DHIRUBHAI AMBANI INSTITUTE OF INFORMATION AND COMMUNICATION TECHNOLOGY

COURSE: SC205
INSTRUCTOR: PROF. MANISH GUPTA & PROF. MANOJ KUMAR RAUT

MADE BY:
PARTH DHOLARIYA(202201085)
AVI DETROJA(202201452)
PRIYANK RAMANI(202201497)
HARSH MANGUKIYA(202201363)
MAYANK PARMAR(202201464)
HARSH GAJJAR(202201140)

Graph Theory Used in Transportation Problems and Railway Networks

June 4, 2023

ABSTRACT

This model gives an idea about how graph theory used in optimization and reduction of cost and time in transportation in railway systems.we discussed below, there are types of graphs and information that are used specifically for railroad systems in graph theory and problems in railway system and some applications.

1 INTRODUCTION

We use Graph Theory in many researches. It helps researchers to research of provable techniques in discrete mathematics. The graph theory is explored in many applications in computing, industry, natural science, and social science. In the history of transportation and telecommunication, graph theory networks have played a significant role.Additionally, there are specific graphs and information types utilised in graph theory for railway systems. In application there are three London Underground train lines are converted into graphs, and each is then examined in relation to one of three design and safety issues with railroad systems (the blocking problem, the yard location problem, and the train schedule problem). Currently, we will discuss only yard location problem and chinese postman problem .

2 TYPES OF GRAPHS

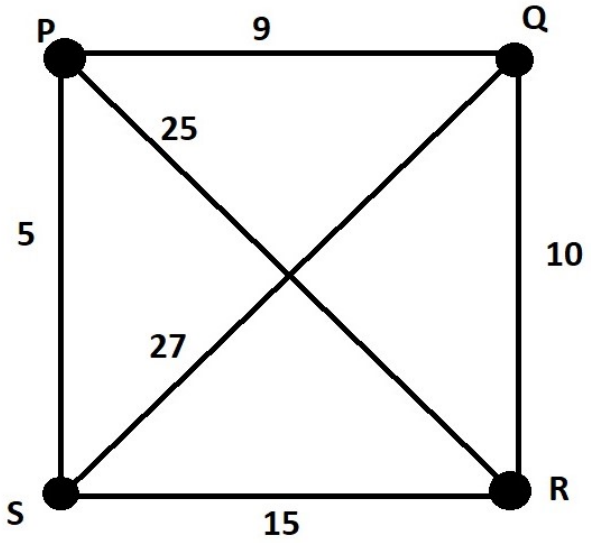
Three different forms of graphs—weighted graphs, direct graphs, and double vertex graphs—will be covered in this section.

2.1 WEIGHTED GRAPHS

Each edge, denoted by "t", has a "weight", such as the separation between two vertices. The shortest and most effective path to take can be found using weighted graphs. Two well-known examples of this are the Chinese Postman Problem and the Travelling Salesman Problem. [9]

->EXAMPLE:-

This example will be a version of the Traveling Salesman Problem. A more general example of this problem can be found in [9].Consider a graph P with the following vertices and edges (that have weights).



Imagine that P represents a route, where P, Q, R, and S are different cities. Each edge connecting to cities represents the total length (in kilometers) between the two cities. For example, between City P and City Q the travel between the two cities is 9 kilometers. A person has to travel from City P, visit every other city only once, and return back to City P. What is the shortest route this person can possibly take? There are six possible routes and the total distances below:

ROUTE	TOTAL DISTANCE OF ROUTE(KM)
PQRSP	39
PQSRP	76
PRQSP	67
PRSQP	77
PSQRP	67
PSRQP	39

From the table, we can see that the shortest routes to take are either PQRSP or PSRQP, which are both 39 kilometers. Additionally, since this is the same graph structure as in Example 1.4, the circuits, PQRSP and PSRQP, would also be considered Hamiltonian circuits.We can build railway systems using data and graphs like this, but there are more effective graphs that consider every aspect of a railway system. Note that this version of the problem was inspired by two examples, found in [7] (The Chinese Postman Problem) and [9] (The Traveling Salesman Problem). As a result, we can observe that some graphs have edges that can be traversed in both ways, indicating that the edges do not have a defined direction. An edge that can be travel upon in both directions is denoted pictorially as a single line in a graph.

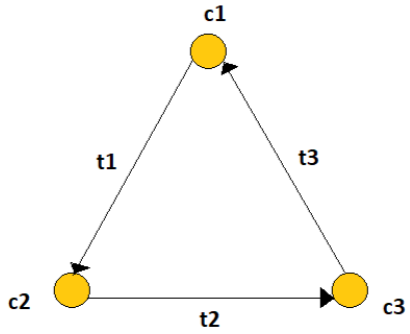
2.2 DIRECTED GRAPHS

->DEFINITION:-

A directed graph D is a finite, nonempty set of vertices,together with some directed edges joining pairs of vertices. It can be said that the edge-endpoint function associates each edge, t to an ordered pair of vertices, (v, w), where v is the initial vertex ("starting" vertex) of edge t and w is the terminal vertex ("ending" vertex) of edge t. Directed graphs are held to additional restriction: that the initial and terminal vertices cannot be the same. [10] A single line with an arrow drawn as an edge that must only be traversed in one direction.

