National University of Singapore
School of Computing
CS3243 Introduction to AI

Tutorial 3: Informed Search (Solutions)

Issued: May 26, 2020 Discussion in: Week 3, Tuesday Session

1. Under the context of informed search, state the difference between the functions: $g(n)$, $h(n)$, and $f(n)$.

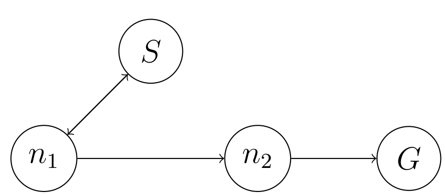
   Solution: The function $g(n)$ denotes the actual path cost from an initial state, s, to state n. This path cost is calculated based on one specific path between s and n.

   The function $h(n)$, known as a heuristic, denotes the approximated path cost from state n to (the closest) goal state g.

   The function $f(n)$ denotes the evaluation function that is adopted by the informed search algorithm in question - e.g., in the case of the greedy best-first search algorithm, $f(n) = h(n)$; in the case of the A* search algorithm, $f(n) = g(n) + h(n)$.

2. (a) Provide a counter-example to show that the tree-based variant for the greedy best-first search algorithm is incomplete.

   Solution: Consider the following search space, where: $h(S) = 3$, $h(n_1) = 4$, $h(n_2) = 5$, and $h(G) = 0$. Each time S is explored, we add n_1 to the front of the Frontier, and each time n_1 is explored, we add S to the front of the Frontier. Notice that n_2 is never at the front of the Frontier. This causes the greedy best-first search algorithm to continuously loop over S and n_1.

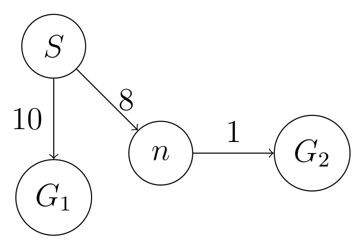


(b) Briefly explain why the graph-based variant of the greedy best-first search algorithm is complete.

Solution: Assuming a finite search space, the graph-based variant of the greedy best-first search algorithm will eventually visit all states within the search space and thus find a goal state.

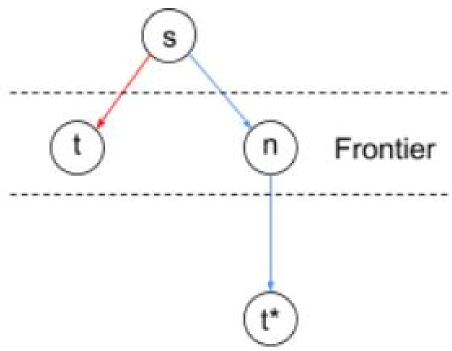
(c) Provide a counter-example to show that neither variant of the greedy best-first search algorithm is optimal.

Solution: Consider the following search space, where: $h(S) = 9$, $h(G_1) = 0$, $h(n) = 1$, and $h(G_2) = 0$. With either variant of the greedy best-first search algorithm, when S is explored, G_1 would be added to the front of the Frontier and then explored next, resulting in the algorithm returning the non-optimal $S \rightarrow G_1$ path.



3. 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 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): $f(t) \leqslant f(n)$
- 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 optimal, 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 $f(t) = f(n)$ since that would mean $f(t)$ is equally optimal

4. 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

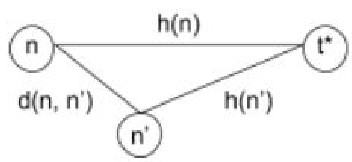
  More specifically, for every node n, and every successor n' of n generated by action a:

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

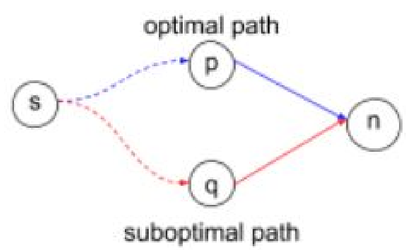
- Lemma:

$$\begin{array}{ll} f(n') = g(n') + h(n') & \\ f(n') = g(n) + d(n, n') + h(n') & \\ f(n') \geqslant g(n) + h(n) & \text{assuming consistency} \\ f(n') \geqslant f(n) & \end{array}$$

  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:
  - Let n be the first node reached by a sub-optimal (via q)
  - But n is on the optimal path (via p)



