Informed Hill Climb

class HillClimbing:

def \_\_init\_\_(self, start, goal, heuristic, neighbors):

self.start = start

self.goal = goal

self.heuristic = heuristic # A dictionary of heuristic values for each state

self.neighbors = neighbors # A function that returns neighboring states of a given state

def search(self):

current\_state = self.start

while current\_state != self.goal:

neighbors = self.neighbors(current\_state)

if not neighbors:

print("No solution found. Stuck in a local optima!")

return None

# Select the best neighbor based on the heuristic value

best\_neighbor = min(neighbors, key=lambda state: self.heuristic[state])

# If no better neighbor exists, exit the loop

if self.heuristic[best\_neighbor] >= self.heuristic[current\_state]:

print("Stuck in a local optima!")

return None

print(f"Moving from {current\_state} to {best\_neighbor}")

current\_state = best\_neighbor

print(f"Goal {self.goal} reached!")

return current\_state

# User input

def neighbors(state):

state\_neighbors = {}

n = int(input("Enter the number of states: "))

for \_ in range(n):

state\_name = input(f"Enter state name: ")

neighbors\_list = input(f"Enter neighbors of {state\_name} (space-separated): ").split()

state\_neighbors[state\_name] = neighbors\_list

return state\_neighbors.get(state, [])

def get\_heuristic():

heuristic = {}

n = int(input("Enter the number of states: "))

for \_ in range(n):

state\_name = input(f"Enter state name: ")

heuristic\_value = int(input(f"Enter heuristic value for {state\_name}: "))

heuristic[state\_name] = heuristic\_value

return heuristic

# Get inputs from user

start = input("Enter the start state: ")

goal = input("Enter the goal state: ")

heuristic = get\_heuristic()

hill\_climbing = HillClimbing(start=start, goal=goal, heuristic=heuristic, neighbors=neighbors)

# Perform search to reach the goal

result = hill\_climbing.search()

if result:

print(f"Path to goal: {result}")

else:

print("No solution found.")

OUTPUT:

Enter the start state: A

Enter the goal state: F

Enter the number of states: 6

Enter state name: A

Enter neighbors of A (space-separated): B C

Enter state name: B

Enter neighbors of B (space-separated): A D E

Enter state name: C

Enter neighbors of C (space-separated): A F

Enter state name: D

Enter neighbors of D (space-separated): B

Enter state name: E

Enter neighbors of E (space-separated): B F

Enter state name: F

Enter neighbors of F (space-separated): C E

Enter state name: A

Enter heuristic value for A: 5

Enter state name: B

Enter heuristic value for B: 4

Enter state name: C

Enter heuristic value for C: 4

Enter state name: D

Enter heuristic value for D: 3

Enter state name: E

Enter heuristic value for E: 2

Enter state name: F

Enter heuristic value for F: 1

Moving from A to B

Moving from B to E

Moving from E to F

Goal F reached!

Path to goal: F