CS3243 : Introduction to Artificial Intelligence

Tutorial 2

NUS School of Computing

July 5, 2022

Admin

- Tutorial Assignment Submission Guidelines
- Diagnostic Quizzes
- Project 1 (Due 6 July)

Review

- Informed Search
- The idea of Heuristics
- Admissibility and Consistency
- Dominance among heuristics

Tutorial Question 1(a)

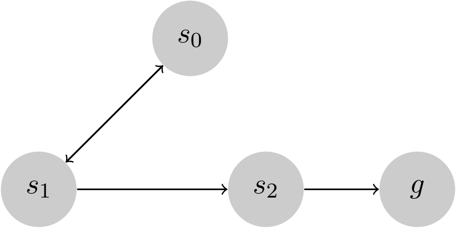
- Claim : Greedy Best-First Search with tree-based implementation is incomplete

Tutorial Question 1(a)

- Claim : Greedy Best-First Search with tree-based implementation is incomplete
- Any counterexamples?

Tutorial Question 1(a)

- Claim : Greedy Best-First Search with tree-based implementation is incomplete
- Any counterexamples?



Initial state s_0, Goal state g, and where $h(s_0) = 3$, $h(s_1) = 4$, $h(s_2) = 5$, and $h(g) = 0$

Tutorial Question 1(b)

- Why is the limited-graph-based implementatin of Greedy Best-First Search *complete*?

Tutorial Question 1(b)

- Why is the limited-graph-based implementatin of Greedy Best-First Search *complete*?
- Assuming a finite search space, the limited-graph-based implementation of the greedy best-first search algorithm will eventually visit all states within the search space and thus find a goal state.

Tutorial Question 1(c)

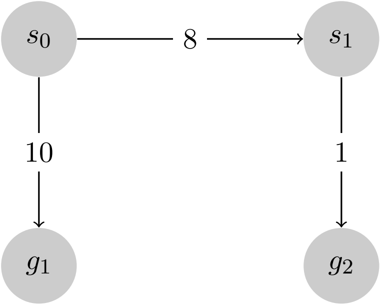
- Claim : Neither the tree-based nor the limited-graph-based implementations of the Greedy Best-First Search algorithm are optimal

Tutorial Question 1(c)

- Claim : Neither the tree-based nor the limited-graph-based implementations of the Greedy Best-First Search algorithm are optimal
- Any counterexamples?

Tutorial Question 1(c)

- Claim : Neither the tree-based nor the limited-graph-based implementations of the Greedy Best-First Search algorithm are optimal
- Any counterexamples?



Initial state s_0, Goal states g_1, g_2, and where $h(s_0) = 9$, $h(s_1) = 1$, $h(g_1) = 0$, and $h(g_2) = 0$

Tutorial Question 2(a)

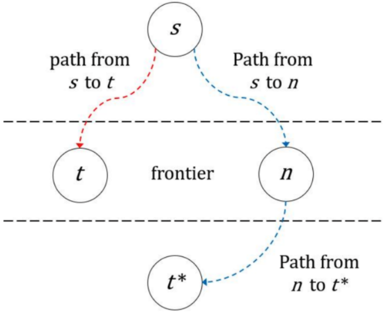
- Prove that the tree-based variant of the A* Search algorithm is optimal when an admissible heuristic is utilised

Tutorial Question 2(a)

- Prove that the tree-based variant of the A* Search algorithm is optimal when an admissible heuristic is utilised
- Any thoughts?

Tutorial Question 2(a)

- Consider the following search tree



- s is the initial state, n is the intermediate state along the optimal path
- t is the sub-optimal goal state
- t^* is the goal along the optimal path

Tutorial Question 2(a)

- An optimal solution implies that n must be expanded before t
- Proof by contradiction :
  - Let us assume that a suboptimal solution is found – ie, that t is expanded before n, which implies that $f(t) \leq f(n)$ ...(1)

Tutorial Question 2(a)

- An optimal solution implies that n must be expanded before t
- Proof by contradiction :
  - Let us assume that a suboptimal solution is found – ie, that t is expanded before n, which implies that $f(t) \leq f(n)$ ...(1)
  - In other words, given the above frontier, only when $f(t) < f(n)$ would we expand t before n

Tutorial Question 2(a)

