# CS 471: Introduction to AI

Module 3: Solving Problems by Searching

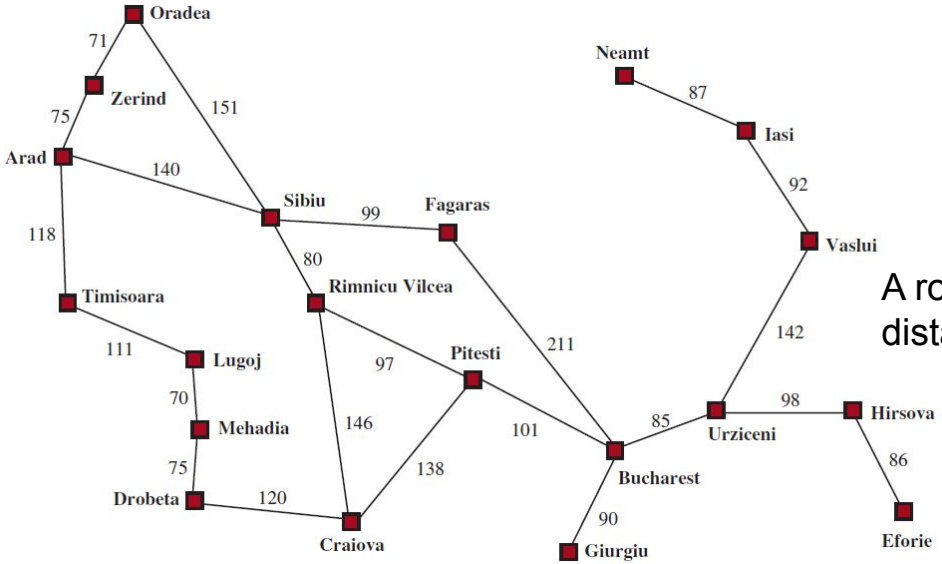
# Solving Problems By Searching

Problem-solving agent: An agent planning a sequence of actions that form a path to a goal state.

- Informed algorithms: Agent can estimate how far it is from the goal
- Uninformed algorithms: Agent has no estimate of how far it is from the goal.

# Problem-Solving Agents

- Example: Suppose a person is currently in the city of Arad (A) and has to fly to Bucharest (B). There are three roads leading out of Arad: one toward Sibiu, one to Timisoara, one to Zerind.
- None of these are the goal, unless the person is familiar with the geography of Romania, they will not know which road to follow.
- If the agent has no additional information, then the agent can do no better than to execute one of the actions at random.




A road map of part of Romania, with road distances in miles

# Search Problems and Solutions

A search problem can be defined formally as follows:

- A set of possible states that the environment can be in: state space
- The initial state that the agent starts in
- A set of one or more goal states
- The actions available to the agent
    - ACTIONS(s) returns a finite set of actions that can be executed in s.

      ACTIONS(Arad) = {ToSibiu, ToTimisoara, ToZerind}.
- A transition model: describes what each action does.

  Eg: RESULT(Arad, ToZerind) = Zerind
- An action cost function: ACTION-COST(s, a, s′) or c(s, a, s′) gives the numeric cost of applying action a in state s to reach state s′.

  Eg: for route-finding agents, the cost of an action might be the length in miles, or it might be the time it takes to complete the action.

# Formulating Problems

A good problem formulation has the right level of detail.

