SCHOOL OF ENGINEERING

INTRODUCTION TO AIML

ASSIGNMENT-1

BOTBRAIN: A SMART CAMPUS NAVIGATOR

SUBMITTED BY

THANUSHREE G M

24UG00562

'B'- SECTION

BotBrain: A smart campus navigator (week-1 report)

Abstract

BotBrain is a smart campus navigator which helps in assisting visitors and students to explore the university campus with ease. By making a campus map/ model and using some simple algorithms, it is easier to find quick route from one building to the other. Using this visitors and students can get to go easily without any kind of confusions

Introduction

BotBrain is a smart campus navigator which guides visitors and new students to move around Chanakya University. Sometimes it might be confusing to move around the campus and find the best route among the multiple routes. By representing the campus map as connections of buildings and paths. This project helps them to find the best and shortest route to reach their destination in campus. This uses some search algorithms and provides the route from building to building.

Problem Statement

Visitors, newcomers and some students face difficulty to move around large campus. It might be confusing for them to find the easier path from building to building. So to solve this problem there is a need of smart navigator which helps in guiding them across the campus, provide easy to follow directions and provide relevant information about the particular buildings. Through this project we build a simple smart navigator which uses simple algorithms to search the best path and accurate map of the campus, thereby assisting the visitors, newcomers and students to move around the campus with ease without facing any kind of difficulty.

Objectives

- Create a precise and unambiguous digital map of Chanakya University campus with major buildings and pedestrian walkways.
- Apply effective search algorithms (BFS, DFS, UCS, A*) to discover paths between two points on the campus.
- Build a user-friendly interface to get navigation routes and basic building details.
- Analyze the performance of implemented algorithms based on path length and explored nodes.
- Describe the intelligent agent principles under BotBrain, namely environment, agent type, and decision-making.
- Optional add-on features include visual maps, building services information, alternative routing, or suggestions.

Scope

The scope of this project is to design and develop a smart campus navigation system for Chanakya University. It includes creating a digital map of major campus buildings and pathways, using simple search algorithms to find efficient routes, and providing users with navigation assistance and basic information about locations. The system focuses on outdoor campus navigation and determine to improve accessibility for students, and visitors. While primarily targeting walking routes between buildings, the system can be extended later with features like indoor navigation and visual route displays. This project aims to enhance the overall campus experience by reducing confusion and saving time in daily navigation.

