PathGuide: Travel Management System

A Project Report Submitted

for the Course of

Minor Project - 2

In

Third year – Sixth Semester of

Bachelor of Technology

In

Computer Science Engineering

With Specialization

In

Graphics and Gaming

Under

Mr. Lalit Kane

Professor-Department of

Computer Science

Prepared by

Name	SAP ID	Specialization
Vishal Sharma	500068447	Graphics and Gaming

DEPARTMENT OF INFORMATICS
SCHOOL OF COMPUTER SCIENCE
UNIVERSITY OF PETROLEUM AND ENERGY STUDIES, BIDHOLI,
DEHRADUN, UTTRAKHAND, INDIA

INDEX

Project Title:

PathGuide: Travel Management System

Abstract:

The travelling salesman problem's permutation problem asks for the most direct route between N various places that the salesperson will be visiting, also known as the TOUR. This issue seeks to determine a route that encompasses every city the salesman will visit while minimising the distance covered. The best route for a salesman travelling to various cities, which are randomly selected as the initial population, can be found using the genetic algorithm technique employed in this research. The new generations are continuously produced until the stopping condition is met in an effort to find the most efficient path with the shortest distance travelled.

Introduction:

The PathGuide word suggests that it has something to do with cities Path or moving around cities. Our top concern is finding the fastest path between locations, regardless of whether you're a newcomer to the area, travelling for work or school, shopping, or chores. In these situations, we frequently turn to Google Maps for guidance, but as Google is a global search engine, it only provides information on the most popular routes, not necessarily the most convenient ones. And let's imagine we need to make several stops along the route or we need to schedule our daily journey as efficiently as feasible. The PathGuide, your buddy and companion, will come to your aid in this situation. Using different algorithms, we help people identify the best commute options to make their lives even more simple.

Literature Review:

The Travelling Salesman Problem (TSP) [1] is an effective way to identify the most direct way to travel to any number of cities, with any number of stops. It states that the traveler will be traveling to all the cities exactly once. The traveler should complete a loop in a way that if the traveler started from a city A and travels to N number of cities, he should return to the city A, with simultaneously minimizing the cost of the travel. The travelling from one city to another through N number of cities and returning to the city where he started is called the tour. We assigned the cost of the way to the distance travelled. In the complete weighted undirected graph attached below (Figure 1) G (V, E), where V stands for vertices and E stands for Edges. The V is assigned to the cities and the edges E connecting the vertices represents the distance between the cities. The TSP is designed to identify the minimal Hamiltonian Cycle that initiates from a particular vertex and ends at the very particular vertex and includes every other vertex just once. The TSP has a lot of other applications [2] Some of the areas where TSP is applied are manufacturing of microchips, planning and scheduling, vehicle routing problems, logistics, DNA sequencing, robotics, time and job scheduling of machines. The TSP is further divided into three categories Symmetric Travelling Salesman Problem (STSP), Asymmetric Travelling Salesman Problem (ATSP), and Multi Travelling Salesman Problem (MTSP). (i) STSP: In STSP the distance between the two cities is equal in both the directions which results in an undirected graph. (ii) ATSP: In ATSP The distance between the two cities is unequal in both the direction which results in a directed graph with different distance in either direction. (iii) MTSP: In MTSP we consider ‘m’ salesman in given set of nodes which is located at a single depot node. The residual nodes (cities) that are supposed to be travelled are intermediate nodes. Then the MTSP calculates the tours of m number of salesman, all of them started from one place and ended at the place they started,

and they included each intermediate node (city) only once and the entire cost of travelling all nodes is minimal. The solution offered by TSP can be of two types. 1) In the first solution it will identify the most effective way which will be the closest to the precise solution. This method assures the quality of a solution but is rather slow. 2) The second solution can identify the most effective way within a reasonable time, but the solution will be farther than the precise solution. The way to improve this problem and to increase the assurance and the performance genetic algorithms are used. They solve the TSP wand gives us the result in a reasonable time frame.

Objective:

- The primary objective of this project is to build a console-based C++ application called PathGuide that can determine the most efficient path between different locations.
- Using the C++ language and data structures to tackle problems in the real world
- Learning the famous Dijkstra Algorithm and applying the same for the PathGuide

Methodology:

In the TSP, a salesman must choose the quickest and most direct route to visit each city only once and return to the starting location.

