AI1/AI&ML - Informed Search

Dr Leonardo Stella

Aims of the Session

This session aims to help you:

- Describe the difference between uninformed and informed search

- Understand the concept of a heuristic function in informed search

- Analyse the performance of A* and apply the algorithm to solve search problems

Overview

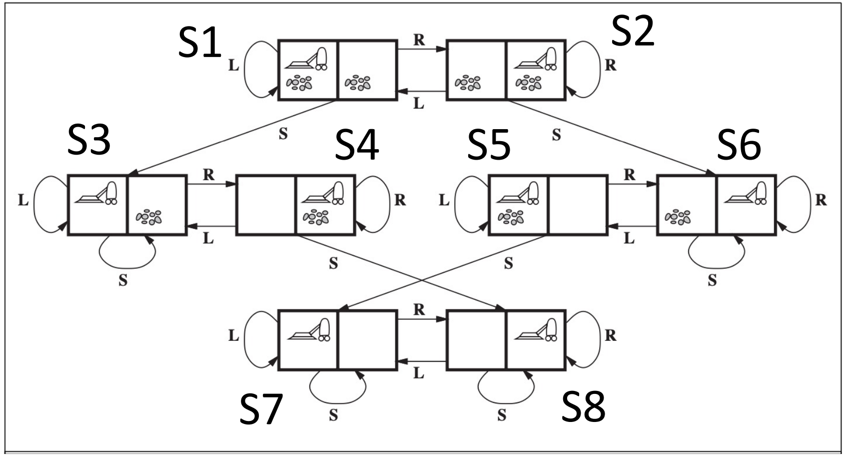
- **Recap – Uninformed Search**
- Informed Search
- A* Search

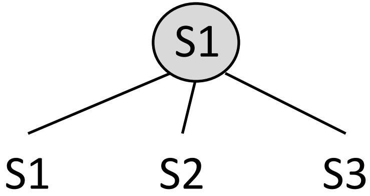
Searching for Solutions

- A solution is an action sequence from an initial state to a goal state
- Possible action sequences form a **search tree** with initial state at the root; actions are the branches and nodes correspond to the state space
- The idea is to expand the current state by applying each possible action: this generates a new set of states

Searching for Solutions

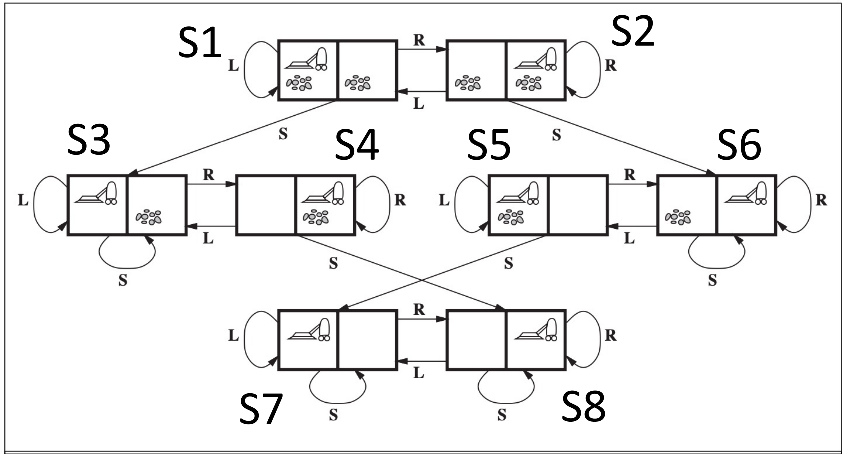
- Let us consider the example from before
- If S1 is the initial state and {S7, S8} is the set of goal states, the corresponding search tree after expanding the initial state is:

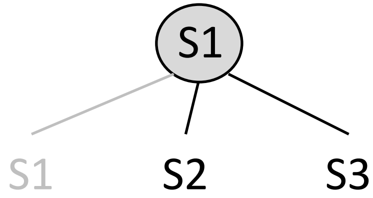




Searching for Solutions

- Each of the three nodes resulting from the first expansion is a **leaf node**
- The set of all leaf nodes available for expansion at any given time is called the **frontier** (also sometimes called the **open list**)
- The path from S1 to S1 is a **loopy path** and in general is not considered





Uninformed Search Strategies

- **Uninformed search** (also called **blind search**) means that the strategies have no additional information about states beyond that provided in the problem definition