  - On the suboptimal path, n is expanded before p, thus, we have:
    $$f(n) < f(p)$$
  - However, since p precedes n on the optimal path, we have:
    $$f(p) < f(n)$$
  - This is a contradiction

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

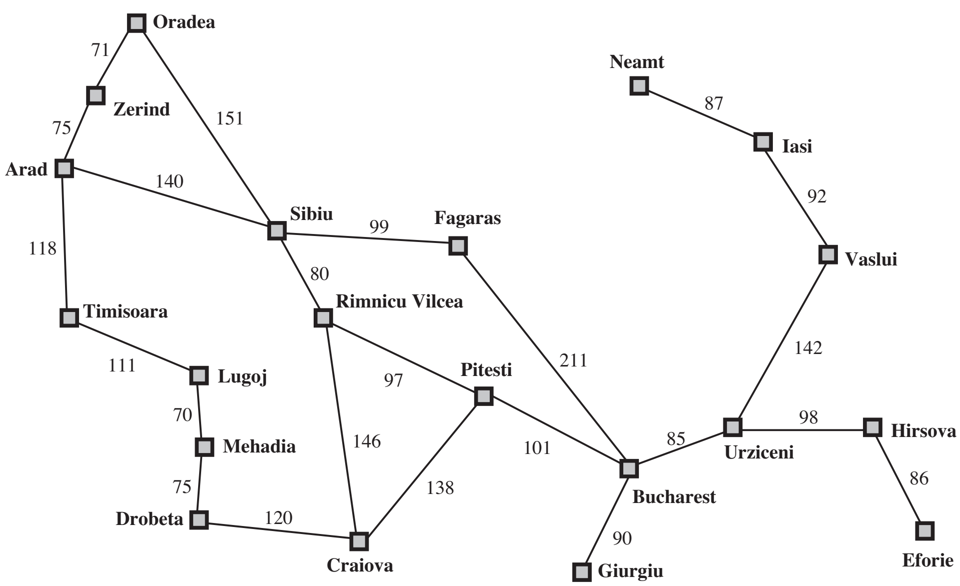
$$f(n) = g(n) + h(n)$$

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

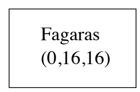
Arad	366
Bucharest	0
Craiova	160
Drobeta	242
Eforie	161
Fagaras	176
Giurgiu	77
Hirsova	151
Iasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitesti	100
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374

Figure 1: Graph of Romania.

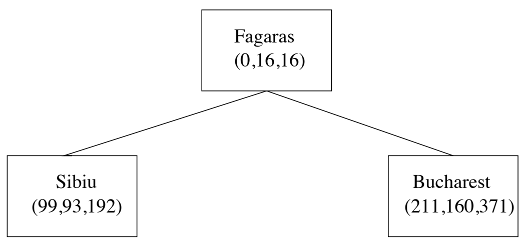
function TREE-SEARCH(*problem*) **returns** a solution, or failure
  initialize the frontier using the initial state of *problem*
  loop do
    if the frontier is empty **then return** failure
    choose a leaf node and remove it from the frontier
    if the node contains a goal state **then return**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

Figure 2: Tree search algorithm.

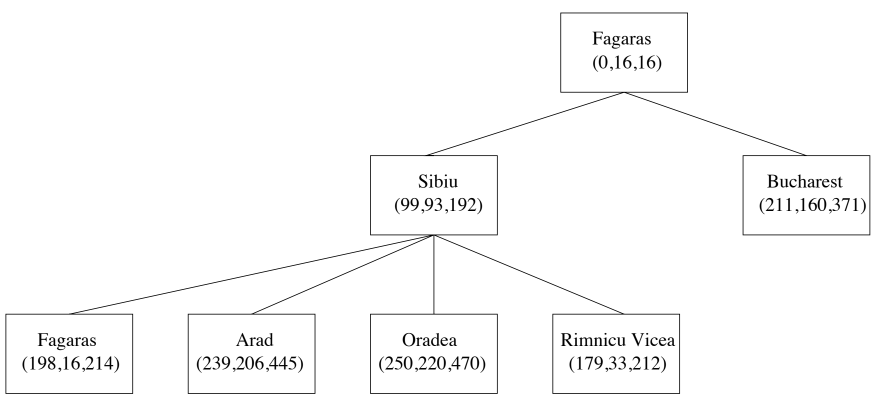
(a) The series of search trees, where the 3-tuple in each node denotes (g, h, f):



