Heuristic Search (Part 1)

- Reading note: Chapter 4 covers heuristic search.

Heuristic functions

We can encode each notion of the " merit" of a state into a heuristic function, h(n).

A heuristic function maps a state onto an estimate of the cost to the goal from that state.

Can you think of examples of heuristics?

Heuristics are sensitive to the problem domain.

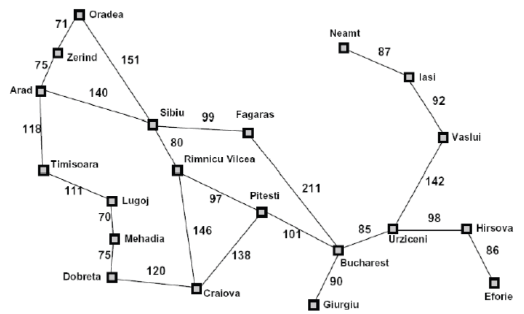
Heuristic for planning a path through a maze?

For solving a Rubick's cube?

Heuristic Search

- If $h(n_1) < h(n_2)$ this means that we guess that it is cheaper to get to the goal from n_1 than from n_2.
- We require that
  - $h(n) = 0$ for every node n whose terminal state satisfies the goal.
  - Zero cost of achieving the goal from node that already satisfies the goal.

Euclidean distance as h(s)

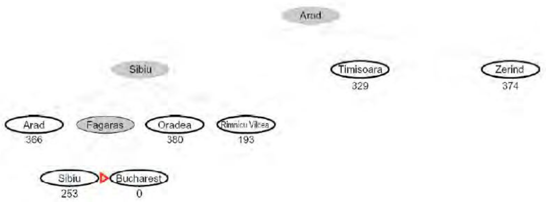


Straight−line distance to Bucharest

Arad	366
Bucharest	0
Craiova	160
Dobreta	242
Eforie	161
Fagaras	178
Giurgiu	77
Hirsova	151
Iasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitesti	98
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374

Say we want to plan a path from Arad to Bucharest, and we know the straight line distance from each city to our goal. This lets us plan our trip by picking cities at each time point that minimize the distance to our goal (or maximize our heuristic).

Best first (greedy) search

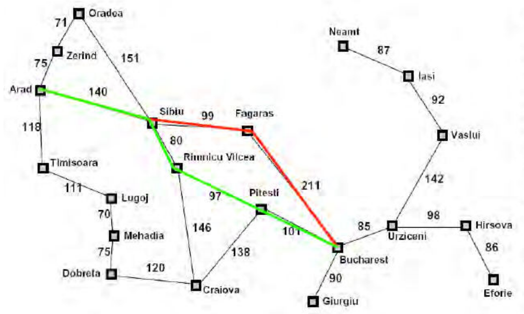


Straight−line distance to Bucharest	
Arad	366
Bucharest	0
Craiova	160
Dobreta	242
Eforie	161
Fagaras	178
Giurgiu	77
Hirsova	151
Iasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitesti	98
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374

If we only use h(n) to guide our search, the search strategy is called Greedy or Best First Search

How can it possibly go wrong??

Best first (greedy) search



Straight−line distance to Bucharest	
Arad	366
Bucharest	0
Craiova	160
Dobreta	242
Eforie	161
Fagaras	178
Giurgiu	77
Hirsova	151
Iasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitesti	98
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374

In red is the path we selected. In green is the shortest path between Arad and Bucharest. What happened?

Modifying the search

How to avoid the mistake?

Modifying the search

How to avoid the mistake?

Take into account the cost of getting to the node as well as our estimate of the cost of getting to the goal from the node.

Define an evaluation function f(n):

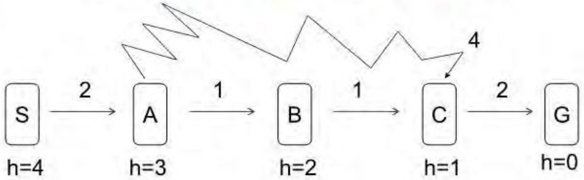
$$f(n) = g(n) + h(n)$$

g(n) is the cost of the path to node n
h(n) is the heuristic estimate of the cost of achieving the goal from n.

Always expand the node with lowest f-value on Frontier.

The f-value, f(n) is an estimate of the cost of getting to the goal via the node (path) n.
I.e., we first follow the path n then we try to get to the goal. f(n) estimates the total cost of such a solution.

A* Looking Non-Stupid

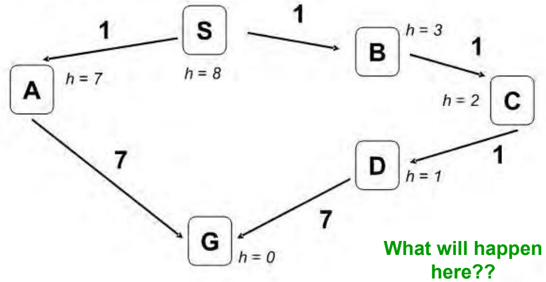


What will A* do here?