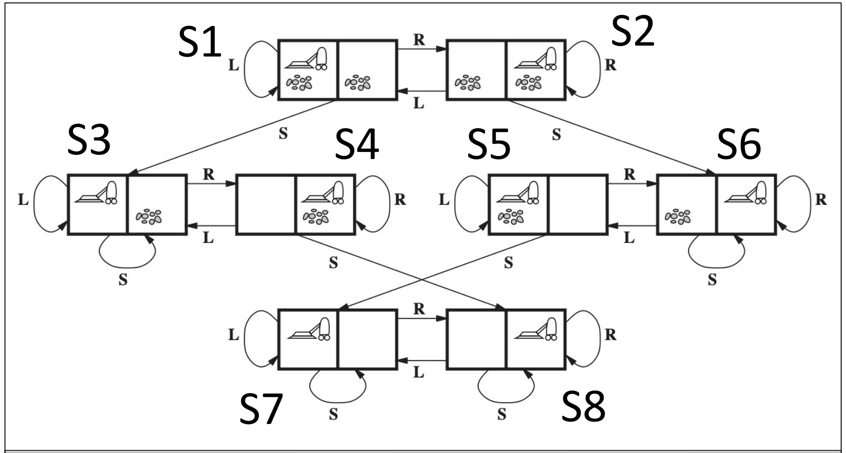
- Uninformed search strategies can only generate successors and distinguish a goal state from a non-goal state

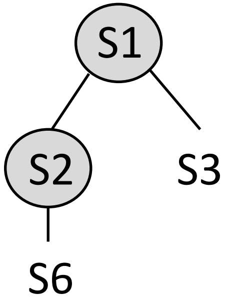
- The key difference between two uninformed search strategies is the **order** in which nodes are expanded

Breadth-First Search vs Depth-First Search

- BFS would expand the shallowest node, namely S3

- DFS would expend the deepest node, namely S6





Overview

- Recap – Uninformed Search
- **Informed Search**
- A* Search

Informed Search Strategies

- **Informed search** strategies use problem-specific knowledge beyond the definition of the problem itself

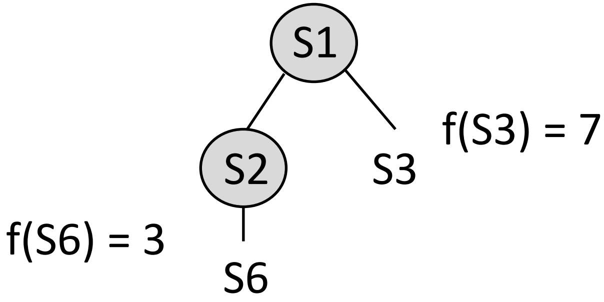
- Informed search strategies can find solutions more efficiently compared to uninformed search

Informed Search Strategies

- The general approach, called **best-first search**, is to determine which node to expand based on an **evaluation function**

$$f(n): node \rightarrow cost\ estimate$$

- This function acts as a cost estimate: the node with the lowest cost is the one that is expanded next



Informed Search Strategies - Heuristics

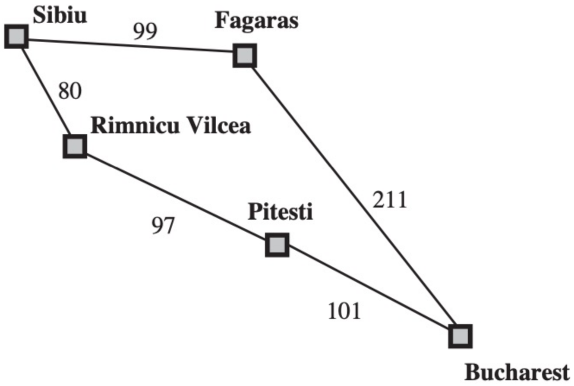
- The evaluation function $f(n)$ for most best-first algorithms includes a **heuristic function** as a component:
  $h(n)$ = *estimated cost of the cheapest path from node* n *to a goal node*

- Heuristic functions are the most common form in which new knowledge is given to the search algorithm. If n is a goal node, then $h(n) = 0$

- A heuristic can be a rule of thumb, common knowledge; it is quick to compute, but not guaranteed to work (nor to yield optimal solutions)

