

Solving the 8-puzzle problem with search

ICS 171, Summer 2000

Stephen D. Bay
August 10, 2000

1 Administrative Details

- Due: August 31, 2000 (in discussion)
- This project must be done individually. You may discuss the problem with other students but you must write the program by yourself. Copying code from other students, past projects in ICS, or any other source is prohibited.
- We will run your code through an automatic program to detect similarities between programs. It is very effective at detecting cheating.
- You may program in any language that is available on the ICS UNIX accounts including C, C++, JAVA, and Lisp.
- If you do not have an ICS UNIX account please contact the TAs immediately.

2 The Programming Problem

For this project, you will use search to solve the 8-puzzle program as described in Russell and Norvig (page 63). For our assignment, the goal state will be

1	2	3
4	5	6
7	8	

Implement the following search strategies to solve the 8-puzzle problem:

- Breadth First Search
- Depth Limited Search (choose an appropriate depth limit)
- Best First Search with the following heuristics
 - H1 = the number of misplaced tiles
 - H2 = sum of the Manhattan distances for misplaced tiles

- $H3 = 0$
- $H4 =$ a random number between 1 and 10
- A* Search (with above heuristics)
- (extra credit) IDA* Search (with above heuristics)

You will also have to write a random problem generator which will make 8 puzzle problems at random. Note that not all random configurations of puzzles are solvable.

3 Analysis

In class, we analyzed theoretically the performance of the various search algorithms. In this project, you will analyze the performance experimentally. Once you have your algorithms working correctly, generate at least 10 random problems and run the search algorithms on each one. For each algorithm record:

- the percentage of problems which were solvable by the algorithm
- the percentage of problems for which the algorithm found the optimal solution
- the average number of nodes expanded (include only runs where the algorithm found a solution)
- the maximum size of the fringe (queue) during search (include only runs where the algorithm found a solution)

From the statistics that you collected, answer the following questions:

- How did the uninformed search methods compare with the informed methods?
- How did the choice of heuristic affect the performance (search time and memory usage) of the informed search methods? How well did the random non-admissible heuristic perform?
- How did your experimental results reflect your theoretical understanding of the algorithms?

Limit the discussion to a maximum of 3 pages.

4 What To Hand In

You will hand in a report containing the following:

- A lab report describing your representation of the search problem, the design of your algorithms and your experimental write up.
- Your report need not be long, but it must be written in a clear and concise manner.
- Source code of your programs with clear comments. Your code does not have to be perfect, but it should be clear, correct and efficient.
- Printouts demonstrating that your program works correctly. Show one solution trace for each algorithm implemented.
- Documentation describing where your program is located and how to run it. You must make your code and executable available on your ICS account so that we can run the program. Remember to set your security permissions so that we can access your program. Check with a friend in the class to make sure they are set properly. If we cannot run your program you will lose marks.