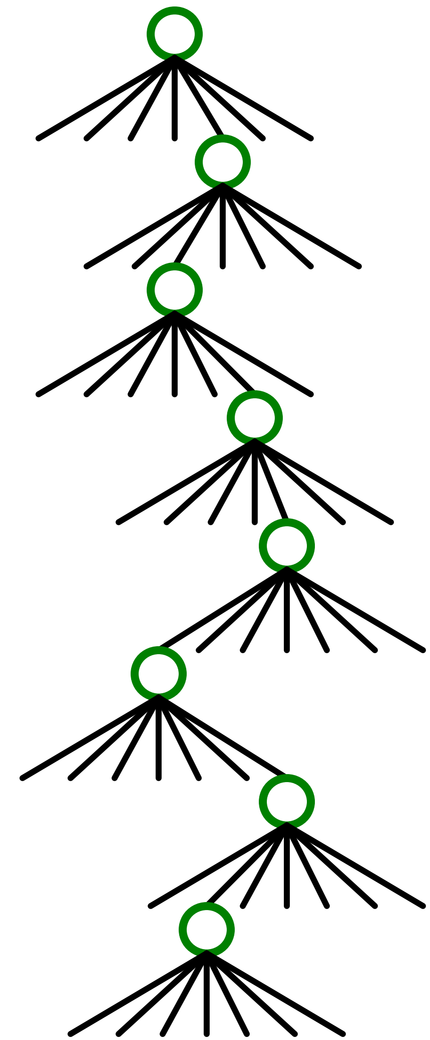
Search Strategies

- A strategy is defined by picking the order of node expansion.
- Strategies are evaluated along the following dimensions:
  - Completeness: does it always find a solution if one exists?
  - Time complexity: number of nodes generated/expanded
  - Space complexity: maximum number of nodes in memory
  - Optimality: does it always find a least-cost solution?



Search Strategies (cont'd)

- 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



Two Categories of Search

- Uninformed search (or blind search):
  - The strategies have no idea about which successor is promisingly closer to the goal state.
  - All they can do is to generate successors and distinguish a goal state from a non-goal state.
- Informed search (or heuristic search):
  - The strategies have some idea about which successor is promisingly/heuristically closer to the goal state.

Outline

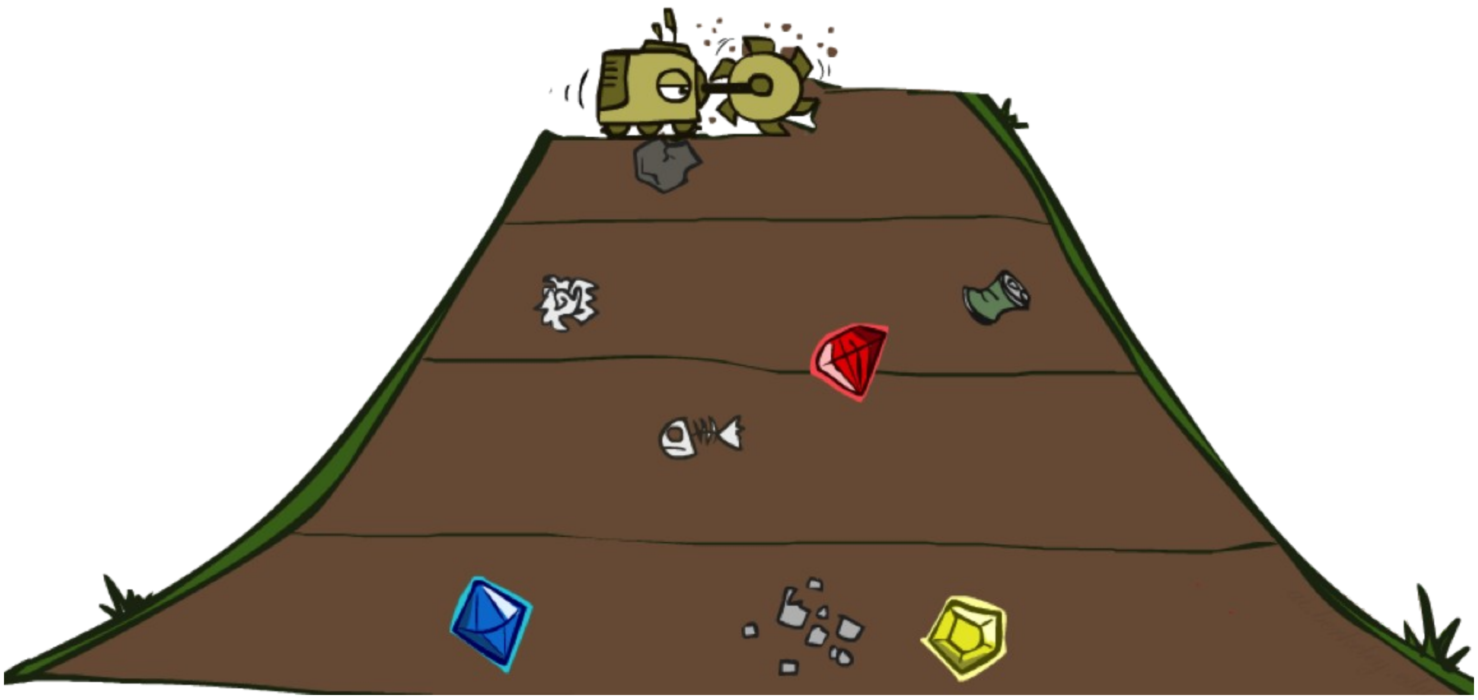
- Problem-Solving Agents
- Searching for Solution
- Uninformed Search

Uninformed Search Algorithms

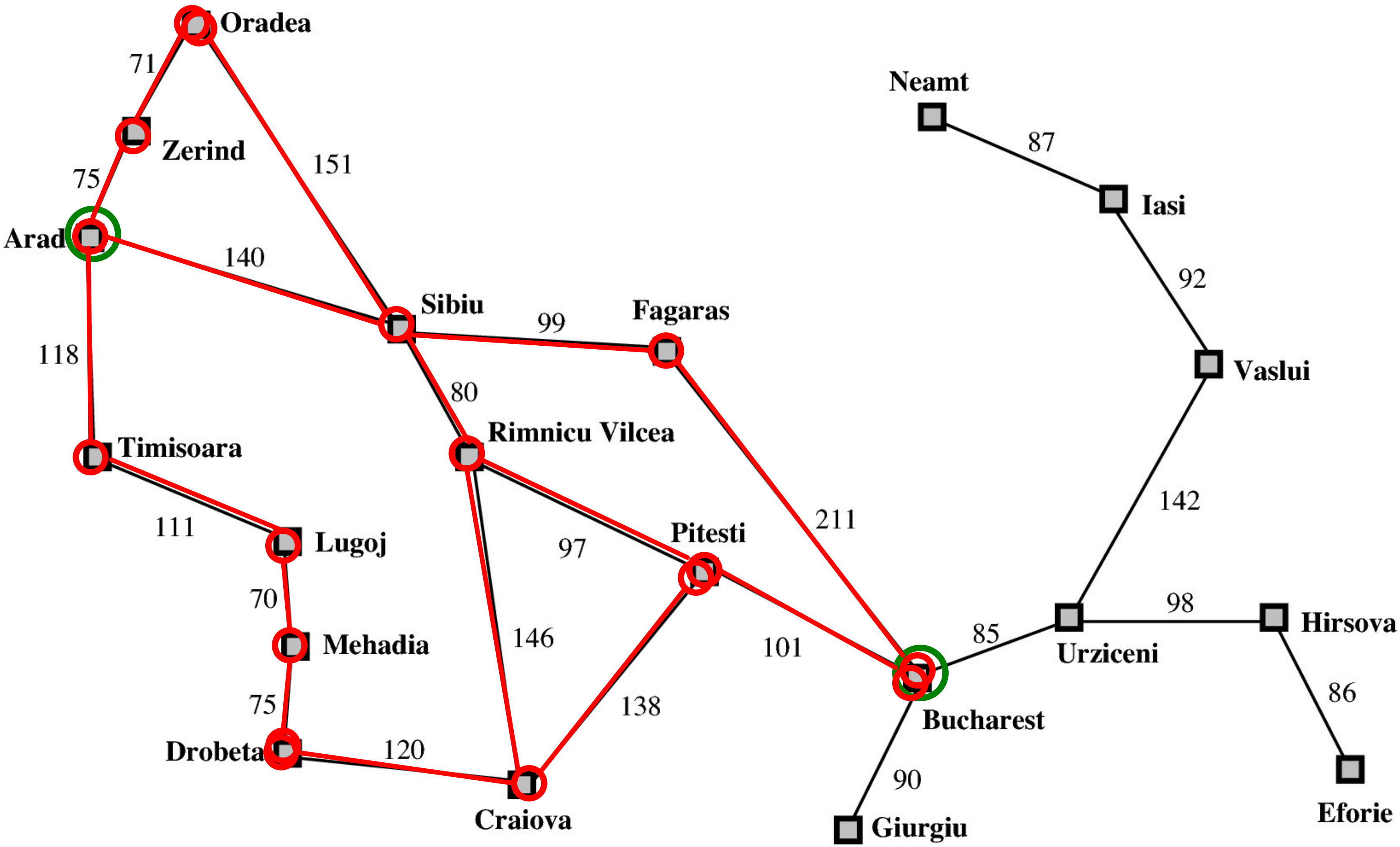
- Uninformed search strategies can be further distinguished by the order in which nodes in the frontier are expanded.
  - Breadth-First Search
  - Uniform-Cost Search
  - Depth-First Search

Breadth-First Search (BFS)

