# Programming Assignment PA1 – Results

## Heuristics:

* H0
  + This is my default heuristic to simulate a regular BFS. This heuristic assumes that the initial state is already at the goal state and never overestimates the cost of movement.
  + h(n) will always be zero.
* H1
  + This was my first heuristic that would count the number of blocks that are not currently in their goal state (misplaced). It iterates through each stack of both the current and goal state and checks whether the current block is different, if so then it is misplaced. The problem with this heuristic after looking back was it doesn’t consider how far each block is from its goal.

A computer screen with white text

Description automatically generated

* H2
  + This heuristic estimates the cost based on how far each block is from its goal stack level. For example, if a block needs to go up or down to reach its goal state that would add to the cost. The function iterates through both the current and goal states and checks if the blocks are the same and the heuristic returns the difference between the total number of stacks in the goal state and the number of correct blocks. The only problem I found with this heuristic is that it does not account for the blocks horizontal position, this was one of my attempts at solving a problem like B19.

A computer screen with white text

Description automatically generated

* H3
  + This heuristic was like H1 in that it again focuses on finding misplaced blocks however, it only looks are the bottom layer of each stack. The heuristic first checks whether the stack is empty and if not, it will compare the bottom blocks of the current stack to the goal stack, if they are different then this counts as a misplaced block. This heuristic only targeted problems whose bottom layer was widely different from the goal state.

A computer screen with white text

Description automatically generated

* H4
  + This heuristic was my best because it creates a custom cost depending on a block’s vertical and horizontal movement. For this heuristic, I implemented the Manhattan distance for the blocks within the goal stack and a cost was added for moving the blocks not in their goal position. For each block, it checks whether it is in the goal stack if it is it, it will calculate the Manhattan distance between the positions of that block in its current stack to the goal stack. Cost is added for moving a block that is not in its goal stack, therefore the cost is the remaining distance for the block to get to the top of the stack.

A computer screen with white text

Description automatically generated

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Problem | Method | Plan Length | Iterations | Maximum Queue |
| A03 | A\* | 4 | 17 | 10 |
| A04 | A\* | 5 | 35 | 17 |
| A05 | A\* | 6 | 38 | 19 |
| A06 | A\* | 7 | 234 | 75 |
| A07 | A\* | 8 | 174 | 76 |
| A08 | A\* | 11 | 426 | 168 |
| A09 | A\* | 10 | 261 | 87 |
| A10 | A\* | 11 | 752 | 251 |
| A11 | A\* | 12 | 510 | 207 |
| B03 | A\* | 4 | 69 | 41 |
| B04 | A\* | 5 | 79 | 50 |
| B05 | A\* | 6 | 109 | 74 |
| B06 | A\* | 7 | 91 | 59 |
| B07 | A\* | 8 | 283 | 177 |
| B08 | A\* | 9 | 487 | 308 |
| B09 | A\* | 10 | 690 | 430 |
| B10 | A\* | 11 | 814 | 515 |
| B11 | A\* | 10 | 1084 | 651 |
| B12 | A\* | 15 | 34168 | 15887 |
| B13 | A\* | 15 | 1211 | 776 |
| B14 | A\* | 15 | 12099 | 7247 |
| B15 | A\* | 17 | 15556 | 9262 |
| B16 | A\* | 18 | 83579 | 43914 |
| B17 | A\* | 19 | 85659 | 42526 |
| B18 | A\* | 17 | 63891 | 31251 |
| B19 | A\* | 17 | 567573 | 215099 |
| B20 | A\* | 19 | 34294 | 19815 |