Path-finding

CPSC 383: Explorations in Artificial Intelligence and Machine Learning
Fall 2025

Jonathan Hudson, Ph.D
Associate Professor (Teaching)
Department of Computer Science
University of Calgary

August 27, 2025

UNIVERSITY OF CALGARY

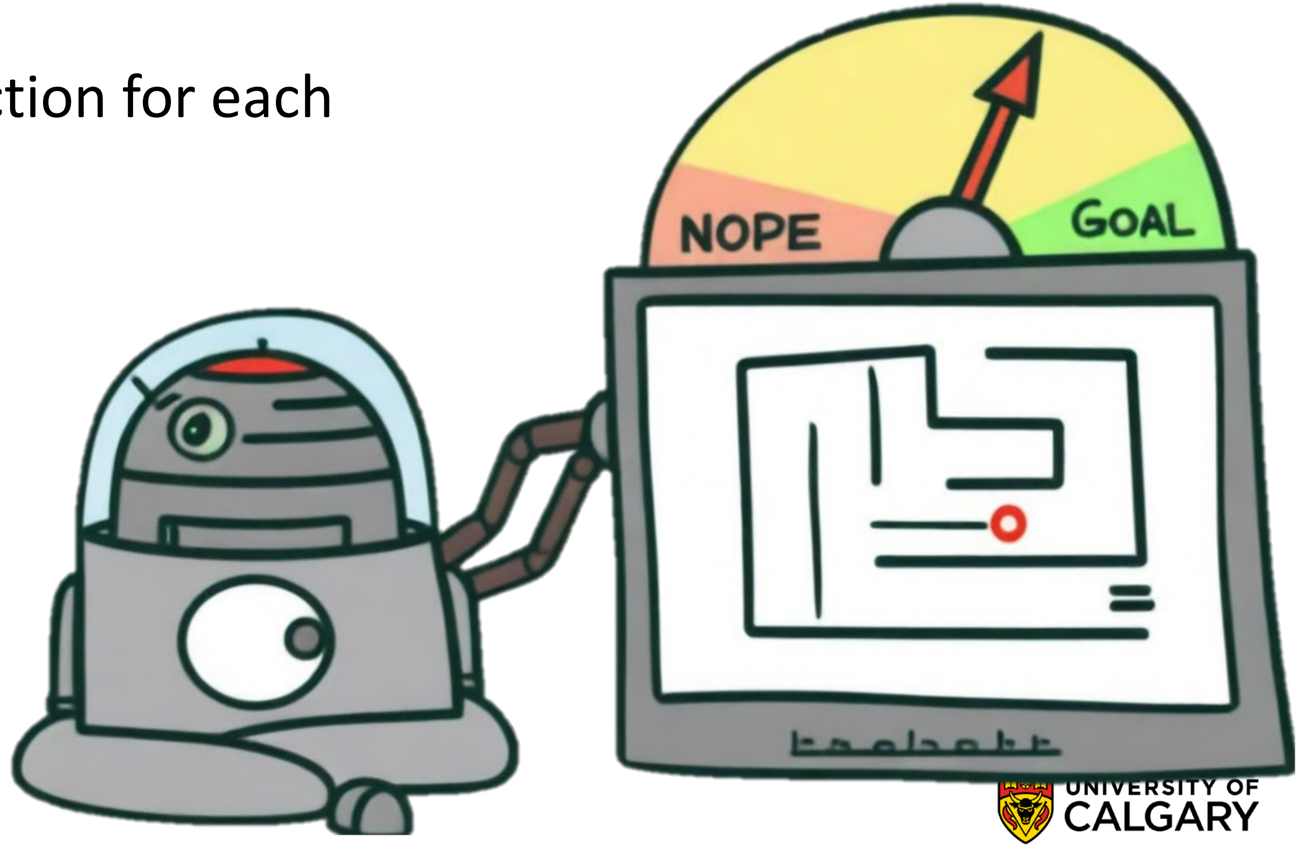
Outline

- Informed Search
- Best-First Search
- Greedy Search
- A* Search
- Comparison and Use
- Admissable Heuristics
- Generating Admissable Heuristics

Informed Search

Best-first Search

- **Informed search** methods have access to a **heuristic function** that estimates the cost of a solution
- **Best-First Search**: use an evaluation function for each node estimate of “desirability”
- Rationality!
- Special cases:
  - greedy search
  - A∗ search

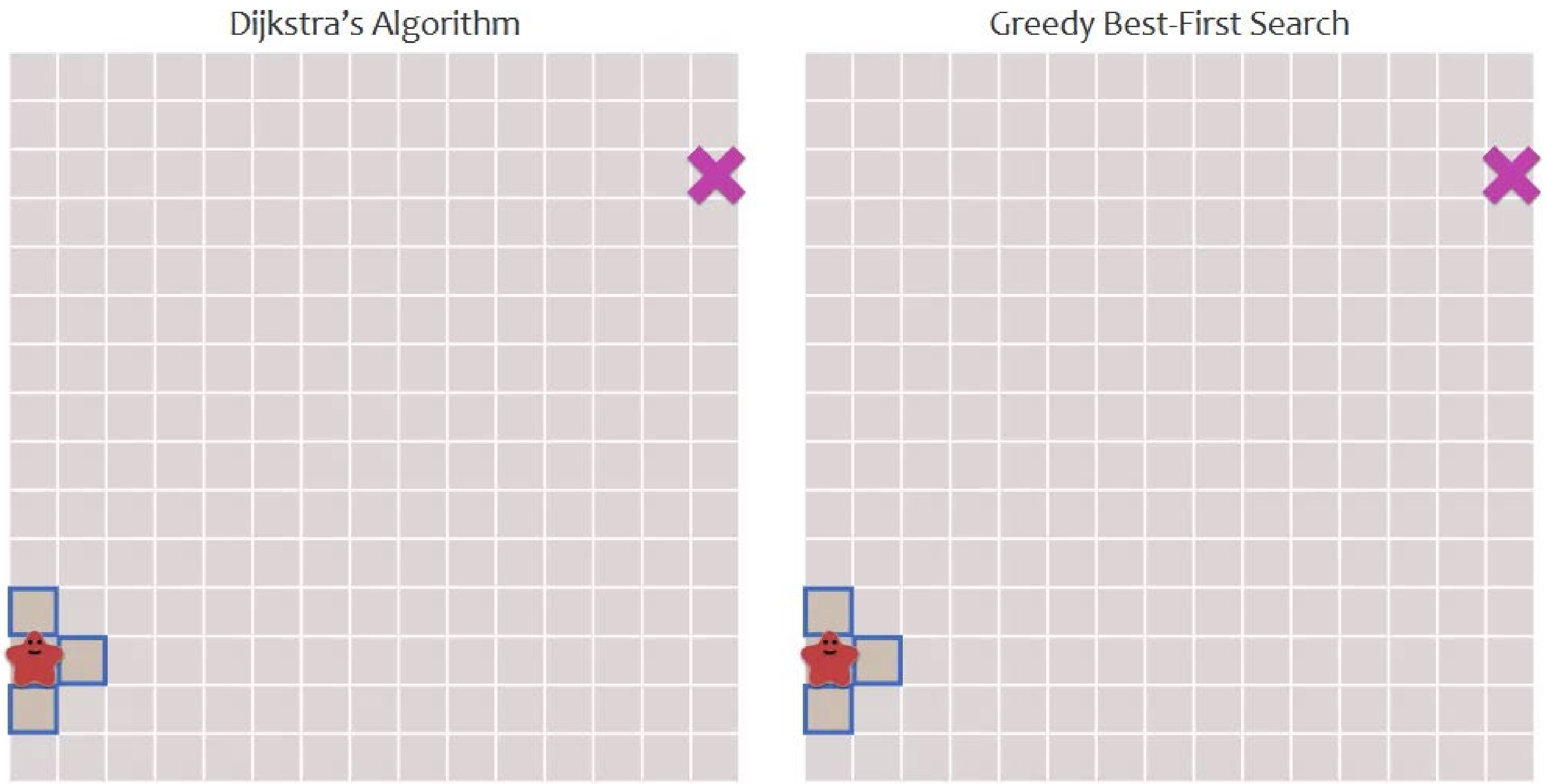


Greedy Search

Greedy Search

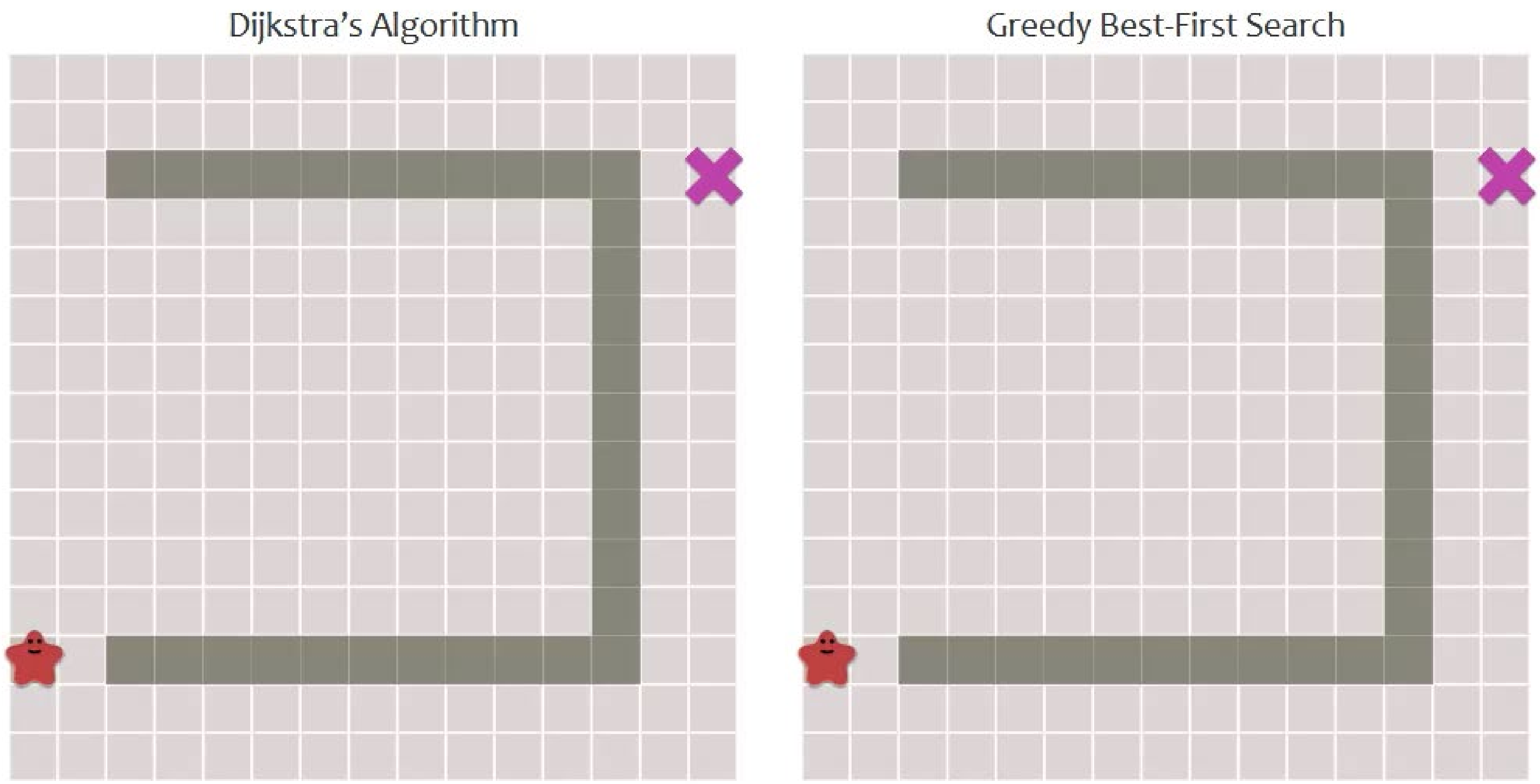
- Evaluation function **h** (heuristic)
- Estimate value of node expansion to solution and perform it next
- Variant of uniform cost search
  - costing is not heuristic and based on specific problem
- Greedy search expands the node that appears to be closest to goal
  - As evaluated by **h**

