Homework Assignment 1: Search Algorithms

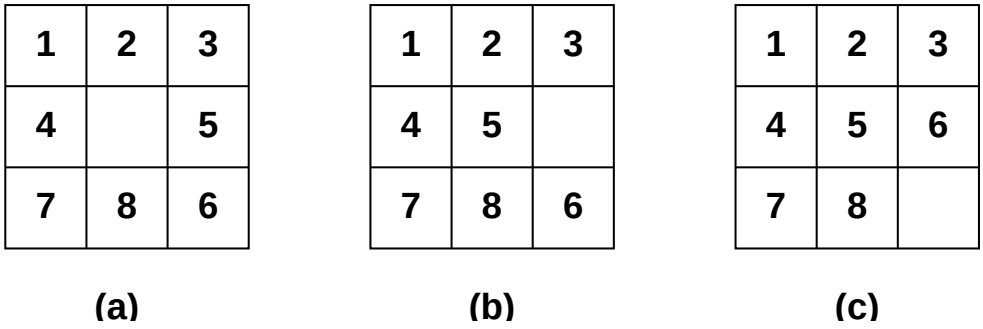
CS486/686 – Fall 2008
Instructor: Pascal Poupart

Out: Sept 16, 2008
Due: Oct 2, 2008 (no late assignment accepted)

Be sure to include your name and student number with your assignment.

1 Informed Search

Consider the 8-puzzle, which is a simple (one-person) game that we discussed briefly in class. In this game, tiles numbered 1 through 8 are moved on a 3-by-3 grid. Any tile adjacent to the blank position can be moved into the blank position. By moving tiles in sequence we attempt to reach the goal configuration. For example, in the figure below, we see three game configurations: the configuration (b) can be reached from configuration (a) by sliding tile 5 to the left; configuration (c) can be reached from configuration (b) by sliding tile 6 up. Configuration (c) is the *goal configuration*. The objective of the game is to reach the goal configuration from some starting configuration with as few moves as possible. Note that not all starting configurations can reach the goal.



1. **[20 pts]** Consider the following two heuristic functions that we discussed in class:

   - *Misplaced tile heuristic:* number of tiles (exluding the blank) that are misplaced (with respect to the goal configuration)
   - *Manhattan distance heuristic:* total Manhattan distance of all the tiles (excluding the blank). That is, for each tile, the Manhattan distance is the sum of the horizontal and vertical distances between its current position and the desired position in the goal configuration.

   Suppose that you use the above heuristics in A* search to solve the 8-puzzle game.

(a) Which heuristic do you expect to perform best? You do not need to implement the heuristics with A*. Based on the properties of each heuristic, just explain which one you expect to perform best.

(b) We have seen in class that those heuristics are *admissible*. Are they also *consistent*? Give a proof or a counter example for each heuristic.

2. **[20 pts]** IDA* combines iterative deepening with A* search. Answer the following questions regarding the expected performance of IDA* (you do not need to implement IDA*).

   (a) What is the time and space complexity of IDA*?

   (b) Is IDA* complete? Explain briefly.

   (c) Is IDA* optimal? Explain briefly.

2 Constraint Satisfaction

Using the language of your choice, you will implement a program that solves Sudoku puzzles using backtracking search. In a Sudoku puzzle, the goal is to fill a 9-by-9 grid so that each column, each row, and each of the nine 3-by-3 boxes (also called blocks or regions) contains the digits from 1 to 9, only one time each. In other words, a solved Sudoku puzzle has each digit from 1 to 9, exactly once in each row, once in each column and once in each of the nine 3-by-3 boxes. An example is shown below. The grid on the left is a starting configuration, and the grid on the right is the solution. You may not change any of the digits that are in the start configuration. A good source for information about Sudoku puzzles is http://www.websudoku.com.

	3						1	
				4			6	
4		8		1	5			3
							8	4
1			5		4			2
5	9							
9			2	6		1		7
	1			5				
	2						3	

(a) Starting configuration

6	3	9	7	8	2	4	1	5
2	5	1	9	4	3	7	6	8
4	7	8	6	1	5	9	2	3
3	6	2	1	7	9	5	8	4
1	8	7	5	3	4	6	9	2
5	9	4	8	2	6	3	7	1
9	4	3	2	6	8	1	5	7
8	1	6	3	5	7	2	4	9
7	2	5	4	9	1	8	3	6

(b) Solved puzzle

1. **[15 pts]** How did you formulate the Sudoku puzzle as a constraint satisfaction problem (CSP)?

   What to hand in: a detailed description of all variables, domains and constraints that are sufficient to model Sudoku as a CSP.

2. **[45 pts]**Implement a Sudoku solver (for 9-by-9 puzzles only) using the CSP formulation that you came up with. More precisely, implement the backtracking seach algorithm, the three CSP heuristics discussed in class (e.g., most constrained variable, most constraining variable and least constraining value) as well as forward checking. For more details about these algorithms, review the lecture slides and read pages 143-145 of the textbook.

   You should implement the following combinations:

   - B: basic backtracking search (no heuristics, no forward checking)
   - B+FC: backtracking search with forward checking
   - B+H: backtracking search with the 3 heuristics
   - B+FC+H: backtracking search with the 3 heuristics and forward checking

A set of test puzzles (easy, medium and difficult puzzles) will be posted to the class website within a week. Run each of the above combinations on the test puzzles and report the following information for each of the categories (easy, medium, difficult):

- average time to complete the puzzles
- average number of nodes searched

You should hand in a table like this one:

	B	B+FC	B+H	B+FC+H
Easy	time, nodes	time, nodes	time, nodes	time, nodes
Medium	time, nodes	time, nodes	time, nodes	time, nodes
Difficult	time, nodes	time, nodes	time, nodes	time, nodes

If your algorithm does not finish in a reasonable amount of time (e.g., 10 minutes), record this in the table.

What to hand in:

(a) Your code

(b) Table showing your algorithm's performance on the 3 classes of test problems

(c) Brief dicussion to explain why your algorithm performed the way it did (max one page).