Optimal Path Selection of Hong Kong's MTR Route Based on Dijkstra's Algorithm

Achmad Dani Nursanto
Department of Computer Science and Electronics, Universitas Gadjah Mada
Building C, 4th Floor North
Yogyakarta, Indonesia
achmad.dani0703@mail.ugm.ac.id

Izaaz Rahman Akbar
Department of Computer Science and Electronics, Universitas Gadjah Mada
Building C, 4th Floor North
Yogyakarta, Indonesia
izaaz.rahman.akbar@mail.ugm.ac.id

Izzeldin Rayyan Bastian
Department of Computer Science and Electronics, Universitas Gadjah Mada
Building C, 4th Floor North
Yogyakarta, Indonesia
izzeldin.rayyan.bastian@mail.ugm.ac.id

Rabbani Nur Kumoro
Department of Computer Science and Electronics, Universitas Gadjah Mada
Building C, 4th Floor North
Yogyakarta, Indonesia
rabbani.nur.kumoro@mail.ugm.ac.id

Salsabila Alyanitazahra
Department of Computer Science and Electronics, Universitas Gadjah Mada
Building C, 4th Floor North
Yogyakarta, Indonesia
salsabila.alyanitazahra@mail.ugm.ac.id

Abstract — **Mass Transit Railway (MTR) System in Hong Kong connects some critical districts to historical places, business places, tourist spots, and shopping malls. The train services run independently but have interchanges to integrate from one different MTR line. This may lead the traveler to face difficulties when they are choosing the incorrect destination stations, especially on different MTR lines which contribute to time-consuming and high costing. In this project, we proposed the use of Dijkstra's Algorithms to provide a more effective and intelligent shortest path route finder system to provide the solution for travelers to reach the desired destination.**

Keywords — Dijkstra Algorithm, Shortest Path-Route, MTR, Maps, Data Structures

I. Introduction

Mass Transit Railroad (MTR) is the leading public transport network utilized in Hong Kong. Operated by MTR Corporation Limited (MTRCL), it consists of heavy rail, light rail, and feeder bus services centered on a high-speed transportation network of 10 routes that provide services to the urban areas of Hong Kong Island, Kowloon, and the New Territories. The 2018 system consists of railroads with an approximate length of 230.9 km. This includes 166 stations, of which 98 are heavy rail stations and 68 are light rail stations.

Besides the ten railway lines, a light rail network serves the New Territories communities of Tuen Mun and Yuen Long, while a bus fleet provides convenient feeder services. The Corporation also runs the Airport Express, a dedicated high-speed rail link that provides the quickest connections to Hong Kong International Airport and AsiaWorld-Expo, the city's newest exhibition, and conference center. MTR trains' operational reliability is unaffected by traffic conditions, so passengers will always arrive on time, barring severe weather conditions. Furthermore, MTR trains operate 19 hours a day, seven days a week, from early morning to 1:00 a.m. the next morning.

II. Problem

As to no surprise, tourism is a huge industry in itself. The concept of traveling to foreign countries with means of entertainment is widely admired. However, as tourists, roaming around foreign lands could be quite the predicament. Public transportation is a common amenity that tourists will use to traverse around due to its quickness and affordability. An example would be the Mass Transit Railroad (MTR) network in Hong Kong which offers a wide variety of routes that might confuse the common tourist. Tourists would want to balance convenience and spending while planning their trip. This is where Dijkstra's algorithm could be used to find the optimal path while traversing the MTR.

As an example, suppose we are helping a group of tourists in Hong Kong. They are planning to visit Hong Kong Disneyland. They want their primary source of transportation to be a-reliable-yet-affordable public transportation, thus they chose the MTR (Mass Transit Railway). The nearest MTR station from their hotel is Kennedy Town station. They also want to calculate the shortest-path route, alternative-path route, route length, and estimated time from the nearest MTR station to Disneyland Park. This allows the tourists to plan for stops in between stations which might be useful if they want to use certain amenities at a station or the local area. To make their itinerary work, they decided to use Dijkstra's algorithm to figure out this problem.

III. Define the Problem into a Graph

To use Dijkstra's algorithm, we must first construct the graph by declaring every node, path, and connected node. A simple graph representation of the MTR system map is shown below. The goal is to find the shortest route, longest route, and all possible routes along with their prices from Kennedy Town station to Disneyland Resort station.

