Introduction to Artificial Intelligence

Lecture 13: Intelligent search II

October 20, 2025

Lecture plan

- Intelligent search
  - Wrapping up un-informed search
  - Informed search algorithms: greedy, A^* search

Example search problems

- Traveling salesperson problem (TSP): find a minimum cost traveling tour across all the locations

- VLSI layout problem: positioning millions of components and connections on a chip to minimize area, minimize circuit delays, minimize stray capacitances, and maximize manufacturing yield
  - Real robots must deal with errors in their sensor readings and motor controls, with partial observability, and with other agents that might alter the environment

- Automatic assembly sequencing: find an order in which to assemble the parts of some object
  - If the order is wrong, there will be no way to add some part later in the sequence without undoing some of the work already done

Summary of uninformed search algorithms

Scenario	Best Algorithm	Why?
Equal step costs, shallow solution	**Breadth-first search**	Guarantees shortest path by steps
Memory limited, deep tree	**Depth-first search**	Linear space complexity
Varying costs, need optimal solution	**Uniform cost search**	Finds least-cost path
Need all shortest paths	**Dijkstra's algorithm**	Computes complete shortest path tree

Informed search algorithms

- With the aid of domain-specific knowledge
  - Can help people solve hard problems without search
  - Can help computers find solutions more efficiently
- Informed heuristic search
  - Estimate the cost from a given node to a goal node
  - Take the estimate towards the goal into account when selecting the path
  - Cost function: $f(n) = g(n) + h(n)$
  - $g(n)$: actual cost from start to node n
  - $h(n)$: heuristic cost from node n to the target

Cost function

The choice of f determines the search strategy:

- Uniform cost search: $f(n) = g(n)$ (expand lowest path cost)
- Greedy search: $f(n) = h(n)$ (expand closest to goal)
- A^*: $f(n) = g(n) + h(n)$ (balance both factors)
- Weighted A^*: $f(n) = g(n) + w \times \ h(n)$

Heuristic function

- Heuristic function: $h(n)$ = estimated cost of the cheapest path from the state at node n to a goal state
  - $h(n)$ is arbitrary, non-negative, and problem-specific
  - If n is a goal node, $h(n) = 0$
  - $h(n)$ must be easy to compute (without search)

- Common heuristics functions for path planning
  - **Manhattan Distance:** $h(n) = |x_1 - x_2| + |y_1 - y_2|$
  - **Euclidean Distance:** $h(n) = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$
  - **Diagonal Distance:** $h(n) = \max(|x_1 - x_2|, |y_1 - y_2|)$

Example: Path planning in Romania

Abstracted graph representations

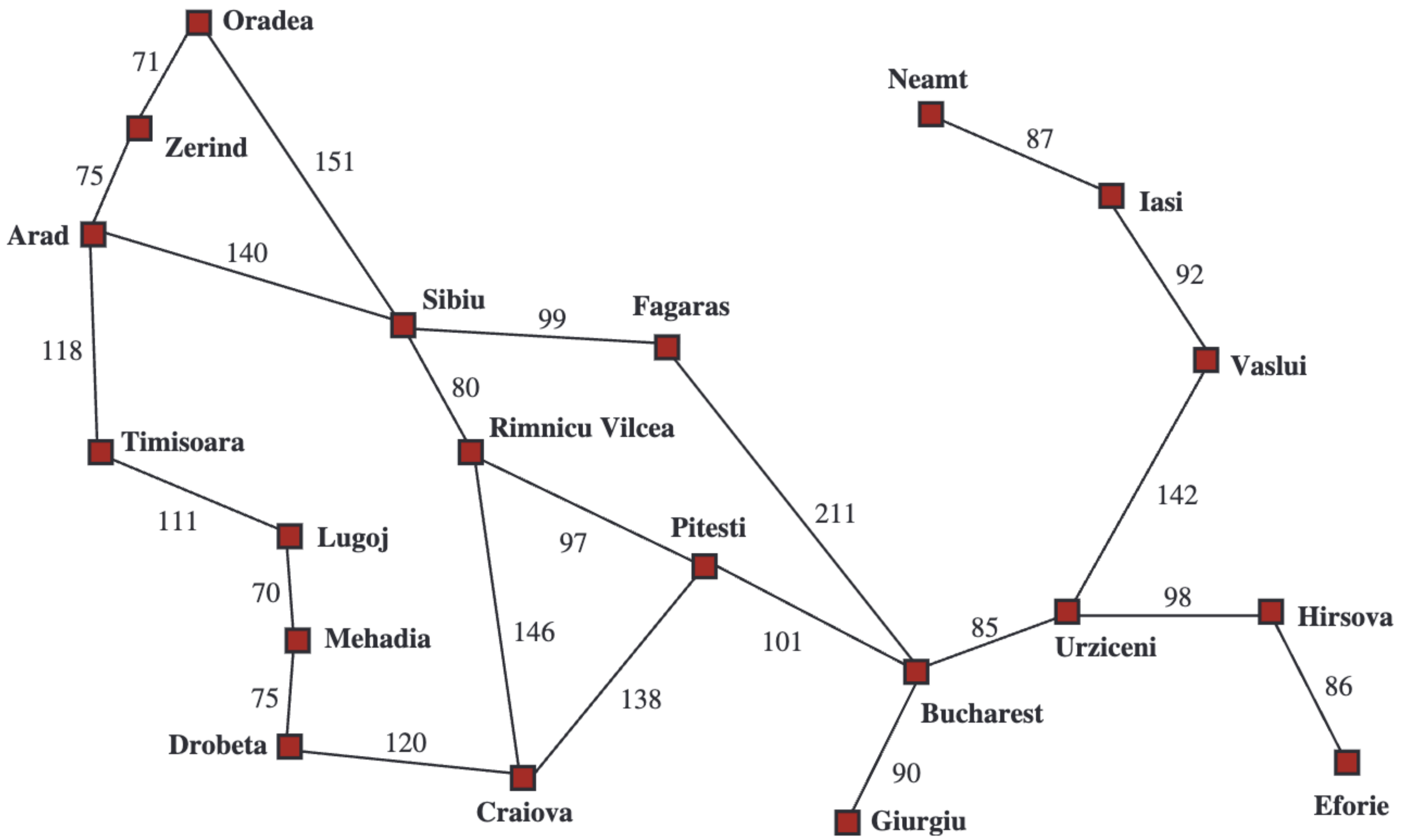


Illustration of greedy search

- Goal: minimize the estimated cost to the goal, $f(n) = h(n)$
- E.g. $\mathrm{h}_{\mathrm{SLD}}(n)$ = straight-line distance from n to Bucharest

