Graph Lab

Header information:

- Author #1: Maged Armanios (armanm5@mcmaster.ca)
- Author #2: Jinal Sanjula Kasturiarachchi (kasturij@mcmaster.ca)
- Gitlab URL: https://gitlab.cas.mcmaster.ca/sfwreng-3xb3-fall-22/l1-graph-lab
- Avenue to Learn group name: Group 32

Week 1: Manipulating

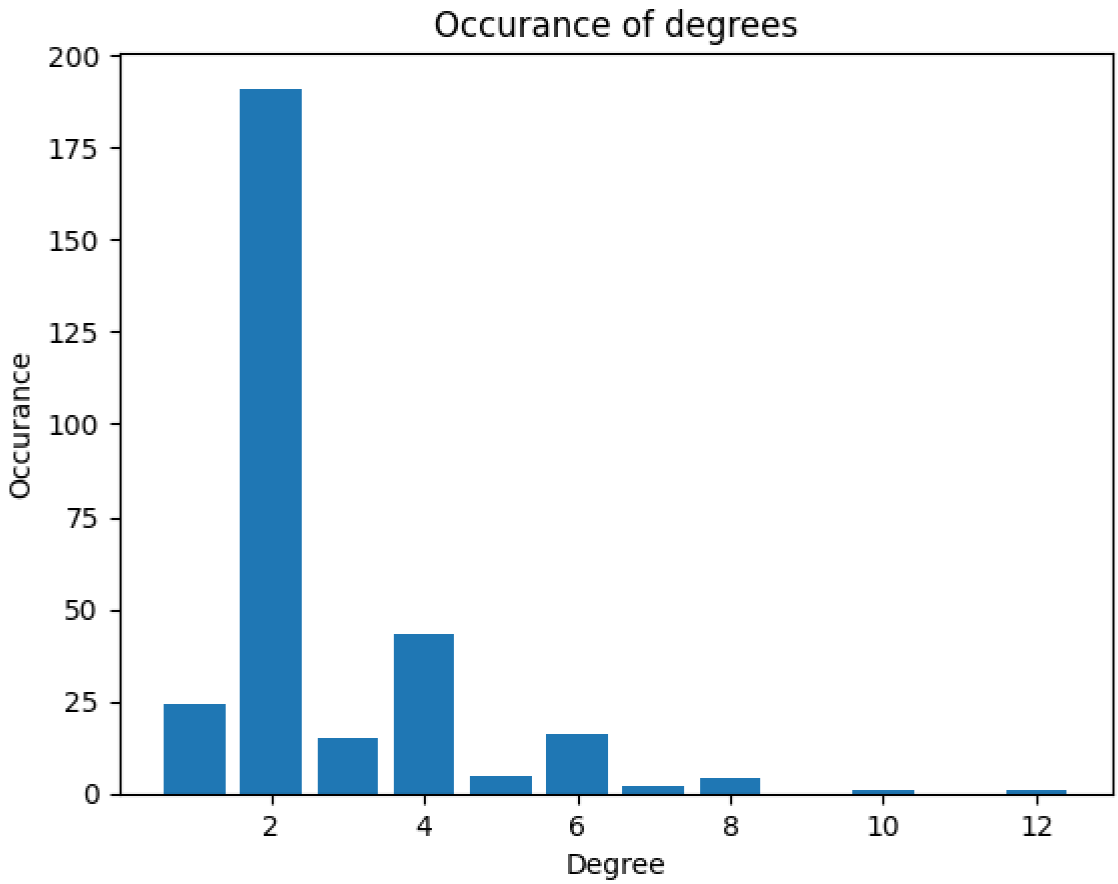
In week 1, we were tasked with designing an object oriented model for reading and storing data from CSV files. These files contained data regarding the London subway system. Specifically, one file contained information about the stations, one file contained information about the connections and another contained information about the lines.

In [1]:
```
import sys
from graph.Graph import Graph
from metrics_extractor.MetricsExtractor import MetricsExtractor
from shortest_path.ShortestPath import PathFactory
from graph_builder.GraphBuilder import GraphBuilder
from matplotlib import pyplot as plt
```

In [2]:
```
pathToStations = "_dataset/london.stations.csv"
pathToConnections = "_dataset/london.connections.csv"
g = GraphBuilder.build(pathToStations,pathToConnections)
```

The class GraphBuilder containts the method buildGraph() with two parameters pertaining to the paths of the stations and connections files. This approach to building the graph enables us to add more methods to the class in the case that the CSV file format changes. This makes the graph class indepedant from the graph

In [3]:
```
degrees = MetricsExtractor.compute_sum_of_degrees(g)
plt.bar(*zip(*degrees.items()))
plt.title("Occurance of degrees")
plt.xlabel("Degree")
plt.ylabel("Occurance")
plt.show()
```



Metrics extractor computes different metrics about the graph using various functions. The graph above above shows the occurance of each degree in the graph. It is clear that on average, most stations connect to only two other stations.

The class PathFactory contains multiple methods which all return an itinerary derived using different algorithms. PathFactory.a_star and PathFactory.djikstra return the shortest path as an intinerary given a graph, source, and destination

In [4]:
```
print("Find the shortest path from 11 to 283\n")
print("Computed with djikstra's algorithm")
pathA = PathFactory.dijkstra(g,11,283)
pathA.printPath()
print("\nComputed with a* algorithm")
pathB = PathFactory.a_star(g,11,283)
pathB.printPath()
```

```
Find the shortest path from 11 to 283

Computed with djikstra's algorithm
Go from 11 to 193 in 6.0 stops with line 1
Go from 193 to 283 in 3.0 stops using line 6

Computed with a* algorithm
Go from 11 to 193 in 6.0 stops with line 1
Go from 193 to 283 in 3.0 stops using line 6
```

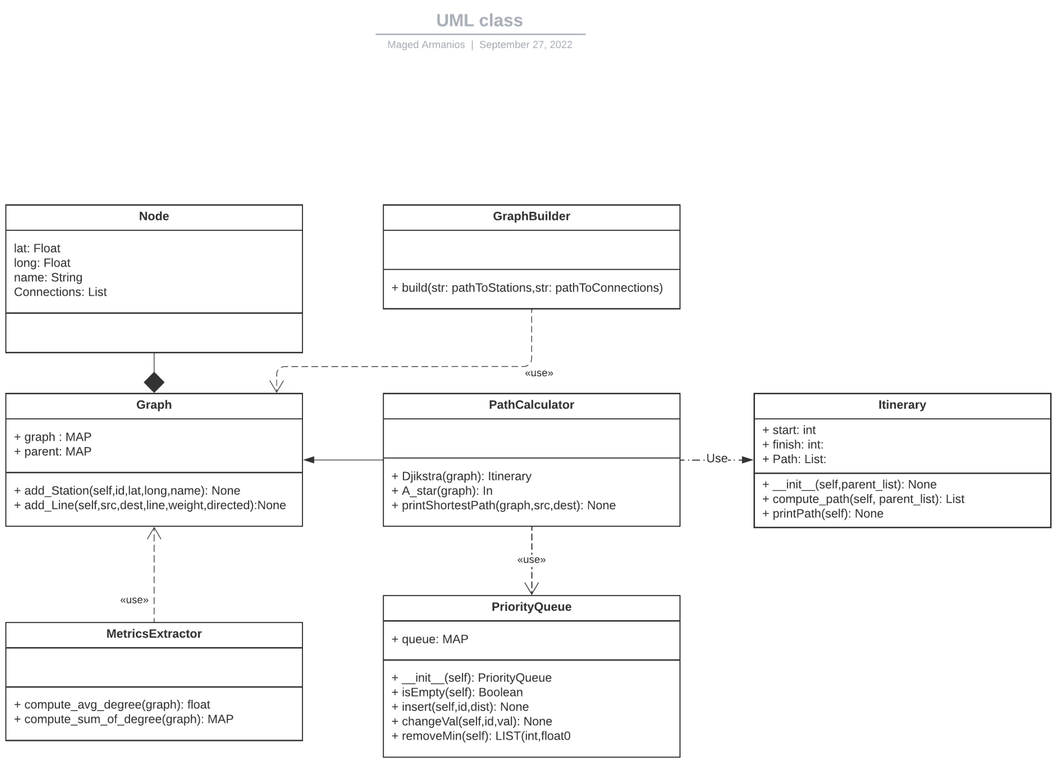
Going from station 11 (Baker Street) to 283 (Westbourne Park) can be done in a total of 9 stops and 2 lines

In [5]:
```
pathA.path
```

Out[5]:
```
[[11, 1.0, 1], [163, 2.0, 1], [82, 3.0, 1], [193, 1.0, 6], [218, 2.0, 6]]
```

The path variable stores every stop along they way, but printPath() only prints other stops when a line change is required. Lines are the third item in the arrays.