Informed Search Strategies - Heuristics

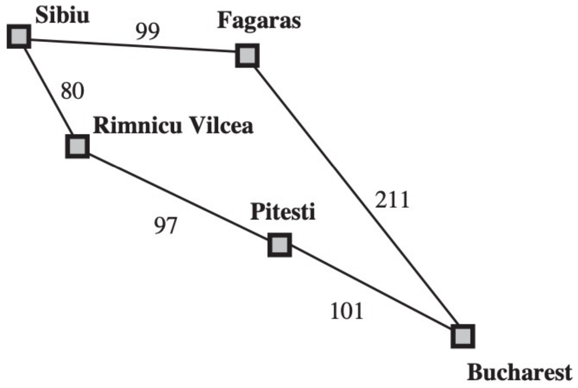
- Consider the problem to find the shortest path to Bucharest in Romania



- We can use the straight-line distance heuristic, denoted by h_{SLD}
- This is a useful heuristic as it is correlated with actual road distances

Informed Search Strategies - Heuristics

- Consider the problem to find the shortest path to Bucharest in Romania

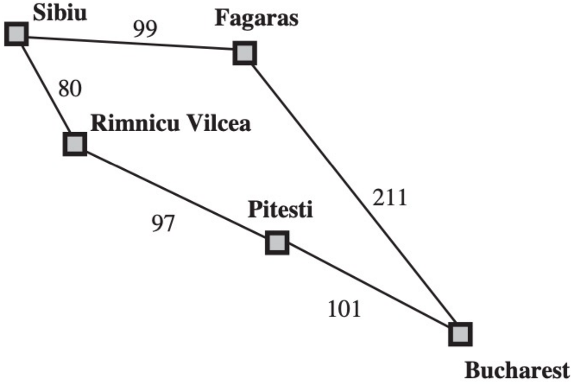


Arad	366	Mehadia	241
Bucharest	0	Neamt	234
Craiova	160	Oradea	380
Drobeta	242	Pitesti	100
Eforie	161	Rimnicu Vilcea	193
Fagaras	176	Sibiu	253
Giurgiu	77	Timisoara	329
Hirsova	151	Urziceni	80
Iasi	226	Vaslui	199
Lugoj	244	Zerind	374

- The straight-line distances h_{SLD} are shown in the table above
- For example, the SLD from Sibiu would be 253

Informed Search Strategies - Heuristics

- Consider the problem to find the shortest path to Bucharest in Romania

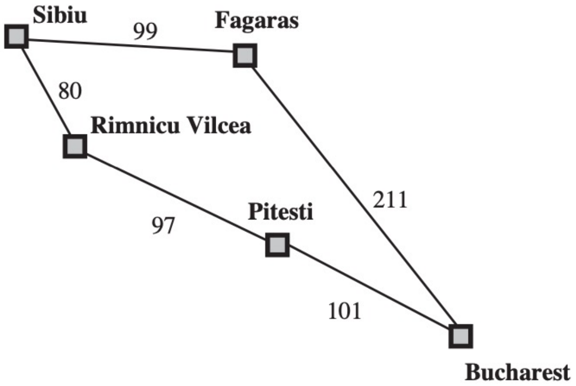


Arad	366	**Mehadia**	241
Bucharest	0	**Neamt**	234
Craiova	160	**Oradea**	380
Drobeta	242	**Pitesti**	100
Eforie	161	**Rimnicu Vilcea**	193
Fagaras	176	**Sibiu**	253
Giurgiu	77	**Timisoara**	329
Hirsova	151	**Urziceni**	80
Iasi	226	**Vaslui**	199
Lugoj	244	**Zerind**	374

- If we use $f(n) = h_{SLD}(n)$, then from Sibiu we expand Fagaras
- This is because Fagaras has SLD 176, while Rimnicu Vilcea 193

Informed Search Strategies - Heuristics

- Consider the problem to find the shortest path to Bucharest in Romania



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

- When $f(n) = h(n)$, we call this strategy **Greedy Best-First Search**

Overview

- Recap – Uninformed Search
- Informed Search
- **A* Search**

A* Search

- The most widely known informed search strategy is **A***

- This search strategy evaluates nodes using the following cost function

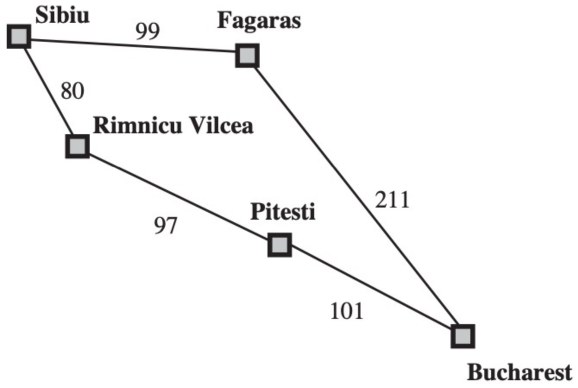
$$f(n) = g(n) + h(n)$$

where $g(n)$ is the cost to reach the node and $h(n)$ is the heuristic from the node to the goal