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

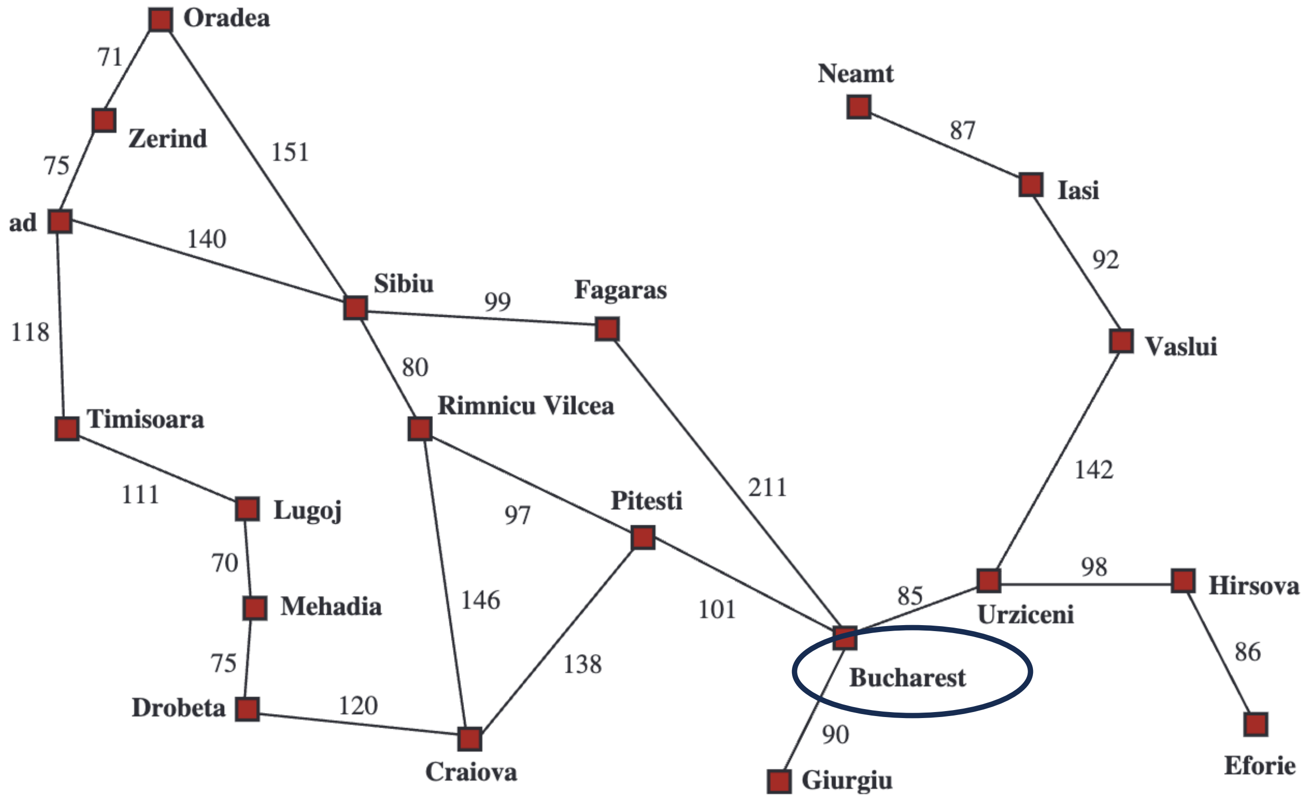


Illustration of greedy search

- Greedy search expands the node that appears to be closest to goal

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

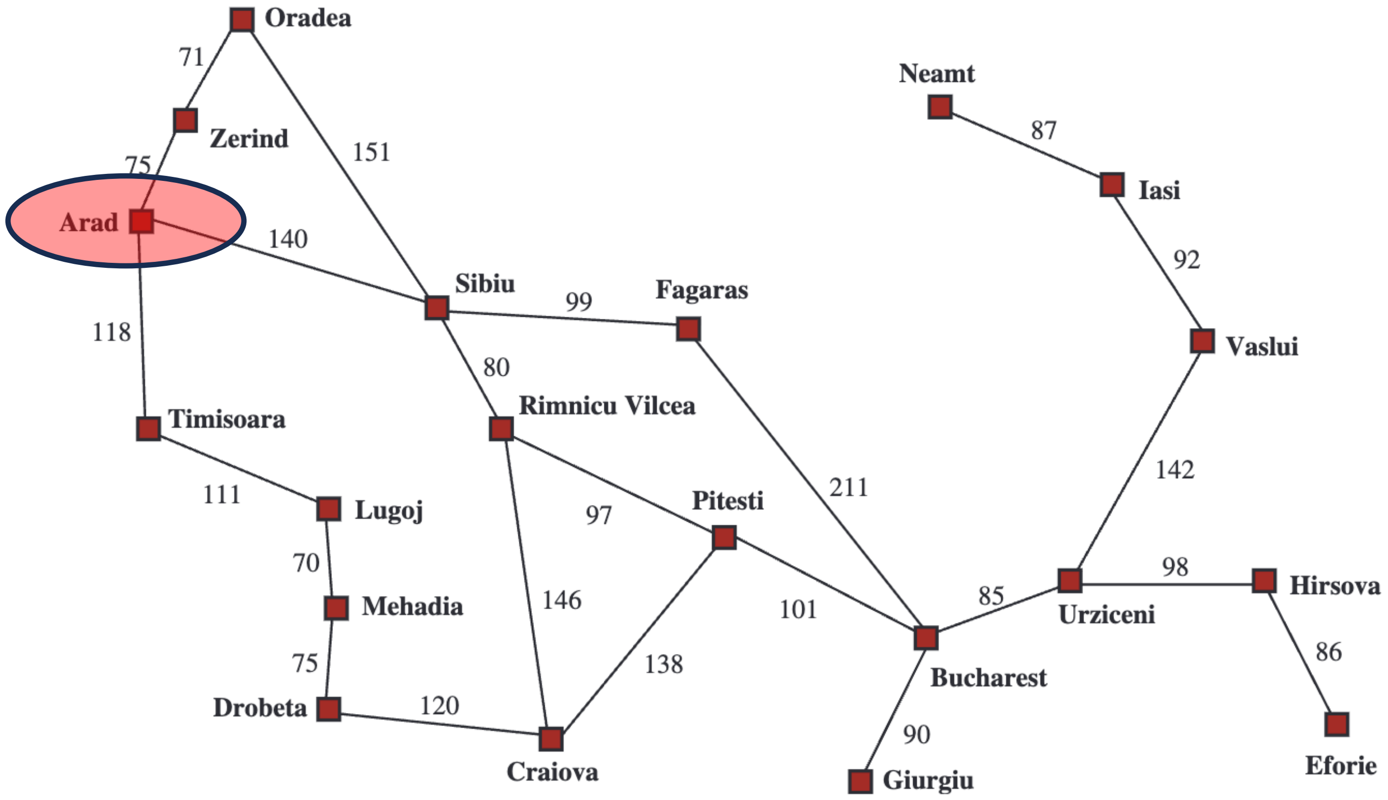
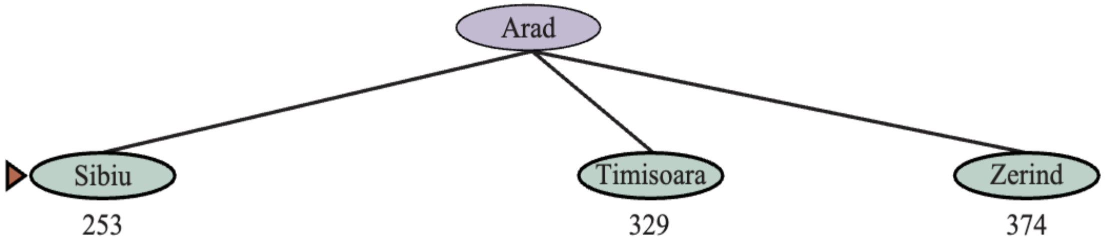