Literature

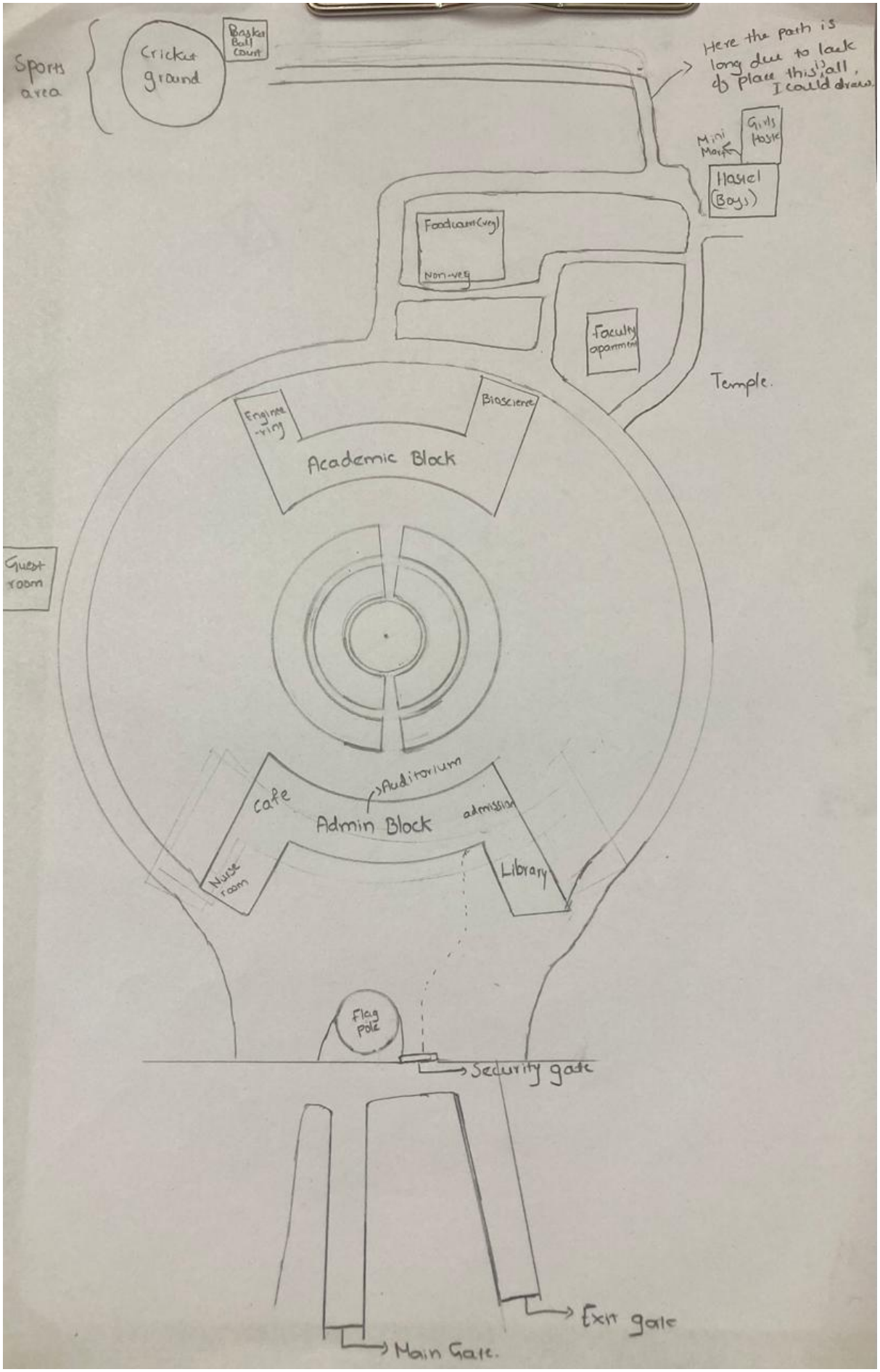
Several universities have designed intelligent navigation systems to enable students to navigate more conveniently around campus. The applications usually incorporate digital maps and intelligent pathfinding algorithms to lead the users between buildings. Some further incorporate mobile applications with QR codes. What they all have in common is simplifying navigation and offering helpful information on campus services. Similar to BotBrain, they intend to decrease the stress of being lost in the campus and ease the people to walk around the large campus.

Required documents

1. Campus Entities:

   Here is the list of the key locations/buildings in the campus map for BotBrain

   - Main gate
   - Exit gate
   - Security gate
   - Admin block
   - Library
   - Medical center
   - Cafeteria
   - Auditorium
   - Academic block
   - Food court
   - Hostel
   - Mart
   - Sports area


Sports area
Cricket ground
Basket Ball court
Here the path is long due to lack of place this is all, I could draw.
Mini Mart
Girls Hostel
Hostel (Boys)
Food court (veg)
Non-veg
Faculty apartment
Temple.
Engineering
Bioscience
Academic Block
Guest room
Cafe
Auditorium
Admin Block
admission
Nurse room
Library
Flag pole
Security gate
Exit gate
Main Gate.

2. Physical Connections and distance:
   Walkways between these buildings with direct access between them will be charted with approximated distances in meters. Distances will be based on Google Maps or rough campus information. Paths can be one-way or have varying walking speeds.

3. Building information:
   This will contain some basic information about the buildings (it maybe the opening and closing time or facilities available in the building or information about classes or labs)