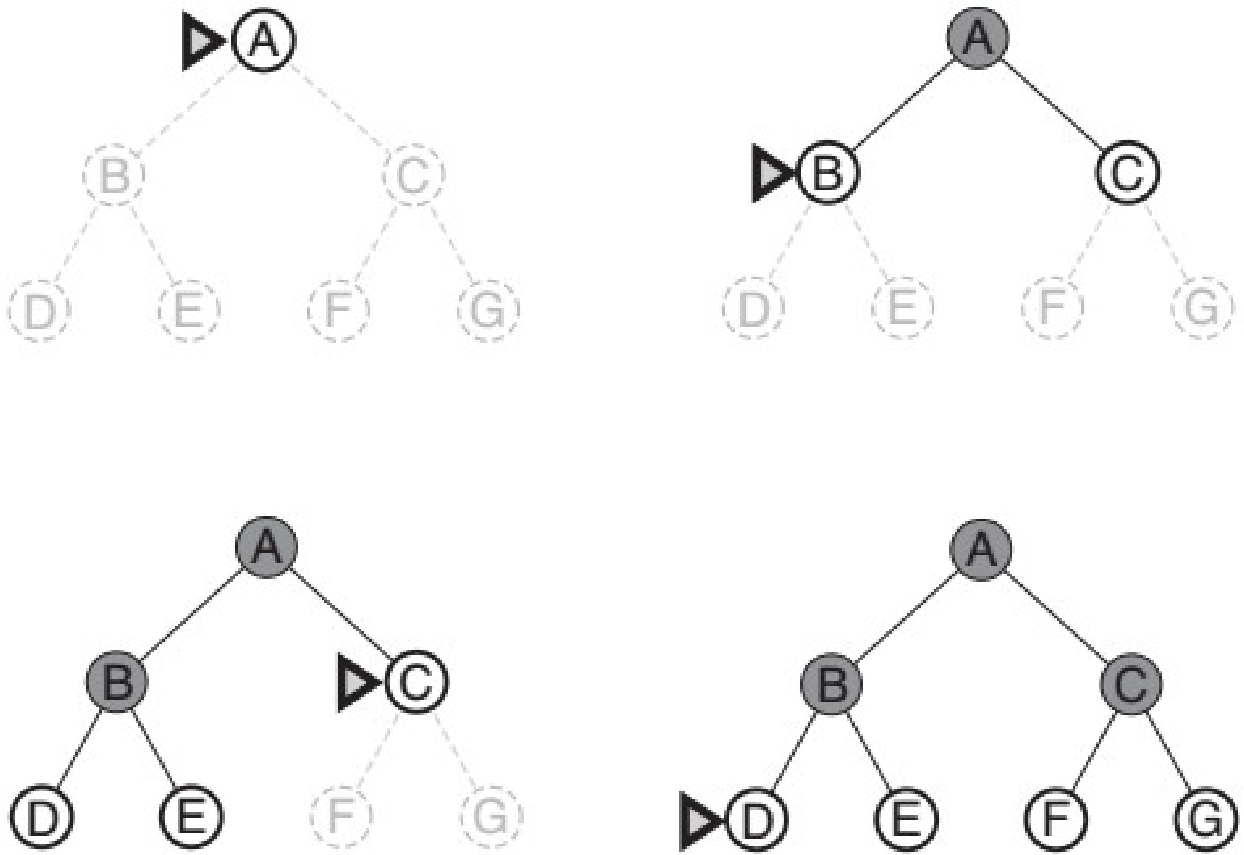
- Strategy: Expand a shallowest node first
- Implementation: Frontier is a FIFO queue



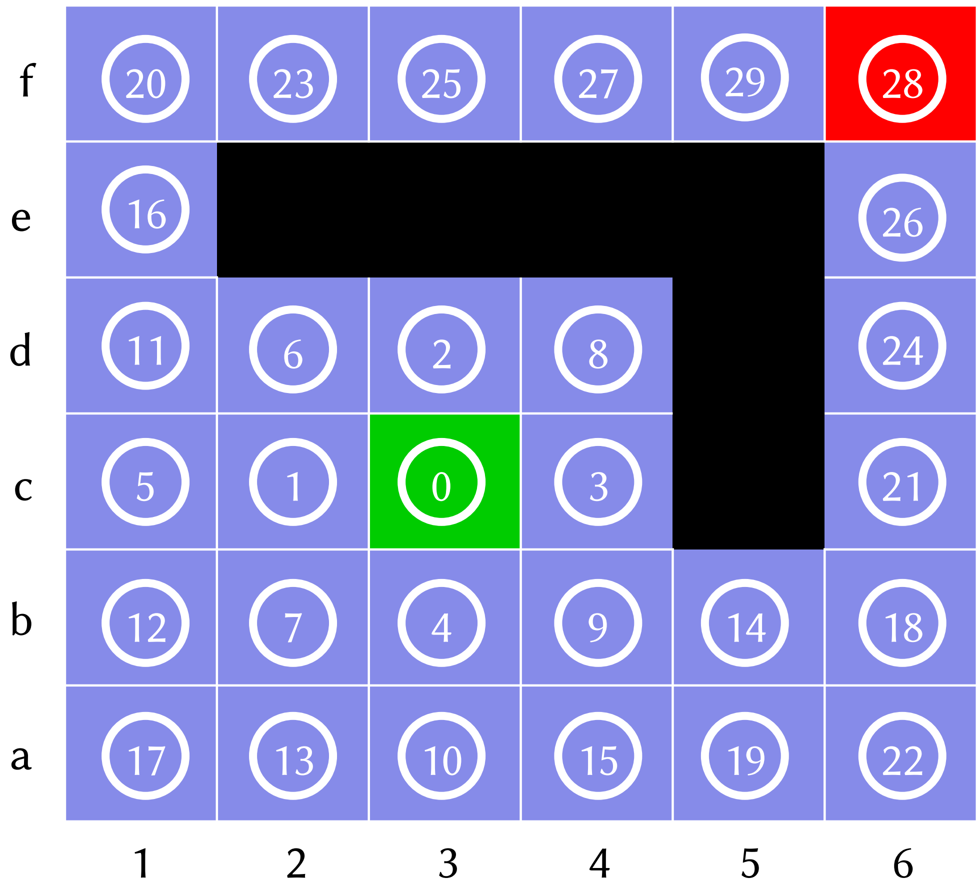
Example: Traveling in Romania



Example: BFS on a Simple Binary Tree

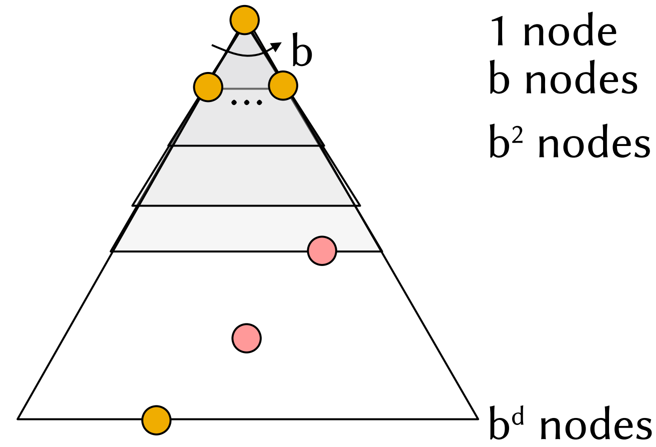


Quiz: BFS on a Maze



Label inside each circle is the order of choosing this square and adding it into the frontier.