Illustration of greedy search

- Greedy search expands the node that appears to be closest to goal

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



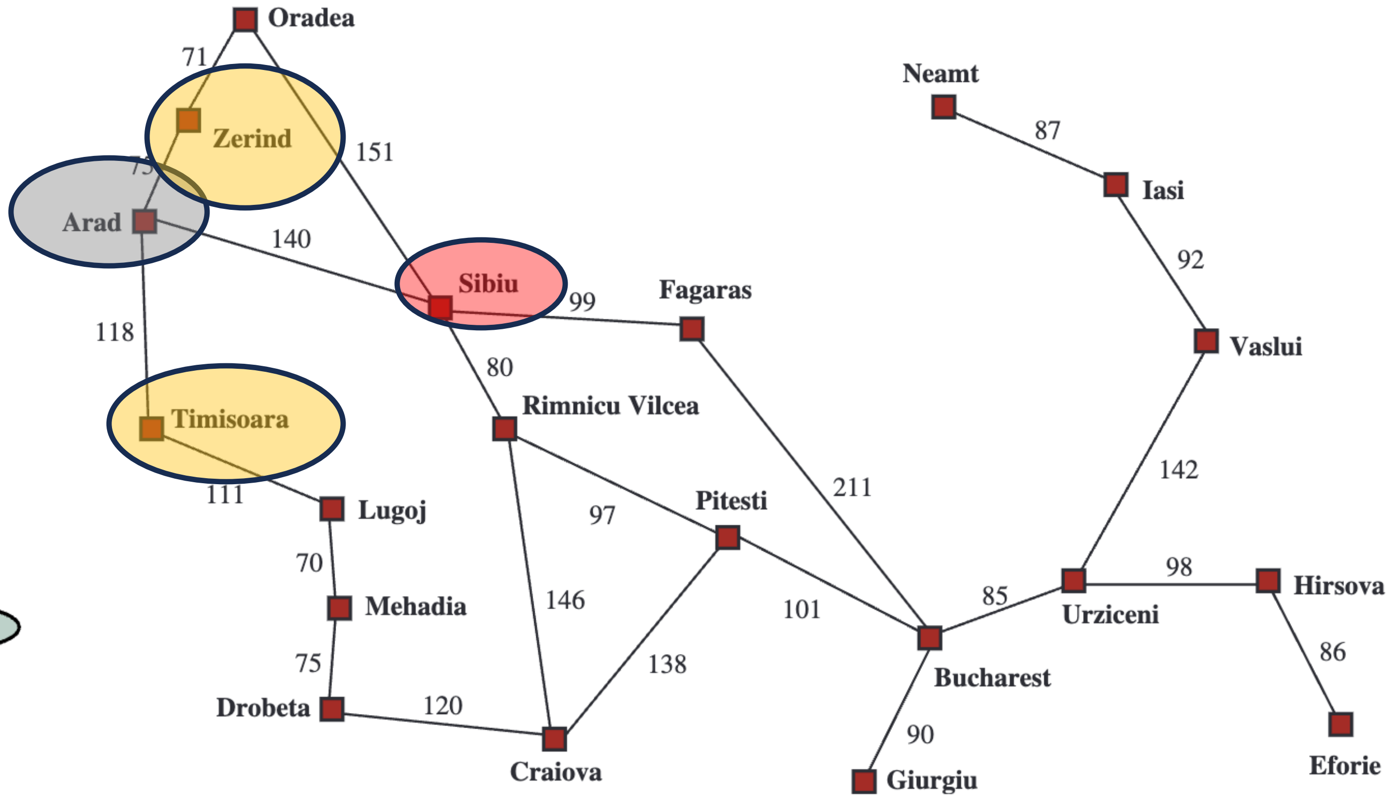
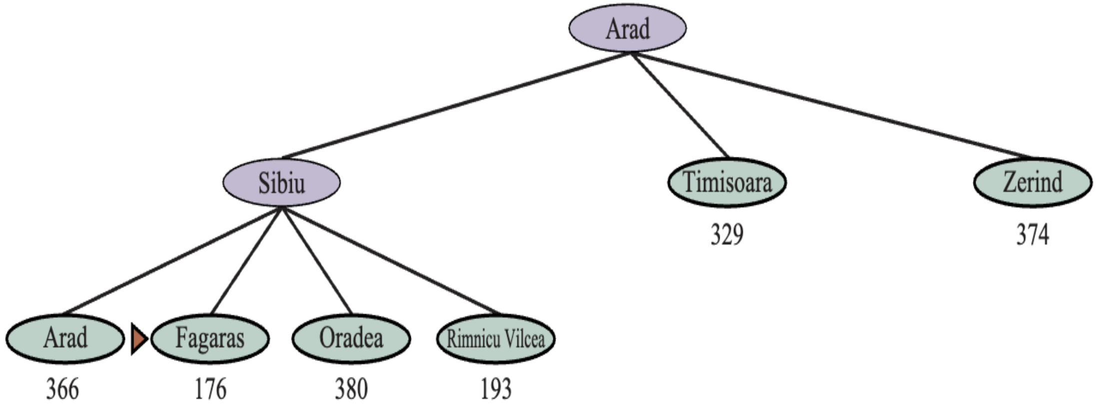


Illustration of greedy search

- Greedy search expands the node that appears to be closest to goal

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



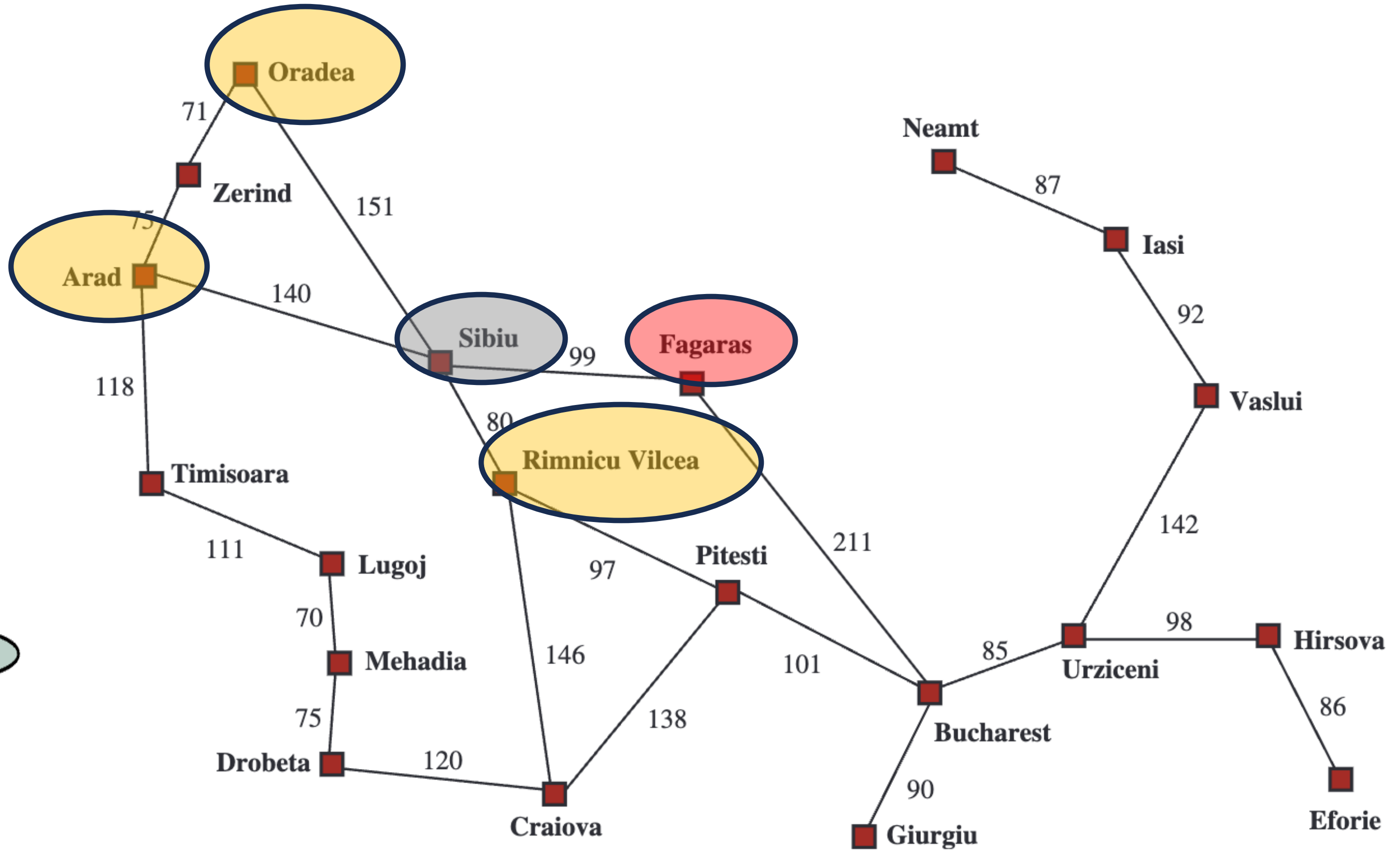
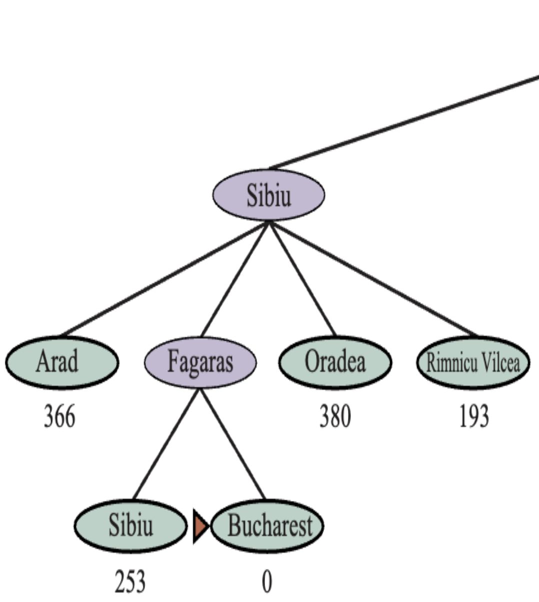
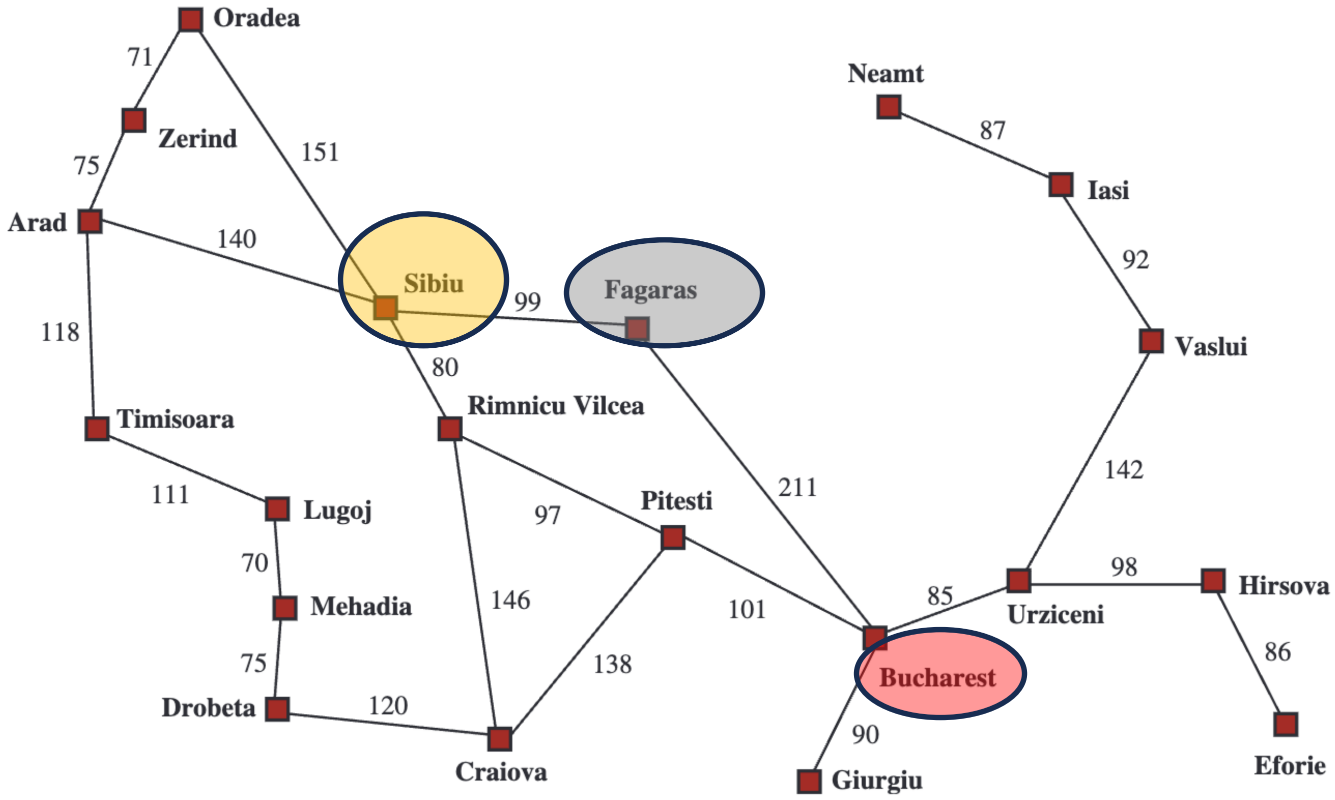


