



UNIVERSITY OF  
BIRMINGHAM

# ARTIFICIAL INTELLIGENCE 1 INFORMED SEARCH

DR LEONARDO STELLA

SCHOOL OF COMPUTER SCIENCE

2023/2024 - Week 9

# AIMS OF THE SESSION

This session aims to help you:

- Understand the concept of a heuristic function
- Explain the steps involved in  $A^*$
- Analyse the performance of  $A^*$  and apply the algorithm to solve search problems

1 Recap: Uninformed Search

2 Informed Search

# OUTLINE

1 Recap: Uninformed Search

2 Informed Search

# RECAP: QUIZ

Q1. What does it mean for an environment to be **known**?

- ☐ There are only finitely many actions at any state
- ☐ Possible to determine which states are reached by which action
- ☐ The agent is able to know the current state
- ☐ Each action has exactly one outcome

# RECAP: QUIZ

Q1. What does it mean for an environment to be **known**?

- ☐ There are only finitely many actions at any state
- ☒ Possible to determine which states are reached by which action
- ☐ The agent is able to know the current state
- ☐ Each action has exactly one outcome

## RECAP: QUIZ

- Q2. If we relax the assumption for an environment to be discrete, BFS is still complete.
- ☐ True
  - ☐ False

## RECAP: QUIZ

- Q2. If we relax the assumption for an environment to be discrete, BFS is still complete.
- ☐ True
  - ☒ False, as  $b$  would be infinite



# RECAP: QUIZ

Q3. Which components form the **state space** of a search problem?

- ☐ Transition model
- ☐ Initial state
- ☐ Goal test
- ☐ Path cost function
- ☐ Action set

# RECAP: QUIZ

Q3. Which components form the **state space** of a search problem?

- ☒ Transition model
- ☒ Initial state
- ☐ Goal test
- ☐ Path cost function
- ☒ Action set

# RECAP: QUIZ

Q4. The time complexity of DFS is  $\mathcal{O}(b^m)$ .

- ☐ True
- ☐ False

# RECAP: QUIZ

Q4. The time complexity of DFS is  $\mathcal{O}(b^m)$ .

- ☒ True
- ☐ False

# OUTLINE

1 Recap: Uninformed Search

2 Informed Search

# INFORMED SEARCH

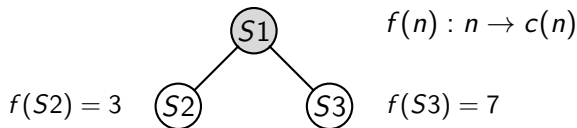
- **Informed search** strategies use problem-specific knowledge beyond the definition of the problem itself

# INFORMED SEARCH

- **Informed search** strategies use problem-specific knowledge beyond the definition of the problem itself
- In general, informed search strategies can find solutions more efficiently compared to uninformed search

# INFORMED SEARCH

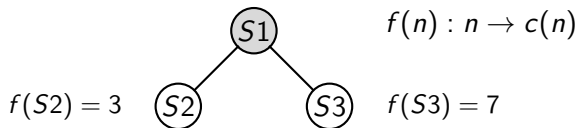
- **Informed search** strategies use problem-specific knowledge beyond the definition of the problem itself
- In general, informed search strategies can find solutions more efficiently compared to uninformed search
- The general approach, called best-first search, is to determine which node to expand based on an evaluation function  $f(n)$





# INFORMED SEARCH

- **Informed search** strategies use problem-specific knowledge beyond the definition of the problem itself
- In general, informed search strategies can find solutions more efficiently compared to uninformed search
- The general approach, called best-first search, is to determine which node to expand based on an evaluation function  $f(n)$



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

# HEURISTICS

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

# HEURISTICS

- The evaluation function  $f(n)$  for most best-first search 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

$$h(n) = 0$$

# HEURISTICS

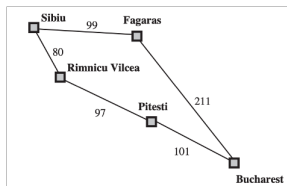
- The evaluation function  $f(n)$  for most best-first search algorithms includes a heuristic function as a component:  
 $h(n)$ : estimated cost of the cheapest path from node  $n$  to a goal node
- Heuristic functions are the most common form in which new knowledge is given to the search algorithm. 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)

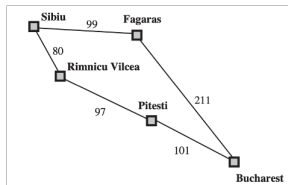
# HEURISTICS

- Consider the following problem, where we want to find the shortest path to Bucharest (from, e.g., Sibiu) in Romania