Design Choices



Explanations and Justifications

GraphBuilder

Seperating GraphBuilder from all other classes was key to ensuring the design is scalable. Now, the graph class is completely independant from the format of the CSV file and if the CSV file is to change we can simply add another method to GraphBuilder that can read it. In summary GraphBuilder enables Graph to be independant of the CSV files and can be adjusted in an open-closed manner.

Node

The Node class is what represents the subway stations. It containts all information with the exception of some columns about the zone and rail as they weren't used. Representing each station as an object does use more memory, but enables us to encapsulate the information easily and makes the data at each station easily accessable.

Graph

Graph uses nodes to represent the entire subway system in one class. It possess a graph attribute which contains a dictionary (Map) of every station. We used a map instead of an array so that we can accomodate any stationID. Graph also possess the ability to add stations (Nodes) and connections (vertices)

MetricsCalculator

MetricsCalculator seperates the functionality of pulling information from the graph from Graph. Each desired function is a public function that can be called as long as a graph is passed in. Additionally, it enables us to create more functions to compute metrics in an open-closed principle without overloading the GraphClass.

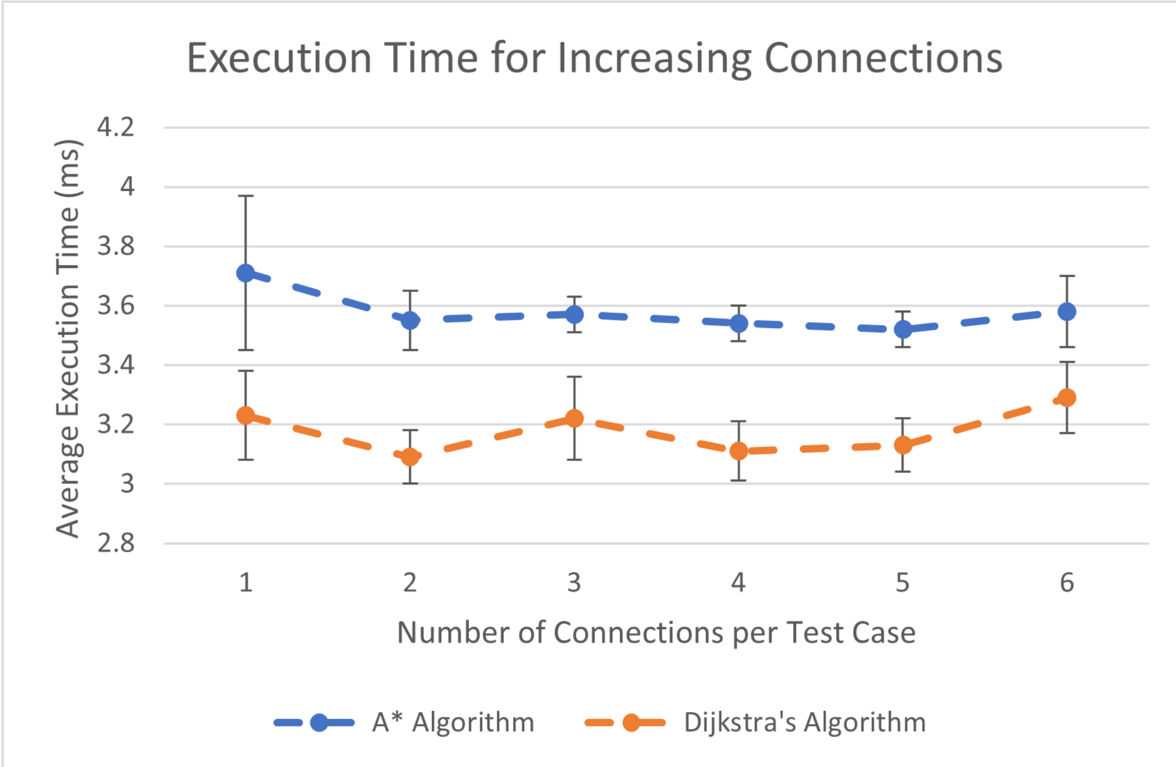
PathFactory

PatchFactory is a class that contains all algorithms used for generating itineraries. As of now it contains a method for computing path using Djikstras and the A* algorithm. If any algorithms were to be added to the library, they can be appended to PathFactory without modifying any other algorithms.

Itinerary

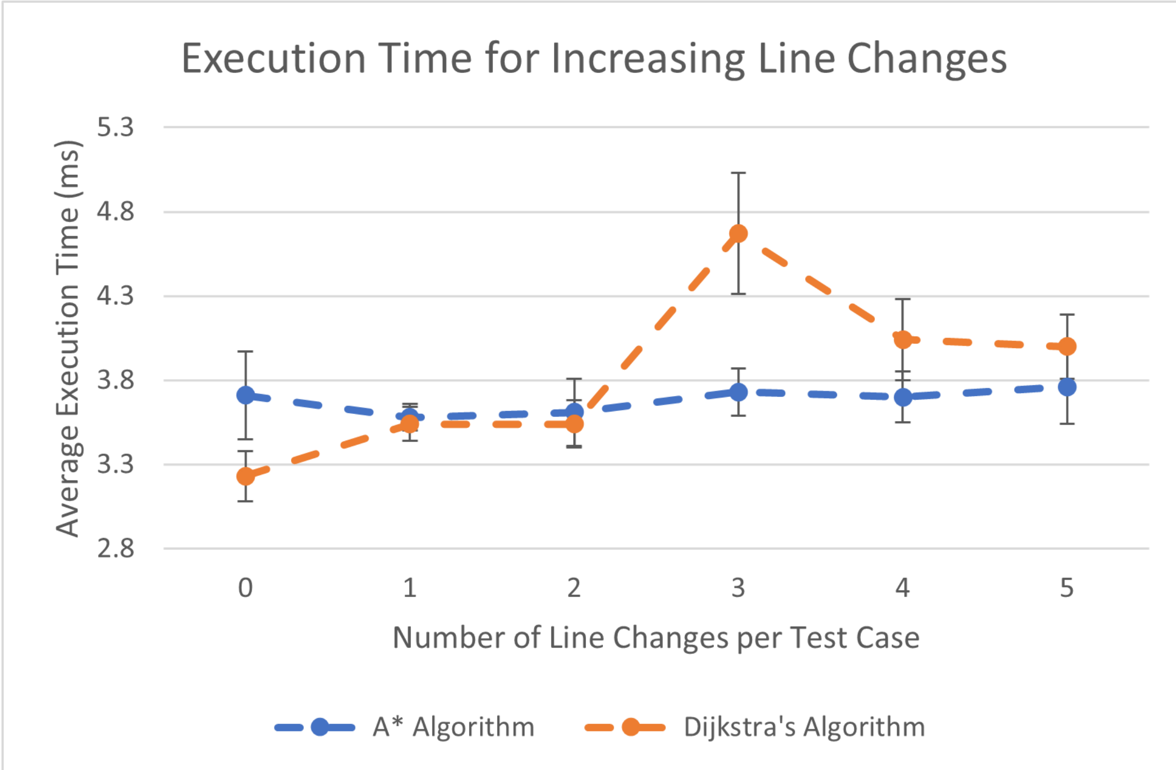
The itinerary class holds all relevant information about a path from one place to another. It is generated by the different methods of PathFactory. It can display the path from one station to another and removes redudant information like every stop on the line a user would take.