Illustration of greedy search

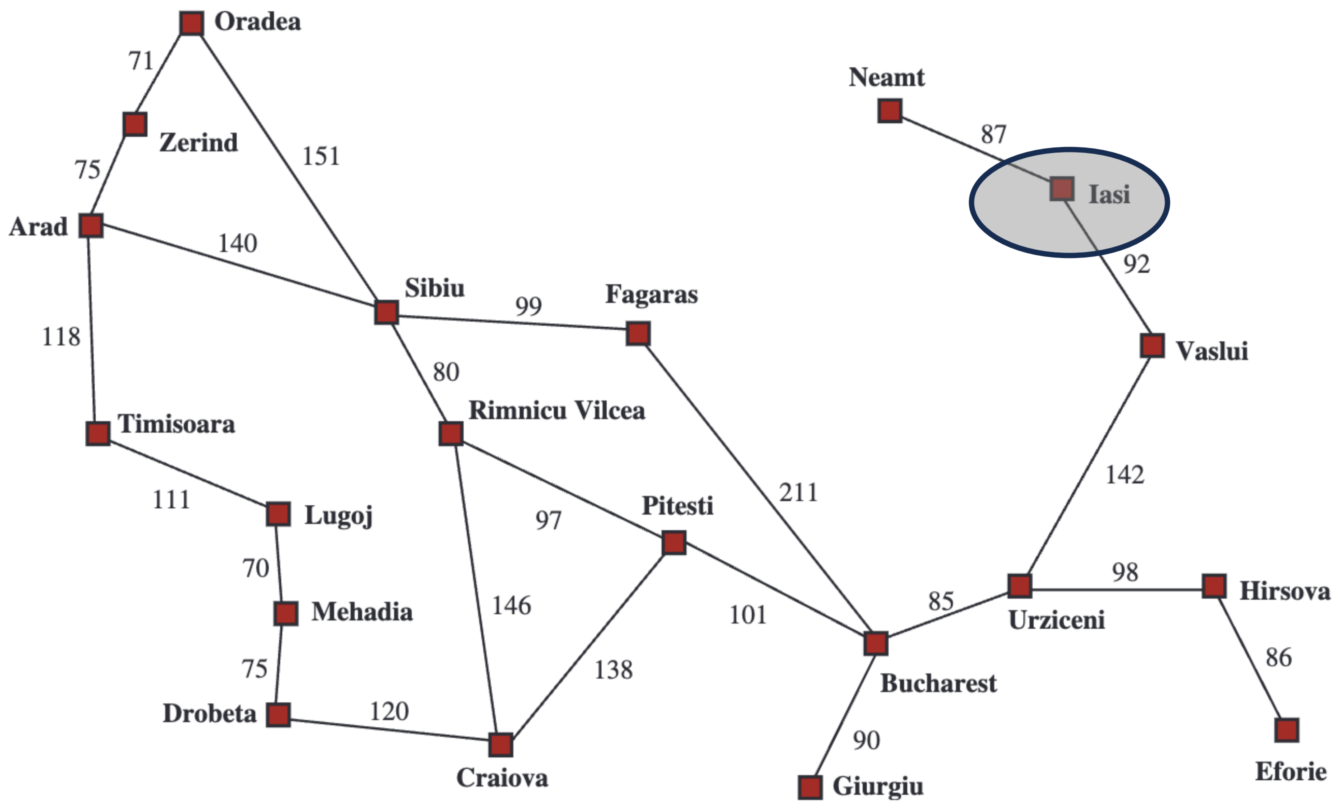
- Greedy search expands the node that appears to be closest to goal

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





Is greedy guaranteed to find a solution?



- No, an get stuck in loops, e.g., with Oradea as goal, Iasi → Neamt → Iasi → Neamt → …

