

Worksheet 3.1

Student Name: Rydem

Branch: MCA (AI & ML)

Semester: I

Subject Name: Artificial Intelligence Tools - I

UID: 25MCI10275

Section/Group: MAM-1 (A)

Date of Performance: 27/10/25

Subject Code: 25CAP-614

1. Aim/Overview of the practical:

Experiment-1:

Implementing Hill Climbing Algorithm

2. Coding:

EXAMPLE-1

```
import matplotlib.pyplot as plt
import networkx as nx

def hill_climbing(graph, heuristic, start, goal):
    path = [start]
    current = start
    while current != goal:
        neighbors = graph[current]
        if not neighbors:
            print("No path found")
            return path
        next_node = min(neighbors, key=lambda x: heuristic[x])
        if heuristic[next_node] >= heuristic[current]:
            print("Local optimum reached or dead end.")
            break
        path.append(next_node)
        current = next_node
    return path
```

```

path.append(next_node)
current = next_node
return path

graph = {
    'A': ['B', 'C'],
    'B': ['D', 'E', 'G'],
    'C': ['F'],
    'D': [],
    'E': ['G'],
    'F': ['G'],
    'G': []
}

heuristic = {
    'A': 7,
    'B': 6,
    'C': 2,
    'D': 3,
    'E': 1,
    'F': 1,
    'G': 0
}

solution_path = hill_climbing(graph, heuristic, 'A', 'G')
print("Solution Path:", solution_path)

```

```

def visualize_graph(graph, heuristic, path):
    G = nx.DiGraph()

    for node, neighbors in graph.items():
        for neighbor in neighbors:
            G.add_edge(node, neighbor)

```

```

pos = nx.spring_layout(G, seed=42) # consistent layout

node_colors = []
for node in G.nodes():
    if node in path:
        node_colors.append("lightgreen")
    else:
        node_colors.append("lightcoral")

plt.figure(figsize=(8, 6))
nx.draw(
    G, pos,
    with_labels=True,
    node_color=node_colors,
    node_size=1500,
    font_size=12,
    font_weight="bold",
    arrows=True,
    arrowstyle="->",
    arrowsize=20
)

labels = {n: f"{n}\n{h={heuristic[n]}}" for n in G.nodes()}
nx.draw_networkx_labels(G, pos, labels=labels, font_size=10, font_color="black")

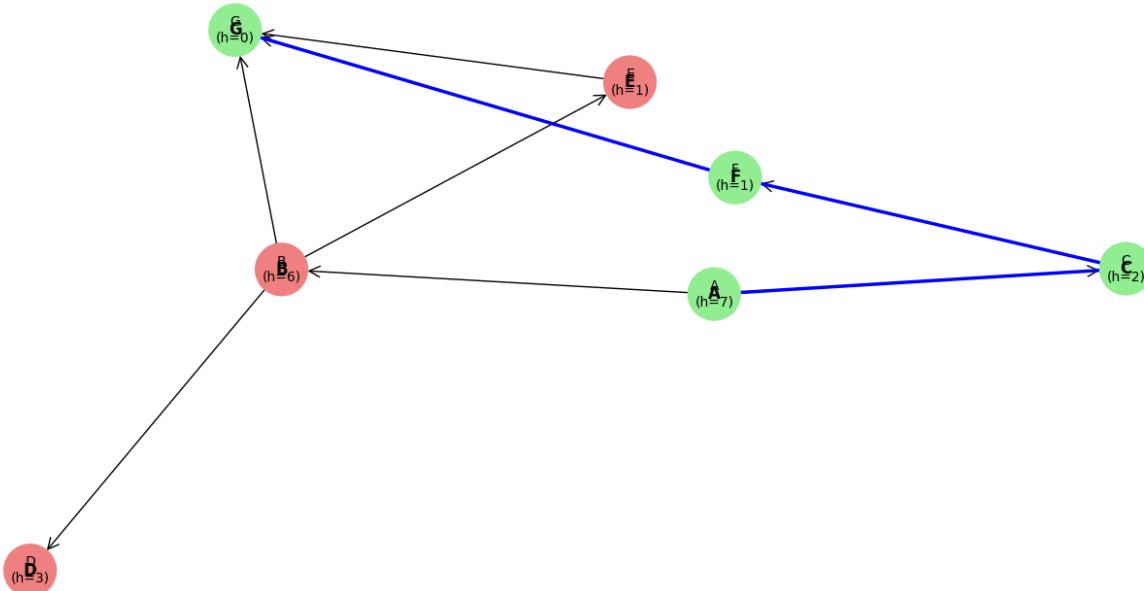
path_edges = list(zip(path, path[1:]))
nx.draw_networkx_edges(G, pos, edgelist=path_edges, edge_color="blue", width=2.5)

plt.title("Hill Climbing Algorithm Visualization", fontsize=14)
plt.show()
visualize_graph(graph, heuristic, solution_path)

```

OUTPUT:-

Figure 1



File ← → | + Q E |

(x, y) = (-0.1378, 0.701)

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

PS E:\CLASS CODING> & C:/Users/Dell/AppData/Local/Programs/Python/Python310/python.exe "e:/CLASS CODING/AI/hill_climbing_vis.py"
Solution Path: ['A', 'C', 'F', 'G']

EXAMPLE-2

```
import random

def objective_function(x):
    return -(x - 3) **2 + 9

def hill_climb(start_x, step_size=0.1, max_iteration=100):
    current_x = start_x
    current_value= objective_function(current_x)

    for _ in range (max_iteration):
        next_x = current_x + random.choice([-step_size, step_size])
        next_value = objective_function(next_x)

        if next_value > current_value:
            current_x, current_value = next_x, next_value
        else:
            break

    return current_x, current_value

start_x= random.uniform(0,6)
best_x, best_value = hill_climb(start_x)

print(f"Best x found: {best_x:2f}, Maximum value: {best_value:2f}")
```

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

PS E:\CLASS CODING> & C:/Users/Dell/AppData/Local/Programs/Python/Python310/python.exe "e:/CLASS CODING/AI/hill_climbing2.py"
Best x found: 0.155643, Maximum value: 0.909633
PS E:\CLASS CODING>
```



DEPARTMENT OF ACADEMIC AFFAIRS

Discover. Learn. Empower.



3. Learning outcomes (What I have learnt):

- Understand the concept of heuristic-based local search and how Hill Climbing uses heuristic values to iteratively move toward the goal state.
- Differentiate between Hill Climbing and uninformed search algorithms, recognizing its reliance on heuristics and lack of backtracking or path cost consideration.
- Learn how Hill Climbing selects the next state based on the best heuristic value and understand its limitations, such as getting stuck in local maxima, plateaus, or ridges.
- Gain hands-on experience in implementing and visualizing the Hill Climbing algorithm for optimization and pathfinding problems in areas like AI, game development, and robotics.
- Develop skills to analyze algorithm behavior and performance, including detecting local optima and verifying the effectiveness of heuristic functions.



DEPARTMENT OF ACADEMIC AFFAIRS

Discover. Learn. Empower.

NAAC
GRADE A+
ACCREDITED UNIVERSITY