**S.Y. B. Tech Students' Advanced Data Structures Project Paper, SEM III A.Y. 2023-24**
**Vishwakarma Institute of Technology, Pune, INDIA.**

# Tour Planner using Graphs.

**Pankaj Kunekar, Rahul Daga, Saakshi Salvi, Mohammad Shaikh, Shrid Dagwar, Omkar Surve**

**Department of Information Technology**
**Vishwakarma Institute of Technology, Pune, 411037, Maharashtra, India**

***Abstract*: This paper presents the research and implementation of a simple Tour Planner made using the Graph Theory in Advanced Data Structures. The most common algorithm used for these projects is Dijkstra's Algorithm. This project is implemented using the C programming language. With this we were able to develop a system which will plan a tour at the lowest cost as per the route.**

***Keywords:* Graph Theory, Advanced Data Structures, C programming, Dijkstra's Algorithm.**

## Introduction

Every person requires a period to unwind from their daily duties and rest their mind and body. They have a variety of activities they may engage in to unwind, including sports, watching films, playing video games, listening to music, shopping, or simple lounging on the bed. However, most people utilize their vacation time to travel, either with friends, family, or by themselves. When people desire to take a trip, they are certain that they want it to be flawless. To save time and money, we needed a professional tour planner in the modern world. The user of Tour planner may design their vacation to suit their needs. In this project, we developed a system that would enable you to arrange your travels more effectively and efficiently. To determine the shortest path with the lowest cost in this system we employed graph theory. Specifically, users can load a travel graph, then request routes according to one of three priorities – Price (the route that costs the lowest amount of money), Time (the route with the lowest total time travel time), Directness (the route with the fewest number of connections). Since we have used graphs for our Tour Planner project, we have used the Dijkstra's Algorithm to find the shortest path to the destination selected by the user.

### Basic Theory

The study of graphs, which are mathematical structures used to represent pairwise relationships between items in a collection, is known as graph theory in mathematics and computer science. A graph is an abstract representation of a collection of things where certain object pairs are linked together. Vertex, a mathematical abstraction used to represent the linked items, and edges, the connections between specific pairs of vertices, are what connect the connected

things. A graph is often represented diagrammatically as a collection of dots for the vertices and lines or curves for the edges.

A graph can be represented with no edge, but must have (minimal one) vertex, this kind of graph is called trivial graph. Graphs also can be grouped into some of the following categories: 1. Simple Graph (undirected with no loop and no more than one edge) 2. Multigraph (Graph that have a double edge between vertices) 3. Pseudograph (Graph that have a loop, including a double edge) 4. Undirected Graph (Graph in which edges have no orientation at all) 5. Directed Graph (Graph that each of the edge has an orientation). Circuit or Cycle is a path that starts and ends in the same vertex. Length of the circuit is sum of the edge in that circuit. A circuit called a simple circuit if every edge that be passed is different.

## Algorithm Used

Dijktra's Shortest Path Algorithm is a commonly taught and widely used path finding algorithm. It's useful because it works on a weighted graph and it's simple to implement. The algorithm keeps track of the currently known shortest distance from each node to the source node and it updates these values if it finds a shorter path. Once the algorithm has found the shortest path between the source node and another node, that node is marked as visited and added to the path. The process continues until all the nodes in the graph have been added to the path. This way, we have a path that connects the source node to all other nodes following the shortest path possible to reach each node.

All the vertices, or nodes, are initially marked as unvisited since they haven't been reached. Aside from the source node, the path to every node is likewise set to infinite. Additionally, the source node's path is set to zero (zero). The source node is then chosen and marked as visited. The next step is to access and relax each of the nodes that are close to the source node. Reduction of the cost of contacting a node with the aid of another node is the process of relaxation. Each node's path is adjusted during the relaxation process to the value that is least among its current path, the sum of its paths to the previous nodes, and the paths from the previous nodes to the current nodes.

