S VISVESVARAYA TECHNOLOGICAL UNIVERSITY

**JNANA SANGAMA,BELAGAVI – 590018**

**KARNATAKA**



**Mini Project Report**

**On**

**“”**

**SUBMITTED IN PARTIAL FULFILLMENT OF THE ASSIGNMENT**

**FOR THE Analysis & Design of Algorithms (BCS401)**

**COURSE OF IV SEMESTER**

Submitted by

|  |  |  |
| --- | --- | --- |
| Name:  USN:  Name:  USN: |  |  |
|  |  |  |

**Guide: HOD:**

**Mr.Asif Ulla Khan.M. Tech. Dr. Shantala C PPhD.,**

Asst. Prof., Dept. of CSE Head, Dept. of CSE

CIT, Gubbi. CIT, Gubbi.

**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**



**Channabasaveshwara Institute of Technology**



(Affiliated to VTU, Belgaum & Approved by AICTE, New Delhi)

(**NAAC Accredited & ISO 9001:2015 Certified Institution)**

NH 206 (B.H. Road), Gubbi, Tumkur – 572 216. Karnataka

**2023-24**

**Rubric – B.E. Mini-Project [BCS401]**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Course**  **outcome** | **Rubric/Level** | **Excellent**  **(91-100%)** | **Good**  **(81-90%)** | **Average**  **(61-80%)** | **Moderate**  **(40-60%)** | **Score** |
| **CO1** | **Identification of project**  **proposal**  **(05 Marks)** |  |  |  |  |  |
| **CO2** | **Design and**  **Implementation**  **(05 Marks)** |  |  |  |  |  |
| **CO3** | **Presentation skill**  **(05 Marks)** |  |  |  |  |  |
| **CO 4** | **Individual or**  **in a team**  **development** |  |  |  |  |  |
| **CO5** | **Report**  **(05 Marks)** |  |  |  |  |  |
| **Total** | | | | | |  |

**Course outcome:**

**CO 1: Identification of project proposal which is relevant to subject of engineering.**

**CO 2: Design and implement proposed project methodology.**

**CO 3: Effective communication skill to assimilate their project work.**

**CO 4: Work as an individual or in a team in development of technical projects.**

**CO 5:Understanding overall project progress and performance.**

**Student Signature Faculty signature**

**ABSTRACT**

Dijkstra Algorithm is one of the most famous algorithms in computer science. There might be several possible routes to reach a destination point. If someone doesn’t travel through optimal path, it will consume more time and energy. This project aims to determine locations of the node that reflect all the nodes in the list, build the route by connecting nodes and evaluate the optimal path by using Dijkstra algorithm. Dijkstra’s Algorithm is also known as a single source shortest path algorithm which is used to find the shortest distance/path from one node to another node in a graph. This algorithm can be used only for positive distances from one location to another.

**CHAPTER 1:**

**INTRODUCTION**

In our daily life, people commonly face many problems in finding which path leads. People usually explore every possible solution in finding their way within the campus, but not every solution can produce the optimal path. Shortest path problem is a problem in finding the fastest route or path from a directed graph. The “Campus Navigation” is used for navigating the user from one place to another within the campus. All navigation system takes user location as a starting point and gives the best optimal path to be followed. Every direction in a graph (map) have a cost to be calculated. This shortest path problem is a way to find a new route or path in a graph with a minimum sum of weight travelled through the direction. This shortest path problem is solved by using Dijkstra’s algorithm of finding the best edge path between vertices in a graph. There are several variations of algorithm that can be used to determine the node that was pursued based on the direction in given graph.

Variations of the shortest path can be distinguished from single-source objective, pair path and generalization. A pair of shortest path is finding the shortest path for two points of nodes. All pair of shortest path is a technique to find the shortest path among all directed nodes. Single-source shortest path is finding the shortest form travelled, starting from a certain node to all other nodes in the graph.

Single-objective shortest path problem is to find the shortest path from any node on the graph are directed to a single destination node. Intermediate shortest path is finding the shortest path between two nodes selected through other nodes. Generalization is significantly more efficient than the simple approach to run one-pair of shortest path algorithm on all pairs of vertices that are relevant.

**CHAPTER:2**

**PROBLEM STATEMENT**

There exist many advanced navigation systems but most of them are unable to provide routes precisely as well as information of building within a region such as campus, shopping mall, hospital and etc.

Nowadays, as people are getting more and more connected to technology, they lost their human touch. Also, people feel more convenient to search for the problem themselves rather than asking someone for help.

