Shaheed Zulfikar Ali Bhutto Institute of Science & Technology

COMPUTER SCIENCE DEPARTMENT

Total Marks: 04

Obtained Marks: _____

Artificial Intelligence (Lab)

Task # 01
Section-D

Submitted To: Mam Khadija

Student Name: Syeda Anabia Wahid

Reg Number: 2212248

Question no:01

Implement GBFS to find the shortest path between a start node and a goal node.

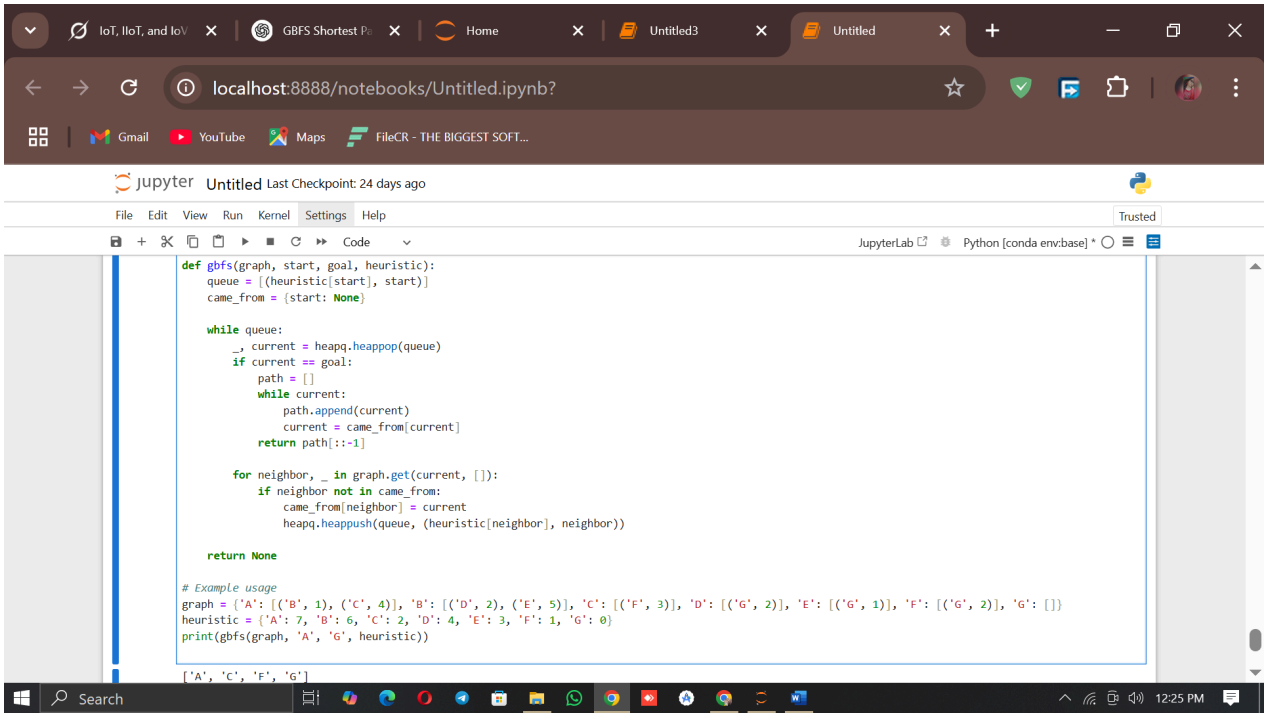
Create a document containing:

- Code (Can be screenshot of implemented code)
- Output (Screenshot)

Code:

```
import heapq

def gbfs(graph, start, goal, heuristic):
    queue = [(heuristic[start], start)]
    came_from = {start: None}

    while queue:
        _, current = heapq.heappop(queue)
        if current == goal:
            path = []
            while current:
                path.append(current)
                current = came_from[current]
            return path[::-1]

        for neighbor, _ in graph.get(current, []):
            if neighbor not in came_from:
                came_from[neighbor] = current
                heapq.heappush(queue, (heuristic[neighbor], neighbor))

    return None

# Example usage
graph = {'A': [('B', 1), ('C', 4)], 'B': [('D', 2), ('E', 5)], 'C': [('F', 3)], 'D': [('G', 2)], 'E': [('G', 1)], 'F':
[('G', 2)], 'G': []}
heuristic = {'A': 7, 'B': 6, 'C': 2, 'D': 4, 'E': 3, 'F': 1, 'G': 0}
print(gbfs(graph, 'A', 'G', heuristic))
```

Output:


IoT, IIoT, and IoV
GBFS Shortest Pa
Home
Untitled3
Untitled
localhost:8888/notebooks/Untitled.ipynb?
Gmail
YouTube
Maps
FileCR - THE BIGGEST SOFT...
jupyter Untitled Last Checkpoint: 24 days ago
File Edit View Run Kernel Settings Help
Trusted
Code
JupyterLab
Python [conda env:base]
def gbfs(graph, start, goal, heuristic):
queue = [(heuristic[start], start)]
came_from = {start: None}
while queue:
_, current = heapq.heappop(queue)
if current == goal:
path = []
while current:
path.append(current)
current = came_from[current]
return path[::-1]
for neighbor, _ in graph.get(current, []):
if neighbor not in came_from:
came_from[neighbor] = current
heapq.heappush(queue, (heuristic[neighbor], neighbor))
return None
# Example usage
graph = {'A': [('B', 1), ('C', 4)], 'B': [('D', 2), ('E', 5)], 'C': [('F', 3)], 'D': [('G', 2)], 'E': [('G', 1)], 'F': [('G', 2)], 'G': []}
heuristic = {'A': 7, 'B': 6, 'C': 2, 'D': 4, 'E': 3, 'F': 1, 'G': 0}
print(gbfs(graph, 'A', 'G', heuristic))
['A', 'C', 'F', 'G']
Search
12:25 PM