Greedy Search



https://www.redblobgames.com/pathfinding/a-star/introduction.html#greedy-best-first

Greedy Search



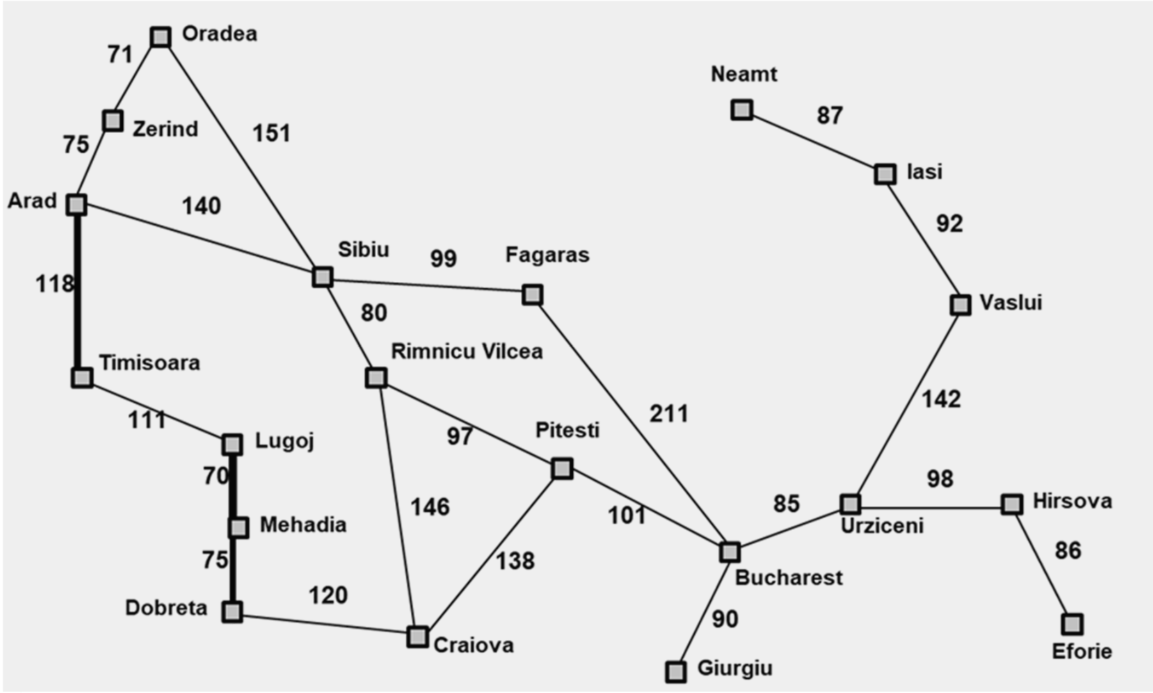
https://www.redblobgames.com/pathfinding/a-star/introduction.html#greedy-best-first

UNIVERSITY OF CALGARY

Example: Romania

- Currently in Arad.
- Need to get to Bucharest

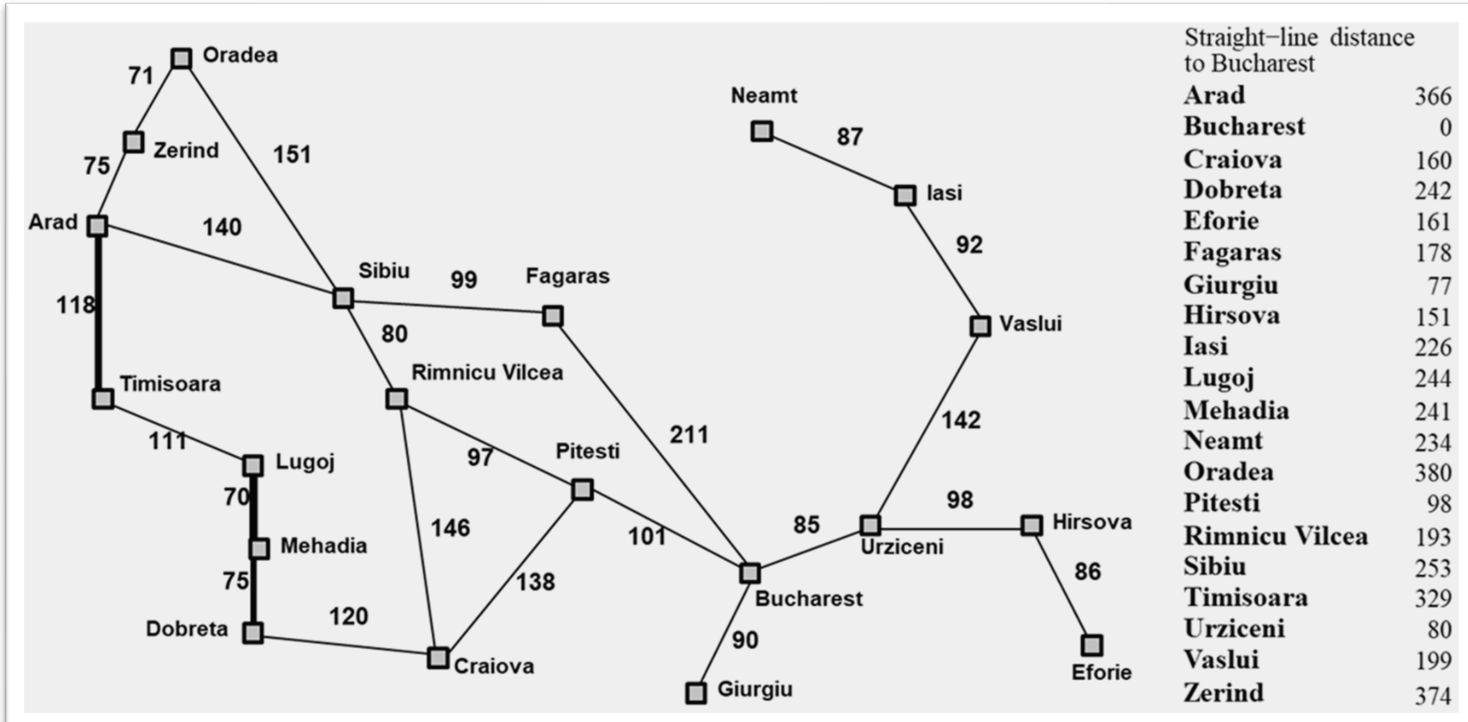
- Formulate goal:
  - be in Bucharest
- Formulate problem
  - states: various cities
  - actions: drive between cities
- Find solution
  - sequence of cities



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

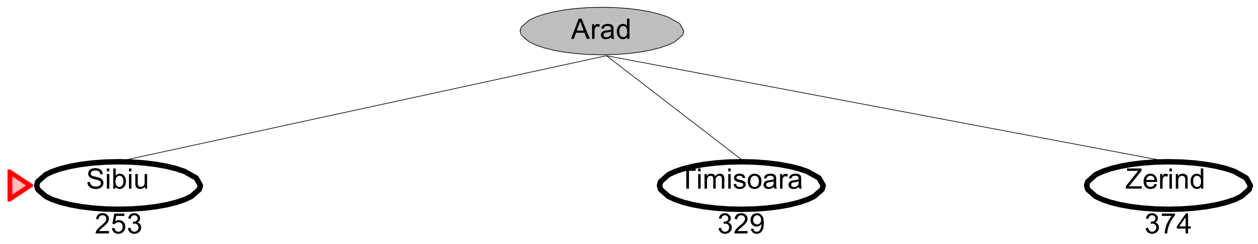
Greedy search example

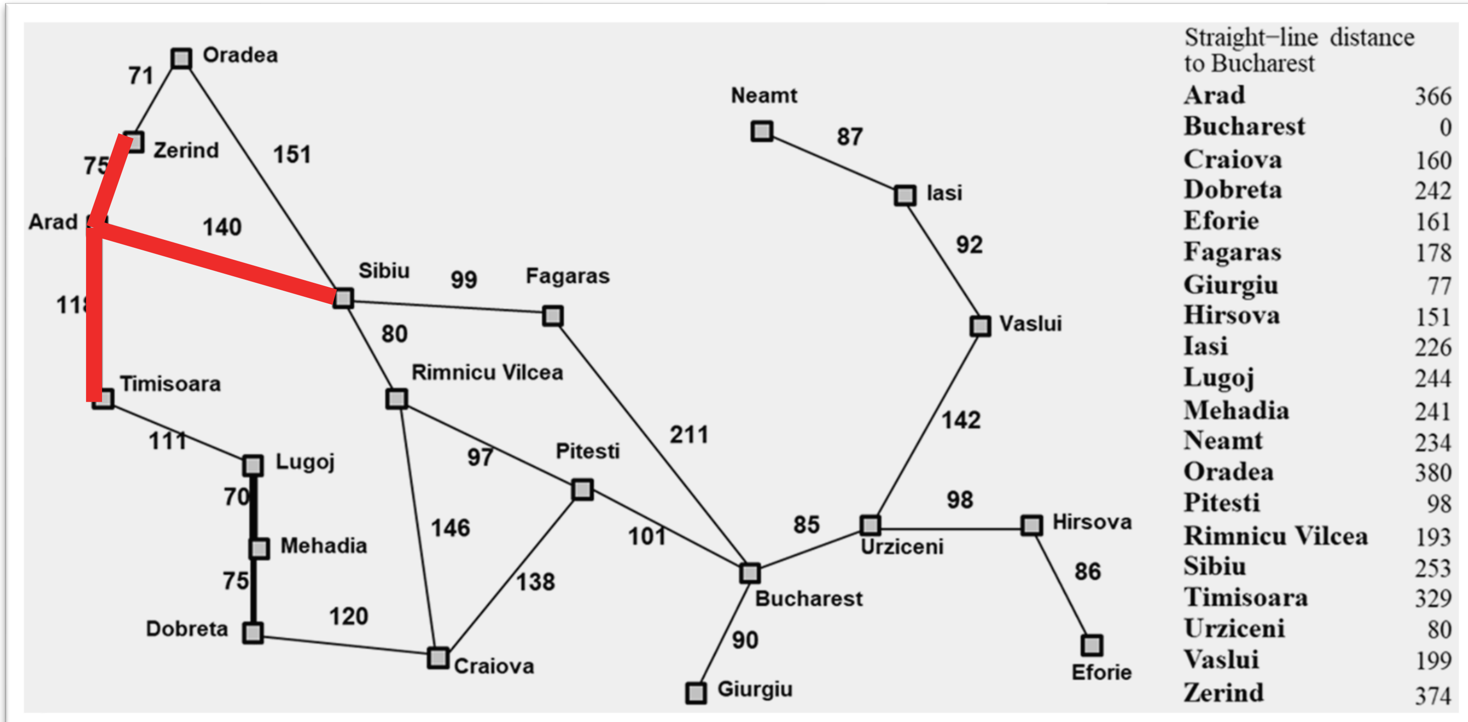
E.g., $h_{\mathrm{SLD}}(n)$ = straight-line distance from n to Bucharest