Next: A^* search algorithm

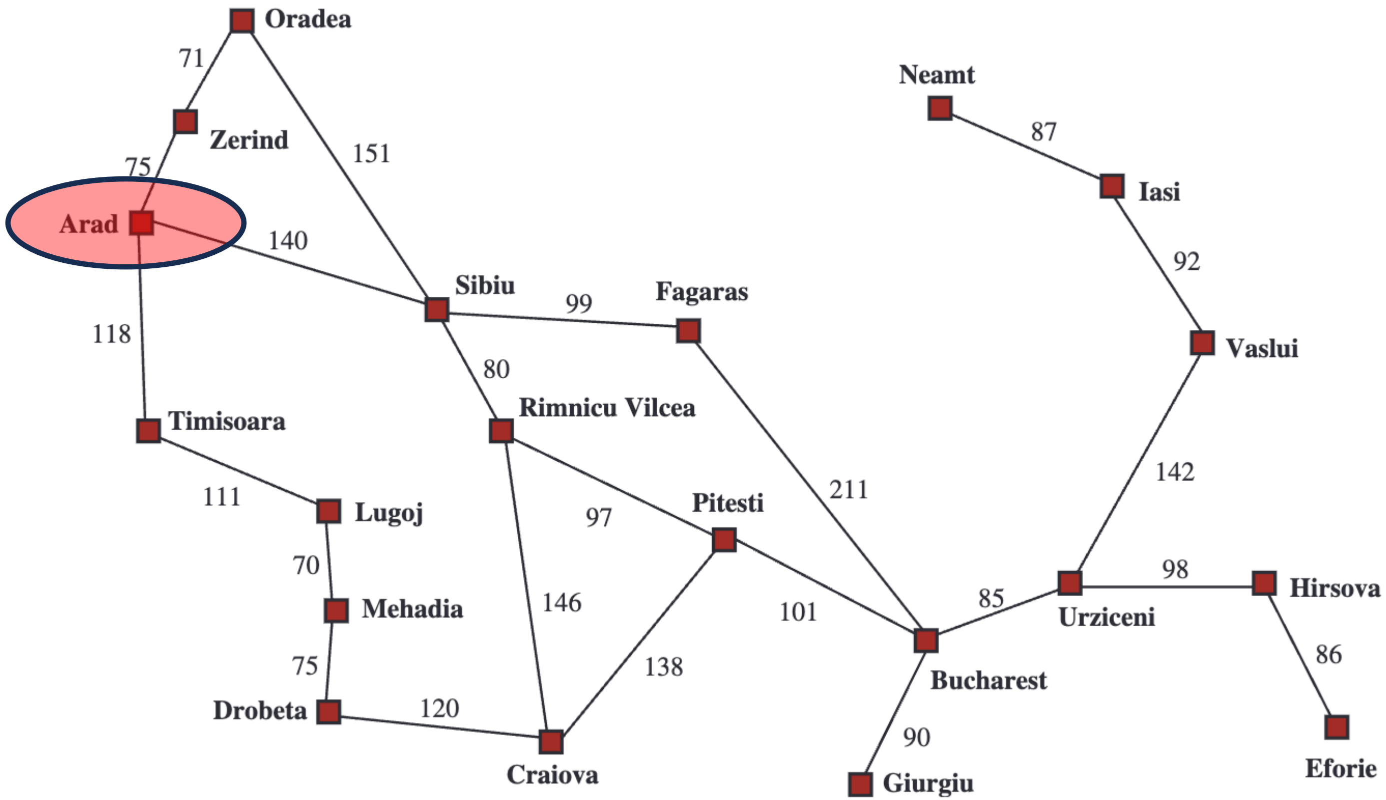
- Idea: avoid expanding paths that are already expensive
- Combine uniform cost search and greedy search: let the cost function $f(n) = g(n) + h(n)$
  - $g(n)$ is the incurred cost at step n, which ensures we account for cost so far
  - $h(n)$ is a heuristic function that estimates how much more cost we would incur to reach the goal

An illustration of A^* search

A table of heuristic functions

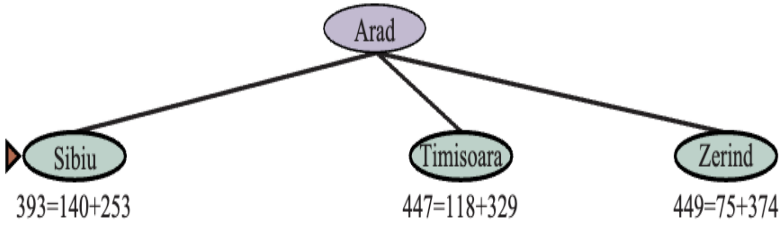
Arad	366	**Mehadia**	241
Bucharest	0	**Neamt**	234
Craiova	160	**Oradea**	380
Drobeta	242	**Pitesti**	100
Eforie	161	**Rimnicu Vilcea**	193
Fagaras	176	**Sibiu**	253
Giurgiu	77	**Timisoara**	329
Hirsova	151	**Urziceni**	80
Iasi	226	**Vaslui**	199
Lugoj	244	**Zerind**	374

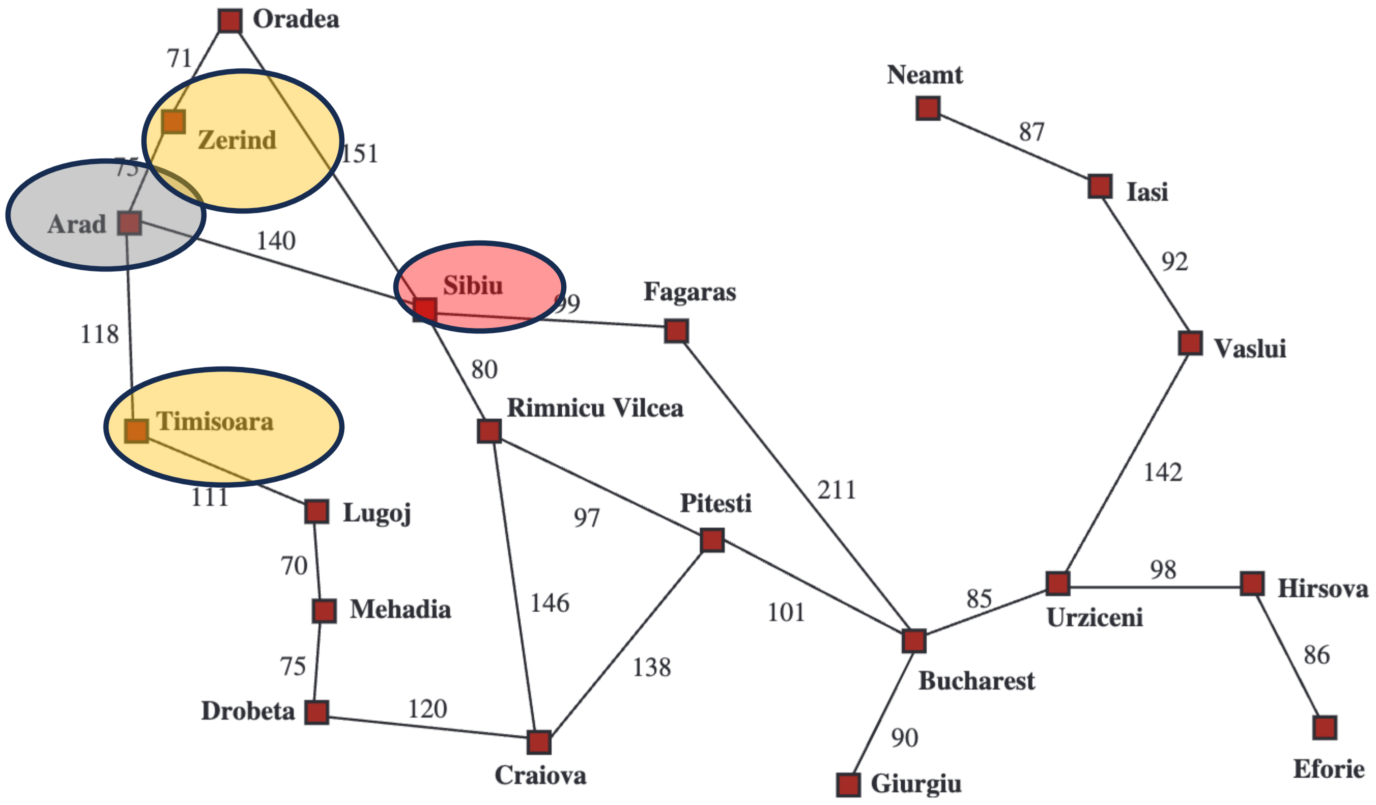
366=0+366



An illustration of A^* search

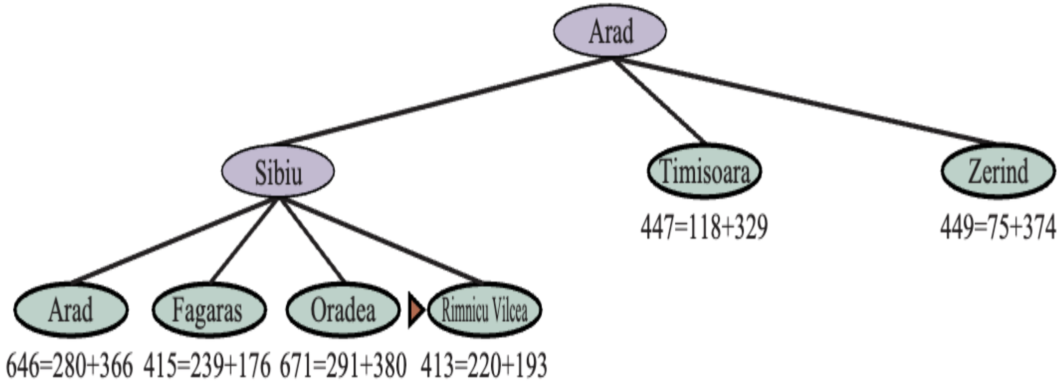
Arad	366	Mehadia	241
Bucharest	0	Neamt	234
Craiova	160	Oradea	380
Drobeta	242	Pitesti	100
Eforie	161	Rimnicu Vilcea	193
Fagaras	176	Sibiu	253
Giurgiu	77	Timisoara	329
Hirsova	151	Urziceni	80
Iasi	226	Vaslui	199
Lugoj	244	Zerind	374