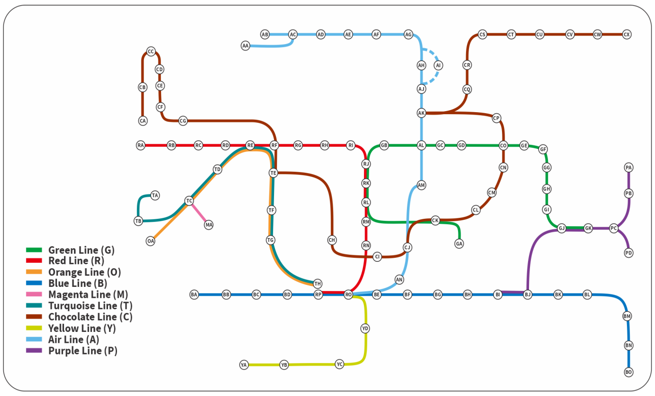


Fig. 3.1 Graph Representation

Each node represents a station in the MTR map. Therefore, we need to give each node a name or a String to represent the station during the final output. The following are the names of the stations and their respective nodes, grouped based on their own lines:

Green Line

Code	Station Name
GA/70	Whampoa
GB/34	Shek Kip Mei
GC/36	Lok Fu
GD/37	Wong Tai Sin
GE/39	Choi Hung
GF/40	Kowloon Bay
GG/61	Ngau Tau Kok
GH/62	Kwun Tong
GI/63	Lam Tin
GJ/66	Yau Tong
GK/67	Tiu Keng Leng

Red Line

Code	Station Name
RA/25	Tsuen Wan
RB/26	Tai Wo Hau
RC/27	Kwai Hing
RD/28	Kwai Fong
RE/29	Lai King
RF/30	Mei Foo
RG/31	Lai Chi Kok
RH/32	Cheung Sha Wan
RI/33	Sham Shui Po
RJ/50	Prince Edward
RK/51	Mong Kok
RL/52	Yau Ma Tei
RM/53	Jordan
RN/54	Tsim Sha Tsui
RO/79	Admiralty
RP/78	Central

Orange Line

Code	Station Name
OA/46	Tung Chung

Blue Line

Code	Station Name
BA/74	Kennedy Town
BB/75	HKU
BC/76	Sai Ying Pun
BD/77	Sheung Wan
BE/80	Wan Chai
BF/81	Causeway Bay
BG/82	Tin Hau
BH/83	Fortress Hill
BI/84	North Point
BJ/85	Quarry Bay
BK/86	Tai Koo
BL/87	Sai Wan Ho
BM/92	Shau Kei Wan
BN/93	Heng Fa Chuen
BO/94	Chai Wan

Magenta Line

Code	Station Name
MA/45	DisneyLand Resort

Turquoise Line

Code	Station Name
TA/42	AsiaWorld-Expo
TB/44	Airport

TC/43	Sunny Bay
TD/41	Tsing Yi
TE/47	Nam Cheong
TF/48	Olympic
TG/49	Kowloon
TH/73	Hong Kong

Chocolate Line	
Code	**Station Name**
CA/97	Tuen Mun
CB/96	Siu Hong
CC/18	Tin Shui Wai
CD/19	Long Ping
CE/20	Yuen Long
CF/21	Kam Sheung Road
CG/22	Tsuen Wan West
CH/55	Austin
CI/72	East Tsim Sha Tsui
CJ/71	Hung Hom
CK/60	Ho Man Tin
CL/59	To Kwa Wan
CM/58	Sung Wong Toi
CN/57	Kai Tak
CO/38	Diamond Hill
CP/24	Hin Keng
CQ/17	Che Kung Temple
CR/15	Sha Tin Wai
CS/7	City One
CT/8	Shek Mun
CU/9	Tai Shui Hang
CV/10	Heng On
CW/11	Ma On Shan
CX/12	Wu Kai Sha

Yellow Line	
Code	**Station Name**
YA/89	South Horizons
YB/90	Lei Tung
YC/91	Wong Chuk Hang
YD/88	Ocean Park