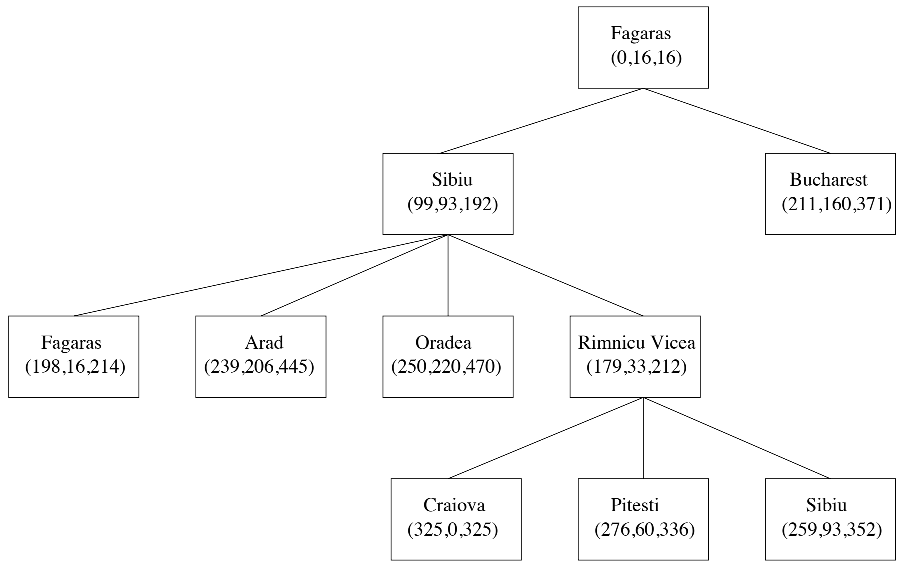
(I)



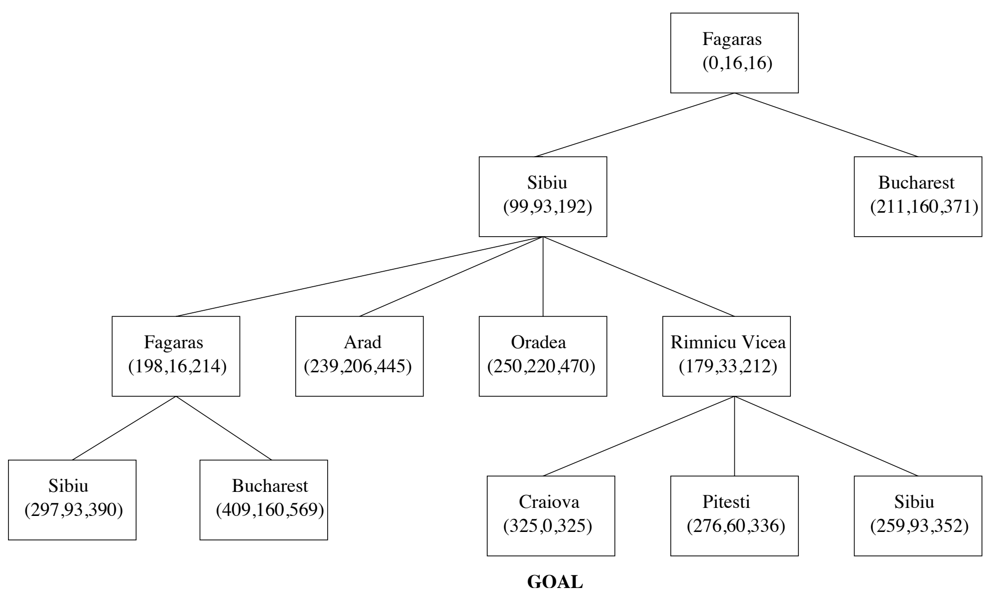
(II)



(III)



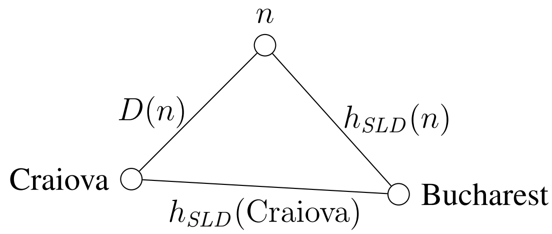
(IV)



(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.

