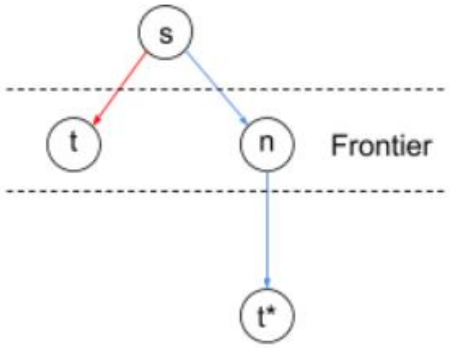
National University of Singapore
School of Computing
CS3243 Introduction to AI

Tutorial 3: Informed Search (Solutions)

Issued: January 27, 2021 Discussion in: Week 5

1. Prove that the tree-based variant of the A* search algorithm is optimal when an admissible heuristic is utilised.

   Solution: Assume the following search tree below.



   Let s be the start state, n an intermediate state (including, possibly, a goal state) along the optimal path, t be a suboptimal goal state and t^* be the goal along an optimal path.

   An optimal solution implies that n must be expanded before t.

   Proof by contradiction:

   - Let us assume that a non-optimal solution is found - i.e., that t is expanded before n, which implies that (A): $f(t) < f(n)$
   - In other words, given the above frontier, only when $f(t) < f(n)$ would we expand t before n

- However, since t is not on the optimal path and t^* is the goal on the optimal path, we have:

$f(t) > f(t^*)$	
$f(t) > g(t^*)$	since $h(t^*)$ = 0
$f(t) > g(n) + d(n, t^*)$	where $d(n, t^*)$ is the actual cost from n to t^*
$f(t) > g(n) + h(n)$	asserting admissibility
$f(t) > f(n)$	this contradicts (A)

- Note: we do not consider $f(t) = f(t^*)$ since that would mean $f(t)$ is equally optimal

2. Prove that the graph-based variant of the A* search algorithm is optimal when a consistent heuristic is utilised.

Solution: A heuristic h is consistent if $f(n)$ is non-decreasing along any path

- Theorem: if h is consistent, A* is optimal under a graph-based implementation

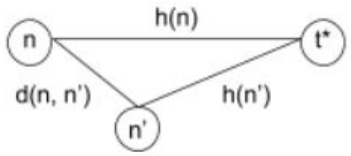
  More specifically, for every node n, and every successor n' of n generated by action a:

$$h(n) \leqslant d(n, n') + h(n')$$

- Lemma:

$f(n') = g(n') + h(n')$	
$f(n') = g(n) + d(n, n') + h(n')$	
$f(n') \geqslant g(n) + h(n)$	assuming consistency
$f(n') \geqslant f(n)$	

  This constitutes a triangular inequality:



- The A* search algorithm thus explores nodes in a non-decreasing order of f value; essentially, with each exploration, we may add a new contour (similar to how UCS explores nodes in a non-decreasing order of g value)
- When A* expands n, the optimal path to n has been found (again, similar to UCS)
- Proof by contradiction:
  - Assume that n is explored. However, the path to n is not optimal.
  - This implies that there exists some node on the optimal path to n that was not explored, but that is on the frontier. Denote this node m.
  - Note that the second node on the optimal path - i.e., the node after the start node, must be on the frontier, which implies, that at least one unexpanded node on the optimal path will be on the frontier when the node n was explored via a non-optimal path.
  - However, since A*, with a consistent heuristic, explores nodes in a non-decreasing order of f value, m should have been explored before n.

3. Refer to Figure 1 below. Apply the best-first search algorithm to find a path from Fagaras to Craiova, using the following evaluation function $f(n)$:

