Program 4

AIM: BLOCK WORLD PROBLEM USING HILL CLIMBING

Theoretical Description :

Hill climbing is a simple optimization algorithm used in Artificial Intelligence to find the best possible solution for a given problem. It belongs to the family of local search algorithms and is often used in optimization problems where the goal is to find the best solution from a set of possible solutions. In Hill Climbing, the algorithm starts with an initial solution and then iteratively makes small changes to it in order to improve the solution. These changes are based on a heuristic function that evaluates the quality of the solution. The algorithm continues to make these small changes until it reaches a local maximum, meaning that no further improvement can be made with the current set of moves.

Algorithm:

1. Selecting an initial state: Choose a starting solution, either randomly or based on a heuristic
2. Evaluating the initial state: Calculate the objective function value for the current state
3. Generating neighbors: Create the neighboring states of the current state
4. Evaluating neighbors: Check each neighbor to see if it improves the objective function.
5. Selecting the best neighbor: Move to the neighbor with the best objective

Source Code:

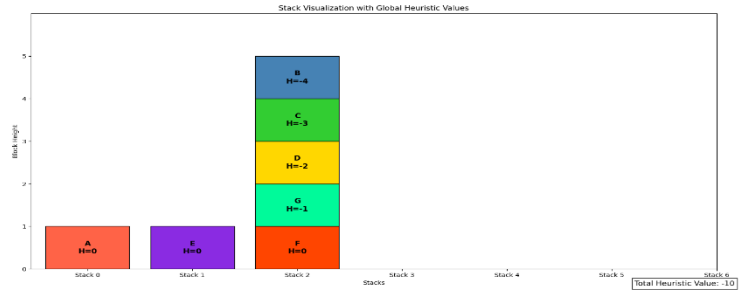
```
import matplotlib.pyplot as plt

import numpy as np

global_heuristic_values = {}

class Stack:

  def __init__(self, blocks=None):

    self.blocks = blocks[::-1] if blocks else []  # Inverted for bottom-to-top display

  def add(self, block):

    self.blocks.append(block)

  def remove(self):

    return self.blocks.pop() if self.blocks else None

  def peek(self):

    return self.blocks[-1] if self.blocks else Non
```

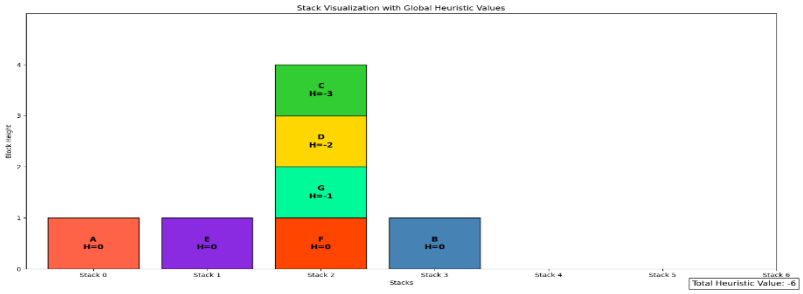
```
    def __str__(self):
        return str(self.blocks[::-1])  # Display stack from top to bottom
class State:
    def __init__(self, stacks):
        self.stacks = stacks
    def __str__(self):
        return " | ".join(str(stack) for stack in self.stacks)
def find_block(state, block):
    for i, stack in enumerate(state.stacks):
        if block in stack.blocks:
            return i
    return -1
def move_block(state, block, target_stack):
    source_stack = find_block(state, block)
    if source_stack == -1 or source_stack == target_stack:
        return
    temp_blocks = []
    while state.stacks[source_stack].peek() != block:
        temp_block = state.stacks[source_stack].remove()
        temp_blocks.append(temp_block)
        update_global_heuristic(state)  # Update heuristic after removing
    moved_block = state.stacks[source_stack].remove()
    state.stacks[target_stack].add(moved_block)
    print(f"\nMoved {moved_block} to Stack {target_stack}")
    update_global_heuristic(state)  # Update heuristic after moving
    for temp_block in reversed(temp_blocks):
        state.stacks[target_stack].add(temp_block)
        print(f"Moved temporary block {temp_block} back to Stack {target_stack} with H=0")
        update_global_heuristic(state)  # Update heuristic after adding back
    visualize_state(state)
def block_heuristic_value(block, stack_index, position_in_stack):
    return -position_in_stack if position_in_stack > 0 else 0
def update_global_heuristic(state):
    global global_heuristic_values
    for i, stack in enumerate(state.stacks):
        for position_in_stack, block in enumerate(stack.blocks):
            if i == 6:  # Goal stack
                global_heuristic_values[block] = position_in_stack
```