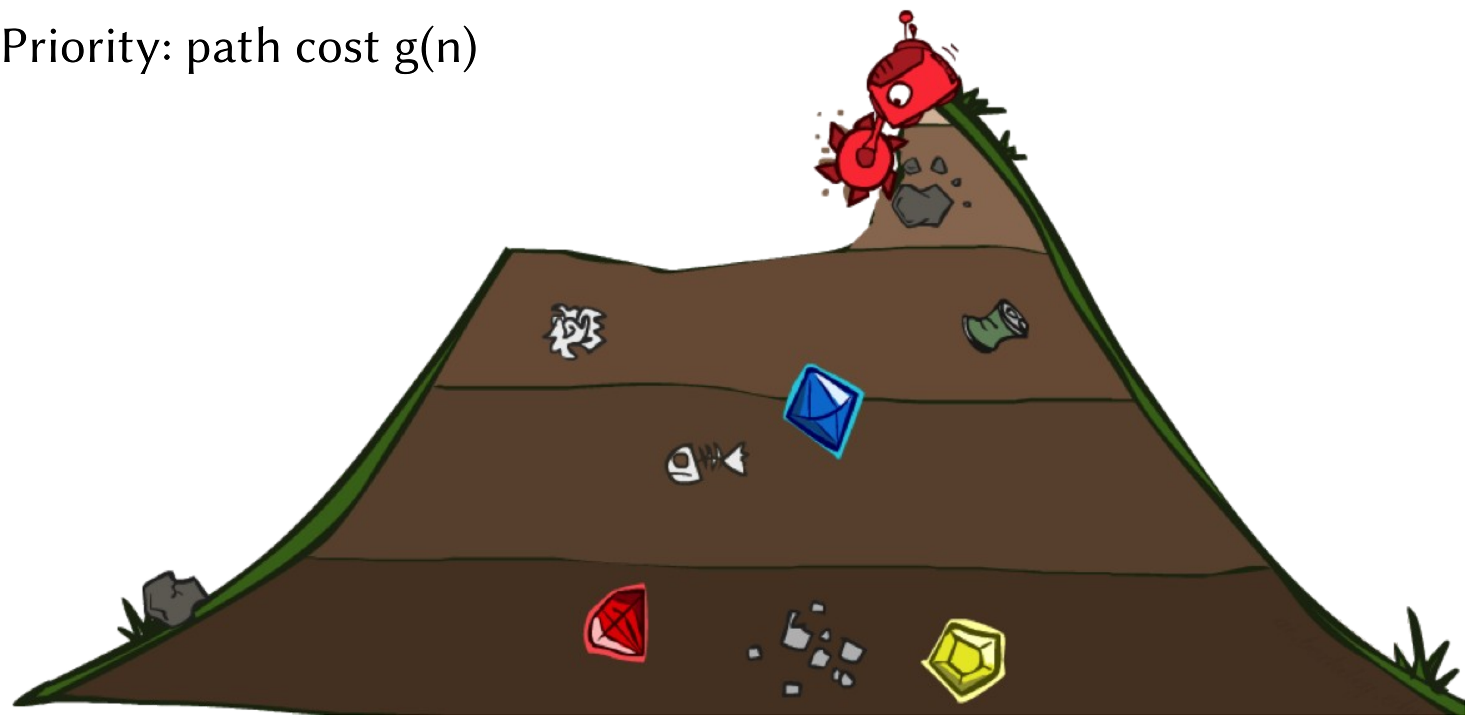
BFS Properties



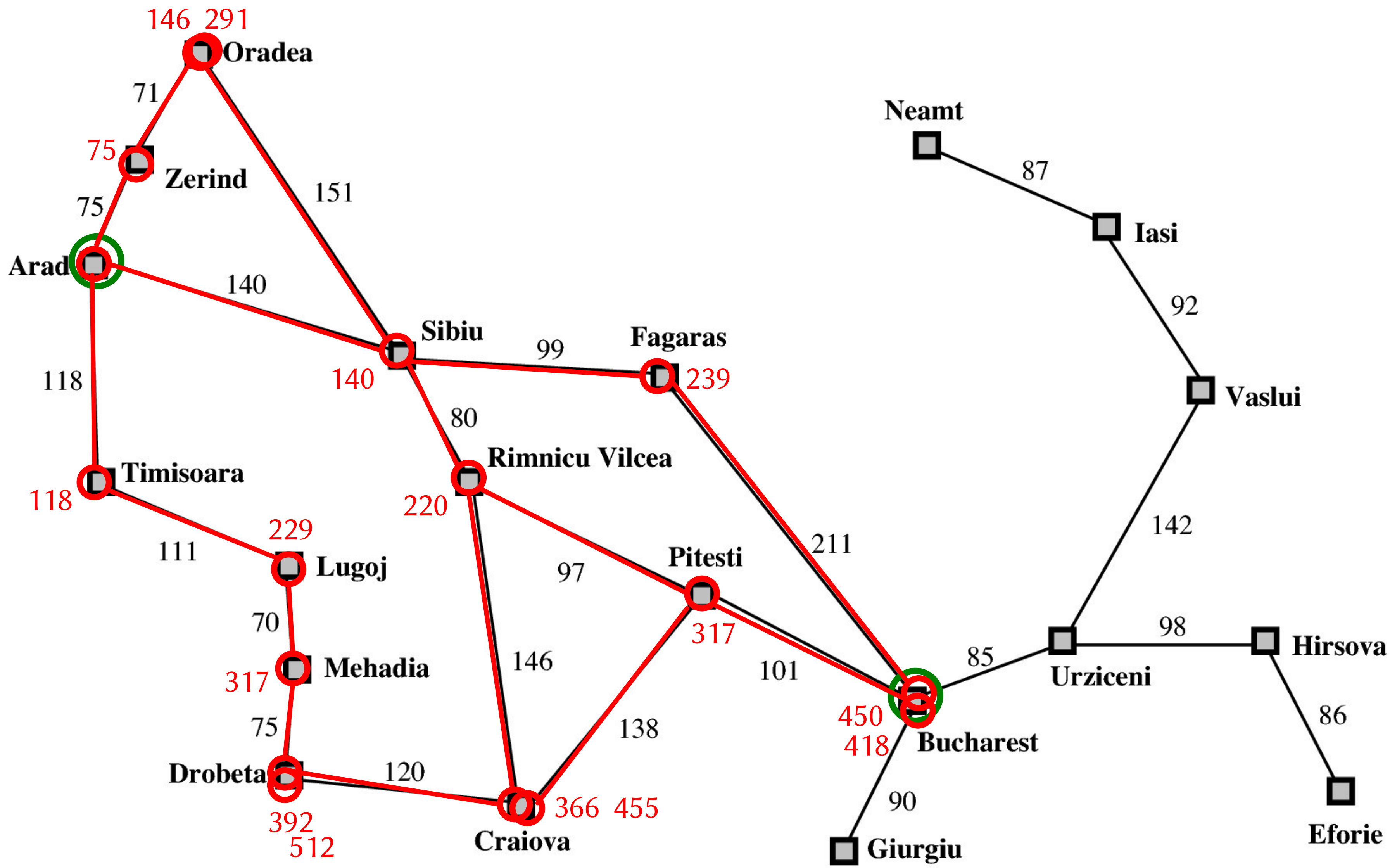
Complete	Yes (if b is finite)
Time	$1 + b + b^2 + b^3 + ::: + b^d + b(b^d-1) = O(b^{d+1})$
Space	• $O(b^{d-1})$ for the explored set; • $O(b^d)$ for the frontier set • Problem: it can easily generate nodes at 100MB/sec, so 24hrs = 8640GB.
Optimal	• Yes (if cost = 1 per step) • Not optimal in general

Uniform Cost Search (UCS)

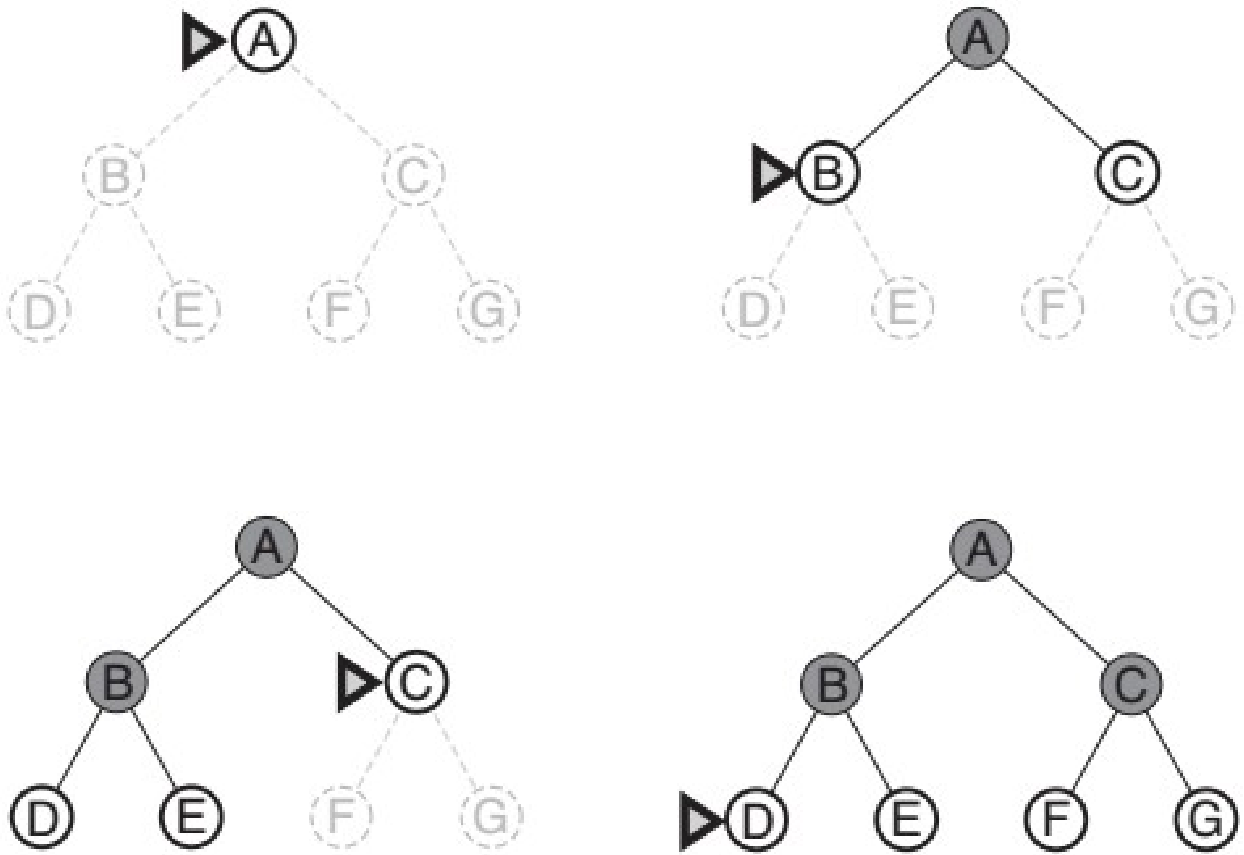
- It is also called Dijkstra's algorithm by theoretical computer scientists.
- Strategy: expand a least-cost unexpanded node first.
  - Path-cost function: g(n)
- Implementation: Frontier is a priority queue
  - Priority: path cost g(n)