Air Line	
Code	**Station Name**
AA/0	Lok Ma Chau
AB/1	Lo Wu
AC/2	Sheung Shui
AD/3	Fanling
AE/4	Tai Wo
AF/5	Tai Po Market
AG/6	University
AH/13	Fo Tan
AI/14	Racecourse
AJ/16	Sha Tin
AK/23	Tai Wai
AL/35	Kowloon Tong
AM/56	Mong Kok East
AN/95	Exhibition Center

Purple Line	
Code	**Station Name**
PA/64	Po Lam
PB/65	Hang Hau
PC/68	Tseung Kwan O
PD/69	LOHAS Park

IV. Solution and Method

For this implementation, the Java programming language will be used. This implementation will showcase which stations a person will have to pass through in order to reach their destination. The user might want to find the fastest route possible or maybe would want to plan a few stops during the trip.

The user will be greeted with a welcome message and is then required to choose which starting and final station they would like to check for possible routes.

4.1 Code Functions

There are two classes: dijkstra and Main. One is used for graph representation and Dijkstra's algorithm implementation while the other is for testing. Below are the functions that are used for some features:

1. dijkstra method

This is the constructor for the object and it initiates the attributes of the object such as the number of nodes(stations) and the matrices that are being used to represent the graph. There are two matrices to represent the same graph, the first matrix holds information about distance between stations while the second matrix holds information about the time between stations It also immediately adds edges to the graph, which means our graph is predetermined and unchangeable.

```java
public class Dijkstra {
  int N;
  int[][] Matrix;
  int[][] timeMatrix;
  String[] Station = {"Lok Ma Chau", "Lo Wu", "Sheung Shui",
  "Fanling", "Tai Wo", "Tai Po Market", "University", "City One",
  "Shek Mun", "Tai Shui Hang", "Heng On", "Ma On Shan", "Wu Kai Sha",
  "Fo Tan", "Racecourse", "Sha Tin Wai", "Sha Tin", "Che Kung Temple",
  "Tin Shui Wai", "Long Ping", "Yuen Long", "Kam Sheung Road",
  "Tsuen Wan West", "Tai Wai", "Hin Keng", "Tsuen Wan", "Tai Wo Hau", "Kwai Hing",
  "Kwai Fong", "Lai King", "Mei Foo", "Lai Chi Kok", "Cheung Sha Wan",
   "Sham Shui Po", "Shek Kip Mei", "Kowloon Tong", "Lok Fu", "Wong Tai Sin", "Diamond Hill",
  "Choi Hung", "Kowloon Bay", "Tsing Yi", "AsiaWorld-Expo", "Sunny Bay",
  "Airport", "Disneyland Resort", "Tung Chung", "Nam Cheong", "Olympic", "Kowloon",
  "Prince Edward", "Mong Kok", "Yau Ma Tei", "Jordan", "Tsim Sha Tsui",
  "Austin", "Mong Kok East", "Kai Tak", "Sung Wong Toi", "To Kwa Wan", "Ho Man Tin",
  "Ngau Tau Kok", "Kwun Tong", "Lam Tin", "Po Lam", "Hang Hau", "Yau Tong",
  "Tiu Keng Leng", "Tseung Kwan O", "LOHAS Park", "Whampoa", "Hung Hom", "East Tsim Sha Tsui",
  "Hong Kong", "Kennedy Town", "HKU", "Sai Ying Pun", "Sheung Wan", "Central",
  "Admiralty", "Wan Chai", "Causeway Bay", "Tin Hau", "Fortress Hill", "North Point", "Quarry Bay",
  "Tai Koo", "Sai Wan Hoo", "Ocean Park", "South Horizons", "Lei Tung", "Wong Chuk Hang",
  "Shau Kei Wan", "Hang Fa Chuen", "Chai Wan", "Exhibition Centre", "Siu Hong", "Tuen Mun"};
```

```java
public Dijkstra(int N) {
  this.N = N;
  Matrix = new int[N][N];
  timeMatrix = new int[N][N];
  addEdge(from: 0, to: 2, len: 6, time: 7);
  addEdge(from: 1, to: 2, len: 3, time: 5);
  addEdge(from: 2, to: 3, len: 3, time: 5);
  addEdge(from: 3, to: 4, len: 3, time: 5);
  addEdge(from: 4, to: 5, len: 3, time: 5);
  addEdge(from: 5, to: 6, len: 3, time: 4);
  addEdge(from: 6, to: 13, len: 7, time: 9);
  addEdge(from: 6, to: 14, len: 9, time: 11);
  addEdge(from: 7, to: 8, len: 4, time: 6);
  addEdge(from: 7, to: 15, len: 7, time: 9);
  addEdge(from: 8, to: 9, len: 4, time: 6);
  addEdge(from: 9, to: 10, len: 4, time: 6);
  addEdge(from: 10, to: 11, len: 4, time: 6);
  addEdge(from: 11, to: 12, len: 4, time: 6);
  addEdge(from: 13, to: 16, len: 3, time: 5);
  addEdge(from: 14, to: 16, len: 4, time: 6);
  addEdge(from: 15, to: 17, len: 4, time: 6);
  addEdge(from: 16, to: 23, len: 3, time: 5);
  addEdge(from: 17, to: 23, len: 7, time: 9);
  addEdge(from: 18, to: 19, len: 1, time: 3);
  addEdge(from: 18, to: 96, len: 5, time: 7);
```

