15-381 Spring 2007 Assignment 1: Search

Out: January 24th, 2007
Due: February 6th, 1:30pm Tuesday
Questions to Rebecca Hutchinson (rah@cs.cmu.edu)

The written portion of this assignment must be turned in at the beginning of class at 1:30pm on February 6th. Type or write neatly; illegible submissions will not receive credit. Write your name and andrew id clearly at the top of your assignment. If you do not write your andrew id on your assignment, you will lose 5 points.

The code portion of this assignment must be submitted electronically by 1:30pm on February 6th. To submit your code, please copy all of the necessary files to the following directory:

/afs/andrew.cmu.edu/course/15/381/hw1_submit_directory/yourandrewid

replacing yourandrewid with your Andrew ID. The TAs grading this assignment would prefer that you use C/C++, Java, or Matlab. If you would like to use a different programming language, please check with us first. All code will be tested on a Linux system, we will not accept Windows binaries. No matter what language you use, you must ensure that the code compiles and runs in the afs submission directory. Clearly document your program.

Late Policy. Both your written work and code are due at 1:30pm on 2/6. Submitting your work late will affect its score as follows:

- If you submit it after 1:30pm on 2/6 but before 1:30pm on 2/7, it will receive 90% of its score.
- If you submit it after 1:30pm on 2/7 but before 1:30pm on 2/8, it will receive 50% of its score.
- If you submit it after 1:30pm on 2/8, it will receive no score.

Collaboration Policy. You are to complete this assignment individually. However, you are encouraged to discuss the general algorithms and ideas in the class in order to help each other answer homework questions. You are also welcome to give each other examples that are not on the assignment in order to demonstrate how to solve problems. But we require you to:

- not explicitly tell each other the answers
- not to copy answers
- not to allow your answers to be copied

In those cases where you work with one or more other people on the general discussion of the assignment and surrounding topics, we ask that you specifically record on the assigment the names of the people you were in discussion with (or "none" if you did not talk with anyone else). This is worth five points: for each problem, you solution should either contained the names of people you talked to about it, or "none." If you do not give references for each problem, you will lose five points. This will help resolve the situtaion where a mistake in general discussion led to a replicated weird error among multiple solutions. This policy has been established in order to be fair to everyone in the class. We have a grading policy of watching for cheating and we will follow up if it is detected.

1. The missionaries and cannibals problem is as follows. Three missionaries and three cannibals are on one side of a river, along with a boat. The boat can hold one or two people (and obviously cannot be paddled to the other side of the river with zero people in it). The goal is to get everyone to the other side, without ever leaving a group of missionaries outnumbered by cannibals. Your task is to formulate this as a search problem.

   (a) Define a state representation.

   (b) Give the initial and goal states in this representation.

   (c) Define the successor function in this representation.

   (d) What is the cost function in your successor fuction?

   (e) What is the total number of reachable states? Justify your answer.

2. Consider a state space where the start state is number 1 and the successor function for state n returns two states, numbers $2n$ and $2n + 1$.

   (a) Draw the portion of the state space for states 1 to 15.

   (b) Suppose the goal state is 13. List the order in which nodes will be visited for breadth first search, depth-limited search with limit 3, and iterative deepening search.

   (c) Would bidirectional search be appropriate for this problem? Why or why not?

3. Give a description of a search space in which greedy best-first search performs worse than breadth-first search. How many nodes are visited by each strategy in your domain?

4. Trace the operation of the A* search algorithm applied to the problem of getting to Bucharest from Oradea, using the map in the text (Figure 3.2). Use the straight-line-distance heuristic values given in Figure 4.1.

5. What algorithms are the following special cases equivalent to? EXPLAIN YOUR ANSWERS.

   (a) Local beam search with $k = 1$.

   (b) Local beam search with one initial state and no limit on the number of states retained.

   (c) Simulated annealing with $T = 0$ at all times (an omitting the termination test).

   (d) Genetic algorithm with population size $N = 1$.

6. Column Jump is a game played on a grid with balls of different colors. The inital board has some configuration of colored balls with at least one empty space. The objective is to remove all but one ball from the board. Balls are removed when they are jumped according to the following rules. If a ball of one color jumps over a different colored ball to an empty space, the jumped ball is removed from the board. Additionally, if multiple balls of the same color are in a line, they can be jumped and removed together (by a different colored ball, provided that an empty space is on the other side of the line). You can play the game at http://www.2flashgames.com/f/f-354.htm to get a sense of the rules. Be sure to play in Remover mode instead of Color Life mode.

   In this problem, we will consider a version of this game, and you will implement search algorithms to solve it. Our version can use grids of various sizes with different numbers of colors. You will be provided with several example input and output files for testing your implementation. The input format will be a text file. The first line of the file is the size of the (square) grid. The second line is the number of colors. This is followed by characters representing the colors in each location of the grid, with 0 representing an empty space. For example, an input file might look like this:

   ```
   4
   3
   1 2 2 1
   2 1 3 2
   3 1 3 2
   0 2 3 0
   ```

Your output should contain the series of moves required to solve the puzzle. There should be one move on each line, where a move is represented by two locations on the grid with (row,column) numbers. The grid locations follow this convention:

```
(1,1) (1,2) (1,3) (1,4)
(2,1) (2,2) (2,3) (2,4)
(3,1) (3,2) (3,3) (3,4)
(4,1) (4,2) (4,3) (4,4)
```

So, the output file for the input given above looks like this:

```
(2,1) (4,1)
(1,4) (4,4)
(1,1) (1,4)
(4,2) (1,2)
(4,4) (4,2)
(4,1) (4,3)
(4,3) (1,3)
(1,4) (1,1)
```

(a) How many different possible game states are there for the 7x7 version of this puzzle with 6 colors?

(b) Define a state representation.

(c) Give two admissible heuristics for this problem.

(d) Implement depth-first search to solve the Column Jump puzzle.

(e) Implement A* search to solve the Column Jump puzzle. Describe the heuristic you are using for A*.

(f) Compare the performance of depth-first search and A* search on the examples provided (testExample1.txt, testExample2.txt, testExample3.txt) in terms of running time and solution length.

(g) How would you expect the performance of iterative-deepening search to compare to depth-first search in this domain?

(h) Would bidirectional search be useful in this domain? Why or why not?