CS 161 Fundamentals of Artificial Intelligence
Lecture 4

Informed Search Algorithms

Quanquan Gu

Department of Computer Science
UCLA

Jan 18, 2022

Outline

- Best-first search
- A^* search
- Heuristics

Review: Tree search

```
function TREE-SEARCH( problem, fringe) returns a solution, or failure
    fringe ← INSERT(MAKE-NODE(INITIAL-STATE[problem]), fringe)
    loop do
        if fringe is empty then return failure
        node ← REMOVE-FRONT(fringe)
        if GOAL-TEST[problem] applied to STATE(node) succeeds return node
        fringe ← INSERTALL(EXPAND(node, problem), fringe)
```

A strategy is defined by picking the **order of node expansion**

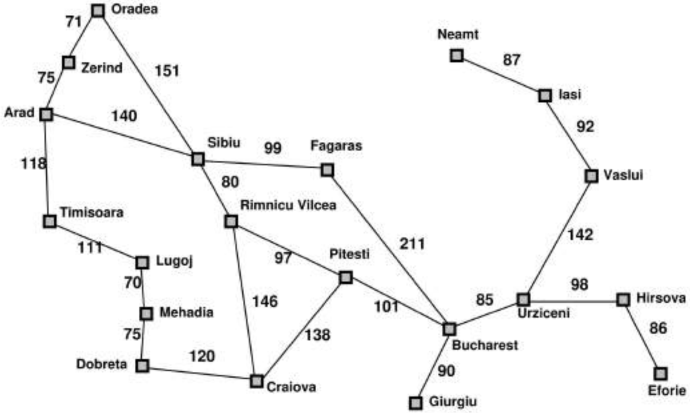
Best-first search

Idea: use an **evaluation function** for each node
– estimate of "desirability"
$\Rightarrow$ Expand most desirable unexpanded node

Implementation:
fringe is a queue sorted in decreasing order of desirability

Special cases:
greedy search
A^* search

Romania with step costs in km



Straight–line distance to Bucharest

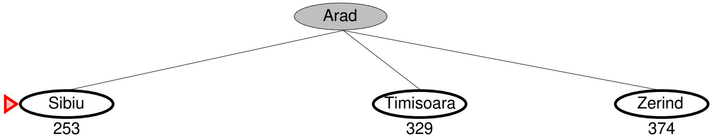
Arad	366
Bucharest	0
Craiova	160
Dobreta	242
Eforie	161
Fagaras	176
Giurgiu	77
Hirsova	151
Iasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitesti	98
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374

Greedy search

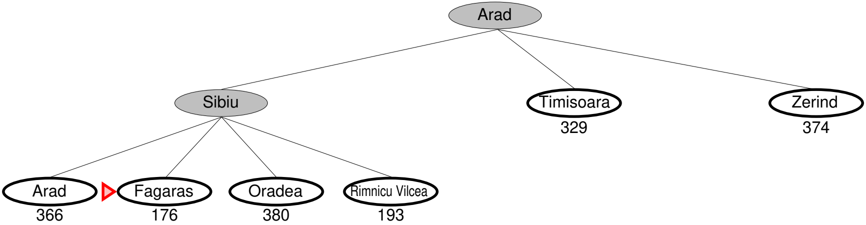
Evaluation function $h(n)$ (**h**euristic)
$\quad = $ estimate of cost from n to the closest goal
E.g., $h_{\mathrm{SLD}}(n) =$ straight-line distance from node n to Bucharest
Greedy search expands the node that **appears** to be closest to goal

Greedy search example

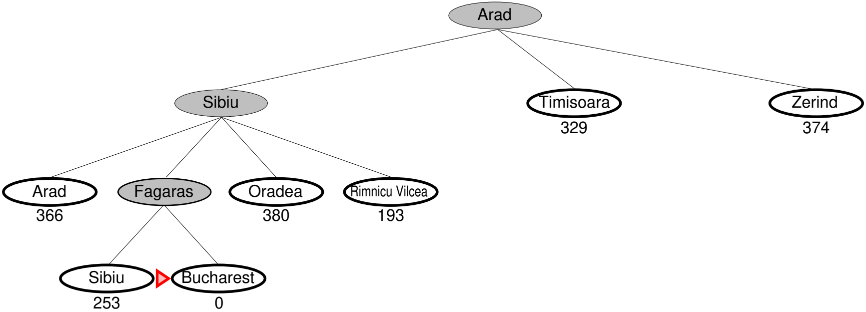
Greedy search example



Greedy search example



Greedy search example



Properties of greedy search

Complete?? No–can get stuck in loops
Complete in finite space with repeated-state checking