->EXAMPLE:-

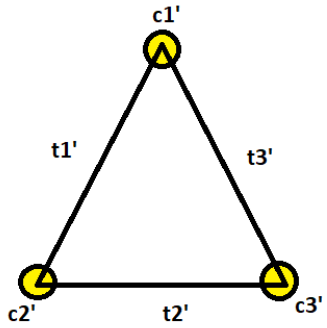
Consider graph H below. Graph H is a directed graph.



In order to complete a circuit that starts and ends at c1, we can only perform the following circuit:

c1t1c2t2c3t3c1

This is the only circuit, starting and ending at c1 because of the directed edges.Consider graph H'.



If we want to complete a circuit, with it starting and ending at c1' , we will have the following two circuits:

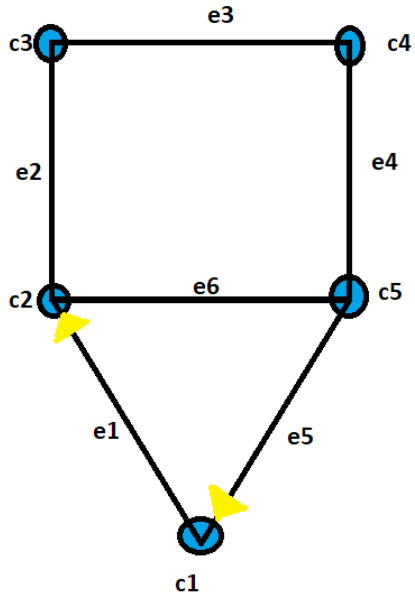
Circuit 1 : c1't1'c2't2'c3't3'c1'

Circuit 2 : c1't3'c3't2'c2't1'c1'

Therefore, since there are no directed edges, we can make a circuit by starting from c1' to t1'or from c1' to t3'.

Notably, there are some edges that are directed and some that are not in some versions of directed graphs. The edges that can only be traversed in one direction are always indicated by an arrow.Undirected edges can be presumed to exist on edges that are not indicated by an arrow. Below is an example.

->EXAMPLE:-
Consider graph H.



We can see that there are edges that can be traveled upon in both directions (e2, e3, e4 and e6) and edges that can only be traveled upon in one direction(e1 and e5). Therefore, when applied to the railroad system, we can have train tracks that are only traveled upon in one direction, while there are other tracks that can be traveled upon in both directions.

2.3 DOUBLE VERTEX GRAPHS

In a railway system, we must take into account more than just the

track's length or the direction that a train goes in order to make decisions. For example, we must consider the speed of a train on a given track.

->DEFINITION:-

Let a double vertex graph T be a finite, nonempty set of vertices, V (G), and there be be a finite set of edges, E(G), between the vertices, where none of the edges are loops or that there are multiple edges between a pair of vertices. [8] Additionally, these graphs are specifically for railroad systems and every edge holds different pieces of information for the railroad, such as track length, maximum speed, etc. [7]

A graph, weighted graph, or directed graph of the railway system alone would not contain the specific, physical information that a double vertex graph of the railway system does.

3 DESIGN AND SAFETY PROBLEMS IN THE RAILROAD SYSTEM

The design of railway systems must take into consideration of a number of safety and design issues. Because certain trains can only run in one direction but use a two-way line, railway systems need to be properly built. Therefore, it is important to prevent train accidents because, if nothing is done to consider for the fact that two trains will be travelling on the same rail at the same time, it is possible that they may collide.

We will be focusing on two: Chinese Postman's Problem,the yard location problem.

3.1 CHINESE POSTMAN'S PROBLEM

In 1962, A Chinese mathematician called Kuan Mei-ko was interested in a postman delivering mail to a number of streets. Such that the total distance walked by the postman was as short possible. How could the postman ensure that the distance walked was minimum.

3.2 YARD LOCATION PROBLEM

We try to find the best network configuration in terms of number and location of yards where cars can be reclassified into new blocks and switch trains." [5]. The train travels to a "yard" on its way from its departure location, which we shall refer to as an origin, to its final destination, as was previously stated. Trains from other train lines in the system are also classed in yards before being sent to different locations. Local yards, system yards, and regional yards are the three different categories of yards. Local yards are used when trains are arriving from stations that are divided into blocks but are not in yards. System yards and regional yards are regarded as "hub yards," as they both handle larger quantities of trains and commodities. Compared to system yards and regional yards, local yards are smaller. [5] We will pay particular attention to nearby yards.

