SCICOMP302 Algorithms and Data Structures

Project 3 - Objective 2 - Investigation of graphs based on a real model

This is all my own work. I have not knowingly allowed others to copy my work. This work has not been submitted for assessment in any other context.

Author:

Joanikij Chulev

Professor:

Dr. Tamas Vinko

December, 2023

Abstract

Graphs serve as fundamental data structures, capturing relationships between entities in various domains. This exploration delves into the realm of graph theory, focusing on Minimum Spanning Trees (MSTs) and their significance. The investigation centers on a geographical graph modeling towns in Zeeland, employing real-life distances as edge weights. The study extends to a comprehensive analysis of Prim's and Kruskal's algorithms, comparing their execution times and visualizing MSTs. Dijkstra's algorithm is applied for shortest path determination, showcasing its relevance in optimizing travel routes. Sensitivity analysis explores the resilience of MSTs and Dijkstra's algorithm to edge removal, providing insights into network adaptability. Results affirm the effectiveness of both algorithms and their practical applicability. Conclusions on the experiments were very firm, although the simplicity of the Zeeland graph was acknowledged.

Contents

1 **Introduction** 4
2 **Experiments** 5
2.1 Some Code Analysis 6
2.2 Shortest Path - Dijkstra 6
2.3 Sensitivity Analysis 7
3 **Results and Conclusions** 8
4 **Additional Testing and Future Work** 10

Experimental platform

Hardware technology platform:

OS Name: Microsoft Windows 11 Pro

Windows version: 22H2

System Type: x64-based PC

Processor: Intel(R) Core (TM) i7-1065G7 CPU @ 1.30GHz, 1498 Mhz, 4 Core(s), 8 Logical Proc.

Installed Physical Memory (RAM): 4.00 GB

GPU: Integrated Intel GPU (Intel Graphics)

Software technology platform:

Spyder version: 5.4.3 (conda)

Python version: 3.9.17 64-bit

Qt version: 5.15.2

PyQt5 version: 5.15.7

Operating System: Windows 11

1 Introduction

Graphs are fundamental data structures that model relationships between entities. In the realm of graph theory, a graph is defined as a collection of nodes (or vertices) and edges connecting pairs of nodes. These edges may carry additional information, such as weights, which represent the cost or distance associated with traversing from one node to another. Graphs find applications in various domains, ranging from social networks and transportation systems to computer networks and optimization problems. A particularly important concept in graph theory is that of a Minimum Spanning Tree (MST). A spanning tree of a graph is a subgraph that is a tree (an acyclic, connected graph) and spans all the nodes of the original graph. The "minimum" in Minimum Spanning Tree implies that the total weight of the tree is minimized. In practical terms, an MST represents the most efficient way to connect all the nodes in a graph while minimizing the total cost or distance.

Two well-known algorithms for finding a Minimum Spanning Tree are Prim's algorithm and Kruskal's algorithm.

Prim's algorithm is a greedy algorithm that starts with an arbitrary node and grows the MST one node at a time. At each step, the algorithm selects the edge with the smallest weight that connects a node in the MST to a node outside the MST. This process continues until all nodes are included in the MST. Prim's algorithm ensures that the selected edges form a connected, acyclic subgraph with the minimum possible total weight.

Kruskal's algorithm, another greedy approach, takes a different strategy. It begins with a forest (a collection of disjoint trees), each consisting of a single node. The algorithm then iteratively selects the smallest-weight edge from the remaining edges and adds it to the forest, as long as it doesn't create a cycle. This process continues until all nodes are part of a single tree. Kruskal's algorithm guarantees that the final result is a connected, acyclic subgraph with the minimum total weight.

Both Prim's and Kruskal's algorithms are widely used in various applications, including network design, circuit wiring, and resource optimization. The choice between the two often depends on factors such as the specific characteristics of the graph, the implementation details, and the desired performance criteria. By understanding these algorithms, one can effectively navigate the landscape of graph theory and apply these principles to solve real-world problems efficiently.

