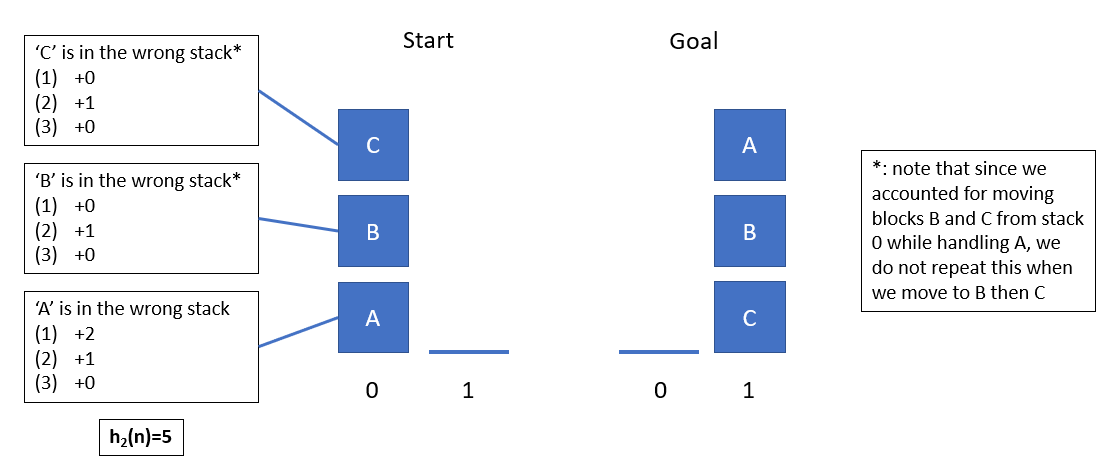
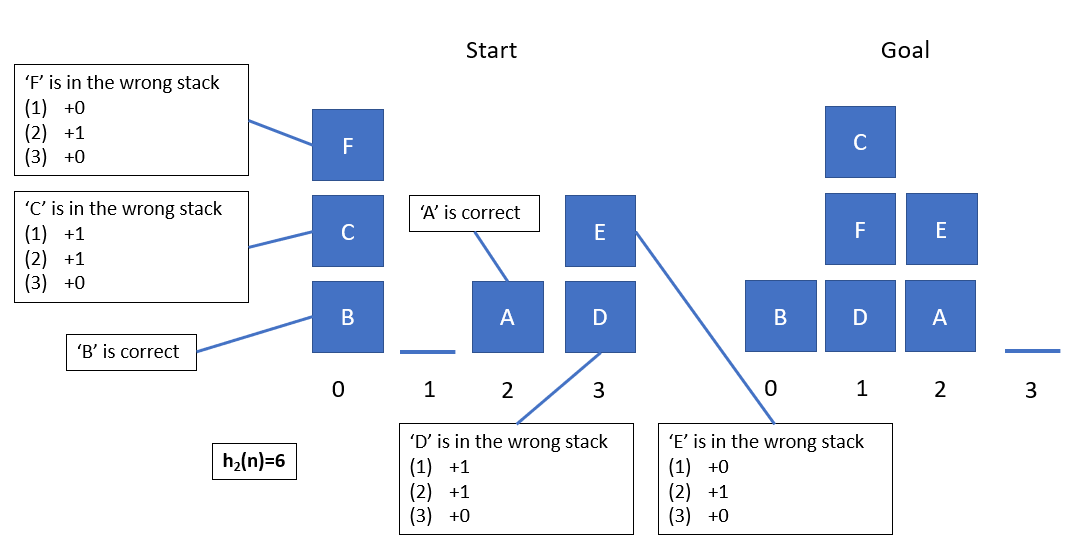
**Heuristics**

* Trivial
  + Solved: 11
  + Failed: 34
  + Simulates the BFS search from Proj2 with similar results
* Default
  + Solved: 21
  + Failed: 24
  + Simply counts how many blocks are not where they appear in the solution
* My Heuristic
  + Solved: 33
  + Failed: 12
  + My heuristic works by moving through the current state one stack at a time checks each block in a stack starting from the bottom whether or not it is **out of place** (OOP).
    - If a block is OOP and in the wrong stack
      * 1) Score is increased by the number of blocks on top of the block in question (simulates removing all blocks from stack above the target block)
      * 2) Score is increased by one to move the block in question to the correct stack
      * 3) Score is increased by one for each block moved in (1) that needs to be returned to its original stack
    - If a block is OOP and in the correct stack
      * 1) Score is increased by the number of blocks on top of the block in question (simulates removing all blocks from stack above the target block)
      * 2) Score is increased by one to move the block in question to a temporary stack
      * 3) Score is increased by one for each block moved in (1) that needs to be returned to its original stack
      * 4) Score in increased by one to move the block in question back to the original stack now in the correct spot
    - **Important Note:** ifsteps (1) and (3) were carried out (meaning the score already accounts for a group of blocks being removed to access the target block that is OOP), they are not carried out again in either above case in the same stack. This helps to refrain from drastically overestimating how many moves are remaining since moving some of these blocks will help place others into the correct spot.
  + Example 1: 
  + Example 2: 
  + In general, this heruistic is a bit of an overestimate (as in both of the above examples), however it can effectivly solve most of the challenge problems with reasonable iteration count by heavily penalizing states containing deeply buried OOP blocks in a stack.
    - It would make sense that my program had difficulties solving the problems in which there were significantly more blocks than stacks. This would imply that many blocks were OOP and deeply nested which were penalized.
  + For a few of the challenge problems, my heuristic found solutions that were one step longer, but it also found a few solutions that were one step shorter (compared to Dr. Ioerger’s).
  + Of the problems solved, solutions with depths less than 25 generally had max queue sizes under 7,000 with a median of 156. Solutions with solutions over 25 tended to increase exponentially up to about 60,000.
  + Likewise, of the problems solved, solutions with depths less than 20 generally had iterations under 10,000 with a median of 229. Solutions with solutions over 20 tended to increase exponentially up to about 82,000 iterations.