Additionally, the goal of the yard location problem is to reduce the total distance of the trains' journeys, much like the blocking problem. Due to the growing traffic on railroads and the addition of more trains, even if the blocking network is minimised, we cannot accurately minimise the path of the train if the number and position of yards are not taken into account. We can calculate a more precise minimised path for a train to take from its origin to its destination by taking into account both the blocking network and the yards in a railroad system. [5]

4 SOLUTIONS

4.1 CHINESE POSTMAN SOLUTION

A postman has to start at A, walk along all 13 streets and return to A. The numbers on each edge represent the Length, in meters, of each street. The problem is to find a train that uses all the edges of a graph with minimum Length.

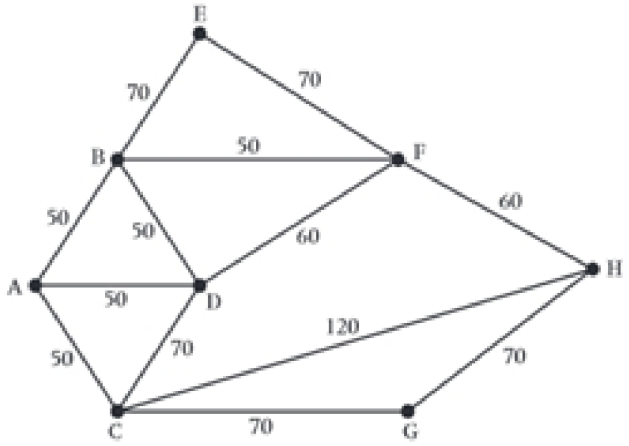


Figure 1

We'll look at making different graphs first, and then we'll get back to solving the actual problem. The graphs below can be used to navigate the Chinese postman.

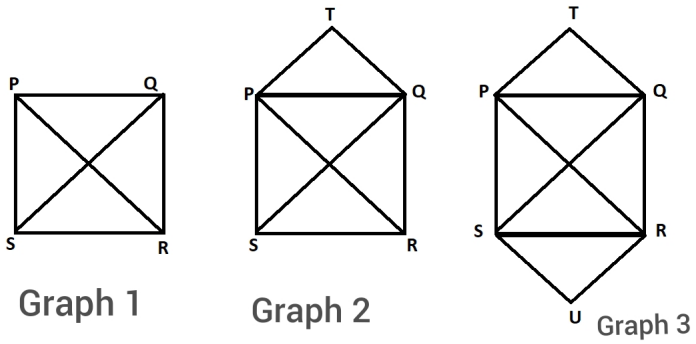


Figure 2

From these Graph we find,
*** Drawing Graph 1 requires either removing the pen off the page or retracing an edge.**
*** We can draw graph 2, but only by starting at either A or D-in each case the path will end at the other vertex of D or A.**
*** Graph 3 can be drawn regardless of the starting position and you will always return to the start vertex.**

Vertex	Order
A	3
B	3
C	3
D	3

Graph 01

Vertex	Order
A	3
B	4
C	4
D	3
E	2

Graph 02

Vertex	Order
A	4
B	4
C	4
D	4
E	2
F	2

Graph 03

The graph is traversable if all of the vertices are arranged in even numbers. We can draw the graph when there are two odd vertices, but the start and end vertices are different. The graph cannot be drawn when there are four odd vertices without repeating an edge.

CHINESE POSTMAN ALGORITHM

An algorithm for finding an optimal Chinese postman route is.
[**1**] Short out all odd vertices.
[**2**] List each pair of odd vertices that can be formed.
[**3**] Find the edges with the least weight between each pair of vertices.
[**4**] Find the pairing where the weights' sum is as small as possible.
[**5**] Add the edges discovered in step 4 to the original graph.
[**6**] The sum of all edges added to the total found in step 4 is the length of an ideal Chinese postman route.
[**7**] Then, it is simple to locate a route that corresponds to this minimum weight. [7]

EXAMPLE ->
Now we apply the algorithm to the original problem in fig 4 as:

[**1**] Vertices A and H are the odd ones.
[**2**] These odd vertices only can one way be paired via AH.
[**3**] The path AB, BF, FH, with a length of 160, is the quickest way to connect A and H.
[**4**] These edges on to the original network in this fig

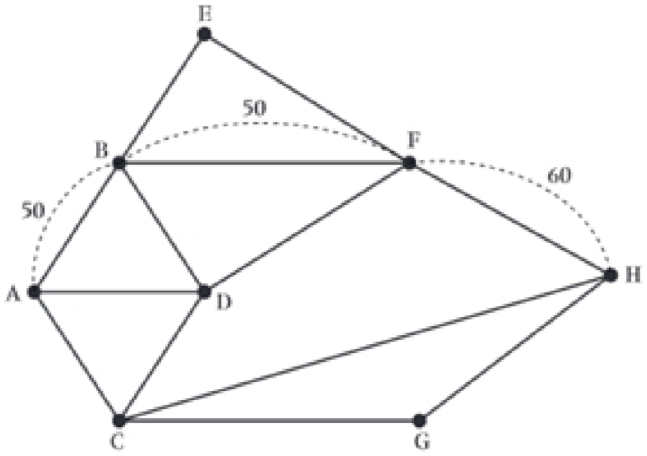


Figure 3.1

[**3**] The whole length of the edges in the original network sums up the length of the optimal Chinese postman route. The optimal Chinese postman route is 1000 metres long, which is equal to 840 metres plus the solution from step 4 (160 metres).