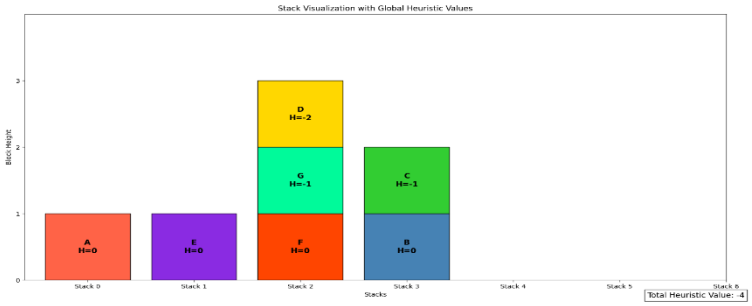
```
        else:
            global_heuristic_values[block] = block_heuristic_value(block, i, position_in_stack)
def calculate_total_heuristic():
    return sum(global_heuristic_values.values())
def visualize_state(state):
    plt.figure(figsize=(12, 8))
    num_stacks = len(state.stacks)
    max_height = max(len(stack.blocks) for stack in state.stacks)

    block_colors = {
        'A': '#FF6347', 'B': '#4682B4',  'C': '#32CD32',    'D': '#FFD700', 'E': '#8A2BE2', 'F': '#FF4500', 'G': '#00FA9A'}
    for i, stack in enumerate(state.stacks):
        for j, block in enumerate(stack.blocks):
            heuristic_value = global_heuristic_values.get(block, 0)
            plt.bar(i, 1, align='center', color=block_colors.get(block, 'gray'),
                    edgecolor='black', bottom=j, label=block if j == 0 else "")
            plt.text(i, j + 0.5, f'{block}\nH={heuristic_value}',
                     ha='center', va='center', color='black', fontsize=12, fontweight='bold')
    plt.xticks(np.arange(num_stacks), [f'Stack {i}' for i in range(num_stacks)])
    plt.yticks(np.arange(max_height + 1))
    plt.xlabel('Stacks')
    plt.ylabel('Block Height')
    plt.title('Stack Visualization with Global Heuristic Values')
    plt.ylim(0, max_height + 1)
    total_heuristic_value = calculate_total_heuristic()
    plt.figtext(0.99, 0.01, f'Total Heuristic Value: {total_heuristic_value}',
                horizontalalignment='right', verticalalignment='bottom',
                fontsize=12, color='black', bbox=dict(facecolor='white', alpha=0.8))
    plt.tight_layout()
    plt.show()
def solve_block_stacking(initial_state):
    goal_stack = 6  # Target stack index
    temp_stack = 3  # Temporary stack index
    print("\n--- Phase 1: Leveling Blocks to Temporary Stack ---")
    for block in ['B', 'C', 'D', 'F', 'E', 'A']:
        current_stack = find_block(initial_state, block)
        while current_stack != goal_stack and initial_state.stacks[current_stack].peek() != block:
            top_block = initial_state.stacks[current_stack].peek()
```

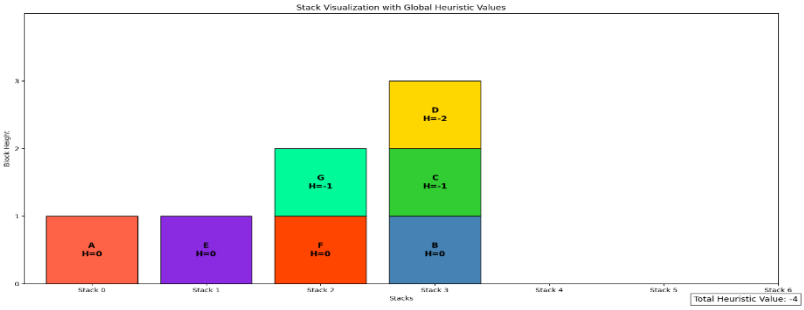
```
            move_block(initial_state, top_block, temp_stack)
        move_block(initial_state, 'G', goal_stack)
        for block in ['F', 'E', 'D', 'C', 'B', 'A']:
            move_block(initial_state, block, goal_stack)
        print("\nGoal state reached!")
        return initial_state
initial_stacks = [
    Stack(['A']),              # Stack 0 (A on ground)
    Stack(['E']),              # Stack 1 (E on ground)
    Stack(['B', 'C', 'D', 'G', 'F']),  # Stack 2 (BCDGF stacked)
    Stack(),                   # Stack 3 (temporary)
    Stack(),                   # Stack 4
    Stack(),                   # Stack 5
    Stack()                    # Stack 6 (goal)
]
initial_state = State(initial_stacks)
print("Initial state:", initial_state)
update_global_heuristic(initial_state)
visualize_state(initial_state)
final_state = solve_block_stacking(initial_state)
print("Final state:", final_state)
```

