Lecture 4 - Informed Search Strategies

We will use some "extra information" in the form of domain-specific hints about the location of the goal state. The hints come in the form of a heuristic function, $h(n)$. $h(n) =$ estimated cost from the state at node n to the goal state. Let's start.

Greedy best-first search:

- Expand nodes closest to the goal state.
- Implementation: Use a priority queue that prioritizes nodes based on the heuristic. At any point, all the leaf nodes are in the frontier.

We only consider the heuristic value $h(n)$. We do not look at the actual value. Let's evaluate this search algorithm.

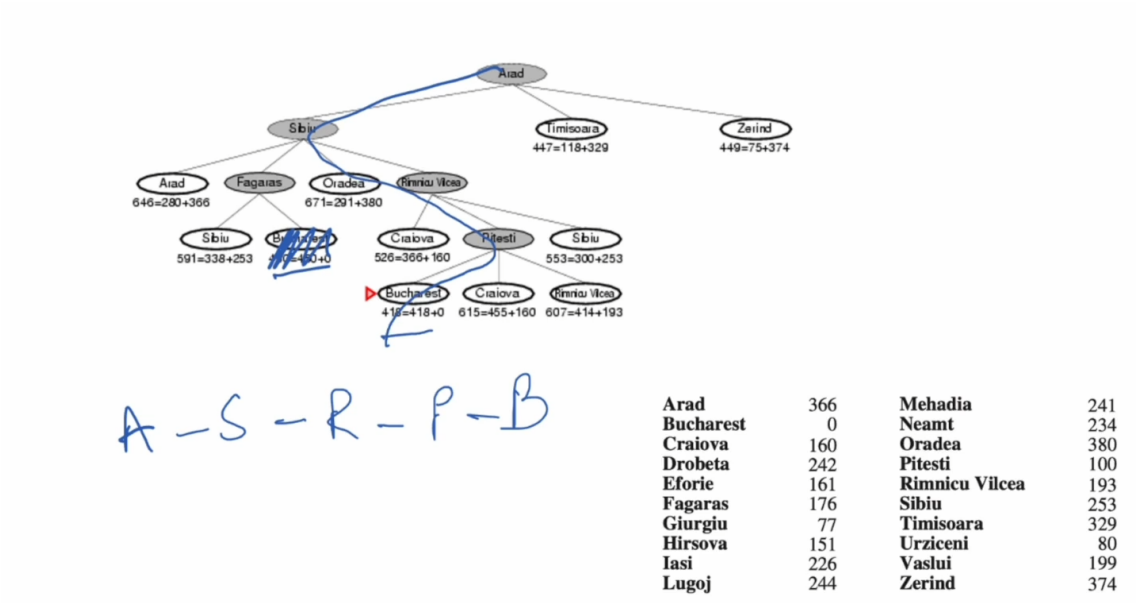
- Completeness? If it's a tree search, then no. If it's a graph search, then it could be complete, but not necessarily. Infinite search space is also an issue. It is only complete if we are doing graph search in a finite state space.
- Optimality? No, it does not return the optimal path.
- Time complexity? $O(b^m)$
- Space complexity? $O(b^m)$

A* algorithm:

We kind of combine both the greedy best-first search and the uniform cost search. It would be:

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

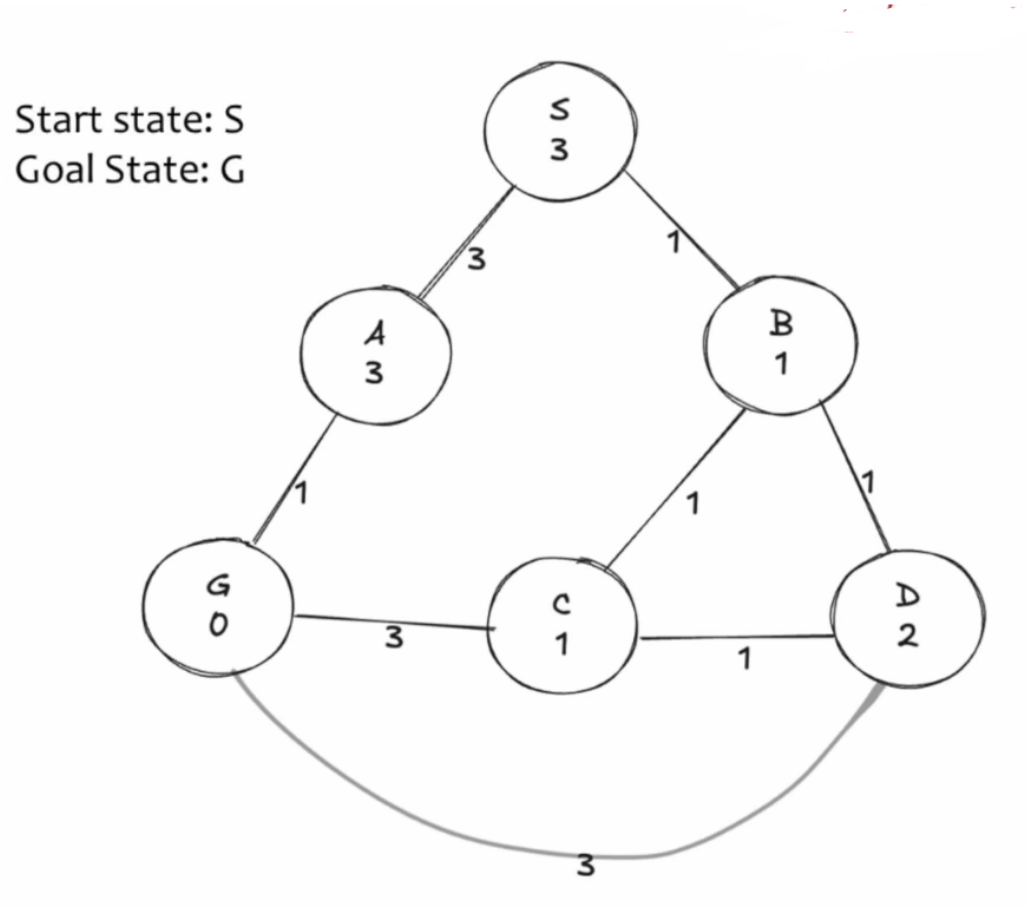
Where $g(n)$ is the actual cost so far to reach the node n. This is best demonstrated through an example.



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

As you can see, this returns the optimal path in this case. Some examples have been done on paper and are included in the GitHub. Would A* always be optimal? No. It is only optimal if the heuristics satisfy some conditions.

A heuristic is **admissible** if $h(n) \leqslant h^*(n)$, where $h^*(n)$ is the true cost to reach the goal state from n. Check this example (done on paper):



$$h(s) = 3 \leqslant h^*(s) = 4$$
$$h(a) = 3 \nleqslant h^*(a) = 1$$

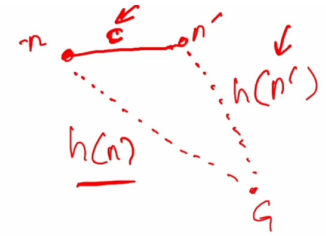
Therefore, the heuristic used here is not admissible. Therefore, the path that we found is not optimal. An admissible heuristic never overestimates the cost to reach the goal.

Theorem: If $h(n)$ is admissible, then A* using tree search is optimal. What if we are instead using graph search? Let's look at consistency first.

A heuristic is **consistent** if, for every node n, every successor n' of n generated by any action, a:

$$h(n) \leqslant c(n, a, n') + h(n')$$

This is a form of the triangle inequality.



If a heuristic is consistent, then it is also admissible.

Theorem: If $h(n)$ is consistent, then A* using graph search is optimal.

Let's look at some of the properties of A*.

- Completeness? Yes
- Time complexity? It is better than BFS / DFS / UFS, but it is still exponential. It uses heuristics to guide the search, exploring fewer states and reaching the goal more efficiently.
- Space complexity? Same as BFS / UCS
- Optimal? Yes, if the heuristic is admissible for tree search, and consistent for graph search.

How do we come up with heuristic functions? Let's consider the case of the 8-puzzle. If we were to instead have a 15-puzzle, then we would have 10 trillion states to explore from. Therefore, a heuristic function would be necessary. We could count the number of misplaced tiles. This is admissible, because the number of actions we need to take is definitely larger than the heuristic $h(n)$. Can we do better? We want to get a heuristic that is closer to the true cost. Let's sum all of the best-case actions that we need to take to get the tiles in the right place. This is better, because we are at least somewhat closer to the true cost. It is the same as the Manhattan distance.