1. Code Design

The whole programming is made up with three parts. First, the node stands for the 9 space in the matrix which has basic functions and param such as index number and movement.

The second part is two search method A* Seach and Bean Search. The last part is the import port file(EightPuzzle) and randomized states(The GoalPuzzle) and the test file called test1.txt.

Bascially, the EightPuzzle reads the text file or we generate a sovlable state to test two seach method.

2.Code Correctness

We can check previous method by using EightPuzzle's main method to read the text file called "test1.txt", you can change the sequence of number in the txt file and the output should be right.

The randomizeState, printState, and move method can be shown by the main method in GoalPuzzle. Tip; the n in randomizeState(n) is the moving steps from the goal state, you can randomly change it to test the move method and print method.

Two seach methods can be runned correctly if the input is reasonable. All details are in the comments of java files in ASearch. And the maxNode method is in both search method

3.Experiments

1. How does fraction of solvable puzzles from random initial states vary with the maxNodes limit?

If the maxNodes limit is too low, it may prevent the search algorithm from exploring enough of the search space to find a solution. On the other hand, if the maxNodes limit is too high, it may allow the search algorithm to explore too much of the search space and spend too much time searching, even for unsolvable puzzles. Therefore, it is possible that there exists an optimal value for the maxNodes limit that produces the highest fraction of solvable puzzles.

2. For A* search, which heuristic is better, i.e., generates lower number of nodes?

the Manhattan distance heuristic tends to be better than the misplaced tiles heuristic

3. How does the solution length (number of moves needed to reach the goal state) vary

across the three search methods?

A* search tends to produce the shortest solutions, as it is an optimal search algorithm that guarantees finding the shortest path to the goal state. Beam search may produce shorter solutions than iterative deepening search, but it is not guaranteed to find the shortest path.

4. For each of the three search methods, what fraction of your generated problems were

solvable?

It depends on the specific initial state of the puzzle and the search algorithm being used. But 6 cases of 10 were unsolvable.

4. Discussion The final part of your write up should discuss the following:

1. Based on your experiments, which search algorithm is better suited for this problem? Which one finds shorter solutions (solutions with less number of moves)? Which algorithm seems superior in terms of time and space?

A*search A*search Bean search

2. Discuss any other observations you made, such as the difficulty of implementing and testing each of these algorithms.

In most cases, it cannot be solvable if I random put the current state.