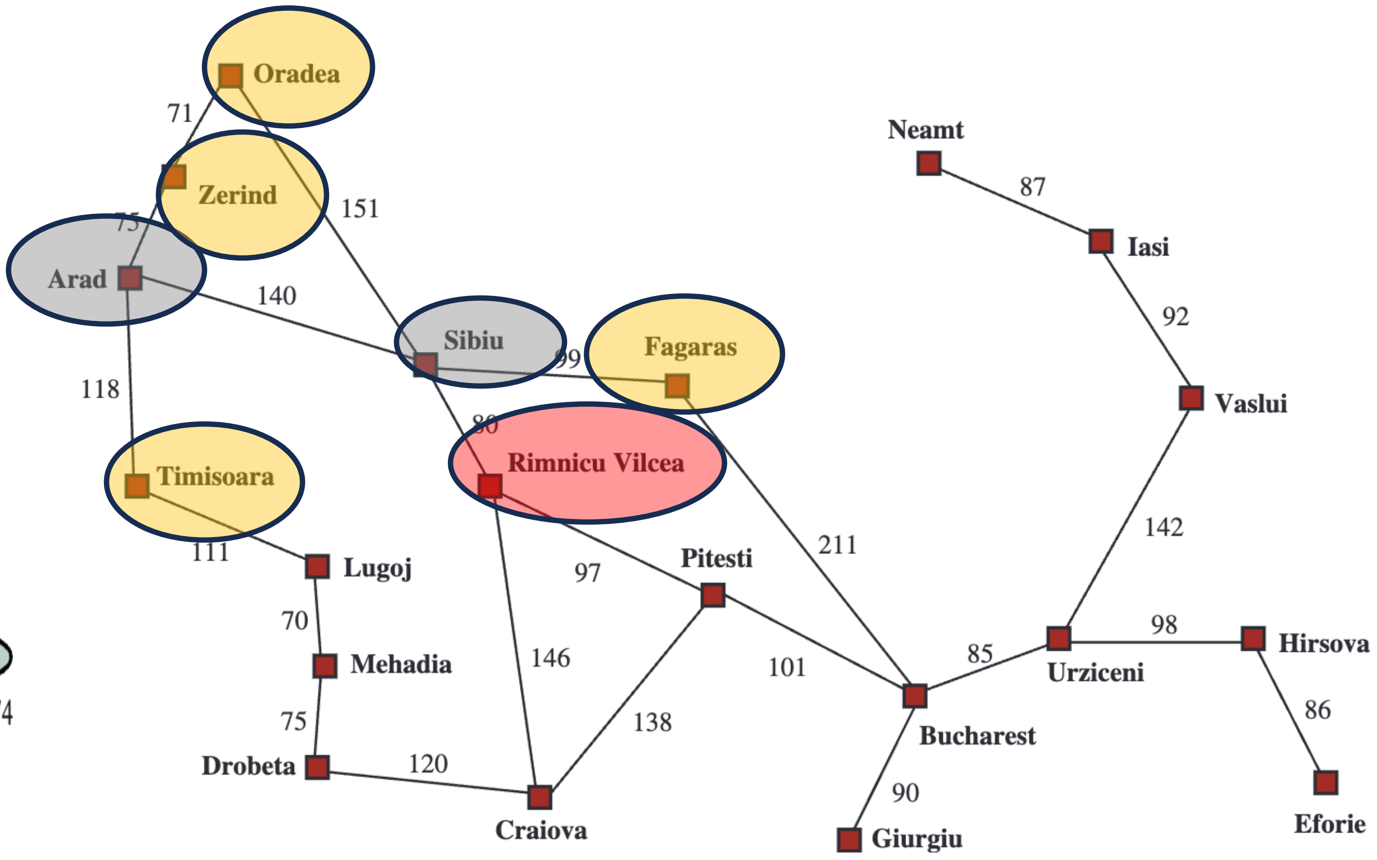




An illustration of A^* search

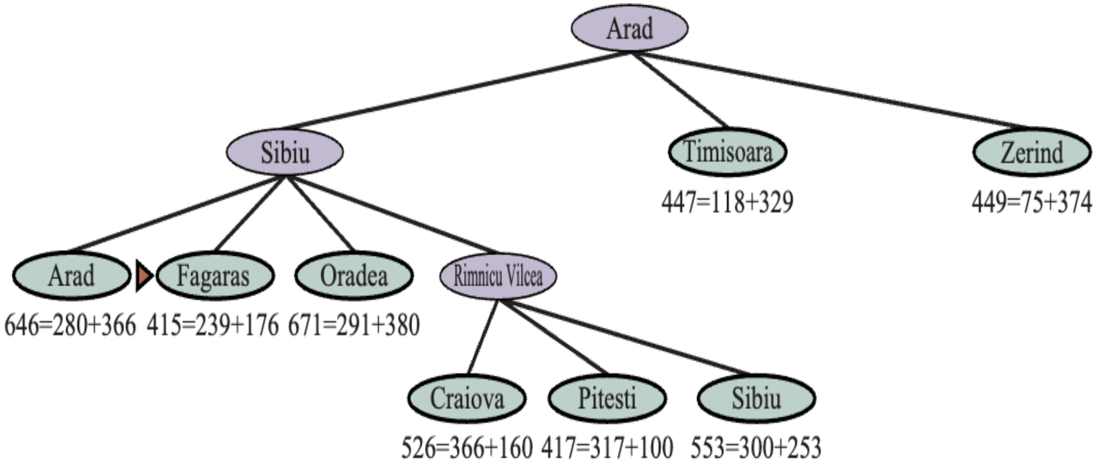
Arad	366	**Mehadia**	241
Bucharest	0	**Neamt**	234
Craiova	160	**Oradea**	380
Drobeta	242	**Pitesti**	100
Eforie	161	**Rimnicu Vilcea**	193
Fagaras	176	**Sibiu**	253
Giurgiu	77	**Timisoara**	329
Hirsova	151	**Urziceni**	80
Iasi	226	**Vaslui**	199
Lugoj	244	**Zerind**	374