Greedy search example

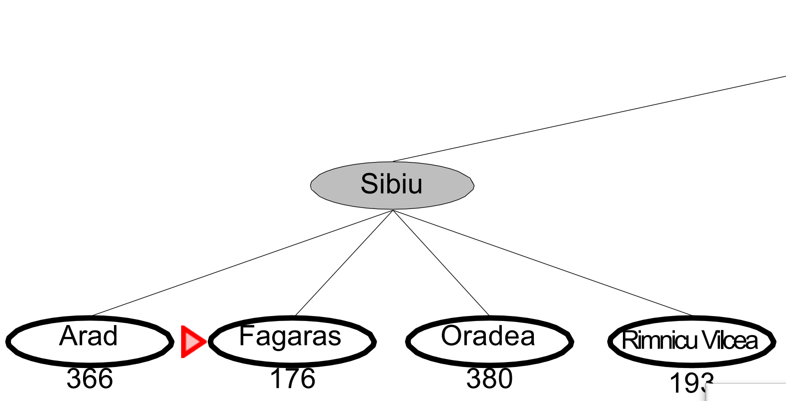
E.g., $h_{\mathrm{SLD}}(n)$ = straight-line distance from n to Bucharest

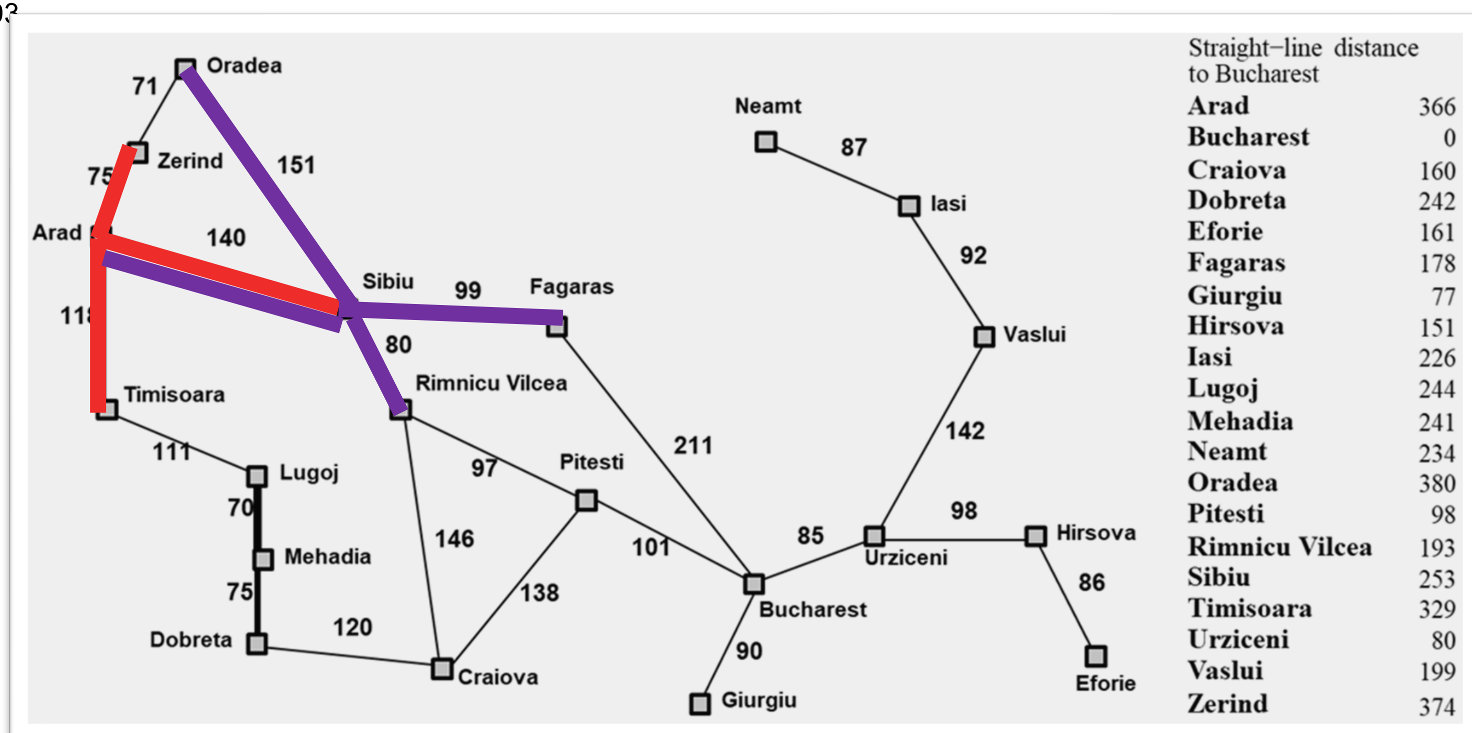




Greedy search example

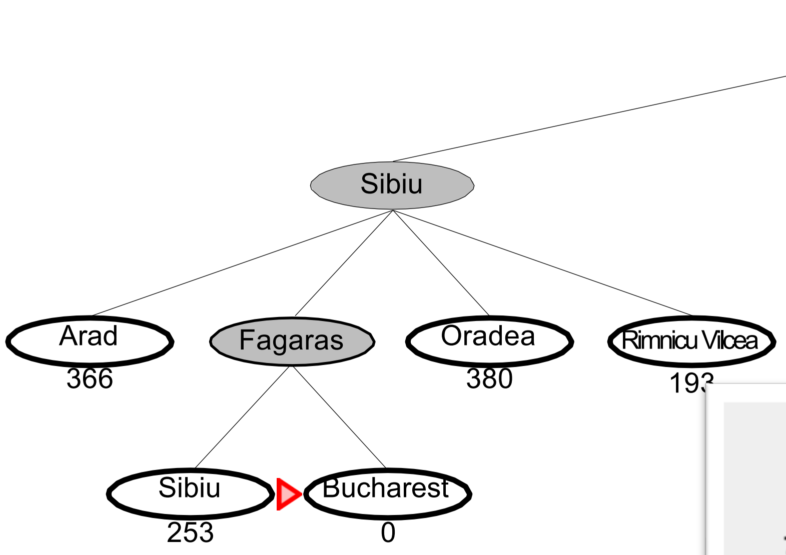
E.g., $h_{\mathrm{SLD}}(n)$ = straight-line distance from n to Bucharest

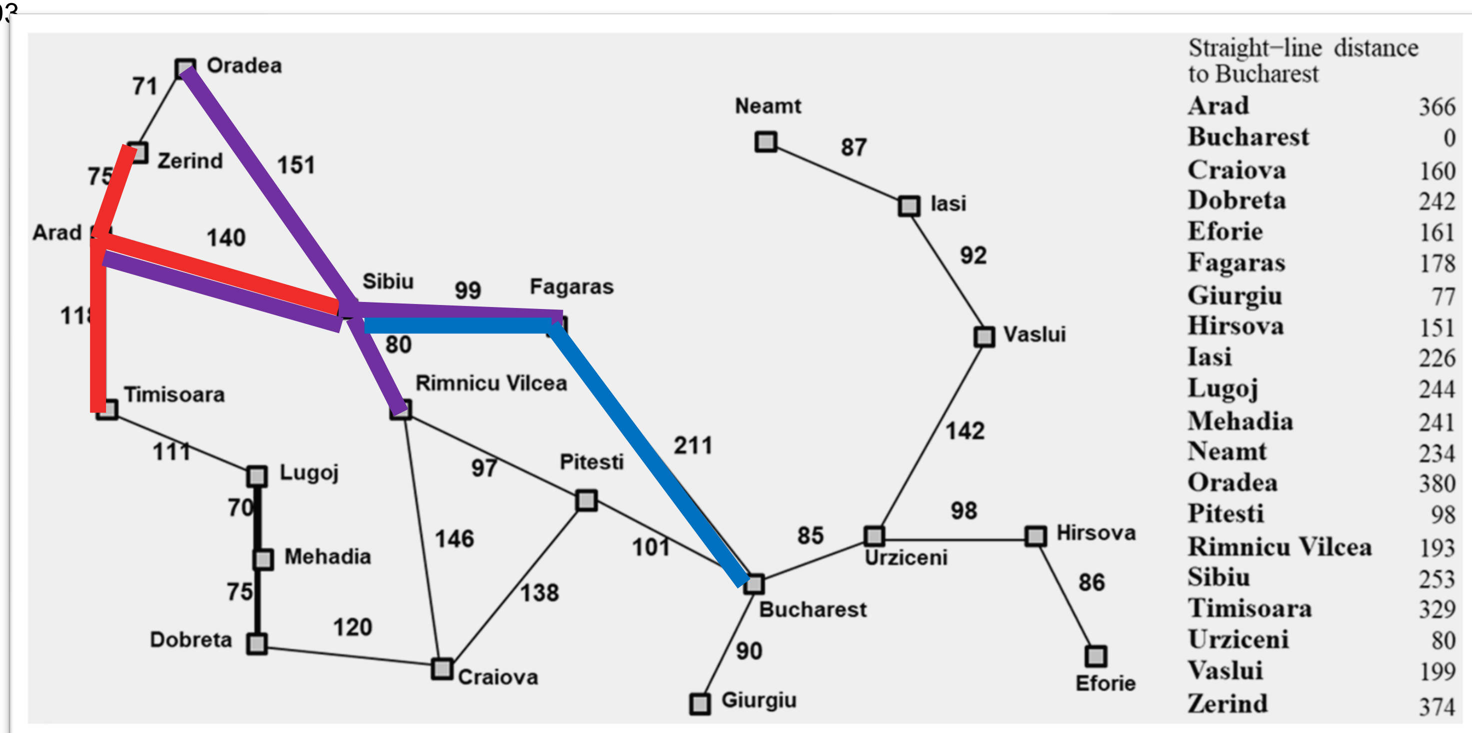




Greedy search example

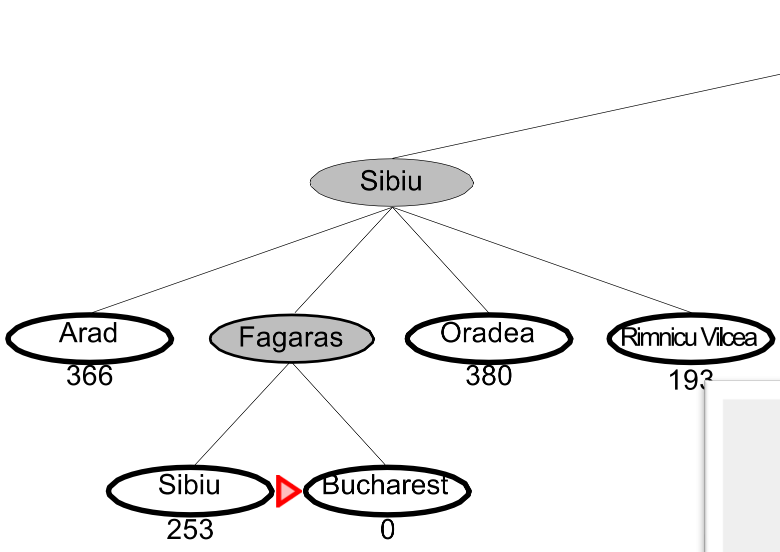
E.g., $h_{\mathrm{SLD}}(n)$ = straight-line distance from n to Bucharest