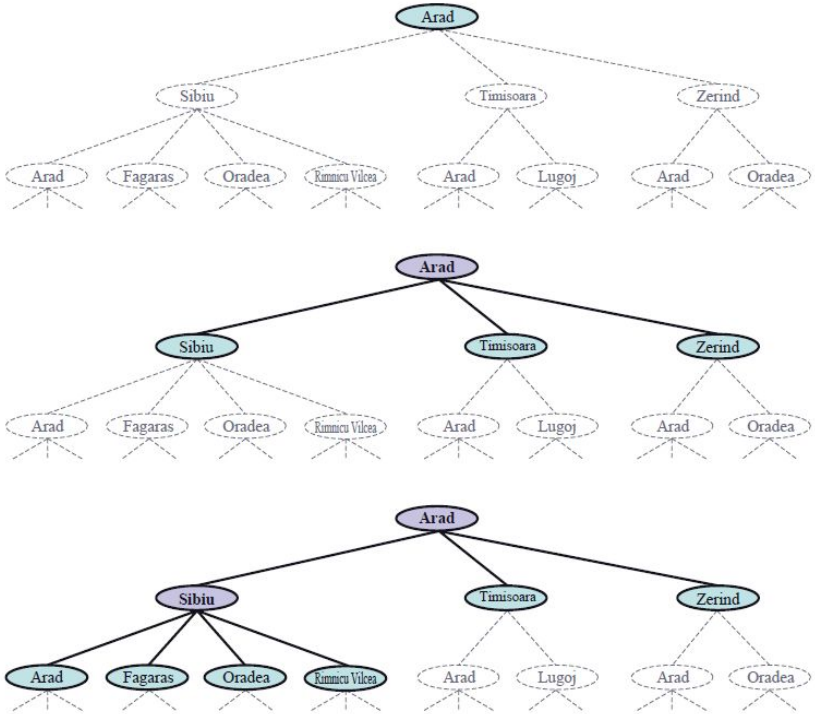
If the actions were at the level of “move the right foot forward a centimeter” or “turn the steering wheel one degree left,” the agent would never find its way to Bucharest.

# Example Problems

The vacuum world can be formulated as a grid world problem:

- States: Agent can be in either of the two cells, and each cell can either contain dirt or not.
- Initial state: Any state
- Actions: Suck, move Left, and move Right.
- Transition model: Suck removes any dirt from the agent's cell; Forward moves the agent ahead one cell in the direction it is facing, unless it hits a wall. Backward moves the agent in the opposite direction, while TurnRight and TurnLeft change the direction it is facing by 90◦.
- Goal states: States in which every cell is clean.
- Action cost: Each action costs 1.

# Search Algorithms




The root node of the search tree is at the initial state, Arad

Nodes that have been expanded are lavender with bold letters

Nodes that have been generated but not yet expanded are in green

Nodes that could be generated next are shown in faint dashed lines

# Best-first Search

How do we decide which node to expand next?

- Best-first search - chooses a node, n with minimum value of some evaluation function, f(n).
- By employing different f(n) functions, we get different specific algorithms.

# Measuring Problem-Solving Performance

- Completeness: Is the algorithm guaranteed to find a solution when there is one, and to correctly report failure when there is not?
- Cost optimality: Does it find a solution with the lowest path cost of all solutions?
- Time complexity: How long does it take to find a solution?
- Space complexity: How much memory is needed to perform the search?

# Uninformed Search Algorithms

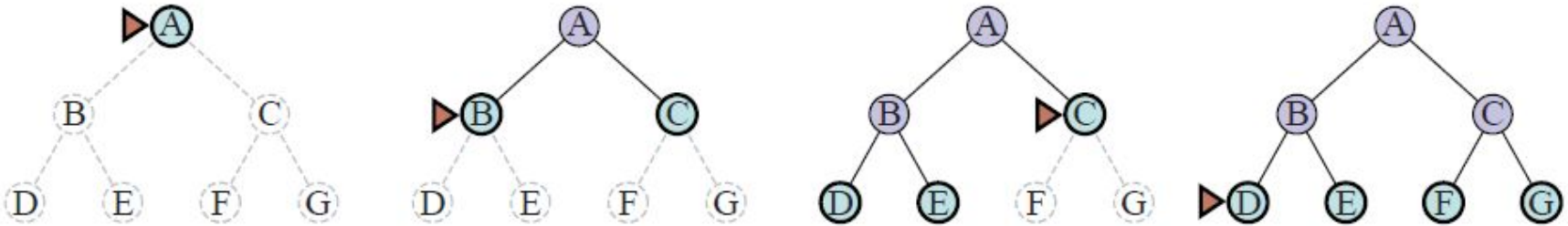
- Breadth-first Search
- Uniform-cost Search
- Depth-first search

# Uninformed Search Strategies

- Uninformed search algorithm: no clue about how close a state is to the goal(s).
- Informed agent knows the location of each city; knows that Sibiu is much closer to Bucharest and thus more likely to be on the shortest path.