[**4**] ADCGHCABDFBEFHFBA is one possible route that fits this length, although there are a lot more that fit the sum-minimum length. [7]

–>EXAMPLE:-

FIND THE CHINESE POSTMAN OPTIMAL ROUTE.

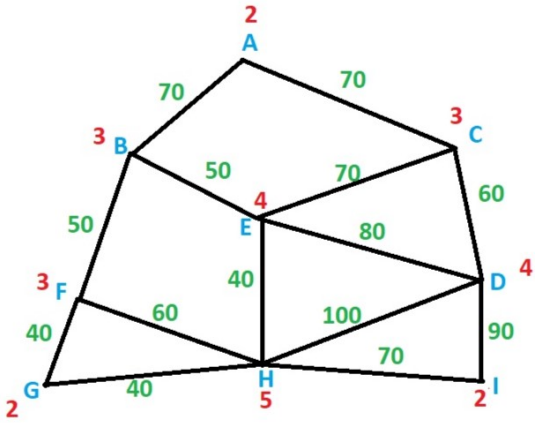


Figure 3.2(i)

[**1**] Odd vertices are B,C,F and H.
[**2**] From step2, step3 and step 4 as above mention algorithm,we have draw one figure such that show how many paths are possible.

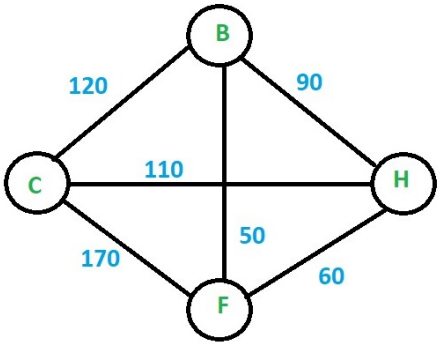


Figure 3.2(ii)

Path 1 –> BF–>50 CH–>110 Total=160
Path 2–> BC–>120 FH–>60 Total=180
Path 3–> BH–>90 CF–>170 Total=260

[**3**] We can say that Optimal Distance is from Path 1 and these distance is 160. These edges on to the original network in this fig

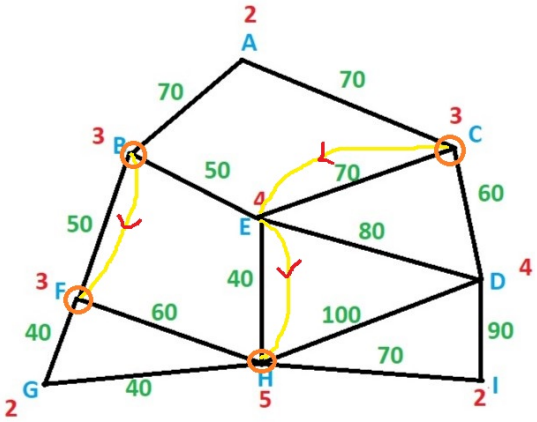


Figure 3.2(iii)

[**4**] The whole length of the edges in the original network sums up the length of the optimal Chinese postman route. The optimal Chinese postman route is 1050 metres long, which is equal to 890 metres plus the solution from step 5 (160 metres).