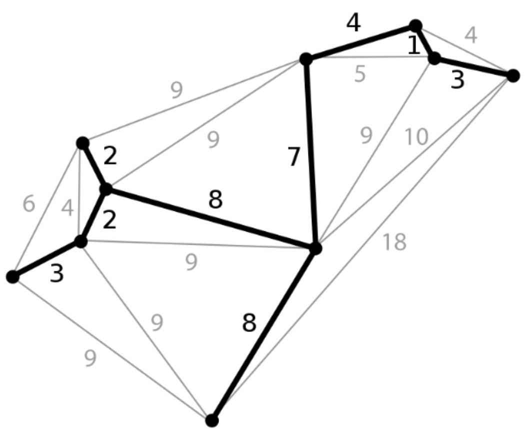


Figure 1: example of an MST (Minimum Spanning Tree)

2 Experiments

This experiment presents a detailed analysis of the geographical graph formulated for towns in the Zeeland region, with Middelburg as the central location. The graph is constructed based on real-life distances between towns, providing a comprehensive representation of the connectivity in the region. The methodology involves the identification of important locations, creation of nodes, establishment of connections, assignment of weights, and application of graph algorithms for analysis. The objective of this report is to create and analyze a geographical graph that accurately reflects the real-world distances and connections between towns in the Zeeland region. The graph serves as a valuable tool for understanding the spatial relationships and optimizing travel routes. Middelburg was selected as the central location, and other significant towns such as Vlissingen, Goes, Terneuzen, Zierikzee, Tholen, and additional towns were included. Each town was represented as a node in the graph, forming the fundamental structure of the geographical network. The distances between towns served as weights for the edges, reflecting the real-life cost or distance associated with traveling between nodes.

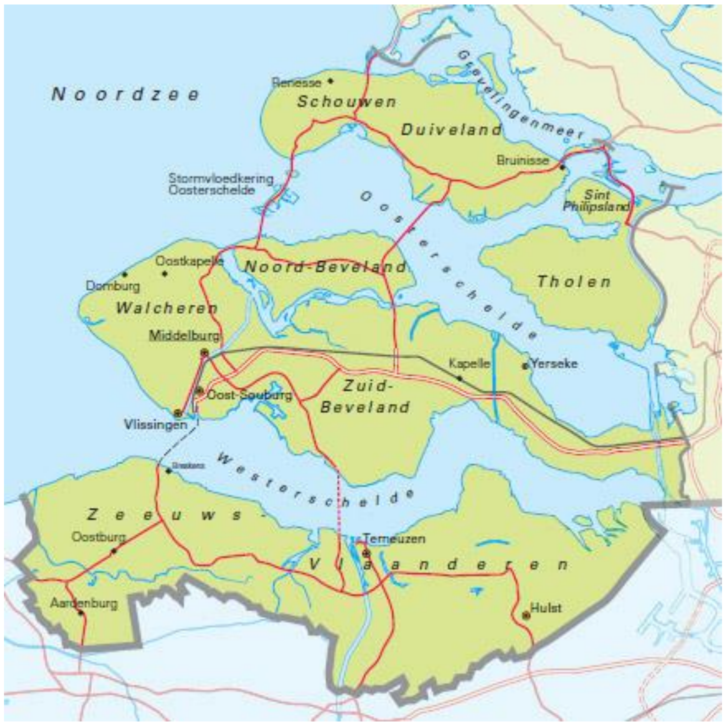


Figure 2: Image of Zeeland layout

Now, once we have formulated the graph representation, we can test implementations of MST algorithms on it. Prim's algorithm and Kruskal's algorithm were chosen because they are both well-established MST algorithms, and they have different approaches to constructing the minimum spanning tree. Comparing their execution times can provide insights into their relative efficiency for your specific graph. Thus, we can formulate the following hypotheses:

$H0$ - *There is no significant difference in the running time of Prim's algorithm and Kruskal's algorithm when applied to a simple connected graph representing Zeeland*

Ha - *There is a significant difference in the running time of Prim's algorithm and Kruskal's algorithm when applied to a simple connected graph representing Zeeland*

The independent variable is the Zeeland Graph we used, as well as the algorithm we use. Thus, the dependent variable would be the average running time elapsed. When running algorithms, a large number of times can provide a more accurate measure of their average performance. However, it's important to note that the actual running times may vary based on factors such as the size and structure of the graph. Running the algorithms 1000000 times will provide insight on timings efficiently enough.