Benchmarking the Solution



Reflecting on the data

The number of connections refers to how many stops the rail line passes from point A to point B. As we increase the number of stops without switching lines, there seems to be no drastic change in the execution time of either algorithm. We can extrapolate these results to infer that regardless of path distance, the algorithms perform a similar, if not the same amount of operations. The data makes sense with Djikstras as it solves the single source shortest path problem and for A* since it is a heuristic approach to Djikstra's. In a practical application, it would be possible to precompute all single source shortest paths from each node for a total running time of O(N^2). If our priority queue was implemented using a minheap, this could be reduced to O(N*Log(N)). To conclude, for a sparse graph, it would be feasible to precompute all single source shortest paths for long-term use in some applications. In the case of something like a subway system, it would be safe to assume the graph doesn't change except with the occasional maintenance of certain lines.



Reflecting on the data

From the above graphs, line changes per test case indicates the number lines changes on the shortest path(i.e. going from line 2 to line 4), in order to get from the starting station to the final destination. It is interesting to note that this second graph definitely contradicts the first one, highlighting that there are applications for which the A* algorithm would be beneficial to use over Dijkstra's. If it is known that the incoming dataset/station map includes many lines with the majority of those stations being exclusively accessible on a small amount of those lines, then it seems that the A* search algorithm would provide a more efficient solution to the shortest path problem. We suspect the reason behind this result is due to the fact that A* computes the shortest path using the heuristic approach (physical distance between stations) prior to comparing connections, and as such, determines the shortest path with line changes faster than Dijkstra's - which always compares connections and looks for minimal line changes first.

```
itinerary1 = PathFactory.dijkstra(g, 28, 277)
print()
itinerary2 = PathFactory.a_star(g, 28, 277)
print()
itinerary1 = PathFactory.dijkstra(g, 11, 143)
print()
itinerary2 = PathFactory.a_star(g, 11, 143)
```

```
Nodes visited 812
Edges crossed 302

Nodes visited 812
Edges crossed 302

Nodes visited 812
Edges crossed 302

Nodes visited 812
Edges crossed 302
```

Reflecting on the data

The number of nodes and edges traversed during every run of either algorithm is always the same. 812 edges is double the amount of connections present and 302 is the number of stations (vertices) in the graph. This means the algorithm crosses every edge once and visits every station twice. With this data we can conclude that regardless of the solution size (path size), there will always be an overhead cost proportional to (2E + V). Where E and V represent edges and nodes in the graph

Why we chose these test cases and KPIs.

Nodes visited, edges crossed, and execution time were the chosen KPIs for this week's problems. Nodes visited and edges crossed provide us insight on our solution in respect to the size of the graph (vertices and edges). Execution time allows us to understand how feasible it would be to consistently compute shortest paths using our algorithm and if in a business application, would we be able to satisfy some hypothetical demand of X shortest paths computed/second.

Division of Work

Jinal

Benchmarking, test suite, a* algorithm, graphing benchmarking results

Maged

Graph & itinerary generation, djikstras, class diagram.

Together

Reflection and analysis

Week 2: Planning

Subway Patrol

The subway patrol problem requires that given a subset of stations, compute the shortest cycle that contains all the stations. The application of this would be to provide subway patrols with the shortest route to complete their patrol. This problem is similar to the travelling sales man problem, to solve this our team used a brute force approach.

In [6]:
```
from subway_patrol.SubwayPatrol import SubwayPatrol

pathToStations = "_dataset/london.stations.csv"
pathToConnections = "_dataset/london.connections.csv"
g = GraphBuilder.build(pathToStations, pathToConnections)

i1 = SubwayPatrol.travellingSalesmanProblem(g, [15, 32])
i2 = SubwayPatrol.travellingSalesmanProblem(g, [15, 32, 36])
# Each array in path correspondes to [src, line, edgeweight] so while it appears that
# vertex is missing, it is represented by "take line 78 line 4 for 4 stops"
print(i1.path)
print(i1.finish)
print()
print(i2.path)
print(i2.finish)
```

```
[[15, 4.0, 4], [78, 2.0, 4], [270, 2.0, 4], [200, 2.0, 4], [289, 3.0, 7], [247, 2.0,
13], [204, 2.0, 13], [32, 2.0, 13], [204, 2.0, 13], [247, 3.0, 7], [289, 2.0, 4], [20
0, 2.0, 4], [270, 2.0, 4], [78, 4.0, 4]]
15

[[15, 4.0, 4], [78, 2.0, 4], [270, 2.0, 4], [200, 2.0, 4], [289, 3.0, 7], [247, 2.0,
13], [204, 2.0, 13], [32, 2.0, 13], [204, 2.0, 13], [247, 3.0, 7], [289, 2.0, 4], [3
6, 2.0, 4], [289, 2.0, 4], [200, 2.0, 4], [270, 2.0, 4], [78, 4.0, 4]]
15
```

Why this approach?

The travelling salesman problem is a proven NP-Hard problem and implementing a more complex solution to reduce the running time would in our opinion, be more trouble than it's worth. As of now, our solution can handle a subgraph of size 7 (potentially 8 on stronger systems). Since this algorithm is to be used by the officers of the subway system and not users trying to find a path in a short amount of time, it's acceptable for this algorithm to take a considerable amount of time as the officers patrol will most likely not change throughout the day and will be already decided before they start working.

Running time analysis

Our approach to the Subway Patrol problem can be broken down into the following steps:

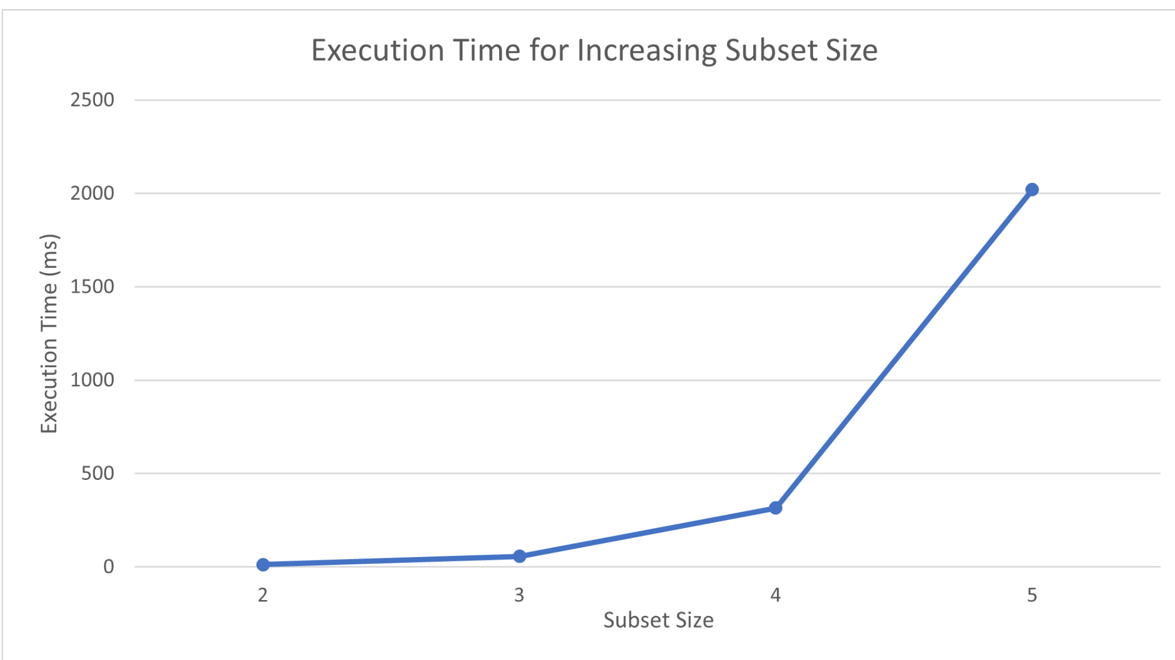
```
1) Compute all permutations of the of the subset of stations. The
subset of stations are just an array of integers and
   computing all permutations takes O(N\*N!) time, where N is the
size of the subset.

2) For each permutation, compute the shortest path along all the
paths. EX: if the permutation was (5,101,38), compute
   the shortest path from station 5 to 101, then 101 to 38. The
itineraries are then combined using a combign itinerary
   method present in the class. Therefore, each permutation takes (N-
1)\*(Running time of the shortest path)

3) For each itinerary present (should be N!), find one with the
shortest path. Each itinerary has a variable in it
    storing the total path length, so we just have to iterate over N!
itineraries and its finished.
```

We can approximate the running time to be proportional to step 2, as that is where all of the logic takes place. In total the approximate running time is O(N!*(N-1)*(E+V))

Benchmark results



Urbanism Planning

London's network is organized into different zones. This week's second problem required us to indentify how these zones are connected to each other and to identify "transportation islands".

