Best Heuristic Description:

The functionality of my heuristic is rooted in comparing the current positions of blocks with their respective goal positions. If a state is far away from the goal, and misplaced blocks are in inconvenient locations, that state is considered more costly than a state that is closer to the goal with fewer misplaced blocks and misplaced blocks being in more convenient locations.

Therefore, states that are close to being solved get low heuristic scores since they have more blocks in the correct locations and remaining misplaced blocks are easy to move to their correct locations, while states that are far from being solved get high heuristic scores since they have more blocks in incorrect locations and remaining misplaced blocks are difficult to move to their correct locations.

Here's how it works:

The heuristic function iterates through each block in the current state, comparing its true position to its goal position. If a block is not in its correct place, the heuristic calculates the cost of moving it. This cost is determined by the difference between the height of the stack the block is in and its current height within that stack. The 'max(1, ...)' function ensures that there is always a minimum cost of 1 associated with any move, preventing zero values that should only be returned if a block is already in the goal position. Simply put, if a block is in the wrong position it is guaranteed to contribute at least 1 to the h_score. We then add 1 for each additional block on top of our current block in the current state, since these blocks make it difficult to move our current block to its goal position.

In essence, my heuristic captures the intuition that it's generally more costly to be in a state where an incorrectly placed block is blocked from movement by blocks above it that must be moved first. By summing these costs for all mispositioned blocks, my heuristic assigns a given state a higher score if it contains many misplaced blocks that must wait on other blocks before they can be moved to their goal position, and this score decreases as the number of misplaced blocks decreases and/or the number of blocks blocking a misplaced block from moving decreases.

Heuristic Example 1 (probA03):

```
initial_state looks like:
CE
AD
B
>>>>>>>>>>>>>>>
goal_state looks like:

ADBC
E
>>>>>>>>>>>>>>>
heuristic starts as: 0
about to iterate through items in current_state_positions_dict
current_state_positions_dict looks like:
{'C': (0, 0), 'E': (0, 1), 'A': (1, 0), 'D': (1, 1), 'B': (2, 0)}

goal_state_positions_dict looks like:
{'A': (1, 0), 'D': (1, 1), 'B': (1, 2), 'C': (1, 3), 'E': (2, 0)}

iteration: 1
for current state, block C is at stack_index 0 and height 0.
in the goal state, block C is at stack_index 1 and height 3.
true position and goal position for block C did not match.
block C is currently at height 0.
the height of C's stack in the current state is 2.
will add max between 1 and 2 to h_score.
h_score is now: 2
iteration: 2
for current state, block E is at stack_index 0 and height 1.
in the goal state, block E is at stack_index 2 and height 0.
true position and goal position for block E did not match.
block E is currently at height 1.
the height of E's stack in the current state is 2.
will add max between 1 and 1 to h_score.
h_score is now: 3
iteration: 3
for current state, block A is at stack_index 1 and height 0.
in the goal state, block A is at stack_index 1 and height 0.
block A is at the correct position. h_score does not need to be adjusted.
iteration: 4
for current state, block D is at stack_index 1 and height 1.
in the goal state, block D is at stack_index 1 and height 1.
block D is at the correct position. h_score does not need to be adjusted.
iteration: 5
for current state, block B is at stack_index 2 and height 0.
in the goal state, block B is at stack_index 1 and height 2.
true position and goal position for block B did not match.
block B is currently at height 0.
```

```
the height of B's stack in the current state is 1.
will add max between 1 and 1 to h_score.
h_score is now: 4
heuristic for A03 initial state is: 4
```