# Breadth-first Search

- A systematic search strategy in which the root node is expanded first, then all the successors of the root node are expanded next, then their successors, and so on.
  - Requires lot of memory to store these nodes

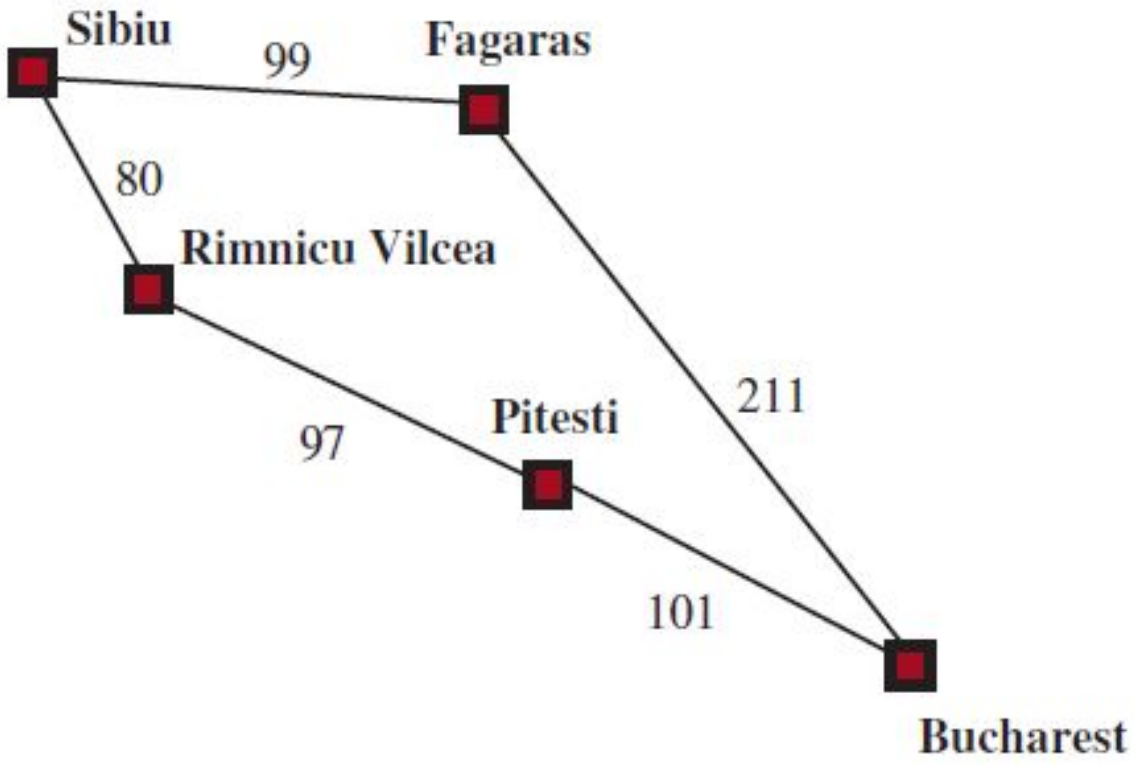



At each stage, the node to be expanded next is indicated by the triangular marker.

