CS3243 Tutorial 2

Eric Han

Aug 30, 2022

Annoucements

Important admin

1. Attendance Marking on telegram; Honor system, check-in if you are here - else select that you are not in / ignore the check-in. Sum check will be performed.
2. Assignment 1 scores and comments will be out soon
    2.1 "Your submission must contain 20 words ot more." - This is a limitation on turnitin; fill in with some placeholder text to get around this.
3. Show me your bonus to collect your snacks; Note - early goal test for BFS only.
    3.1 Note: Early goal test for BFS only; AIMA Python.

Tidbits from tutorials

1. What is the type of proof for Q4? See Wiki Math proof.
2. Different formulations for a particular environment is possible; Some are better!

Welcome Survey - Why are you taking CS3243?

Figure 1: Why are you taking CS3243?

I love AI - simply amazing that we can create intelligence with high degree of autonomy; I believe in automation and it is the future of automation. General machine intelligence is still an open question.

Welcome Survey - How much do you know about AI/ML?

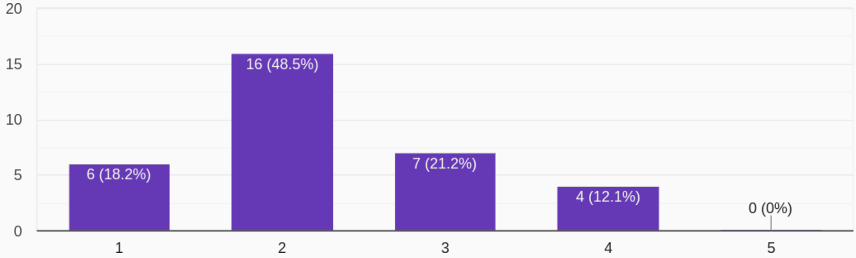

Figure 2: How much do you know about AI/ML?

Wherever you are, you will some takeaways; ask for help whenever you need - your peers and I are here to help!

Welcome Survey

What do you want to achieve in tutorials?

1. Get to know like-minded individuals and learn more about AI through class & peers.
2. I want to learn and be able to keep up with the class instead of blindly copying the slides and not understanding anything.
   2.1 Learn more in depth than lecture
   2.2 Improving understanding
   2.3 How to better understand AI and the algorithms that they utilise
3. Reinforce concepts taught in lecture
   3.1 Iron out misconceptions
   3.2 Fill in any gaps in my knowledge

Any suggestions to make our classes effective?

1. Make students answer question! [Yes you will; feel free to volunteer!]
2. Would be better if the slide is uploaded ASAP, . . . [Wed - Bonus]
3. using something like classkick allows for . . . [will explore, might be too late]

Previously from T01, Q5

Recap

Question 2b.

Question

Determine the **final path** found from the start (S) to the goal (G)?

Answer

`S-B-F-H-D-G`

Question 1

Recap

- What is Greedy-Best-First Search (GBFS) algorithm?
  - $f(n) = h(n)$
- What is - incomplete?
  - Incomplete describes the algorithm being stuck in a loop where $h(.)$ are lowest.
- What is the difference between tree / graph based algorithm?

Question 1a

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

Question 1a

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

Answer
Intuition: Create a situation where the path to G never gets explored.

s	$S0$	$S1$	$S2$	G
$h(s)$	3	4	5	0

Figure 3: GBFS Infinite Loop Example

1. $S0$ is explored, $S1$ is added to front of frontier.
2. $S1$ is explored, $S0$ is added to the front of frontier. etc. . .

Question 1b

Briefly explain why the **graph search** implementation of the **Greedy Best-First Search** algorithm is **complete**.

Question 1b

Briefly explain why the **graph search** implementation of the **Greedy Best-First Search** algorithm is **complete**.

Answer

- Graph-based implementation of GBFS will not explore redundant paths.
- Assuming a finite search space, there are finite paths.
- Hence, graph-based implementation of GBFS will eventually visit all states within the search space and find a goal state or not (all states visited).