Transportation islands can be best described as a set of stations that can be reached without switching zones.

Finding the transportation island's

To find transportation islands we begin by finding out which stations are in which zone using the metricsExtractor

In [7]:
```
pathToStations = "_dataset/london.stations.csv"
pathToConnections = "_dataset/london.connections.csv"
g = GraphBuilder.build(pathToStations,pathToConnections)
zone_list = MetricsExtractor.return_zone_list(g)
```

In [8]:
```
# Print all stations in each zone
for zone in zone_list:
    print("Zone {} has stations:".format(zone))
    print(zone_list[zone])
    print()
```

```
Zone 3 has stations:
{1, 257, 258, 260, 8, 265, 266, 12, 270, 274, 19, 20, 152, 153, 26, 286, 160, 289, 3
4, 36, 297, 298, 299, 172, 300, 43, 303, 176, 52, 181, 182, 183, 56, 59, 63, 64, 65,
194, 69, 71, 72, 73, 200, 203, 204, 77, 79, 80, 86, 217, 219, 224, 97, 100, 231, 234,
106, 108, 111, 112, 113, 246, 247, 124}

Zone 1 has stations:
{2, 3, 259, 133, 262, 7, 263, 11, 13, 14, 145, 18, 146, 148, 149, 273, 151, 277, 25,
279, 28, 29, 156, 157, 285, 161, 162, 163, 166, 167, 44, 48, 49, 60, 188, 192, 193, 1
97, 198, 208, 82, 83, 212, 87, 89, 90, 92, 223, 99, 229, 102, 104, 233, 250, 107, 23
6, 248, 122, 126, 255}

Zone 2 has stations:
{4, 8, 10, 17, 22, 23, 24, 27, 32, 33, 35, 36, 39, 40, 41, 42, 47, 54, 55, 56, 61, 6
4, 69, 70, 74, 76, 79, 80, 84, 86, 94, 95, 96, 101, 106, 110, 111, 120, 123, 127, 12
8, 135, 136, 137, 138, 139, 142, 143, 147, 150, 152, 155, 159, 160, 164, 170, 171, 17
4, 175, 181, 183, 186, 191, 195, 201, 204, 205, 206, 209, 216, 218, 225, 226, 227, 22
8, 238, 242, 244, 245, 249, 253, 254, 264, 265, 272, 276, 278, 283, 284, 287, 290, 29
2, 293, 295, 296, 297}

Zone 4 has stations:
{130, 131, 5, 261, 9, 140, 141, 269, 15, 144, 275, 281, 154, 282, 30, 31, 288, 165, 3
8, 169, 173, 301, 302, 178, 185, 58, 190, 196, 202, 78, 207, 211, 213, 93, 98, 230, 2
32, 105, 237, 240, 241, 119, 251, 252}

Zone 5 has stations:
{132, 16, 21, 291, 37, 45, 177, 51, 184, 57, 187, 66, 67, 199, 75, 81, 210, 215, 91,
221, 103, 235, 109, 239, 114, 115, 243, 121}

Zone 6 has stations:
{256, 129, 134, 267, 268, 271, 158, 294, 168, 179, 180, 68, 85, 88, 220, 222, 116, 11
7, 118, 125}

Zone 7 has stations:
{168, 62, 214}

Zone 10 has stations:
{50, 6}

Zone 9 has stations:
{46}

Zone 8 has stations:
{280, 53}
```

After returning the zone list, we compute the connected components for each zone

In [9]:
```
from connected_components.ConnectedComponents import connectedComponents
```

In [10]:
```
g.cc = connectedComponents.returnCC(g,zone_list)
```

In [11]:
```
# Print all transportation islands per zone
for zone in g.cc:
    print("Zone {} has stations:".format(zone))
```

```
    print(g.cc[zone])
    print()
```

```
Zone 3 has stations:
[[1, 52, 73, 234, 265, 265, 108, 108, 234, 176, 176, 73, 72, 182, 182, 194, 194, 72,
286, 286, 181, 181, 112, 112, 52], [257, 12, 258, 258, 59, 59, 12, 56, 56], [260, 26,
224, 224, 26, 274, 274], [8, 124, 124, 77, 77], [266, 160, 303, 303, 160], [270, 200,
200, 289, 289, 36, 43, 247, 247, 153, 204, 204, 153, 43, 79, 219, 183, 183, 219, 63,
63, 203, 203, 217, 217, 20, 20, 65, 65, 97, 97, 19, 19, 79, 36], [152, 86, 86, 69, 6
9, 106, 106, 64, 64], [34, 100, 100, 111, 111], [297, 71, 71, 172, 172], [298, 113, 1
13, 246, 246], [299, 300, 300, 231, 231, 80, 80]]

Zone 1 has stations:
[[2, 156, 263, 263, 166, 3, 3, 166, 44, 44, 161, 161, 25, 25, 255, 255, 87, 87, 49, 2
79, 285, 285, 248, 107, 107, 28, 133, 197, 192, 273, 273, 229, 198, 198, 229, 236, 23
6, 99, 146, 146, 99, 122, 122, 192, 212, 259, 277, 277, 89, 102, 102, 89, 145, 145, 9
0, 92, 7, 223, 223, 126, 126, 48, 60, 60, 151, 151, 48, 250, 250, 13, 13, 157, 167, 1
67, 14, 188, 188, 14, 157, 233, 29, 29, 233, 7, 92, 90, 104, 104, 11, 11, 163, 83, 8
3, 193, 193, 82, 18, 18, 82, 163, 259, 212, 197, 133, 28, 162, 162, 149, 149, 208, 20
8, 248, 279, 148, 148, 49, 156], [262]]

Zone 2 has stations:
[[4, 70, 201, 201, 27, 284, 292, 292, 42, 42, 120, 41, 183, 183, 41, 216, 253, 23, 2
3, 253, 174, 175, 175, 174, 216, 276, 276, 225, 225, 155, 295, 295, 244, 228, 228, 24
4, 164, 164, 24, 33, 33, 36, 36, 24, 155, 120, 238, 238, 61, 61, 171, 171, 135, 135,
64, 64, 106, 106, 69, 69, 86, 86, 152, 152, 284, 27, 79, 79, 70, 32, 32, 204, 204],
[8, 264, 264, 139, 139, 40, 40, 47, 170, 170, 47, 22, 22, 111, 111], [10, 95, 128, 12
8, 39, 39, 95, 160, 123, 123, 160], [17, 110, 293, 74, 74, 138, 287, 287, 96, 96, 19
5, 195, 205, 205, 80, 80, 138, 293, 110, 209, 101, 265, 265, 242, 242, 101, 227, 227,
150, 150, 147, 147, 283, 283, 218, 218, 209], [35, 245, 245, 55, 191, 272, 272, 191,
136, 136, 84, 84, 55, 54, 54, 56, 56], [76, 181, 296, 296, 226, 226, 127, 127, 186, 1
86, 181], [94, 254, 290, 290, 142, 142, 297, 297, 254, 249, 249], [137, 206, 206, 14
3, 143, 159, 159, 278, 278]]

Zone 4 has stations:
[[130, 131, 131, 190, 190, 30, 30], [5, 252, 252, 251, 251], [261, 302, 302, 288, 28
8, 93, 93, 165, 165], [9, 31, 232, 232, 31], [140, 237, 237, 185, 185, 281, 281], [14
1, 213, 213], [269, 15, 15, 78, 78], [144, 207, 282, 282, 202, 202, 178, 178, 207],
[275, 154, 211, 211, 98, 98, 173, 173, 154, 230, 230, 241, 241, 301, 301], [38, 58, 5
8, 119, 119], [169, 240, 240], [196, 105, 105]]

Zone 5 has stations:
[[132], [16, 91, 91, 109, 109, 103, 103, 51, 51, 215, 215], [21, 67, 67, 66, 66], [29
1, 115, 210, 210, 75, 235, 235, 75, 115, 184, 184, 199, 199], [37], [45, 243, 243],
[177, 239, 239, 221, 221], [57, 187, 187], [81], [114], [121]]

Zone 6 has stations:
[[256, 68, 88, 88, 68, 158, 158], [129, 85, 268, 268, 267, 267, 85], [134, 125, 220,
220, 222, 222, 125, 271, 271], [294], [168, 179, 179, 180, 180], [116, 117, 118, 118,
117]]

Zone 7 has stations:
[[168, 62, 214, 214, 62]]

Zone 10 has stations:
[[50], [6]]

Zone 9 has stations:
[[46]]
```

```
Zone 8 has stations:
[[280], [53]]
```