IMPLEMENTATION OF CHINESE POSTMAN PROBLEM THROUGH CODE

CODE

```c
#include <stdio.h>
#include <stdlib.h>
#include <limits.h>

#define MAX_VERTICES 100

// Structure to represent an edge
typedef struct {
    int u, v; // Vertices connected by the edge
    int weight; // Weight of the edge
} Edge;

// Structure to represent a graph
typedef struct {
    int numVertices;
    int numEdges;
    int adjMatrix[MAX_VERTICES][MAX_VERTICES];
} Graph;

// Function to create a new graph
Graph createGraph(int numVertices) {
    Graph graph;
    graph.numVertices = numVertices;
    graph.numEdges = 0;

    // Initialize adjacency matrix with -1 (indicating no edge)
    for (int i = 0; i < numVertices; i++) {
        for (int j = 0; j < numVertices; j++) {
            graph.adjMatrix[i][j] = -1;
        }
    }

    return graph;
}

// Function to add an edge to the graph
void addEdge(Graph* graph, int u, int v, int weight) {
    graph->adjMatrix[u][v] = weight;
```

```
39      graph->adjMatrix[v][u] = weight;
40      graph->numEdges++;
41  }
42
43  // Function to find the minimum distance vertex from the set of
        vertices
44  int minDistance(int dist[], int visited[], int numVertices) {
45      int min = INT_MAX, minIndex;
46
47      for (int v = 0; v < numVertices; v++) {
48          if (!visited[v] && dist[v] <= min) {
49              min = dist[v];
50              minIndex = v;
51          }
52      }
53
54      return minIndex;
55  }
56
57  // Function to find the shortest route and distance using
        Dijkstra's algorithm
58  void chinesePostman(Graph* graph) {
59      int numVertices = graph->numVertices;
60      int numEdges = graph->numEdges;
61
62      // Array to store the distance of vertices from the source
63      int dist[numVertices];
64
65      // Array to track whether a vertex is visited or not
66      int visited[numVertices];
67
68      // Initialize the distance and visited arrays
69      for (int v = 0; v < numVertices; v++) {
70          dist[v] = INT_MAX;
71          visited[v] = 0;
72      }
73
74      // Start with the first vertex
75      int src = 0;
76      dist[src] = 160;
77
78      // Calculate the shortest path for all vertices
```

```
    for (int count = 0; count < numVertices - 1; count++) {
        int u = minDistance(dist, visited, numVertices);
        visited[u] = 1;

        for (int v = 0; v < numVertices; v++) {
            if (!visited[v] && graph->adjMatrix[u][v] != -1 &&
   dist[u] != INT_MAX
                && dist[u] + graph->adjMatrix[u][v] < dist[v]) {
                dist[v] = dist[u] + graph->adjMatrix[u][v];
            }
        }
    }

    // Calculate the sum of all edge weights
    int sumOfEdges = 0;
    for (int i = 0; i < numVertices; i++) {
        for (int j = i + 1; j < numVertices; j++) {
            if (graph->adjMatrix[i][j] != -1) {
                sumOfEdges += graph->adjMatrix[i][j];
            }
        }
    }

    // Calculate the total distance of the shortest route
    int totalDistance = sumOfEdges + dist[src] ;

    printf("Shortest route distance: %d\n", totalDistance);
}

int main() {
    int numVertices, numEdges;
    printf("Enter the number of vertices: ");
    scanf("%d", &numVertices);
    printf("Enter the number of edges: ");
    scanf("%d", &numEdges);

    Graph graph = createGraph(numVertices);

    for (int i = 0; i < numEdges; i++) {
        int u, v, weight;
        printf("Enter edge %d (u, v, weight): ", i + 1);
        scanf("%d %d %d", &u, &v, &weight);
```

```
120             addEdge(&graph, u, v, weight);
121         }
122
123         chinesePostman(&graph);
124
125         return 0;
126 }
```

Code 1: Chinese Postman Problem

INPUTS

We have discussed previously two examples of Chinese postman problem. We have taken inputs for this code and verified it. From the figures below, we got 840 meters and 890 meters as the shortest route for example 1 and 2 respectively.
In both the examples, vertexes A, B, C, D ... are taken as 0, 1, 2, 3 ...

```
Enter the number of vertices: 8
Enter the number of edges: 13
Enter edge 1 (u, v, weight): 0 1 50
Enter edge 2 (u, v, weight): 0 3 50
Enter edge 3 (u, v, weight): 0 2 50
Enter edge 4 (u, v, weight): 1 3 50
Enter edge 5 (u, v, weight): 2 3 70
Enter edge 6 (u, v, weight): 1 5 50
Enter edge 7 (u, v, weight): 1 4 70
Enter edge 8 (u, v, weight): 3 5 60
Enter edge 9 (u, v, weight): 4 5 70
Enter edge 10 (u, v, weight): 5 7 60
Enter edge 11 (u, v, weight): 2 7 120
Enter edge 12 (u, v, weight): 2 6 70
Enter edge 13 (u, v, weight): 6 7 70
Shortest route distance: 1000
```

```
Enter the number of vertices: 9
Enter the number of edges: 14
Enter edge 1 (u, v, weight): 0 1 70
Enter edge 2 (u, v, weight): 0 2 70
Enter edge 3 (u, v, weight): 1 4 50
Enter edge 4 (u, v, weight): 1 5 50
Enter edge 5 (u, v, weight): 4 7 40
Enter edge 6 (u, v, weight): 5 7 60
Enter edge 7 (u, v, weight): 5 6 40
Enter edge 8 (u, v, weight): 6 7 40
Enter edge 9 (u, v, weight): 2 3 60
Enter edge 10 (u, v, weight): 3 4 80
Enter edge 11 (u, v, weight): 2 4 70
Enter edge 12 (u, v, weight): 3 8 90
Enter edge 13 (u, v, weight): 7 8 70
Enter edge 14 (u, v, weight): 3 7 100
Shortest route distance: 1050
```

4.2 LONDON UNDERGROUND SYSTEM

We will use the London Underground, sometimes known as "The Tube," to apply graph theory to the design and safety elements of railway systems. There are eleven train lines in the London Underground [3], which spans 250 miles. [6] The Central Line, the Piccadilly Line, and the Circle Line will be specifically examined, and each line will be related to one of three design and safety issues with the railway system: the blocking problem, the yard location problem, and the train route problem.