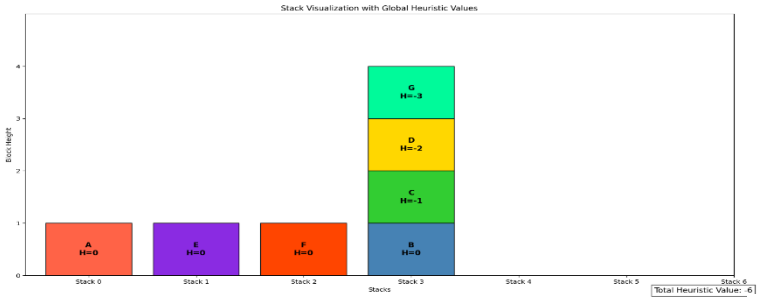
Output:

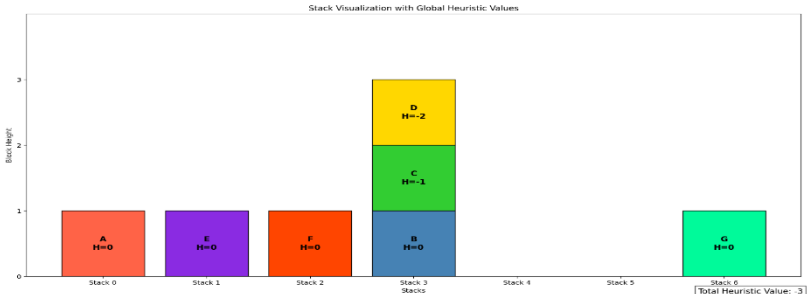

Stack Visualization with Global Heuristic Values
Block Height
Stacks
A H=0
E H=0
F H=0
G H=-1
D H=-2
C H=-3
B H=-4
Total Heuristic Value: -10

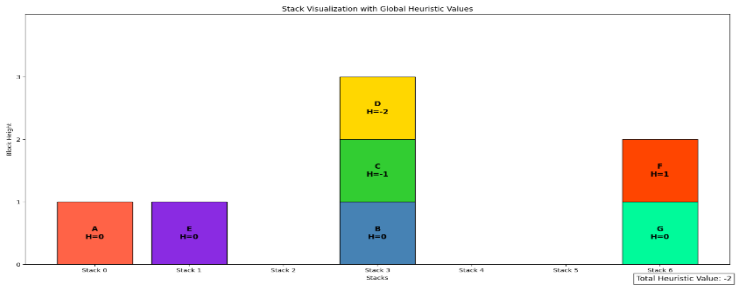

Stack Visualization with Global Heuristic Values
Block Height
Stacks
A H=0
E H=0
F H=0
G H=-1
D H=-2
C H=-3
B H=0
Total Heuristic Value: -6

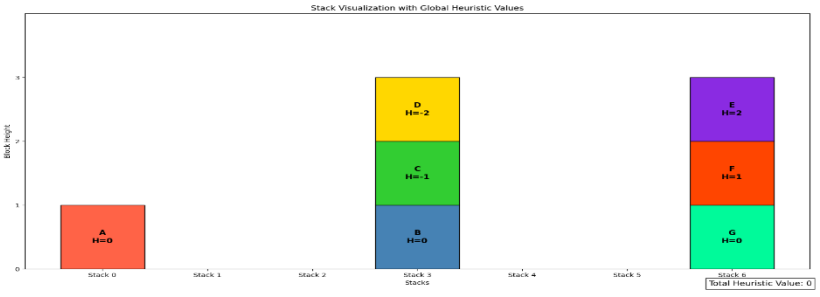

Stack Visualization with Global Heuristic Values
Block Height
Stacks
A H=0
E H=0
F H=0
G H=-1
D H=-2
C H=-1
B H=0
Total Heuristic Value: -4

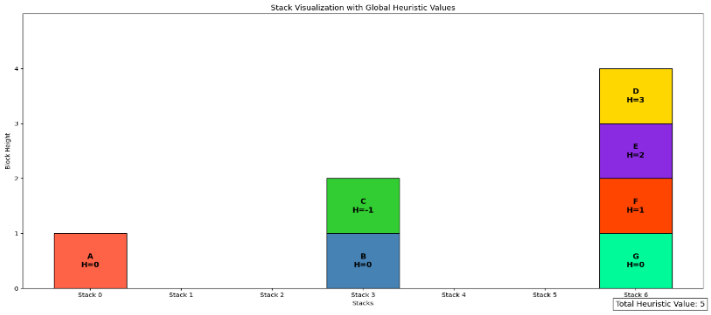

Stack Visualization with Global Heuristic Values
Block Height
Stacks
A H=0
E H=0
F H=0
G H=-1
D H=-2
C H=-1
B H=0
Total Heuristic Value: -4

