Programming Assignment #2 - Paris Metro Network

CSI2110

Alperen Akin 300290090

Discussion

i) Description of your chosen data structures, including the data structure used for your graph

Adjacency List: (creating a graph)
An ArrayList of lists (List<List<Edge>>) was used to represent the metro network as a graph. That graph in the java program was represented as an adjacency list. Each vertex(station) in the graph(metro network) corresponded to a list of edges (subway lines) connected to it. This allowed for quick access to neighbouring vertices and their corresponding edge details. The arrayList data structure was chosen because it provided quick access to the neighbours of a vertex, making algorithms like DFS more efficient for part 1.

HashMap: (reading the metro.txt file)
Hashmap was used to map each station number read from metro.txt to an integer index. To read and create the graph efficiently, the metro station numbers need to be mapped to integer indices that correspond to the vertices within the graph's data structure. Hashmap was chosen because it could store mappings between string and integers (even though it was not required. An integer was used to index the stations) vertexMap helps in converting these string-based identifiers to their corresponding integer indices, making it easier to work with the graph data structure while maintaining readability through named vertices.(operations requiring station indices were indexed by querying the Hashmap object)

Priority Queue(Min-Heap): (Djikstra shortest path)
The Priority Queue was used for Dijkstra's algorithm to maintain the vertices with their corresponding distances. A min-heap is used to continually select the vertex with the smallest distance from the current vertex, allowing for efficient selection of the next vertex to explore during Dijkstra's algorithm. This allowed the algorithm to explore vertices with the shortest known distances first.

ii) High level description of algorithms used to answer task #2

A `PriorityQueue`was used to store stations based on stations having the smallest time from the source station. An array distance[] was then used to store the shortest known time from the source to each vertex.The distance of the source vertex was set to “0” and all other vertices were set to infinity (In the program -->Integer.MAX_VALUE).

The source station is then enqueued into the priority queue. Then, while the priority queue is not empty: the station with the minimum corresponding weight(time) is extracted from the priority queue. All neighbouring stations of the extracted station is then explored.

For each neighbouring station: the corresponding time from the source through the current station to the neighbouring station is calculated. If this new time is shorter than the current known time to that station:the time of the neighbouring station is updated in the `time[]` array. Lastly, the priority queue is updated with the new shorter distance for that station.

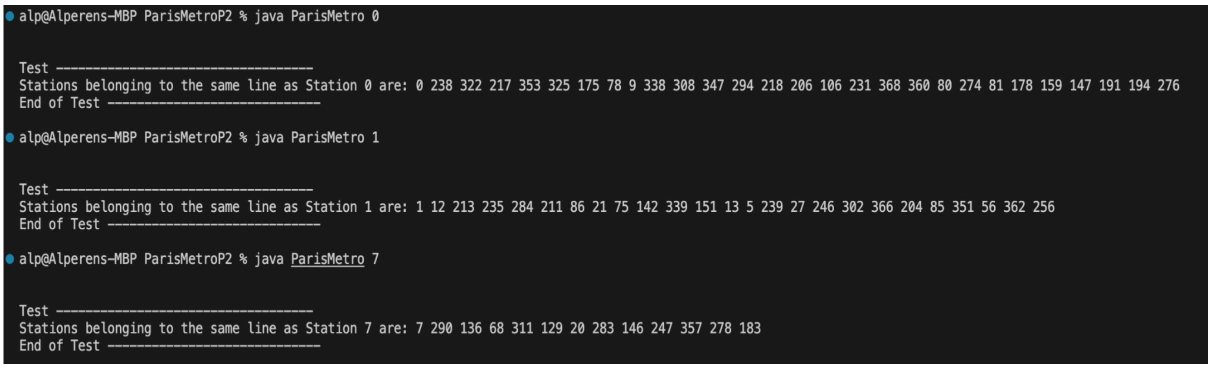
<u>Obtaining the Path</u>
Once the destination station's shortest time has been updated, the shortest path is then constructed by backtracking through the `distance[]` array from the destination to the source, considering stations with decreasing distances until the source station is reached. Finally, the sequence of stations is then obtained to create a “path” which is stored in a linkedList and calculated through the method “printPath”.