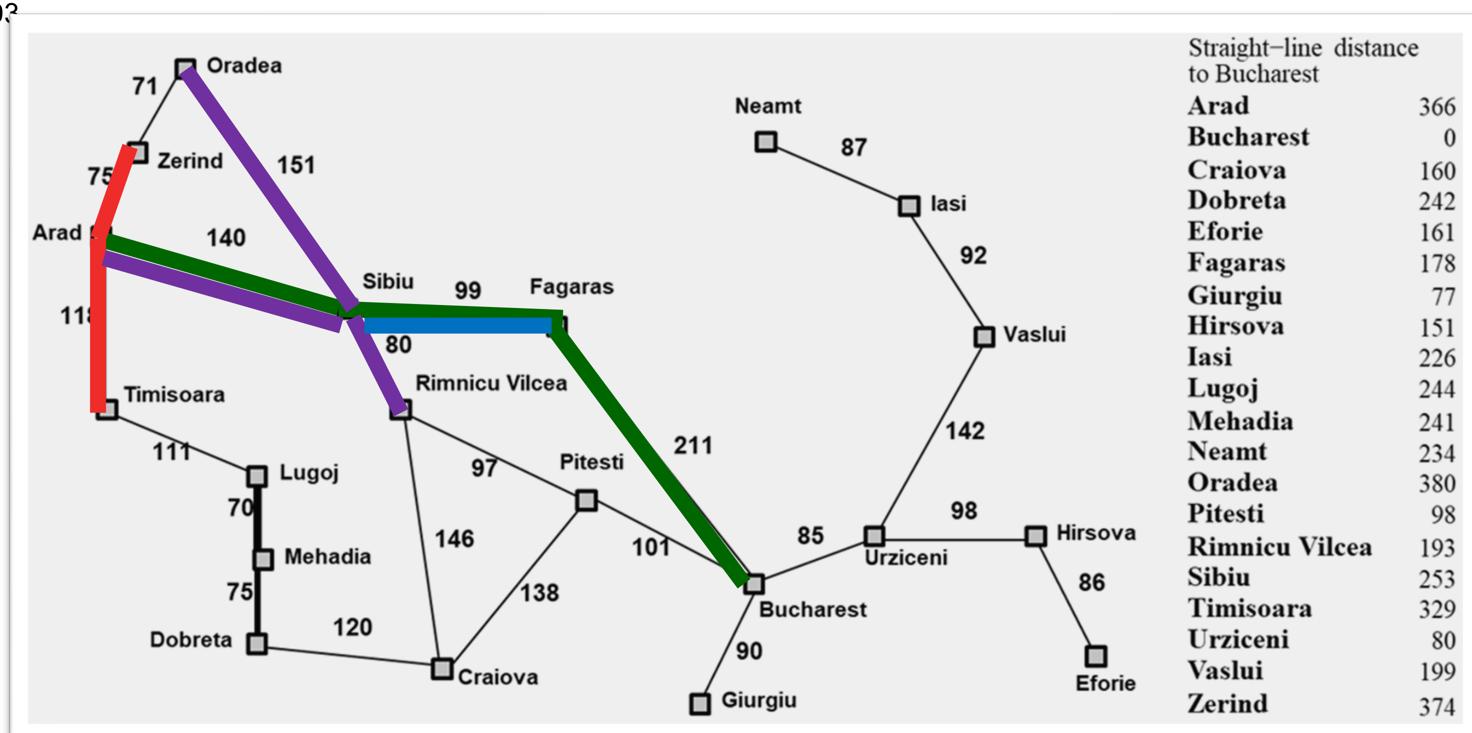


Greedy search example

E.g., $h_{SLD}(n)$ = straight-line distance from n to Bucharest

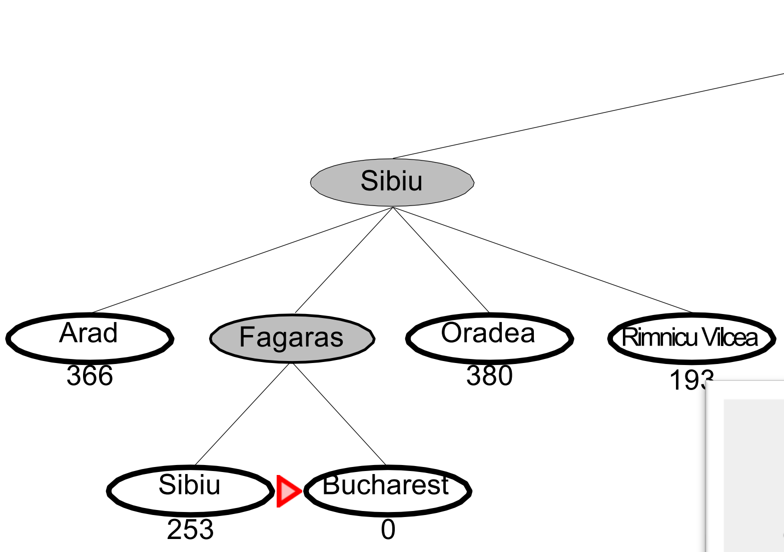


140+99+211



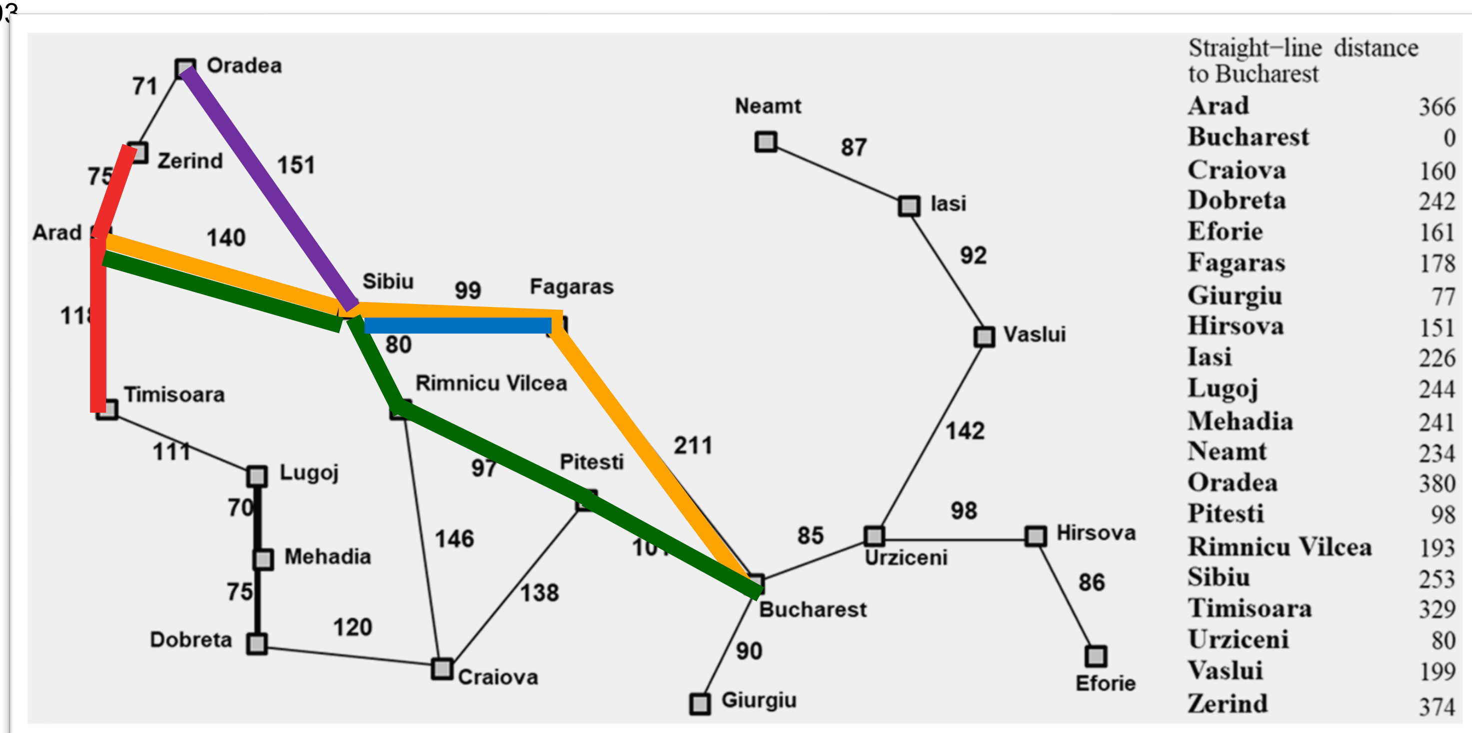
Greedy search example

E.g., $h_{\mathrm{SLD}}(n)$ = straight-line distance from n to Bucharest



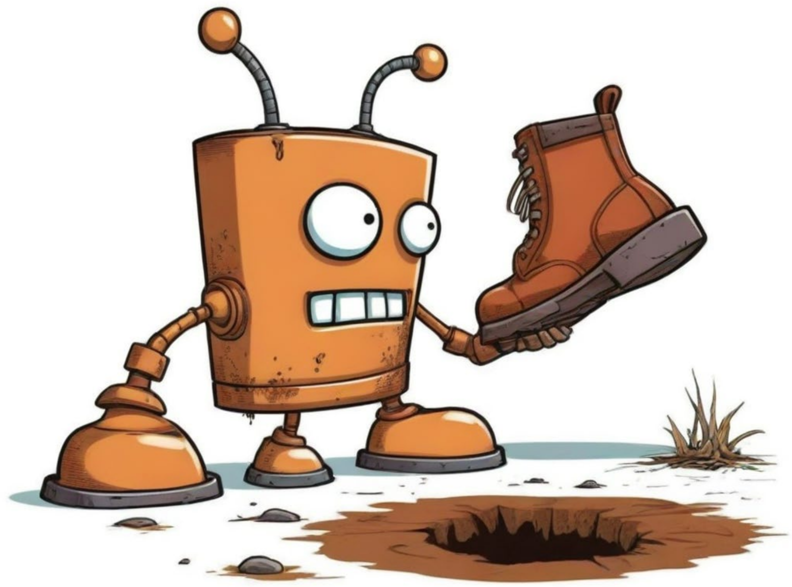
But 140+99+211 is more than
140+80+97+101

By following a local optima via heuristic we missed the global optima



Properties of Greedy Search

- Complete: No
  - can get stuck in infinite tree
  - Complete in finite space with repeated-state checking
- Time: exponential
  - but a good heuristic can give dramatic improvement
- Space: Keeps all nodes in memory
- Optimal: No (we reach Bucharest and don't explore other paths)