2. void addEdge

This adds a new edge and also determines the weight of it based on its real-life counterpart. The addEdge method uses the similar method used to input new edges and nodes to a graph object. The constructor consists of 4 variables that will hold the source node to destination node, the weight of the edge, and a time variable used to calculate the sum of time needed to travel from one station to another

```java
public void addEdge(int from, int to, int len, int time) {
  Matrix[from][to] = len;
  Matrix[to][from] = len;
  timeMatrix[from][to] = time;
  timeMatrix[to][from] = time;
}
```

3. void dijkstraAlg

This is a function that implements Dijkstra's shortest path algorithm for a graph represented using an adjacency matrix. The algorithm is called recursively to print multiple paths. The first iteration prints the shortest path and the next iterations print the alternative paths by deleting the edge that connects the destination and its parent of the previous path. It loops recursively until no path can be made.

```java
public void dijkstraAlg(int src, int dst) {
  int[][] distance = new int[N][2];
  int[][] time = new int[N][2];
  boolean[] fixed = new boolean[N];
  Stack<Integer> stack = new Stack<Integer>();

  for (int i = 0; i < N; i++) {
    distance[i][0] = Integer.MAX_VALUE;
    distance[i][1] = -1;
    fixed[i] = false;
  }
  distance[src][0] = 0;

  while (true) {
    int marked = minIndex(distance, fixed);
    if (marked < 0)
      break;
    if (distance[marked][0] == Integer.MAX_VALUE)
      break;
    fixed[marked] = true;
    for (int j = 0; j < N; j++) {
      if (Matrix[marked][j] > 0 && !fixed[j]) {
        int newDistance = distance[marked][0] + Matrix[marked][j];
        int newTime = time[marked][0] + timeMatrix[marked][j];
        if (newDistance < distance[j][0]) {
          distance[j][0] = newDistance;
          distance[j][1] = marked;
          time[j][0] = newTime;
          time[j][1] = marked;
        }
      }
    }
  }
```

```java
  int idx = dst;
  int parent = distance[dst][1];
  while (distance[idx][1] != -1) {
    stack.push(distance[idx][1]);
    idx = distance[idx][1];
  }

  while (!stack.empty())
    System.out.print(Station[stack.pop()] + " > ");

  if (distance[dst][0] == Integer.MAX_VALUE) {
    System.out.printf(format: "%84s",...args: "No other routes found.\n");
  } else {
    Matrix[parent][dst] = 0;
    Matrix[dst][parent] = 0;
    System.out.println(Station[dst]);
    System.out.printf(format: "\n%90s\n\n","ROUTE LENGTH : " + distance[dst][0] + " | TIME = " + time[dst][0] + " minutes");
    System.out.printf(format: "\n\n%82s\n\n",...args: "[ ALTERNATIVE PATH ]");
    dijkstraAlg(src, dst);
  }
}
```