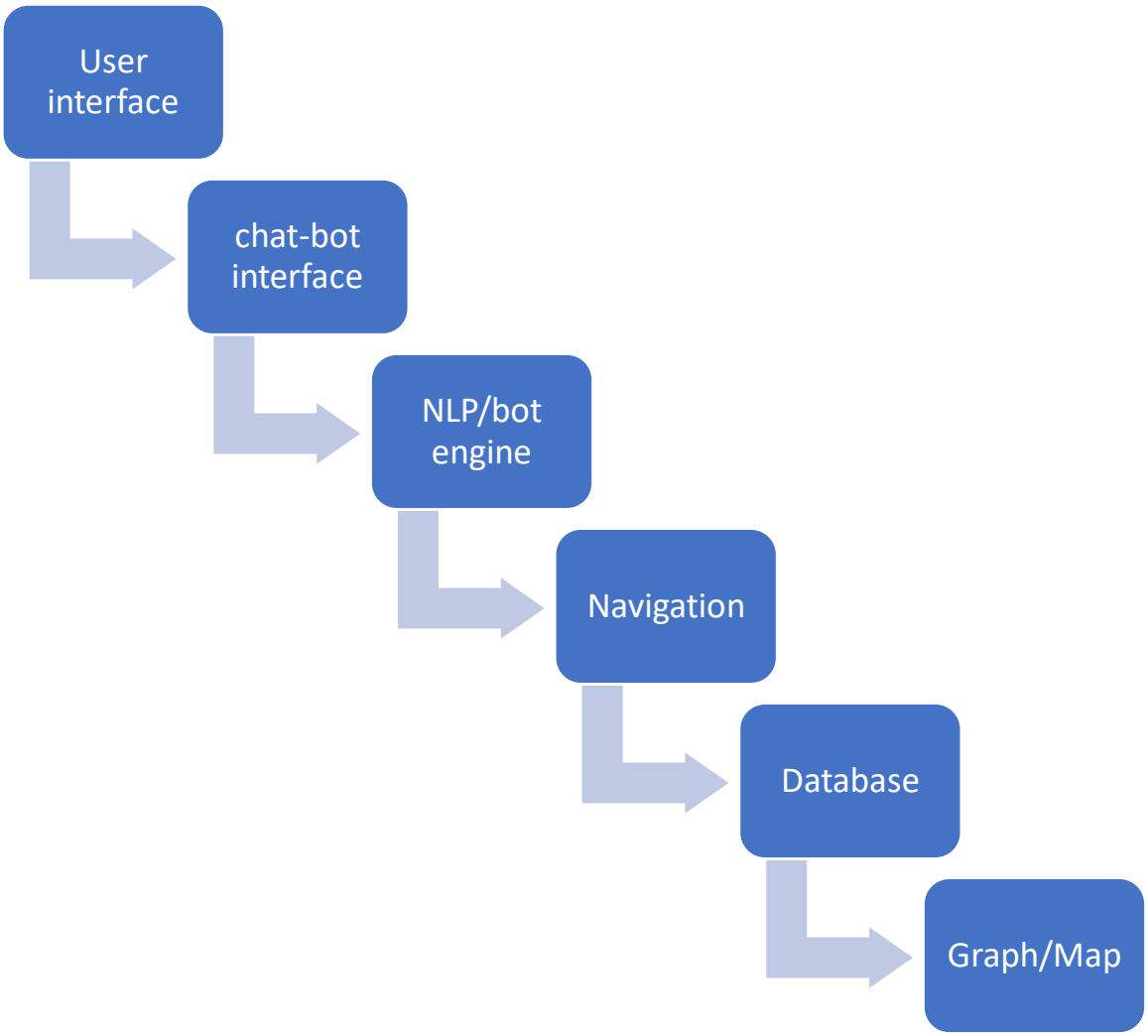
4. System Features:
   The system will accommodate queries like "Show route from Hostel to Library" and show route information such as path, total walking distance, and estimated time. Different search algorithms for pathfinding can also be chosen by users.

Tools and Technology

- Python: Programming language to implement search algorithms such as BFS, DFS, UCS and A*.
- Google maps: To find the accurate distance between the buildings in the campus and helps in visual mapping.
- GitHub: It helps in interpreting the code and organizing the project.
- PowerPoint: To make presentation slides.
- Tkinter / PySimpleGUI / Flask: Tkinter is used to create desktop applications using Python. PySimpleGUI takes less code. Flask is for creating small web applications.
- Later on I may include some other tools as per the requirement.

Week 2 Report- BotBrain: A Smart Campus Navigator System Design and Architecture

Flowchart



Module and Feature Definitions

User Interface – Takes input of queries/location from the user and shows the result. (i.e., shows the best route and gives the information about the building they are looking for)

NLP/ Bot Engine – Processes and understands the user Requirements and extracts the details like location and some relevant information.

Navigation – Uses campus graph and some search algorithms like BFS, UCS, A* to find the shortest path between the quires

Database – stores campus locations and the distance from every location, contains building information and availabities and the connections (paths)

Methodology

In order to construct the campus navigation system, the first step is to sketch a map of the campus as a network—every building became node, and every walkway or road between buildings became a line (edge) of its distance.

Everything relating to the buildings, timings and some more important information is saved in a campus database. When a person enters a question or request into the chatbot (such as "Show path from Hostel to Library"), the system initially attempts to interpret what they are asking for using simple language processing.

In case the query is regarding locating a path, BotBrain uses search algorithms (such as BFS, UCS, or A*) to determine the optimal path—either moving the least number of steps or the smallest distance depending on the campus map. If the query is regarding the timing or building information, the BotBrain retrieves that data directly from its database.