Example: Traveling in Romania

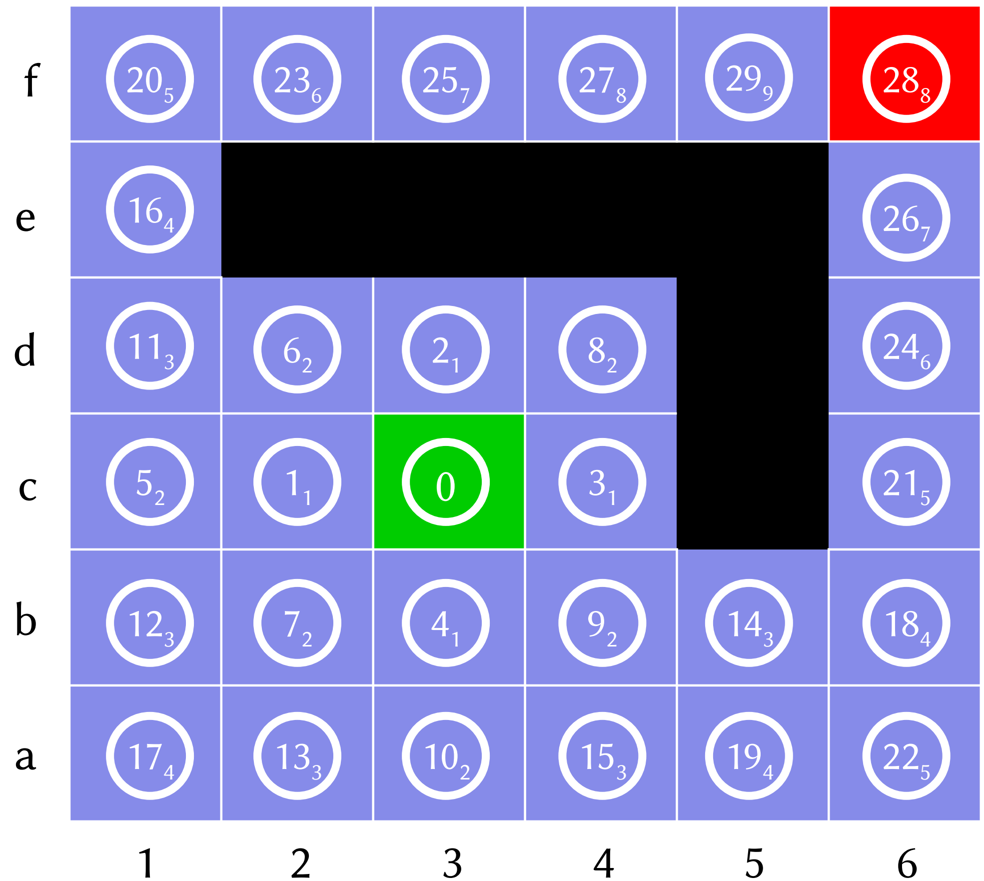


Example: UCS on a Simple Binary Tree



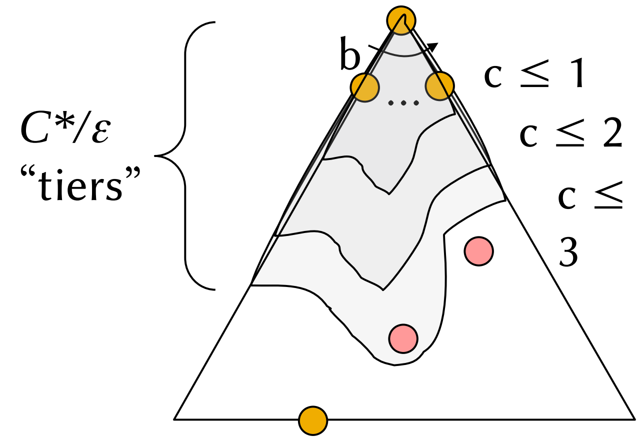
Same as BFS

Quiz: UCS on a Maze



Subscript is the path-cost by far, which decides the priority.

UCS Properties



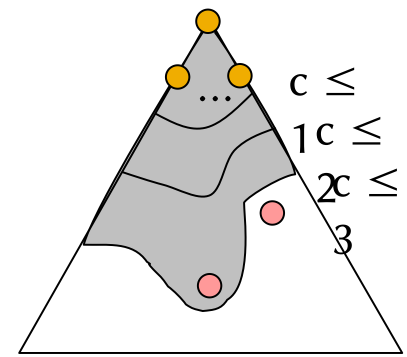
ε: the low bound of step cost

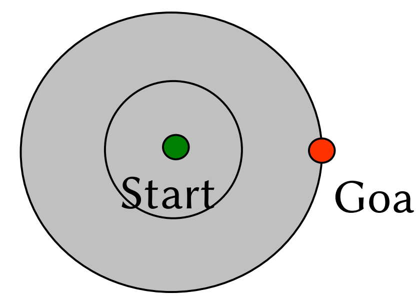
C*: the cost of the optimal solution

Complete	Yes
Time	$O(b^{C^*/\varepsilon})$
Space	$O(b^{C^*/\varepsilon})$
Optimal	Yes, nodes expanded in increasing order of g(n).

UCS Issues

- Remember: UCS explores increasing cost contours
- The good: UCS is complete and optimal!
- The bad:
  - Explores options in every "direction"
  - No information about goal location
- We'll fix that soon!

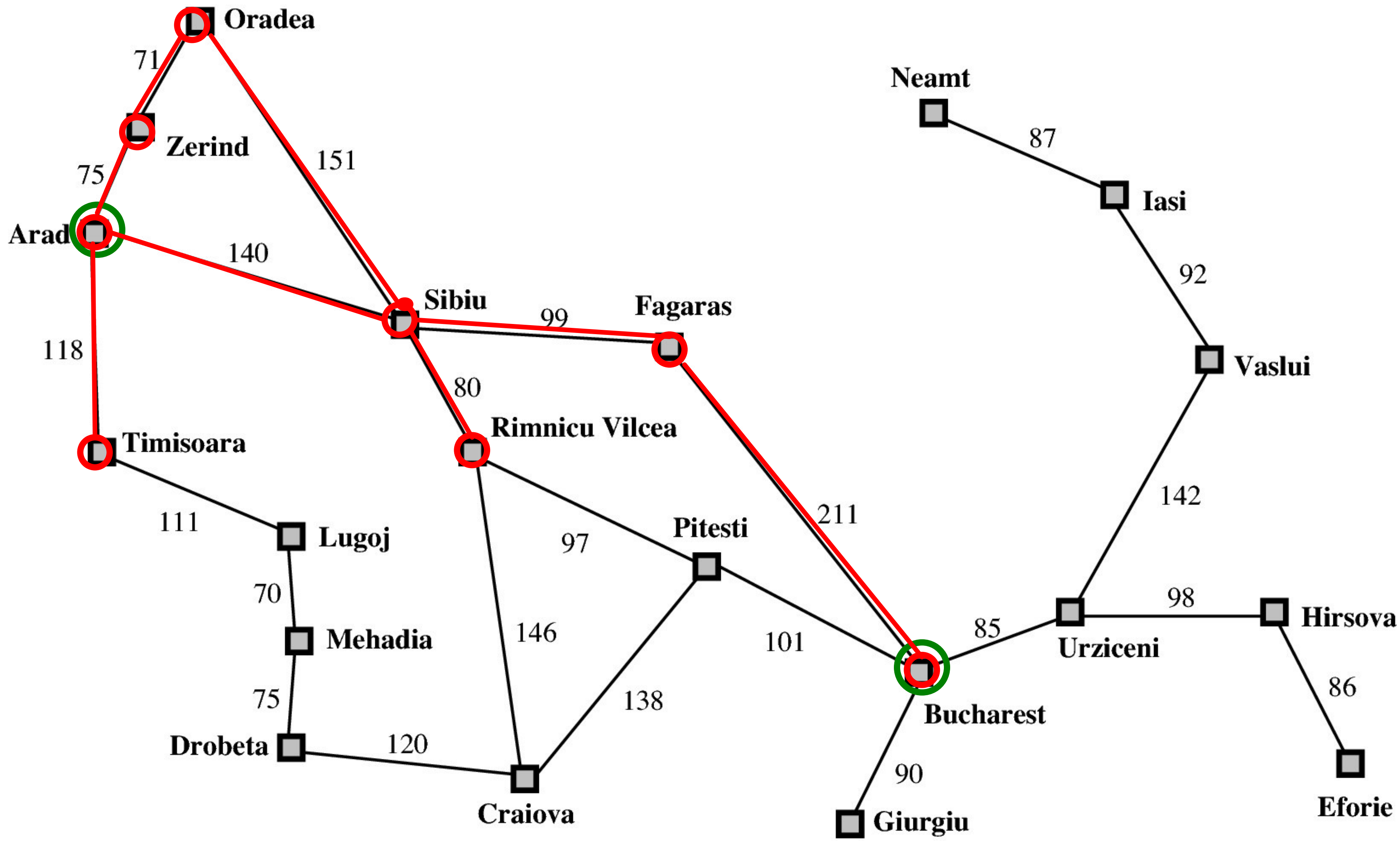




Depth-First Search (DFS)

