CS 2710/ISSP 2160 Fall 2015

Assignment 1: Problem-Solving Search

This must be your own individual work.

1 Introduction

After starting with a couple short-answer questions, this assignment will give you experience with heuristic search algorithms by comparing alternative search strategies and experimenting with heuristic evaluation functions.

Your work will be graded for correctness, completeness, clarity, and the logic of your presentation.

The submission instructions will be available on the TA's website. **Please be sure to follow the instructions.** A **TA Grief penalty** will be assessed for failing to follow them.

Please submit by 11:59pm on the due date to avoid a late penalty.

Please do this work on your own. Note that we have saved solutions to similar assignments from previous semesters.

2 Short-Answer Questions

Begin your report with answers to the following questions.

- Give a PEAS description (Section 2.3.1 and Figure 2.5) of each of the following task environments. In addition, give its characteristics in terms of the properties described in Section 2.3.2.

  Q1 3% Playing checkers.

  Q2 3% Shopping for computers on the Internet.

  Q3 3% Baking a cake.

3 Problem Implementation and Analysis

You should use searchP*.py and utils.py from the course web page.

The features of Python used in searchP*.py should be sufficient for your project. If you don't know Python, go through the tutorial (the first for Python 2* and the second for Python 3*):

`http://docs.python.org/2/tutorial/`
`http://docs.python.org/3/tutorial/`

3.1 Problem Implementation

On his website, Luca will specify the exact I/O interface your code should use. He will also post a test that your code must pass before you may turn it in. The test will check the correctness of your implementation on Q4-Q8. He will run your code on different inputs when he grades your assignment. Your code must produce the expected output exactly, so that this check may be done automatically.
For the 8-Puzzle

Q4 (8%) Rewrite goalp, successor, and edgecost to implement the 8-puzzle. (The board we consider to be the goal state is shown at the end of this document). Also, define 4 heuristic evaluation functions for the 8-puzzle:

Q5 (3%) Number of tiles out of place

Q6 (3%) Manhattan distance: This is the sum of the (horizontal and vertical) distances of each tile from its goal position.

Q7 (20%) A heuristic based on a precomputed pattern database. Your function should be called *hPatternDatabase*, and your comments at the top of the function should clearly and completely specify what the function computes. You do not need to use a database system; just create an appropriate data structure. The book says to construct the database by performing a backward search from the goal to each subproblem configuration. However, you can simply use one of the optimal algorithms already implemented in searchP*.py – since the algorithm is optimal, you will know that the g-val of the found goal is the g-val of a least-cost path to the goal from that subproblem configuration.

(Thinking ahead to the Midterm Exam, think about which algorithms you would use in different situations. Breadthfirst, iterative deepening search, uniform-cost search, or a heuristic algorithm? You do not need to answer these questions here.)

Q8 (1%) The MAX of the other three heuristics.

3.2 Statistics Gathering

Modify the code and add global variables, *maxQlen* and *numGenerated* that record the maximum fringe (queue) size and the total number of nodes generated during search, respectively. Add appropriate print statements to gather the statistics of *maxQlen* and *numGenerated*. You will need to report and use these statistics to analyze the performance of different search algorithms.

3.3 Report

Run treesearch (not graphsearch) versions of depth-first-search, breadth-first-search, uniform-cost-search, iterative-deepening-search, best-first-search, Astar search, and IDAstar search. Run them using each of the heuristics you defined. Run them on multiple start states. Compute the statistics mentioned above (e.g., *maxQlen*).

Your report should compare the various search algorithms and heuristics, supporting your arguments with statistics and observations from your runs. The statistics include (1) *maxQlen* and *numGenerated* mentioned above, (2) information about whether a goal was found, and (3) information about the quality of that goal (is it optimal?). In your report, you may report raw counts and averages of *maxQlen* and *numGenerated*. Specific issues to address are:

Q8 (11%): What are the properties of the problem? That is, how big is the search space? what is the average branching factor?

Q9 (15%): What are the strengths and weaknesses of each algorithm for this problem, and why.

Q10 (15%): For each of your heuristics, is it admissible? Please explain why it is or is not. Please provide evidence from your experiments that the heuristic is or is not admissible.

Q11 (15%): For each pair of heuristics, is one is more informed than the other? Please explain your answer. If one is more informed than the other, show the effect of this (i.e., give the relevant statistic(s) showing the effect of one being more informed than another).

Please use common sense in managing your computational processes. We don't want the class to drag the computers down too much.

Not all initial states for the 8-puzzle are solvable; that is, there are initial states such that no path exists from it to a goal state.

Here are some sample solvable initial states for the 8-puzzle:

```
Goal:        Easy:        Medium:        Hard:        Worst:
```

```
1 2 3       1 3 4       2 8 1       2 8 1       5 6 7
8   4       8 6 2         4 3       4 6 3       4   8
7 6 5       7   5       7 6 5         7 5       3 2 1
```