Properties of greedy search

Complete?? No–can get stuck in loops
Complete in finite space with repeated-state checking
Time?? $O(b^m)$, m is the max depth, but a good heuristic can give dramatic improvement

Properties of greedy search

Complete?? No–can get stuck in loops
Complete in finite space with repeated-state checking
Time?? $O(b^m)$, m is the max depth, but a good heuristic can give dramatic improvement
Space?? $O(b^m)$—keeps all nodes in memory

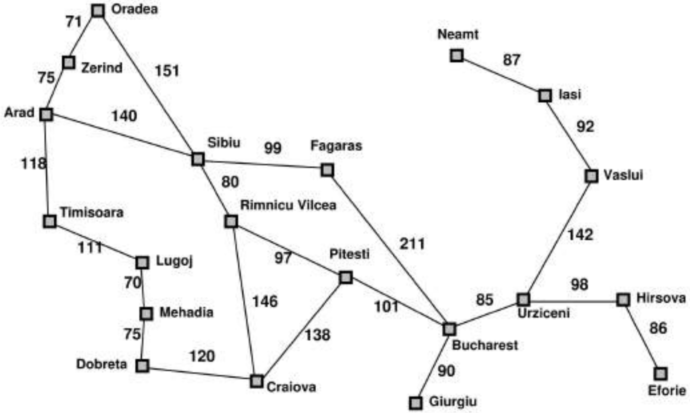
Properties of greedy search

Complete?? No–can get stuck in loops
  Complete in finite space with repeated-state checking
Time?? $O(b^m)$, m is the max depth, but a good heuristic can give dramatic improvement
Space?? $O(b^m)$—keeps all nodes in memory
Optimal?? No

A* search

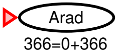
Idea: avoid expanding paths that are already expensive
Evaluation function $f(n) = g(n) + h(n)$
$g(n) =$ cost so far to reach node n
$h(n) =$ estimated cost to goal from n
$f(n) =$ estimated total cost of path through n to goal
A* search uses an **admissible** heuristic
i.e., $h(n) \leq h^*(n)$ where $h^*(n)$ is the **true** cost from n.
(Also require $h(n) \geq 0$, so $h(G) = 0$ for any goal G.)
E.g., $h_{\mathrm{SLD}}(n)$ never overestimates the actual road distance
Theorem: A* search is optimal

Romania with step costs in km

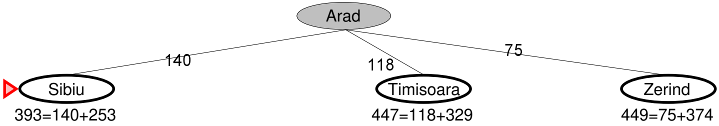


Straight−line distance to Bucharest	
Arad	366
Bucharest	0
Craiova	160
Dobreta	242
Eforie	161
Fagaras	176
Giurgiu	77
Hirsova	151
Iasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitesti	98
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374