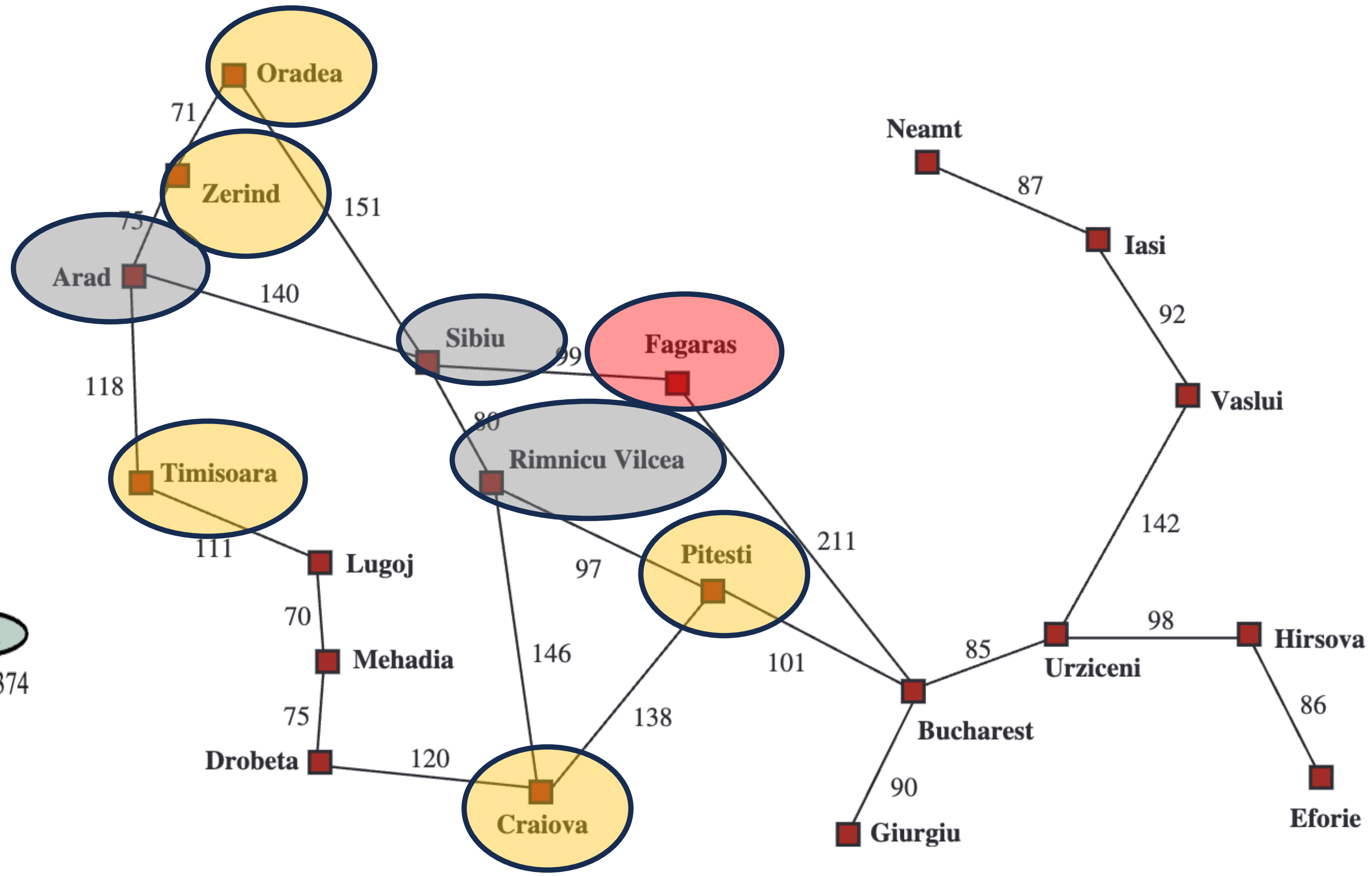




An illustration of A^* search

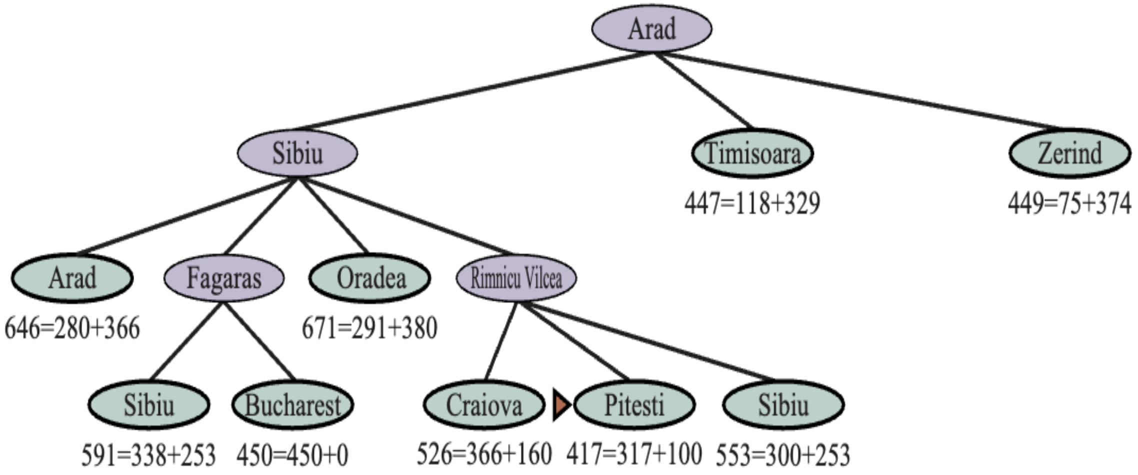
Arad	366	**Mehadia**	241
Bucharest	0	**Neamt**	234
Craiova	160	**Oradea**	380
Drobeta	242	**Pitesti**	100
Eforie	161	**Rimnicu Vilcea**	193
Fagaras	176	**Sibiu**	253
Giurgiu	77	**Timisoara**	329
Hirsova	151	**Urziceni**	80
Iasi	226	**Vaslui**	199
Lugoj	244	**Zerind**	374