Relating Zones

In [12]:
```
crossing_edges = connectedComponents.generateCrossingEdgesBetweenZones(g)
for zone in crossing_edges:
    print("Zone {} can reach...:".format(zone))
    for crossing_edge in crossing_edges[zone]:
        print("    Zone {} by using {} to get to {}".format(crossing_edge[2],crossing_
    print()
```

```
Zone 3 can reach...:
    Zone 2 by using 1 to get to 265
    Zone 2 by using 8 to get to 264
    Zone 2 by using 265 to get to 242
    Zone 2 by using 265 to get to 110
    Zone 2 by using 266 to get to 160
    Zone 2 by using 12 to get to 56
    Zone 4 by using 270 to get to 78
    Zone 4 by using 270 to get to 78
    Zone 2 by using 152 to get to 86
    Zone 4 by using 153 to get to 154
    Zone 2 by using 286 to get to 181
    Zone 2 by using 160 to get to 95
    Zone 2 by using 289 to get to 36
    Zone 2 by using 289 to get to 36
    Zone 4 by using 34 to get to 119
    Zone 2 by using 36 to get to 33
    Zone 2 by using 36 to get to 33
    Zone 2 by using 297 to get to 142
    Zone 2 by using 298 to get to 137
    Zone 4 by using 172 to get to 282
    Zone 2 by using 43 to get to 79
    Zone 2 by using 43 to get to 183
    Zone 4 by using 303 to get to 31
    Zone 4 by using 176 to get to 30
    Zone 2 by using 52 to get to 265
    Zone 2 by using 181 to get to 76
    Zone 2 by using 183 to get to 42
    Zone 2 by using 56 to get to 54
    Zone 4 by using 59 to get to 240
    Zone 2 by using 64 to get to 106
    Zone 2 by using 64 to get to 135
    Zone 4 by using 194 to get to 5
    Zone 2 by using 69 to get to 86
    Zone 2 by using 69 to get to 106
    Zone 2 by using 71 to get to 297
    Zone 2 by using 204 to get to 32
    Zone 4 by using 77 to get to 93
    Zone 2 by using 79 to get to 27
    Zone 2 by using 80 to get to 205
    Zone 2 by using 86 to get to 69
    Zone 2 by using 86 to get to 152
    Zone 2 by using 224 to get to 95
    Zone 2 by using 100 to get to 111
    Zone 2 by using 231 to get to 80
    Zone 2 by using 106 to get to 64
    Zone 2 by using 106 to get to 69
    Zone 4 by using 108 to get to 141
    Zone 2 by using 108 to get to 265
    Zone 2 by using 111 to get to 22
    Zone 2 by using 112 to get to 181
    Zone 4 by using 112 to get to 196
    Zone 4 by using 246 to get to 281
    Zone 2 by using 247 to get to 164
    Zone 2 by using 247 to get to 204
    Zone 2 by using 124 to get to 8
```

```
Zone 1 can reach...:
    Zone 2 by using 3 to get to 295
    Zone 2 by using 3 to get to 295
    Zone 2 by using 262 to get to 225
    Zone 2 by using 11 to get to 249
    Zone 2 by using 11 to get to 94
    Zone 2 by using 13 to get to 225
    Zone 2 by using 145 to get to 39
    Zone 2 by using 145 to get to 123
    Zone 2 by using 18 to get to 186
    Zone 2 by using 18 to get to 186
    Zone 2 by using 148 to get to 84
    Zone 2 by using 279 to get to 136
    Zone 2 by using 29 to get to 84
    Zone 2 by using 156 to get to 24
    Zone 2 by using 157 to get to 23
    Zone 2 by using 193 to get to 278
    Zone 2 by using 193 to get to 218
    Zone 2 by using 198 to get to 272
    Zone 2 by using 208 to get to 186
    Zone 2 by using 89 to get to 40
    Zone 2 by using 89 to get to 170
    Zone 2 by using 99 to get to 74
    Zone 2 by using 99 to get to 74
    Zone 2 by using 122 to get to 186
    Zone 2 by using 122 to get to 74
    Zone 2 by using 122 to get to 186

Zone 2 can reach...:
    Zone 3 by using 8 to get to 124
    Zone 3 by using 22 to get to 111
    Zone 1 by using 23 to get to 157
    Zone 1 by using 24 to get to 156
    Zone 3 by using 27 to get to 79
    Zone 3 by using 32 to get to 204
    Zone 3 by using 33 to get to 36
    Zone 3 by using 33 to get to 36
    Zone 3 by using 36 to get to 289
    Zone 3 by using 36 to get to 289
    Zone 1 by using 39 to get to 145
    Zone 1 by using 40 to get to 89
    Zone 3 by using 42 to get to 183
    Zone 3 by using 54 to get to 56
    Zone 3 by using 56 to get to 12
    Zone 3 by using 64 to get to 106
    Zone 3 by using 69 to get to 86
    Zone 3 by using 69 to get to 106
    Zone 1 by using 74 to get to 99
    Zone 1 by using 74 to get to 122
    Zone 1 by using 74 to get to 99
    Zone 3 by using 76 to get to 181
    Zone 3 by using 79 to get to 43
    Zone 3 by using 80 to get to 231
    Zone 1 by using 84 to get to 148
    Zone 1 by using 84 to get to 29
    Zone 3 by using 86 to get to 69
    Zone 3 by using 86 to get to 152
```

```
        Zone 1 by using 94 to get to 11
        Zone 4 by using 94 to get to 282
        Zone 3 by using 95 to get to 160
        Zone 3 by using 95 to get to 224
        Zone 3 by using 106 to get to 64
        Zone 3 by using 106 to get to 69
        Zone 3 by using 110 to get to 265
        Zone 3 by using 111 to get to 100
        Zone 1 by using 123 to get to 145
        Zone 3 by using 135 to get to 64
        Zone 1 by using 136 to get to 279
        Zone 3 by using 137 to get to 298
        Zone 3 by using 142 to get to 297
        Zone 3 by using 152 to get to 86
        Zone 3 by using 160 to get to 266
        Zone 3 by using 164 to get to 247
        Zone 1 by using 170 to get to 89
        Zone 3 by using 181 to get to 112
        Zone 3 by using 181 to get to 286
        Zone 3 by using 183 to get to 43
        Zone 1 by using 186 to get to 208
        Zone 1 by using 186 to get to 18
        Zone 1 by using 186 to get to 122
        Zone 1 by using 186 to get to 18
        Zone 1 by using 186 to get to 122
        Zone 3 by using 204 to get to 247
        Zone 3 by using 205 to get to 80
        Zone 1 by using 218 to get to 193
        Zone 1 by using 225 to get to 13
        Zone 1 by using 225 to get to 262
        Zone 3 by using 242 to get to 265
        Zone 1 by using 249 to get to 11
        Zone 3 by using 264 to get to 8
        Zone 3 by using 265 to get to 52
        Zone 3 by using 265 to get to 108
        Zone 3 by using 265 to get to 1
        Zone 1 by using 272 to get to 198
        Zone 1 by using 278 to get to 193
        Zone 1 by using 295 to get to 3
        Zone 1 by using 295 to get to 3
        Zone 3 by using 297 to get to 71

    Zone 4 can reach...:
        Zone 5 by using 130 to get to 132
        Zone 3 by using 5 to get to 194
        Zone 5 by using 261 to get to 121
        Zone 5 by using 140 to get to 114
        Zone 3 by using 141 to get to 108
        Zone 5 by using 269 to get to 21
        Zone 3 by using 281 to get to 246
        Zone 3 by using 154 to get to 153
        Zone 3 by using 282 to get to 172
        Zone 2 by using 282 to get to 94
        Zone 3 by using 30 to get to 176
        Zone 3 by using 31 to get to 303
        Zone 5 by using 38 to get to 81
        Zone 5 by using 173 to get to 16
```

```
        Zone 5 by using 301 to get to 37
        Zone 5 by using 301 to get to 215
        Zone 5 by using 178 to get to 115
        Zone 3 by using 196 to get to 112
        Zone 3 by using 78 to get to 270
        Zone 3 by using 78 to get to 270
        Zone 5 by using 207 to get to 45
        Zone 3 by using 93 to get to 77
        Zone 5 by using 232 to get to 187
        Zone 5 by using 105 to get to 177
        Zone 3 by using 240 to get to 59
        Zone 3 by using 119 to get to 34
        Zone 5 by using 251 to get to 235

    Zone 5 can reach...:
        Zone 6 by using 132 to get to 116
        Zone 4 by using 132 to get to 130
        Zone 4 by using 16 to get to 173
        Zone 4 by using 21 to get to 269
        Zone 6 by using 37 to get to 158
        Zone 4 by using 37 to get to 301
        Zone 4 by using 45 to get to 207
        Zone 4 by using 177 to get to 105
        Zone 4 by using 187 to get to 232
        Zone 6 by using 66 to get to 85
        Zone 6 by using 199 to get to 180
        Zone 6 by using 75 to get to 222
        Zone 6 by using 75 to get to 222
        Zone 4 by using 81 to get to 38
        Zone 4 by using 215 to get to 301
        Zone 6 by using 221 to get to 294
        Zone 4 by using 235 to get to 251
        Zone 4 by using 114 to get to 140
        Zone 4 by using 115 to get to 178
        Zone 4 by using 121 to get to 261

    Zone 6 can reach...:
        Zone 5 by using 158 to get to 37
        Zone 5 by using 294 to get to 221
        Zone 7 by using 168 to get to 62
        Zone 7 by using 168 to get to 214
        Zone 7 by using 179 to get to 168
        Zone 5 by using 180 to get to 199
        Zone 5 by using 85 to get to 66
        Zone 5 by using 222 to get to 75
        Zone 5 by using 222 to get to 75
        Zone 5 by using 116 to get to 132

    Zone 7 can reach...:
        Zone 6 by using 168 to get to 179
        Zone 6 by using 62 to get to 168
        Zone 8 by using 62 to get to 280
        Zone 8 by using 214 to get to 53
        Zone 6 by using 214 to get to 168

    Zone 10 can reach...:
        Zone 9 by using 50 to get to 46
```

```
    Zone 9 by using 6 to get to 46

Zone 9 can reach...:
    Zone 10 by using 46 to get to 6
    Zone 10 by using 46 to get to 50
    Zone 8 by using 46 to get to 53

Zone 8 can reach...:
    Zone 7 by using 280 to get to 62
    Zone 9 by using 53 to get to 46
    Zone 7 by using 53 to get to 214
```

