

**Department of Computer Engineering**  
**Academic Term II: 23-24**

**Class: B.E (Computer), Sem – VI**

**Subject Name: Artificial Intelligence**

**Student Name: Jatin Kadu**

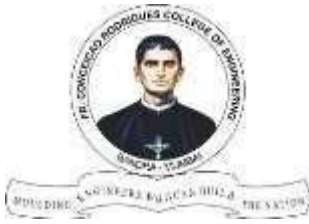
**Roll No: 9548**

<b>Practical No:</b>	<b>7</b>
<b>Title:</b>	Block World Problem solving by hill climbing approach
<b>Date of Performance:</b>	<b>18/3/24</b>
<b>Date of Submission:</b>	<b>25/3/24</b>

**Rubrics for Evaluation:**

<b>Sr. No</b>	<b>Performance Indicator</b>	<b>Excellent</b>	<b>Good</b>	<b>Below Average</b>	<b>Marks</b>
1	On time Completion & Submission (01)	01 (On Time)	NA	00 (Not on Time)	
2	Logic/Algorithm Complexity analysis (03)	03(Correct)	02(Partial)	01 (Tried)	
3	Coding Standards (03): Comments/indentation/Naming conventions Test Cases /Output	03(All used)	02 (Partial)	01 (rarely followed)	
4	Post Lab Assignment (03)	03(done well)	2 (Partially Correct)	1(submitted)	
<b>Total</b>					

**Signature of the Teacher:**



## Experiment No: 7

**Title:** Block world problem solving by Hill Climbing method

**Objective:** To study the solution for block word problem by Hill Climbing approach

### Theory:

#### SIMPLE HILL CLIMBING

1. Evaluate the initial state. If it is also a goal state, then return it and quit. Otherwise, continue with the initial state as the current state.

2. Loop until a solution is found or until there are no new operators left to apply in the current state:

a) Select an operator that has not yet been applied to the current state and apply it to produce a new state

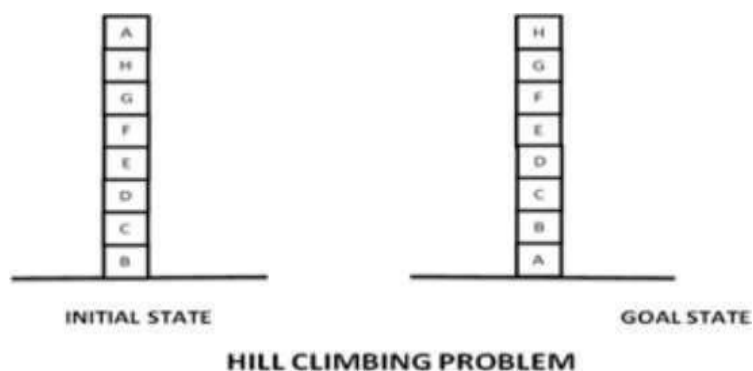
b) Evaluate the new state,

i. If it is a goal state, then return it and quit.

ii. If it is not a goal state but it is better than the current state, then make it the current state.

iii. If it is not better than the current state, then continue in the loop.

Consider the blocks world problem. Assume the same operators (i.e., pick up one block and put it on the table; pick up one block and put it on another one) suppose it uses the following heuristic function:

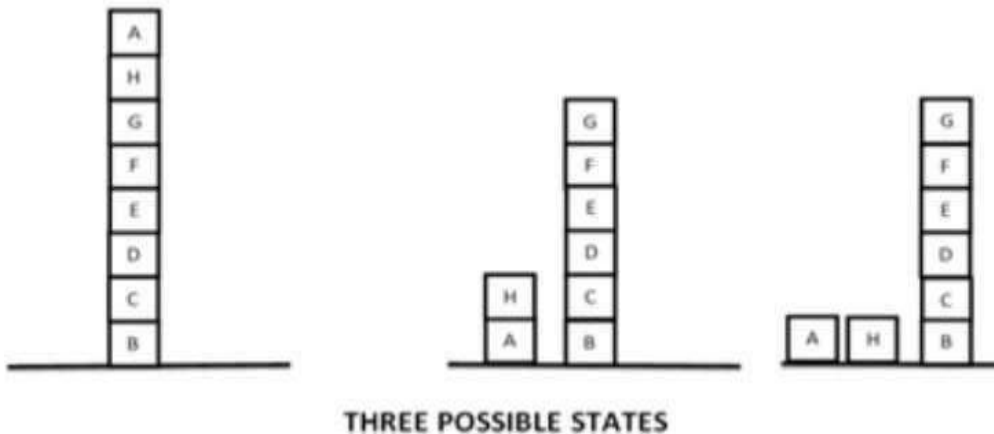


Local: Add one point for every block that is resting on the thing it is supposed to resting on.



