CS 312: Artificial Intelligence Laboratory

Lab 2 report

— B Siddharth Prabhu (200010003)

— Sourabh Bhosale (200010004)

- **Introduction:**

The objective of this task is to implement Block World Domain**.** Blocks World Domain Game starts with an initial state consisting of a fixed number of blocks arranged in 3 stacks and we can move only top blocks of the stacks. Blocks World is a planning problem where we know the goal state beforehand and the path to the Goal state is more important. We have to achieve a goal state that is a particular arrangement of blocks by moving these blocks.

- **Description:**

1. State space:

A State space is the set of all states reachable (directly or indirectly) from the initial state. The state space for this problem is all possible permutations of the given ‘n’ blocks on 3 stacks.

2. Start node and goal node:

Start node is the initial arrangement of blocks at the start of the game. Goal node is the arrangement which we want to achieve in order

to complete the game. The start states and goal states will be given in the input file.

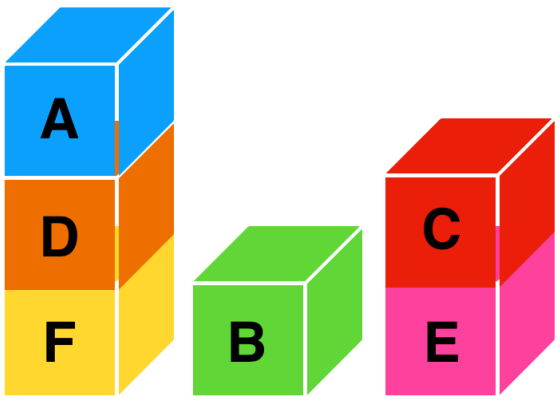
Example:

Start State:

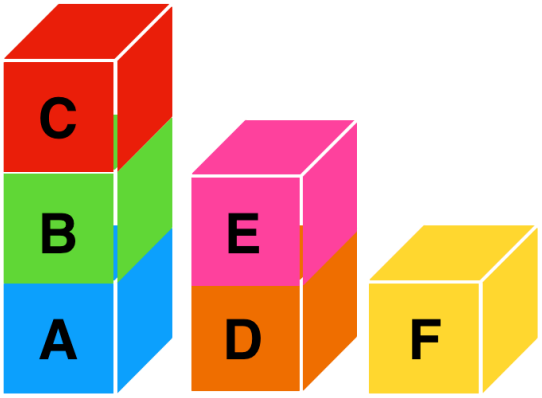
Stack1 = [‘F’, ‘D’, ‘A’], Stack2 = [‘B’], Stack3 = [‘E’, ‘C’].

Goal State:

Stack1 = [‘A’, ‘B’, ‘C’], Stack2 = [‘D’, ‘E’], Stack3 = [‘F’].



Start State



Goal State

- **Pseudocode:**

a) GoalTest :

```
function GoalTest (current_state, goal_state) :
    if current_state is goal_state :
        return True
    else :
        return False
```

b) MoveGen :

```
function MoveGen (current_state):

    initialise empty list for neighbours

    if stack1:
        state_copy = deepcopy(current_state)
        pop element from stack1 and append it to stack2
        append state_copy to neighbours
        state_copy = deepcopy(current_state)
        pop element from stack1 and append it to stack3
        append state_copy to neighbours

    if stack2:
        state_copy = deepcopy(current_state)
        pop element from stack2 and append it to stack1
        append state_copy to neighbours
        state_copy = deepcopy(current_state)
        pop element from stack2 and append it to stack3
        append state_copy to neighbours

    if stack3:
```

```
        state_copy = deepcopy(current_state)
        pop element from stack3 and append it to stack1
        append state_copy to neighbours
        state_copy = deepcopy(current_state)
        pop element from stack3 and append it to stack2
        append state_copy to neighbours

    return neighbours
```

- **Heuristic Functions:**

A) Heuristic Function - 1:

In this heuristic function, we implemented following logic:

1) +1 for block if it is in correct stack and on the top of correct block. -1 for block otherwise.

Example:

Start state: [a, b, f], [d], [c, e] Goal state: [b, f], [c, a], [d, e]

Heuristic of start state = -1-1+1-1-1-1 = -4

Heuristic of goal state = +6

B) Heuristic Function - 2:

In this heuristic function, we implemented following logic.

1)+1 x (level of block) if block is in correct stack and on the top of correct structure i.e., all the block below it must be in correct form

2)-1 x (level of block) otherwise *

3) NOTE: ground block has level 1

Example:

Start state – [a, b, f], [d], [c, e] Goal state – [b, f], [c, a], [d, e]

Heuristic of start state = -1-2-3-1-1-2 = -10

Heuristic of goal state = +9

C) Heuristic Function - 3:

In this heuristic function, we implemented following logic:

Calculate the Manhattan distance between each block's current position with its position in the goal state and sum it.(This has to be maximised.

- **Tables:**

Algorithm	Parameters					
	Heuristic Fn	Input File	No. of states in path	No. of states explored	Time taken (s)	Reached goal state?
BFS	1	input1.txt	61	20160	325.6	YES
	2	input1.txt	78	20160	301.8	YES
	3	input1.txt	72	15178	285.4	YES
HC	1	input2.txt	2	3	0.001	NO
	2	input2.txt	2	3	0.001	NO
	3	input2.txt	4	5	0.004	NO
BFS	1	input3.txt	7	60	0.02	YES
	2	input3.txt	7	60	0.01	YES
	3	input3.txt	7	17	0.01	YES
HC	1	input4.txt	1	2	0.002	NO
	2	input4.txt	1	2	0.001	NO
	3	input4.txt	2	3	0.002	NO

- **Conclusion:**

  We can get a solution using best first search which can take longer time for search but in hill climbing it will take lesser time but might not give the solution at times.