- This is equivalent to *the cost of the cheapest solution through node* n

A* Search - Example

- Consider the problem to find the shortest path to Bucharest in Romania



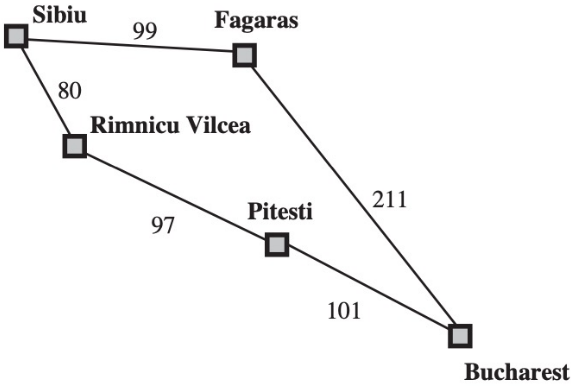
Arad	366	**Mehadia**	241
Bucharest	0	**Neamt**	234
Craiova	160	**Oradea**	380
Drobeta	242	**Pitesti**	100
Eforie	161	**Rimnicu Vilcea**	193
Fagaras	176	**Sibiu**	253
Giurgiu	77	**Timisoara**	329
Hirsova	151	**Urziceni**	80
Iasi	226	**Vaslui**	199
Lugoj	244	**Zerind**	374

- Let us consider Sibiu as the initial state. Calculate $f(n)$ to choose which node to expand, starting with Fagaras

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

A* Search - Example

- Consider the problem to find the shortest path to Bucharest in Romania



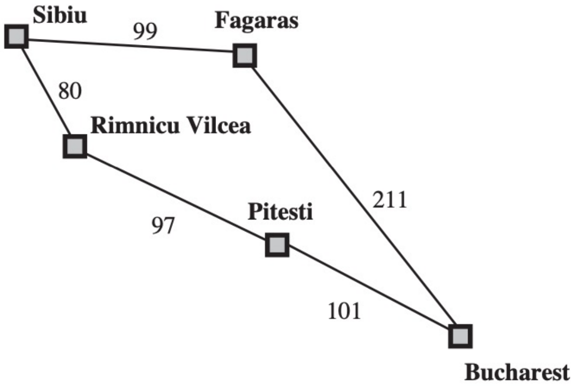
Arad	366	Mehadia	241
Bucharest	0	Neamt	234
Craiova	160	Oradea	380
Drobeta	242	Pitesti	100
Eforie	161	Rimnicu Vilcea	193
Fagaras	176	Sibiu	253
Giurgiu	77	Timisoara	329
Hirsova	151	Urziceni	80
Iasi	226	Vaslui	199
Lugoj	244	Zerind	374

- Let us consider Sibiu as the initial state. Calculate $f(n)$ to choose which node to expand, starting with Fagaras