UNIVERSITY OF CALGARY

A* Search

A* search

- Idea: Start greedy (only forward looking was an issue)
  - Add backwards looking, confirm one property about new heuristic

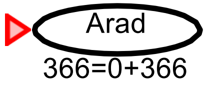
- Evaluation function f (n) = g(n) + h(n)
  - g(n) = cost so far to reach n (backwards looking)
  - h(n) = estimated cost to goal from n (greedy forward-looking part)
  - f (n) = estimated total cost of path (**A* heuristic**)

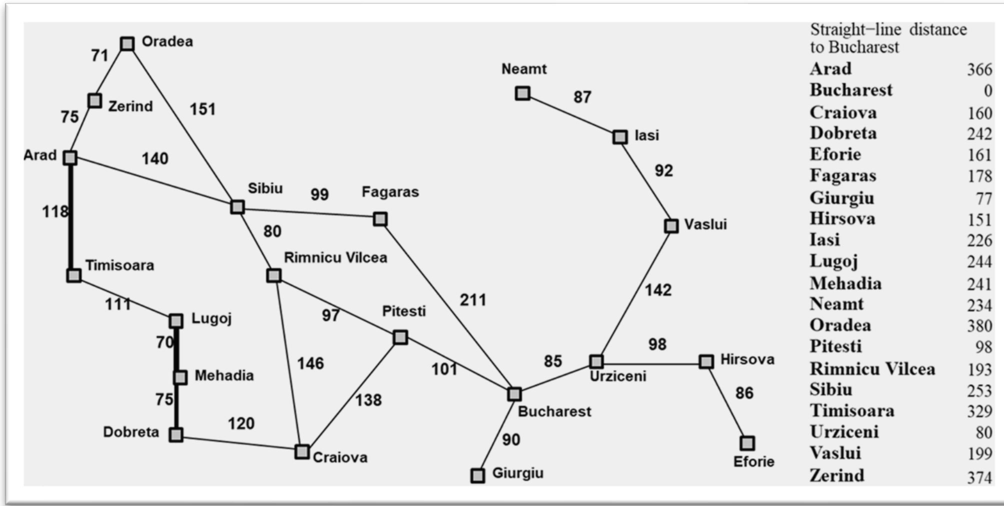
- A∗ search requires an **admissible heuristic** (fully defined later)
  - Short defn: **never overestimates the cost**

- **Theorem: A∗ search is optimal**

A* search example

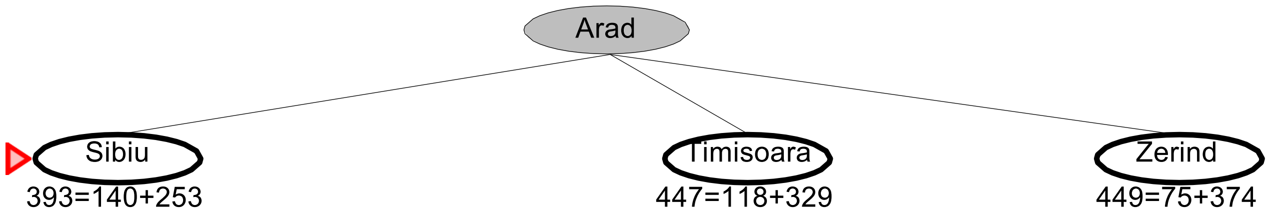
E.g., $h_{\mathrm{SLD}}(n)$ = straight-line distance from n to Bucharest

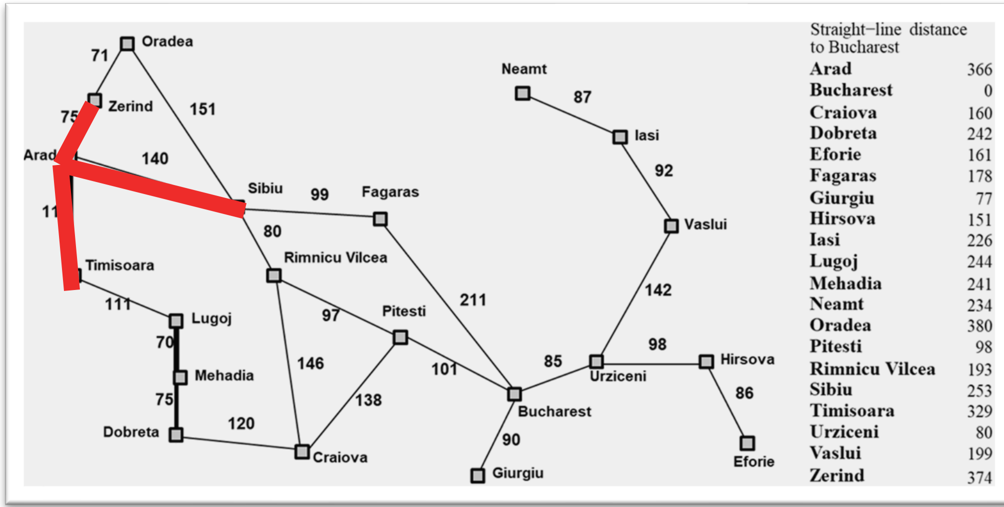




A* search example

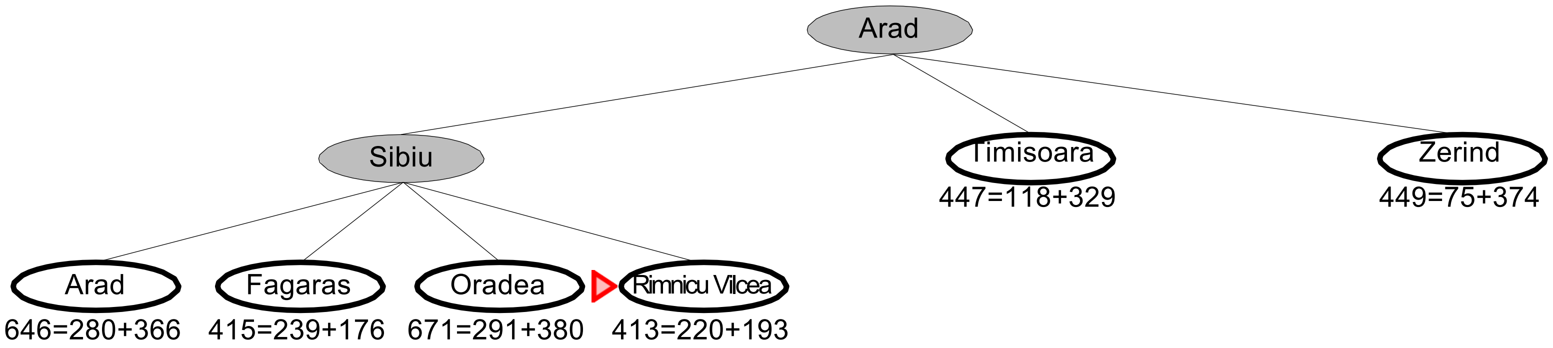
E.g., $h_{\mathrm{SLD}}(n)$ = straight-line distance from n to Bucharest



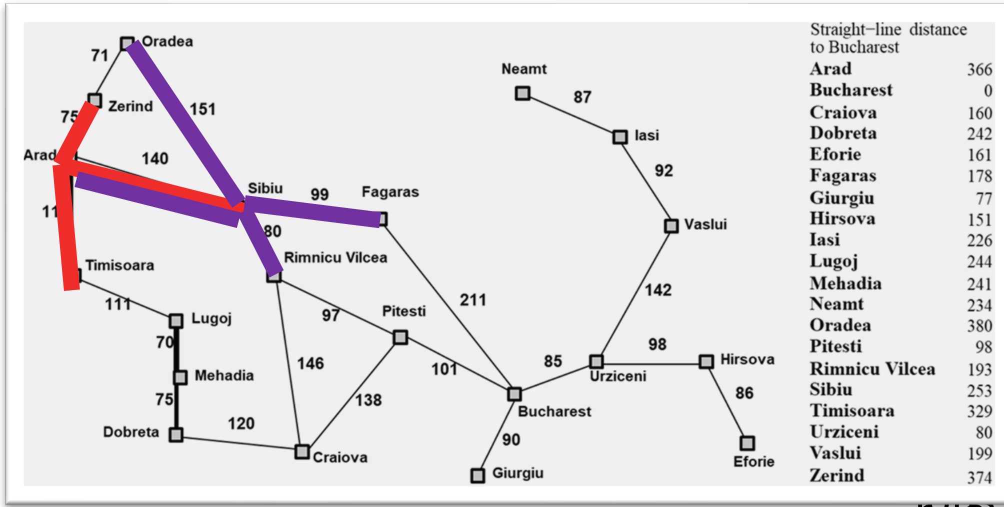


A* search example

E.g., $h_{\mathrm{SLD}}(n)$ = straight-line distance from n to Bucharest

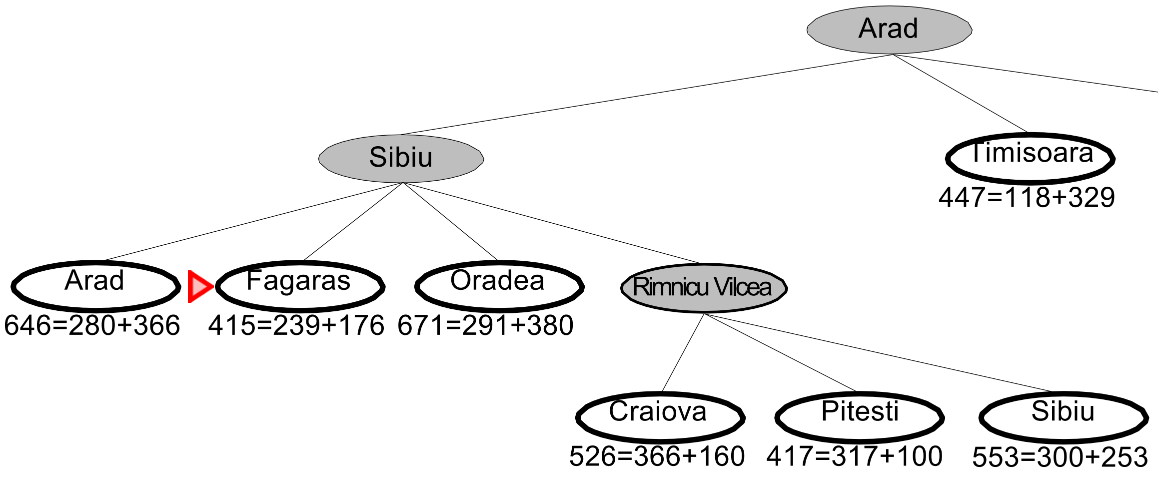


Here we are different than Greedy as we explore Rimnicu Vilcea instead of Faragas next due to heuristic