2.1 Some Code Analysis

Please, consult the code files to see all code with in-depth explanations. We present the code of the main test file, although there are multiple files:

The provided Python code utilizes the NetworkX library to model and analyze a graph representing a network of towns with weighted connections. The first part of the code defines a graph using NetworkX, where towns are represented as nodes, and the distances between them are represented as weighted edges. The weights of the edges are based on the distances between towns.

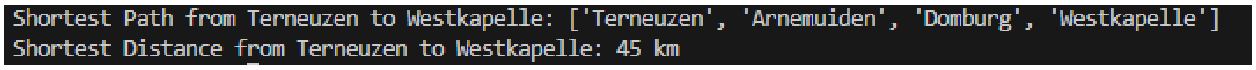
The second part of the code focuses on analyzing the graph using two minimum spanning tree algorithms: Prim's algorithm and Kruskal's algorithm.

The code runs both Prim's and Kruskal's algorithms multiple times (specified by num_iterations) and calculates the average execution time for each algorithm. The time module is used to measure the execution time of each algorithm.

Finally, the code prints the average execution times for both Prim's and Kruskal's algorithms over the specified number of iterations.

2.2 Shortest Path - Dijkstra

The shortest path between two locations in a graph represents the most efficient route with the minimal cumulative distance. In the context of the graph representing towns in Zeeland, finding the shortest path between two specific towns, such as the example of Terneuzen and Westkapelle in Figure 3 is crucial for optimizing travel between them. The implemented algorithm, Dijkstra's algorithm, systematically explores the graph, considering various paths and selecting the one with the lowest total distance. The highlighted path on the graph visualization showcases the optimal route, and the associated distance quantifies the efficiency of that journey. This approach is instrumental in various practical scenarios, such as transportation planning or network optimization, providing valuable insights into the most time and resource-efficient connections between towns in the region. We can see some results in Figure 3. Starting town is colored green, end town is colored red. Path is bolded in red lines.



```
Shortest Path from Terneuzen to Westkapelle: ['Terneuzen', 'Arnemuiden', 'Domburg', 'Westkapelle']
Shortest Distance from Terneuzen to Westkapelle: 45 km
```