Assume that p[n] represents the value of the current path for node n, p[m] represents the value of the path up to the previously visited node m, and w represents the weight of the edge (edge weight between n and m) between the current node and previously visited one. In the mathematical sense, relaxation can be exemplified as:

***p[n] = minimum(p[n], p[m] + w)***

This can be even simplified with the following example:

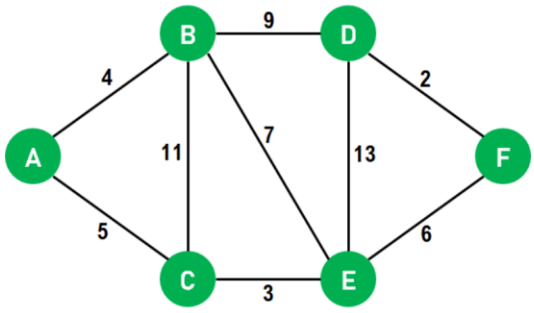


**Figure 1.**

Hence, the final path we concluded are:

***A = 0***

***B = 4 (A ->B)***

***C = 5 (A -> C)***

***D = 4 + 9 = 13 (A -> B -> D)***

***E = 5 + 3 = 8 (A -> C -> E)***

***F = 5 + 3 + 6 = 14 (A -> C -> E -> F)***

## Methodology

The basic flow of the system is shown below in the given flow chart:

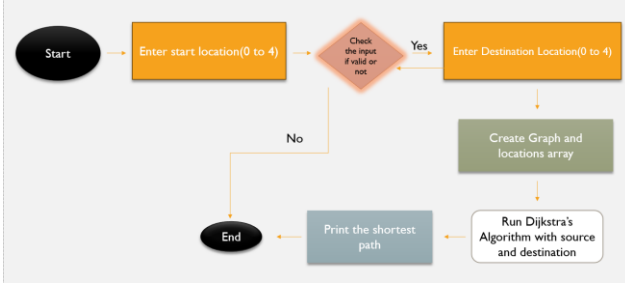


**Figure 2.**

The program begins by importing the required header files. The number of places is represented by a constant called V. It is set to 5 in this example, but you may adjust it to correspond to the number of places in your unique use case.

To locate the least distance vertex that hasn't yet been added to the shortest path tree, utilize the minDistance function. An essential component of Dijkstra's algorithm is this function. Dist[] is used to monitor distances, while sptSet[] is used to track which vertices are present in the shortest path tree. The function searches all vertices in turn to identify the one that is still outside the shortest path tree and has the smallest distance. The following vertex should be added. It gives back the index (vertex) of the vertex with the shortest distance. The 'printSolution' function first prints the distance from the source to all locations, associating each distance with the name of the corresponding location. It then prints the shortest path from source to destination. It does this by following the parent pointers from the destination to the source and printing each location along the path. The Dijkstra function is where the main logic of finding the shortest path using the Dijkstra algorithm is implemented. Three parameters are required: 1. The graph represents a weighted graph (distance between locations), 2. src (source location), and 3. location [] (an array of location names). Now, the main function is the entry point of the program. If all the inputs are valid, the main function calls the 'dijkstra' function with the source and destination locations to find and print the shortest path.

## Results and Discussion

We have successfully created a travel planning tool using graph theory. We tested our project with several inputs and concluded that it worked correctly. This helps learn how to implement Dijkstra's algorithm and apply graph theory in real life. Charts make it easy to design a travel plan, organize activities in chronological order, and find the best route to use during your trip. The types of graphs we can use to help are diverse, they can be directed graphs, undirected graphs, weighted undirected graphs, etc. Some graph theory is also useful to us, such as using Dijkstra's algorithm to solve the shortest path problem and using Euler's cycle to go through the entire route once and return to the city Firstly.

Here are some screenshots of out project implementation:

```
Enter the source location (0 to 4): 2
Enter the destination location (0 to 4): 4
Location        Distance from Source
Location A              50
Location B              50
Location C              0
Location D              20
Location E              10

Shortest Path from Source to Location E: Location E <- Location C
```

**Figure 3.**

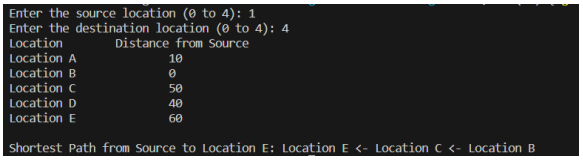


**Figure 4.**

Here as we can see the values of the distance are being changed accordingly to the provided source and destination.

## References

[1] Implementation Graph Theory in Making A Tour Trip – Benedikus Holyson Tjuatja / 13510101, Program Studi Teknik Informatika, Sekolah Teknik Elektro dan Informatika.

[2] https://www.javatpoint.com/dijkstras-algorithm

[3] https://medium.com/@assertis/so-you-want-to-build-a-journey-planner-f99bfa8d069d

[4] https://www.freecodecamp.org/news/dijkstras-shortest-path-algorithm-visual-introduction/#:~:text=Dijkstra's%20Algorithm%20finds%20the%20shortest,node%20and%20all%20other%20nodes.