$$f(Fagaras) = 99 + 176 = 275$$

A* Search - Example

- Consider the problem to find the shortest path to Bucharest in Romania



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

- Repeat the calculation for all the children, i.e., for Rimnicu Vilcea

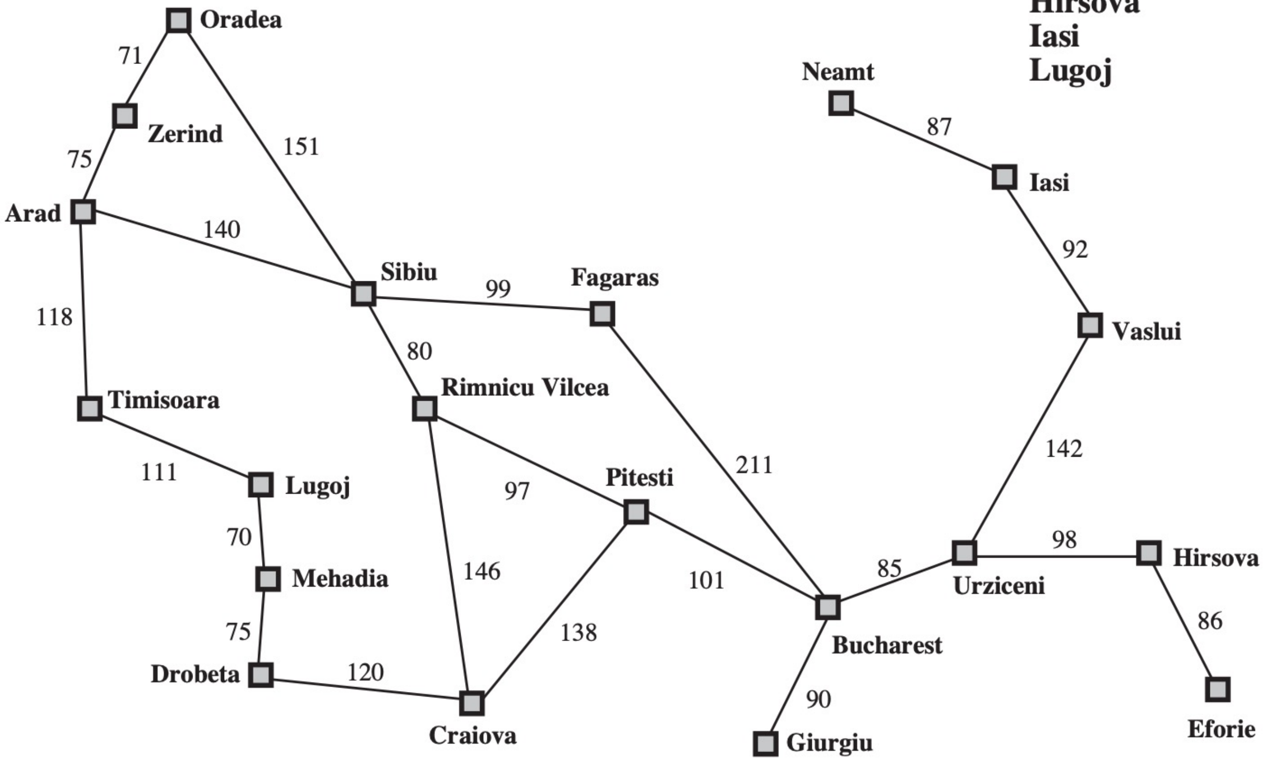
$$f(Rimnicu\ Vilcea) = 80 + 193 = 273$$

A* Search - Algorithm

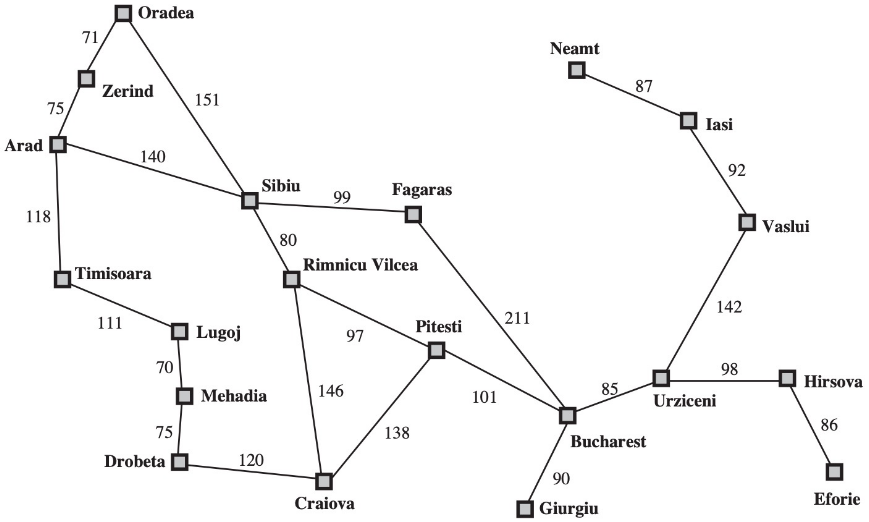
- A* search algorithm:
  - **Expand** the node in the frontier with smallest cost $f(n) = g(n) + h(n)$
  - **Do not add** children in the frontier if the node is already in the frontier or in the list of visited nodes (to avoid loopy paths)
  - If the state of a given child is in the frontier
    - If the frontier node has a larger $g(n)$, place the child into the frontier and remove the node with larger $g(n)$ from the frontier
  - **Stop** when a goal node is visited