Question 1c

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

Question 1c

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

Answer

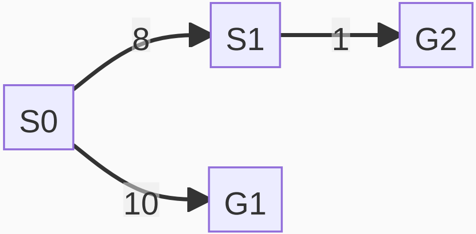

Figure 4: Non-Optimal Example

s	$S0$	$S1$	$G1$	$G2$
$h(s)$	9	1	0	0

1. $S0$ is explored, $G1$ is added to front of frontier - $f(.) = h(.)$
2. Non-optimal path $S0 \rightarrow G1$ returned.

Question 2

Recap

- What is the intuition behind A* Search?
- What is an admissible heuristic?
- What is a consistent heuristic?

Question 2

Recap

- What is the intuition behind A* Search?
- What is an admissible heuristic?
- What is a consistent heuristic?

Summary

- A* Search: $f(N) = g(N) + h(N)$
- Admissible heuristic: $h(N) \leq h^*(N)$
- Consistent heuristic: $h(N) \leq c(N, N') + h(N')$

Where,

- $f(.)$ - Evaluation Function.
- $g(.)$ - Cost Function from start to current node.
- $h(.)$ - Estimated cost from current node to goal.
- $c(N, N')$ - Action cost from N to N'.

Question 2a

Prove that the **tree** impl. of **A* Search** is optimal (**admissible heuristic** is used).

Question 2a

Prove that the **tree** impl. of **A* Search** is optimal (**admissible heuristic** is used).

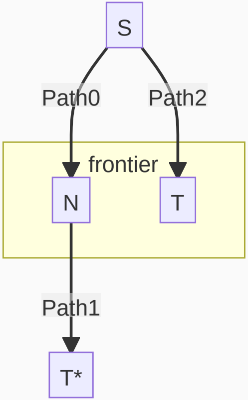

Figure 5: Illustration

- S - Inital State
- N - Intermediate State along optimal Path0, Path1
- T - Sub Optimal Goal state along sub-optimal Path2
- T^* - Goal state along optimal Path0, Path1

Proof by Contradiction:

1. Assume sub-optimal T is found instead of the T^*
2. $\implies$ T is found before $N \rightarrow f(T) < f(N)$
3. Since T^* is optimal, $f(T) > f(T^*)$
   3.1 $f(T) > f(T^*)$
   3.2 $f(T) > g(T^*) + h(T^*) \implies f(T) > g(T^*)$
   3.3 $f(T) > g(N) + h^*(N) \implies f(T) > g(N) + h(N)$
   3.4 $\implies f(T) > f(N)$
4. Tree-Search would have explored $T^*, \Rightarrow\Leftarrow f(T) < f(N)$

Question 2b

Prove that **graph** impl. of **A* Search** is optimal when a **consistent heuristic** is used.

Question 2b

Prove that **graph** impl. of **A* Search** is optimal when a **consistent heuristic** is used.

Intuition: Show that when a node is popped from frontier, optimal path to it is found.

Let the optimal path to any node s_g be $P_{0\to g} = s_0, s_1, \cdots, s_g$. We show that when s_g is popped, $f(s_g) = g(s_g) + h(s_g) = g^*(s_g) + h(s_g)$, where $g^*(s_g)$ is the optimal path cost.

- **Base**: $f(s_0) = g(s_0) + h(s_0) = h(s_0)$.
- **Inductive**: Assume for $P_{0\to k}$, $\implies g(s_k) = g^*(s_k)$ (optimal path to it is found)
  - $g^*(s_{k+1})$ is the minimum path cost, $\implies g(s_{k+1}) \geq g^*(s_{k+1})$
  - Since consistent, $h(s_k) \leq c(s_k, s_{k+1}) + h(s_{k+1})$; s_k is popped.
    - $f(s_{k+1}) \leq g(s_k) + c(s_k, s_{k+1}) + h(s_{k+1})$
    - $\implies g(s_{k+1}) + h(s_{k+1}) \leq g(s_k) + c(s_k, s_{k+1}) + h(s_{k+1})$
    - $\implies g(s_{k+1}) \leq g(s_k) + c(s_k, s_{k+1}) = g^*(s_k) + c(s_k, s_{k+1}) = g^*(s_{k+1})$
    - $\implies g(s_{k+1}) \leq g^*(s_{k+1})$
  - $g(s_{k+1}) \leq g^*(s_{k+1}) \wedge g(s_{k+1}) \geq g^*(s_{k+1}) \implies g(s_{k+1}) = g^*(s_{k+1})$

Question 3

Recap

- Admissible heuristic: $h(N) \leq h^*(N)$
- Consistent heuristic: $h(N) \leq c(N, N') + h(N')$

Question 3a

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**.