When should A* terminate?

Idea: As soon as it generates a goal state?

Look at this example:

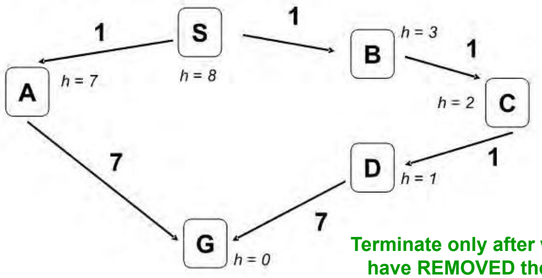


What will happen here??

When should A* terminate?

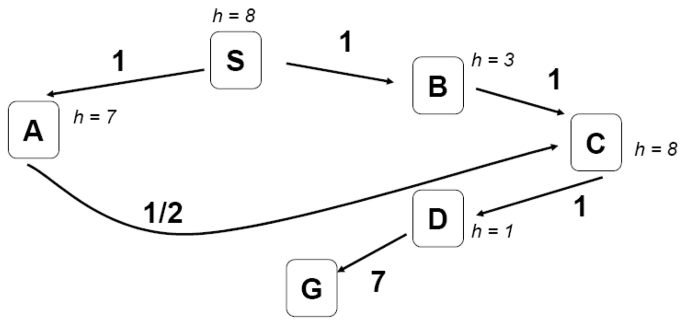
Idea: As soon as it generates a goal state?

Look at this example:

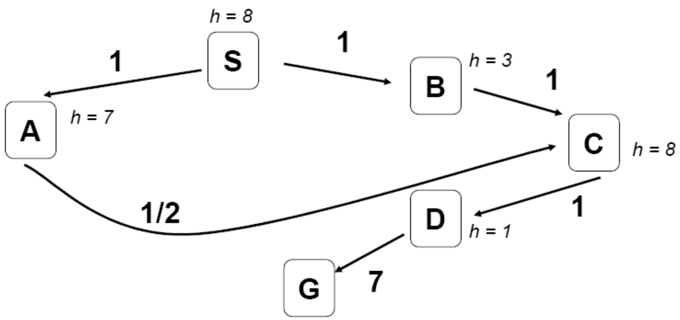


Terminate only after we have REMOVED the goal from the Frontier.

What happens when we visit nodes twice?

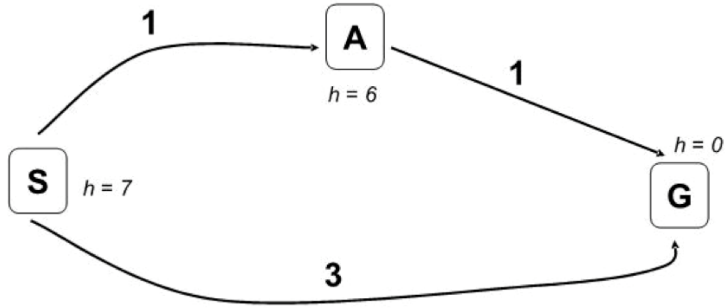


What happens when we visit nodes twice?



If A* discovers a lower cost path through a state as it is searching, it should update the order of that state on the Frontier, based on the lower path cost.

Is A* Guaranteed Optimal?

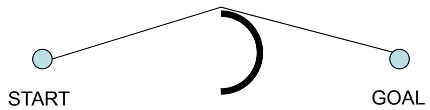


Weighted A*

- Weighted A* defines an evaluation function f(n):

$f(n) = g(n) + \varepsilon * h(n)$

  - **ε > 1** introduces a bias towards states that are closer to the goal.
  - **ε == 1** generates a provably optimal solution (assuming admissible heuristic).



Weighted A*

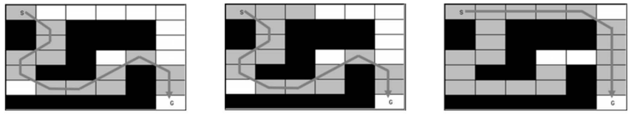
- Weighted A* defines an evaluation function f(n):

$f(n) = g(n) + \varepsilon * h(n)$

  - **ε > 1** introduces a bias towards states that are closer to the goal.
  - **ε == 1** generates a provably optimal solution (assuming admissible heuristic).
- Search is typically orders of magnitude faster
- Path that is discovered may be sub-optimal (by factor that depends on ε)

Anytime A*

- Weighted A* can be used to construct an anytime algorithm:
  - Find the best path for a given ε
  - Reduce ε and re-plan



ε = 2 13 node expansions Solution length: 12	ε = 1.5 15 node expansions Solution length: 12	ε = 1 20 node expansions Solution length: 10