Relating Islands

We relate transportation islands by creating a "component graph" that is, a graph in which every node in it is a connected component. The names of the connected components are arbitrary and the nodes only relevant informations are their adjacency lists and zones. To find out which stations are in which components, we have a dictionary which at any station ID, returns a list of components containing that station. Note most stations are only in one component, but some are in two

In [13]:

```
componentsHolding,graph_of_components = GraphBuilder.buildComponentGraph(g,g.cc)
for component in graph_of_components.graph:
    print("Component {} connects to:".format(component))
    for edge in graph_of_components.graph[component].connections:
        print("Component {} using station {} to station {}".format(edge[0],edge[1],edg
    print()
```

```
Component 0 connects to:
Component 16 using station 1 to station 265
Component 16 using station 52 to station 265
Component 16 using station 265 to station 242
Component 16 using station 265 to station 110
Component 26 using station 108 to station 141
Component 16 using station 108 to station 265
Component 21 using station 176 to station 30
Component 22 using station 194 to station 5
Component 18 using station 286 to station 181
Component 18 using station 181 to station 76
Component 18 using station 112 to station 181
Component 32 using station 112 to station 196

Component 1 connects to:
Component 17 using station 12 to station 56
Component 31 using station 59 to station 240
Component 17 using station 56 to station 54

Component 2 connects to:
Component 15 using station 224 to station 95

Component 3 connects to:
Component 14 using station 8 to station 264
Component 14 using station 124 to station 8
Component 23 using station 77 to station 93

Component 4 connects to:
Component 15 using station 266 to station 160
Component 15 using station 160 to station 95
Component 24 using station 303 to station 31

Component 5 connects to:
Component 27 using station 270 to station 78
Component 13 using station 289 to station 36
Component 13 using station 36 to station 33
Component 13 using station 43 to station 79
Component 13 using station 43 to station 183
Component 13 using station 247 to station 164
Component 13 using station 247 to station 204
Component 29 using station 153 to station 154
Component 13 using station 204 to station 32
Component 13 using station 79 to station 27
Component 13 using station 183 to station 42

Component 6 connects to:
Component 13 using station 152 to station 86
Component 13 using station 86 to station 69
Component 13 using station 86 to station 152
Component 13 using station 69 to station 86
Component 13 using station 69 to station 106
Component 13 using station 106 to station 64
Component 13 using station 106 to station 69
Component 13 using station 64 to station 106
Component 13 using station 64 to station 135

Component 7 connects to:
```

```
Component 30 using station 34 to station 119
Component 14 using station 100 to station 111
Component 14 using station 111 to station 22

Component 8 connects to:
Component 19 using station 297 to station 142
Component 19 using station 71 to station 297
Component 28 using station 172 to station 282

Component 9 connects to:
Component 20 using station 298 to station 137
Component 25 using station 246 to station 281

Component 10 connects to:
Component 16 using station 231 to station 80
Component 16 using station 80 to station 205

Component 11 connects to:
Component 13 using station 156 to station 24
Component 13 using station 3 to station 295
Component 17 using station 279 to station 136
Component 17 using station 198 to station 272
Component 16 using station 99 to station 74
Component 18 using station 122 to station 186
Component 16 using station 122 to station 74
Component 14 using station 89 to station 40
Component 14 using station 89 to station 170
Component 15 using station 145 to station 39
Component 15 using station 145 to station 123
Component 13 using station 13 to station 225
Component 13 using station 157 to station 23
Component 17 using station 29 to station 84
Component 19 using station 11 to station 249
Component 19 using station 11 to station 94
Component 20 using station 193 to station 278
Component 16 using station 193 to station 218
Component 18 using station 18 to station 186
Component 18 using station 208 to station 186
Component 17 using station 148 to station 84

Component 12 connects to:
Component 13 using station 262 to station 225

Component 13 connects to:
Component 5 using station 27 to station 79
Component 5 using station 42 to station 183
Component 5 using station 183 to station 43
Component 11 using station 23 to station 157
Component 11 using station 225 to station 13
Component 12 using station 225 to station 262
Component 11 using station 295 to station 3
Component 5 using station 164 to station 247
Component 11 using station 24 to station 156
Component 5 using station 33 to station 36
Component 5 using station 36 to station 289
Component 6 using station 135 to station 64
Component 6 using station 64 to station 106
```

```
Component 6 using station 106 to station 64
Component 6 using station 106 to station 69
Component 6 using station 69 to station 86
Component 6 using station 69 to station 106
Component 6 using station 86 to station 69
Component 6 using station 86 to station 152
Component 6 using station 152 to station 86
Component 5 using station 79 to station 43
Component 5 using station 32 to station 204
Component 5 using station 204 to station 247

Component 14 connects to:
Component 3 using station 8 to station 124
Component 3 using station 264 to station 8
Component 11 using station 40 to station 89
Component 11 using station 170 to station 89
Component 7 using station 22 to station 111
Component 7 using station 111 to station 100

Component 15 connects to:
Component 4 using station 95 to station 160
Component 2 using station 95 to station 224
Component 11 using station 39 to station 145
Component 4 using station 160 to station 266
Component 11 using station 123 to station 145

Component 16 connects to:
Component 0 using station 110 to station 265
Component 11 using station 74 to station 99
Component 11 using station 74 to station 122
Component 10 using station 205 to station 80
Component 10 using station 80 to station 231
Component 0 using station 265 to station 52
Component 0 using station 265 to station 108
Component 0 using station 265 to station 1