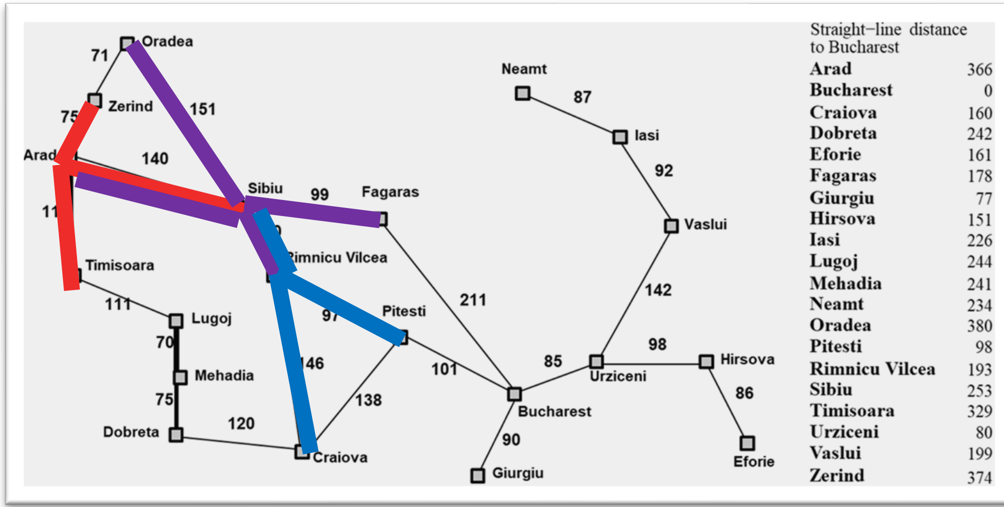


A* search example

E.g., $h_{\mathrm{SLD}}(n)$ = straight-line distance from n to Bucharest

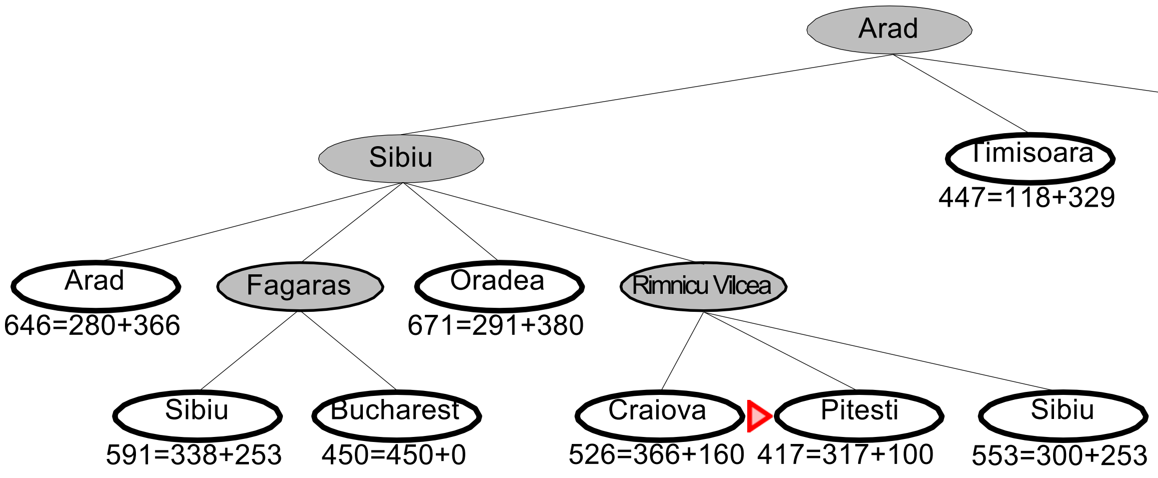


We return to look at Faragas because paths out of Rimnicu Vilcea aren't clearly better

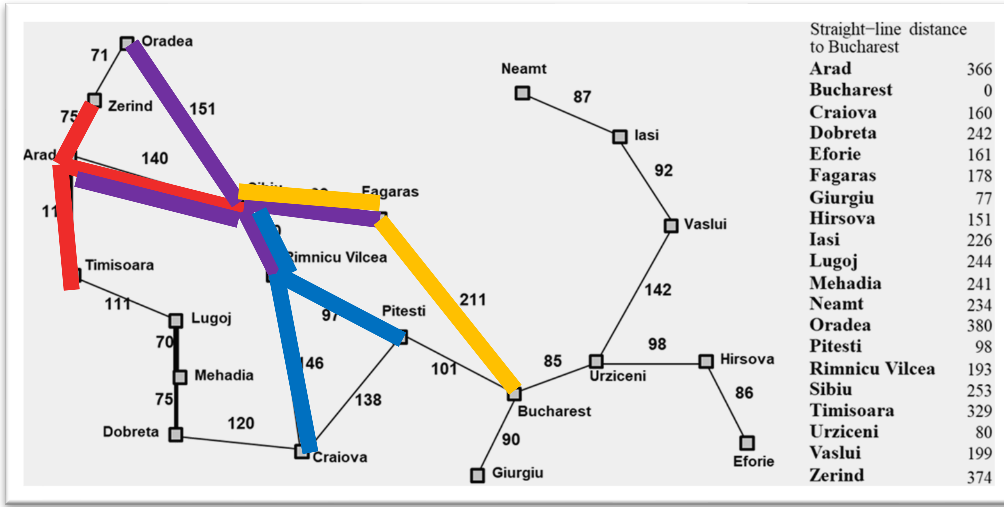


A* search example

E.g., $h_{\mathrm{SLD}}(n)$ = straight-line distance from n to Bucharest

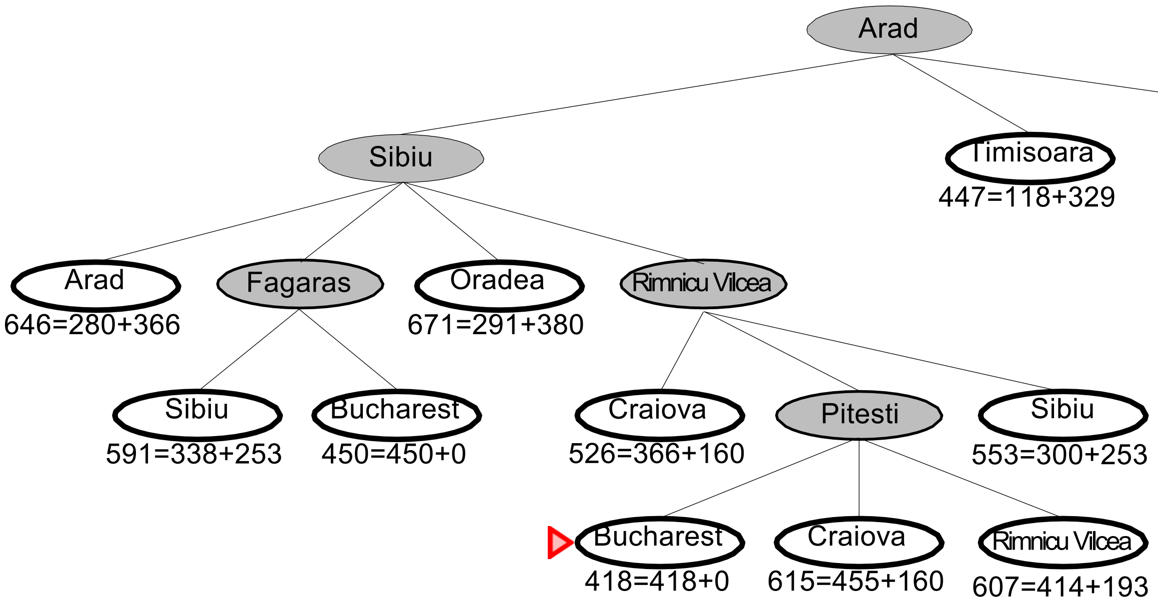


We go back to Rimnicu Vilcea to explore as at path there is more intriguing than through Faragas (at the moment)