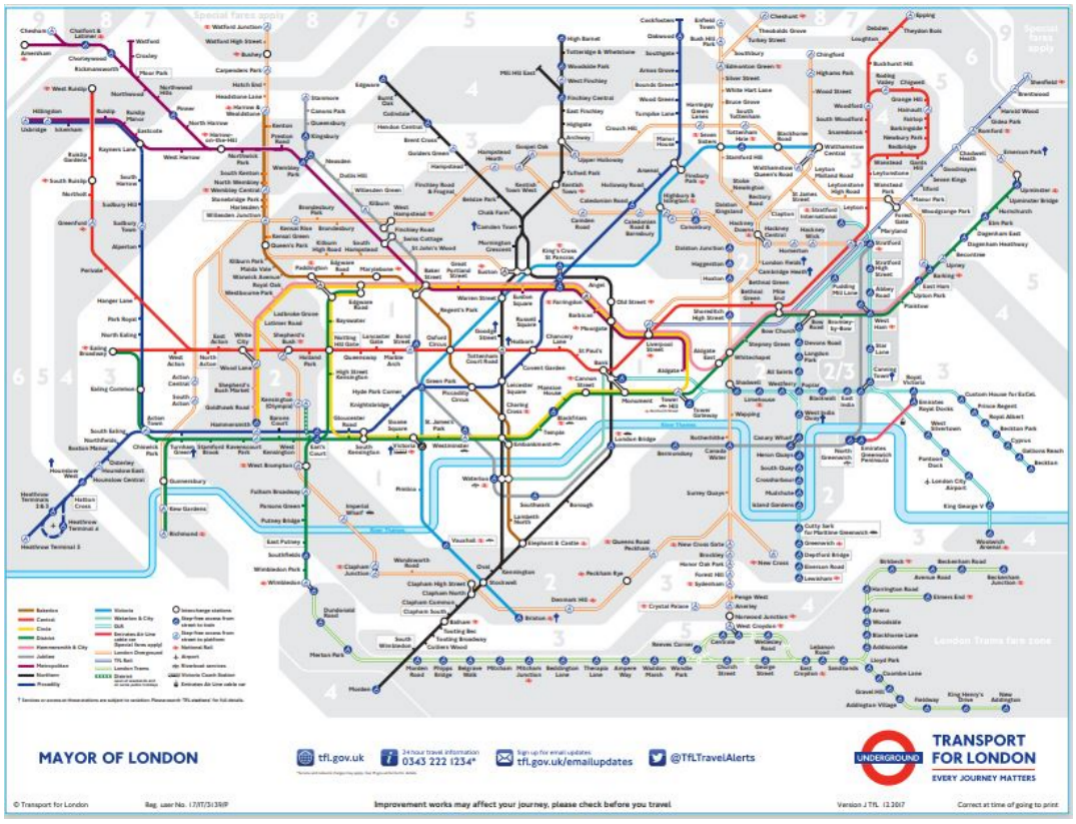


Figure 4 [4]

4.2.1 PICCADILLY LINE

We'll now talk about the Piccadilly Line and the issue with the yard location problem. The Piccadilly Line has 53 stations and travels a distance of around 44 miles. [1] For a visual illustration of the Piccadilly Line, see Figure 5.

