

Worksheet 2.1

Student Name: Ayush Yadav

Branch: MCA (AI & ML)

Semester: I

Subject Name: Artificial Intelligence Tools - I

UID: 25IMC13004

Section/Group: MAM-1 (A)

Date of Performance: 15/09/25

Subject Code: 25CAP-614

1. Aim/Overview of the practical:

Experiment-2:

To implement the Best-First Search algorithm in Python using a priority queue and heuristics, in order to find the optimal path from a given start node to the goal node in a graph.

2. Coding:

EXAMPLE-1

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

def best_first_search(graph, start, goal, heuristic):
    visited = set()
    pq = []
    heapq.heappush(pq, (heuristic[start], start))
    parent = {start: None}

    while pq:
        cost, current_node = heapq.heappop(pq)

        if current_node == goal:
```

```
path = []
while current_node is not None:
    path.append(current_node)
    current_node = parent[current_node]
return path[::-1]
```

```
if current_node not in visited:
    visited.add(current_node)
    for neighbour in graph[current_node]:
        if neighbour not in visited:
            parent[neighbour] = current_node
            heapq.heappush(pq, (heuristic[neighbour], neighbour))
```

```
return None
```

```
# Graph definition
```

```
Graph = {
    'A': ['B', 'C'],
    'B': ['A', 'D', 'E'],
    'C': ['A', 'F'],
    'D': ['B'],
    'E': ['B', 'F'],
    'F': ['C', 'E']
}
```

```
heuristic = {
```

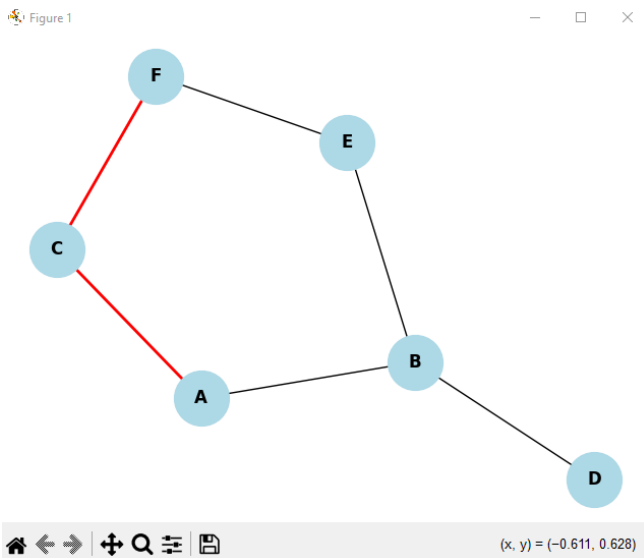
```
    'A': 5,
    'B': 4,
    'C': 2,
    'D': 6,
    'E': 1,
    'F': 0
}
```

```
# Run the search
path = best_first_search(Graph, 'A', 'F', heuristic)
print("Best First Search Path:", path)

# Visualize the graph
G = nx.Graph()
for node, neighbors in Graph.items():
    for neighbor in neighbors:
        G.add_edge(node, neighbor)

pos = nx.spring_layout(G, seed=42)
nx.draw(G, pos, with_labels=True, node_color='lightblue', node_size=1500,
        font_size=12, font_weight='bold')
if path:
    nx.draw_networkx_edges(G, pos,
        edgelist=[(path[i], path[i+1]) for i in range(len(path)-1)],
        edge_color='red', width=2)
plt.title("Graph Visualization with Best First Search Path")
plt.show()
```

OUTPUT:-



EXAMPLE-2

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

def best_first_search(graph, start, goal, heuristic):
    visited = set()
    pq = []
    heapq.heappush(pq, (heuristic[start], start))
    parent = {start: None}

    while pq:
        _, current = heapq.heappop(pq)

        if current == goal:
            path = []
            while current:
                path.append(current)
                current = parent[current]
            return path[::-1]

        visited.add(current)

        for neighbour in graph[current]:
            if neighbour not in visited and neighbour not in parent:
                parent[neighbour] = current
                heapq.heappush(pq, (heuristic[neighbour], neighbour))

    return None
```

Graph definition

```
Graph = {  
    'X': ['Y', 'A', 'H'],  
    'Y': ['X', 'A', 'M', 'Z'],  
    'Z': ['Y', 'N'],  
    'A': ['X', 'Y', 'M'],  
    'M': ['Y', 'A', 'P'],  
    'N': ['Z', 'P'],  
    'H': ['X', 'O', 'R'],  
    'O': ['H', 'R'],  
    'P': ['M', 'N', 'R', 'S'],  
    'R': ['H', 'O', 'P', 'T', 'K'],  
    'S': ['P', 'K'],  
    'T': ['R', 'K', 'V'],  
    'V': ['T', 'K'],  
    'K': ['R', 'T', 'V', 'S']  
}
```