Used Data Structures:

- Priority Queue: to maintain stations ordered by their corresponding weights(time)
- Array(map): to map and store the shortest known time from the source to all other stations.

iii) Example Outputs

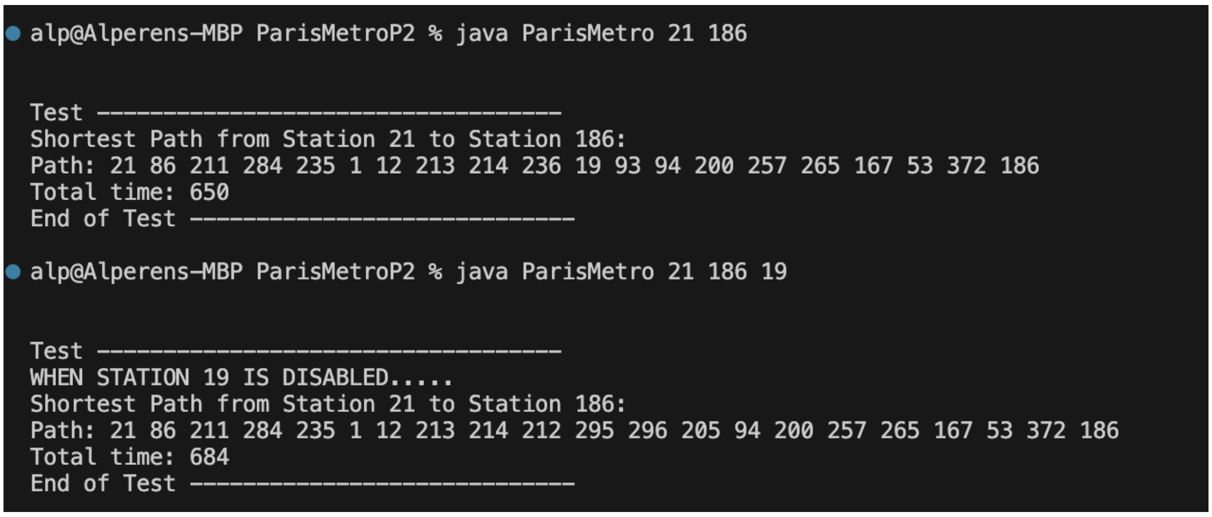
PART 1



```
● alp@Alperens-MBP ParisMetroP2 % java ParisMetro 0


  Test ----------------------------------
  Stations belonging to the same line as Station 0 are: 0 238 322 217 353 325 175 78 9 338 308 347 294 218 206 106 231 368 360 80 274 81 178 159 147 191 194 276
  End of Test -----------------------------

● alp@Alperens-MBP ParisMetroP2 % java ParisMetro 1


  Test ----------------------------------
  Stations belonging to the same line as Station 1 are: 1 12 213 235 284 211 86 21 75 142 339 151 13 5 239 27 246 302 366 204 85 351 56 362 256
  End of Test -----------------------------

● alp@Alperens-MBP ParisMetroP2 % java ParisMetro 7


  Test ----------------------------------
  Stations belonging to the same line as Station 7 are: 7 290 136 68 311 129 20 283 146 247 357 278 183
  End of Test -----------------------------
```

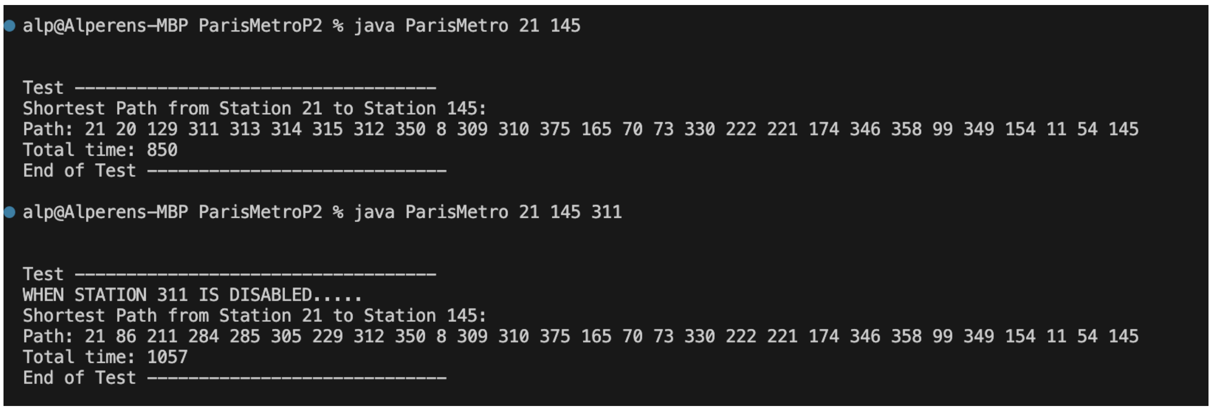
PART 2 AND PART 3 TEST PAIRS

1) N1 = 21 N2 = 186 (N3 = 19)



