CSC520 - Artificial Intelligence

Lecture 5

Dr. Scott N. Gerard

North Carolina State University

Jan 21, 2025

Task Environment Properties Recap

- Fully vs. Partially Observable
- Deterministic vs. Stochastic
- Episodic vs. Sequential
- Static vs. Dynamic
- Discrete vs. Continuous
- Known vs. Unknown
- Single vs. Multi-agent

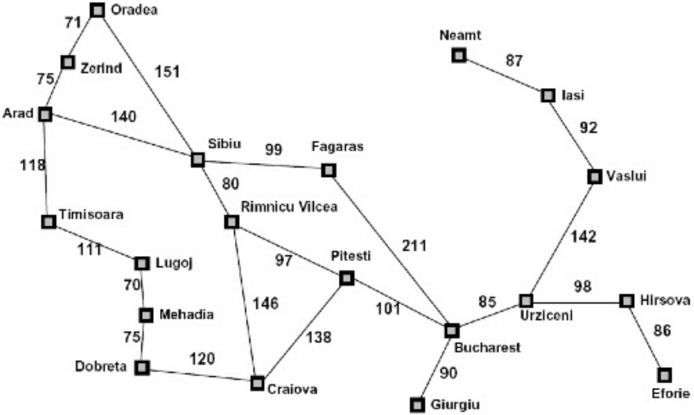
Agenda

- Informed Search
  - Greedy Best-First Search
  - A^* Search

Search Heuristics

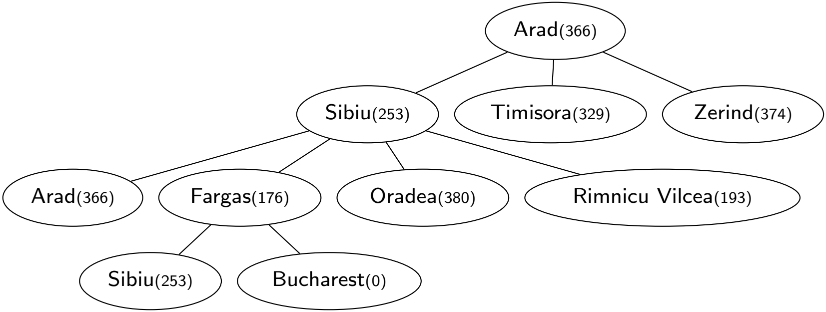
- Informed search employs an estimate of how far a goal is from a state
- A heuristic function that estimates the path cost from a state to a goal state: $h(n)$
  - Euclidean distance or Manhattan distance between points
  - Number of misplaced tiles in an 8-puzzle
- Heuristic function is specific to a search problem

Romania Straight-line Distances



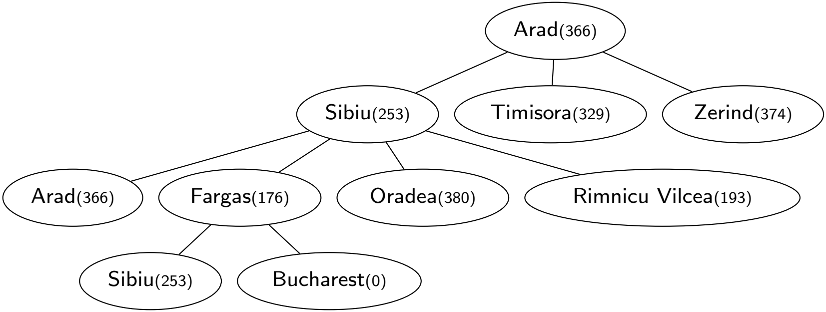
Straight−line distance to Bucharest	
Arad	366
Bucharest	0
Craiova	160
Dobreta	242
Eforie	161
Fagaras	178
Giurgiu	77
Hirsova	151
Iasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitesti	98
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374

Greedy Search



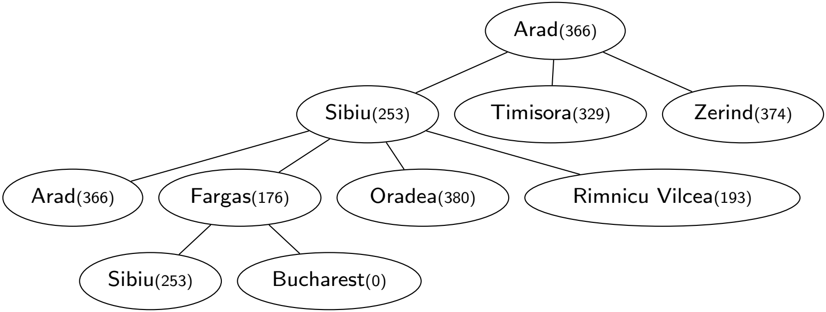
- Expand a node with lowest $h(n)$

Greedy Search



- Expand a node with lowest $h(n)$
- Cheaper path from Arad→Sibiu→Rimnicu Vilcea→Pitesti→Bucharest