- It is complete and cost-optimal

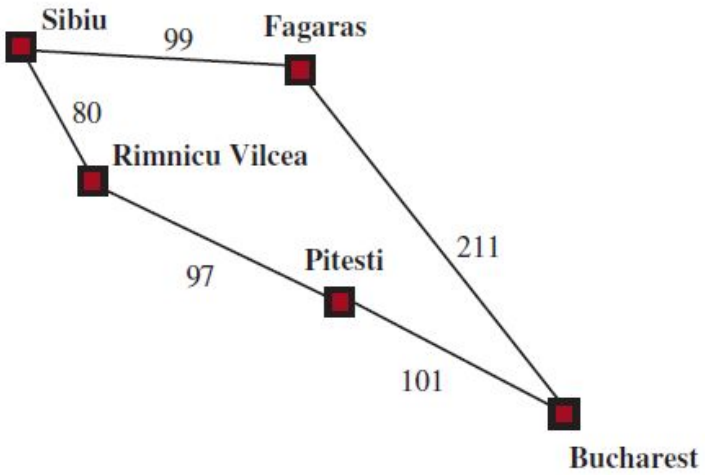
# Uniform-cost Search

Uniform-cost search spreads out in waves of uniform path-cost.



# Uniform-cost Search

- Consider the problem to get from Sibiu to Bucharest.
- The successors of Sibiu are Rimnicu Vilcea and Fagaras, with costs 80 and 99, respectively.
- The least-cost node, Rimnicu Vilcea, is expanded next, adding Pitesti with cost 80+97=177.
- The least-cost node is now Fagaras, so it is expanded, adding Bucharest with cost 99+211=310.
- Bucharest is the goal, but the algorithm tests for goals only when it expands a node, not when it generates a node, so it has not yet detected that this is a path to the goal.
- The algorithm continues on, choosing Pitesti for expansion next and adding a second path to Bucharest with cost 80+97+101=278.
- It turns out this node now has the lowest cost, so it is considered next, found to be a goal, and returned.



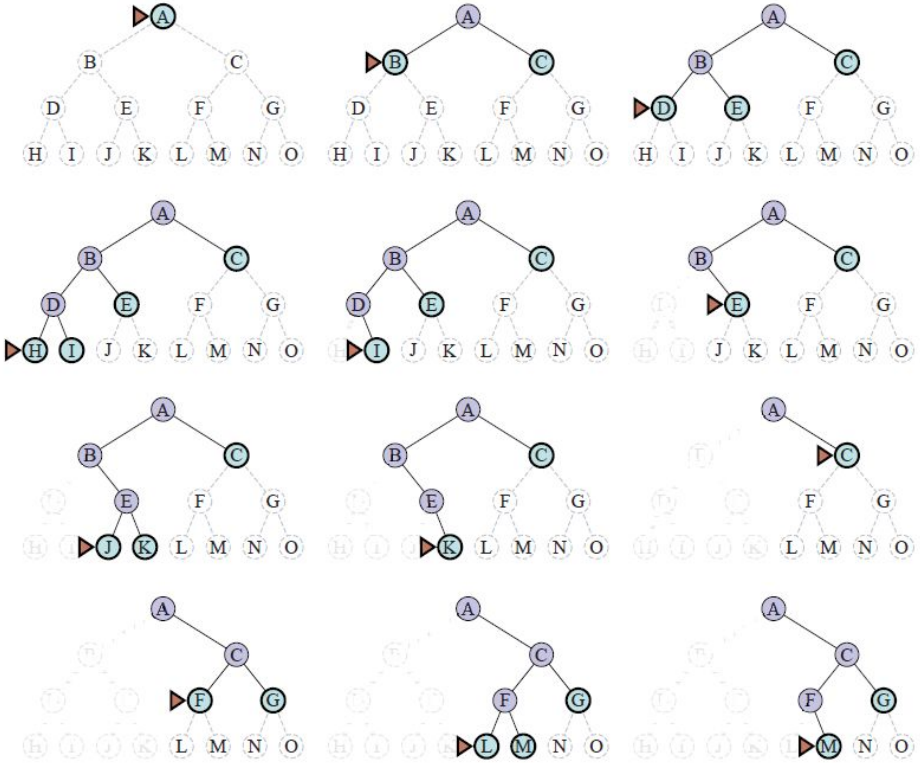
# Uniform-cost Search

- When all action costs are equal, uniform-cost search is similar to breadth-first search.
- Uniform-cost search is complete and cost-optimal, because it considers all paths systematically

# Depth-first Search

- Depth-first search proceeds to the deepest level of the search tree
- Depth-first search is not cost-optimal; it returns the first solution it finds, even if it is not cheapest

# Depth-first Search




Progress of a depth-first search on a binary tree from start state A to goal M.