Heuristic Example 2 (probB11):

```
initial_state looks like:

F
EJ
BCHI
ADG
>>>>>>>>>>>>>>>
goal_state looks like:
I

AD
BCHFGE
J
>>>>>>>>>>>>>>>
heuristic starts as: 0
about to iterate through items in current_state_positions_dict
current_state_positions_dict looks like:
{'F': (1, 0), 'E': (2, 0), 'J': (2, 1), 'B': (3, 0), 'C': (3, 1), 'H': (3,
2), 'I': (3, 3), 'A': (4, 0), 'D': (4, 1), 'G': (4, 2)}

goal_state_positions_dict looks like:
{'I': (0, 0), 'A': (2, 0), 'D': (2, 1), 'B': (3, 0), 'C': (3, 1), 'H': (3,
2), 'F': (3, 3), 'G': (3, 4), 'E': (3, 5), 'J': (4, 0)}

iteration: 1
for current state, block F is at stack_index 1 and height 0.
in the goal state, block F is at stack_index 3 and height 3.
true position and goal position for block F did not match.
block F is currently at height 0.
the height of F's stack in the current state is 1.
will add max between 1 and 1 to h_score.
h_score is now: 1
iteration: 2
for current state, block E is at stack_index 2 and height 0.
in the goal state, block E is at stack_index 3 and height 5.
true position and goal position for block E did not match.
block E is currently at height 0.
the height of E's stack in the current state is 2.
will add max between 1 and 2 to h_score.
h_score is now: 3
iteration: 3
for current state, block J is at stack_index 2 and height 1.
in the goal state, block J is at stack_index 4 and height 0.
true position and goal position for block J did not match.
block J is currently at height 1.
the height of J's stack in the current state is 2.
will add max between 1 and 1 to h_score.
```

```
h_score is now: 4
iteration: 4
for current state, block B is at stack_index 3 and height 0.
in the goal state, block B is at stack_index 3 and height 0.
block B is at the correct position. h_score does not need to be adjusted.
iteration: 5
for current state, block C is at stack_index 3 and height 1.
in the goal state, block C is at stack_index 3 and height 1.
block C is at the correct position. h_score does not need to be adjusted.
iteration: 6
for current state, block H is at stack_index 3 and height 2.
in the goal state, block H is at stack_index 3 and height 2.
block H is at the correct position. h_score does not need to be adjusted.
iteration: 7
for current state, block I is at stack_index 3 and height 3.
in the goal state, block I is at stack_index 0 and height 0.
true position and goal position for block I did not match.
block I is currently at height 3.
the height of I's stack in the current state is 4.
will add max between 1 and 1 to h_score.
h_score is now: 5
iteration: 8
for current state, block A is at stack_index 4 and height 0.
in the goal state, block A is at stack_index 2 and height 0.
true position and goal position for block A did not match.
block A is currently at height 0.
the height of A's stack in the current state is 3.
will add max between 1 and 3 to h_score.
h_score is now: 8
iteration: 9
for current state, block D is at stack_index 4 and height 1.
in the goal state, block D is at stack_index 2 and height 1.
true position and goal position for block D did not match.
block D is currently at height 1.
the height of D's stack in the current state is 3.
will add max between 1 and 2 to h_score.
h_score is now: 10
iteration: 10
for current state, block G is at stack_index 4 and height 2.
in the goal state, block G is at stack_index 3 and height 4.
true position and goal position for block G did not match.
block G is currently at height 2.
the height of G's stack in the current state is 3.
will add max between 1 and 1 to h_score.
h_score is now: 11
heuristic for B11 initial state is: 11
```

Heuristic performance

HEURISTIC PERFORMANCE: H6

testcase set	testcase #	solution length	number of iterations	max queue size
A	3	3	8	21
	4	4	11	26
	5	5	14	40
	6	6	95	223
	7	7	29	83
	8	10	204	515
	9	9	124	288
	10	10	361	799
	11	11	261	628
B	3	3	38	508
	4	4	36	564
	5	6	51	787
	6	6	31	418
	7	7	100	1414
	8	8	509	6531
	9	8	207	2818
	10	10	185	2476
	11	10	551	7252
	12	12	1533	21463
	13	14	2253	33056
	14	13	1445	19692
	15	14	5874	84203
	16	17	86294	1128601
	17	16	24384	318498
	18	16	63203	734692
	19	16	11677	159446
	20	18	26776	364381