Greedy Search

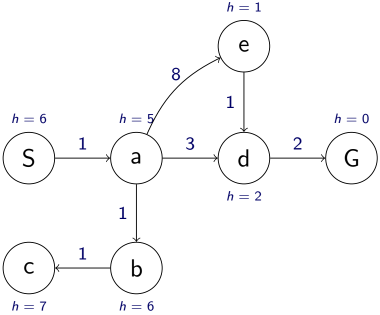


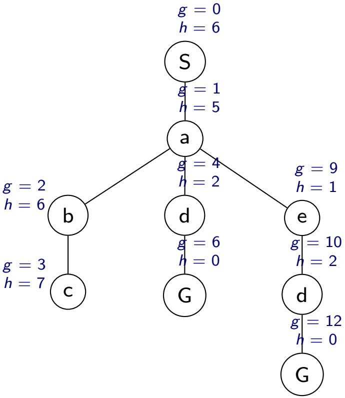
- Expand a node with lowest $h(n)$
- Cheaper path from Arad→Sibiu→Rimnicu Vilcea→Pitesti→Bucharest
- Completeness: Not complete in infinite state space
- Cost optimality: Not optimal

A* Search

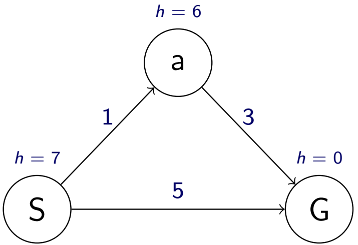
- Uniform cost search expands a node with least cost from start node to current node: $g(n)$
- Greedy search expands a node with estimated cost from current node to a goal node: $h(n)$
- A* search expands a node with least sum of costs: $f(n) = g(n) + h(n)$

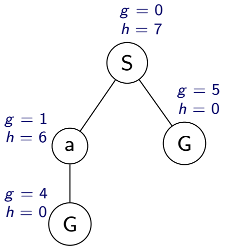
A^* Search Example





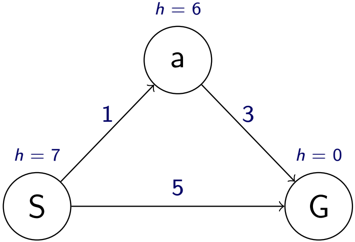
Admissibility Property





A* solution is not optimal here. Why?

Admissibility Property



A* solution is not optimal here. Why?

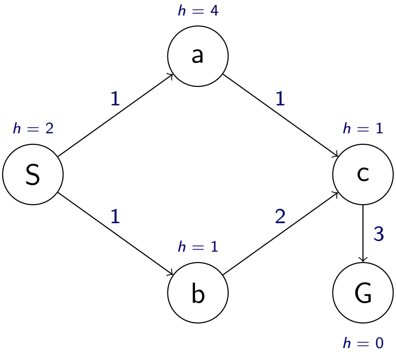
- For optimality, estimates need to be less than or equal to actual costs
- A heuristic that never overestimates the cost is *admissible* heuristic (optimistic)

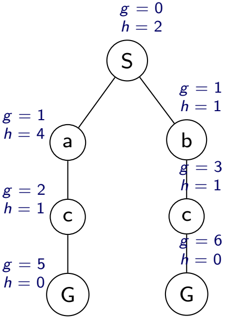
Formally, a heuristic h is *admissible* if $0 \leq h(n) \leq h^*(n)$ for each node n where $h^*(n)$ is the true optimal cost to a goal

Graph Search

- Idea is to not expand a state twice
- Implementation like tree search but with a set (usually termed as a *reached*/*closed* set) in which expanded nodes are added
- Before expanding a node, check if it in the set
  - If reached, don't expand again
  - If not reached, add it to the *reached* set

Consistency Property





$h(a) - h(c) \nleq cost(a, c)$

- Formally, a heuristic is *consistent* if for every node n and every successor n' of n generated by an action a, we have: $h(n) \leq c(n, a, n') + h(n')$.
- Every consistent heuristic is admissible, but not vice versa.

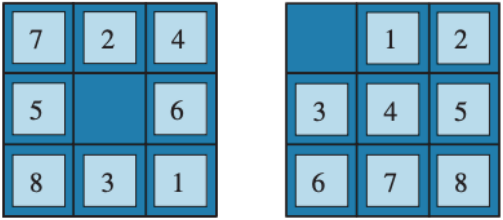
A^* Optimality

- Tree search
  - A^* is optimal if heuristic is *admissible*
  - Admissible if $0 \leq h(n) \leq h^*(n)$ for each node n where $h^*(n)$ is the true optimal cost to a goal
- Graph search
  - A^* is optimal if heuristic is *admissible*
  - If heuristic is *consistent*, never have to check/update reached states
  - Consistent if for every node n and every successor n' of n generated by an action a, we have: $h(n) \leq c(n, a, n') + h(n')$
- Consistency implies admissibility

Creating Admissible Heuristics

- Designing admissible heuristics is the biggest part of solving a search problem using A*
- Methods for generating heuristics
  - *Relaxed* problems
    - A problem with fewer restrictions on the actions is a *relaxed problem*
  - Pattern Databases
  - Machine learning