- However, since t is not on the optimal path but t^* is, upon asserting tree-search such that all paths are considered, we have :

$$f(t) > f(t^*)$$

$$f(t) > g(t^*) \qquad \text{since } h(t^*) = 0$$

$$f(t) > g(n) + h^*(n) \qquad \text{where } h^*(n)$$

is the actual cost from n to t^*

$$f(t) > g(n) + h(n) \qquad \text{asserting admissibility*}$$

$$f(t) > f(n) \qquad \text{this contradicts } (1)$$

Tutorial Question 2(a)

- However, since t is not on the optimal path but t^* is, upon asserting tree-search such that all paths are considered, we have :

$$
\begin{aligned}
& f(t) > f(t^*) && \\
& f(t) > g(t^*) && \text{since } h(t^*) = 0 \\
& f(t) > g(n) + h^*(n) && \text{where } h^*(n) \\
& && \text{is the actual cost from } n \text{ to } t^* \\
& f(t) > g(n) + h(n) && \text{asserting admissibility*} \\
& f(t) > f(n) && \text{this contradicts } (1)
\end{aligned}
$$

- *Note: we do not consider $f(t) = f(n)$ as that means $f(t)$ is equally optimal

Tutorial Question 2(b)

- Prove that the limited-graph-based variant of the A* Search algorithm is optimal when a consistent heuristic is utilised

Tutorial Question 2(b)

- Prove that the limited-graph-based variant of the A* Search algorithm is optimal when a consistent heuristic is utilised
- Any thoughts?

Tutorial Question 2(b)

- Similar to the UCS proof of optimality, we must show that when a node n is popped from the frontier, we have found the optimal path to it

Tutorial Question 2(b)

- Similar to the UCS proof of optimality, 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

Tutorial Question 2(b)

- Similar to the UCS proof of optimality, 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 $P = s_0, s_1, ..., s_{g-1}, s_g$

Tutorial Question 2(b)

- Similar to the UCS proof of optimality, 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

Tutorial Question 2(b)

- Proof by induction
- **Base case**

$$
\begin{aligned}
f(s_0) &= g(s_0) + h(s_0) \\
&= g^*(s_0) + h(s_0) \\
&= h(s_0) && \text{as } s_0 \text{ is the start node}
\end{aligned}
$$

Tutorial Question 2(b)

- **Inductive case**
- 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, $g(s_i) = g^*(s_i)$

Tutorial Question 2(b)

- **Inductive case**
- 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 :

$$g(s_{k+1}) + h(s_{k+1}) \geq g^*(s_{k+1}) + h(s_{k+1})$$

$$g(s_{k+1}) \geq g^*(s_{k+1}) \qquad ...(1)$$

Tutorial Question 2(b)

- 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.

Tutorial Question 2(b)

- 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 :

$$\begin{aligned} g(s_{k+1}) + h(s_{k+1}) &= min\ \{g(s_{k+1}) + h(s_{k+1}), \\ &\qquad\qquad g(s_k) + c(s_k, s_{k+1}) + h(s_{k+1})\} \\ g(s_{k+1}) &= min\ \{g(s_{k+1}),\ g(s_k) + c(s_k, s_{k+1})\} \\ &\leq g(s_k) + c(s_k, s_{k+1}) \\ &= g^*(s_k) + c(s_k, s_{k+1}) \\ &= g^*(s_{k+1}) \qquad ...(2) \end{aligned}$$

Tutorial Question 2(b)

- From (1) and (2), 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

Tutorial Question 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

Tutorial Question 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
- 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

Tutorial Question 3(a)

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

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

Tutorial Question 3(a)

- **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.

Tutorial Question 3(a)

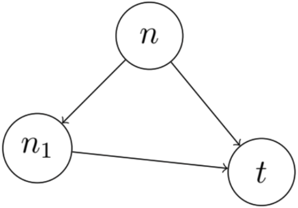
- **Inductive case** Suppose that our assumption holds for every node that is $k-1$ actions away from t, and let us observe a node n that is k actions away from t; that is, the least-actions optimal path from n to t has $k > 1$ steps. We write the optimal path from n to t as

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

Tutorial Question 3(a)

- Since h is consistent, we have

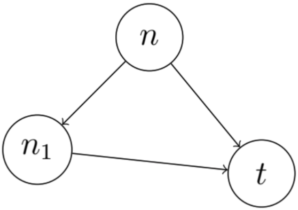
$$h(n) \leq c(n, a, n_1) + h(n_1).$$



Tutorial Question 3(a)

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

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



Tutorial Question 3(a)

- Combining the two inequalities we have that

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

Tutorial Question 3(a)

- 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 $n_1 \rightarrow n_2 \rightarrow \ldots \rightarrow n_{k-1} \rightarrow t$ 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.