Component 0 using station 242 to station 265
Component 11 using station 218 to station 193

Component 17 connects to:
Component 11 using station 272 to station 198
Component 11 using station 136 to station 279
Component 11 using station 84 to station 148
Component 11 using station 84 to station 29
Component 1 using station 54 to station 56
Component 1 using station 56 to station 12

Component 18 connects to:
Component 0 using station 76 to station 181
Component 0 using station 181 to station 112
Component 0 using station 181 to station 286
Component 11 using station 186 to station 208
Component 11 using station 186 to station 18
Component 11 using station 186 to station 122

Component 19 connects to:
Component 11 using station 94 to station 11
Component 28 using station 94 to station 282
```

```
Component 8 using station 142 to station 297
Component 8 using station 297 to station 71
Component 11 using station 249 to station 11

Component 20 connects to:
Component 9 using station 137 to station 298
Component 11 using station 278 to station 193

Component 21 connects to:
Component 33 using station 130 to station 132
Component 0 using station 30 to station 176

Component 22 connects to:
Component 0 using station 5 to station 194
Component 36 using station 251 to station 235

Component 23 connects to:
Component 43 using station 261 to station 121
Component 3 using station 93 to station 77

Component 24 connects to:
Component 4 using station 31 to station 303
Component 40 using station 232 to station 187

Component 25 connects to:
Component 42 using station 140 to station 114
Component 9 using station 281 to station 246

Component 26 connects to:
Component 0 using station 141 to station 108

Component 27 connects to:
Component 35 using station 269 to station 21
Component 5 using station 78 to station 270

Component 28 connects to:
Component 38 using station 207 to station 45
Component 8 using station 282 to station 172
Component 19 using station 282 to station 94
Component 36 using station 178 to station 115

Component 29 connects to:
Component 5 using station 154 to station 153
Component 34 using station 173 to station 16
Component 37 using station 301 to station 37
Component 34 using station 301 to station 215

Component 30 connects to:
Component 41 using station 38 to station 81
Component 7 using station 119 to station 34

Component 31 connects to:
Component 1 using station 240 to station 59

Component 32 connects to:
Component 0 using station 196 to station 112
Component 39 using station 105 to station 177
```

```
Component 33 connects to:
Component 49 using station 132 to station 116
Component 21 using station 132 to station 130

Component 34 connects to:
Component 29 using station 16 to station 173
Component 29 using station 215 to station 301

Component 35 connects to:
Component 27 using station 21 to station 269
Component 45 using station 66 to station 85

Component 36 connects to:
Component 28 using station 115 to station 178
Component 46 using station 75 to station 222
Component 22 using station 235 to station 251
Component 48 using station 199 to station 180

Component 37 connects to:
Component 44 using station 37 to station 158
Component 29 using station 37 to station 301

Component 38 connects to:
Component 28 using station 45 to station 207

Component 39 connects to:
Component 32 using station 177 to station 105
Component 47 using station 221 to station 294

Component 40 connects to:
Component 24 using station 187 to station 232

Component 41 connects to:
Component 30 using station 81 to station 38

Component 42 connects to:
Component 25 using station 114 to station 140

Component 43 connects to:
Component 23 using station 121 to station 261

Component 44 connects to:
Component 37 using station 158 to station 37

Component 45 connects to:
Component 35 using station 85 to station 66

Component 46 connects to:
Component 36 using station 222 to station 75

Component 47 connects to:
Component 39 using station 294 to station 221

Component 48 connects to:
Component 50 using station 168 to station 62
Component 50 using station 168 to station 214
```

```
Component 50 using station 179 to station 168
Component 36 using station 180 to station 199

Component 49 connects to:
Component 33 using station 116 to station 132

Component 50 connects to:
Component 48 using station 168 to station 179
Component 48 using station 62 to station 168
Component 54 using station 62 to station 280
Component 55 using station 214 to station 53
Component 48 using station 214 to station 168

Component 51 connects to:
Component 53 using station 50 to station 46

Component 52 connects to:
Component 53 using station 6 to station 46

Component 53 connects to:
Component 52 using station 46 to station 6
Component 51 using station 46 to station 50
Component 55 using station 46 to station 53

Component 54 connects to:
Component 50 using station 280 to station 62

Component 55 connects to:
Component 53 using station 53 to station 46
Component 50 using station 53 to station 214
```

Note the adjacency matrix is modified from the original format of [dest, line, edge_weight] to [dest_component, starting_station, ending_station]

In [14]:

```
# To check which zones you must pass while travelling, simply find the shortest path a
# the zones of each vertex in the path

pathToStations = "_dataset/london.stations.csv"
pathToConnections = "_dataset/london.connections.csv"
g = GraphBuilder.build(pathToStations,pathToConnections)
itin1 = PathFactory.dijkstra(g,11,283)

zones = set()
for i in itin1.path:
    zones.add(g.graph[i[0]].zone)
zones.add(g.graph[itin1.finish].zone)
print(zones)
```

```
{1.0, 2.0}
```

Runtime analysis of connected components

Finding connected components

Finding connected components looks at each node in each zone and runs DFSUtil on it. DFSUtil utilizes a stack to look through all neighbours of every node, giving it a running time proportional to O(V+E). Therefore the total running time would be O(V(E+N))

Finding crossing edges between zones

Finding crossings edges between zones has the algorithm look through every nodes components and check its edges and compare if its in the same zone as its neighbours. This has a runtime proportional to O(V+E)

Generating components graph

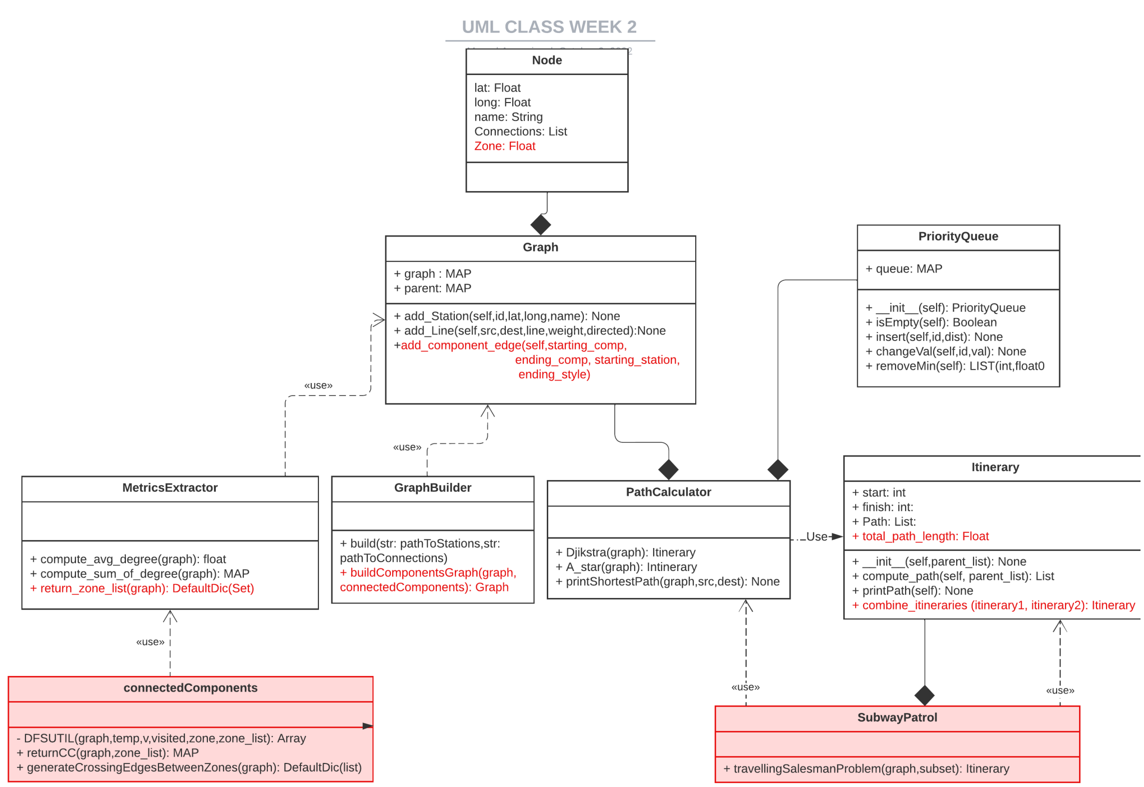
Again this looks at every node and checks if the component its neighbours belongs to is different than its own. This problem compares zones a total of (N+E) times.

