**A\* Search to solve the Block Stacking Problem**

**-Harihara K Narayanan**

I used two heuristics to solve this problem. Heuristic 2 is an improvement over Heuristic 1, as Heuristic 2 uses the condition in H1 plus it includes an extra condition.

Essentially for any state S that is generated, H1(S) <= H2(S). As a result, Heuristic 2 is closer is guaranteed to find the goal state in fewer steps. This is proven empirically – refer to the table and graph.

**Coming up with the Heuristics**

The main components of the problem are the blocks and the stacks. The main goal state introduces the following constraints:

1. Ordering of the stack in the order from 1 to N.
2. All blocks are in the rightmost stack.

So by looking at these two conditions, the heuristics I develop must be related to how the blocks in a single stack are ordered, and also how many stacks are occupied.

So in Heuristic 1 – I use the ordering condition. In Heuristic 2, I use Heuristic 1 plus the number of stacks occupied constraint.

**Heuristic 1**

In this case, for each stack, I check every block Bi with the block Bj, where Bj is immediately on top of Bi.

If Value(Bj) != Value(Bi) – 1, then I increment the heuristic H1 by 1.

The reasoning here is that, if Bj does not satisfy the condition represented here, then either Bj or Bi would have to be moved in order to reach the goal state ordering, wherein each block on top of another block has a value that is 1 lesser. So essentially this heuristic states that – for a given block, if the block on top of it has a value that is not one less than its value, it needs to be moved in order to reach the goal state.

It can also be verified that H1 (goal state) = 0. This is because, in the goal state, for every block – the block on top has a value that is one less. So effectively we do not increment H1 at all and it is equal to zero.

**Heuristic 2**

I use the condition in Heuristic 1, but add another condition here.

If all 3 stacks contain at least one block – then increment H2 by 2.

If there is 1 empty stack, increment H2 by 1.

If 2 empty stacks, do not increment H2.

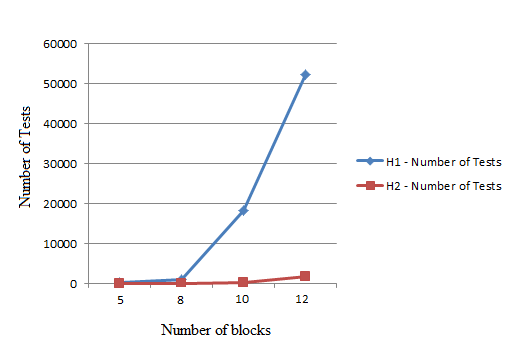
Basically the reasoning here is that, the goal state will have only one stack filled up and the remaining empty. So if initially n stacks are occupied, you at least need an additional (n-1) moves to reach the final state.

Note that Heuristic 2 is same as Heuristic 1 for states, wherein only a single stack is filled up and the other two stacks are empty. So always H2 yields a higher or equal value compared to H1. Also it can be easily verified that H2 value for the goal state is zero. This is because all the blocks are in the same stack and the blocks are in the order 1 to n from the top.

The following table shows the performance of H1 and H2 for different values of blocks (as input). Note that the value here is the Average value obtained for 5 random runs with the same number of blocks.

|  |  |  |
| --- | --- | --- |
|  | **H1 – Avg. No. of Tests** | **H2 – Avg. No. of Tests** |
| **No. of blocks = 5** | **189.4** | **26.8** |
| **No. of blocks = 8** | **978.6** | **95.8** |
| **No. of blocks = 10** | **18415.2** | **290.8** |
| **No. of blocks = 12** | **52360.6** | **1729.2** |

**Table 1. Average No. of Test cases for H1 and H2 based search for different number of blocks. The numbers here are average values for a run of 5 randomly generated tests.**



**Figure 1. Plot of Average Number of Tests versus the number of blocks.**

**Code Information**

I coded the Stacking Problem using C++, and compiled it using a g++ compiler on an Ubuntu Linux Machine. I have used the Standard Template Library extensively within the code.