Let's think about the following illustration to better understand TSP:

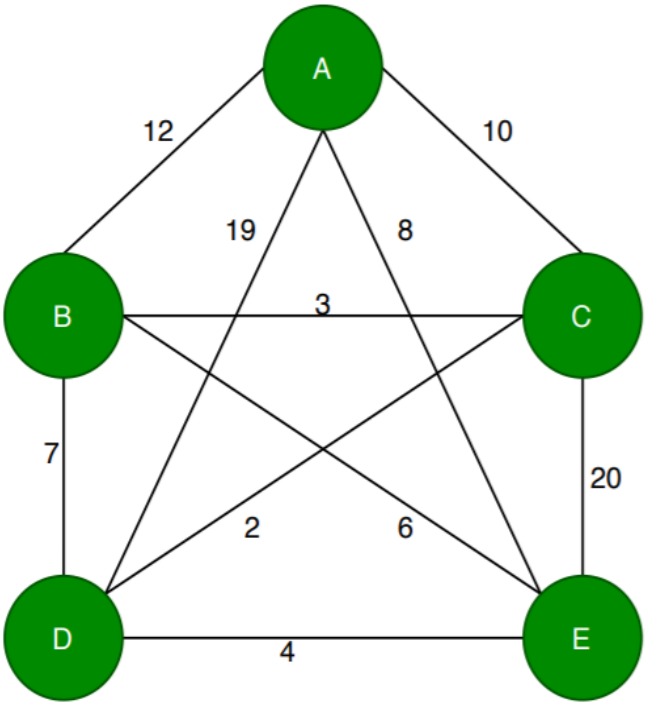


Figure 1

In Figure 1, The vertices are given the identities of various cities, and the value of each edge connecting the vertices denotes the separation between the two cities it links. Consider the salesperson who departs from city A. According to TSP, the salesperson should only visit each other city once before returning to city A in the most efficient manner. The salesman's total travel distance between each city, which should be shorter than any other route, constitutes the most efficient route in this case.

The problem already looks quite complicated with only cities involved. As the graph in the figure 1 is a complete graph, the salesman can start from any city and travel to other cities in the graph. The salesman needs to identify a travelling plan so that he travels to every city just once while also minimizing the distance travelled.

Here we are also gonna do the same thing we are going to consider a student travelling from some start node n and travel through all of the nodes which will be given by the user

Algorithm:

Brute Force :