Subtract one point for every block that is sitting on the wrong thing.

Using this function, the goal state has a score of 8. The initial state has a score of 4 (since it gets one point added for blocks C, D, E, F, G and H and one point subtracted for blocks A and B).



There is only one move from the initial state, namely to move block A to the table. That produces a state with a score of 6. The hill-climbing procedure will accept that move. From the new state, there are three possible moves, leading to the three states.

These states have the score: (a) 4, (b) 4, and (c) 4. Hill climbing will halt because all these states have lower scores than the current state. The process has reached a local maximum that is not the global maximum.

## OUTPUT:

### Post Lab Questions:

1. What are the advantages and disadvantages of state space search?
2. What are the advantages and disadvantages of the Hill Climbing approach?
3. Describe variations of Hill Climbing approach
4. Solve the Block World problem by using the STRIPS method.

## Code:

```
class BlockWorldProblem:
    def __init__(self, initial_state, goal_score):
        self.current_state = initial_state
        self.goal_score = goal_score

    def evaluate_state(self, state):
        score = 0
        for block, resting_place in state.items():
            if block == resting_place:
                score += 1
            else:
                score -= 1
        return score

    def find_possible_moves(self):
        possible_moves = []
        for block in self.current_state.keys():
            for resting_place in self.current_state.keys():
                if block != resting_place:
                    possible_moves.append((block, resting_place))
        return possible_moves

    def make_move(self, move):
        new_state = self.current_state.copy()
        block, resting_place = move
        new_state[block] = resting_place
        return new_state

    def hill_climbing(self, max_iterations=9999):
        current_score = self.evaluate_state(self.current_state)
        iterations = 0

        print("Initial State:")
        for block, resting_place in self.current_state.items():
            print(f"Block {block} is on {resting_place}")

        while iterations < max_iterations:
            possible_moves = self.find_possible_moves()
            new_states = [self.make_move(move) for move in possible_moves]
            best_state = max(new_states, key=self.evaluate_state)
            best_score = self.evaluate_state(best_state)

            if best_score >= current_score:
                self.current_state = best_state
                current_score = best_score
```

```

        if current_score >= self.goal_score:
            print("\nFinal State:")
            for block, resting_place in self.current_state.items():
                print(f"Block {block} is on {resting_place}")
            return self.current_state
        else:
            print("No better move found.")
            return self.current_state

    iterations += 1

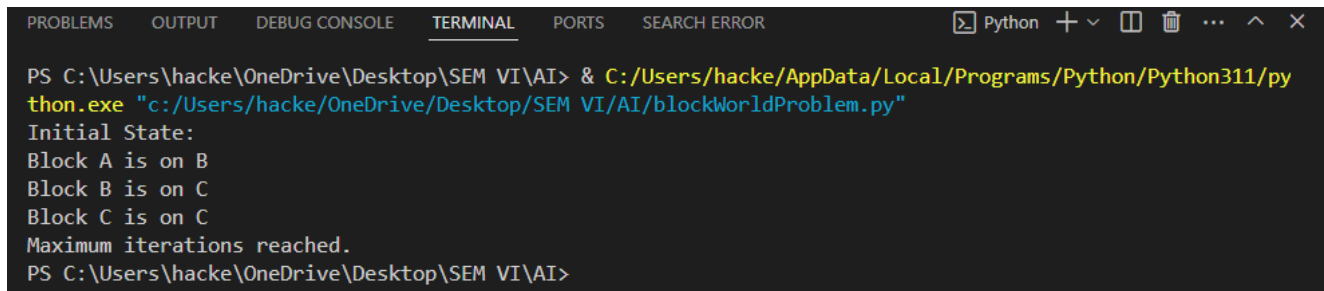
print("Maximum iterations reached.")
return self.current_state

# Example usage:
initial_state = {'A': 'B', 'B': 'C', 'C': 'C'}
goal_score = 3

block_world_problem = BlockWorldProblem(initial_state, goal_score)
solution = block_world_problem.hill_climbing()

```

## Output:



```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS SEARCH ERROR Python + - [] [X] ... ^ X
PS C:\Users\hacke\OneDrive\Desktop\SEM VI\AI> & C:/Users/hacke/AppData/Local/Programs/Python/Python311/python.exe "c:/Users/hacke/OneDrive/Desktop/SEM VI/AI/blockWorldProblem.py"
Initial State:
Block A is on B
Block B is on C
Block C is on C
Maximum iterations reached.
PS C:\Users\hacke\OneDrive\Desktop\SEM VI\AI>

```