At the end the solution is presented to the user- the route of buildings to take to go forward, the path, or information regarding a campus building. Therefore, all students, visitors are given quick help in navigating or information.

PEAS ANALYSIS

Performance

Providing the shortest path, displaying accurate time to reach the destination, providing information as per the queries of the user which satisfies the user.

Environment

Includes Chanakya university campus model, campus data (buildings, paths, facilities available and information of the buildings,..), campus map

Actuators

Responds to the user according to their queries by providing the shortest routes and instructions or providing the information about the buildings.

Sensors

Senses the user queries and accesses the database of the campus location and information before acting and looks for any updates on campus.(adding any kind of information)

➢ Campus graph code

```
campus_graph = {

    "Medical Center": {"Food Court": 405,"Hostel": 490,"Cafeteria": 90,"Exit Gate": 160,"Auditorium": 80,"Main Gate": 180,"Library": 75,"Admin Block": 80,"Academic Block": 85,"Security Gate": 90,"Sports Area": 840,"Mart": 510

    },

    "Mart": {"Exit Gate": 600,"Academic Block": 410,"Main Gate": 620,"Cafeteria": 585,"Security Gate": 520,"Food Court": 836,"Sports Area": 320,"Library": 465,"Hostel": 200,"Auditorium": 574,"Admin Block": 434,"Medical Center": 510

    },

    "Exit Gate": {"Main Gate": 100,"Academic Block": 250,"Cafeteria": 270,"Security Gate": 180,"Auditorium": 260,"Library": 240,"Hostel": 580,"Admin Block": 220,"Sports Area": 350,"Food Court": 609,"Medical Center": 160,"Mart": 600

    },

    "Hostel": {"Admin Block": 506,"Sports Area": 300,"Academic Block": 380,"Mart": 200,"Food Court": 816,"Library": 450,"Auditorium": 510,"Main Gate": 600,"Cafeteria": 520,"Medical Center": 490,"Security Gate": 500
```

},

"Cafeteria": {"Main Gate": 276,"Auditorium": 10,"Admin Block": 35,"Library": 63,"Exit Gate": 270,"Food Court": 355,"Academic Block": 27,"Security Gate": 130,"Medical Center": 90,"Mart": 585,"Hostel": 520,"Sports Area": 900

},

"Security Gate": {"Library": 100,"Exit Gate": 180,"Main Gate": 220,"Cafeteria": 130,"Academic Block": 240,"Auditorium": 120,"Medical Center": 90,

"Mart": 520,"Hostel": 500,"Sports Area": 210,"Food Court": 471,"Admin Block": 90

},

"Library": {"Auditorium": 3,"Food Court": 470,"Exit Gate": 240,"Admin Block": 16,"Main Gate": 275,"Mart": 465,"Academic Block": 46,"Medical Center": 75,"Hostel": 450,"Sports Area": 810,"Security Gate": 100,"Cafeteria": 63

},

"Sports Area": {"Admin Block": 740,"Cricket Ground": 350,"Main Gate": 580,"Hostel": 300,"Academic Block": 560,"Exit Gate": 350,"Library": 810,"Security Gate": 210,"Medical Center": 840,"Mart": 320,"Food Court": 430,"Cafeteria": 900

},

"Admin Block": {"Main Gate": 160,"Academic Block": 126,"Mart": 434,"Cafeteria": 35,"Exit Gate": 220,"Library": 16,"Security Gate": 80,"Hostel": 506,"Auditorium": 17,"Medical Center": 80,"Food Court": 355,"Sports Area": 740

},

"Auditorium": {"Food Court": 344,"Security Gate": 120,"Exit Gate": 260,"Library": 3,"Main Gate": 286,"Academic Block": 150,"Mart": 574,"Medical Center": 80,"Hostel": 510,"Sports Area": 830,"Admin Block": 17,"Cafeteria": 10

},

"Academic Block": {"Food Court": 192,"Security Gate": 240,"Exit Gate": 250,"Mart": 410,"Admin Block": 126,"Library": 46,"Hostel": 380,"Auditorium": 150,"Medical Center": 85,"Cafeteria": 27,"Sports Area": 560,"Main Gate": 210

},

"Food Court": {"Exit Gate": 609,"Library": 470,"Mart": 836,"Medical Center": 405,"Auditorium": 344,"Main Gate": 699,"Admin Block": 355,"Hostel": 816,"Academic Block": 192,"Security Gate": 471,"Cafeteria": 355,"Sports Area": 430

