**Solving *Problem Name* by Searching**

1. **Team Members and Time Report:**

|  |  |  |  |
| --- | --- | --- | --- |
| **First Name** | **Last Name** | **Total Time** | **Contributions** |
| **Tod** | **Jones** |  |  |
|  |  |  |  |
|  |  |  |  |

1. **Problem Description:**

*[Write a paragraph to describe the problem.]*

1. **Problem Modeling:**

*[Model the problem into state-space.]*

1. **Implementation**

*[Describe the language you use to implement your solutions. Why do you choose this language? What data structure do you use to implement the state space?]*

1. **Uninformed Search Algorithm: *Name of the algorithm***
   1. **Algorithm Description**
   2. **Algorithm (pseudo code)**
   3. **Algorithm Properties**
   4. **Results**

*[List sample inputs and results from your program.]*

1. **Heuristic Search Algorithm 1: *Name of the algorithm***
   1. **Heuristic Description**
   2. **Algorithm (pseudo code)**
   3. **Algorithm Properties**
   4. **Results**

*[List sample inputs and results from your program.]*

1. **Heuristic Search Algorithm 2:  *Beam Search***
   1. **Heuristic Description Creates goal state of puzzle. Compares current state of puzzle to goal state. Assigns discrete value based on the distance of each piece. Then checks all possible moves from current state. First orders possible moves by number of times it has been visited and then least distance from solution. Then picks two best moves out of the two nodes and repeats until solution found.**
   2. **Algorithm (pseudo code)**

**Create root node(puzzle state)**

**Create two branch nodes equal to root node**

**while loop (left by checking heuristic)**

**put two branches on linked list of visited nodes**

**look through linked list, check and update number of times visited**

**get two next best nodes by times visited and least scrambled state-space**

* 1. **go back to top of loop**
  2. **Algorithm Properties**

**Stumbles over every local minimum for heuristic, that’s why checking previous links was necessary to force algorithm into new links. Heuristic can tell right away whether puzzle is solvable, if positive and negative distance are not equal, then puzzle has been randomized in a way that can’t be solved. Heuristic is a little better than a depth first search, but ultimately not that intelligent.**

* 1. **Results**

*[List sample inputs and results from your program.*

Due to my puzzle shuffler being taken offline and without enough time to reprogram and test it on a range of puzzles, I only had tested it on the preset puzzle. It took over 1600 steps to solve the one puzzle, and while that may be not typical, the algorithm certainly takes a long trial and error approach to finding a solution. Solution time took several seconds.

1. **Empirical Analysis**

*[Be creative here. Compare your experiment results in different ways. Use charts or tables to demonstrate the comparison.]*

1. **Conclusion**

*[What have you learned from this project? Did you encounter any issues? How did you solve them? What related topics would you like to explore more in the future?]*