Description of my heuristic:

H0:

0 heuristic, basically BFS.

H1:

Loops through the initial and end state and adds one in the heuristic for every mismatch there is.

H2:

For each block in the current state:

a. If the block is not in its goal stack, increase the heuristic by 2, since I add a count for picking up the block from its current stack and another for placing it in the desired stack.

b. If the block is in its goal stack but not in the desired position within the stack, increase the heuristic by 1. Since the block needs to be moved to its correct position within its current stack.

Bottom Block Weighting:

Special consideration is given to the blocks at the bottom of the stacks. If the bottom block of a stack is not in its correct position, increase the heuristic by the number of blocks in that stack.

This is because misplacement of a bottom block affects all blocks above it. The number of blocks in that stack indicates how many blocks would need to be moved to correct the position of the bottom block.

EXAMPLE:  
 initial\_stacks = [

[],

['C', 'F'],

['A', 'D', 'E'],

['G', 'H', 'I', 'J'],

['B']

]

goal\_stacks = [

['A', ],

['C', 'F'],

['D', 'E'],

['G', 'H', 'I', 'J'],

['B']

]

This would give a 4 distance since A is a bottom block and is out of position. A is weighted by 2 since its a bottom block and the block above it are counted as mispositioned so another 2 is added. Thus the heuristic is 4. This incentivizes the blocks above to be taken off before the A is moved.

H3:

Pairwise distance heuristic:

For each block I calculate the total distance between its position and end.

Horizontal Distance: Number of stacks the block is away from its goal position.

Vertical Distance: Number of blocks above the block in its current stack + number of blocks above its target position in the goal stack.

Total Distance = Horizontal Distance + Vertical Distance

EXAMPLE:  
 initial\_stacks = [

['A', 'E'],

['C', 'F'],

['D'],

['G', 'H', 'I', 'J'],

['B']

]

goal\_stacks = [

['A', ],

['C', 'F'],

['D', 'E'],

['G', 'H', 'I', 'J'],

['B']

]

This would give a 2 distance since E is 2 away from its final position.

Table Summarizing:

| **Heuristic** | **Test Case** | **Solution Length** | **Number of Iterations** | **Maximum Queue Size** |
| --- | --- | --- | --- | --- |
| H2 | probA03 | 3 | 3 | 11 |
| H2 | probA04 | 4 | 9 | 23 |
| H2 | probA05 | 5 | 7 | 23 |
| H2 | probA06 | 6 | 68 | 110 |
| H2 | probA07 | 7 | 25 | 67 |
| H2 | probA08 | 8 | 109 | 224 |
| H2 | probA09 | 9 | 180 | 308 |
| H2 | probA10 | 10 | 615 | 916 |
| H2 | probA11 | 11 | 602 | 1039 |
| H2 | probB03 | 3 | 3 | 51 |
| H2 | probB04 | 4 | 4 | 73 |
| H2 | probB05 | 5 | 5 | 87 |
| H2 | probB06 | 6 | 6 | 85 |
| H2 | probB07 | 7 | 101 | 1184 |
| H2 | probB08 | 9 | 539 | 5515 |
| H2 | probB09 | 9 | 511 | 5226 |
| H2 | probB10 | 9 | 47 | 544 |
| H2 | probB11 | 9 | 58 | 698 |
| H3 | probB12 | 15 | 5333 | 63956 |
| H2 | probB13 | 13 | 59378 | 422940 |
| H3 | probB14 | 16 | 1915 | 21372 |
| H3 | probB15 | 18 | 3783 | 45882 |
| H2 | probB16 | FAILED | 100000 | 957975 |
| H3 | probB17 | 21 | 4111 | 53618 |
| H3 | probB18 | 14 | 774 | 9944 |
| H3 | probB19 | FAILED | 100000 | 649608 |
| H3 | probB20 | 22 | 78482 | 944030 |