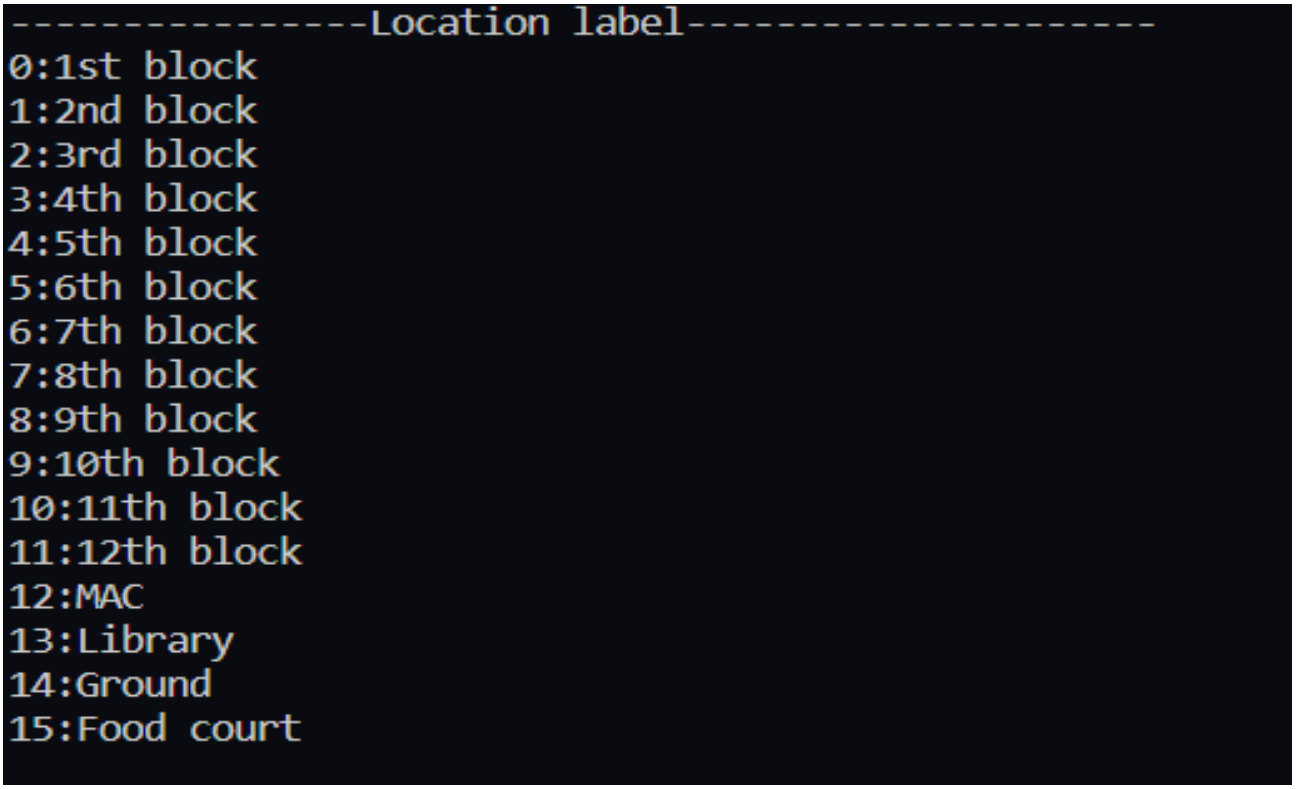
Step 1:Take input of the codes of the places and accordingly prepare the cost matrix(also Adjacency matrix here)
Step 2: Putting a vector containing the indexes of all the input nodes but the start node in storage
Step 3: Using a cost matrix to calculate the matrix's cost and adding the start node cost both upfront and at the conclusion
Step4: If the distance is less than the minimum distance then added the distance as optimal cost and update the minimum distance along with the min path
Step 6: Output the least costly path utilising vectors that have been declared earlier.

Dijkstra Algo:

Step 1:Take input of the codes of the places and accordingly prepare the cost matrix(also Adjacency matrix here)
Step 2: Passing the Adjacency matrix as with Weight of travelling from one one to another
Step 3: Declaring two priority queue and pushing all the nodes(say x) expect the start node with the cost of travelling from start node to the node x into one of the priority queue(say pq).
Step 4: Using priority queue inserting rest of the nodes in the top element of the pq(as it has the lowest cost) with the help of a while loop.
Step 5: As soon as the top element covers all the nodes, we push it into another priority queue(say anspq) with adding the cost to return to initial node.
Step 6: Now we know the top element of the anspq will be our desired path and minimum cost.
Step 7: Output the path i.e. the top element of the anspq with the help of other vectors

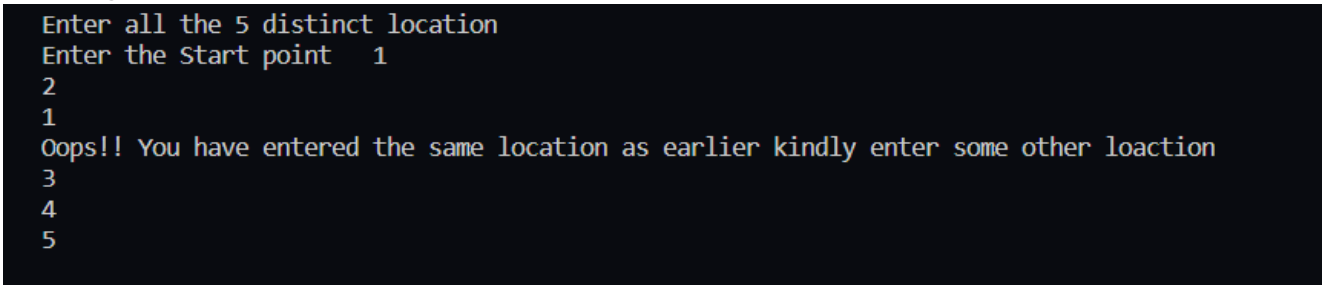
Result:
For testing purposes, the programme is only intended to work in locations that are part of the UPES campus.