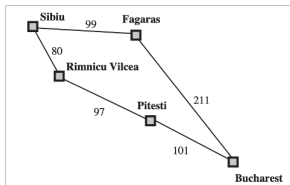
# HEURISTICS

- Consider the following problem, where we want to find the shortest path to Bucharest (from, e.g., Sibiu) in Romania
- A possible heuristic is the straight-line distance heuristic,  $h_{SLD}$
- This is a useful heuristic as it is correlated with actual road distances



# HEURISTICS

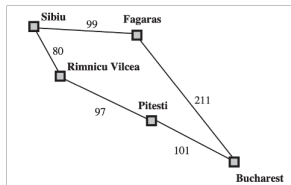
- Consider the following problem, where we want to find the shortest path to Bucharest (from, e.g., Sibiu) in Romania



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

# HEURISTICS

- Consider the following problem, where we want to find the shortest path to Bucharest (from, e.g., Sibiu) in Romania
- The straight-line distances  $h_{SLD}(n)$  are shown in the table below
- For example, the SLD from Sibiu would be  $h_{SLD}(\text{Sibiu}) = 253$

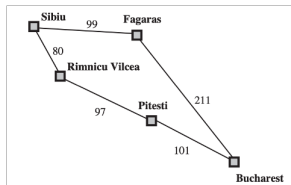


Arad	366	Mehadia	241
Bucharest	0	Neamt	234
Craiova	160	Oradea	380
Drobeta	242	Pitesti	100
Eforie	161	Rimnicu Vilcea	193
Fagaras	176	Sibiu	253
Giurgiu	77	Timisoara	329
Hirsova	151	Urziceni	80
Iasi	226	Vaslui	199
Lugoj	244	Zerind	374



# GREEDY BEST-FIRST SEARCH

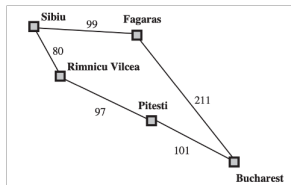
- Consider the following problem, where we want to find the shortest path to Bucharest (from, e.g., Sibiu) in Romania



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

# GREEDY BEST-FIRST SEARCH

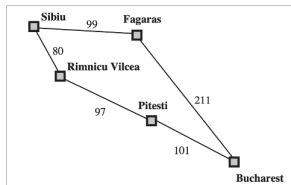
- Consider the following problem, where we want to find the shortest path to Bucharest (from, e.g., Sibiu) in Romania
- If we use  $f(n) = h_{SLD}(n)$ , then from Sibiu we expand Fagaras
- This is because  $h_{SLD}(Fagaras) = 176 < h_{SLD}(Rimnicu\ Vilcea) = 193$



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

# GREEDY BEST-FIRST SEARCH

- Consider the following problem, where we want to find the shortest path to Bucharest (from, e.g., Sibiu) in Romania
- If we use  $f(n) = h_{SLD}(n)$ , then from Sibiu we expand Fagaras
- This is because  $h_{SLD}(Fagaras) = 176 < h_{SLD}(Rimnicu\ Vilcea) = 193$
- When  $f(n) = h(n)$ , we call this strategy **greedy best-first 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<sup>\*</sup> SEARCH

- The most widely known informed search strategy is **A<sup>\*</sup>**

# A<sup>\*</sup> SEARCH

- The most widely known informed search strategy is **A<sup>\*</sup>**
- This search strategy evaluates nodes using the following evaluation function

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

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

# A<sup>\*</sup> SEARCH

- The most widely known informed search strategy is **A<sup>\*</sup>**
- This search strategy evaluates nodes using the following evaluation function

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

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

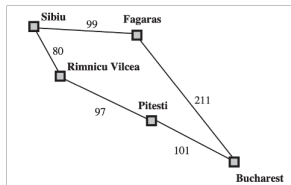
- This is equivalent to the cost of the cheapest solution through node  $n$

# A<sup>\*</sup> SEARCH

- Steps of the A<sup>\*</sup> algorithm:
  - **Expand** the shallowest node in the frontier with smallest  $f(n) = g(n) + h(n)$
  - **Repeated states and loopy paths.** If a node is in the list of visited nodes, **do not add** it to the frontier.  
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

**Activity.** Consider the following problem, where we want to find the shortest path to Bucharest from Sibiu in Romania. In pairs or small groups, calculate  $f(n)$  at each step of the A\* algorithm.