Benchmark Results

```
%run benchmark_connectedComponents
```

```
.....................
Finding Connected Components: Mean +- std dev: 257 us +- 25 us
.....................
Finding Crossing Edges between zones: Mean +- std dev: 572 us +- 42 us
.....................
Generating Components graph: Mean +- std dev: 624 us +- 37 us
```

Updated UML Diagram



Two new classes added were connectedComponents and SubwayPatrol. Connected components returns Arrays, Maps, and dictionaries depending on the method called. It Uses metrics extractor to compute which nodes are in which zone and has no other dependancies. Sunway patrol has an itinerary which it consistently updates and uses PathFactory to find the shortest paths for each permutation.

MetricsExtractor, Graph, GraphBuilder, Node, and Itinerary all got some additional features.

Metrics extractor can now return which nodes are in which zones. This was implemented to help solve the connected components problem.

GraphBuilder can now build a components graph. This is done using a method which when given a normal graph and a set of connected components, creates a graph where each component is a vertex. Graph

Graph now has a new method called add_component_edge. This method is used for components graphs. Component graphs have different representations of edges than traditional graphs, so a new method was created to accomodate that.

Itinerary now has an additional variable called total_path_length and a new method, combignItineraries. total_path_length stores the total path length of the current instance and is used by SubwayPatrol to quickly calculate the shortest path. CombignItineraries is used by SubwayPatrol to create a cycle including all stations in the subset. (See our detailed explanation on SubwayPatrol).

Discussion of SOLID design principles

SOLID design is an acronym incorperating 5 design principles. These principles are Single-responsibility, open-closed priniciple, Liskovs substitution principles, interface segregation, and dependancy inversion.

The tworinciples we are most concerned with are single-responsibility and open-closed. Single-responsibility states a class should only have one responsibility, in our case all our classes meet this criteria. This can be easily seen in classes like graphBuilder and PathFactory which build graphs and itineraries respectively.

Open-closed principle states that a module can be open for extension and closed for modification. Between weeks 1 and 2, we added various features to multiple of our classes and even added new classes themselves. This demonstrates our code follows the open-closed principle since without modifying old cold, we extended the features of existing modules. Some new features we added included the generation of components graphs and the ability to merge itineraries.

Week 3 Bonus

Random Graph Generator

In [15]:
```
randomG = GraphBuilder.buildRandomGraph(11,5,uniform=True)
for i in randomG.graph:
    print(len(randomG.graph[i].connections))
```

```
5
5
5
5
5
10
5
5
5
5
5
```

In [ ]:
```
randomG = GraphBuilder.buildRandomGraph(11,5,uniform=False)
for i in randomG.graph:
    print(len(randomG.graph[i].connections))
```

Self Reflection - Jinal

Backward

In terms of applying knowledge learned in the past (specifically regarding algorithms), I have not had much experience with similar work. However, the content that was discussed and eventually applied related to topics learned in previous software engineering courses such as 2C03 (Algorithms & Data Structures), and 2AA4 (Introduction to Software Development). This is not surprising as the name of this course is "Binding Theory to Practice" so it's nice to see the connection between the theory discussed in courses prior, to the application of that learning in this course.

Inward

Overall, I feel grateful to have gone through the process of creating this work, as this is the first time I've had the task of providing a programming solution to an open-ended problem, but I will say that there are aspects of the final product that are undesirable and could have been improved. Before I get into that though, I would like to say that graphing the benchmarking results for the first week (a* vs. dijkstra) was something that I enjoyed doing, and am proud of the result since it depicted the differences between the two algorithms with different inputs. In terms of dislikes for the project, I would say the main one would have to be the efficiency with some of the implementations - specifically for subway patrol. Although I understand that this is an np-hard problem, the solution that we came up with seems quite trivial. As discussed previously in the report, we felt like the work that we had to put in to improve it would outweigh the overall gains in performance. However, if we did have more time, I believe it would have been interesting to have been able to try and engineer a more efficient solution.

Outward

One thing that I want people to particularly notice about my work is the compatibility with SOLID principles. I think the time that was taken to ensure that all of the modules were separate in their folders and files (single responsibility) and further ensure that each module only accessed the features that were required from the others (dependency inversion). It was a time-consuming process to ensure that these principles were followed, and I believe that it should be showcased and apparent when looking at the code from an outward perspective.

Forward

If I had the chance to do this project over again, I would spend more time in the beginning (during week one) focusing on laying the groundwork for the coding environment, ensuring that it could be easily extendable. I think this is something that we had difficulty with during the end of week 1, and resulted in more work, and having to refactor the code to meet the SOLID criteria. In addition, I think there was a lot of focus on the actual code itself, and it was mentioned in one of the lectures that although the code is important, the experience of engineering a solution was the goal of this lab and this course in general. So, I would try and focus less on the code and more on the steps that we took along the way to come to that solution.

Self Reflection - Maged

Backward

The most similar work I've done would be finding and computing the running times of Djikstra's algorithm and minimum heaps in SFWRENG 2C03, the data structures and algorithms course. I feel 2C03 was the prerequisite to this course and without it, I would have struggled a lot more. For reading CSV files, traveling salesman, and connected components, I was familiar with the problems from some Leetcode questions, but nothing more than knowing they existed.

For the analysis section of the analysis, SFWRENG 2AA4 software design principles where we learned about solid design principles except for Liskov's.

I have had no prior experience with benchmarking. In hindsight, this isn't that good because as software engineers, we need to be able to gather data on our solutions to see if we've made efficient solutions.

Inward

I feel that this work was very beneficial to me. I haven't had a lot of experience implementing algorithms and coding, so to be given some problems to code out was fun. My favorite parts were developing classes for Graphs, Itineraries, and Nodes because there was so much freedom as to how I could implement them. I disliked the benchmarking and testing as I didn't feel that there was a strong reference to test my code against. I wish that alongside the problems, sample inputs and outputs were provided to give us some direction while developing.

Outward

One thing I want people to notice about the work in this lab is our implementation of the traveling salesman problem. While it may not have been implemented very efficiently (possibly slower than standard brute force methods), I feel that the code is surprisingly compact and readable. The reason for this is we already had other classes and methods to rely on. We created a method to merge itineraries for this problem. It made the solution so easy to implement, and I would like people to notice that.

Forward

One thing I would change if we redid this project would be breaking down our code more. Initially, a lot of our code was in very few files. As the project got larger it became harder to figure out which file contained which piece of code. By the end of week 2, we separated our files and folders, as well as made our naming scheme more consistent, but if we did this from the start, we would have not only worked faster but not had to do it later. Aside from that though, I feel we did a great job throughout the lab. Our solutions are functional, and our code is well documented throughout the report and python files.