AI LAB 2-REPORT

D.Sakeena Sadhika : 210010062

Poorvasha Dhanunjay Chauhan : 210010040

GROUP_5

1.BRIEF DESCRIPTION OF THE DOMAIN:

a. State space

•From the given problem statement, each statement consists of six neighbors.

•**State space S** is a set which consists of all possible configuration of the boxes in the set of three stacks.

b. Start node and goal node

• We will read the start node and goal node from the file *"input.txt".*

• Start node is the initial node from where we will use our **BFS** or **Hill Climbing algorithm** to reach the final destination i.e. **Goal State.**

• Goal Node is used to find the heuristic value of every state that we visit and it is also used in **goal test()** function to check whether the final state is reached or not.

• First three lines of *"input.txt*" contains the start node and the next three lines contain the goal state.

c. MOVEGEN and GOALTEST algorithm

Psuedo code for move gen() and goal test() goal test() :

if currentState == goalState

return true

=0

move_gen():

```
1: nbors=[]
2: for i in range(len(state)):
3: if(len(state[i] ≥ 1) : for j in range(len(state)) :
4: if i!=j:
5: temp = copy.deepcopy(state)
6: top = temp[i].pop()
```

7: temp[j].append(top)
8: rnbors.append(temp)
9: return nbors

2. HEURISTIC FUNCTIONS CONSIDERED (MINIMUM OF 2)

Consider that in the presence state the coordinate of ith block is (xi , yi).

* Heuristic 1

If xi is equal to yi then we add 1 to the heuristic. The goal state have heuristic value equal to the number of block so to maximize the heuristic.

Figure 1: Code for **heuristic 1**

```
###     HEURISTICS START     ###
#    First heuristic START
def oneadder(state1, state2):
    heu=0
    for m,i in enumerate(state1):
        for n,j in enumerate(i):
            for p,x in enumerate(state2):
                for q,y in enumerate(x):
                    if j == y:
                        if n == q and m==p:
                            heu+=1
    return -heu
#    First heuristic END
```

* Heuristic 2

It is same as Heuristic 1 instead of adding 1 , we add the depth of the block.

```
40  #    Second heuristic START
41  def leveladder(state1, state2):
42      heu=0
43      for m,i in enumerate(state1):
44          for n,j in enumerate(i):
45              for p,x in enumerate(state2):
46                  for q,y in enumerate(x):
47                      if j == y:
48                          if n == q and m==p:
49                              heu+=(q+1)
50      return -heu
51  #    Second heuristic END
52
```

*Heuristic 3

For each block we add the Manhattan distance between (xi , yi) to (x, y) is equal to |x − xi | + |y – yi| The heuristic value of goal state is zero and our objective is to minimize the heuristic.

```
#    Third heuristic START
def manhattan(state1, state2):
    heu=0
    for m,i in enumerate(state1):
        for n,j in enumerate(i):
            for p,x in enumerate(state2):
                for q,y in enumerate(x):
                    if j == y:
                        heu = heu + abs(m-p) + abs(n-q)
    return heu
#    Third heuristic END
###     HEURISTICS END     ###
```

3. Hill Climbing:

a. State Explored

• In general BFS approach will explore more state than Hill climbing ap proach.
• In worst case BFS, will explore all states, while Hill climbing approach will explore O(bd), where b is maximum beam width and d is depth of search graph from start node.

b.Time taken

• In worst case BFS will explore all states hence time taken by BFS approach will be O(b ^d).

• In worst case time taken by Hill climb approach will be O(bd). where b is maximum beam width and d is depth of search graph from start node. Hence from above to points we can conclude that, with respect to time taken Hill climbing approach is better than BFS approach.

c. Reaching the optimal solution

• In case of Hill Climbing algorithm we visit neighbouring state only when they seems to be more optimal than current state. Mathematically at least one of the neighbouring states would have more heuristic value than current state, if that's not the case then function will return current state because current state has more heuristic value than the neighbour states hence that's more optimal than neighbouring states.

• While in case of BFS approach, BFS algorithm will ensure to explore all entire search space. If this will be the case than if solution exist we will definitely reach to our goal state.

Hence from the above points we can conclude that there may be the cases possible where we can not reach to the goal cases by Hill climbing ap proach but if solution will exist then we will definitely reach there by BFS approach because it's giving enclosure over entire search space while Hill climbing algorithm is giving enclosure over limited space only.