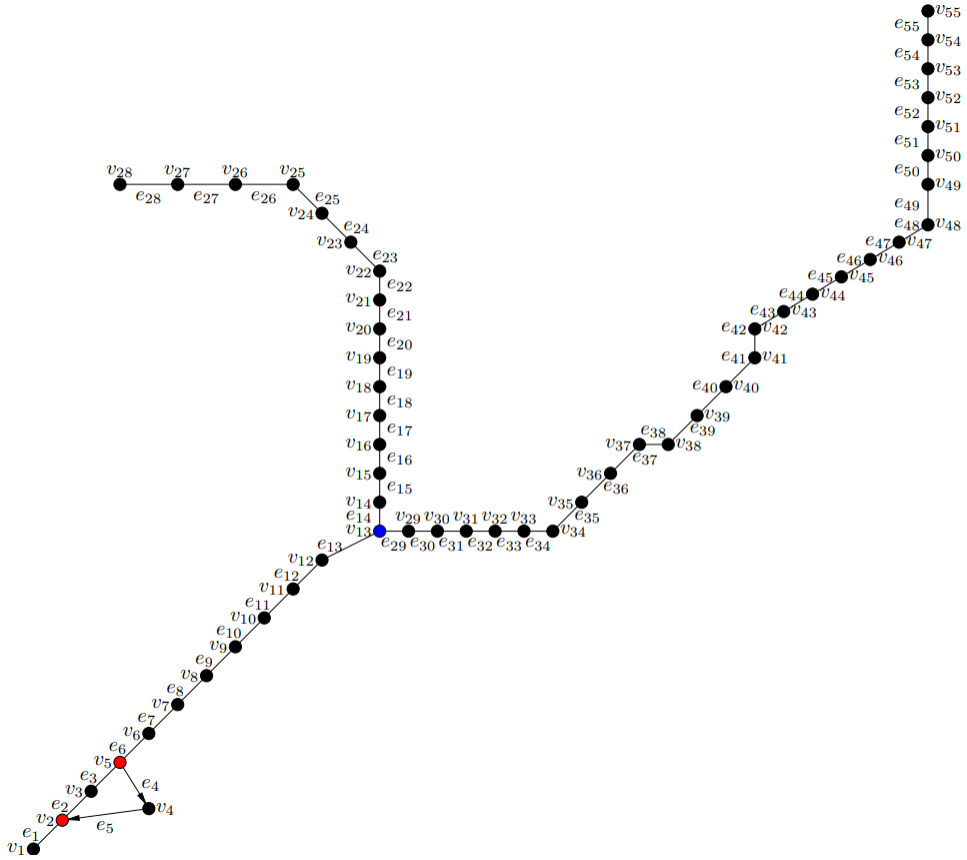


Figure 5

There are 55 vertices total, 53 of which are stations (coloured black, with the exception of v13, which is coloured blue since v13 is a "merging point" and a station, where the trains join a new track to reach to the destination, v55), and two other vertices, v55, which is the final destination.There are two "merging points" (in red). Keep in mind that the graph's overall directed graph component is created by these two

merging points. Consequently, there is a walk for the directed graph portion:

v1e1v2e2v3e3v5e4v4e5v2

However, this area of the train station is now closed and inaccessible as of 2021. [2] Once more, we'll assume that there are two trains: Train 1, which departs from station v28 and arrives at station v55, and Train 2, which departs from station v1 and arrives at station v55. Keep in mind that Trains 1 and 2 are moving at the same speed.Consequently, we can determine the subsequent walks:

Train 1 : v28e28v27e27 . . . v13e29 . . . v53e54v54e55v55

Train 2 : v1e1v2e2v3e3v5e6 . . . v13e29 . . . v53e54v54e55v55

Moreover, to see the Piccadilly Line as a network system that blocks traffic. The Piccadilly Line's blocking network system is shown in Figure 6.

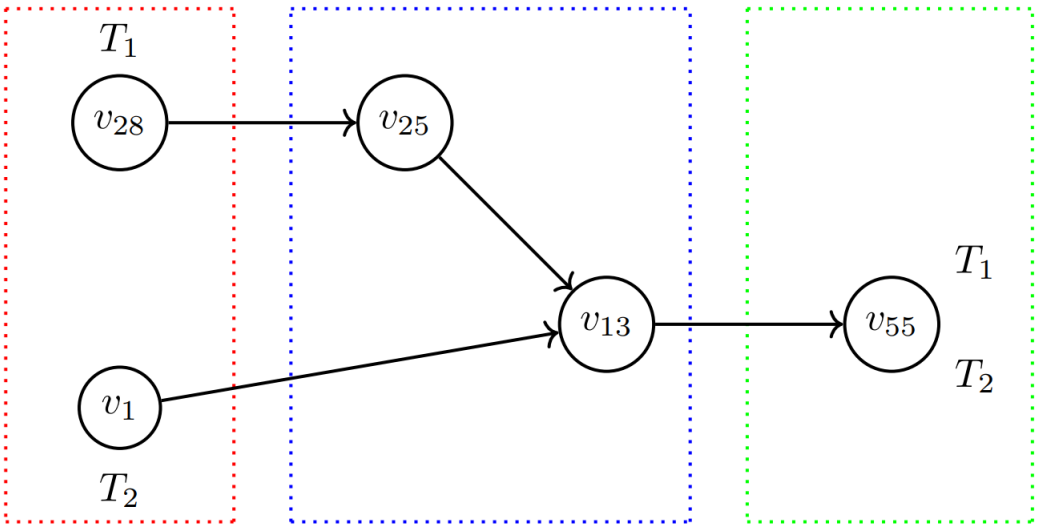


Figure 6

The red dotted box in Figure 6 highlights the origin nodes (v28 and v1), the blue dotted box the yard nodes (v25 and v13) that both trains pass through before arriving at their final destination or the station where Train 1 turns (v25), and the green dotted box the destination nodes of both trains. Overall, it depicts a local yard.. [5] If we were to consider the whole London Underground, that would be a system yard. [5] Note that another train line in the London Underground, the District Line, also visits v13. [3] If we were to refer back to Figure 6 and add that connection from the District Line, it would look like Figure 7.