The programme contains hard coded distance data between different locations and user is asked to enter the code for the various places.



Taking input from the user

Consider a case where user input same node twice, in that case our program displays a warning and asks user to re enter the location as follows



Further the path is displayed as following-

```
Path-
From - 2nd block - 3rd block - 4th block - 5th block - 6th block - 2nd block
The stated path cost us 40
Path Using Brute Force-
From 1st block -  - 6th block - 5th block - 4th block - 3rd block - 1st block
The stated path cost us 40
```

Another example for the same-

```
----------------Location label--------------------
0:1st block
1:2nd block
2:3rd block
3:4th block
4:5th block
5:6th block
6:7th block
7:8th block
8:9th block
9:10th block
10:11th block
11:12th block
12:MAC
13:Library
14:Ground
15:Food court

Enter all the 5 distinct location
Enter the Start point   0
1
2
3
4

Path-
From - 1st block - 2nd block - 3rd block - 4th block - 5th block - 1st block
The stated path cost us 26
Path Using Brute Force-
From 1st block -  - 5th block - 4th block - 3rd block - 2nd block - 1st block
The stated path cost us 26
```

From the above two results it is also clear that there can be multiple ways to achieve the least cost path.

Result Analysis:

The Brute Force algorithm for TSP allows for a maximum of (N-1)! outcomes. Each result is solvable in O(N) steps. As a result, the algorithm's temporal complexity is given as O ((N-1)! x N).

Further for the Dijkstra algorithm we are keeping a priority queue which will have all the possible paths The time complexity in this case is { O(N * 2^N *Log(N * 2^N)) }, where V is the total number of nodes, 2^N is the total number of node subsets that can be created, and Log(N * 2^N) is the time complexity for adding each state to the priority queue.

TSP is a popular NP-Hard problem, but depending on the size of the input cities, it is possible to find an optimal or a near-optimal solution using various algorithms.

Conclusion:

In this report, we talked about the difficulty of figuring out the shortest path that stops at each node. We presented the main ideas of two different approaches to resolving this problem and illustrated how to put each one into practise. We then put these ideas into practise by assisting students in locating the shortest route for travelling between various locations within our college and returning to the same point.
Further it is also seen that multiple paths can also lead to the same cost.
.

References:

GIT link : https://github.com/Abhijeet-try/CityMate

- See the TSP world tour problem which has already been solved to within 0.05% of the optimal solution. [1]
- ^ *Ross, I. M.; Proulx, R. J.; Karpenko, M. (6 May 2020). "An Optimal Control Theory for the Traveling Salesman Problem and Its Variants". arXiv:2005.03186 [math.OC].*
- Zomato solving travelling salesman problem https://zenodo.org/record/2656305/files/Online%20FDS.pdf
- Problem solving state representation https://www.geeksforgeeks.org/travelling-salesman-problem-set-1/
- ^ A discussion of the early work of Hamilton and Kirkman can be found in *Graph Theory, 1736–1936* by Biggs, Lloyd, and Wilson (Clarendon Press, 1986).
- *Informed and Uninformed Search Algorithms https://www.geeksforgeeks.org/difference-between-informed-and-uninformed-search-in-ai/*
- ^ *Klarreich, Erica (8 October 2020). "Computer Scientists Break Traveling Salesperson Record". Quanta Magazine. Retrieved 13 October 2020.*
- ^ *Karlin, Anna R.; Klein, Nathan; Gharan, Shayan Oveis (30 August 2020). "A (Slightly) Improved Approximation Algorithm for Metric TSP". arXiv:2007.01409 [cs.DS].*
- Hash Map https://www.geeksforgeeks.org/hashing-data-structure/
- Niels Agatz, Paul Bouman, and Marie Schmidt. Optimization approaches for the traveling salesman problem with drone. Transportation Science, Forthcoming, 2017
- Richard Bellman. Dynamic programming treatment of the travelling salesman problem. Journal of the ACM (JACM), 9(1):61–63, 1962.
- https://core.ac.uk/download/pdf/154419746.pdf
- https://www.baeldung.com/cs/shortest-path-visiting-all-nod