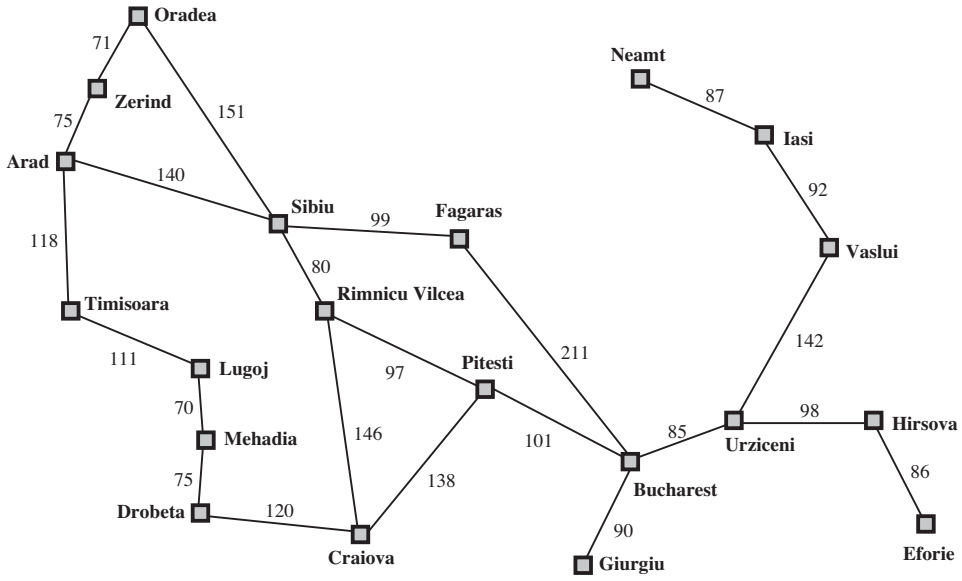
$$f(n) = g(n) + h(n)$$

where $h(n) = |h_{SLD}(\text{Craiova}) - h_{SLD}(n)|$ and $h_{SLD}(n)$ is the straight-line distance from any city n to Bucharest given in Figure 3.22 of AIMA 3rd edition (reproduced in Fig. 1).

   (a) Trace the best-first search algorithm by showing the series of search trees as each node is expanded, based on the `TREE-SEARCH` algorithm below (Fig. 2).

   (b) Prove that $h(n)$ is an admissible heuristic.

Solution:



Values of h_{SLD} - straight-line distances to Bucharest

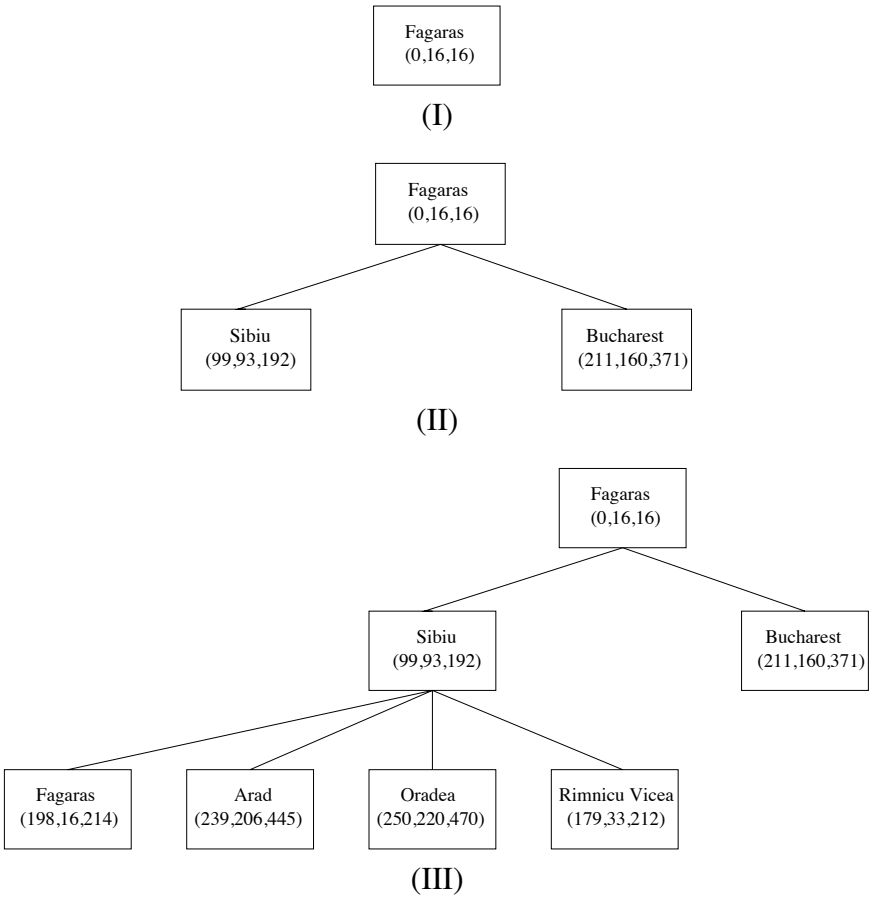
Arad	366
Bucharest	0
Craiova	160
Drobeta	242
Eforie	161
Fagaras	176
Giurgiu	77
Hirsova	151
Iasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitesti	100
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374