# Depth-first Search

- In infinite state spaces, depth first search is not systematic: it can get stuck going down an infinite path => incomplete.

Why would anyone consider using depth-first search rather than breadth-first or best-first?

- Some problems that require exabytes of memory with breadth-first search can be handled with only kilobytes using depth-first search.
- Because of its less memory usage, depth-first tree search has been adopted in many areas of AI.

# Informed Search Algorithms

- Greedy Best-first Search
- $A^*$ Search

# Informed (Heuristic) Search Strategies

- An informed search strategy uses domain-specific hints about the location of goals.
- The hints come in the form of a heuristic function, h(n):

  h(n) = estimated cost of the cheapest path from the state at node n to a goal state.

# Greedy Best-first Search

- Greedy best-first search is a form of best-first search that expands first the node with the lowest h(n) value; the node that appears to be closest to the goal.

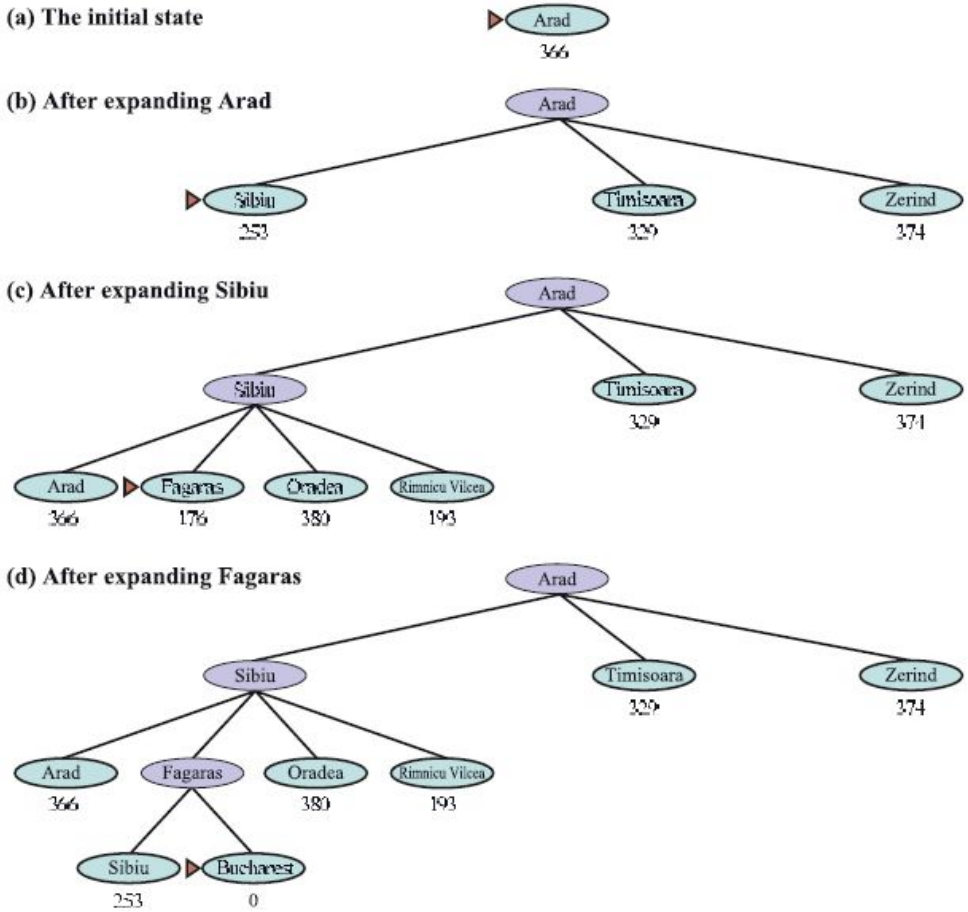
  So the evaluation function f(n) = h(n).
- For route-finding problems in Romania, we use the straight-line distance heuristic, $h_{SLD}$.