4. int minIndex

This is a utility function to find the vertex with minimum distance.

```java
public int minIndex(int[][] distance, boolean[] fixed) {
  int idx = 0;
  for (; idx < fixed.length; idx++) {
    if (!fixed[idx])
      break;
  }
  if (idx == fixed.length)
    return -1;
  for (int i = idx + 1; i < fixed.length; i++) {
    if (!fixed[i] && distance[i][0] < distance[idx][0])
      idx = i;
  }
  return idx;
}
```

5. main method

```java
import java.util.Scanner;

public class Main {
  Run | Debug
  public static void main(String[] args) {
    Scanner scan = new Scanner(System.in);

    System.out.printf(format: "\n%104s\n\n",...args: "Welcome to Hong Kong's Mass Transit Railway Guidance System.");
    System.out.printf(format: "%106s\n\n",...args: "We will help you find the ideal route to reach your destination.");

    do{
      Dijkstra MTR = new Dijkstra(N: 98);

      System.out.println();
      for(int i=0; i<=140;i++){
        System.out.print(s: "-");
      }
      System.out.printf(format: "\n\n%85s\n\n",...args: "-=[ LIST OF STATIONS ]=-");

      for(int i = 0; i < 98; i++){
        if((i + 1) % 4 == 0){
          System.out.println("\t["+(i+1)+"]"+ " " + MTR.Station[i]);
        } else if((i + 1) % 3 == 0){
          System.out.printf(format: "%15s%-20s", "["+(i + 1)+"]"+ " ", MTR.Station[i]);
        } else if((i + 1) % 2 == 0){
          System.out.printf(format: "%15s%-20s", "["+(i + 1)+"]"+ " ", MTR.Station[i]);
        } else{
          System.out.printf(format: "%15s%-20s", "["+(i + 1)+"]"+ " ", MTR.Station[i]);
        }
      }
```

```java
      System.out.println(x: "\n");
      for(int i=0; i<=140;i++){
        System.out.print(s: "-");
      }

      System.out.println();
      System.out.printf(format: "\n%85s",...args: "Choose Starting Station: ");
      int src = scan.nextInt();
      while(src < 1 || src > 98){
        System.out.printf(format: "\n%89s\n",...args: "Please enter a valid input.");
        System.out.printf(format: "\n%85s",...args: "Choose Starting Station: ");
        src = scan.nextInt();
      }
      System.out.printf(format: "\n%88s",...args: "Choose Destination Station: ");
      int dst = scan.nextInt();
      while(dst < 1 || dst > 98){
        System.out.printf(format: "\n%89s\n",...args: "Please enter a valid input.");
        System.out.printf(format: "\n%88s",...args: "Choose Destination point: ");
        dst = scan.nextInt();
      }

      System.out.println();
      for(int i=0; i<=140;i++){
        System.out.print(s: "-");
      }
```

```java
      System.out.println();
      for(int i=0; i<=140;i++){
        System.out.print(s: "-");
      }

      System.out.println();
      System.out.printf(format: "\n\n%78s\n\n",...args: "[ MAIN PATH ]");
      MTR.dijsktraAlg(src - 1, dst - 1);

      System.out.printf(format: "\n\n%89s\n",...args: "Would you like to continue?");
      System.out.printf(format: "\n%68s",...args: "[1] Yes");
      System.out.printf(format: "\n%67s",...args: "[2] No");
      System.out.printf(format: "\n\n%77s",...args: "Insert Number: ");
      int i = scan.nextInt();
      if(i != 1)
        break;
    }while(true);

    System.out.printf(format: "\n\n\n%107s\n\n",...args: "Thank You for using the Mass
    for(int i=0; i<=140;i++){
      System.out.print(s: "-");
    }
    scan.close();
}
```

4.2 Output

From the functions that are used to create the program, we can see the results in the form of a console program.

1. Users can input their desired location.
2. Users can find out their routes throughout the output. The output will display the Main Path and Alternative Path, along with the Route Length and Estimated Time.