6. Consider the 8-puzzle that we discussed in class. Suppose we define a new heuristic function h_3 which is the average of h_1 and h_2, and another heuristic function h_4 which is the sum of h_1 and h_2. That is,

$$h_3 = \frac{h_1 + h_2}{2}$$
$$h_4 = h_1 + h_2$$

where h_1 and h_2 are defined as "the number of misplaced tiles", and "the sum of the Manhatten distances between current and goal positions for each tile", respectively. Are h_3 and h_4 admissible? If admissible, compare their dominance with respect to h_1 and h_2.

Solution: Since $h_1(n) \leq h_2(n)$ for all n,

$$h_3(n) = \frac{h_1(n) + h_2(n)}{2} \leq \frac{h_2(n) + h_2(n)}{2} = h_2(n) \leq h^*(n)$$

where the last inequality holds since h_2 is admissible. Hence, h_3 is admissible. Since for all n,

$$h_1(n) = \frac{h_1(n) + h_1(n)}{2} \leq \frac{h_1(n) + h_2(n)}{2} = h_3(n),$$

we have $h_1(n) \leq h_3(n) \leq h_2(n)$ for all n. That is, h_2 dominates h_3, and h_3 dominates h_1.

On the other hand, h_4 is not admissible. Consider a board n in which moving one tile will reach the goal. In this case, $h_1(n) = h_2(n) = h^*(n) = 1$, and

$$h_4(n) = h_1(n) + h_2(n) = 1 + 1 > h^*(n) \; .$$

7. (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,

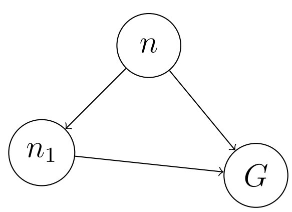
$$h(n) \leq c(n, a, G) + h(G)$$

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

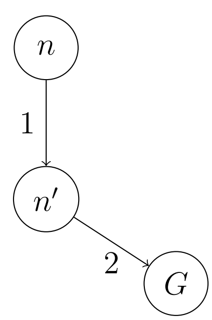
Solution: An example of an admissible heuristic function that is not consistent is as follows:

$$h(n) = 3$$
$$h(n') = 1$$
$$h(G) = 0$$

h is admissible since

$$h(n) \leq h^*(n) = 1 + 2 = 3$$
$$h(n') \leq h^*(n') = 2$$

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



8. 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

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

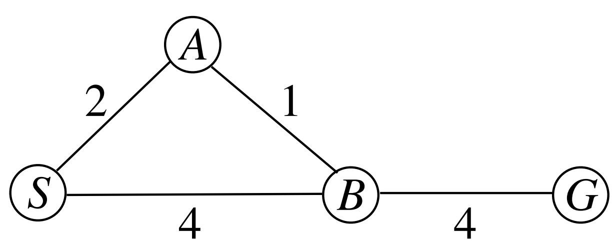


Figure 3: Graph.

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

$$\begin{aligned}&[\langle B, 5, (S, A, B, A, B)\rangle, \langle G, 7, (S, A, B, G)\rangle, \langle G, 8, (S, B, G)\rangle,\\ &\langle S, 11, (S, A, S)\rangle, \langle S, 14, (S, A, B, A, S)\rangle].\end{aligned}$$

We process B again and obtain

$$\begin{aligned}&[\langle G, 7, (S, A, B, G)\rangle, \langle G, 8, (S, B, G)\rangle, \langle A, 9, (S, A, B, A, B, A)\rangle,\\ &\langle G, 9, (S, A, B, A, B, G)\rangle, \langle S, 11, (S, A, S)\rangle, \langle S, 14, (S, A, B, A, S)\rangle].\end{aligned}$$

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

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

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

9. Suppose that the A* search algorithm utilizes $f(n) = w \times g(n) + (1 - w) \times h(n)$, where $0 \leq w \leq 1$ (instead of $f(n) = g(n) + h(n)$). For any value of w, an optimal solution will be found whenever h is a consistent heuristic.

   Determine if the statement above is True or False, and provide a rationale.

   Solution: This is false. When $w = 0$ we just get greedy search which is suboptimal (as shown with Question 2c above).