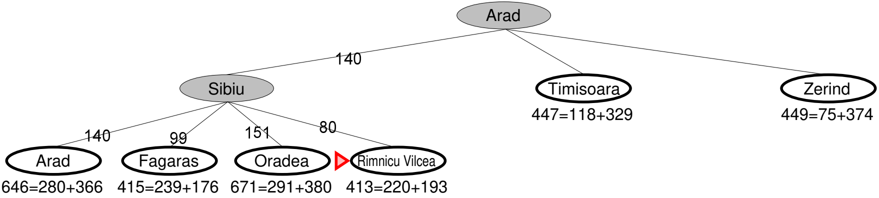
A* search example



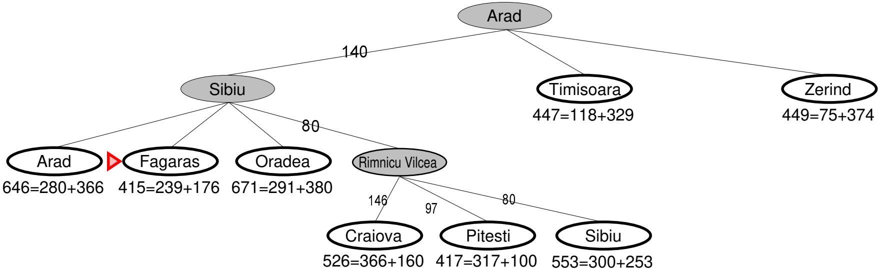
A* search example



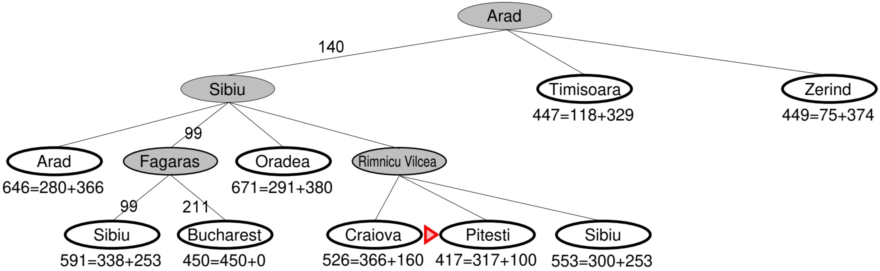
A* search example



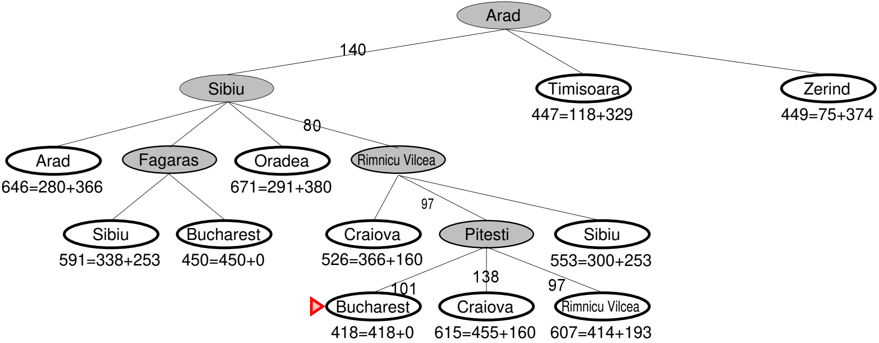
A* search example



A^* search example

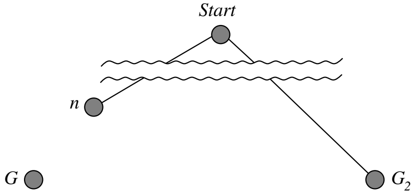


A* search example



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.



$$
\begin{aligned}
f(G_2) &= g(G_2) && \text{since } h(G_2) = 0 \\
&> g(G) && \text{since } G_2 \text{ is suboptimal} \\
&= g(n) + h^*(n) && \text{By the definition of } h^*(n) \\
&\geq g(n) + h(n) && \text{since } h \text{ is admissible} \\
&= f(n)
\end{aligned}
$$

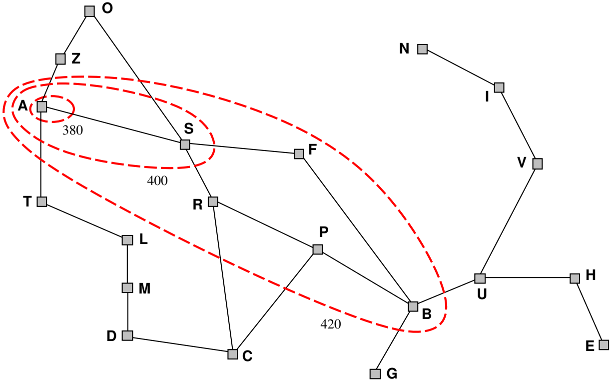
Since $f(G_2) > f(n)$, A* will never select G_2 for expansion

Optimality of A* (more useful)

Lemma: A* expands nodes in order of increasing f value*

Gradually adds "f-contours" of nodes (cf. breadth-first adds layers)

Contour i has all nodes with $f \leq f_i$, where $f_i < f_{i+1}$



Properties of A*

Complete?? Yes, unless there are infinitely many nodes with $f \leq f(G)$

Properties of A*

Complete?? Yes, unless there are infinitely many nodes with $f \leq f(G)$
Time?? $O(b^{\Delta})$, $\Delta = h^* - h$, h^*: actual cost from the root to the goal; h: estimated cost
Proof idea: $f(G) - f(S) = \Delta$, S: starting node. f increase along path $\rightarrow O(b^{\Delta})$(time complexity of BFS)

Properties of A*

Complete?? Yes, unless there are infinitely many nodes with $f \leq f(G)$

Time?? $O(b^{\Delta})$, $\Delta = h^* - h$, h^*: actual cost from the root to the goal; h: estimated cost