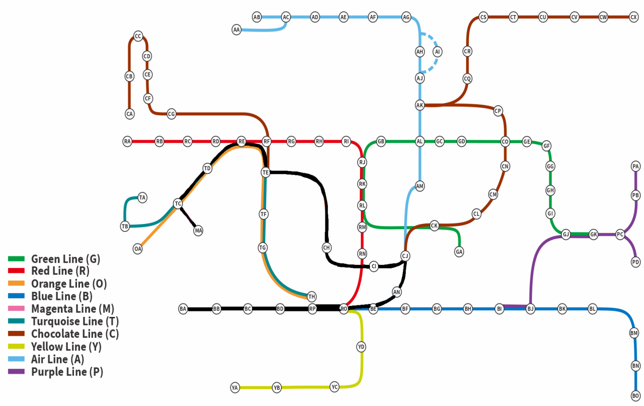
The user is first introduced to the following interface, which includes a welcome message and a list of stations.

```
Welcome to Hong Kong's Mass Transit Railway Guidance System.

We will help you find the ideal route to reach your destination.

---------------------------------------------------------------------------------------------

-=[ LIST OF STATIONS ]=-

 [1] Lok Ma Chau           [2] Lo Wu              [3] Sheung Shui        [4] Fanling
 [5] Tai Wo                [6] Tai Po Market      [7] University         [8] City One
 [9] Shek Mun             [10] Tai Shui Hang     [11] Heng On           [12] Ma On Shan
[13] Wu Kai Sha           [14] Fo Tan            [15] Racecourse        [16] Sha Tin Wai
[17] Sha Tin              [18] Che Kung Temple   [19] Tin Shui Wai      [20] Long Ping
[21] Yuen Long            [22] Kam Sheung Road   [23] Tsuen Wan West    [24] Tai Wai
[25] Hin Keng             [26] Tsuen Wan         [27] Tai Wo Hau        [28] Kwai Hing
[29] Kwai Fong            [30] Lai King          [31] Mei Foo           [32] Lai Chi Kok
[33] Cheung Sha Wan       [34] Sham Shui Po      [35] Shek Kip Mei      [36] Kowloon Tong
[37] Lok Fu               [38] Wong Tai Sin      [39] Diamond Hill      [40] Choi Hung
[41] Kowloon Bay          [42] Tsing Yi          [43] AsiaWorld-Expo    [44] Sunny Bay
[45] Airport              [46] Disneyland Resort [47] Tung Chung        [48] Nam Cheong
[49] Olympic              [50] Kowloon           [51] Prince Edward     [52] Mong Kok
[53] Yau Ma Tei           [54] Jordan            [55] Tsim Sha Tsui     [56] Austin
[57] Mong Kok East        [58] Kai Tak           [59] Sung Wong Toi     [60] To Kwa Wan
[61] Ho Man Tin           [62] Ngau Tau Kok      [63] Kwun Tong         [64] Lam Tin
[65] Po Lam               [66] Hang Hau          [67] Yau Tong          [68] Tiu Keng Leng
[69] Tseung Kwan O        [70] LOHAS Park        [71] Whampoa           [72] Hung Hom
[73] East Tsim Sha Tsui   [74] Hong Kong         [75] Kennedy Town      [76] HKU
[77] Sai Ying Pun         [78] Sheung Wan        [79] Central           [80] Admiralty
[93] Shau Kei Wan         [94] Hang Fa Chuen     [95] Chai Wan          [96] Exhibition Centre
[97] Siu Hong             [98] Tuen Mun

---------------------------------------------------------------------------------------------
```

The user is then required to input the starting and destination stations. This will output the main path or shortest route of the trip and the other possible routes that might be available. It will also output the length of the route and also the amount of minutes it takes to reach the final station. In this case, we would be starting at Kennedy Town (75) station and it will stop at Disneyland Resort (46) station.

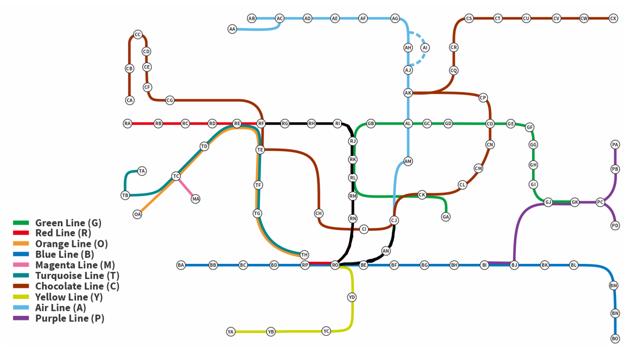
```
---------------------------------------------------------------------------------------------

Choose Starting Station: 75

Choose Destination Station: 46

---------------------------------------------------------------------------------------------

[ MAIN PATH ]

Kennedy Town > HKU > Sai Ying Pun > Sheung Wan > Central > Admiralty > Exhibition Centre > Hung Hom > East Tsim Sha Tsui > Austin > Nam Cheon
g > Lai King > Tsing Yi > Sunny Bay > Disneyland Resort

ROUTE LENGTH : 79 | TIME = 107 minutes

[ ALTERNATIVE PATH ]

No other routes found.

Would you like to continue?

[1] Yes
[2] No

Insert Number: 
```