# Greedy Best-first Search

Progress of a greedy best-first search using straight line distance heuristic ($h_{SLD}$) to find a path from Arad to Bucharest.

- The first node to be expanded from Arad will be Sibiu because the heuristic says it is closer to Bucharest than is either Zerind or Timisoara.
- The next node to be expanded will be Fagaras because it is now closest according to the heuristic. Fagaras in turn generates Bucharest, which is the goal.

Not cost optimal, the path via Sibiu and Fagaras to Bucharest is 32 miles longer than the path through Rimnicu Vilcea and Pitesti.



# A* Search

- The most common informed search algorithm
- A best-first search that uses the evaluation function

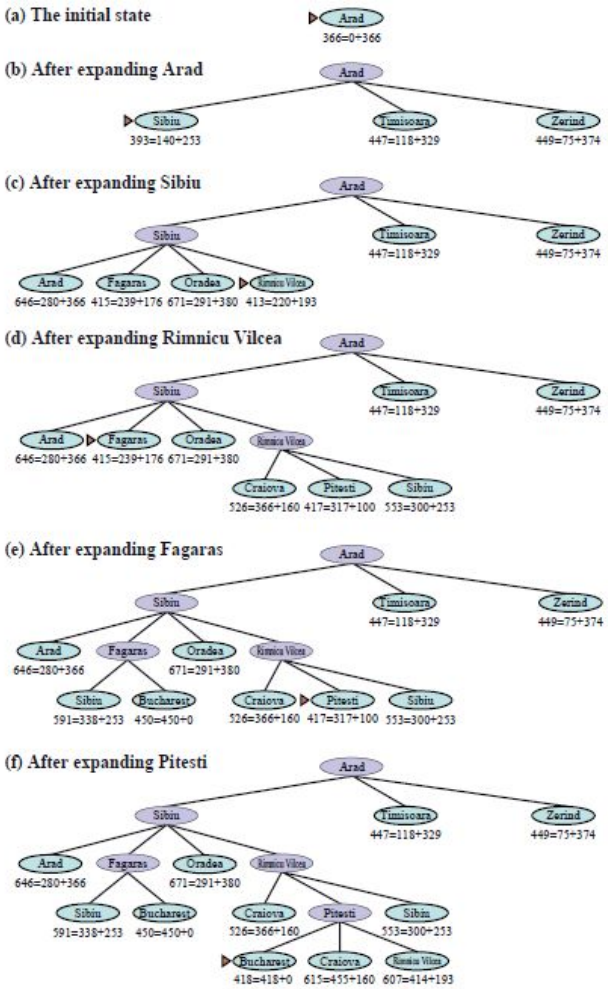
  f(n) = g(n) + h(n)

  g(n) => path cost from the initial state to node n

  h(n) => estimated cost of the shortest path from n to a goal state

  f(n) => estimated cost of the best path that continues from n to a goal.

# A* Search




Stages in an A* search for Bucharest. Nodes are labeled with f = g+h. h values are the straight-line distances to Bucharest.

# Summary

- Search algorithms are judged based on completeness, cost optimality, time complexity, space complexity.
- Best-first search: selects nodes for expansion using an evaluation function.
- Uninformed search methods have access only to the problem definition.
  - Breadth-first search: expands the shallowest nodes first; it is complete, optimal for unit action costs, but has exponential space complexity.
  - Uniform-cost search: expands the node with lowest path cost, g(n)
  - Depth-first search: expands the deepest unexpanded node first
- Informed search methods have access to a heuristic function h(n)
  - Greedy best-first search: expands nodes with minimal h(n)
  - $A^*$ search: expands nodes with minimal f(n) = g(n) + h(n).

THANK YOU!