****

**Course Name:** 15B17CI371

Data Structures Lab

**SHIP MANAGEMENT SYSTEM**

**Semester:** III

**Batch:** B8

**Submitted By:**

Arjun Goyal – 17103294

Rhythm Gupta – 17103310

Sankalp Biswal – 17103314

Pranav Sachdeva – 17103318

**PROBLEM STATEMENT**

A **Ship Management System** is a fundamental component of a modern ship [system](https://en.wikipedia.org/wiki/Avionics). An SMS is a specialized computer system that automates a wide variety of in-ship tasks, reducing the workload on the ship crew to the point that modern civilian ships no longer carry [ship engineers](https://en.wikipedia.org/wiki/Flight_engineer) or [navigators](https://en.wikipedia.org/wiki/Navigator). A primary function is in-ship management of the ship plan.

So in order to make it easier for the ship crew members as well as passengers we have to make a model of SMS which is convenient to them. This model can easily find all the details of booking of tickets, status and time of ship departure and also the shortest path and its cost. It also displays path between two nodes (or countries).

**INTRODUCTION**

**Objective of the project:**

The main objective of this C++ project on Ship Management System is to manage the details of Passenger Reservation, Ticket Booking, and Booking Enquiry. It manages all the information about displaying path between two nodes and shortest path with its cost.

**Task to be performed:**

* Booking tickets.
* Searching the details of the customers.
* Finding the shortest path between some countries using graph.
* Finding the shortest path and it’s cost between a source and destination.

**Need of Computerization:**

A few factors that direct us to develop a digital system for ship management are given below -:

* Faster System
* Accuracy
* Reliability
* Informative
* Reservations and cancellations from anywhere to any place

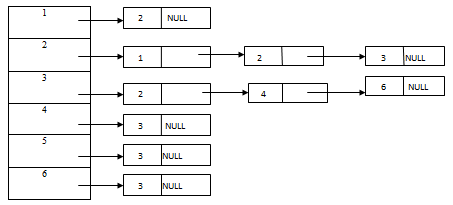
**DATA STRUCTURES USED**

In this project, we have mainly used Multilists, Queue, pair, vector from Standard Template Library, Graph, Searching, Graph Traversals, Floyd Warshall Algorithm for Shortest Path.

Multilist:

A multi linked list is a linked list where each node may contain pointers to more than one nodes of the linked list.

In our project we have used multilist to store the passenger details. Two multilists were created. In the first multilist we have stored the details of passengers in a sorted order by name and by ticket number. The second multilist contains the information of different ships. The passengers who board that particular ship gets added in the list.

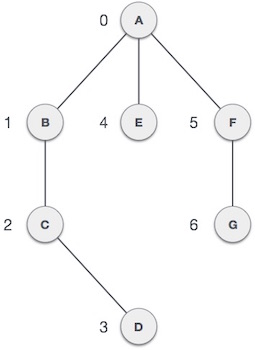


Graph Data Structure:

A graph is a pictorial representation of a set of objects where some pairs of objects are connected by links. The interconnected objects are represented by points termed as **vertices**, and the links that connect the vertices are called **edges**.

Formally, a graph is a pair of sets **(V, E)**, where **V** is the set of vertices and **E**is the set of edges, connecting the pairs of vertices.

Mathematical graphs can be represented in data structure. We can represent a graph using an array of vertices and a two-dimensional array of edges.



**DETAILED DESIGN**

5000

SINGAPORE

INDIA

2100

3000

2500

1500

2200

CANADA

GERMANY

1700

1900

1800

KENYA

For finding all the paths and the shortest paths we have used the directed graph as shown in the figure (a).

For finding all paths, we have create a queue which stores the paths and a path vector to store the current path.

->If the last vertex is the desired vertex then print the path

->Else traverse to all the nodes connected to current vertex and push new path to queue

And to find the shortest path we have used the Floyd-Warshall Algorithm.