The following graph is the visual representation of the route that the program had determined. The black line represents the main path from Kennedy Town (75) station to Disneyland Resort (46) station.

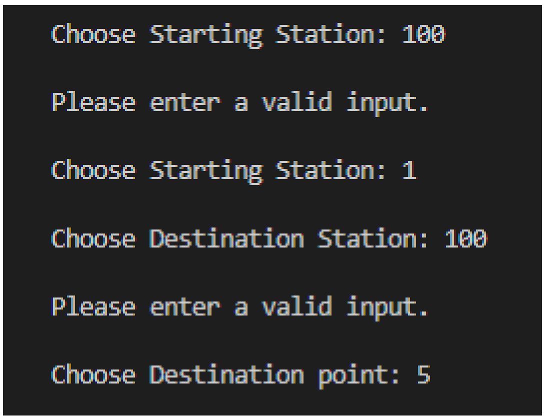


If there are any alternative routes available, it will also be printed out. Otherwise it will state that there are no other routes available. This time, we will be starting at Mei Foo (31) station and will stop at Hung Hom (72). This will have three alternate routes shown as the following:

The following graph is the visual representation of the route that the program had determined. The black line represents the main path from Mei Foo (31) station to Hung Hom (72) station.



If the user enters an invalid input (under 1 or over 98), the following message will be displayed and the user is required to enter a valid input.



V. Conclusion

We can conclude that this program is a "Smart and Simple" navigation system for finding the best route to reach our chosen destination.

It came as a solution for the tourists to find the shortest path, alternative path, route length, and estimated time using Dijkstra's algorithm for their vacation.

VI. Acknowledgment

We very much appreciate the guidance and advice that Mr. Wahyono, S. Kom., Ph.D. gave us throughout the Data Structures and Algorithms course in this second semester. Thank you for all the knowledge that you have given to us.

We are also aware of the program's drawbacks that it isn't comparable to other navigation system apps out there that use complex programming languages and many design keys. But throughout the completion of the project, we are proud of what we can create and of how it could be used and applied by the users.

VII. References

[1] Chan, B., 2021. In praise of Hong Kong's MTR – still one of the best in the world. [online] South China Morning Post. Available at: <https://www.scmp.com/comment/opinion/article/3159842/praise-hong-kongs-mtr-still-one-best-world?module=perpetual_scroll_0&pgtype=article&campaign=3159842> [Accessed 20 May 2022].

[2] Tsang, D., 2022. 'End of an era' for Hong Kong's cross-border through-train services. [online] South China Morning Post. Available at: <https://www.scmp.com/news/hong-kong/hong-kong-economy/article/3175918/end-era-hong-kong-mtrs-cross-border-through-train> [Accessed 21 May 2022].

[3] Hanwen, Z., 2015. Human Resource Management in MTR. [online] Article.sapub.org. Available at: <http://article.sapub.org/10.5923.j.hrmr.20150504.03.html> [Accessed 20 May 2022].

[4] Schwandl, R., 2004. UrbanRail.Net > Asia > HONG KONG Mass Transit Railway. [online] Urbanrail.net. Available at: <https://www.urbanrail.net/as/cn/hong/hong-kong.htm> [Accessed 21 May 2022].

[5] Vivien, L., 2021. Shortest path and 2nd shortest path using dijkstra | La Vivien Post. [online] La Vivien Post. Available at: <https://www.lavivienpost.com/shortest-path-and-2nd-shortest-path-using-dijkstra-code/> [Accessed 26 May 2022]