Figure 3: Shortest Path information from Terneuzen to Westkapelle

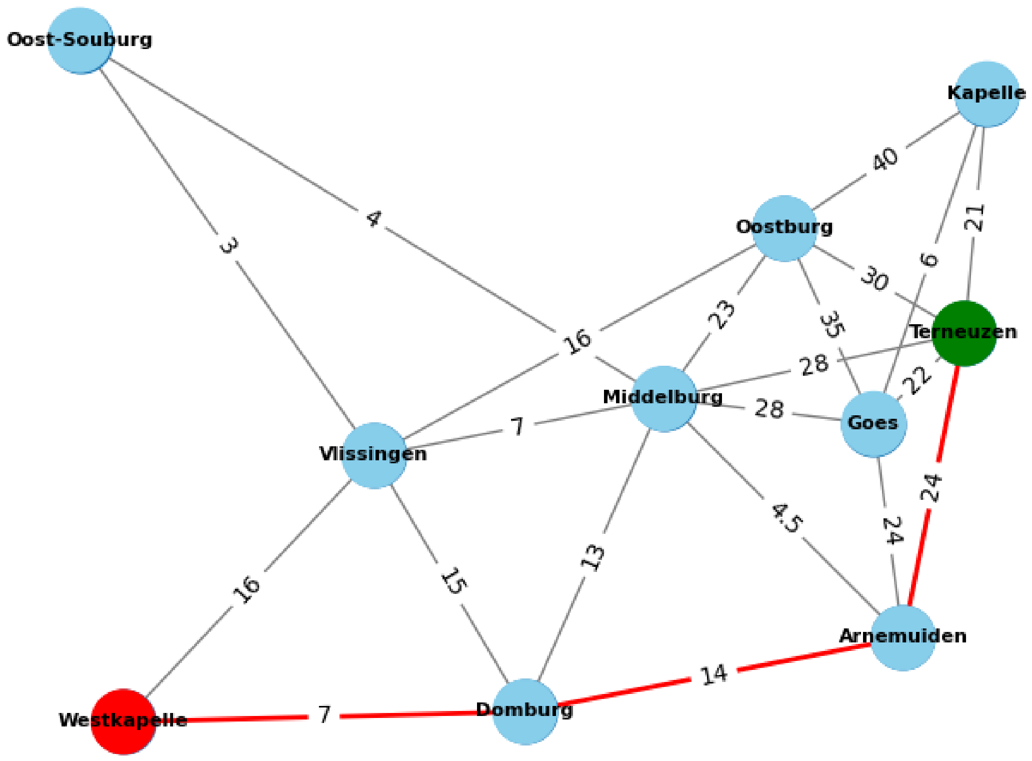
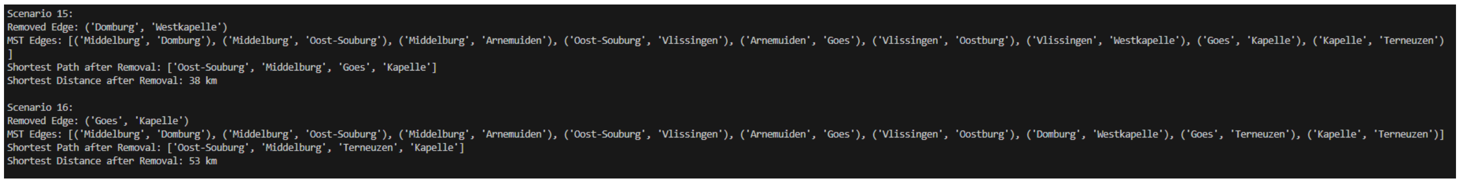


Figure 4: Shortest Path Graph from Terneuzen to Westkapelle

2.3 Sensitivity Analysis

In the conducted sensitivity analysis, the impact of temporarily removing each individual edge from the graph was systematically investigated using both Prim's and Kruskal's algorithms. For each scenario, the Minimum Spanning Tree (MST) was recalculated after the removal of the specified edge, shedding light on how the structural integrity of the graph is affected.

Simultaneously, the shortest path between two designated towns was determined, providing insights into the altered connectivity and distances within the modified graph. The results revealed the resilience of both algorithms to edge removal, showcasing their ability to adapt to changing network conditions. This analysis contributes to a deeper understanding of the robustness and adaptability of Prim's and Kruskal's algorithms in the context of Minimum Spanning Trees.



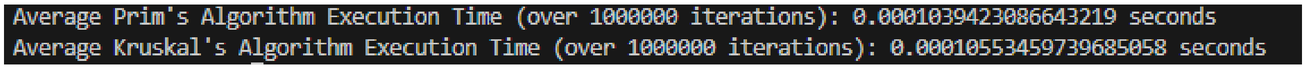
```
Scenario 15:
Removed Edge: ('Domburg', 'Westkapelle')
MST Edges: [('Middelburg', 'Domburg'), ('Middelburg', 'Oost-Souburg'), ('Middelburg', 'Arnemuiden'), ('Oost-Souburg', 'Vlissingen'), ('Arnemuiden', 'Goes'), ('Vlissingen', 'Oostburg'), ('Vlissingen', 'Westkapelle'), ('Goes', 'Kapelle'), ('Kapelle', 'Terneuzen')
]
Shortest Path after Removal: ['Oost-Souburg', 'Middelburg', 'Goes', 'Kapelle']
Shortest Distance after Removal: 38 km

Scenario 16:
Removed Edge: ('Goes', 'Kapelle')
MST Edges: [('Middelburg', 'Domburg'), ('Middelburg', 'Oost-Souburg'), ('Middelburg', 'Arnemuiden'), ('Oost-Souburg', 'Vlissingen'), ('Arnemuiden', 'Goes'), ('Vlissingen', 'Oostburg'), ('Domburg', 'Westkapelle'), ('Goes', 'Terneuzen'), ('Kapelle', 'Terneuzen')]
Shortest Path after Removal: ['Oost-Souburg', 'Middelburg', 'Terneuzen', 'Kapelle']
Shortest Distance after Removal: 53 km
```

