Computer Science 2542
St. George Campus

Monday, October 3, 2016
University of Toronto

Homework Assignment #1: Search
Due: Monday, October 17, 2016 by 11:59 PM

Silent Policy: *A silent policy will take effect 24 hours before this assignment is due, i.e. no question about this assignment will be answered, whether it is asked on the discussion board, via email or in person.*

Late Policy: 10% per day after the use of 2 grace days.
(Each student has 2 grace days for the entire term that can be used to extend the deadline on this assignment, the project proposal, or the final project report without incurring a late penalty.)

Total Marks: This assignment is worth a total of 110 marks, which represents 20% of the course grade.

Handing in this Assignment

What to hand in on paper: Nothing.

What to hand in electronically: You are only required to submit a report that contains your answers to the various questions. This should be sent by email to `csc2542profs@cs.toronto.edu`. Please send your report as a **pdf** file, if you can. You do not need to email us your code. You can submit a new version of any file at any time, though the lateness penalty applies if you submit after the deadline. For the purposes of determining the lateness penalty, the submission time is considered to be the time of your latest submission.
Clarification Page: Important corrections (hopefully few or none) and clarifications to the assignment will be posted on the Assignment 1 Clarification page:
`http://www.cdf.toronto.edu/~csc2542h/fall/A1/a1_faq.html`.
It is also linked to the A1 webpage. *You are responsible for monitoring the A1 Clarification page.*
Help Sessions: We will be available following class on Thursday, October 6 and Thursday, October 13 to answer questions about the assignment.
Questions: Questions about the assignment should be asked on Piazza:
`https://piazza.com/class#fall2016/csc2542`.
If you have a question of a personal nature, please email us at `csc2542profs@cs.toronto.edu`, placing 2542 and A1 in the subject line of your message.

Introduction

The goal of this assignment is to give you experience running and experimenting with heuristic search code. There are three main components. The first is on tiebreaking in A*, the second is on tiebreaking in WA* and GBFS, and the third is on re-expansions in WA* and GBFS.

For this assignment, you are provided with C++ code for generic best-first search and A*, as well as for code for working with the map pathfinding and the 3×4 sliding tile puzzle domains. You are strongly encourage to use this code, though you are free to complete the experiments in whatever language you so choose. If you choose to use your own code, you will need to use the following input files, which have been provided (see the framework documentation for information on the format of these files):

- "starcraft_bgh.map" is a file representation of a map to run experiments on.
- "starcraft_bgh.probs" is a file containing a list of pathfinding problems for the map given by "starcraft_bgh.map."
- "3x4_puzzle.probs" is a list of start states for experiments on the 3×4 sliding tile puzzle domain.

All sections have both an experimental and theoretical component. For the theoretical component, feel free to use any of the theorems or lemmas we have already discussed in class.

Finally, note that different authors sometimes define a count of the number of node expansions slightly differently. For this assignment, we will count the number of node expansions as the number of times a node is taken off the open list, or equivalently, the number of calls to the goal test function that are performed.

1 Tiebreaking in A* [30 marks]

The first part of this assignment is on the importance of tiebreaking in A*. By default, A* compares nodes according to the evaluation $f(n) = g(n) + h(n)$, called the **f-value**, and no other criteria. Ties are then dealt with arbitrarily (usually just by their ordering in the comparison). In the rest of this document, we refer to this as the **Default** tiebreaking rule. For the following experiments, we also define the following two different methods, each of which you will have to implement:

- If there is a tie in the f-value between two nodes, the node with the lower value of $g(n)$ (*i.e.* the **g-cost**) is considered "best. We call this tiebreaking rule, **Low-G** tiebreaking.
- If there is a tie in the f-cost between two nodes, the node with the higher g-cost is considered best. We call this tiebreaking rule, **High-G** tiebreaking.

You are now required to complete the following tasks:

(a) **[5 mark]** Create an empty grid map of size 20×20. This means there are no obstacles. Set the map to be 4-connected, which means that in any location, the agent can only move in the 4 cardinal directions (i.e., north, east, south, and west) provided there is no obstacle in the way or they are prevented from moving in a given direction because they are at the boundary of the map. Consider the task of finding a path from location (1,10) and to location (11, 10). Run A* with each of the three different tiebreaking rules on this problem (Default, Low-G, and High-G). In at most two sentences, describe how the different tiebreaking rules compare in terms of nodes expanded.

(b) **[5 mark]** Run the same experiment as in 1(a), except this time set the goal state to location (6, 15). In at most three sentences, describe how the results compare to those seen in the experiment in 1(a).

(c) **[5 mark]** Notice that because there are no obstacles the Manhattan distance heuristic provides perfect estimates on the empty 20×20 grid map that you constructed. Also notice that the optimal solution cost to the tasks in questions 1(a) and 1(b) are the same. In no more than four sentences, explain why the relative performance of Low-G and High-G tiebreaking differs between these two problems, and what these experiments suggest about the best way to do tiebreaking when using A*.