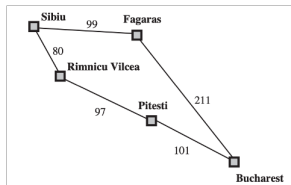
Arad	366	Mehadia	241
Bucharest	0	Neamt	234
Craiova	160	Oradea	380
Drobeta	242	Pitesti	100
Eforie	161	Rimnicu Vilcea	193
Fagaras	176	Sibiu	253
Giurgiu	77	Timisoara	329
Hirsova	151	Urziceni	80
Iasi	226	Vaslui	199
Lugoj	244	Zerind	374



# A\* SEARCH

**Activity.** Consider the following problem, where we want to find the shortest path to Bucharest from Sibiu in Romania. In pairs or small groups, calculate  $f(n)$  at each step of the A\* algorithm.

$$f(\text{Fagaras}) = g(\text{Fagaras}) + h(\text{Fagaras})$$

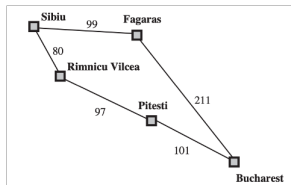


Arad	366	Mehadia	241
Bucharest	0	Neamt	234
Craiova	160	Oradea	380
Drobeta	242	Pitesti	100
Eforie	161	Rimnicu Vilcea	193
Fagaras	176	Sibiu	253
Giurgiu	77	Timisoara	329
Hirsova	151	Urziceni	80
Iasi	226	Vaslui	199
Lugoj	244	Zerind	374

# A\* SEARCH

**Activity.** Consider the following problem, where we want to find the shortest path to Bucharest from Sibiu in Romania. In pairs or small groups, calculate  $f(n)$  at each step of the A\* algorithm.

$$f(\text{Fagaras}) = 99 + 176 = 275$$



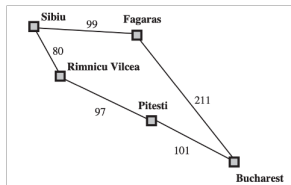
Arad	366	Mehadia	241
Bucharest	0	Neamt	234
Craiova	160	Oradea	380
Drobeta	242	Pitesti	100
Eforie	161	Rimnicu Vilcea	193
Fagaras	176	Sibiu	253
Giurgiu	77	Timisoara	329
Hirsova	151	Urziceni	80
Iasi	226	Vaslui	199
Lugoj	244	Zerind	374

# A\* SEARCH

**Activity.** Consider the following problem, where we want to find the shortest path to Bucharest from Sibiu in Romania. In pairs or small groups, calculate  $f(n)$  at each step of the A\* algorithm.

$$f(\text{Fagaras}) = 99 + 176 = 275$$

$$f(\text{Rimnicu Vilcea}) = g(\text{Rimnicu Vilcea}) + h(\text{Rimnicu Vilcea})$$



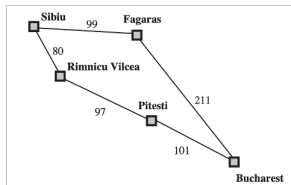
Arad	366	Mehadia	241
Bucharest	0	Neamt	234
Craiova	160	Oradea	380
Drobeta	242	Pitesti	100
Eforie	161	Rimnicu Vilcea	193
Fagaras	176	Sibiu	253
Giurgiu	77	Timisoara	329
Hirsova	151	Urziceni	80
Iasi	226	Vaslui	199
Lugoj	244	Zerind	374

# A\* SEARCH

**Activity.** Consider the following problem, where we want to find the shortest path to Bucharest from Sibiu in Romania. In pairs or small groups, calculate  $f(n)$  at each step of the A\* algorithm.

$$f(\text{Fagaras}) = 99 + 176 = 275$$

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

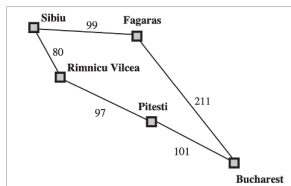


Arad	366	Mehadia	241
Bucharest	0	Neamt	234
Craiova	160	Oradea	380
Drobeta	242	Pitesti	100
Eforie	161	Rimnicu Vilcea	193
Fagaras	176	Sibiu	253
Giurgiu	77	Timisoara	329
Hirsova	151	Urziceni	80
Iasi	226	Vaslui	199
Lugoj	244	Zerind	374

A<sup>\*</sup> SEARCH

$$f(\text{Fagaras}) = 99 + 176 = 275$$

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



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

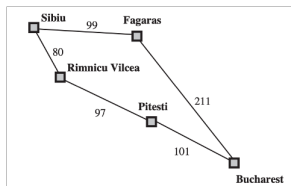
A<sup>\*</sup> SEARCH

$$f(\text{Fagaras}) = 99 + 176 = 275$$

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

$$f(\text{Bucharest}) = (99 + 211) + 0 = 310$$

$$f(\text{Pitesti}) = (80 + 97) + 100 = 277$$



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<sup>\*</sup> SEARCH

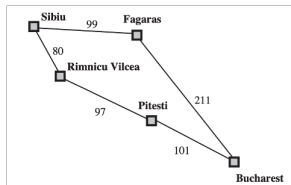
$$f(\text{Fagaras}) = 99 + 176 = 275$$

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

$$f(\text{Bucharest}) = (99 + 211) + 0 = 310$$

$$f(\text{Pitesti}) = (80 + 97) + 100 = 277$$

$$f(\text{Bucharest}) = (80 + 97 + 101) + 0 = 278$$

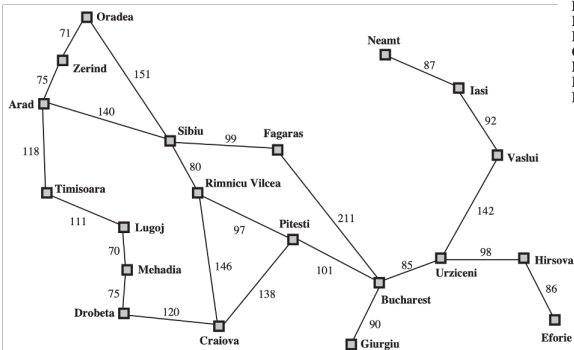


Arad	366	Mehadia	241
Bucharest	0	Neamt	234
Craiova	160	Oradea	380
Drobeta	242	Pitesti	100
Eforie	161	Rimnicu Vilcea	193
Fagaras	176	Sibiu	253
Giurgiu	77	Timisoara	329
Hirsova	151	Urziceni	80
Iasi	226	Vaslui	199
Lugoj	244	Zerind	374

# A\* SEARCH

**Activity.** Consider the following problem, where we want to find the shortest path to Bucharest from Arad in Romania. In pairs or small groups, determine the solution retrieved by A\*.

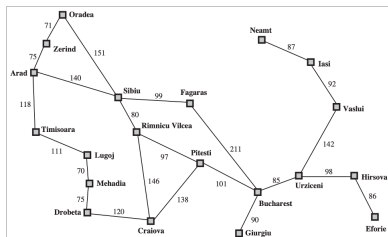
Arad	366	Mehadia	241
Bucharest	0	Neamt	234
Craiova	160	Oradea	380
Drobeta	242	Pitesti	100
Eforie	161	Rimnicu Vilcea	193
Fagaras	176	Sibiu	253
Giurgiu	77	Timisoara	329
Hirsova	151	Urziceni	80
Iasi	226	Vaslui	199
Lugoj	244	Zerind	374



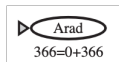


# A\* SEARCH

**Activity.** Consider the following problem, where we want to find the shortest path to Bucharest from Arad in Romania. In pairs or small groups, determine the solution retrieved by A\*.

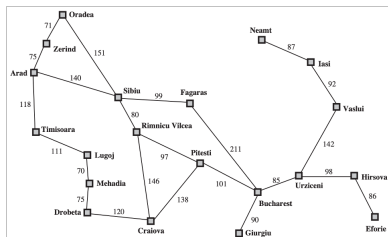


Arad	366	Mehadia	241
Bucharest	0	Neamt	234
Craiova	160	Oradea	380
Drobeta	242	Pitesti	100
Eforie	161	Rimnicu Vilcea	193
Fagaras	176	Sibiu	253
Giurgiu	77	Timisoara	329
Hirsova	151	Urziceni	80
Iasi	226	Vaslui	199
Lugoj	244	Zerind	374



# A\* SEARCH

**Activity.** Consider the following problem, where we want to find the shortest path to Bucharest from Arad in Romania. In pairs or small groups, determine the solution retrieved by A\*.



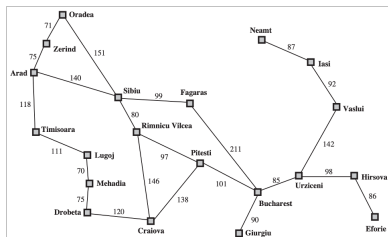
Arad	366	Mehadia	241
Bucharest	0	Neamt	234
Craiova	160	Oradea	380
Drobeta	242	Pitesti	100
Eforie	161	Rimnicu Vilcea	193
Fagaras	176	Sibiu	253
Giurgiu	77	Timisoara	329
Hirsova	151	Urziceni	80
Iasi	226	Vaslui	199
Lugoj	244	Zerind	374

After expanding Arad



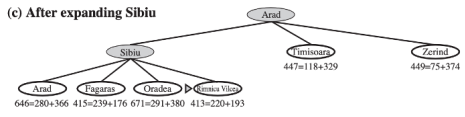
# A\* SEARCH

**Activity.** Consider the following problem, where we want to find the shortest path to Bucharest from Arad in Romania. In pairs or small groups, determine the solution retrieved by A\*.