},

"Main Gate": {"Exit Gate": 100,"Security Gate": 150,"Admin Block": 160,"Academic Block": 210,"Library": 275,"Auditorium": 286,"Cafeteria": 276,"Food Court": 699,"Medical Center": 180,"Sports Area": 580,"Hostel": 600,"Mart": 620

```
    }
}
```

- A*

```
import heapq

campus_graph = {
    "Medical Center": {"Food Court": 405,"Hostel": 490,"Cafeteria": 90,"Exit Gate": 160,"Auditorium": 80,"Main Gate": 180,"Library": 75,"Admin Block": 80,"Academic Block": 85,"Security Gate": 90,"Sports Area": 840,"Mart": 510},

    "Mart": {"Exit Gate": 600,"Academic Block": 410,"Main Gate": 620,"Cafeteria": 585,"Security Gate": 520,"Food Court": 836,"Sports Area": 320,"Library": 465,"Hostel": 200,"Auditorium": 574,"Admin Block": 434,"Medical Center": 510},

    "Exit Gate": {"Main Gate": 100,"Academic Block": 250,"Cafeteria": 270,"Security Gate": 180,"Auditorium": 260,"Library": 240,"Hostel": 580,"Admin Block": 220,"Sports Area": 350,"Food Court": 609,"Medical Center": 160,"Mart": 600},

    "Hostel": {"Admin Block": 506,"Sports Area": 300,"Academic Block": 380,"Mart": 200,"Food Court": 816,"Library": 450,"Auditorium": 510,"Main Gate": 600,"Cafeteria": 520,"Medical Center": 490,"Security Gate": 500},

    "Cafeteria": {"Main Gate": 276,"Auditorium": 10,"Admin Block": 35,"Library": 63,"Exit Gate": 270,"Food Court": 355,"Academic Block": 27,"Security Gate": 130,"Medical Center": 90,"Mart": 585,"Hostel": 520,"Sports Area": 900},

    "Security Gate": {"Library": 100,"Exit Gate": 180,"Main Gate": 220,"Cafeteria": 130,"Academic Block": 240,"Auditorium": 120,"Medical Center": 90,"Mart": 520,"Hostel": 500,"Sports Area": 210,"Food Court": 471,"Admin Block": 90},

    "Library": {"Auditorium": 3,"Food Court": 470,"Exit Gate": 240,"Admin Block": 16,"Main Gate": 275,"Mart": 465,"Academic Block": 46,"Medical Center": 75,"Hostel": 450,"Sports Area": 810,"Security Gate": 100,"Cafeteria": 63},

    "Sports Area": {"Admin Block": 740,"Cricket Ground": 350,"Main Gate": 580,"Hostel": 300,"Academic Block": 560,"Exit Gate": 350,"Library": 810,"Security Gate": 210,"Medical Center": 840,"Mart": 320,"Food Court": 430,"Cafeteria": 900},

    "Admin Block": {"Main Gate": 160,"Academic Block": 126,"Mart": 434,"Cafeteria": 35,"Exit Gate": 220,"Library": 16,"Security Gate": 80,"Hostel": 506,"Auditorium": 17,"Medical Center": 80,"Food Court": 355,"Sports Area": 740},
```

```
    "Auditorium": {"Food Court": 344,"Security Gate": 120,"Exit Gate": 260,"Library": 3,"Main Gate": 286,"Academic Block": 150,"Mart": 574,"Medical Center": 80,"Hostel": 510,"Sports Area": 830,"Admin Block": 17,"Cafeteria": 10},

    "Academic Block": {"Food Court": 192,"Security Gate": 240,"Exit Gate": 250,"Mart": 410,"Admin Block": 126,"Library": 46,"Hostel": 380,"Auditorium": 150,"Medical Center": 85,"Cafeteria": 27,"Sports Area": 560,"Main Gate": 210},

    "Food Court": {"Exit Gate": 609,"Library": 470,"Mart": 836,"Medical Center": 405,"Auditorium": 344,"Main Gate": 699,"Admin Block": 355,"Hostel": 816,"Academic Block": 192,"Security Gate": 471,"Cafeteria": 355,"Sports Area": 430},

    "Main Gate": {"Exit Gate": 100,"Security Gate": 150,"Admin Block": 160,"Academic Block": 210,"Library": 275,"Auditorium": 286,"Cafeteria": 276,"Food Court": 699,"Medical Center": 180,"Sports Area": 580,"Hostel": 600,"Mart": 620}

}


def heuristic(node, goal):
    # No heuristic info, return 0
    return 0


def a_star(graph, start, goal):
    import heapq
    queue = []
    heapq.heappush(queue, (0 + heuristic(start, goal), 0, start, [start]))
    visited = set()

    while queue:
        est_total, cost_so_far, current, path = heapq.heappop(queue)

        if current == goal:
            return path, cost_so_far

        if current in visited:
            continue
        visited.add(current)
```

```
        for neighbor, dist in graph.get(current, {}).items():
            if neighbor not in visited:
                new_cost = cost_so_far + dist
                est = new_cost + heuristic(neighbor, goal)
                heapq.heappush(queue, (est, new_cost, neighbor, path + [neighbor]))

    return None, float('inf')

def main():
    print("Available locations:", list(campus_graph.keys()))
    start = input("Enter start location: ").strip()
    goal = input("Enter goal location: ").strip()

    if start not in campus_graph:
        print(f"Start location '{start}' not found in campus locations.")
        return
    if goal not in campus_graph:
        print(f"Goal location '{goal}' not found in campus locations.")
        return

    path, cost = a_star(campus_graph, start, goal)

    if path is None:
        print(f"No path found from {start} to {goal}.")
    else:
        print(f"Path from {start} to {goal}:")
        print(" -> ".join(path))
        print(f"Total distance: {cost} meters")
```

```
if __name__ == "__main__":
    main()
```