Tutorial Question 3(b)

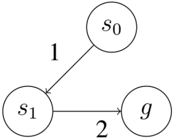
- Give an example of an admissible heuristic that is not consistent

Tutorial Question 3(b)

- Give an example of an admissible heuristic that is not consistent
- Any thoughts?

Tutorial Question 3(b)

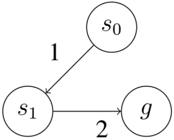
- One example is



- Consider a heuristic function h, such that $h(s_0) = 3$, $h(s_!) = 1$, and $h(t) = 0$ for the following graph

Tutorial Question 3(b)

- One example is

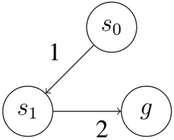


- Consider a heuristic function h, such that $h(s_0) = 3$, $h(s_!) = 1$, and $h(t) = 0$ for the following graph
- h is admissible since

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

Tutorial Question 3(b)

- One example is



- 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

$$\begin{aligned} h(s_0) &\leq h^*(s_0) = 1 + 2 = 3 \\ h(s_1) &\leq h^*(s_1) = 2 \end{aligned}$$

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

Tutorial Question 4

- Worst-case Scenario of Deterministic Search Algorithms
- Claim : 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

Tutorial Question 4

- Worst-case Scenario of Deterministic Search Algorithms
- Claim : 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
- Note: This question/solution assumes the search algorithm is uninformed, ie. does not use a heuristic function.

Tutorial Question 4

- Intuitive understanding of the solution?

Tutorial Question 4

- Intuitive understanding of the solution?
- Formal proof?

Tutorial Question 4

- Intuitive understanding of the solution?
- Formal proof?
- Suppose we set g_t as goal node and that $\mathcal{A}$ did not search through the entire graph until it reached g_t, then we let U_t be the set of unsearched nodes

Tutorial Question 4

- Intuitive understanding of the solution?
- Formal proof?
- Suppose we set g_t as goal node and that $\mathcal{A}$ did not search through the entire graph until it reached g_t, then we let U_t be the set of unsearched nodes
- We then set g_{t+1} to be some arbitrary node in U_t and rerun $\mathcal{A}$

Tutorial Question 4

- Intuitive understanding of the solution?
- Formal proof?
- Suppose we set g_t as goal node and that $\mathcal{A}$ did not search through the entire graph until it reached g_t, then we let U_t be the set of unsearched nodes
- We then set g_{t+1} to be some arbitrary node in U_t and rerun $\mathcal{A}$
- Since $\mathcal{A}$ is deterministic, $\mathcal{A}$ will search in the same order as before plus additional node(s) in U_t until it reaches g_{t+1}
- Hence, the set unsearched nodes will shrink; more specifically: $U_{t+1} \subset U_t$

Tutorial Question 4

- Intuitive understanding of the solution?
- Formal proof?
- Suppose we set g_t as goal node and that $\mathcal{A}$ did not search through the entire graph until it reached g_t, then we let U_t be the set of unsearched nodes
- We then set g_{t+1} to be some arbitrary node in U_t and rerun $\mathcal{A}$
- Since $\mathcal{A}$ is deterministic, $\mathcal{A}$ will search in the same order as before plus additional node(s) in U_t until it reaches g_{t+1}
- Hence, the set unsearched nodes will shrink; more specifically: $U_{t+1} \subset U_t$
- Since $U_1 \supset U_2 \supset \ldots \supset U_{t-1} \supset U_t$ and the number of nodes in graph G is finite, there exists some iteration t^* such that $U_{t^*} = \emptyset$; hence g_{t^*} would be the goal node for which $\mathcal{A}$ would search the entire graph.

Thank you!

If you have any questions, please don't hesitate. Feel free to ask!
We are here to learn together!