CMPSCI 383 Homework 1

YOUR NAME HERE

Assigned: Jan 29 2018; Due: Feb 12 2018 @ 11:59 PM EST

Abstract

To submit this assignment, upload a pdf to Gradescope containing your responses to the written response questions below. You are required to use LaTeX for your write up. We suggest using sharelatex.com or overleaf.com, since they do not require you to install anything on your computer. When submitting your answers use the template LaTeX code provided and put your answers below the question they are answering. Do not forget to put your name on the top of the pdf. To submit the coding portion of the assignment upload your my_hw1.py to the Gradescope programming assignment. Do not change function definitions or return types unless otherwise specified. Your work for all parts of this assignment must be your own (do not collaborate with other students in any way when completing this assignment).

1 Syllabus (10 pts)

Read the course syllabus given on the course webpage. `http://psthomas.com/courses/CMPSCI_383_Spring2018.html`. Type your name below once you have read the course syllabus, including the portion that describes the consequences of cheating (an F in the course, not just for the assignment). If you have any questions about the syllabus, please email one of the TAs or the professor.

I, [**your name here**], have read the syllabus and understand the consequences of cheating.

2 Graph Search vs Tree Search (10 pts)

In class we discussed the graph-based forms of breadth first search, depth first search, and A* search. Read about the tree-based forms of these algorithms and their properties. Write a paragraph about the differences between tree and graph based search methods. You should *at least* answer the following questions:

1. When would you expect the graph-based algorithms to perform better than the tree-based ones?
2. When would you expect the tree-based algorithms to perform better than the graph-based ones?
3. Give examples of problems that would be better formulated as tree search problems, and problems that would be better formulated as graph search problems. Justify these examples (why are these problems better formulated each way?).

Answer:

3 Uninformed Search (20 pts)

This section is about uninformed search methods (methods which do not use a heuristic) to find the goal.

3.1 Depth First Search (DFS) (10 pts)

3.1.1 Summarizing (2 pts)

In a few sentences summarize in your own words how depth first search works. Give the complexity of its storage and computation time for the tree based version.

Answer:

3.1.2 Creating (8 pts)

Create a search problem such that DFS cannot find a solution. Do not use an example from the book or from the lecture. Explain why DFS will not find a solution for the problem. HINT: Think about how depth first search chooses the next node and how that relates to where the goal is.

Answer:

3.2 Breadth First Search (BFS) (10 pts)

3.2.1 Summarizing (2 pts)

In a few sentences summarize in your own words how breadth first search works. Give the complexity of its storage and computation time for the tree based version.

Answer:

3.2.2 Creating (8 pts)

Create 2 examples of a search problem such that in one problem, BFS finds a better solution (shorter path length) than DFS and in the other, DFS finds an optimal path faster. Do not use an examples from the book or the lecture. HINT: Think about when each of these methods will find the best path.

Answer:

4 Informed Search (60 pts)

4.1 Heuristics: (20 pts)

4.1.1 Summarizing: (5 pts)

In you own words explain what a heuristic is and how it is useful for search?

Answer:

In class we talked about *consistent* heuristics, which are relevant to graph-based A*. Read about *admissible* heuristics in the book (a variant of consistent heuristics that is useful for the tree-based form of A*). What is an admissible heuristic? Given an example of admissible heuristic for a car that has to drive between 2 cities.
Answer:

4.1.2 Evaluating: (15 pts)

For each problem below say whether each heuristic is admissible. If it is admissible, explain why. If it is not admissible, give an example of how it violates admissible.

An agent lives in an NxN gridworld. The agent's current position is given by the tuple (x_a, y_a) representing the row and column it is currently in. The goal location the agent wants to reach is represented by the tuple (x_g, y_g). The agent can only move up, down, left, or right 1 square at a time.

Are the following heuristics admissible:

1. $h(a) = \sqrt{(x_a - x_g)^2 + (y_a - y_g)^2}$
   Answer:

2. $h(a) = (x_a - x_g) + (y_a - y_g)$
   Answer:

3. $h(a) = |x_a - x_g| + |y_a - y_g|$
   Answer:

4. $h(a) = |x_a - x_g|$
   Answer:

5. $h(a) = \max\{|x_a - x_g|, |y_a - y_g|\}$
   Answer:

4.2 A* Search (40 pts)

In this section you will be implementing A* search with different distance heuristics. Your solver will be applied to the 8 puzzle problem described in section 3.2 in the text.

You are required to implement A* in python 3.6 using the anaconda distribution. To setup the anaconda environment follow the instructions for your platform at `https://www.anaconda.com/download/`.

Create an environment for this class with the following command:

conda create –name cs383 python=3.6

Activate the environment by:

source activate cs383

Install useful packages for this course:

conda install numpy matplotlib jupyter

When we test your code we will use the environment that is setup this way. So if you install and use other packages your code will not run. Also if you add plotting to your code, it will throw an error when run on the Gradescope servers. So make sure the plotting code is not executed when you submit your code (you do not need to plot for this assignment). If your code errors because you didn't test it on an anaconda environment setup this way, or because there is some plotting code, we will not give points back. You need to make sure your code runs correctly before submitting. If you have a question please post on the forums.

When we provide template code it is assumed that you will not edit the template except to fill in the necessary parts of the functions. Do not change function definitions and only return the specified type of objects. You can however create your own functions to call inside of the functions we provide. We will test your code using both unit tests and full runs of your code. To do this we will only use the functions defined in the templates.

4.2.1 Implementation (30 pts)

Implement A* with the following heuristics in Python.

1. Manhattan distance as described in section 3.6 in the book.
2. RowCol - the number of tiles out of the correct row position plus the number of tiles out of the correct column position.
3. Misplaced Tiles - the number of tiles out of place.
4. Null Heuristics - heuristics is always 0 (return 0).

As an additional check, make sure your implementation returns a solution with 26 steps for the start state given in Figure 3.28 of the text when using the Manhattan distance.

4.2.2 Performance (10 pts)

For each of the heuristic methods above run 100 random start states (use the `Puzzle.shuffle` method). For each trial compute the number of nodes expanded. Compare the mean and standard deviation of the number of nodes expanded for each heuristic method. Report the results in the table below and give a few sentences explaining the relative performance of each method. For the null heuristic you only have to run it 5 random trials because it will take a long time.

A* Nodes Expanded		
Heuristic	Mean	Std
Manhattan	N/A	N/A
RowCol	N/A	N/A
Misplaced Tiles	N/A	N/A
Null	N/A	N/A

Table 1: Number of nodes expanded during A* search on the 8-Puzzle problem

Answer: