Solving the 8 Puzzle in a Minimum Number of Moves: An Application of the A* Algorithm

Daniel R. Kunkle

Computer Science Dept.
College of Computing and Information Sciences
Rochester Institute of Technology
Rochester, NY, 14623
drk4633@rit.edu
http://www.rit.edu/~drk4633/

Introduction to Artificial Intelligence
October 8, 2001

Abstract

The 8 Puzzle is a simple game, but one with a state space large enough to warrant the use of heuristic search, as opposed to an exhaustive or blind search. The A* algorithm is applied, guaranteeing that the best solution (that with the least number of moves) will be found. Heuristics are examined to allow the algorithm to find the optimal solution while examining as few states as possible (maximizing the informedness of the heuristic).

1 USING THE PROGRAM

This solution is implemented using Java. It includes four classes, EightPuzzle, BFSearch, Board and Pos (provided as Java source code). Also included is a test file, tests.txt, which will provide a number of test boards as input.

EightPuzzle contains the main function. It begins by reading two strings, on separate lines, from standard input. They are both of the form ### ### ###, where # is either a number from 1 to 8 or the character b.

If the problem is solvable, the sequence of moves (moves representing the direction of movement of the b character) are printed, as well as the number of board states examined by the algorithm to arrive at the solution.

After printing the results the program checks for more input and terminates when no more input is provided.

Once the code is compiled the program can be tested by using the command

```
java EightPuzzle < tests.txt
```

Where `tests.txt` can be the provided test input file or a custom test file. Input can also be provided by keyboard.

2 PROGRAM STRUCTURE

EightPuzzle is a simple main class that retrieves the starting state of the board and the goal state of the board from standard input and creates a Board object and BFSearch object to solve the problem.

The Pos class is also a simple class, meant only for holding ordered pairs corresponding to the row and column positions of the board and performing simple operations on them. The real work of the algorithm occurs in the other two classes.

2.1 BFSearch CLASS

The act of searching a state space and the state space itself are separated in this implementation, allowing an A* algorithm to be easily applied to any problem. BFSearch is the class responsible for performing a best-first search of the state space.

A Board is passed to the BFSearch function solve(), which then uses functions of Board to perform the search. A Board object is required to have five functions for use by BFSearch: equals(), h(), f(), children() and print(). equals() is a boolean function returning True if the board is equal to another. h() returns the heuristic value of the board. f() returns the sum of the h value and the number of moves (tree depth). children() returns an ArrayList of all of the children of the board reachable by one legal move (with out the inverse of the last move).

The search algorithm requires that the heuristic value passed by the Board is zero if, and only if, the board is in the goal state.

The search algorithm maintains an open and closed list and uses the functions provided by the Board object to perform the following sequence of actions (which define a best-first search):

1. Add the original board to the open list.
2. Check the first Board on the open list, if it is a goal state, print the board and halt operation, otherwise go to step 3.
3. Retrieve a list of children from the first Board on the open list.
4. Move the first Board on the open list to the closed list.
5. If any of the children appear in the closed or open list, remove them.
6. Add the list of remaining children to the open list, which is sorted by the f() value of each board.
7. Go to step two.

This algorithm is an A* algorithm, and therefore guaranteed admissible, if for any value x returned by h(), x is less than or equal to the actual number of moves left to reach the goal state.

As long as the Board object provides the functions specified by BFSearch any problem domain can be searched by this class. The domain specific details are handled by the Board class.

2.2 Board CLASS

The board is responsible for keeping tack of a particular board state, the goal state, and the moves required to reach the goal state.

Each board state (current and goal) are stored in 2D arrays. For this problem the size of those arrays are 3x3, but can easily support boards of other sizes. The list of moves is a linked list.

The Board class provides all of the functions required by the search class. Two Board objects are equal if all tiles in the state of the board are in the same positions. Two equal boards may have different numbers of moves to reach that state. The search algorithm will only keep the board with the least number of moves.

The heuristic function used by the Board to evaluate its state depends on the constant `H_TYPE`. If `H_TYPE = 0`, the h() function always returns 0 (hence turning the BFSearch into a Breadth-First search). `H_TYPE = 1` corresponds to the Manhattan Distance heuristic. `H_TYPE = 2` corresponds to the Manhattan Distance plus a tile reversal penalty heuristic. (Heuristics will be described in more detail in the next section).

The f() function will return the sum of h() and the number of moves so far.

The children() function will return all legal children boards. Each legal move of the blank (Up, Down, Left, Right) is generated, given the current position of the blank. The move that would negate the last move is excluded from the children.

3 HEURISITICS

The most sophisticated heuristic supported by this program is the Manhattan Distance plus a tile reversal penalty.

The Manhattan Distance of one tile is the number of moves that would be required to move that tile to its goal location if it could move over any of the other tiles. (Called the Manhattan Distance because it looks much like moving along city blocks).

The tile reversal penalty will add one penalty move for every direct reversal of tiles. So, the board:

```
213
8b4
765
```

would have a reversal penalty of 2, because 2 is reversed with 1 and 1 is reversed with two. Reversal penalties will always come in pairs.

For testing purposes, the Board can be set to use a subset of the heuristic measures, including none at all. A constant heuristic value of one if the board is not in the goal state and zero if it is in the goal state yields a Breadth-First search of the space (given, it is a very inefficient way to implement a Breadth-First search).

Each of these heuristics results in an admissible search. In other words, it always results in the optimal answer. However, more sophisticated heuristics are more informed, therefore searching fewer states. This can be seen in the results section, which gives the results of using different heuristic calculations.

4 RESULTS

The results listed on the following pages detail the performance of the A* algorithm implemented with three different heuristics. The first is not really a heurisitc at all, it simply returns 0 if the board is in the goal position and 1 otherwise, resulting in a Breadth-First search. The second uses only the Manhattan Distance heuristic. The third uses the Manhattan Distance plus a direct reversal penalty.

Each has listed for them the starting board state, moves taken to solve the board, solution board state, and the number of intermediate board states examined to arrive at the solution.

4.1 Breadth-First Search

The breadth-first search finds an optimal solution (one with the fewest number of possible moves), but does so only after examining a great number of intermediate states. The last test condition in the set (a full reversal of tiles) was unable to be solved by this search after 100,000 state examinations. Even this unsuccessful search took many hours.

4.2 Manhattan Distance Heuristic Search

This heuristic improves greatly over the blind, breadth-first search. For the tests run (that finished in reasonable time by the blind search), the Manhattan Distance Heuristic search examined approximately 25 times fewer states than did the blind search. This means that the initial heuristic is more informed by a factor of 25. For the hardest case presented, this search examined 10805. If the blind search were to examine 25 times this number of states it would search more than two-thirds of the entire space.

Even this very simple heuristic provides enormous efficiency gains over blind search.

4.3 Manhattan Distance plus Reversal Penalty Search

Additional improvements are made in informedness by adding a penalty for directly reversed tiles. This is due to the fact that reversed tiles are much more difficult to deal with because one must "go around" the other.

This heuristic provides an approximately 16% gain in informedness over Manhattan Distance alone when dealing with non-trivial cases. In the most difficult problem presented (the last case in for each in the output summaries), the number of states examined to find an optimal solution was reduced from 10804 to 9176, a gain of about 18%.

4.4 Results Summary

Although all of these search methods find an optimal solution they each take different amounts of time to do so. The more sophisticated (informed) the heuristic, the faster the results.

It is also interesting to note that they do not all find the same optimal solution. Even the same heuristic search given a problem and that problem's inverse will not always simply find the reverse of the moves made in the solution to the first problem (as can be seen in the third and fourth test cases presented to each search below).

RESULTS OF SIMPLE BREADTH-FIRST SEARCH

INITIAL BOARD
134
8b5
726

MOVES TO SOLUTION: D R U U L D
NUMBER OF MOVES: 6

GOAL BOARD
123
8b4
765

ANSWER WAS FOUND IN 69 STATE EXAMINATIONS

INITIAL BOARD
231
7b8
654

MOVES TO SOLUTION: R U L D R D L L U R U L D R
NUMBER OF MOVES: 14

GOAL BOARD
123
8b4
765

ANSWER WAS FOUND IN 3685 STATE EXAMINATIONS

INITIAL BOARD
231
8b4
765

MOVES TO SOLUTION: U L D R U R D L L U R D R U L D
NUMBER OF MOVES: 16

```
GOAL BOARD
123
8b4
765

ANSWER WAS FOUND IN 9137 STATE EXAMINATIONS
----------------------------------------

INITIAL BOARD
123
8b4
765

MOVES TO SOLUTION: U R D L U L D R R U L D L U R D
NUMBER OF MOVES: 16

GOAL BOARD
231
8b4
765

ANSWER WAS FOUND IN 9137 STATE EXAMINATIONS
----------------------------------------

INITIAL BOARD
283
1b4
765

MOVES TO SOLUTION: U L D R
NUMBER OF MOVES: 4

GOAL BOARD
123
8b4
765

ANSWER WAS FOUND IN 21 STATE EXAMINATIONS
----------------------------------------

INITIAL BOARD
```

876

1b5

234

NO ANSWER WAS FOUND WITHIN 100000 STATE EXAMINATIONS

RESULTS OF MANHATTAN DISTANCE SEARCH

INITIAL BOARD

134

8b5

726

MOVES TO SOLUTION: D R U U L D

NUMBER OF MOVES: 6

GOAL BOARD

123

8b4

765

ANSWER WAS FOUND IN 6 STATE EXAMINATIONS

INITIAL BOARD

231

7b8

654

MOVES TO SOLUTION: R U L D R D L L U R U L D R

NUMBER OF MOVES: 14

GOAL BOARD

123

8b4

765

ANSWER WAS FOUND IN 70 STATE EXAMINATIONS

INITIAL BOARD

231
8b4
765

MOVES TO SOLUTION: R U L L D R R U L D L U R R D L
NUMBER OF MOVES: 16

GOAL BOARD
123
8b4
765

ANSWER WAS FOUND IN 292 STATE EXAMINATIONS

INITIAL BOARD
123
8b4
765

MOVES TO SOLUTION: L U R R D L L U R D R U L L D R
NUMBER OF MOVES: 16

GOAL BOARD
231
8b4
765

ANSWER WAS FOUND IN 292 STATE EXAMINATIONS

INITIAL BOARD
283
1b4
765

MOVES TO SOLUTION: U L D R
NUMBER OF MOVES: 4

GOAL BOARD
123
8b4

765

ANSWER WAS FOUND IN 4 STATE EXAMINATIONS

INITIAL BOARD

876

1b5

234

MOVES TO SOLUTION: D L U R R D L U U R D D L U U L D D R U U L D R R U L D

NUMBER OF MOVES: 28

GOAL BOARD

123

8b4

765

ANSWER WAS FOUND IN 10804 STATE EXAMINATIONS

RESULTS OF MANHATTAN DISTANCE PLUS REVERSAL PENALTY SEARCH

INITIAL BOARD

134

8b5

726

MOVES TO SOLUTION: D R U U L D

NUMBER OF MOVES: 6

GOAL BOARD

123

8b4

765

ANSWER WAS FOUND IN 6 STATE EXAMINATIONS

INITIAL BOARD

231

7b8

654

MOVES TO SOLUTION: R U L D R D L L U R U L D R
NUMBER OF MOVES: 14

GOAL BOARD
123
8b4
765

ANSWER WAS FOUND IN 61 STATE EXAMINATIONS

INITIAL BOARD
231
8b4
765

MOVES TO SOLUTION: R U L L D R R U L D L U R R D L
NUMBER OF MOVES: 16

GOAL BOARD
123
8b4
765

ANSWER WAS FOUND IN 257 STATE EXAMINATIONS

INITIAL BOARD
123
8b4
765

MOVES TO SOLUTION: L U R R D L L U R D R U L L D R
NUMBER OF MOVES: 16

GOAL BOARD
231
8b4
765

```
ANSWER WAS FOUND IN 257 STATE EXAMINATIONS
-------------------------------------------

INITIAL BOARD
283
1b4
765

MOVES TO SOLUTION: U L D R
NUMBER OF MOVES: 4

GOAL BOARD
123
8b4
765

ANSWER WAS FOUND IN 4 STATE EXAMINATIONS
-------------------------------------------

INITIAL BOARD
876
1b5
234

MOVES TO SOLUTION: D L U R R D L U U R D D L U U L D D R U U L D R R U L D
NUMBER OF MOVES: 28

GOAL BOARD
123
8b4
765

ANSWER WAS FOUND IN 9176 STATE EXAMINATIONS
-------------------------------------------
```