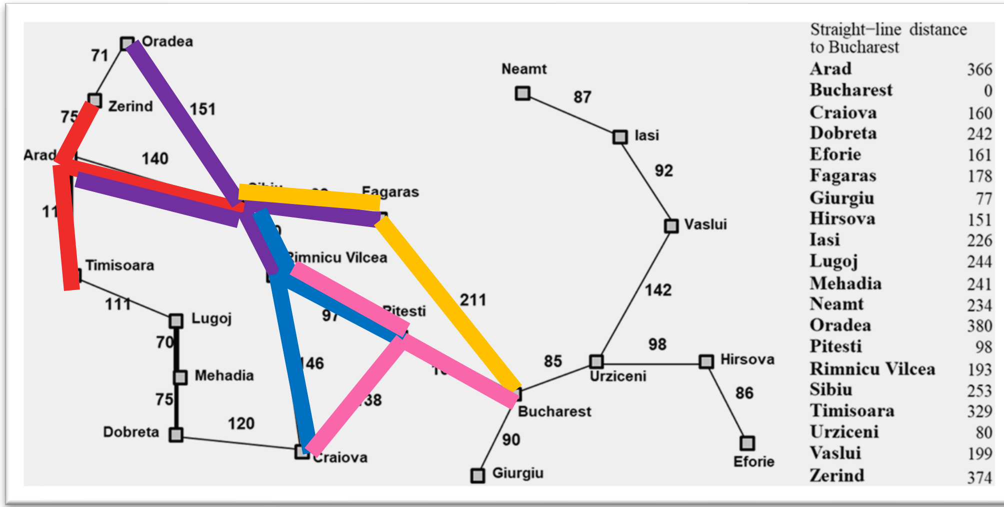


A* search example

E.g., $h_{\mathrm{SLD}}(n)$ = straight-line distance from n to Bucharest

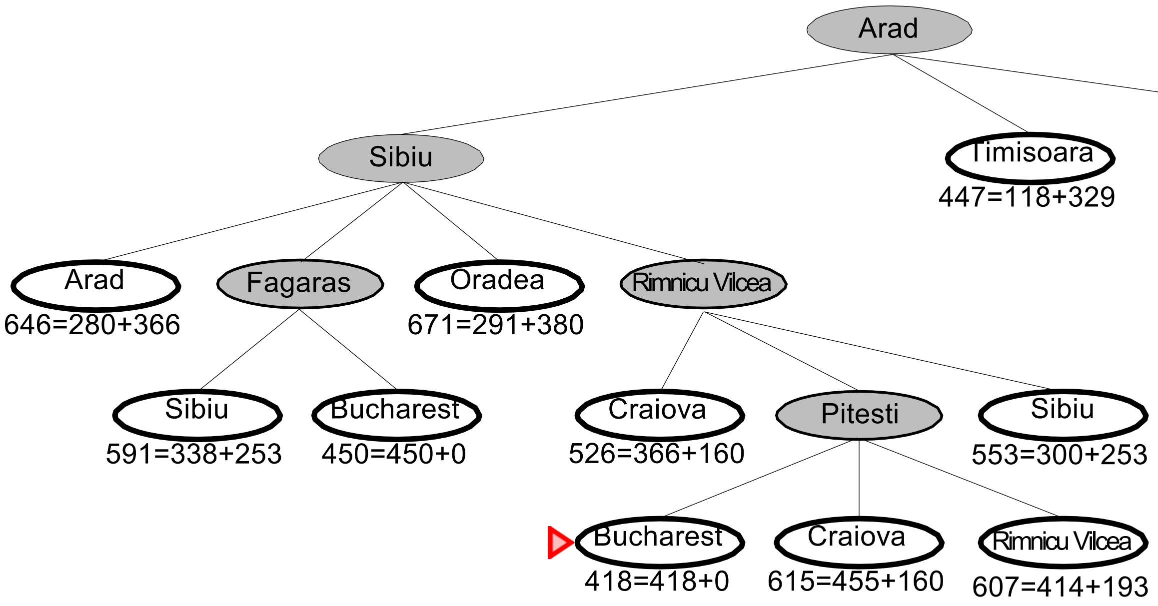


Expand Pitesti

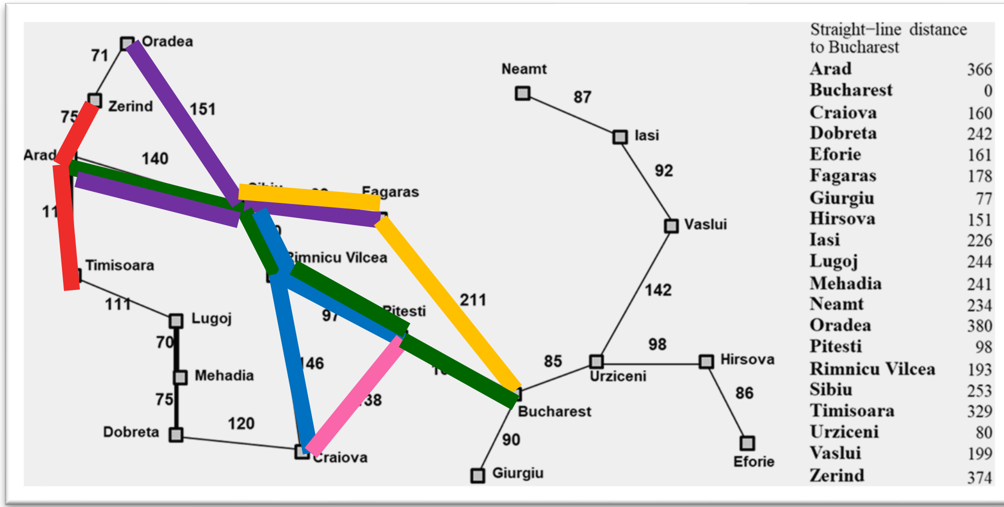


A* search example

E.g., $h_{\mathrm{SLD}}(n)$ = straight-line distance from n to Bucharest

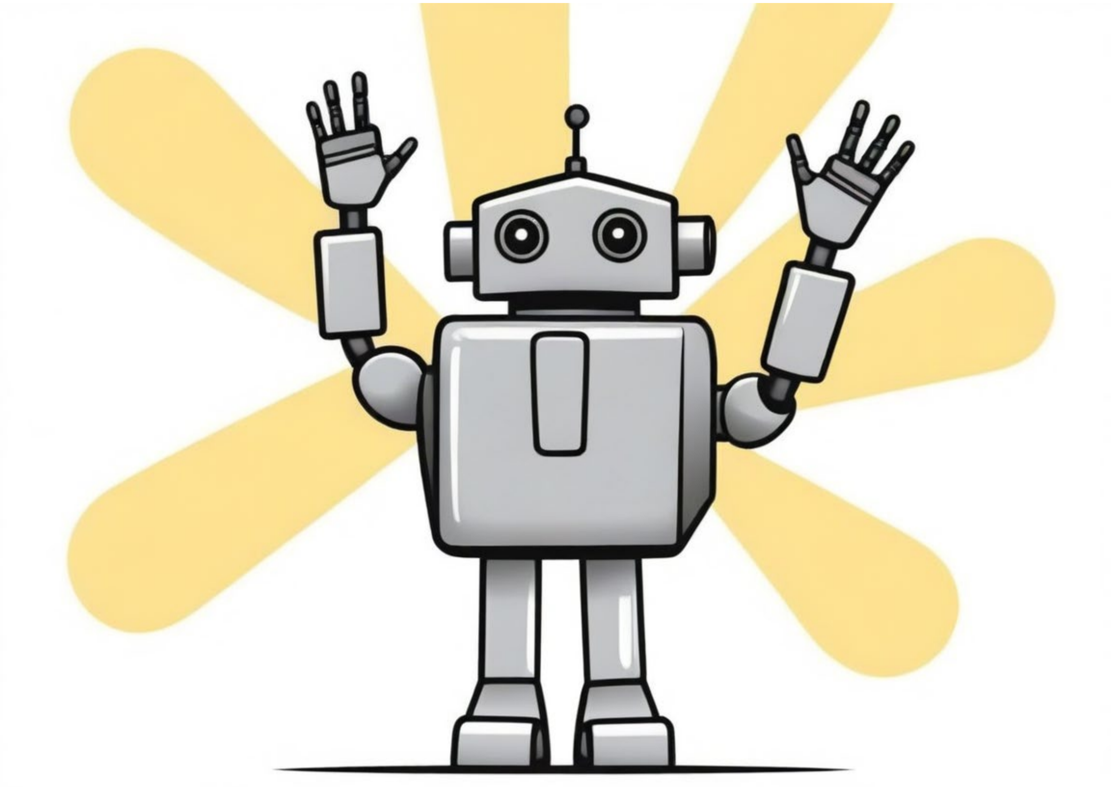


We go to Bucharest as minimal next transition (but out of Pitesti instead of Faragas!) and find the shortest path!



Properties of A* Search

- Complete: Yes
  - Unless infinite expansions
- Time: exponential
  - but only in regard to heuristic error relative to solution
- Space: Keeps all nodes in memory
- Optimal: Yes
  - Cannot move to a great cost contour until smaller one is checked, i.e. will always find smallest first

