Artificial Intelligence
Lab Report 1

S.Surzith Raj Pandi: 200010049
Vidyasagar S Singadi: 200010056

January 2, 2022

Contents

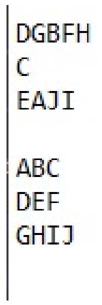
1 Brief Description about main domain

1.1 State Space

- From the given problem statement, each statement consist of six neighbours.
- **State space *S*** is a set which consist of all possible configuration of the boxes in *the set of three stacks.*

1.2 Start node and goal Node

- We will read start node and goal node from file *"input.txt"*.
- **Start node** is the initial node from where we will use our **BFS** or **Hill Climbing** algorithm to reach final destination *i.e.* **Goal State**.
- Goal Node is used to find heuristic value of every state that we visit and it is also used in **goal_test()** function to check whether final state is reached or not.
- First three lines of *"input.txt"* contains the start node and the next three lines contains the goal state.



```
DGBFH
C
EAJI

ABC
DEF
GHIJ
```

Figure 1: Format of input.txt

2 Psuedo code for move_gen() and goal_test()

goal_test() :

if currentState == goalState

return true

```
        =0
```

move_gen():

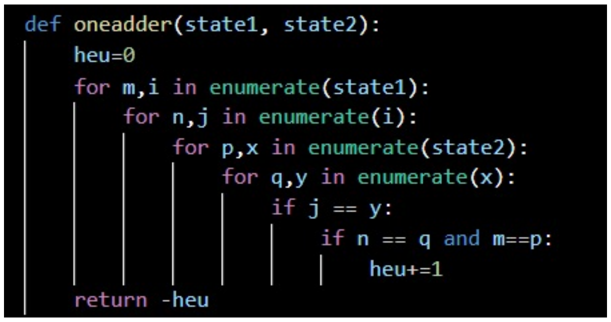
```
nbors=[]
for i in range(len(state)):
if(len(state[i] $\geq$ 1) : $forjinrange(len(state))$ :
    if i!=j:
    temp = copy.deepcopy(state)
    top = temp[i].pop()
    temp[j].append(top)
    rnbors.append(temp)
    return nbors
```

3 Heuristic functions

Consider that in the presence state the coordinte of i^th block is (x_i, y_i).

3.1 Heuristic 1

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

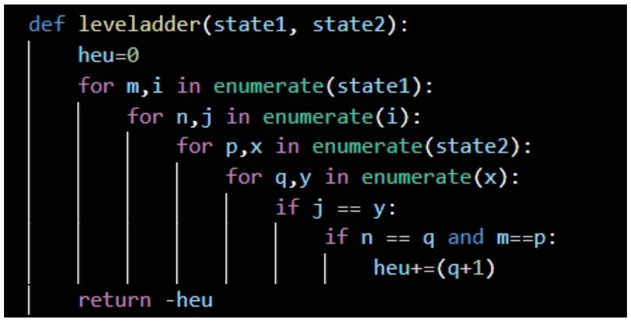


```
def oneadder(state1, state2):
    heu=0
    for m,i in enumerate(state1):
        for n,j in enumerate(i):
            for p,x in enumerate(state2):
                for q,y in enumerate(x):
                    if j == y:
                        if n == q and m==p:
                            heu+=1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

Figure 2: Code for heuristic 1

3.2 Heuristic 2

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

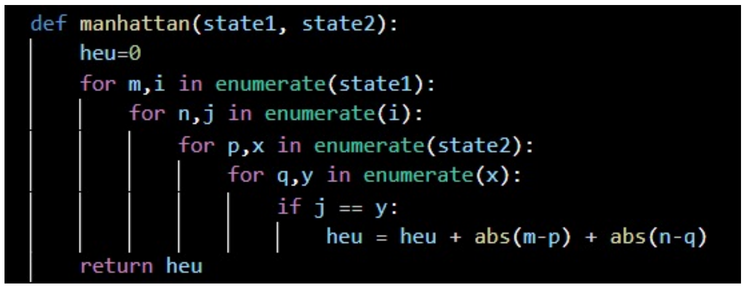


```
def leveladder(state1, state2):
    heu=0
    for m,i in enumerate(state1):
        for n,j in enumerate(i):
            for p,x in enumerate(state2):
                for q,y in enumerate(x):
                    if j == y:
                        if n == q and m==p:
                            heu+=(q+1)
    return -heu
```

Figure 3: Code for heuristic 2

3.3 Heuristic 3

For each block we add the Manhattan distance between (x_i, y_i) to (x, y) is equal to $|x - x_i| + |y - y_i|$. The heuristic value of goal state is zero and our objective is to minimize the heuristic.