A* Search

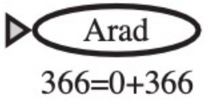
Arad	366	**Mehadia**	241
Bucharest	0	**Neamt**	234
Craiova	160	**Oradea**	380
Drobeta	242	**Pitesti**	100
Eforie	161	**Rimnicu Vilcea**	193
Fagaras	176	**Sibiu**	253
Giurgiu	77	**Timisoara**	329
Hirsova	151	**Urziceni**	80
Iasi	226	**Vaslui**	199
Lugoj	244	**Zerind**	374



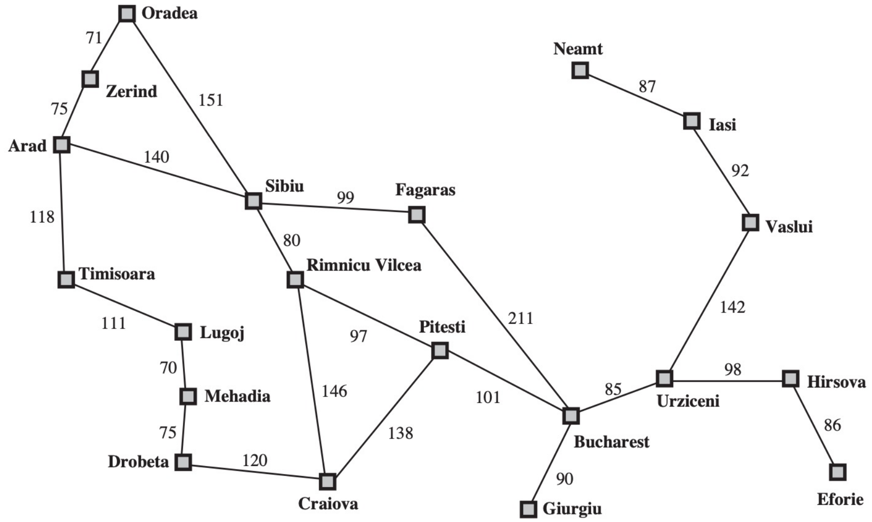
A* Search



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

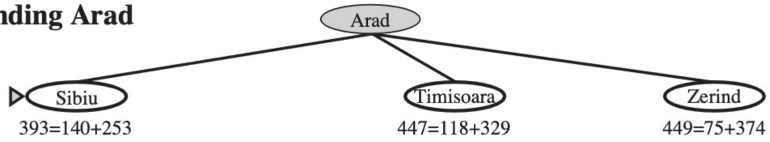


A* Search



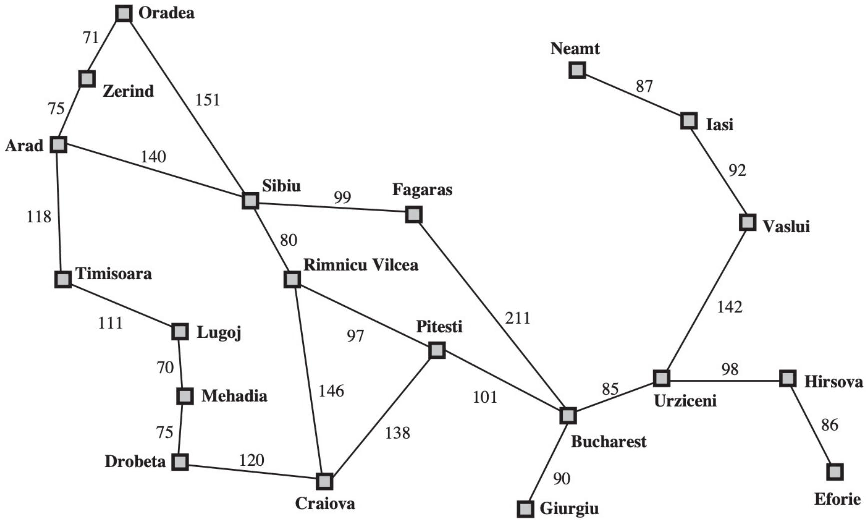
Arad	366	**Mehadia**	241
Bucharest	0	**Neamt**	234
Craiova	160	**Oradea**	380
Drobeta	242	**Pitesti**	100
Eforie	161	**Rimnicu Vilcea**	193
Fagaras	176	**Sibiu**	253
Giurgiu	77	**Timisoara**	329
Hirsova	151	**Urziceni**	80
Iasi	226	**Vaslui**	199
Lugoj	244	**Zerind**	374