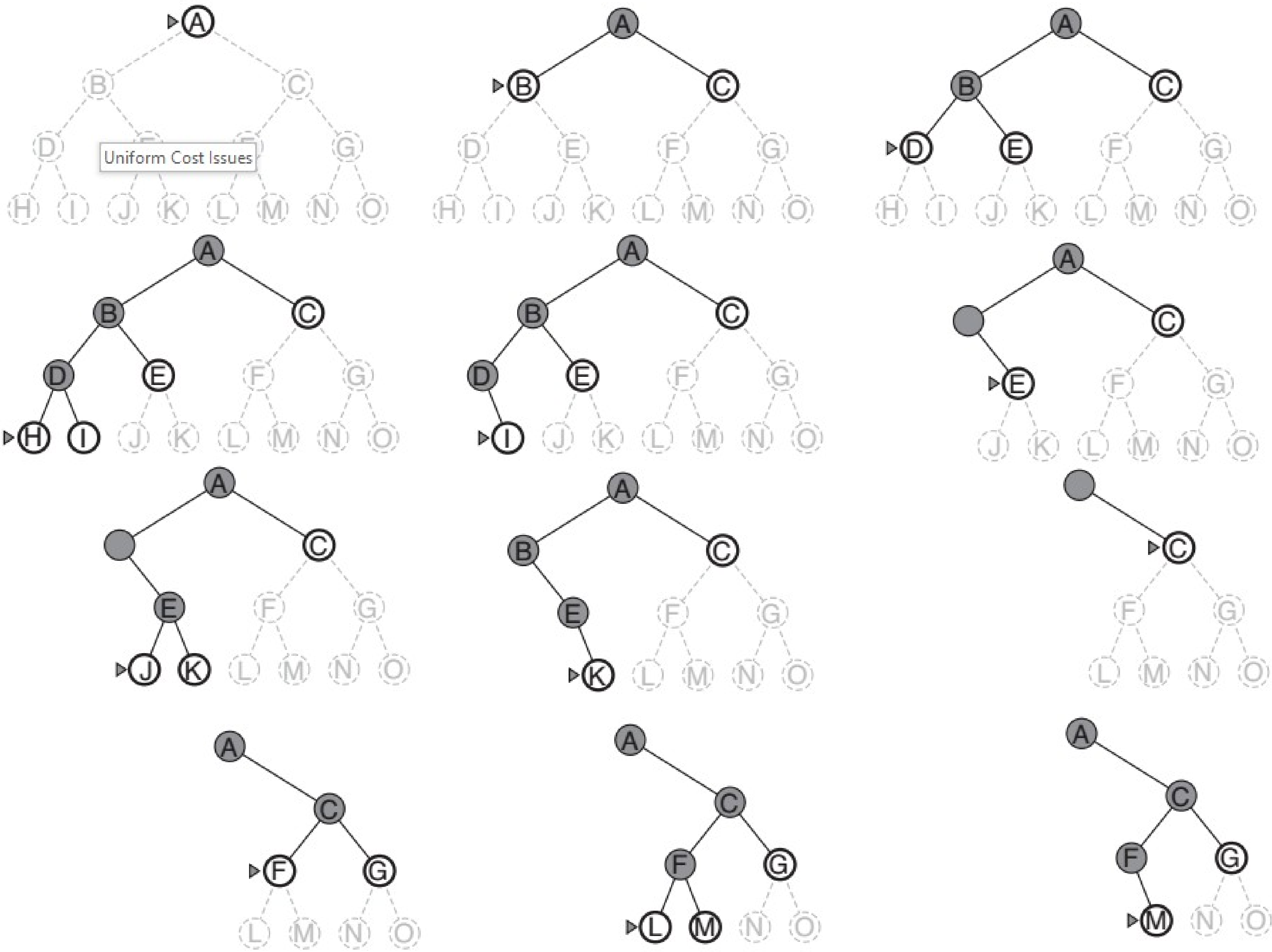
- Strategy: Expand a deepest node first
- Implementation: Frontier is a LIFO stack

Example: Traveling in Romania

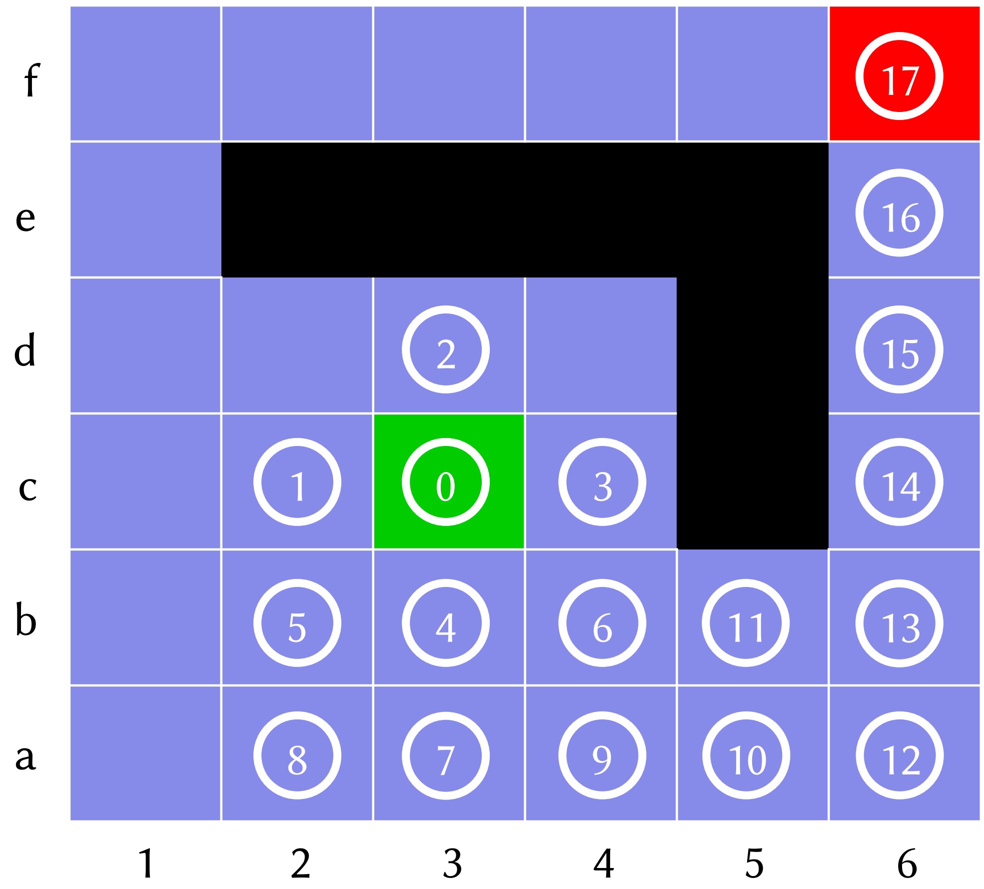


DFS on a Binary Tree

We assume that the right branch is inserted to the frontier first.



Quiz: DFS on a Maze



DFS Properties

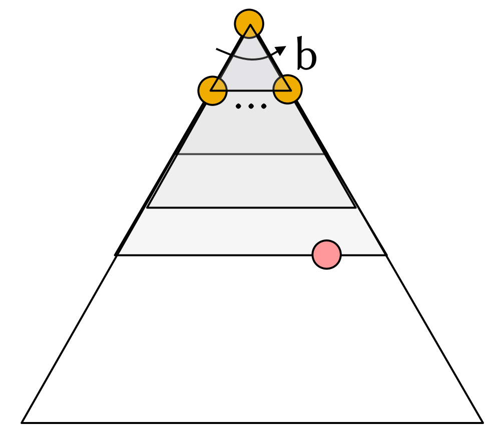
Complete	• Yes: complete in finite spaces with graph-search • No: infinite-depth spaces with tree-search
Time	• $O(b^m)$: terrible if m is much larger than d • but much faster than BFS if solutions are dense
Space	• Exponential with graph-search • $O(bm)$ with tree-search, i.e., linear space!
Optimal	No

