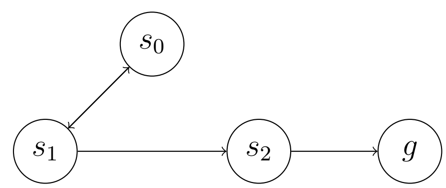
National University of Singapore
School of Computing
CS3243 Introduction to AI

Tutorial 2: Informed Search (Solutions)

1. (a) Provide a counter-example to show that the **tree search** implementation for the **Greedy Best-First Search** algorithm is **incomplete**.

Solution: Consider the following search space with initial state s_0, goal state g, and where $h(s_0)$ = 3, $h(s_1)$ = 4, $h(s_2)$ = 5, and $h(g)$ = 0.



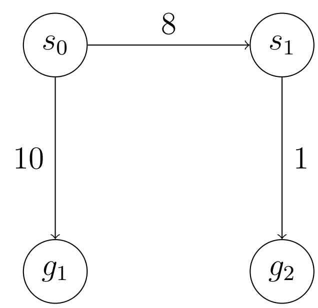
Each time s_0 is explored, we add s_1 to the front of the frontier, and each time s_1 is explored, we add s_0 to the front of the frontier. Notice that s_2 is never at the front of the frontier. This causes the greedy best-first search algorithm to continuously loop over s_0 and s_1.

(b) Briefly explain why the **graph search** implementation for the **Greedy Best-First Search** algorithm is **complete**.

Solution: Assuming a finite search space, a graph search implementation of the greedy best-first search algorithm will eventually visit all states within the search space. As such, the algorithm would either find a goal state and return the path to it, or else, indicate that there is no solution if a goal is not found.

(c) Provide a counter-example to show that neither the **tree search** nor the **graph search** implementations for the **Greedy Best-First Search** algorithm are **optimal**.

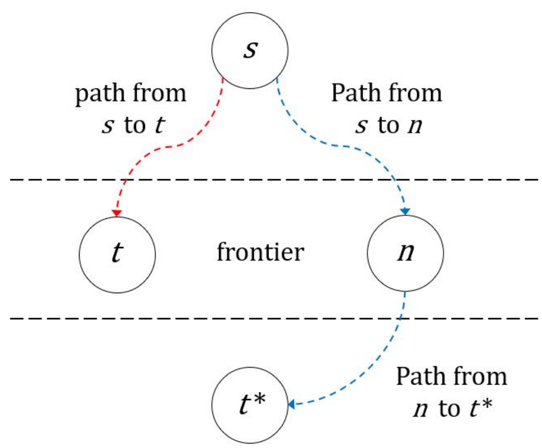
Solution: Consider the following search space with initial state s_0, goal states g_1 and g_2, and where $h(s_0) = 9$, $h(g_1) = 0$, $h(s_1) = 1$, and $h(g_2) = 0$.



With either implementation, when s_0 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 $s_0 \rightarrow g_1$ path.

2. (a) Prove that the **tree search** implementation of the **A* Search** algorithm is optimal when an **admissible heuristic** is utilised.

Solution: Assume the following search tree below.



Let s be the initial state, n be an intermediate state along the optimal path, t be a sub-optimal goal state (i.e., a goal state reached via a suboptimal path), and t^* be the goal along the optimal path.

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

Proof by contradiction:

- Let us assume that a suboptimal solution is found – i.e., that t is expanded before n, which implies that (A): $f(t) \leq f(n)$
- 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 but t^* is, we have:

$f(t) > f(t^*)$	
$f(t) > g(t^*)$	since $h(t^*) = 0$
$f(t) > g(n) + p(n, t^*)$	where $p(n, t^*)$ 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.

(b) Prove that the **graph search** implementation of the **A* Search** algorithm is optimal when a **consistent heuristic** is utilised. Assume graph search **Version 3**.

Solution: Similar to the UCS proof of optimality under graph search, we must show that when a node n is popped from the frontier, we have found the optimal path to it.

Let $f(s_k) = g(s_k) + h(s_k)$ be the minimum f value for s_k we have observed when s_k is popped.

Let the optimal path from the start node, s_0, to any node, s_g , be $P = s_0, s_1, ..., s_{g-1}, s_g$. We must show that when we pop s_g, $f(s_g) = g(s_g) + h(s_g) = g^*(s_g) + h(s_g)$, where $g^*(s_g)$ denotes the optimal path cost from s_0 to s_g via P.