After expanding Arad

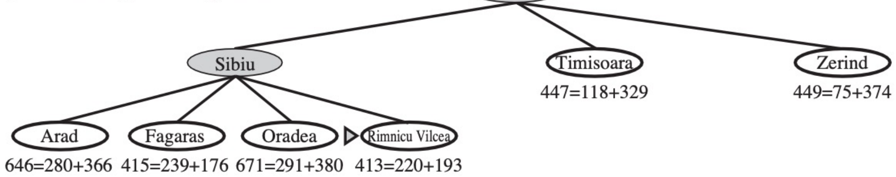


A* Search

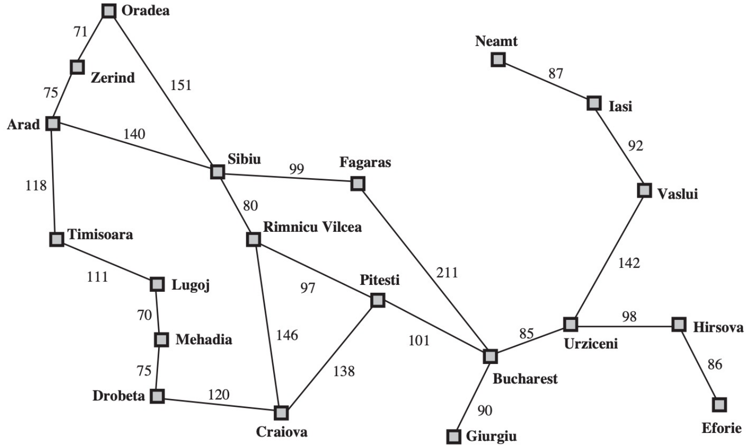
Arad	366	**Mehadia**	241
Bucharest	0	**Neamt**	234
Craiova	160	**Oradea**	380
Drobeta	242	**Pitesti**	100
Eforie	161	**Rimnicu Vilcea**	193
Fagaras	176	**Sibiu**	253
Giurgiu	77	**Timisoara**	329
Hirsova	151	**Urziceni**	80
Iasi	226	**Vaslui**	199
Lugoj	244	**Zerind**	374



(c) After expanding Sibiu

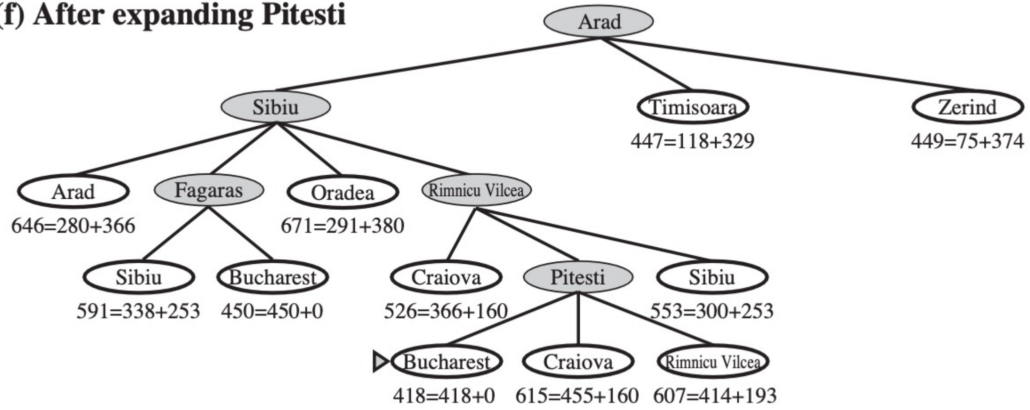


A* Search



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

(f) After expanding Pitesti



A* Search - Completeness and Optimality

- The A* search is **complete** and **optimal** if $h(n)$ is consistent

A* Search - Completeness and Optimality

- The A* search is **complete** and **optimal** if $h(n)$ is consistent

- A heuristic is said to be consistent (or monotone), if the estimate is always no greater than the estimated distance from any neighbouring vertex to the goal, plus the cost of reaching that neighbour

$$h(n) \leq cost(n, n') + h(n')$$

A* Search - Time and Space Complexity

- The number of states for the A* search is **exponential** in the length of the solution, namely for constant step costs: $O(b^{\epsilon d})$
- When h^* is the actual cost from root node to goal node, $\epsilon = \frac{(h^* - h)}{h^*}$ is the relative error