Depth-Limited Search

- DFS with depth limit l, i.e., nodes at depth l have no successors

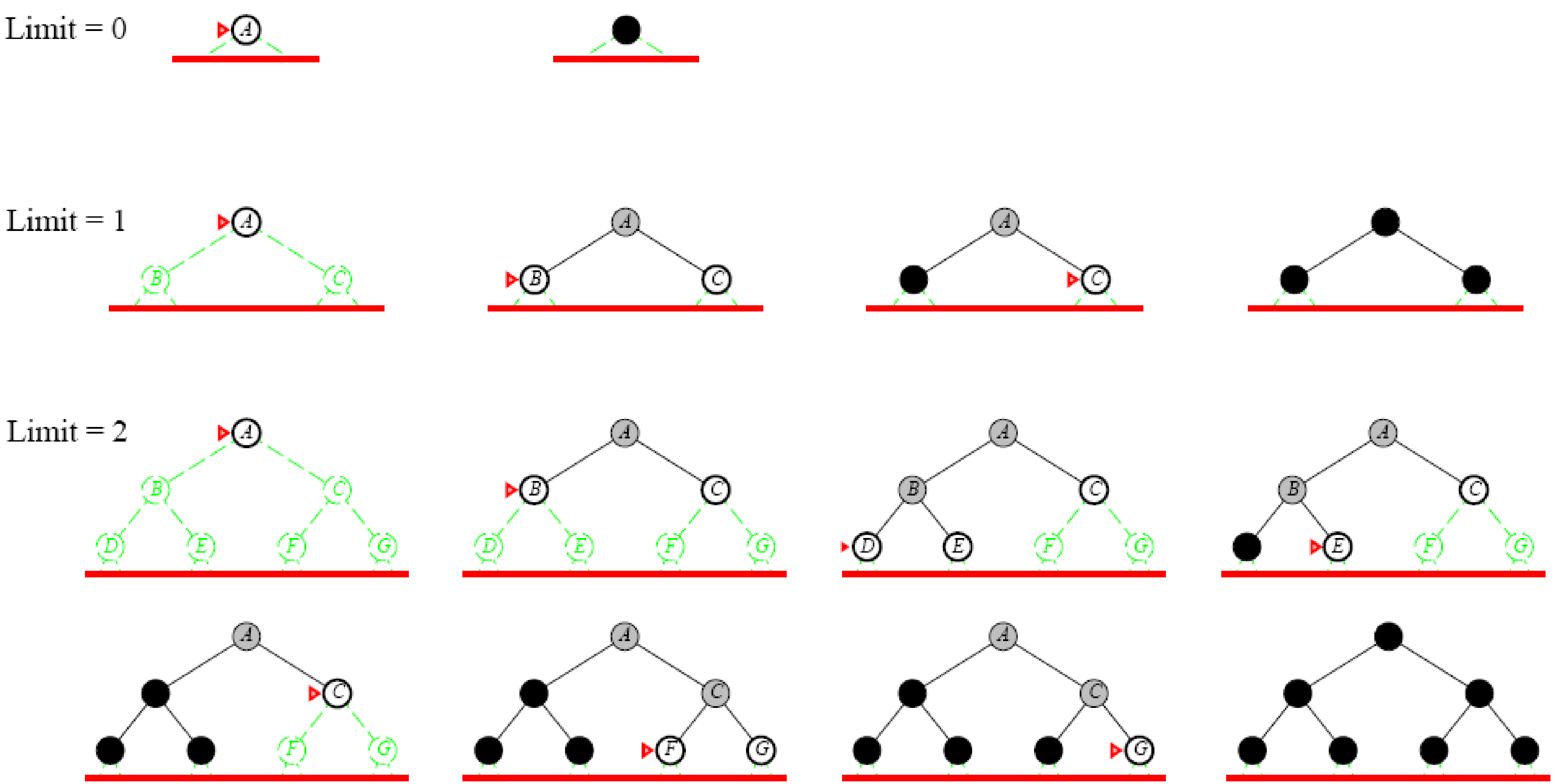
Iterative Deepening DFS

- Idea: get DFS's space advantage with BFS's time/shallow-solution advantages
  - Run a DFS with depth limit 1. If no solution…
  - Run a DFS with depth limit 2. If no solution…
  - Run a DFS with depth limit 3. …..
- Isn't that wastefully redundant?
  - Generally most work happens in the lowest level searched, so not so bad!

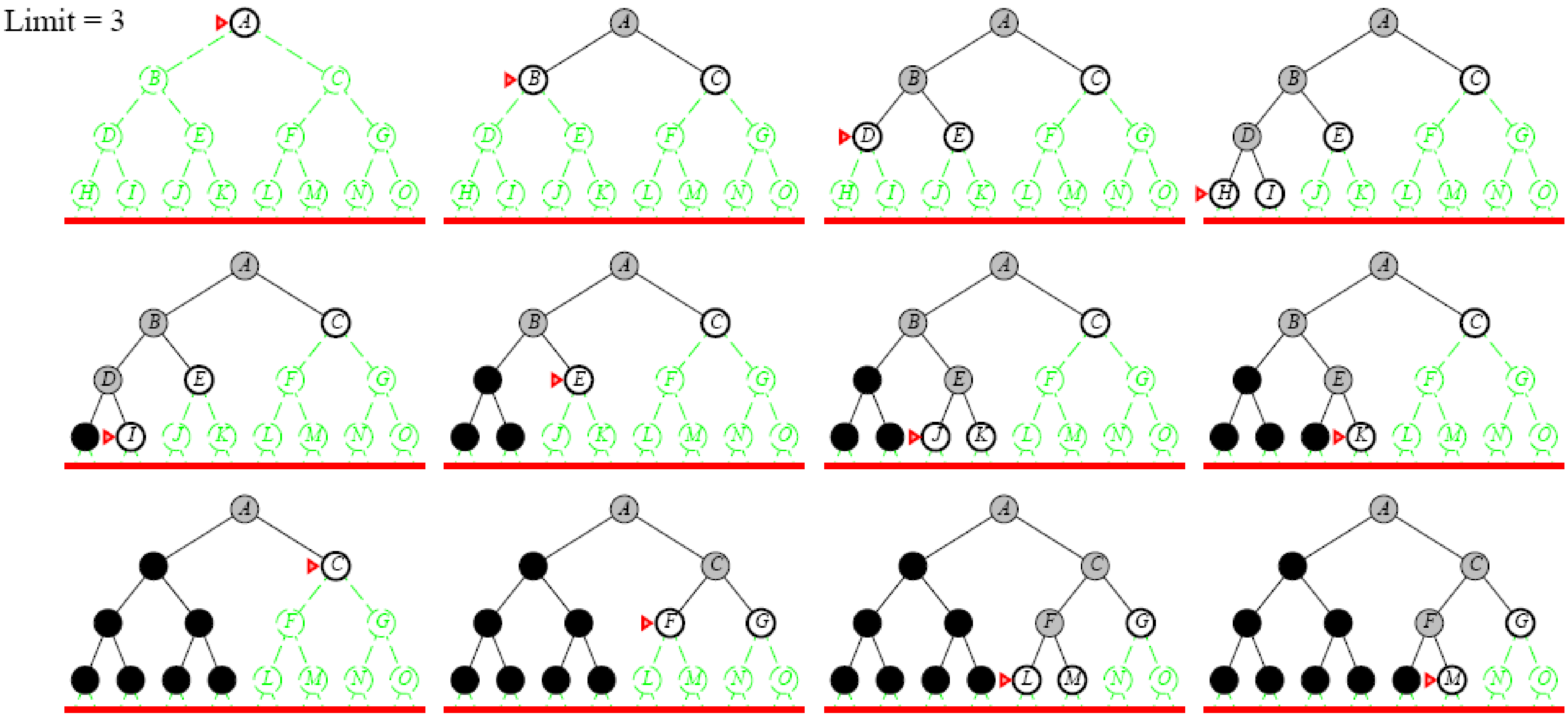


Iterative Deepening DFS on a Binary Tree

We assume that the right branch is inserted to the frontier first.



Iterative Deepening DFS on a Binary Tree



Two Categories of Search

- Uninformed search (or blind search):
  - The strategies have no idea about which successor is promisingly closer to the goal state.
  - All they can do is to generate successors and distinguish a goal state from a non-goal state.
- Informed search (or heuristic search):
  - The strategies have some idea about which successor is promisingly/heuristically closer to the goal state.

Outline

- Problem-Solving Agents
- Searching for Solution
- Uninformed Search
- Informed Search

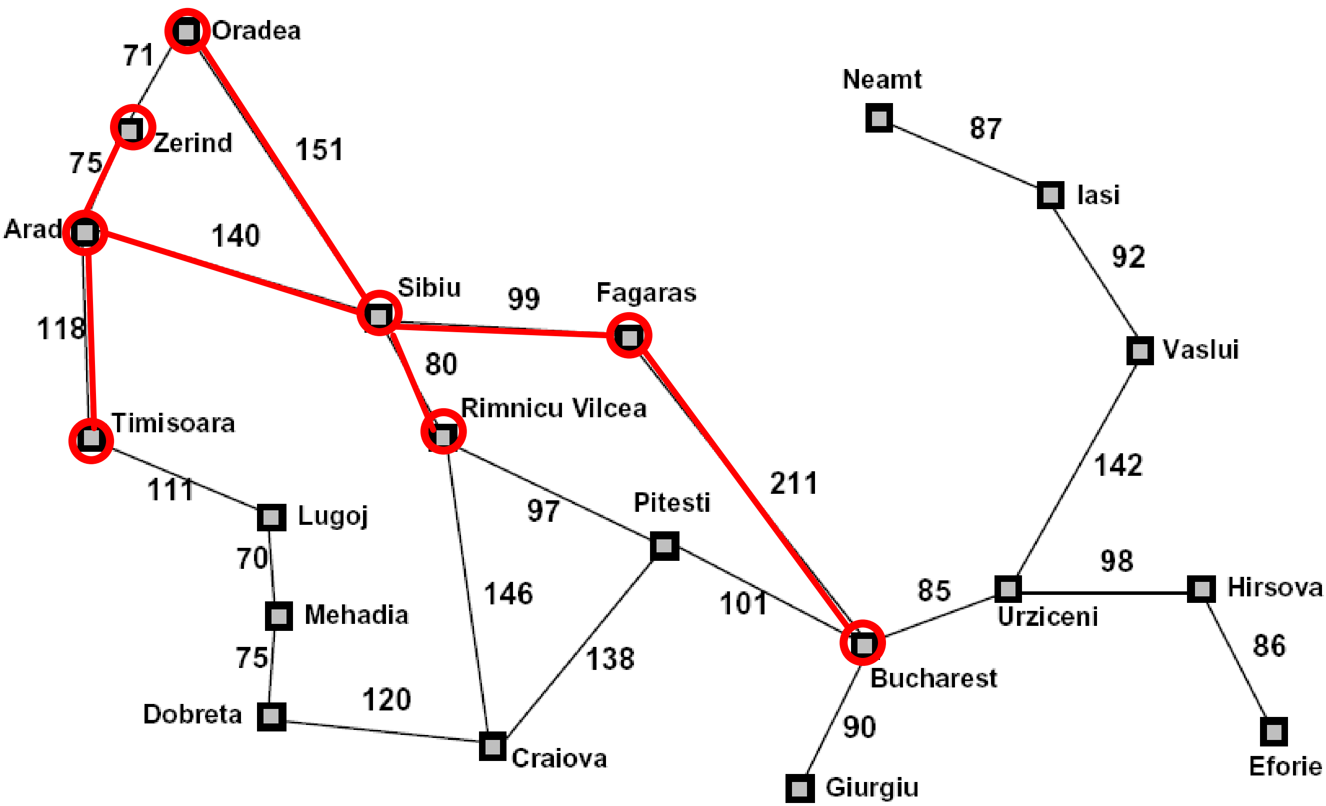
Informed Search

- The general approach we consider is called best-first search.
- The expansion of a node n is based on an evaluation function f(n), which includes a heuristic function component h(n).
  - h(n) is the estimated cost of the cheapest path from the state at node n to a goal state.