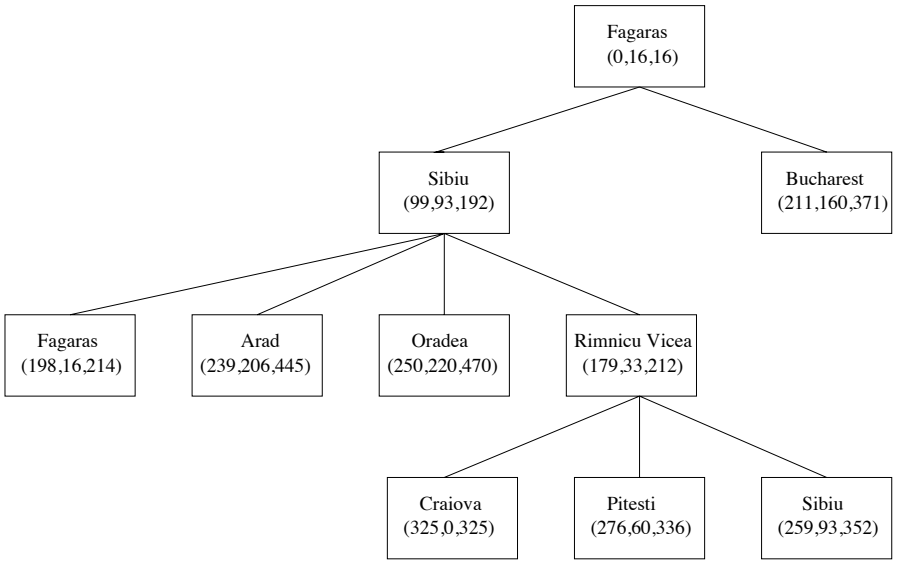
Figure 1: Graph of Romania.

function TREE-SEARCH(*problem*) **returns** a solution, or failure
  initialize the frontier using the initial state of *problem*
  loop do
    if the frontier is empty **then return** failure
    choose a leaf node and remove it from the frontier
    if the node contains a goal state **then return** the corresponding solution
    expand the chosen node, adding the resulting nodes to the frontier

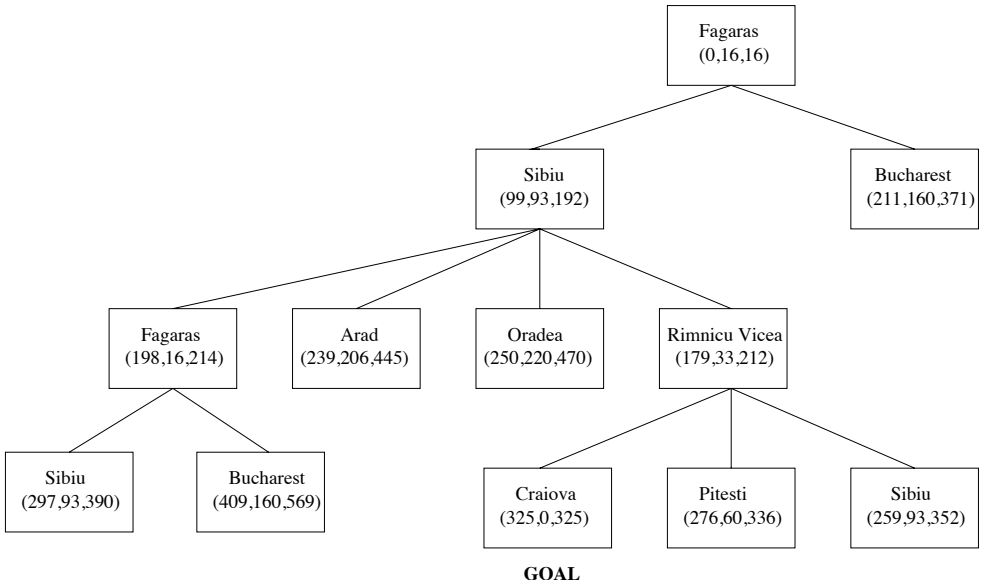
Figure 2: Tree search algorithm.

(a) The series of search trees, where the 3-tuple in each node denotes (g, h, f):





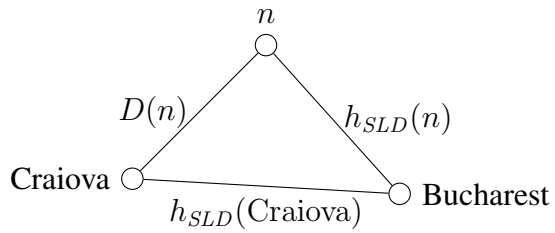
(IV)



(V)

Note that, in step IV, we need to expand Fagaras even though we have reached the destination because getting to the destination is not sufficient. A^* finds the optimal solution.