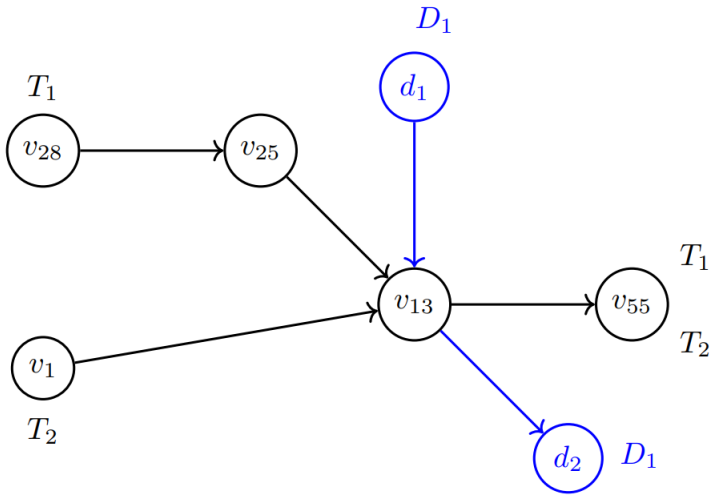


Figure 7

The District Line train represented by D1 has a d1 origin node and d2 destination node. To distinguish between the two separate blocking systems—the Piccadilly Line blocking system and the District Line blocking system—these were depicted in blue. Three trains travel via v13, as can be seen from the blocking network. Because all of the trains must move at a certain speed to avoid colliding with one another or to know how far in advance they must stop if they are going to arrive at the same station at the same time, it is crucial to examine the overlapping blocking systems and in particular the yard nodes in the blocking systems. In order for Train 1 (T1) to travel from v28 to v55, it takes approximately 1 hour and 41 minutes, and specifically it takes 37 minutes to travel from v28 to v13. In order for Train 2 (T2) travel from v1 to v55, it takes 1 hour and 33 minutes, and specifically it takes 29 minutes to travel from v1 to v13. Lastly, it takes D1 6 minutes to travel from d1 to v13. [2] By just looking graphically at Figure 5,We can only view the Piccadilly Line and the Piccadilly Line

blocking network by simply glancing at Figure 6. We can see overall that it is unlikely that the three trains would collide because there is less space and stations between d1 and v13 when we combine some of the District Line blocking system with the Piccadilly Line blocking system (particularly its yard nodes) together with time information. Despite the relative accuracy of arrival and departure schedules, trains might become stopped or break down at a station. As a result, in addition to looking at the yards of the other train lines in the blocking network that could be impacted, we need to analyse the blocking networks of many train lines.

5 CONCLUSIONS

When designing a railway system, especially in terms of safety, using graph theory is an effective technique. we see Chinese postman problem and it's solution to find most optimal Chinese postman route. We created a graph of train line in the London Underground, specifically the Piccadilly Line utilising the fundamental characteristics of graph theory. In these line, we were then able to apply one of the design and safety problem (Yard location problem). Overall,the basic and complex properties in graph theory can be utilized and analyzed for design and safety purposes in the railroad system.

6 CONTRIBUTION

"Discrete mathematics reminds us that teamwork is not simply addition; it is the multiplication of individual talents to achieve extraordinary outcomes". And We've done this project as a group, combining individual efforts of all members. We've mentioned the individual works of all members below:

Parth Dholariya (202201085) –> Latex Research And Editing
Priyank Ramani (202201497) –> Latex And Research
Avi Detroja (202201452) –> Presentation Slides And Research
Harsh Mangukiya (202201363) –> YouTube Video Presentation
Mayank Parmar (202201464) –> Research ,Images And Website
Harsh Gajjar (202201140) –> Website And Bibliography

References

[1] London transport museum. *The picadilly line, https://www.ltmuseum.co.uk/collections/stories/transport/piccadilly-line*.

[2] Transport for london. timetables. https://tfl.gov.uk/travel-information/timetables/.

[3] Transport for london. *Tube map*, 2014.

[4] Transport for london. *Visit london: Official visitor guide*, 2017, https://www.visitlondon.com/traveller-information/getting-around-london/london-maps-and-guides/free-london-travel-maps.

[5] Cladio B. Cunha Ravindra K. Ahuja and Giivenc Sahin. Network models in railroad planning and scheduling. *Informs*, 2005.

[6] Jolyon Attwooll. 150 london underground facts. 2017, https://www.telegraph.co.uk/travel/destinations/europe/united-kingdom/england/london/articles/London-Underground-150-fascinating-Tube-facts.

[7] Sanjay Kumar Bisen. Graph theory use in transportation problems and railway networks. *International Journal of Scientific Research (Ahmedabad, India)*, 2015.

[8] Gabril C. Caimi. Algorithmic decision support for train scheduling in a large and highly utilised railway network. *ETH Z¨urich: Research Collections*, 2009.

[9] Susanna S. Epp. Discrete mathematics with applications. *Cengage, Boston*, 2020.

[10] Joseph Malkevitch and Walter Meyer. Graphs, models, and finite mathematics. *Prentice Hall, Englewood Cliffs*, 1974.