Greedy Best-First Search (GBFS)

- $f(n) = h(n)$.
- For example in route-finding problems, we use the straight-line distance as h(n)

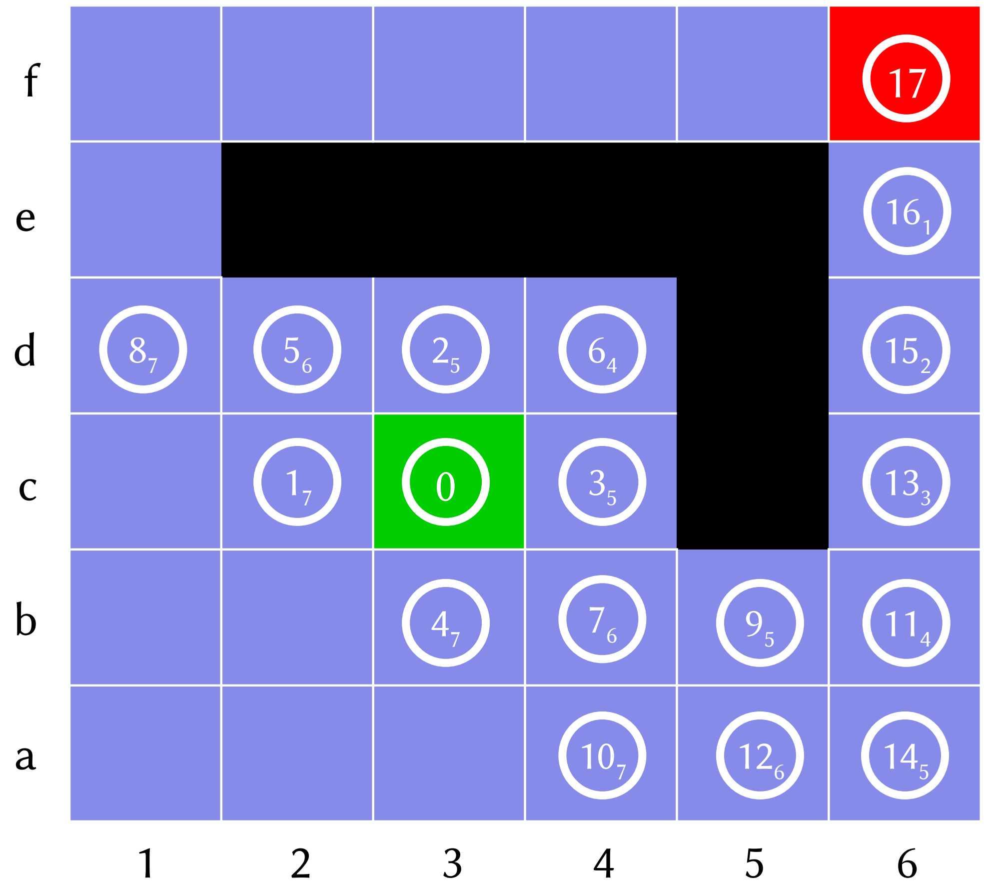
Romania with Step Costs in km



Straight−line distance
to Bucharest

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

Quiz: GBFS on a Maze



Subscript is the f=h value.

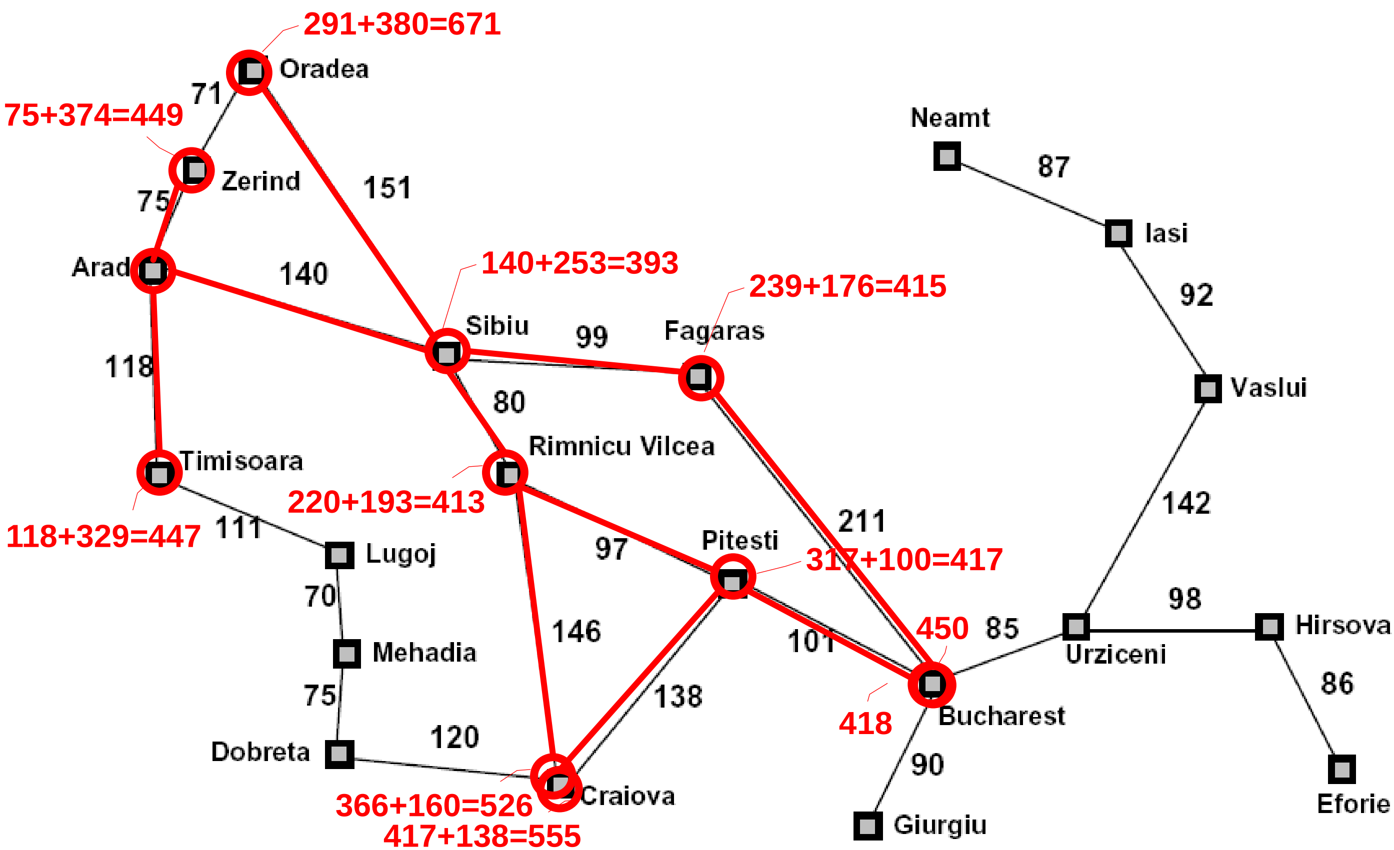
GBFS Properties

Complete	• No: can get stuck in loops with tree-search • Yes: in finite space with graph-search.
Time	$O(b^m)$, but a good heuristic can give dramatic improvement
Space	$O(b^m)$, keeps all nodes in memory
Optimal	No

A* Search

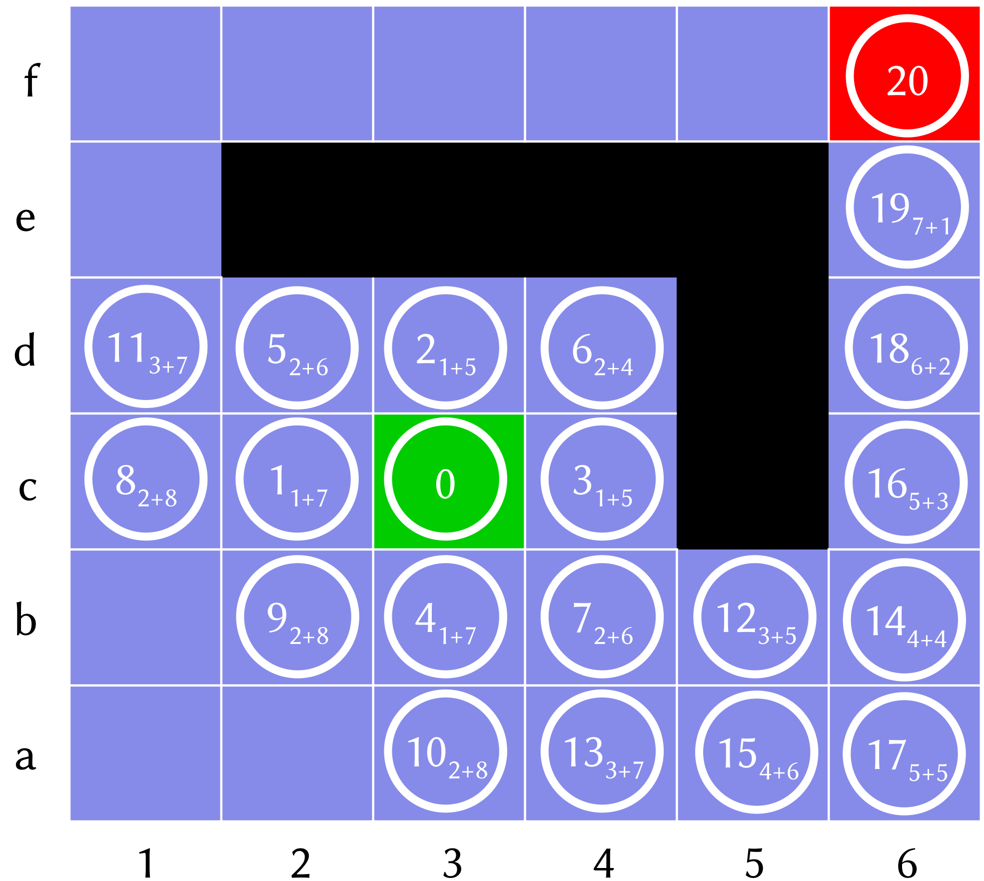
- Idea: avoid expanding paths that are already expensive
- Evaluation function: $f(n) = g(n) + h(n)$
  - $g(n)$ = cost so far to reach n
  - $h(n)$ = estimated cost to goal from n
  - $f(n)$ = estimated total cost of path through n to goal
- A* search uses an admissible heuristic $h()$
  - $h(n) \leq h^*(n)$, where $h^*(n)$ is the true cost from n.
  - Also require $h(n) \geq 0$, so $h(G) = 0$ for any goal G.
  - $h(n)$ never overestimates the actual road distance