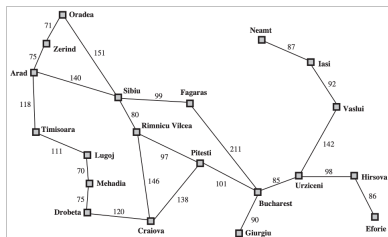
Arad	366	Mehadia	241
Bucharest	0	Neamt	234
Craiova	160	Oradea	380
Drobeta	242	Pitesti	100
Eforie	161	Rimnicu Vilcea	193
Fagaras	176	Sibiu	253
Giurgiu	77	Timisoara	329
Hirsova	151	Urziceni	80
Iasi	226	Vaslui	199
Lugoj	244	Zerind	374

(c) After expanding Sibiu



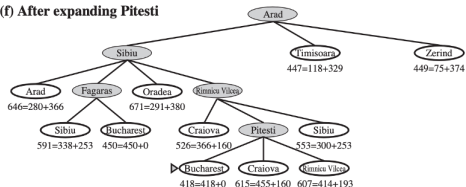
# A\* SEARCH

**Activity.** Consider the following problem, where we want to find the shortest path to Bucharest from Arad in Romania. In pairs or small groups, determine the solution retrieved by A\*.



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



# PERFORMANCE OF $A^*$

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

# PERFORMANCE OF $A^*$

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

**Definition:** A heuristic is said to be **consistent** (or **monotone**), if the estimate is always no greater than the estimated distance from any neighbouring node  $n'$  to the goal, plus the cost of reaching that neighbour:

$$h(n) \leq \text{cost}(n, n') + h(n')$$

# PERFORMANCE OF $A^*$

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

**Definition:** A heuristic is said to be **consistent** (or **monotone**), if the estimate is always no greater than the estimated distance from any neighbouring node  $n'$  to the goal, plus the cost of reaching that neighbour:

$$h(n) \leq \text{cost}(n, n') + h(n')$$

- The number of states generated by  $A^*$  is exponential in the length of the solution, namely for constant step costs:  $\mathcal{O}(b^{\epsilon d})$
- If  $h^*$  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, not suitable for large-scale problems

# PERFORMANCE OF $A^*$

- Let us summarise the performance of  $A^*$ .
  - **Completeness:** if the heuristic  $h(n)$  is consistent, then  $A^*$  is complete



# PERFORMANCE OF $A^*$

- Let us summarise the performance of  $A^*$ .
  - **Completeness:** if the heuristic  $h(n)$  is consistent, then  $A^*$  is complete
  - **Optimality:** if the heuristic  $h(n)$  is consistent, then  $A^*$  is optimal

# PERFORMANCE OF $A^*$

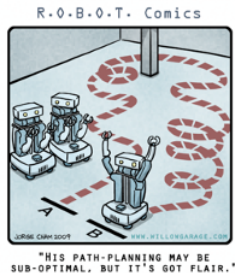
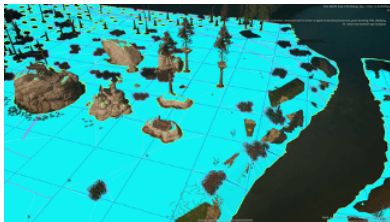
- Let us summarise the performance of  $A^*$ .
  - **Completeness:** if the heuristic  $h(n)$  is consistent, then  $A^*$  is complete
  - **Optimality:** if the heuristic  $h(n)$  is consistent, then  $A^*$  is optimal
  - **Time complexity:**  $\mathcal{O}(b^{\epsilon d})$ , where  $\epsilon$  is the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

# PERFORMANCE OF $A^*$

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

# APPLICATIONS OF $A^*$

- $A^*$  is one of the most widely adopted search algorithms
- The most common settings are in games and in robotics



# References



Russell, A. S., and Norvig, P., *Artificial Intelligence A Modern Approach*, 4<sup>th</sup> Edition. Prentice Hall.



Chapter 3 – “Solving Problems by Searching”, Section 3.5 “Informed (Heuristic) Search Strategies” up to and including Section 3.6.2 “Generating Admissible Heuristics From Relaxed Problems”.

# AIMS OF THE SESSION

You should now be able to:

- Understand the concept of a heuristic function
- Explain the steps involved in  $A^*$
- Analyse the performance of  $A^*$  and apply the algorithm to solve search problems



Thank you!