Figure 5: Change of MST and Shortest Pathing results if certain edges are removed **(please zoom in)**

This is just an example. You may adjust the starting town and destination town as desired to see how dropping edges affects results for those various instances.

3 Results and Conclusions

Coming back to the initial proposal experiment. After running our initial experiment, we set up in the experiments section we get the following results, as seen in Figure 6.



```
Average Prim's Algorithm Execution Time (over 1000000 iterations): 0.0001039423086643219 seconds
Average Kruskal's Algorithm Execution Time (over 1000000 iterations): 0.00010553459739685058 seconds
```

Figure 6: Results of the Experiment as described in section 2

Obviously, we can see that there is no real difference in the running time of the algorithms. It is not nearly statistically significant. Thus, we can accept the null Hypothesis $H0$ with full certainty that for a simple graph like ours there is indeed no difference in the running time of these two algorithms.

Note, it is worth to mention this is a very, very simple graph and not at all complex. Running the same experiment on a different more complex problem scape may result in very different results.

We also show visual representations of the MST of both algorithms in Figures 7 and 8, to confirm they did find the correct spanning tree. Note, the representations only show the connected edges and not all.

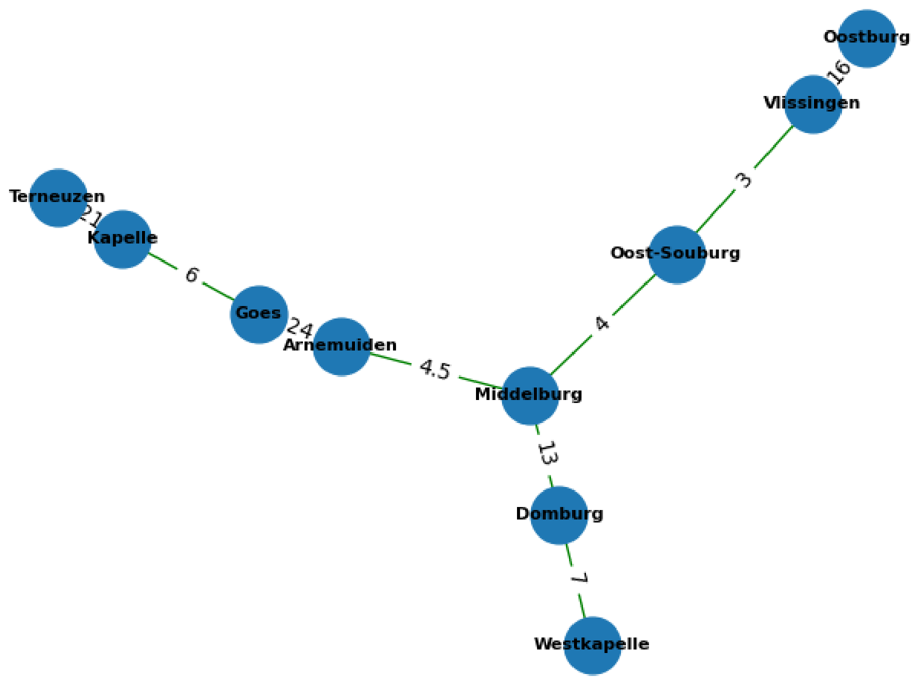


Figure 7: MST of our Zeeland Graph (Prim's)