Romania with Step Costs in km



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

Quiz: A* Search on a Maze



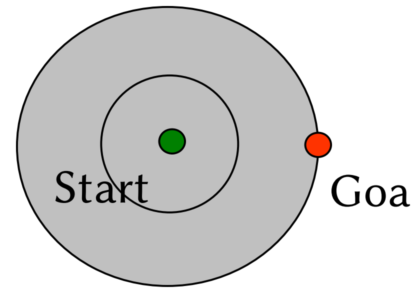
Subscript is the f=g+h value.

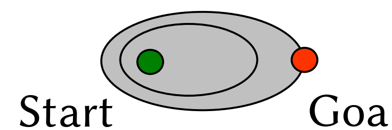
Combining UCS and Greedy Search

- Uniform-cost orders by path cost, or backward cost $g(n)$
- Greedy orders by goal proximity, or forward cost $h(n)$
- A* Search orders by the sum: $f(n) = g(n) + h(n)$

UCS vs A* Contours

- Uniform-cost expands equally in all "directions"
- A* expands mainly toward the goal, but does hedge its bets to ensure optimality





A* Properties

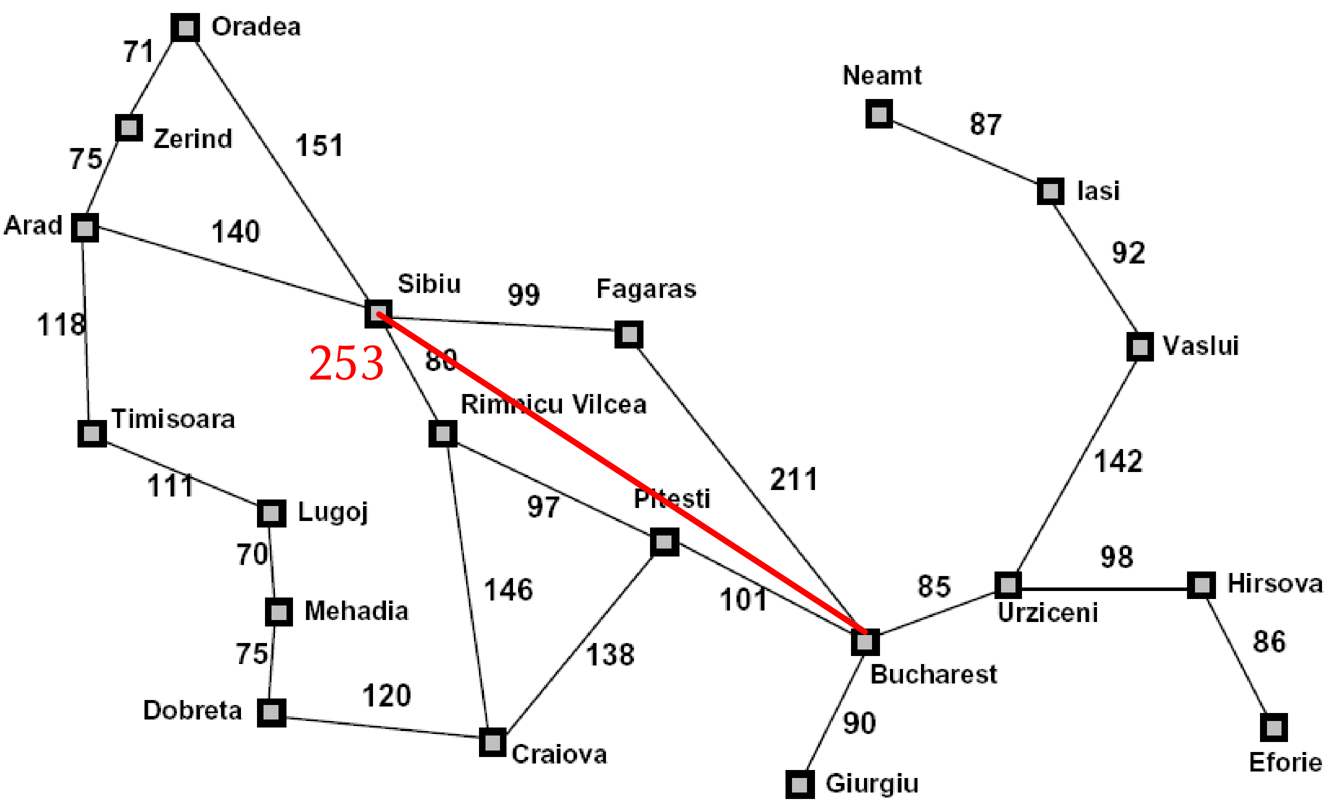
Complete	Yes, unless there are infinitely many nodes with f ≤ f(G), where G is the goal.
Time	Exponential in [relative error in h x length of solution.]
Space	Keeps all nodes in memory
Optimal	Yes, cannot expand f_{i+1} until f_i is finished

Admissible Heuristics h()

- Coming up with admissible heuristics is most of what's involved in using A* in practice.

Example: Romania

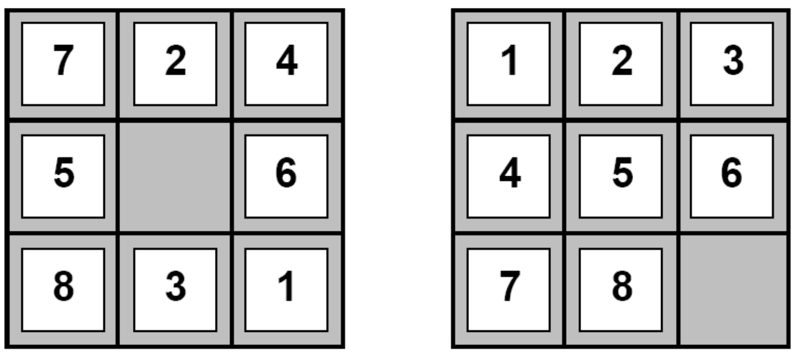
- h(n) = Euclidean distance.



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

Example: 8-Puzzle

- $h_1(n)$ = # of misplaced tiles.
- $h_2(n)$ = total Manhattan distance, i.e., # of squares from desired location of each tile.



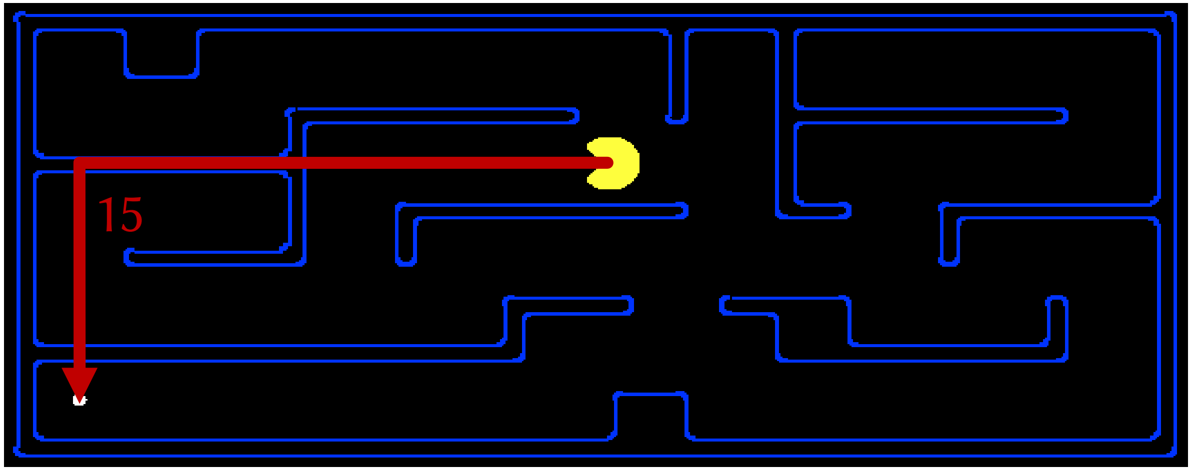
Start State   Goal State

$h_1(S) = 6$

$h_1(S) = 4+0+3+3+1+0+2+1 =$
14

Example: Pac-Man Small Maze

- h(n) = total Manhattan distance.



A* Applications

- Video games
- Pathing / routing problems
- Resource planning problems
- Robot motion planning
- Language analysis
- Machine translation
- Speech recognition
- …

Problem of Problem-Solving Agents

- It works when
  - Full observable
  - Discrete
  - Deterministic
  - Static