CS 486/686 Fall 2022
Assignment 1

Blake VanBerlo
85 marks

Due Date: 11:59pm EST on Tuesday October 4, 2022

Changes

- v1.1: Updated the description of h_2 in Question 1 to enhance clarity.
- v1.2 Updated description of the successor function in Question 2 to remove ambiguity in the set of remaining locations. Clarified sorting in successor function.
- v1.3 Updated instructions for getting successors in Q2. Fixed state examples in Q1.
- v1.4 Specified that $\emptyset$ cannot be a successor of $\emptyset$ in Q1.

Instructions

- Submit the signed academic integrity statement any written solutions in a file named `academic_integrity_statement.pdf` to the A1 project on Crowdmark. **(5 marks)**.
- Submit your written answers in a file named `A1_written.pdf` to the A1 project on Crowdmark. I strongly encourage you to complete your write-up in LaTeX, using this source file. If you do, in your submission, please replace the author with your name and student number. Please also remove the due date, the Instructions section, and the Learning goals section. Thanks!
- Submit any code to `Marmoset` at https://marmoset.student.cs.uwaterloo.ca/. Be sure to submit your code to the project named `Final`.
- No late assignment will be accepted. This assignment is to be done individually.
- Lead TAs:
  - Connor Raymond Stewart (crstewart@uwaterloo.ca)
  - Dake Zhang (dake.zhang@uwaterloo.ca)

  The TAs' office hours will be scheduled and posted on LEARN and Piazza.

Learning goals

Uninformed and Heuristic Search

- Understand the components of a search problem.
- Trace the execution of Breadth-First Search and Depth-First Search.
- Define an admissible heuristic. Explain why a heuristic is admissible.
- Trace the execution of the A* search algorithm with an admissible heuristic.
- Implement the breadth-first search and A* search algorithms.

1 Breakfast Time (26 marks)

Every morning, you eat the same breakfast: fried eggs (sunny-side-up), toast with peanut butter, a banana, and a fresh cup of coffee. Since you like to sleep in, you try to make your breakfast as quickly as possible. You will apply a search algorithm to determine the sequence of tasks that will help you make your breakfast as quickly as possible.

Below is a list of tasks that are needed to prepare each part of your breakfast, along with some notes. Each minute, you can either wait or initiate 1 task, and multiple items can be prepared simultaneously. Any secondary task Y_2 can only be initialized after task Y_1 is complete. Each task can only be completed once.

Task	Time [min]	Symbol
Crack the eggs	1	$\texttt{E}_1$
Fry the eggs	5	$\texttt{E}_2$
Toast the bread	2	$\texttt{T}_1$
Apply peanut butter	1	$\texttt{T}_2$
Peel the banana	1	$\texttt{B}_1$
Brew the coffee	3	$\texttt{C}_1$
Pour the coffee	1	$\texttt{C}_2$

Table 1: Tasks involved in preparing breakfast, along with the time they take to complete.

You define the search problem as follows:

- **States:** The set of breakfast items that have been started and/or completed, written in alphabetical order as a string. The time since the task was initialized is written as a superscript next to the task symbol. Symbols that are not present in the state represent tasks that have not been started yet. For example:
  - $E_2^1\, T_1^2$ means that the eggs have been frying for 1 minute and the toast has been toasting for 2 minutes (so it is ready to eat).
  - $B_1^2\, C_1^1\, E_2^6$ means that the banana was peeled 2 minutes ago, the coffee has been brewing for 1 minute, and the eggs have been frying for 6 minutes. The banana and the eggs are ready to eat.
- **Initial state:** $\emptyset$
- **Goal state:** Any state $B_1^b\, C_2^c\, E_2^e\, T_2^t$ where $b, c, t \geq 1$, and $e \geq 5$
- **Successor function:** State B is a successor of state A if all of the superscripts for unmodified items in A are incremented by 1 and optionally one of the following:
  - An item Y_1^t in A is converted to item Y_2^1 in B (if $\texttt{Y}_2$ is a valid symbol and t is at least the time required for $\texttt{Y}_1$ in Table 1).

– Item Y_1^1 is added to B (if $\texttt{Y}_\texttt{1}$ was not already in A)

Since you want to start breakfast as soon as possible, you cannot wait to begin breakfast. So as an exception to the above, $\emptyset$ is not a successor of $\emptyset$.