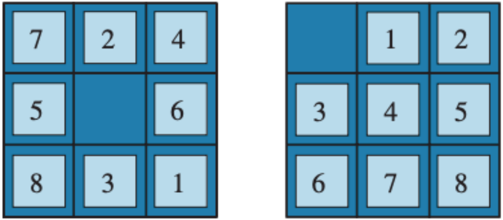
Heuristics For 8-Puzzle



Start State   Goal State

- h_1 = Number of misplaced tiles
- What is h_1 for the start state?
- Admissible since at least one move is required to get an out-of-place tile into its place

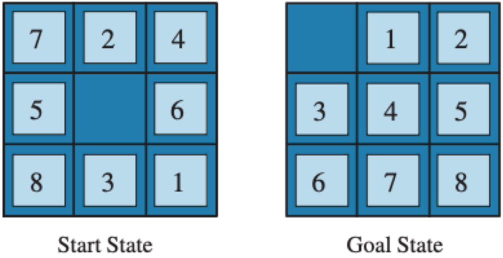
Heuristics For 8-Puzzle



Start State Goal State

- h_2 = Total Manhattan distance to get tiles in their correct place
- What is h_2 for the start state?
  - $h_2 = 3 + 1 + 2 + 2 + 2 + 3 + 3 + 2 = 18$
- Admissible since at least moves equal to the Manhattan distance is required to get an out-of-place tile in its place

Heuristics For 8-Puzzle



- $h_2 =$ Total Manhattan distance to get tiles in their correct place
- What is h_2 for the start state?
  - $h_2 = 3 + 1 + 2 + 2 + 2 + 3 + 3 + 2 = 18$
- Admissible since at least moves equal to the Manhattan distance is required to get an out-of-place tile in its place
- Note that $\forall n,\ h_2(n) > h_1(n)$; h_2 dominates h_1
- More dominant heuristic causes A* to expand fewer nodes thus increases efficiency

Heuristics from Machine Learning

- Learn heuristics from experience
- For example, generate 100 random instances of 8-puzzle configurations and compute their optimal solution cost
- Features can be: $x_1(n) =$ number of misplaced tiles, $x_2(n) =$ number of pairs of tiles that are not adjacent in the goal state, etc.
- Learn a regression model for the heuristic: $h(n) = c_1 x_1(n) + c_2 x_2(n)$
- Heuristic may be inadmissible

Memory Bounded Search

- Main issue with A^* is the amount of required memory
- Algorithms that use less memory
  - Beam search
  - Iterative-deepening A^* search (IDA*)
  - Recursive best-first search (RBFS)
  - Simplified memory-bounded A^* search (SMA)

Beam search

- Limits the size of the frontier set
- Keep only the k nodes with the best f scores
- Keep only the nodes with the $f(best) \leq f(node) \leq f(best) + \delta$

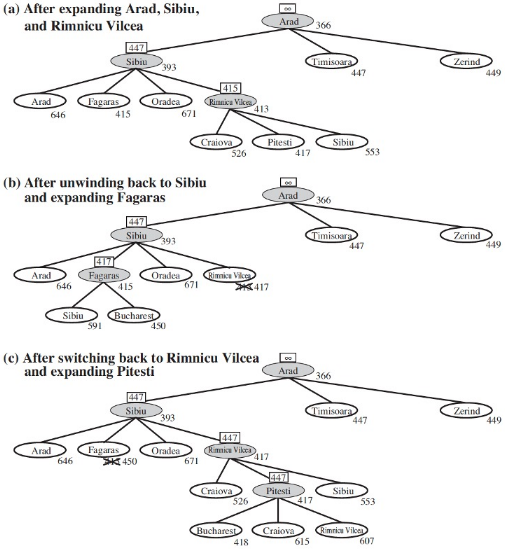
Iterative-deepening A* search

- Similar to iterative-deepening DFS
  - Instead of depth, the bound is in terms of f-*value*
- Initially, the f-*limit* is equal to f-*value*(*start*)
  - Repeat until a goal is found
  - Expand nodes using A* while f-*value*(*node*) $\leq$ f-*limit*
  - Increase f-*limit* to the smallest f-*value* of unexpanded node
  - A* search within slightly larger f-*limit* contour

Recursive Best-First Search

- Linear-space heuristic search algorithm
- Uses a dynamic *f-limit* to keep track of the *f-value* of the best alternative path from any ancestor
- If the current node exceeds *f-limit*, the recursion unwinds to the best alternative and replaces the *f-value* of each node along the path with a *backed_up* value

RBFS Example



Simplified Memory-bounded A*

- IDA* and RBFS do not use all of the available memory
- SMA* proceeds like A* until memory is full
- If memory is full, drops a node with highest *f-value*
- Backs up the value of the forgotten node to its parent

Class Exercise

- Find the solutions generated by UCS, Greedy Search and A* algorithms on the following problem.