CS 5400 – 1B
Puzzle Assignment 3 Bonus
Grant Broadwater
February 18th, 2018

Assignment 3 Bonus: Heuristic Comparison

This document is meant to discuss the design and effectiveness of multiple heuristics. The implementation of the three heuristics can be found in "*GreedyBestFirstSearch.cpp*" from lines 22 – 33. The results of each heuristic can be found in the directories `h1`, `h2`, and `h3` in the repository. The heuristics are as follows:

1. (quota – score) / quota

   This heuristic is designed to be both admissible and consistent, and is the main heuristic used for the assignment submission. This heuristic returns values in range (0, 1) for non-goal states. This heuristic works for greedy best first search. However, for A* this heuristic would result in uniform cost search as all heuristic values are less than the unit step cost. The resulting branching factor for each puzzle file are as follows:

   - Equation:
     - $N + 1 = 1 + (b^*)^1 + (b^*)^2 + \ldots + (b^*)^d$
   - Puzzle 1
     - N: 7
     - d: 2
     - b* = 2.192
   - Puzzle 2
     - N: 15
     - d: 11
     - b*: 1.0509
   - Puzzle 3
     - N: 20
     - d: 16
     - b*: 1.02577
   - Average b*: 1.42289

   An average b* of 1. 42289isn't bad, and it appears to get better as the puzzles go. This puzzle is also getting seriously penalized by puzzle 1 even though it only expanded 7 nodes. A more thorough test would most likely find this heuristic to have an even better b*.

2. 3.0 / score *(if score < 1, returns 3)*

   This heuristic is not admissible or consistent. However, it also would not result in uniform cost search when used with A*. This heuristic produces value in range (0, 3]. The three constant was used because 3 is the minimum score that could result from any swap. The resulting branching factor for each puzzle files are as follows.

   - Equation:
     - $N + 1 = 1 + (b^*)^1 + (b^*)^2 + \ldots + (b^*)^d$
   - Puzzle 1
     - N: 2
     - d: 1
     - b* = 2
   - Puzzle 2
     - N: 12
     - d: 11
     - b*: 1.01443
   - Puzzle 3
     - N: 16
     - d: 15
     - b*: 1.00802
   - Average b*: 1.3408

     An average b* of 1.3408 is even better than before, and is again punished for expanding only 2 nodes because of the solution depth at 1. It should be noted that both of those nodes had the same heuristic value of 3.

3. Always return 1.0

   This heuristic is meant to be intentionally naïve, but is technically admissible and consistent. Since all nodes always have the same heuristic value, the way the nodes are retrieved from the queue is undefined and implementation specific. However even with this being the case, a solution was found fairly quickly for each puzzle file instance. The resulting branching factor for each puzzle files are as follows.

   - Equation:
     - $N + 1 = 1 + (b^*)^1 + (b^*)^2 + \ldots + (b^*)^d$
   - Puzzle 1
     - N: 5
     - d: 1
     - b* = 5
   - Puzzle 2
     - N: 97
     - d: 11
     - b*: 1.34388
   - Puzzle 3
     - N: 206
     - d: 22
     - b*: 1.1684
   - Average b*: 2.504

   A 2.504 average b* is absolutely atrocious; however, this again is only because of a painfully high b* on puzzle 1. This search is essentially random search as it does not have any guidance on how nodes are removed from the frontier. But at least in this implementation, it still found solutions fairly quickly. However, it can be seen that these solutions are likely to not be optimal.