(d) **[5 mark]** Run A* with each of the three different tiebreaking rules on the 100 pathfinding problems given in file "starcraft_bgh.probs" on the map given in file "starcraft_bgh.map". Make a table that shows the average number of nodes expanded when using each rule, the median number of nodes expanded, and the percentage of problems for which each tiebreaking rule is the best. In at most two sentences, describe if the results are consistent with the conclusions you made in question 1(c).

(e) **[10 marks]** Let α and β be two instances of A* that are identical aside from their tiebreaking policy, and both use the same consistent heuristic. Prove that if some node n is expanded by α before a goal is found, while β does not expand n before a goal is found, then $f(n) = C^*$.

2 Tiebreaking in WA* and GBFS [35 marks]

The second part of this assignment is on the importance of tiebreaking in WA* and GBFS. If you are using the provided code, this means you must extend the given best-first search code to create instances of WA* and GBFS.

(a) **[5 mark]** Repeat the experiments in questions 1(a) and 1(b) with WA* (use weights 2 and 5) and GBFS. In no more than 5 sentences, explain if the same trends hold as with A*, and why or why not.

(b) **[10 marks]** Consider the set of 100 3×4 sliding tile puzzle problems given. Run WA* using weights 1, 2, 5, 10, and 100, and GBFS, each using the three different tiebreaking strategies to solve these problems. Make a table that shows the average number of node expansions, the median number of node expansions, and the percentage of problems for which each tiebreaking rule runs the fastest (per weight). Make a second table that shows the average solution cost, the median solution cost, and the percentage of problems for which each tiebreaking rule finds the best solution. In at most 4 sentences, describe what these experiments suggest is the best way to do tiebreaking in WA* and GBFS.

(c) **[10 marks]** Even after applying the High-G and Low-G tiebreaking rules, there may still be ties. Add to your implementation the ability to break these remaining ties randomly. Take the problem from the 15-puzzle test set with the largest optimal solution cost, and solve it 100 times each with WA* with weights 1, 2, 5, 10, and GBFS. Use the best tiebreaking strategy seen in the above experiments for each algorithm. Show the average, median, and variance in terms of number of node expansions for each algorithm and weight. What is the general trend?

(d) **[10 marks]** Let $T(x,y)$ be a boolean function that returns true if node x is less than or equal to y and false otherwise (i.e., T is a tiebreaking rule). Consider any problem with a finite number of states. Let α be an instance of WA* that uses T to tiebreak, and let β be an instance of GBFS that uses the same heuristic as α, Low-G tiebreaking, and T to resolve any remaining ties. Prove that on any problem with a finite state-space, there exists a constant W such that if α uses a weight $w \geq W$, then α and β will expand the exact same set of nodes in the exact same order until a solution is found.

3 Re-expansions in WA* and GBFS [45 marks]

The third part of the assignment is on the impact of re-expansions in WA* and GBFS. For this part of the assignment, you will have to modify the code, so that it has the option to not reopen nodes when a lower-cost path is found to a state. For clarity, we refer to a WA* or GBFS that does reopen nodes as rWA* and rGBFS, respectively, and a WA* or GBFS that does not reopen nodes as nrWA* and nrGBFS, respectively.

(a) **[10 marks]** Run nrGBFS and nrWA* with weights 2, 5, and 10, on the 100 15-puzzle problems and create a table that shows the average and median number of node expansions, and the average and median solution costs. For each algorithm, use the best tiebreaking rule as seen in 2(b). In at most three sentences, describe how the results compare to those seen in question 2(b).

(b) **[10 marks]** Using the results from the runs in 3(a) and 2(b), make a table that shows, for each weight, the percentage of problems for which rWA* expanded fewer nodes than nrWA* (as well as vice versa and ties), and the percentage of problems for which rWA* found solutions of lower cost than nrWA* (as well as vice versa and ties). Construct a similar table for rGBFS and nrGBFS.

(c) **[5 mark]** Run rWA* with weights, 2, 5, and 10 on the 100 pathfinding problems given for map "starcraft_bgh.map". Make a table that shows the average number of nodes expanded (including re-expansions), the average number of unique state expansions (i.e., the average number of states expanded at least once), the median number of node expansions, and the median number of unique state expansions.

(d) **[5 mark]** Run nrGBFS and nrWA* with weights 2, 5, and 10 on the 100 pathfinding problems given for map"starcraft_bgh.map". Make a table that shows the average number of nodes expanded and the median number of node expansions. How do the results compare with those seen in question 3(c)?

(e) **[15 marks]** Prove that if the heuristic nrWA* using weight $w >= 1$ is using a consistent heuristic, then any solution it finds will have a cost of no more than $w \cdot C^*$. For this proof, you may use the following lemma:

Lemma 3.1. *Let $P = [n_0, ..., n_k]$ be the optimal solution path for a given problem on which we are running nrWA*. Then at any time prior to the expansion of a goal node, there will be some node n_i from P such that n_i is in OPEN.*