(b) Now consider the following triangle:



Let $D(n)$ be the straight-line distance between n and Craiova. Using the Triangle Inequality we have that

$$D(n) + h_{SLD}(n) \geq h_{SLD}(\text{Craiova}) \Rightarrow D(n) \geq h_{SLD}(\text{Craiova}) - h_{SLD}(n)$$

and

$$D(n) + h_{SLD}(\text{Craiova}) \geq h_{SLD}(n) \Rightarrow D(n) \geq h_{SLD}(n) - h_{SLD}(\text{Craiova}).$$

We note that for any two numbers $a, b \in \mathbb{R}$

$$(c \geq a - b) \wedge (c \geq b - a) \Rightarrow c \geq |a - b|.$$

so

$$D(n) \geq |h_{SLD}(n) - h_{SLD}(\text{Craiova})| = h(n).$$

Now, let $h^*(n)$ be the true cost of reaching *Craiova* (**not Bucharest**, note that Craiova is the goal node now). We have $h^*(n) \geq D(n) \geq h(n)$, so $h(n)$ is admissible.

4. (a) Given that a heuristic h is such that $h(G) = 0$, where G is any goal state, prove that if h is consistent, then it must be admissible.

Solution: The proof is by induction on $k(n)$, which denotes the number of actions required to reach the goal from a node n to the goal node G.

Base case ($k = 1$, i.e., the node n is one step from G):
Since the heuristic function h is consistent,

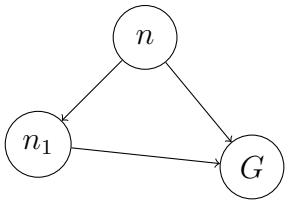
$$h(n) \leq c(n, a, G) + h(G)$$

Since $h(G) = 0$,

$$h(n) \leq c(n, a, G) = h^*(n)$$

Therefore, h is admissible.

Induction case:



Suppose that our assumption holds for every node that is $k - 1$ actions away from G, and let us observe a node n that is k actions away from G; that is, the least-actions optimal path from n to G has $k > 1$ steps. We write the optimal path from n to G as

$$n \to n_1 \to n_2 \to \cdots \to n_{k-1} \to G.$$

Since h is consistent, we have

$$h(n) \leq c(n, a, n_1) + h(n_1).$$

Now, note that since n_1 is on a least-cost path to G from n, we must have that the path $n_1 \to n_2 \to \cdots \to n_{k-1} \to G$ is a minimal-cost path from n_1 to G as well. By our induction hypothesis we have

$$h(n_1) \leq h^*(n_1)$$

Combining the two inequalities we have that

$$h(n) \leq c(n, a, n_1) + h^*(n_1)$$

Note that $h^*(n_1)$ is the cost of the optimal path from n_1 to G; by our previous observation (that $n_1 \to n_2 \to \ldots n_{k-1} \to G$ is an optimal cost path from n_1 to G), we have that the cost of the optimal path from n to G — i.e. $h^*(n)$ — is exactly $c(n, a, n_1) + h^*(n_1)$, which concludes the proof.

(b) Give an example of an admissible heuristic function that is not consistent.

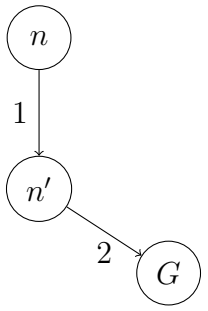
Solution: An example of an admissible heuristic function that is not consistent is as follows:

$$\begin{aligned} h(n) =& 3 \\ h(n') =& 1 \\ h(G) =& 0 \end{aligned}$$

h is admissible since

$$\begin{aligned} h(n) \leq& h^*(n) = 1 + 2 = 3 \\ h(n') \leq& h^*(n') = 2 \end{aligned}$$

However, h is not consistent since $3 = h(n) > c(n, a, n') + h(n') = 1 + 1 = 2$.



5. You have learned before that A* using graph search is optimal if $h(n)$ is consistent. Does this optimality still hold if $h(n)$ is admissible but inconsistent? Using the graph in Figure 3, let us now show that A* using graph search returns the non-optimal solution path (S, B, G) from start node S to goal node G with an admissible but inconsistent $h(n)$. We assume that $h(G) = 0$.

   Give nonnegative integer values for $h(A)$ and $h(B)$ such that A* using graph search returns the non-optimal solution path (S, B, G) from S to G with an admissible but inconsistent $h(n)$, and tie-breaking is not needed in A*.

   Solution: First let us recall that A^* maintains a priority queue containing elements of the form $\langle n, f(n)\rangle$ in increasing order of $f(n)$ (ties broken arbitrarily). The key difference between tree search and graph search is that under graph search, once a node n gets explored, it will never be added to the priority queue again. There are several possible solutions one might consider. First, let us set

$$\begin{aligned} h(S) &= 7 \\ h(B) &= 0 \\ h(A) &= 3, 4 \text{ or } 5 \\ h(G) &= 0 \end{aligned}$$

   Prove to yourself that this h is indeed admissible! Let's assume that $h(A) = 3$; what will A^* do in this case? It starts with a queue holding S with $f(S) = g(S) + h(S) = 0 + 7$: $[\langle S, 7\rangle]$, and it explores S. We have that

$$\begin{aligned} f(A) &= g(A) + h(A) = 2 + 3 = 5 \\ f(B) &= g(B) + h(B) = 4 + 0 = 4 \end{aligned}$$

   Our queue looks like this now

$$[\langle B, 4\rangle, \langle A, 5\rangle]$$

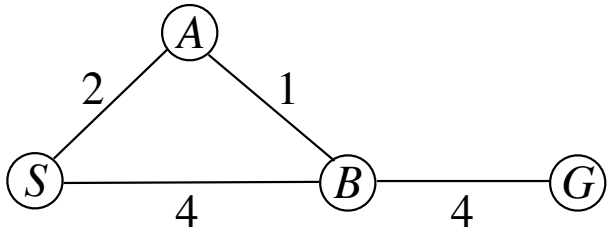


Figure 3: Graph.

so A^* selects B; it adds G to the queue next, with $f(G) = 8 + 0 = 8$, so our queue looks like this

$$[\langle A, 5\rangle, \langle G, 8\rangle].$$

It next explores A, *but does not add any of its neighbors again since this is graph search!* Thus, nothing gets added to the queue, G gets processed and we end up with a suboptimal solution.

What would tree search do on this heuristic? In order to understand more clearly what goes on, we add to our data structure the path that is taken to reach the node as well. Tree search would still select B first and then A, but having explored A it would add *all of its neighbors to the queue in order of* f, so after A gets explored we end up with a priority queue that looks like this:

$$[\langle B, 3, (S, A, B)\rangle, \langle G, 8, (S, B, G)\rangle, \langle S, 11, (S, A, S)\rangle].$$

Note that the entry $\langle S, 11, (S, A, S)\rangle$ comes from the cost of reaching S via A ($g(S) = 2 + 2 = 4$) plus $h(S) = 7$. We would next process B and again add its neighbors to the queue for

$$[\langle A, 7, (S, A, B, A)\rangle, \langle G, 7, (S, A, B, G)\rangle, \langle G, 8, (S, B, G)\rangle, \langle S, 11, (S, A, S)\rangle].$$

Note that $f(A) = g(A) + h(A) = (1 + 2 + 1) + 3 = 7$ and $f(G) = g(G) + h(G) = (3 + 4) + 0 = 7$. After we process A we (again) add its neighbors to have

$$[\langle B, 5, (S, A, B, A, B)\rangle, \langle G, 7, (S, A, B, G)\rangle, \langle G, 8, (S, B, G)\rangle,$$
$$\langle S, 11, (S, A, S)\rangle, \langle S, 14, (S, A, B, A, S)\rangle].$$

We process B again and obtain

$$[\langle G, 7, (S, A, B, G)\rangle, \langle G, 8, (S, B, G)\rangle, \langle A, 9, (S, A, B, A, B, A)\rangle,$$
$$\langle G, 9, (S, A, B, A, B, G)\rangle, \langle S, 11, (S, A, S)\rangle, \langle S, 14, (S, A, B, A, S)\rangle].$$

At this point G gets (finally) processed and we are done. We mention that in this example we broke ties in path costs arbitrarily (the result remains the same for other tie-breaking schemes). Other admissible inconsistent heuristics that work here include setting

$$h(B) = 1, h(A) = 4 \text{ or } 5$$
$$h(B) = 2, h(A) = 5.$$

All we need to ensure is that $h(B) < h(A) - 2$ to hold for h to be admissible but not consistent.