**DATE:01.08.25**

**TASK 2**

**Hill climbing algorithm for Heuristic search**

Implementation of Hill climbing algorithm for Heuristic search approach using following constraints in python.

i. Create a function generating all neighbours of a solution

ii. Create a function calculating the length of a route

iii. Create a random solution generator

iv. Create a Travelling salesman problem

**Tools- Python, Online Simulator -** [**https://graphonline.ru/en/**](https://graphonline.ru/en/)

**PROBLEM STATEMENT: CO1 S3**

Imagine a mountain climber trying to reach the highest point of a mountain range. The terrain is represented as a 1D array of elevations (like hill heights at different points). The climber starts at a random position and uses the hill climbing heuristic to move only to higher neighboring positions. The goal is to find the local or global maximum elevation

**DATE:01.08.25**

**TASK-2**

**Implementation of Hill climbing algorithm for Heuristic search approach**

**AIM**

To implement the Hill Climbing algorithm as a Heuristic Search technique for solving optimization problems, where the objective is to find the best possible solution (maximum or minimum) based on a heuristic value.

**ALGORITHM**

1. Start at a random position on the terrain.
2. Check the neighboring positions (left and right).
3. Compare the elevation of the current position with neighbors.
4. Move to the neighbor with the highest elevation, if it’s higher than the current one.
5. Repeat steps 2–4 until no neighbor has a higher elevation.
6. Stop – you’ve reached a peak (highest nearby point).

**PROGRAM**

**Hill Climbing for Peak Finding**

import random

# i. Function to generate neighbors (left and right positions)

def generate\_neighbors(position, terrain):

neighbors = []

if position > 0:

neighbors.append(position - 1)

if position < len(terrain) - 1:

neighbors.append(position + 1)

return neighbors

# ii. Heuristic function: height of the terrain at a given position

def heuristic(position, terrain):

return terrain[position]

# iii. Random starting point

def get\_random\_position(terrain):

return random.randint(0, len(terrain) - 1)

# Hill Climbing Algorithm

def hill\_climbing(terrain):

current\_position = get\_random\_position(terrain)

current\_value = heuristic(current\_position, terrain)

print(f"Starting at position {current\_position} with elevation {current\_value}")

while True:

neighbors = generate\_neighbors(current\_position, terrain)

best\_neighbor = current\_position

best\_value = current\_value

for neighbor in neighbors:

neighbor\_value = heuristic(neighbor, terrain)

if neighbor\_value > best\_value:

best\_value = neighbor\_value

best\_neighbor = neighbor

if best\_value == current\_value:

break # No better neighbor found — local maximum

else:

current\_position = best\_neighbor

current\_value = best\_value

print(f"Moving to position {current\_position} with elevation {current\_value}")

print(f"Reached peak at position {current\_position} with elevation {current\_value}")

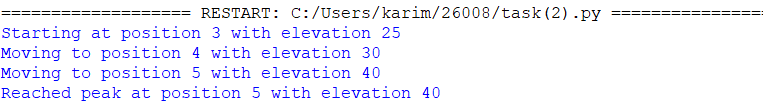
# Example terrain (elevations at different points)

terrain = [10, 20, 15, 25, 30, 40, 35, 25, 50, 45]

# Run the algorithm

hill\_climbing(terrain)

**OUTPUT**



**RESULT**

Thus the implementation of Hill Climbing algorithm as a Heuristic Search technique for solving optimization problems problem using python was successfully executed and output was verified.