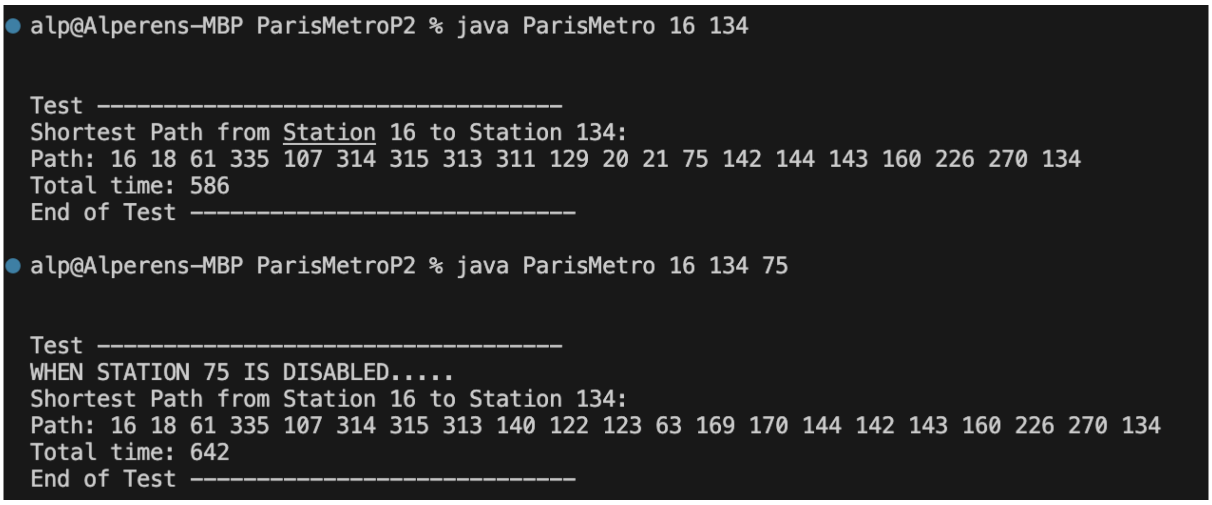
```
● alp@Alperens-MBP ParisMetroP2 % java ParisMetro 21 186


  Test -----------------------------------
  Shortest Path from Station 21 to Station 186:
  Path: 21 86 211 284 235 1 12 213 214 236 19 93 94 200 257 265 167 53 372 186
  Total time: 650
  End of Test -----------------------------

● alp@Alperens-MBP ParisMetroP2 % java ParisMetro 21 186 19


  Test -----------------------------------
  WHEN STATION 19 IS DISABLED.....
  Shortest Path from Station 21 to Station 186:
  Path: 21 86 211 284 235 1 12 213 214 212 295 296 205 94 200 257 265 167 53 372 186
  Total time: 684
  End of Test -----------------------------
```

2) N1 = 21 N2 = 145 (N3 = 311)



```
● alp@Alperens-MBP ParisMetroP2 % java ParisMetro 21 145


  Test -----------------------------------
  Shortest Path from Station 21 to Station 145:
  Path: 21 20 129 311 313 314 315 312 350 8 309 310 375 165 70 73 330 222 221 174 346 358 99 349 154 11 54 145
  Total time: 850
  End of Test -----------------------------

● alp@Alperens-MBP ParisMetroP2 % java ParisMetro 21 145 311


  Test -----------------------------------
  WHEN STATION 311 IS DISABLED.....
  Shortest Path from Station 21 to Station 145:
  Path: 21 86 211 284 285 305 229 312 350 8 309 310 375 165 70 73 330 222 221 174 346 358 99 349 154 11 54 145
  Total time: 1057
  End of Test -----------------------------
```

3) N1 = 16 N2 = 134 (N3 = 75)



```
● alp@Alperens-MBP ParisMetroP2 % java ParisMetro 16 134


  Test -----------------------------------
  Shortest Path from Station 16 to Station 134:
  Path: 16 18 61 335 107 314 315 313 311 129 20 21 75 142 144 143 160 226 270 134
  Total time: 586
  End of Test -----------------------------

● alp@Alperens-MBP ParisMetroP2 % java ParisMetro 16 134 75


  Test -----------------------------------
  WHEN STATION 75 IS DISABLED.....
  Shortest Path from Station 16 to Station 134:
  Path: 16 18 61 335 107 314 315 313 140 122 123 63 169 170 144 142 143 160 226 270 134
  Total time: 642
  End of Test -----------------------------
```