Student Id: **20642433**
Name: **Anurag Banger**

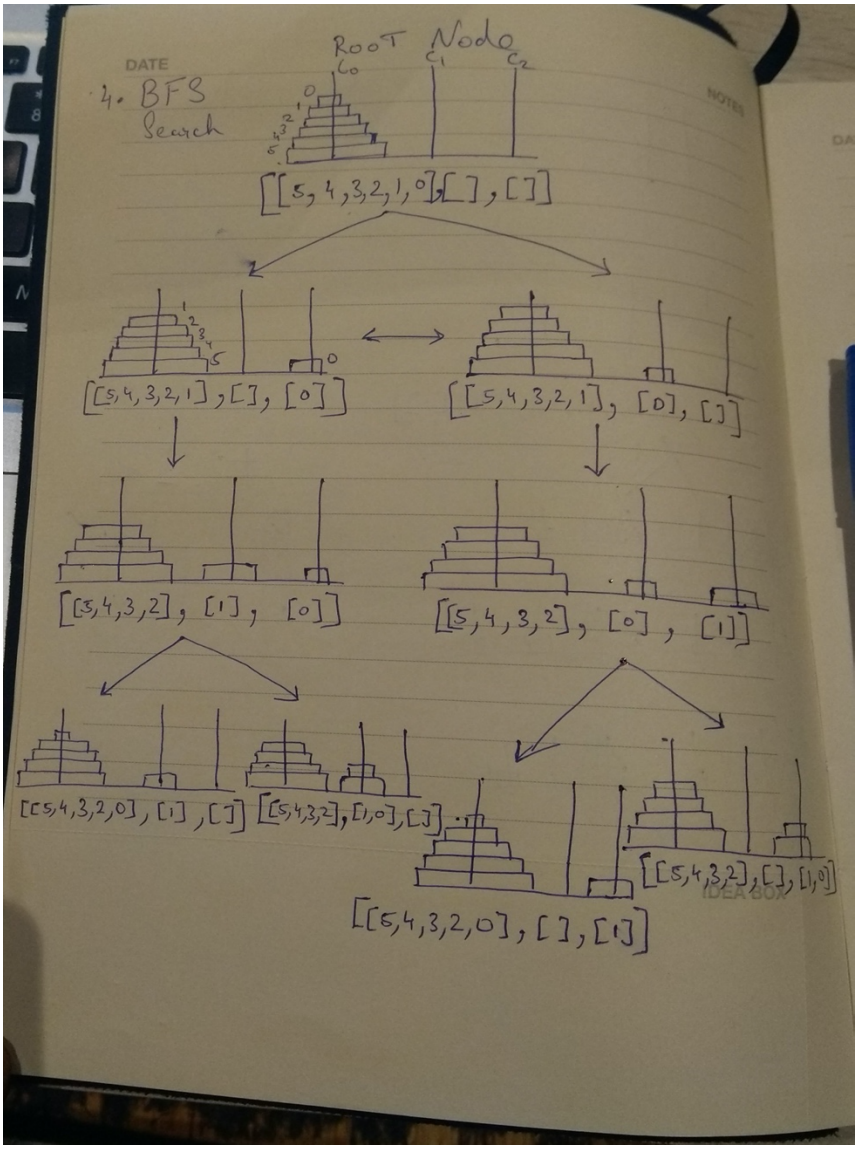
Assignment 1 Tower of Hanoi

1. **Scheme 1** would be best to implement as in this case we have only few information to take in account. It would easy to view and access the ring placement using the length 3 of the list representing the number of columns. Each index would have rings ordered in descending order (from bottom to top).
   [r5, r4, r3, r2], [r1], [r0]]

2. The optimal number of moves for Tower of Hanoi problem for n columns and m rings are:
   2^n - 1
3. We need to check if the stack we are moving from is not empty and if the stack we are moving to is empty or the current ring is smaller than ring on the stack.