Output

```
Available locations: ['Medical Center', 'Mart', 'Exit Gate', 'Hostel', '
Cafeteria', 'Security Gate', 'Library', 'Sports Area', 'Admin Block', 'A
uditorium', 'Academic Block', 'Food Court', 'Main Gate']
Enter start location: Hostel
Enter goal location: Main Gate
Path from Hostel to Main Gate:
Hostel -> Academic Block -> Main Gate
Total distance: 590 meters
```

```
Available locations: ['Medical Center', 'Mart', 'Exit Gate', 'Hostel', '
Cafeteria', 'Security Gate', 'Library', 'Sports Area', 'Admin Block', 'A
uditorium', 'Academic Block', 'Food Court', 'Main Gate']
Enter start location: Academic Block
Enter goal location: Medical Center
Path from Academic Block to Medical Center:
Academic Block -> Medical Center
Total distance: 85 meters
```

➢ UCS

```
import heapq


campus_graph = {
    "Medical Center": {"Food Court": 405,"Hostel": 490,"Cafeteria": 90,"Exit Gate": 160,"Auditorium": 80,"Main Gate": 180,"Library": 75,"Admin Block": 80,"Academic Block": 85,"Security Gate": 90,"Sports Area": 840,"Mart": 510},
    "Mart": {"Exit Gate": 600,"Academic Block": 410,"Main Gate": 620,"Cafeteria": 585,"Security Gate": 520,"Food Court": 836,"Sports Area": 320,"Library": 465,"Hostel": 200,"Auditorium": 574,"Admin Block": 434,"Medical Center": 510},
    "Exit Gate": {"Main Gate": 100,"Academic Block": 250,"Cafeteria": 270,"Security Gate": 180,"Auditorium": 260,"Library": 240,"Hostel": 580,"Admin Block": 220,"Sports Area": 350,"Food Court": 609,"Medical Center": 160,"Mart": 600},
    "Hostel": {"Admin Block": 506,"Sports Area": 300,"Academic Block": 380,"Mart": 200,"Food Court": 816,"Library": 450,"Auditorium": 510,"Main Gate": 600,"Cafeteria": 520,"Medical Center": 490,"Security Gate": 500},
```

```
    "Cafeteria": {"Main Gate": 276,"Auditorium": 10,"Admin Block": 35,"Library": 63,"Exit Gate": 270,"Food Court": 355,"Academic Block": 27,"Security Gate": 130,"Medical Center": 90,"Mart": 585,"Hostel": 520,"Sports Area": 900},

    "Security Gate": {"Library": 100,"Exit Gate": 180,"Main Gate": 220,"Cafeteria": 130,"Academic Block": 240,"Auditorium": 120,"Medical Center": 90,"Mart": 520,"Hostel": 500,"Sports Area": 210,"Food Court": 471,"Admin Block": 90},

    "Library": {"Auditorium": 3,"Food Court": 470,"Exit Gate": 240,"Admin Block": 16,"Main Gate": 275,"Mart": 465,"Academic Block": 46,"Medical Center": 75,"Hostel": 450,"Sports Area": 810,"Security Gate": 100,"Cafeteria": 63},

    "Sports Area": {"Admin Block": 740,"Cricket Ground": 350,"Main Gate": 580,"Hostel": 300,"Academic Block": 560,"Exit Gate": 350,"Library": 810,"Security Gate": 210,"Medical Center": 840,"Mart": 320,"Food Court": 430,"Cafeteria": 900},

    "Admin Block": {"Main Gate": 160,"Academic Block": 126,"Mart": 434,"Cafeteria": 35,"Exit Gate": 220,"Library": 16,"Security Gate": 80,"Hostel": 506,"Auditorium": 17,"Medical Center": 80,"Food Court": 355,"Sports Area": 740},

    "Auditorium": {"Food Court": 344,"Security Gate": 120,"Exit Gate": 260,"Library": 3,"Main Gate": 286,"Academic Block": 150,"Mart": 574,"Medical Center": 80,"Hostel": 510,"Sports Area": 830,"Admin Block": 17,"Cafeteria": 10},

    "Academic Block": {"Food Court": 192,"Security Gate": 240,"Exit Gate": 250,"Mart": 410,"Admin Block": 126,"Library": 46,"Hostel": 380,"Auditorium": 150,"Medical Center": 85,"Cafeteria": 27,"Sports Area": 560,"Main Gate": 210},