Proof idea: $f(G) - f(S) = \Delta$, S: starting node. f increase along path $\rightarrow O(b^{\Delta})$(time complexity of BFS)

Space?? Keeps all nodes in memory

Properties of A*

Complete?? Yes, unless there are infinitely many nodes with $f \leq f(G)$

Time?? $O(b^{\Delta})$, $\Delta = h^* - h$, h^*: actual cost from the root to the goal; h: estimated cost

Proof idea: $f(G) - f(S) = \Delta$, S: starting node. f increase along path $\rightarrow O(b^{\Delta})$(time complexity of BFS)

Space?? Keeps all nodes in memory

Optimal?? Yes—cannot expand f_{i+1} until f_i is finished

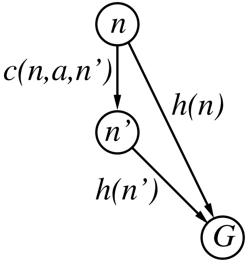
Properties of A*

Complete?? Yes, unless there are infinitely many nodes with $f \leq f(G)$
Time?? $O(b^{\Delta})$, $\Delta = h^* - h$, h^*: actual cost from the root to the goal; h: estimated cost
Proof idea: $f(G) - f(S) = \Delta$, S: starting node. f increase along path $\rightarrow O(b^{\Delta})$(time complexity of BFS)
Space?? Keeps all nodes in memory
Optimal?? Yes—cannot expand f_{i+1} until f_i is finished
A* expands all nodes with $f(n) < C^*$, C^* is the cost of optimal solution path
A* expands some nodes with $f(n) = C^*$
A* expands no nodes with $f(n) > C^*$

Proof of lemma: Consistency

A heuristic is **consistent** if

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



where $c(n, a, n')$ is the cost of path from n to n' by choosing action a

If h is consistent, we have

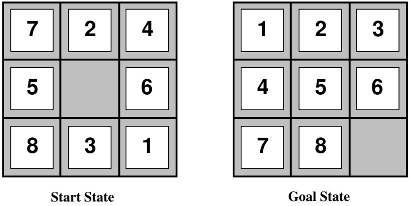
$$\begin{aligned} f(n') &= g(n') + h(n') \\ &= g(n) + c(n, a, n') + h(n') \\ &\geq g(n) + h(n) \\ &= f(n) \end{aligned}$$

I.e., $f(n)$ is nondecreasing along any path. **Thus, the goal state with the lowest f-cost will be found first**

Every consistent heuristic is admissible!

Admissible heuristics

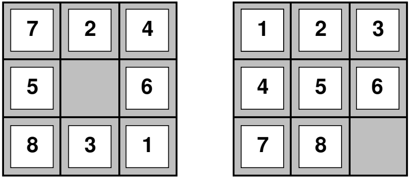
E.g., for the 8-puzzle:
$h_1(n)$ = number of misplaced tiles
$h_2(n)$ = total **Manhattan** distance
(i.e., no. of squares from desired location of each tile)



$\underline{h_1(S) =}$??
$\underline{h_2(S) =}$??

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)



Start State Goal State

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

Relaxed problems

Admissible heuristics can be derived from the **exact** solution cost of a **relaxed** version of the problem

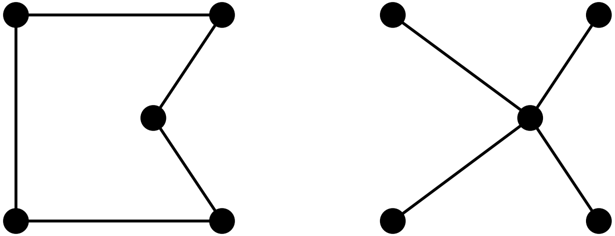
If the rules of the 8-puzzle are relaxed so that a tile can move **anywhere**, then $h_1(n)$ gives the shortest solution

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

Relaxed problems contd.

Well-known example: **travelling salesperson problem** (TSP)
Find the shortest tour visiting all cities exactly once



Minimum spanning tree can be computed in $O(n^2)$
and is a lower bound on the shortest (open) tour

Summary

Heuristic functions estimate costs of shortest paths

Good heuristics can dramatically reduce search cost

Greedy best-first search expands lowest h

– incomplete and not always optimal

A* search expands lowest $g + h$

– complete and optimal

– also optimally efficient (up to tie-breaks, for forward search)

Admissible heuristics can be derived from exact solution of relaxed problems

Acknowledgment

The slides are adapted from Stuart Russell, Guy Van den Broeck et al.