Heuristic values

```
heuristic = {  
    'X': 5,  
    'Y': 10,  
    'Z': 13,  
    'A': 4,  
    'M': 6,  
    'N': 9,  
    'H': 6,  
    'O': 7,  
    'R': 3,  
    'P': 8,  
    'S': 4,  
    'T': 7,  
    'V': 4,  
    'K': 0
```



```
}
```

```
# Run Best First Search
```

```
path = best_first_search(Graph, 'X', 'K', heuristic)
```

```
print("Best First Search Path:", path)
```

```
# Visualization
```

```
G = nx.Graph()
```

```
for node, neighbors in Graph.items():
```

```
    for neighbor in neighbors:
```

```
        G.add_edge(node, neighbor)
```

```
pos = nx.spring_layout(G, seed=42)
```

```
nx.draw(G, pos, with_labels=True, node_color='lightblue', node_size=800, font_size=10)
```

```
if path:
```

```
    path_edges = list(zip(path, path[1:]))
```

```
    nx.draw_networkx_nodes(G, pos, nodelist=path, node_color='orange')
```

```
    nx.draw_networkx_edges(G, pos, edgelist=path_edges, edge_color='red', width=2)
```

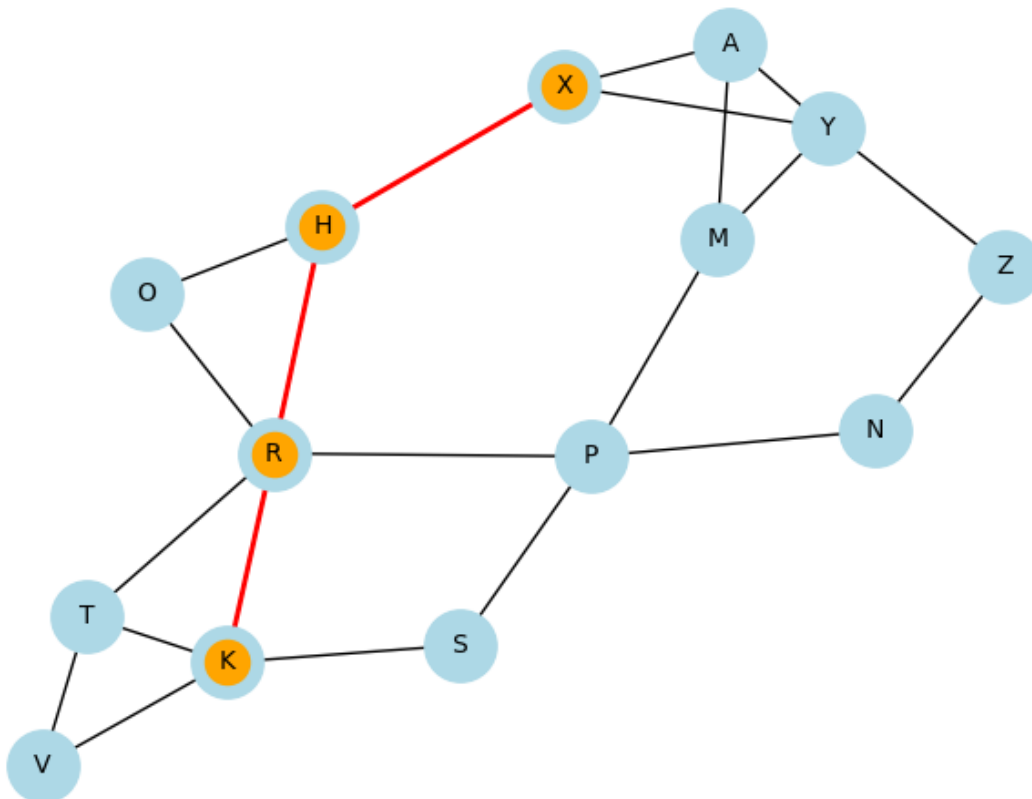
```
plt.title("Best First Search Path from 'X' to 'K'")
```

```
plt.show()
```

(B). OUTPUT:

```
PS E:\CLASS CODING> & C:/Users/Dell/AppData/Local/Programs/Python/Python310/python.exe "e:/CLASS CODING/AI/bestfs1.py"  
Best First Search Path: ['X', 'H', 'R', 'K']
```

Figure 1



(x, y) = (0.492, -0.008)

3. Learning outcomes (What I have learnt):

- Successfully implemented the Best-First Search algorithm in Python using a priority queue.
- Represented the given graph with 13 vertices and 12 edges using adjacency lists.
- Applied heuristic values to guide the search process efficiently from the start node to the goal node.
- Obtained the optimal path ($X \rightarrow \dots \rightarrow U$) between the start and goal nodes based on heuristics.
- Understood how heuristic-based search reduces search space and improves efficiency compared to blind search methods like BFS or DFS.



DEPARTMENT OF ACADEMIC AFFAIRS

Discover. Learn. Empower.

NAAC
GRADE **A+**
ACCREDITED UNIVERSITY



DEPARTMENT OF ACADEMIC AFFAIRS

Discover. Learn. Empower.

NAAC
GRADE **A+**
ACCREDITED UNIVERSITY