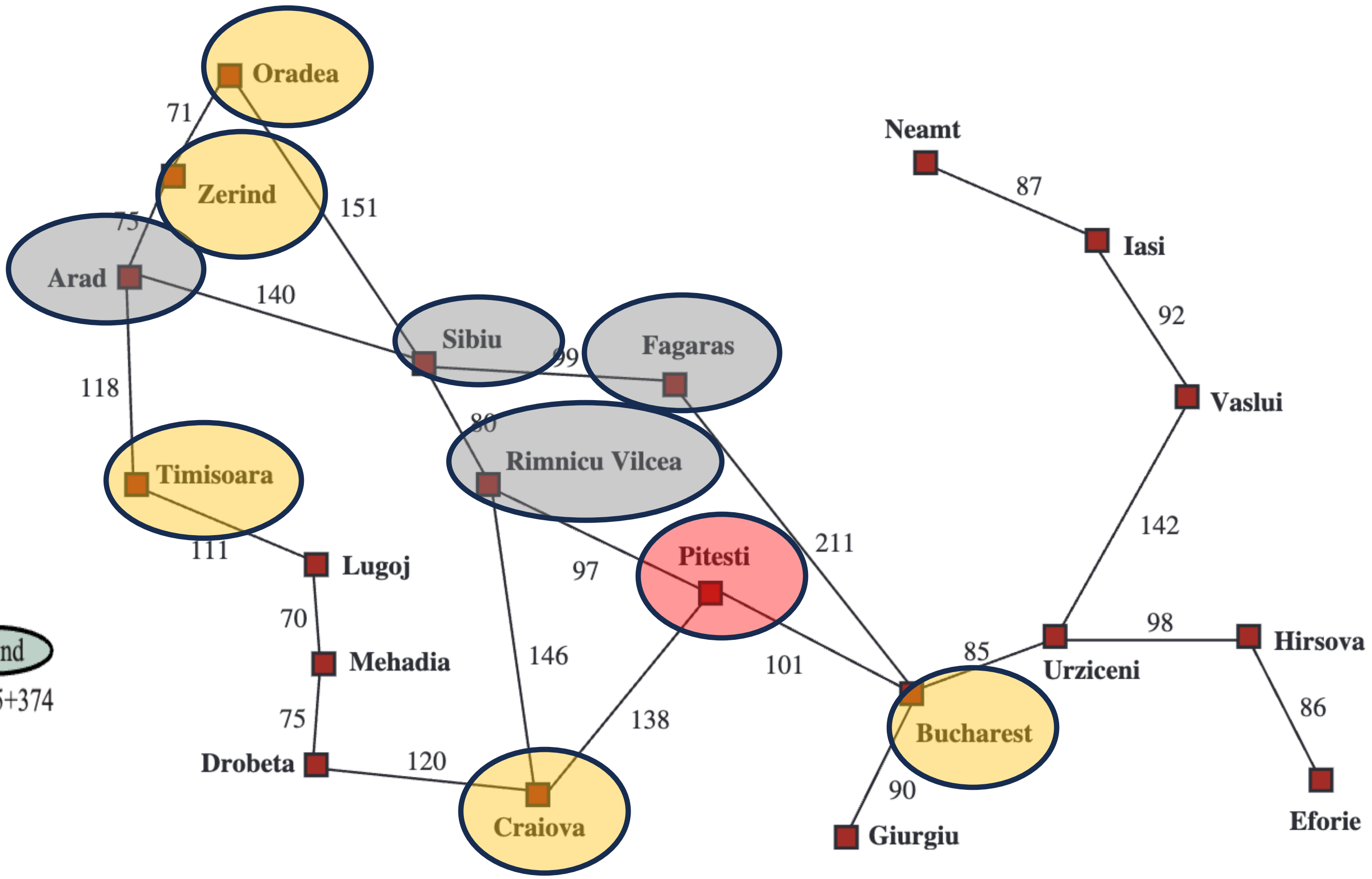




An illustration of A^* search

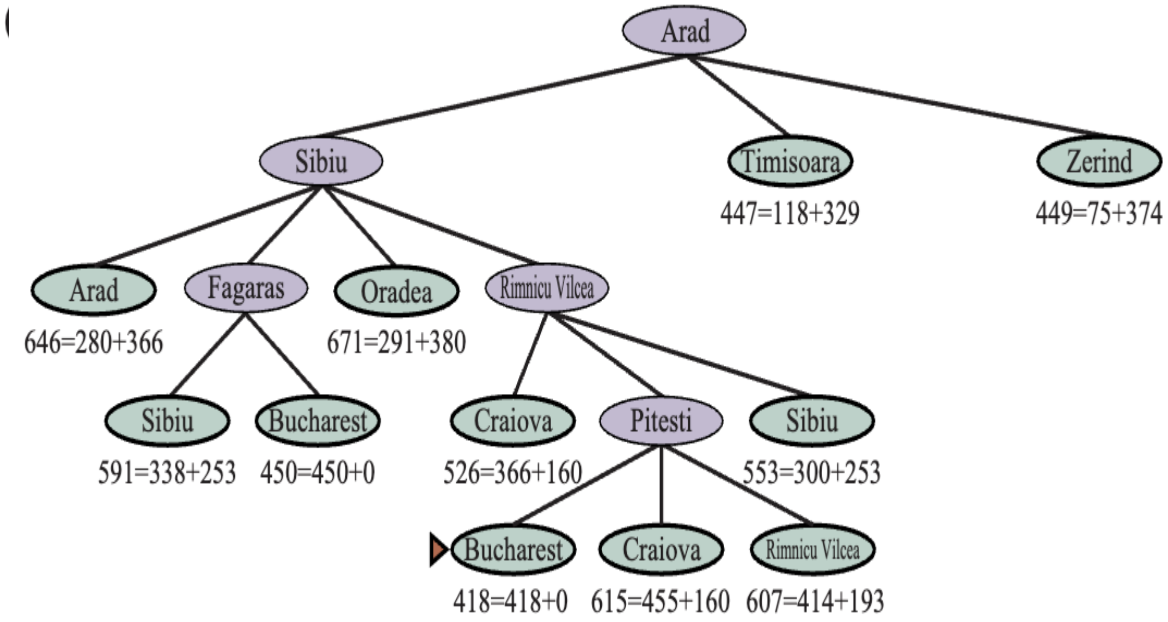
Arad	366	Mehadia	241
Bucharest	0	Neamt	234
Craiova	160	Oradea	380
Drobeta	242	Pitesti	100
Eforie	161	Rimnicu Vilcea	193
Fagaras	176	Sibiu	253
Giurgiu	77	Timisoara	329
Hirsova	151	Urziceni	80
Iasi	226	Vaslui	199
Lugoj	244	Zerind	374

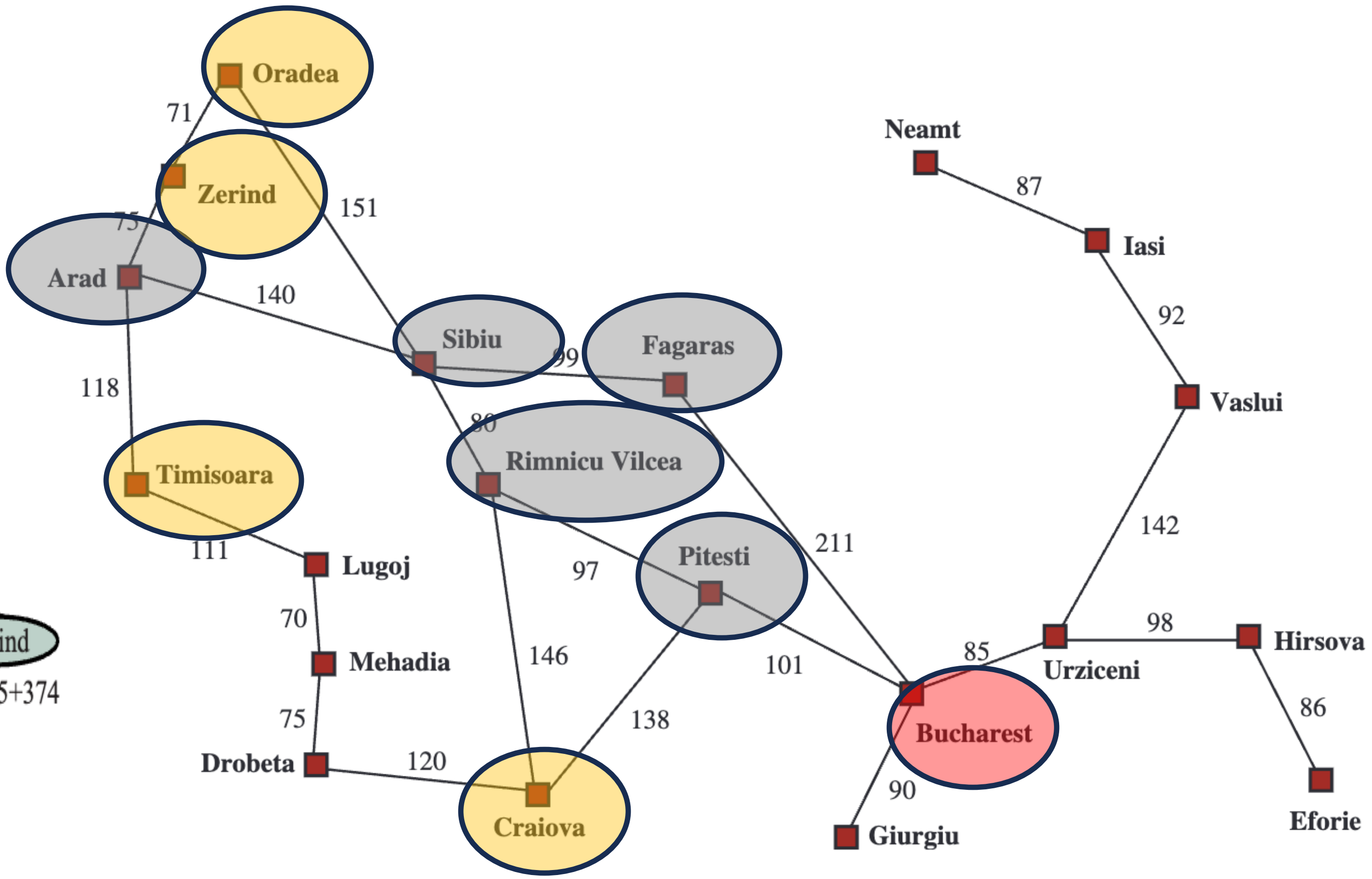




An illustration of A^* search

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 is optimal

- **Property**: The solution found by A^* search is optimal if the heuristic $h(n)$ is admissible, meaning that it is never an overestimate of the cost from node n to a goal node
  - $\forall n, h(n) \leq h^*(n)$, where $h^*(n)$ is the true cost from n
  - Require $h(n) \geq 0$, so $h(G) = 0$ for any goal G

- **Why?** $f(n)$ never decreases along any path due to the above condition on $h(n)$, so first goal found has lowest f = lowest actual cost

Summary

- Greedy search algorithm: Can get stuck in loops, finds a solution quickly but not necessarily best

- A^* search algorithm: guaranteed to find a solution, and ensure optimality
  - Utilizes a heuristic function to help estimate the cost towards the target

Variations: Satisficing search

- Sometimes we need a solution quickly, even if not optimal
  - **Video games:** prioritize response time, such as needing to react in milliseconds
  - **Robotics:** avoid obstacles on the road spontaneously
  - **Web services:** quickly answer a query without extended wait time

- Examples
  - Deliberately overestimate to speed up search
  - Weighted A^*: inflate heuristic influence systematically

- Beam search: Memory-bounded search: Limits the size of the frontier
  - The easiest approach is to keep only the k nodes with the best f-scores, discarding any other expanded nodes

Extensions: Weighted A^* search

- $f(n) = g(n) + w \times h(n)$
  - $\boldsymbol{w = 1}$**:** standard A^* (optimal)
  - $\boldsymbol{w > 1}$**:** weighted A^* (faster, sub-optimal)
  - $\boldsymbol{w \to \infty}$**:** approaches pure greedy search

- **Bounded sub-optimality:** weighted A^* with admissible h finds solution with cost $\leq w \times$ optimal
  - Example: $w = 1.5$ guarantees solution within 50% of optimal

Conclusion

- Informed (heuristic) search algorithms
  - Greedy (best-first) search
  - A^* search
  - Variations: beam search, weighted A^* search
- Search algorithms are extremely useful in motion planning and path finding problems