An informative, reliable and precise guidance system is very important in this technological era. It should be able to navigate the user no matter the user is under the indoor or outdoor environment. The guidance system must be user-friendly and able to process data efficiently.

Since the size of any college/university campus can vary from 30 acres to anywhere around 200 acres, students spend majority of their time in travelling between different buildings. New students feel inconvenient to search their way inside the campus.

Therefore, a navigation system is required to find the optimal path within the campus and for the aforementioned problem.

**CHAPTER:3**

**IMPLEMENTATION**

**ALGORITHM:**

**PROGRAM:**

import matplotlib.pyplot as plt

import networkx as nx

from matplotlib.patches import Rectangle

# Define the campus layout as a graph

G = nx.Graph()

# Add nodes with their positions

nodes = {

'College Gate': (0, 0),

'Canteen Block': (2, 4),

'Admin Block': (6, 6),

'CRC Block': (12,-6),

'MC Block': (16, 2),

'Library':(14,-8),

'Auditorium':(10,-8),

'Workshop & Mech Lab': (12, 6),

'Hostel Block': (4, -12),

'Cricket Ground': (2,-4),

'Sports Room':(4,-10)

}

for node, position in nodes.items():

G.add\_node(node, pos=position)

# Add edges with weights (distances)

edges = [

('College Gate', 'Canteen Block', 1),

('Canteen Block', 'Admin Block', 1),

('Admin Block', 'Workshop & Mech Lab', 1),

('Workshop & Mech Lab', 'MC Block', 1),

('MC Block', 'CRC Block', 1),

('CRC Block', 'Library', 1),

('CRC Block', 'Auditorium', 1),

('College Gate','Cricket Ground',2),

('Cricket Ground','Sports Room',1),

('Sports Room', 'Hostel Block', 1),

('College Gate','CRC Block',2),

('Cricket Ground','CRC Block',1),

('Sports Room','Auditorium',1),

('Admin Block','CRC Block',1)

]

for edge in edges:

G.add\_edge(edge[0], edge[1], weight=edge[2])

# Function to draw the campus layout

def draw\_campus(G, path=None):

pos = nx.get\_node\_attributes(G, 'pos')

fig, ax = plt.subplots(figsize=(14, 10))

# Draw nodes as rectangles

for node, (x, y) in pos.items():

width = 3.4

height = 1.5

ax.add\_patch(Rectangle((x - width/2, y - height/2), width, height, edgecolor='black', facecolor='lightGreen'))

ax.text(x, y, node, ha='center', va='center', fontsize=8)

# Draw edges

for edge in G.edges():

x1, y1 = pos[edge[0]]

x2, y2 = pos[edge[1]]

ax.plot([x1, x2], [y1, y2], 'k-')

# Draw edge labels (weights)

edge\_labels = nx.get\_edge\_attributes(G, 'weight')

for (n1, n2), weight in edge\_labels.items():

x1, y1 = pos[n1]

x2, y2 = pos[n2]

ax.text((x1 + x2) / 2, (y1 + y2) / 2, weight, color='blue', fontsize=10,

ha='left')

# Highlight the path if given

if path:

path\_edges = list(zip(path, path[1:]))

for edge in path\_edges:

x1, y1 = pos[edge[0]]

x2, y2 = pos[edge[1]]

ax.plot([x1, x2], [y1, y2], 'r-', linewidth=2)

plt.title("Campus Layout")

ax.set\_aspect('equal')

plt.axis('off')

plt.show()

# Function to find the shortest path using Dijkstra's algorithm

def find\_shortest\_path(G, start, end):

return nx.dijkstra\_path(G, start, end)

# Function to handle user input and call the appropriate functions

def handle\_user\_input(option):

if option == '1':

draw\_campus(G)

elif option == '2':

start\_node = input("Enter the start node: ")

end\_node = input("Enter the end node: ")

if start\_node not in nodes or end\_node not in nodes:

print("Invalid start or end node. Please make sure the nodes exist in the campus layout.")

else:

shortest\_path = find\_shortest\_path(G, start\_node, end\_node)

print("Shortest Path: ", shortest\_path)

draw\_campus(G, shortest\_path)

else:

print("Invalid option.")

# Main function to present user options

def main():

while True:

print("\nCampus Navigation System")

print("1. Visualize graph")

print("2. Find shortest path")

print("3. Exit")

option = input("Choose an option (1, 2, or 3): ")

if option == '3':

print("Exiting the system.")

break

else:

handle\_user\_input(option)

# Run the main function

if \_\_name\_\_ == "\_\_main\_\_":

main( )