    "Food Court": {"Exit Gate": 609,"Library": 470,"Mart": 836,"Medical Center": 405,"Auditorium": 344,"Main Gate": 699,"Admin Block": 355,"Hostel": 816,"Academic Block": 192,"Security Gate": 471,"Cafeteria": 355,"Sports Area": 430},

    "Main Gate": {"Exit Gate": 100,"Security Gate": 150,"Admin Block": 160,"Academic Block": 210,"Library": 275,"Auditorium": 286,"Cafeteria": 276,"Food Court": 699,"Medical Center": 180,"Sports Area": 580,"Hostel": 600,"Mart": 620}

}


def uniform_cost_search(graph, start, goal):

    queue = []

    heapq.heappush(queue, (0, start, [start]))  # (cost, current_node, path)

    visited = set()


    while queue:

        cost, current, path = heapq.heappop(queue)
```

```
        if current == goal:
            return path, cost

        if current in visited:
            continue
        visited.add(current)

        for neighbor, distance in graph[current].items():
            if neighbor not in visited:
                heapq.heappush(queue, (cost + distance, neighbor, path + [neighbor]))

    return None, float('inf')

def main():
    print("Available locations:", list(campus_graph.keys()))
    start = input("Enter start location: ").strip()
    goal = input("Enter goal location: ").strip()

    if start not in campus_graph:
        print(f"Start location '{start}' not found.")
        return
    if goal not in campus_graph:
        print(f"Goal location '{goal}' not found.")
        return

    path, cost = uniform_cost_search(campus_graph, start, goal)

    if path is None:
        print(f"No path found from {start} to {goal}.")
    else:
```

```
        print(f"Path from {start} to {goal}:\n{' -> '.join(path)}")
        print(f"Total distance: {cost} meters")

if __name__ == "__main__":
    import heapq
    main()
```

Output

```
Available locations: ['Medical Center', 'Mart', 'Exit Gate', 'Hostel', '
Cafeteria', 'Security Gate', 'Library', 'Sports Area', 'Admin Block', 'A
uditorium', 'Academic Block', 'Food Court', 'Main Gate']
Enter start location: Admin Block
Enter goal location: Mart
Path from Admin Block to Mart:
Admin Block -> Mart
Total distance: 434 meters
```

➢ BFS

```
from collections import deque

campus_graph = {
    "Medical Center": {"Food Court": 405,"Hostel": 490,"Cafeteria": 90,"Exit Gate": 160,"Auditorium": 80,"Main Gate": 180,"Library": 75,"Admin Block": 80,"Academic Block": 85,"Security Gate": 90,"Sports Area": 840,"Mart": 510},
    "Mart": {"Exit Gate": 600,"Academic Block": 410,"Main Gate": 620,"Cafeteria": 585,"Security Gate": 520,"Food Court": 836,"Sports Area": 320,"Library": 465,"Hostel": 200,"Auditorium": 574,"Admin Block": 434,"Medical Center": 510},
    "Exit Gate": {"Main Gate": 100,"Academic Block": 250,"Cafeteria": 270,"Security Gate": 180,"Auditorium": 260,"Library": 240,"Hostel": 580,"Admin Block": 220,"Sports Area": 350,"Food Court": 609,"Medical Center": 160,"Mart": 600},
    "Hostel": {"Admin Block": 506,"Sports Area": 300,"Academic Block": 380,"Mart": 200,"Food Court": 816,"Library": 450,"Auditorium": 510,"Main Gate": 600,"Cafeteria": 520,"Medical Center": 490,"Security Gate": 500},
```