Base case: $f(s_0) = g(s_0) + h(s_0) = g^*(s_0) + h(s_0) = h(s_0)$ as s_0 is the start node.

Induction step: Assume that for all $s_0, s_1, ..., s_k$, when we pop s_i. $f(s_i) = g(s_i) + h(s_i) = g^*(s_i) + h(s_i)$, or rather, $g(s_i) = g^*(s_i)$.

Since $g^*(s_{k+1})$ is the minimum path cost from s_0 to s_{k+1}, we know that:

$$\begin{aligned} g(s_{k+1}) + h(s_{k+1}) &\geq g^*(s_{k+1}) + h(s_{k+1}) \\ g(s_{k+1}) &\geq g^*(s_{k+1}) && \text{(A)} \end{aligned}$$

To make sure that each s_{k+1} is only popped after we pop s_k, the condition $f(s_k) \leq f(s_{k+1})$, or rather $h(s_k) \leq c(s_k, s_{k+1}) + h(s_{k+1})$, where $c(s_k, s_{k+1})$ is the action cost from s_k to s_{k+1}, is required, which leads us to assert that h is consistent.

Consequently, just after s_k is popped, we have:

$$\begin{aligned} g(s_{k+1}) + h(s_{k+1}) &= min\{g(s_{k+1}) + h(s_{k+1}), g(s_k) + c(s_k, s_{k+1}) + h(s_{k+1})\} \\ g(s_{k+1}) &= min\{g(s_{k+1}), g(s_k) + c(s_k, s_{k+1})\} \\ &\leq g(s_k) + c(s_k, s_{k+1}) \\ &= g^*(s_k) + c(s_k, s_{k+1}) && \text{from our inductive hyp.} \\ &= g^*(s_{k+1}) && \text{(B)} \end{aligned}$$

From (A) and (B), we obtain $g(s_{k+1}) = g^*(s_{k+1})$. Hence, by induction, whenever we pop a node from the frontier, the optimal path to the node would have been found.

Also, given that graph search Version 3 is utililsed (i.e., only nodes popped from the frontier are added to reached), the optimal path would not be excluded.

3. (a) Given that a **heuristic** h is such that $h(t) = 0$, where t 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 t.

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

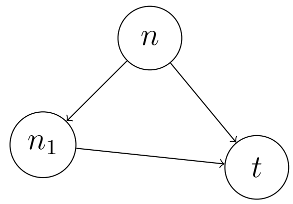
$$h(n) \leq c(n, a, t) + h(t)$$

Since $h(t) = 0$,

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

Therefore, h is admissible.

Induction case:



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

We write the optimal path from n to t as

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

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

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

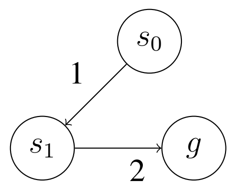
Combining the two inequalities we have that

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

Note that $h^*(n_1)$ is the cost of the optimal path from n_1 to t; by our previous observation (that $n_1 \to n_2 \to \ldots n_{k-1} \to t$ is an optimal cost path from n_1 to t), we have that the cost of the optimal path from n to t — 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** that is **not consistent**.

Solution: An example of an admissible heuristic function that is not consistent is as follows.
Consider a heuristic function h, such that $h(s_0) = 3$, $h(s_!) = 1$, and $h(t) = 0$ for the following graph.



h is admissible since

$$h(s_0) \leq h^*(s_0) = 1 + 2 = 3$$
$$h(s_1) \leq h^*(s_1) = 2$$

However, h is not consistent since $3 = h(s_0) > c(s_0, s_1) + h(s_1) = 1 + 1 = 2$.

4. We have seen various search strategies in class, and analysed their worst-case running time. Prove that ***any deterministic search algorithm*** will, in the worst case, **search the entire state space**. More formally, prove the following theorem

Theorem 1. *Let $\mathcal{A}$ be some complete, deterministic search algorithm. Then for any search problem defined by a finite connected graph $G = \langle V, E \rangle$ (where V is the set of possible states and E are the transition edges between them), there exists a choice of start node s_0 and goal node g so that $\mathcal{A}$ searches through the entire graph G.*