**Floyd Warshall Algorithm**

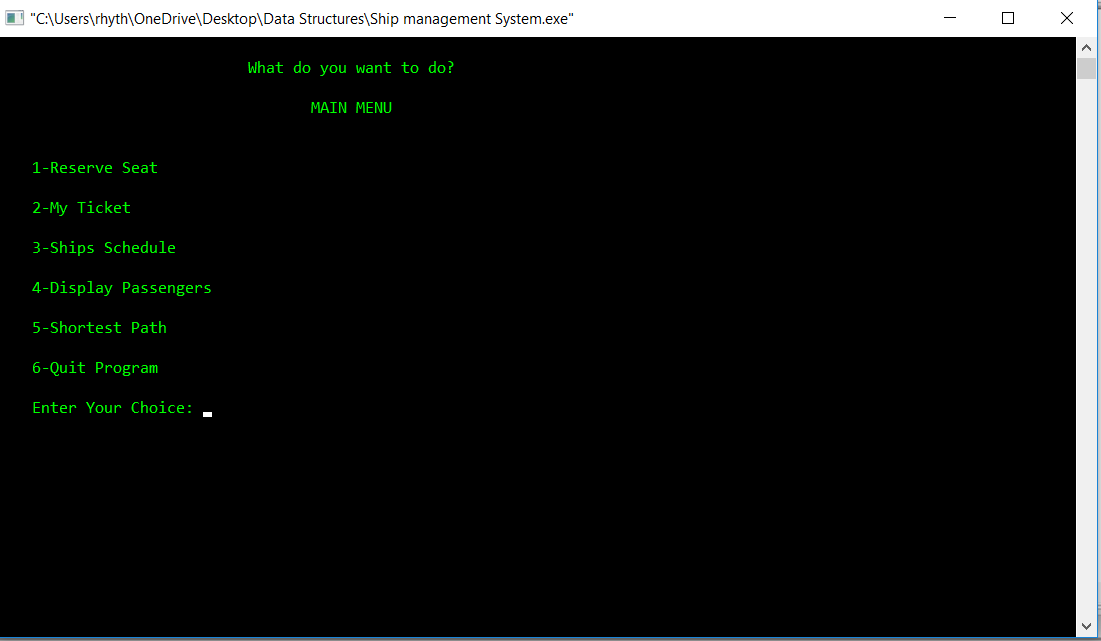
We initialize the solution matrix same as the input graph matrix as a first step. Then we update the solution matrix by considering all vertices as an intermediate vertex. The idea is to one by one pick all vertices and update all shortest paths which include the picked vertex as an intermediate vertex in the shortest path. When we pick vertex number k as an intermediate vertex, we already have considered vertices {0, 1, 2, .. k-1} as intermediate vertices. For every pair (i, j) of source and destination vertices respectively, there are two possible cases.  
**1)** k is not an intermediate vertex in shortest path from i to j. We keep the value of dist[i][j] as it is.  
**2)** k is an intermediate vertex in shortest path from i to j. We update the value of dist[i][j] as dist[i][k] + dist[k][j].

**DESCRIPTION ABOUT IMPORTANT FUNCTIONS**

We created a class named sms and executed all our functions inside the class. The functions are as follows –

* Ret\_tick\_no – Returns the total number of passengers who have booked a ticket
* Gotoxy – points the cursor to a particular destination
* Printpath – prints the given path
* IsNotVisited – returns whether the particular edge is visited or not
* Findpath- finds all possible path from source to the destination
* Enqueue – stores the passenger information into the multilist
* Displaybyship – display the passenger boarding the particular ship
* Displayticket – display all the information of the passenger of a particular ticket no
* Display- displays all passengers
* Shipcode – prints the code of the source and destination
* Show\_reserve\_menu – inputs passenger details and send it to enqueue()
* Ship\_info – display all ships info like code, fare and time.
* Menu – shows the menu, the home page of the project
* Printshortpath - prints the shortest path for a given source and destination
* Ffloyd – calculates the shortest path between two nodes.

**RESULT SCREENSHOTS**

****