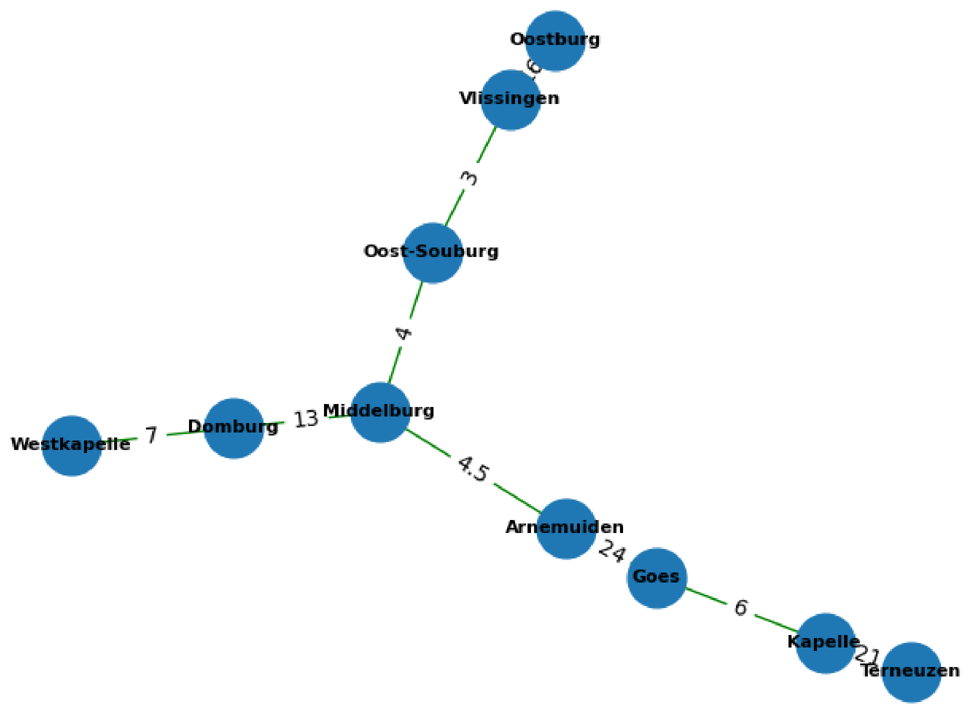


Figure 8: MST of our Zeeland Graph (Kruskal's)

4 Additional Testing and Future Work

For network planners and designers, understanding how edge removal affects Minimum Spanning Trees (MSTs) and shortest paths informs risk assessment and resilience planning. In the realm of logistics and transportation, where optimizing routes is critical, the insights gained can aid in building robust delivery networks. Furthermore, researchers and educators can leverage these findings to enhance their teaching materials and contribute to the broader understanding of graph algorithms. The open-source nature of the code allows for easy adaptation to specific use cases and encourages collaborative exploration of graph-related challenges. To enhance the validity of our investigation, additional testing could involve experimenting with graphs of varying sizes, structures, and edge weights. Analyzing the algorithms' performance on more complex or larger-scale networks would provide a more comprehensive understanding of their behavior. Moreover, considering real-world datasets and more dense regions featuring thousands of towns/cities would show more significant results. Additionally, exploring the impact of edge removal on other graph-related metrics, such centrality measures, could offer deeper insights into the algorithms' adaptability.

References

Google Maps. (n.d.). https://www.google.com/maps

GeeksforGeeks. (2023, October 5). *Kruskal s Minimum Spanning Tree MST Algorithm*. https://www.geeksforgeeks.org/kruskals-minimum-spanning-tree-algorithm-greedy-algo-2/

Kruskal MST Visualzation. (n.d.). https://www.cs.usfca.edu/~galles/visualization/Kruskal.html

Prim MST Visualzation. (n.d.). https://www.cs.usfca.edu/~galles/visualization/Prim.html

Wikipedia contributors. (2023, December 11). *Zeeland*. Wikipedia. https://en.wikipedia.org/wiki/Zeeland