A* Search - Time and Space Complexity

- The number of states for the A* search is **exponential** in the length of the solution, namely for constant step costs: $O(b^{\epsilon d})$
- When h^* is the actual cost from root node to goal node, $\epsilon = \frac{(h^* - h)}{h^*}$ is the relative error
- Space is the main issue with A*, as it keeps all generated nodes in memory, therefore A* is not suitable for many large-scale problems

A* Search - Summary

Let us summarise the performance of the A* search algorithm

- **Completeness**: if the heuristic $h(n)$ is consistent, then the A* algorithm is complete
- **Optimality**: if the heuristic $h(n)$ is consistent, A* is optimal

A* Search - Summary

Let us summarise the performance of the A* search algorithm

- **Completeness**: if the heuristic $h(n)$ is consistent, then the A* algorithm is complete
- **Optimality**: if the heuristic $h(n)$ is consistent, A* is optimal
- **Time complexity**: $O(b^{\epsilon d})$, where ϵ is the relative error of the heuristic

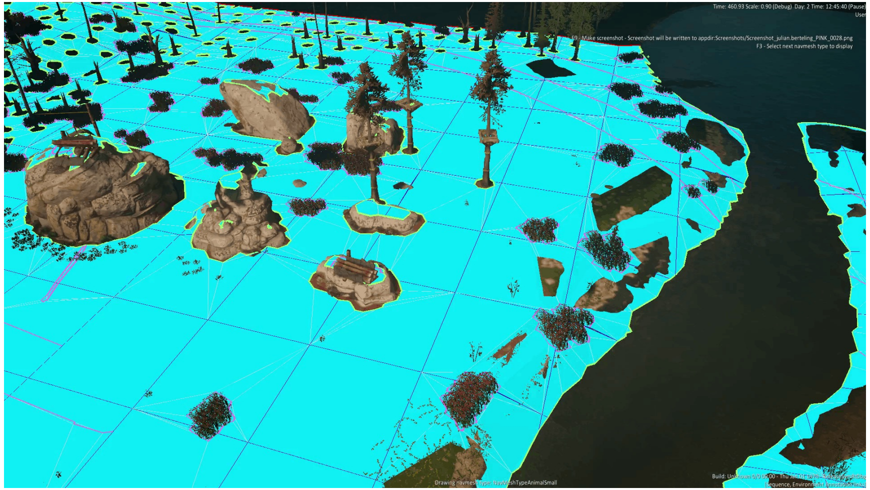
A* Search - Summary

Let us summarise the performance of the A* search algorithm

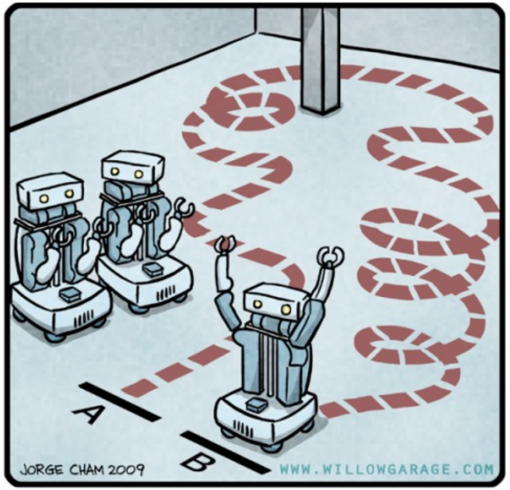
- **Completeness**: if the heuristic $h(n)$ is consistent, then the A* algorithm is complete
- **Optimality**: if the heuristic $h(n)$ is consistent, A* is optimal
- **Time complexity**: $O(b^{\epsilon d})$, where ϵ is the relative error of the heuristic
- **Space complexity**: $O(b^d)$, since we keep in memory all expanded nodes and all nodes in the frontier

A* - Applications

- A* has a large number of applications
- In practice, the most common ones are in games and in robotics



R.O.B.O.T. Comics



"HIS PATH-PLANNING MAY BE SUB-OPTIMAL, BUT IT'S GOT FLAIR."

Summary

- A* is complete and optimal, given a consistent heuristic

- However, A* has typically high time/space complexity, regardless of the heuristic chosen

- Heuristics have a considerable impact on the performance of informed search algorithms, and they can drastically reduce the time and space complexity in comparison to uninformed search algorithms

Aims of the Session

You should now be able to:

- Describe the difference between uninformed and informed search
- Understand the concept of a heuristic function in informed search
- Analyse the performance of A* and apply the algorithm to solve search problems