"Cafeteria": {"Main Gate": 276,"Auditorium": 10,"Admin Block": 35,"Library": 63,"Exit Gate": 270,"Food Court": 355,"Academic Block": 27,"Security Gate": 130,"Medical Center": 90,"Mart": 585,"Hostel": 520,"Sports Area": 900},

"Security Gate": {"Library": 100,"Exit Gate": 180,"Main Gate": 220,"Cafeteria": 130,"Academic Block": 240,"Auditorium": 120,"Medical Center": 90,"Mart": 520,"Hostel": 500,"Sports Area": 210,"Food Court": 471,"Admin Block": 90},

"Library": {"Auditorium": 3,"Food Court": 470,"Exit Gate": 240,"Admin Block": 16,"Main Gate": 275,"Mart": 465,"Academic Block": 46,"Medical Center": 75,"Hostel": 450,"Sports Area": 810,"Security Gate": 100,"Cafeteria": 63},

"Sports Area": {"Admin Block": 740,"Cricket Ground": 350,"Main Gate": 580,"Hostel": 300,"Academic Block": 560,"Exit Gate": 350,"Library": 810,"Security Gate": 210,"Medical Center": 840,"Mart": 320,"Food Court": 430,"Cafeteria": 900},

"Admin Block": {"Main Gate": 160,"Academic Block": 126,"Mart": 434,"Cafeteria": 35,"Exit Gate": 220,"Library": 16,"Security Gate": 80,"Hostel": 506,"Auditorium": 17,"Medical Center": 80,"Food Court": 355,"Sports Area": 740},

"Auditorium": {"Food Court": 344,"Security Gate": 120,"Exit Gate": 260,"Library": 3,"Main Gate": 286,"Academic Block": 150,"Mart": 574,"Medical Center": 80,"Hostel": 510,"Sports Area": 830,"Admin Block": 17,"Cafeteria": 10},

"Academic Block": {"Food Court": 192,"Security Gate": 240,"Exit Gate": 250,"Mart": 410,"Admin Block": 126,"Library": 46,"Hostel": 380,"Auditorium": 150,"Medical Center": 85,"Cafeteria": 27,"Sports Area": 560,"Main Gate": 210},

"Food Court": {"Exit Gate": 609,"Library": 470,"Mart": 836,"Medical Center": 405,"Auditorium": 344,"Main Gate": 699,"Admin Block": 355,"Hostel": 816,"Academic Block": 192,"Security Gate": 471,"Cafeteria": 355,"Sports Area": 430},

"Main Gate": {"Exit Gate": 100,"Security Gate": 150,"Admin Block": 160,"Academic Block": 210,"Library": 275,"Auditorium": 286,"Cafeteria": 276,"Food Court": 699,"Medical Center": 180,"Sports Area": 580,"Hostel": 600,"Mart": 620}

}

```
def bfs(graph, start, goal):
    queue = deque([(start, [start])])
    visited = set()

    while queue:
        current, path = queue.popleft()

        if current == goal:
```

```
            return path

        if current not in visited:
            visited.add(current)
            for neighbor in graph[current]:
                if neighbor not in visited:
                    queue.append((neighbor, path + [neighbor]))

    return None

def main():
    print("Available locations:", list(campus_graph.keys()))
    start = input("Enter start location: ").strip()
    goal = input("Enter goal location: ").strip()

    if start not in campus_graph:
        print(f"Invalid start location: {start}")
        return
    if goal not in campus_graph:
        print(f"Invalid goal location: {goal}")
        return

    path = bfs(campus_graph, start, goal)

    if path:
        print("Path found (shortest by number of edges):")
        print(" -> ".join(path))
        print(f"Number of steps: {len(path)-1}")
    else:
        print("No path found.")
```

```
if __name__ == "__main__":
    main()
```

Output

```
Available locations: ['Medical Center', 'Mart', 'Exit Gate', 'Hostel', '
Cafeteria', 'Security Gate', 'Library', 'Sports Area', 'Admin Block', 'A
uditorium', 'Academic Block', 'Food Court', 'Main Gate']
Enter start location: Hostel
Enter goal location: Library
Path found (shortest by number of edges):
Hostel -> Library
Number of steps: 1
```

Note: These are the sample codes for providing the path and the distance, I'm working on the campus information codes and I will upload them shortly. Later if there are any changes I will update them.