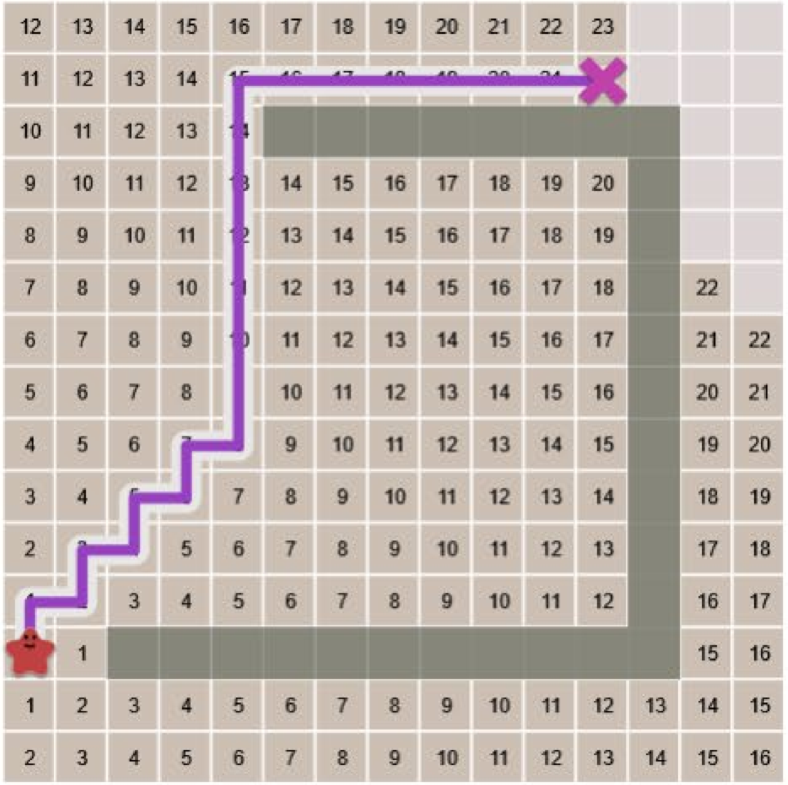


UNIVERSITY OF CALGARY

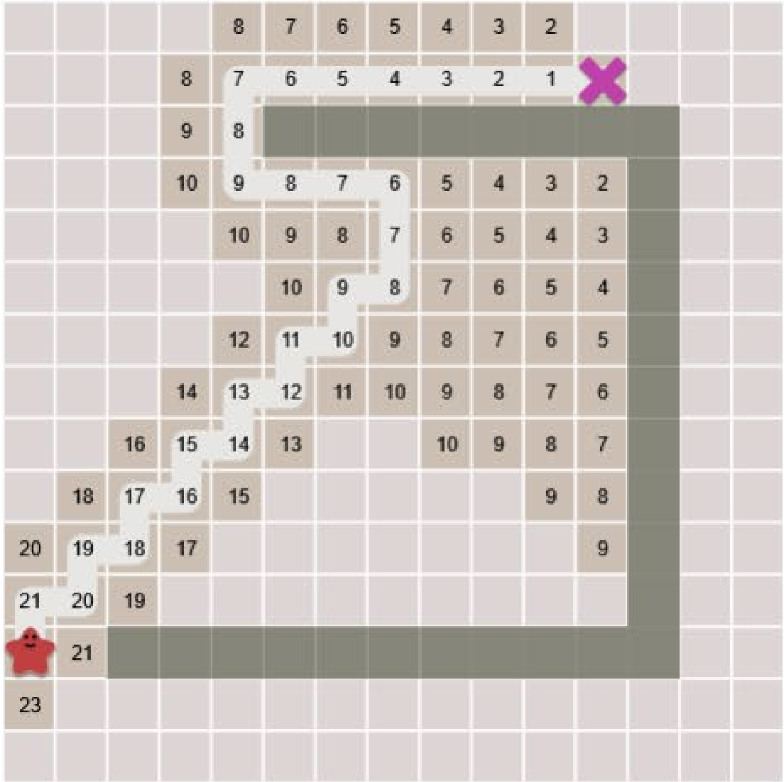
Comparison and Use

Comparison

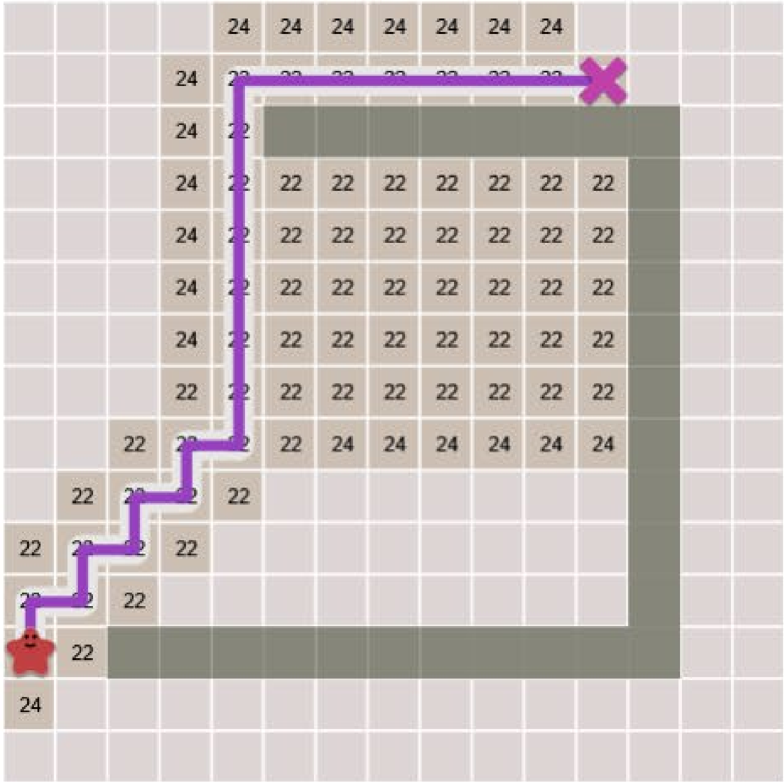
Dijkstra's Algorithm



Greedy Best-First



A* Search



Uniform Cost

https://www.redblobgames.com/pathfinding/a-star/introduction.html#astar

Admissable Heuristics

Admissable Heuristic

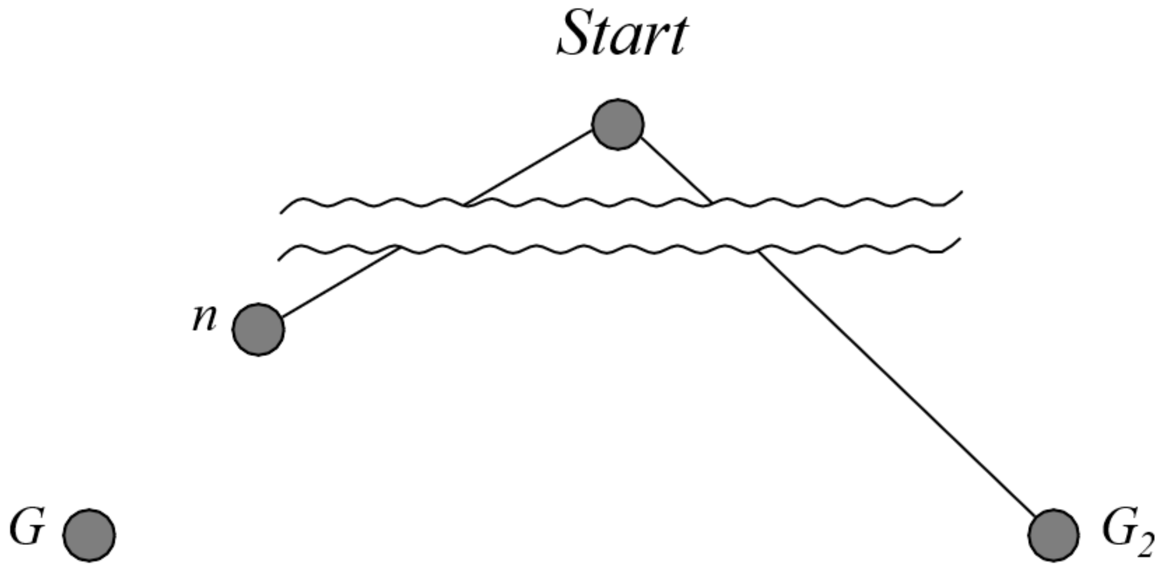
- Evaluation function **f** = g + h
  - g = cost so far to reach n
  - h = estimated cost to goal (heuristic)
  - f = estimated total cost goal

- **h is still an estimate of cost allows guidance of what to explore first**
- **An admissable heuristic h -> never overestimates**
  - If something has true additional cost of 500 then h never returns larger than 500
  - We are allowed to treat things as better than they truly are
    - How often we are inaccurate like this just costs us wasted effort

- **A good admissible heuristic will be more accurate, a useless one would estimate 0 and have no benefit to search**

Optimality of A∗ (standard proof)

- Suppose some suboptimal goal G_2 has been generated and is in the queue. Let n be an unexpanded node on a shortest path to an optimal goal G.

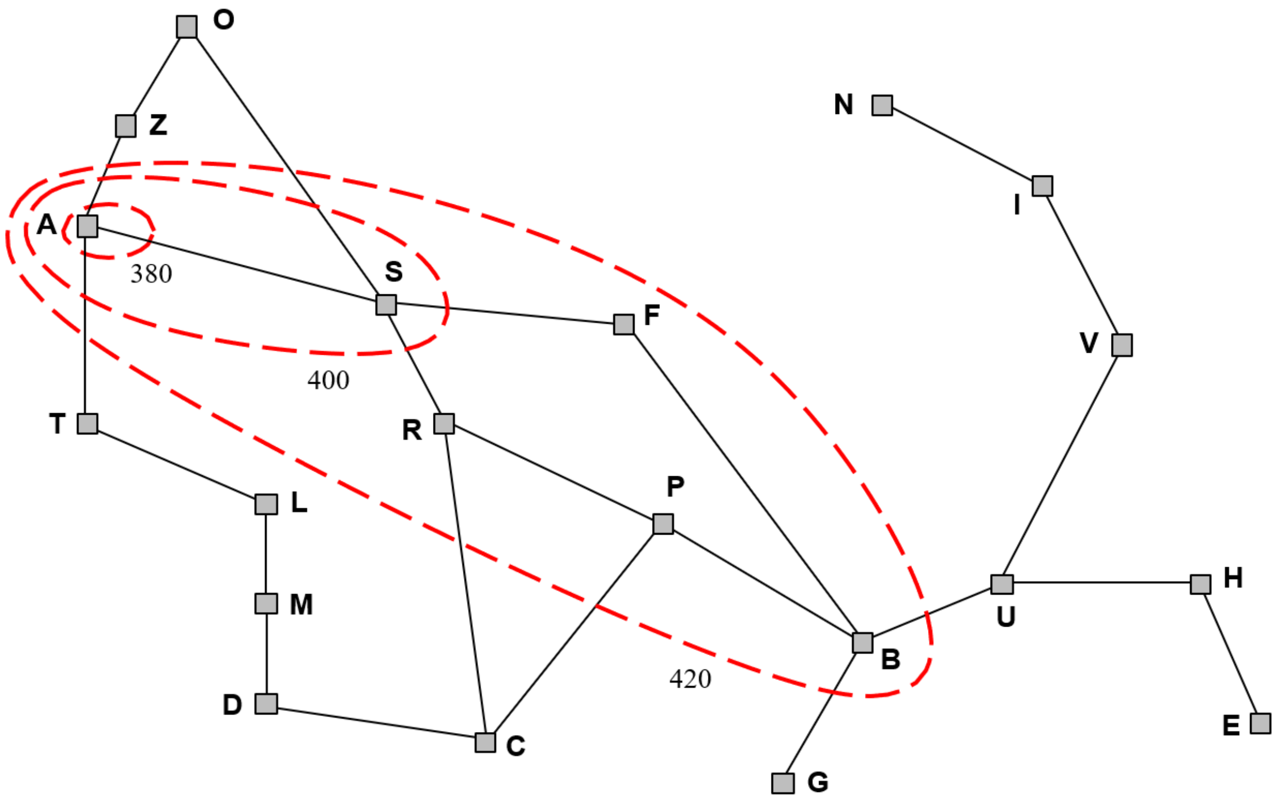


$$
\begin{aligned}
f(G_2) &= g(G_2) && \text{since } h(G_2) = 0 \\
&> g(G) && \text{since } G_2 \text{ is suboptimal} \\
&\geq f(n) && \text{since } h \text{ is admissible}
\end{aligned}
$$

Since $f(G_2) > f(n)$, A* will never select G_2 for expansion

Optimality of A∗ (more useful)

- Lemma: A∗ expands nodes in order of increasing f value
- Gradually adds “f -contours” of nodes (lowest cost breadth like expansion)

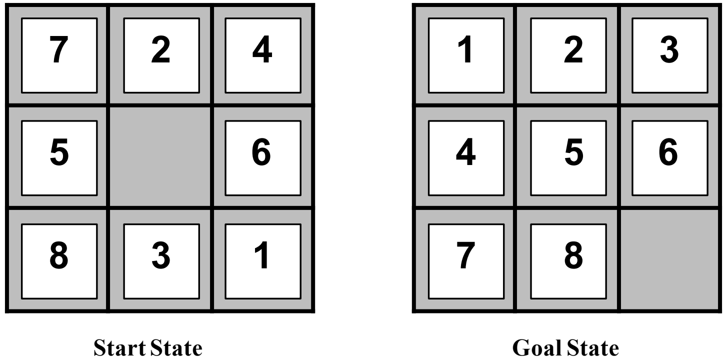


Generating Admissable Heuristic

Relax

Admissible heuristics

- E.g., for the 8-puzzle:



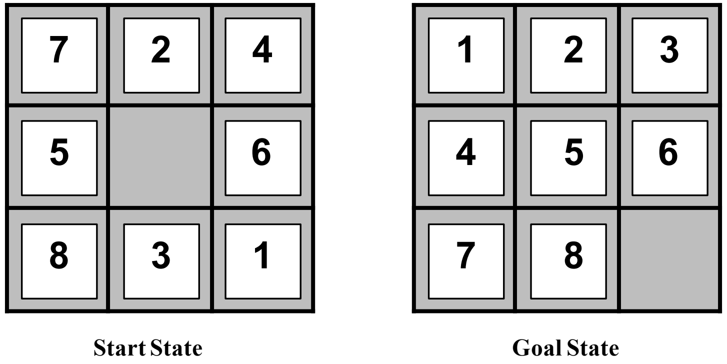
- $h_1(n)$ = number of misplaced tiles
- $h_2(n)$ = total Manhattan distance
  - (i.e., no. of squares from desired location of each tile)

- $h_1(S)$ =
- $h_2(S)$ =

https://murhafsousli.github.io/8puzzle/#/

Admissible heuristics

- E.g., for the 8-puzzle:



- $h_1(n)$ = number of misplaced tiles
- $h_2(n)$ = total Manhattan distance
  - (i.e., no. of squares from desired location of each tile)

- $h_1(S)$ =?? 6
- $h_2(S)$ =?? 4+0+3+3+1+0+2+1 = 14

Dominance

- If $h_2(n) \geq h_1(n)$ for all n (both admissible), then h_2 dominates h_1 and is better for search
- Typical search costs:
- $d = 14$
  - Iterative deepening = 3,473,941 nodes
  - A$*(h_1)$ = 539 nodes
  - A$*(h_2)$ = 113 nodes
- $d = 24$
  - Iterative deepening $\approx$ 54,000,000,000 nodes
  - A$*(h_1)$ = 39,135 nodes
  - A$*(h_2)$ = 1,641 nodes

Dominance

- If $h_2(n) \geq h_1(n)$ for all n (both admissible), then h_2 dominates h_1 and is better for search

- Given any admissible heuristics h_a, h_b, $h(n) = \max(h_a(n), h_b(n))$
- is also admissible and dominates h_a, h_b

Relaxed problems

- Admissible heuristics can be derived from the exact
- solution cost of a relaxed version of the problem
- If the rules of the 8-puzzle are relaxed so that a tile can move anywhere, then $h_1(n)$ gives the shortest solution
- If the rules are relaxed so that a tile can move to any adjacent square, then $h_2(n)$ gives the shortest solution
- Key point: the optimal solution cost of a relaxed problem is no greater than the optimal solution cost of the real problem

Summary

Summary

- Informed search methods have access to a heuristic function h that estimates the cost of a solution
- Best-First Search is a node expansion version that ranks nodes using heuristic evaluation function of best gain
- Greedy Search in best-first algorithm that guesses cost of adding node to find one solution, but heuristics does not guarantee optimal
- A* Search is variant of greedy that uses admissible heuristic to explore different options of paths and guarantees optimal (but more exploration).
- Admissable Heuristics are optimistic cost predictions that help guide exploration but don't make mistakes that miss best solution.
- You can often relax requirements of problem to generate admissible heuristics, and combine multiple to get an even better one

Next…complex search

Jonathan Hudson, Ph.D.
jwhudson@ucalgary.ca
https://cspages.ucalgary.ca/~jwhudson/

UNIVERSITY OF CALGARY