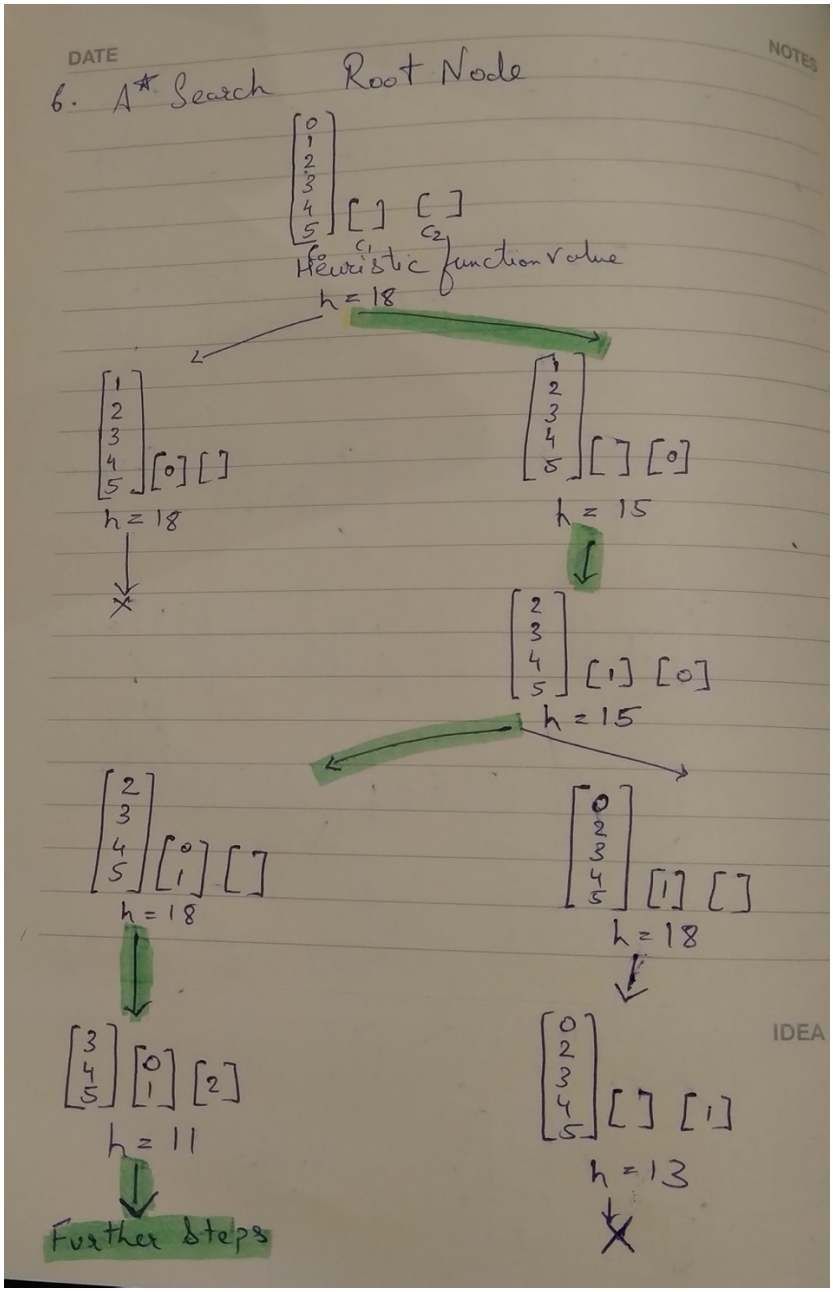
```
def move_ring(self, from_stack, to_stack):
        if len(self.stacks[from_stack]) and
        (not self.stacks[to_stack] or
        self.stacks[to_stack][0] > self.stacks[from_stack][0]):
                stacks_copy = deepcopy(self.stacks)
                stacks_copy[to_stack].insert(0, stacks_copy[from_stack].pop(0))
                self.create_child(stacks_copy)
```

4. Breadth First Search



5. In order for our node to be correct, all the rings must be arranged in the final stack in right order (0 to 5). In **Part 1** as each ring not on the right column, we would do at least 1 move to get it into the correct stack. So, we add the number of rings not in the right most stack to the **Heuristic Value 'h'.** In **Part 2** each ring on the right where there is a bigger ring in the other stack, we would have to first move the right most ring out of the way and move it back, therefore adding 2 to 'h' for each ring in the right most stack where there is a bigger ring in other stacks.

6. A* Search



Output Table:

N Columns	R Rings	BFS			A*		
		Total Steps	Path Node Count	Total Time (in seconds)	Total Steps	Path Node Count	Total Time (in Seconds)
3	3	27	7	0.00196	20	7	0.00111
3	6	729	63	0.04796	530	129	0.03326
3	7	2187	127	0.14776	1715	391	0.12157
3	9	19683	511	2.65506	13701	2291	2.18951
4	7	16384	25	3.25789	12295	35	6.33632

REPORT

In the assignment 1 we will be solving the Tower of Hanoi Problem using Breadth First Search algorithm (BFS) and A Star Search algorithm. The objective of this problem is to move the entire stack of rings from the first, to the last column, while obeying 3 simple rules. First, only one ring can be moved at a time. Second, each move consists of taking the upper ring from one of the stacks and placing it on the top of another stack or an empty pole. Third, a larger ring must not be placed on top of a smaller ring. For this problem we will be using the scheme 1, where a state is represented by a list and the list has a length 3 to represent the number of columns. Each index in this list contains the list of all rings stacked on this column. Each inner list rings will be stored from bottom to top which means the smallest ring of that column will be at 0^{th} index of the inner list.

We will be using a Node class which would store the parent node for back tracing once goal is achieved, List of child nodes, current state, ring count and the f where heuristic value for the node is calculated. For moving the rings, we have to check the column we are moving "**FROM**" is not empty and also, we have to check the "**FROM**" stack if 0^{th} index ring is smaller than the "**TO**" stack 0^{th} index. Once this condition is satisfied, we will make a copy of the current stacks and make the changes for moving the ring. For the breadth first search algorithm we will take 2 lists open list and visited list. The next nodes have to be taken from the top of the open list. Until we find the correct or goal node, we will be looping through all the neighboring nodes. Each new node has to be added at the end of open_list. Once you find the correct state or goal you have to return the path to solution by back tracing using each node's parent. In A Star Search algorithm, we have to do only 1 extra thing as to BFS, once a node is visited as we are doing in BFS algorithm we have then sort the open_list based on the **heuristic value 'f'.** Heuristic value can be calculated by 2 parts. In part 1 as each ring not on the right column, we would do at least 1 move to get it into the correct stack. So, we add the number of rings not in the right most stack. In part 2 each ring on the right where there is a bigger ring in the other stack, we would have to first move the right most ring out of the way and move it back, therefore adding 2 for each ring in the right most stack where there is a bigger ring in other stacks. Utilizing the above heuristic function should allow to solve this problem using A star Search Algorithm.

Executing Breadth first search algorithm and A Star Search algorithm with N columns and R rings we have recorded the Total steps, final path node count and total time in seconds. From the output table above, we can see that for most of the cases A star Search algorithm is performing better than the BFS Search algorithm. Total Steps and Total Time are lesser for A* Search Algorithm.