Question 3a

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**.

Answer
Intuition: Show on the number of action required to reach the goal from n to goal t.

Let the no. of actions k to be required to reach from n_k to t on optimal path $P_{n_k \to t}$. Node n_k is k steps away from t, ie. $k = 3 \implies P_{n_3 \to t} : n_3 \to n_2 \to n_1 \to t$.

- **Base**: 1 action; i.e. node n_1 is one step away from t.
  - Since consistent, $h(n_1) \leq c(n_1, t) + h(t)$ and $h(t) = 0$
  - $\implies h(n_1) \leq c(n_1, t) = h^*(n_1) \implies$ admissible.
- **Inductive**: Assume for $k - 1$ actions, path $P_{n_{k-1} \to t}, h(n_{k-1}) \leq h^*(n_{k-1})$.
  - Since consistent, $h(n_k) \leq c(n_k, n_{k-1}) + h(n_{k-1})$
  - $\implies h(n_k) \leq c(n_k, n_{k-1}) + h^*(n_{k-1}) = h^*(n_k) \implies$ admissible.

Question 3b

Give an example of an **admissible heuristic** that is **not consistent**.

Question 3b

Give an example of an **admissible heuristic** that is **not consistent**.

Answer

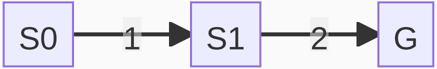

Figure 6: Example

s	$S0$	$S1$	G
$h(s)$	3	1	0
$h^*(s)$	3	2	0
$h(s) \leq h^*(s)$	T	T	T

Heuristic h is

- Admissible - $\forall s : h(s) \leq h^*(s)$
- Not consistent - $3 = h(s_0) > c(s_0, s_1) + h(s_1) = 2$

Question 4

We have seen various search strategies in class, and analyzed 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.

Answer

Intuition: Fixing s_0, we try to find a goal node g such that $\mathcal{A}$ searches all of G.

In the t-th iteration (Count only iterations where $\mathcal{A}$ explores a new node),

- Let g_t to be the goal node chosen only in t, to be $U_{t-1} \setminus U_t$
- Let U_t be the set of unsearched nodes, $V = U_0$.

Since there is finite $|V|$ and $\mathcal{A}$ is complete, then $U_0 \subset U_1 \subset U_2 \subset \cdots \subset U_{|V|} = \emptyset$,

1. Pick a random node in U_0 to be goal node and run $\mathcal{A}$.
2. $\mathcal{A}$ terminates on the k-th iteration with search sequence S_k, finding goal node g_k.
3. Since $\mathcal{A}$ is deterministic, when we reset the goal node to be any in U_k:
   3.1 Worse case to be g_{k+1} one step away from g_k.
   3.2 It will take the same search route $S_k + [g_{k+1}]$ to reach the new goal g_{k+1}.

Hence, we can always choose $g_{|V|}$ to be the goal node, such that $\mathcal{A}$ searches all of G.

Question 5

Assignment Question, due on Sunday.

Bonus Question - Work for Snack

To help you further your understanding, not compulsory.

Tasks

1. Fork the repository https://github.com/eric-vader/CS3243-2223s1-bonus
2. Look for `tutorial2.py`, we will be solving Q5a,b,d using code.
3. Some boilerplate code is given that has the 5 heuristics (also 5d), with the graph.
4. Compute whether each heuristic is admissible and/or consistent in the table per Q5.
5. Implement the **A* Search** algorithm; Assume a **graph** implementation that utilises heuristic h_4. Further, assume graph search **Version 3**.

To claim your snack, show me your forked repository and your code's output.