For example, suppose the current state is $E_1^1 T_1^2$. You could either wait, start brewing the coffee, peel the banana, or apply peanut butter to the toast. If the action is to wait, the next state will be $E_1^2 T_1^3$. If the action is to start brewing the coffee, the next state will be $C_1^1 E_1^2 T_1^3$. If the action is to apply peanut butter, the next state will be $E_1^2 T_2^1$.

- **Cost function:** The time elapsed (in minutes)

Please complete the following tasks.

(a) Recall that a heuristic $h(n)$ is consistent if, for every node n and every successor n' of n, we have $h(n) \leq c(n, n') + h(n')$. Prove that a heuristic function that is consistent is also admissible. You may assume that $h(n_G) = 0$ for any goal node n_G.

Hint: try a direct proof. Apply the definition of a consistent heuristic across an optimal path from some arbitrary node to a goal.

Marking Scheme: (6 marks)

- (4 marks) Correct proof
- (2 marks) The proof is clear, succinct, and easy to understand

Solutions: For $h(n)$ to be admissible, $h(n) \leq c(n, n_G) + h(n_G)$

For nodes which are directly connected to n_G, n_G can be subsituted into n', hence it is admissible. (These nodes shall be named n').

For nodes which are connected to nodes above, since in order to be consistent, $h(n) \leq c(n, n') + h(n')$, and $h(n') \leq c(n, n_G) + h(n_G)$. Thus, assuming that the cost is not negative, $h(n)$ is also admissible.

This reasoning can be repeated for all nodes along a path. Thus, a heuristic function that is consistent is also admissible. (Shown)

(b) Define $h_1(n)$ as the number of tasks that have yet to be initiated. Recall that each row in the table is a task. In **4 sentences or less**, explain why $h_1(n)$ is consistent.

Marking Scheme: (4 marks)

- (4 marks) A reasonable explanation

Solutions: As one traverses down the graph, tasks would be initiated and no task can be uninitiated. Therefore, number of tasks at the start is always $\geq$ than those after it. Thus, $h_1(n) \leq c(n, n') + h_1(n')$ for any node, and the hueristic is consistent.

(c) In **4 sentences or less**, explain why $h_1(n)$ is admissible.

Marking Scheme: (4 marks)

- (4 marks) A reasonable explanation

Solutions: From part a, a consistent function is also admissible. From part b, $h_1(n)$ is consistent. Thus, $h_1(n)$ is admissible.

(d) Define the heuristic $h_2(n)$ as the maximum time to complete an unprepared breakfast item. For example, $h_2(E_1^2\, C_2^1) = 5$ because the eggs require the maximum time to complete (5 minutes to fry the eggs). As another example, $h_2(B_1^2\, C_1^1\, E_2^5\, T_1^3) = 3$ because the coffee requires the maximum time to complete (2 minutes to finish brewing + 1 minute to pour).

Does h_2 dominate h_1? Explain why or why not in **4 sentences or less**

Marking Scheme: (4 marks)

- (2 marks) Correct answer
- (2 marks) A reasonable explanation

Solutions: No. To dominate,$h_2(n) \geq h_1(n)\ \forall\ n$ in the graph. In the case of $E_1^1 E_2^4$,$h_2(n) = 1$, while $h_1(n) = 5$.

(e) Execute the A* search algorithm on the breakfast problem using $h_2(n)$ as described above. Do not perform any pruning. Add nodes to the frontier in alphabetical order. Remember to stop if you expand the goal state. **Draw the search tree until you have expanded 5 nodes.**

When drawing nodes, remember to write the state in the node. Annotate each node n in the following format: $c + h = f$ where c is the cost of the path to n, h is $h_2(n)$, and f is the sum of the cost and the heuristic values. Clearly indicate which nodes you expanded and in what order. You do not need to write out the frontier, but the tree must show all paths expanded after removing a node from the frontier.

Break any f-value ties using lexicographical order. For example, $\texttt{B}_1\texttt{E}_1$ precedes $\texttt{E}_1$ and should be expanded first if the f-values for both nodes are the same.

We recommend using https://app.diagrams.net/ to draw the search tree.

Marking Scheme: (8 marks)

- (8 marks) Correct search tree with all nodes correctly expanded in order.

Solutions: refer to attached image (I'm sorry i don't know how to use LaTeX well)