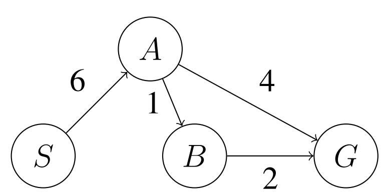
Solution: Let us begin by running $\mathcal{A}$ on the graph G, without setting any goal node at all: that is, there are no goal nodes at all in G. In this case, the algorithm $\mathcal{A}$ will return "False" when it explores the entire set V. Let $H_t(\mathcal{A}, s_0) \subseteq V$ be the set of nodes that $\mathcal{A}$ explores if it starts at s_0, and does not encounter a goal node at steps $1, \dots, t$ (at $t = 1$ we have $H_1(\mathcal{A}, s_0) = \{s_0\}$). We also let v_t be the node that $\mathcal{A}$ selects at time t given that it has observed the set $H_{t-1}(\mathcal{A}, s_0)$ so far. We note that it is entirely possible that $\mathcal{A}$ selects $v_t \in H_{t-1}(\mathcal{A}, s_0)$; however, we make a simple observation: the sequence $(H_t(\mathcal{A}, s_0))_{t=1}^{\infty}$ is weakly increasing in size, and there exists some time t^* such that for all $t > t^*$, $H_t(\mathcal{A}, s_0) =$

V; in other words, since $\mathcal{A}$ is a complete search algorithm, it will continue exploring the nodes in G until all nodes have been exhausted. Let us assume that t^* is the first time step for which $H_t(\mathcal{A}, s_0) = V$. In other words, at time $t^* - 1$, $|H_{t^*-1}(\mathcal{A}, s_0)| = |V| - 1$.

We now set the goal node to be v_{t^*}. From our previous argument, we know that when $\mathcal{A}$ starts at s_0 it will explore a set of size $|V| - 1$ before reaching v_{t^*}, realizing that it is a goal node and terminating. In other words, for any node s_0, if we select a goal node according to the above procedure, the algorithm $\mathcal{A}$ will exhaustively search through the entire graph before reaching a goal node.

Here is another, inductive proof. let us set the goal node to some arbitrary node g_1. If $\mathcal{A}$ searches through the entire graph G when g_1 is the goal, we are done; otherwise, let U_1 be the set of unsearched nodes when g_1 is the goal node. We take an arbitrary node g_2 in U_1 to be the goal; since $\mathcal{A}$ is deterministic and complete it will run the same search order that it did when g_1 was the goal, and then search through the nodes in U_1 until it reaches g_2. If it searched through all the nodes in U_1 as well, we are done, otherwise repeat. In general, suppose that we have set g_t to be the goal node and that $\mathcal{A}$ did not search through the entire graph until it reached g_t; let U_t be the set of unsearched nodes when g_t is the goal node. We set g_{t+1} to be some arbitrary node in U_t and rerun $\mathcal{A}$; since $\mathcal{A}$ is deterministic we know that when g_{t+1} is the goal we have $U_{t+1} \subset U_t$. Since $U_1 \supset U_2 \supset \cdots \supset U_t$ and the number of nodes in G is finite, there exists some iteration t^* such that $U_{t^*} = \emptyset$; thus g_{t^*} is a goal node for which $\mathcal{A}$ searches through the entire graph.

5. (a) In the search problem below, we have listed 5 heuristics. Indicate whether each **heuristic** is **admissible** and/or **consistent** in the table below.



	S	A	B	G	**Admissible**	**Consistent**
h_1	0	0	0	0		
h_2	8	1	1	0		
h_3	9	3	2	0		
h_4	6	3	1	0		
h_5	8	4	2	0		

Solution:

	S	A	B	G	Admissible	Consistent
h_1	0	0	0	0	True	True
h_2	8	1	1	0	True	False
h_3	9	3	2	0	True	True
h_4	6	3	1	0	True	False
h_5	8	4	2	0	False	False

(b) Write out the order of the nodes that are explored by the **A* Search** algorithm. Assume a **graph search** implementation that utilises heuristic h_4. Further, assume graph search **Version 3**.

You should express your answer in the form A–B–C (i.e., no spaces, all uppercase letters, delimited by the dash (–) character), which, for example, corresponds to the order A, B, and C.

Solution: S–A–B–G

(c) Which heuristic would you use? Explain why.

Solution: The heuristic h_3 corresponds to the exact cost from each node to the goal node (i.e., $h_3 = h^*$), and therefore it is the optimal heuristic.

(d) Prove or disprove the following statement:

The heuristic $h(n) = max\{h_3(n), h_5(n)\}$ is admissible.

Solution: This is false, since $4 = h(A) > h^*(A) = 3$.