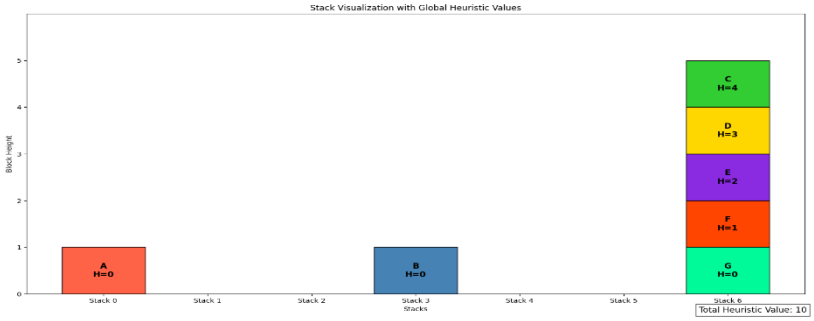

Stack Visualization with Global Heuristic Values
Block Height
Stacks
A H=0
E H=0
F H=0
G H=-3
D H=-2
C H=-1
B H=0
Total Heuristic Value: -6

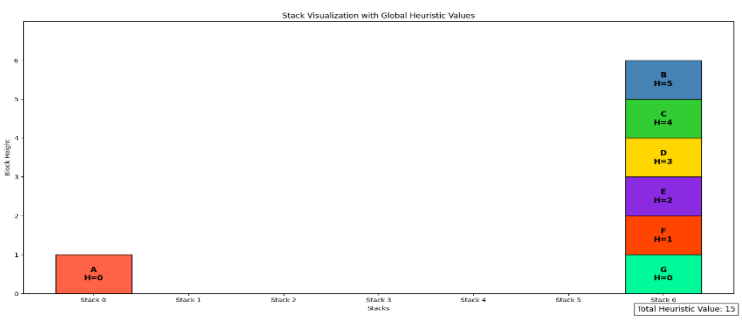

Stack Visualization with Global Heuristic Values
Block Height
Stacks
A H=0
E H=0
F H=0
D H=-2
C H=-1
B H=0
G H=0
Total Heuristic Value: -3

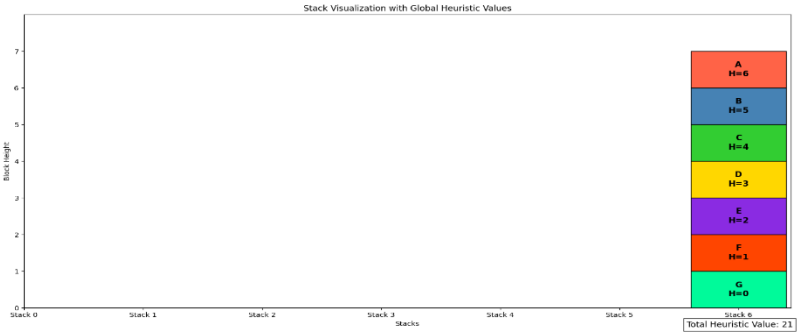

Stack Visualization with Global Heuristic Values
Block Height
Stacks
A H=0
E H=0
D H=-2
C H=-1
B H=0
F H=1
G H=0
Total Heuristic Value: -2


Stack Visualization with Global Heuristic Values
Block Height
Stacks
A H=0
D H=-2
C H=-1
B H=0
E H=2
F H=1
G H=0
Total Heuristic Value: 0


Stack Visualization with Global Heuristic Values
Block Height
Stacks
A H=0
C H=-1
B H=0
D H=3
E H=2
F H=1
G H=0
Total Heuristic Value: 5


Stack Visualization with Global Heuristic Values
Block Height
Stacks
A H=0
B H=0
C H=4
D H=3
E H=2
F H=1
G H=0
Total Heuristic Value: 10


Stack Visualization with Global Heuristic Values
Block Height
Stacks
A H=0
B H=5
C H=4
D H=3
E H=2
F H=1
G H=0
Total Heuristic Value: 15


Stack Visualization with Global Heuristic Values
Block Height
Stacks
A H=6
B H=5
C H=4
D H=3
E H=2
F H=1
G H=0
Total Heuristic Value: 21