```python
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
```

Figure 4: Code for heuristic 3

4 Comparison between BFS and Hill Climbing

4.1 State Explored

- In general BFS approach will explore more state than Hill climbing approach.
- 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.

4.2 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.

4.3 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 approach 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.

5 Best First search analysis and observation

- BFS provides the enclosure to entire search space.
- In general space complexity and time complexity of BFS approach is greater than Hill climbing approach because size of search space of BFS algorithm is used to be greater than Hill climbing approach.

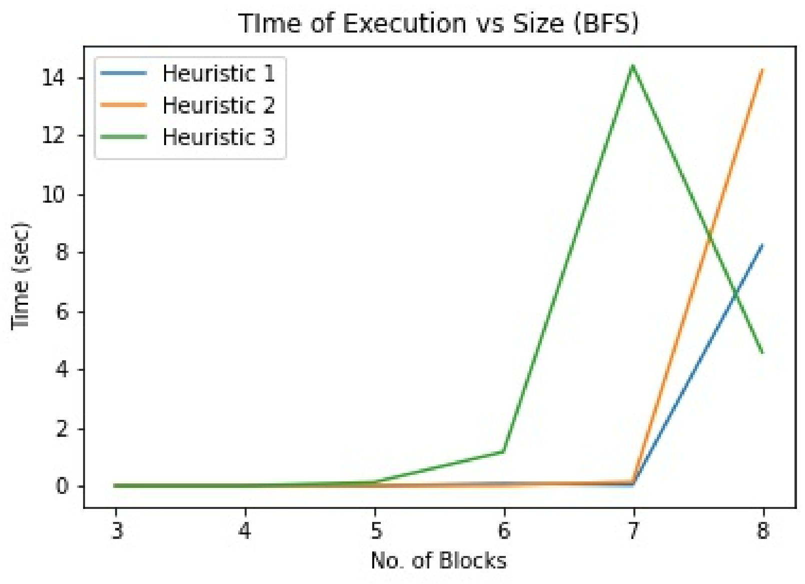
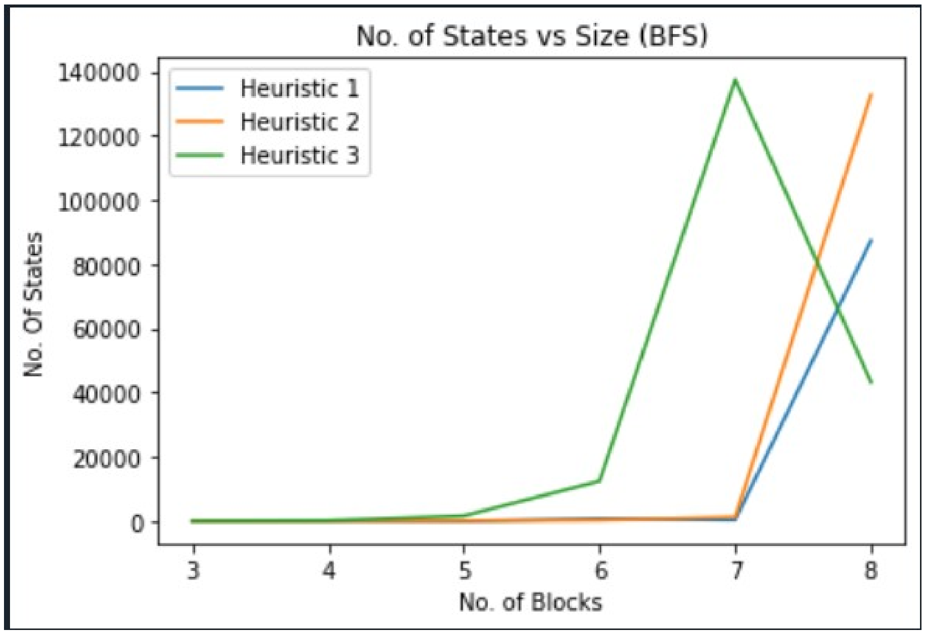


Figure 5: Time Execution graph of BFS for three different Heuristic values



6 Result

In terms of time and space complexity Hill Climbing is better than BFS because in hill climbing it stores only one neighbours while in BFS it stores all possible neighbours. And there is no back-tracking in Hill Climbing

7 References

- http://geeksforgeeks.com
- https://stackoverflow.com/