CSC4202 **DESIGN AND ANALYSIS OF ALGORITHM**

AMBULANCE DISPATCH TEAM

GROUP 3

GROUP MEMBERS

NUR MUHAMMAD AIDIL BIN NUR ANUAR
222355

MUHAMMAD HAKIM ASYRAF BIN ISMAIL
223234

MOHAMMAD ARRAZAQ ZIKRY BIN MOHD AZHAR
222641

MUHAMMAD DANIAL HAZIQ BIN AB MANAH
225031

INTRODUCTION

Problem Statement

The challenge is to select the optimal route using a consistent, pre-mapped road network that minimizes total travel distance, especially in environments where real-time traffic data is unavailable.

Without a shortest-distance algorithm in place, ambulances may take longer routes unnecessarily, increasing response times and putting patients' lives at greater risk.

Objectives

To implement a shortest-distance routing system using a consistent, pre-mapped road network.

To reduce response time by avoiding unnecessary delays caused by suboptimal routing decisions.

Importance for Finding The Solution

1. Reduces Critical Response Time
   - To minimize the travel distance of the ambulance to reach the patient and transport them to the hospital.
   - Increase the chance of survival of time-sensitive emergencies (cardiac arrest or accident) and also reduce long-term complications.

2. Reduce The Algorithm Running Time
   - Choosing the best algorithm is crucial to find shortest path for the ambulance.
   - Ensure all incidents are covered efficiently without unnecessary delay caused by algorithm.

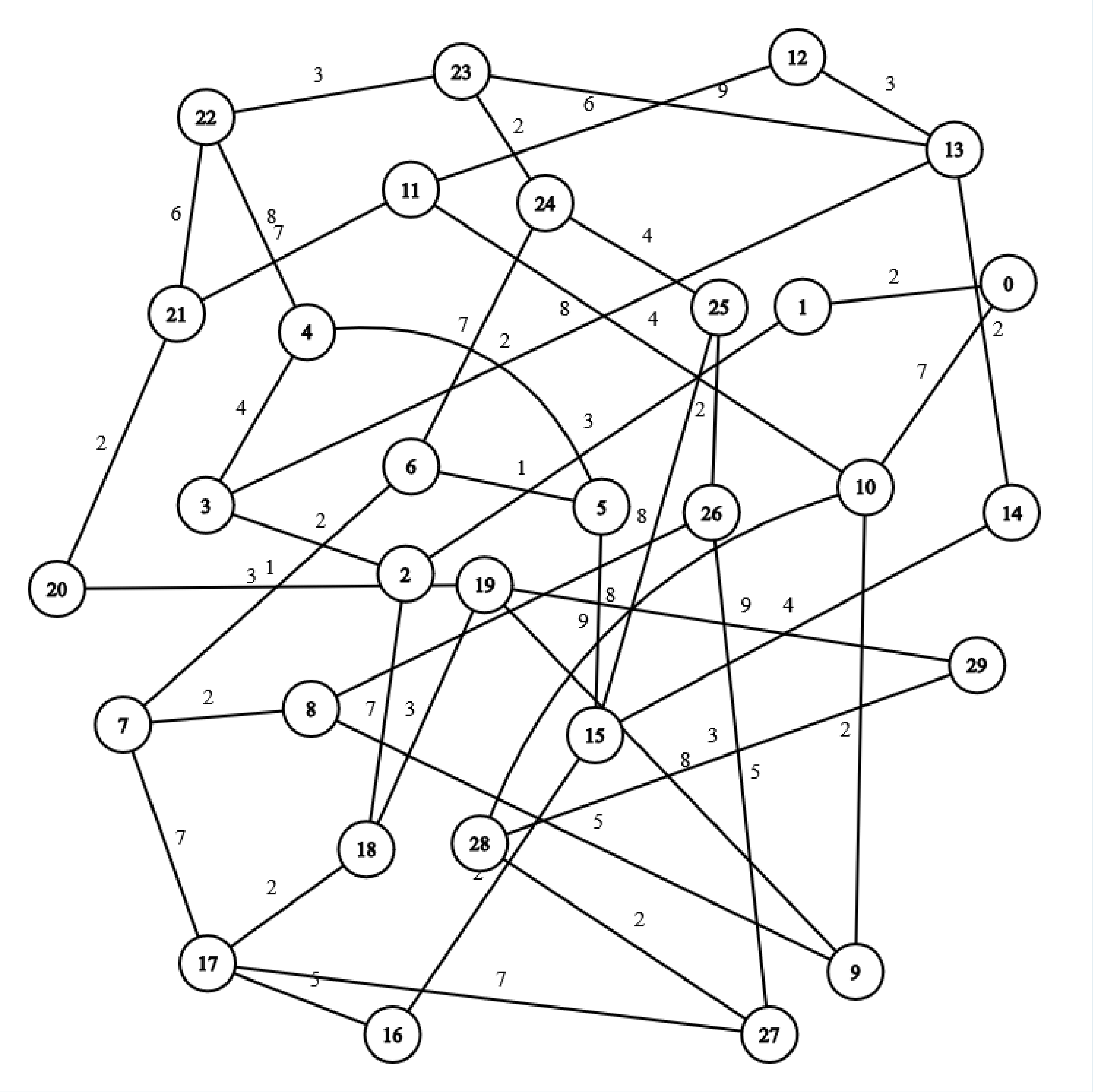
Comparison Of Brute Force and Dijkstra (Theoretically)

REFERENCE
V=Number of Nodes
E=Number of Edges

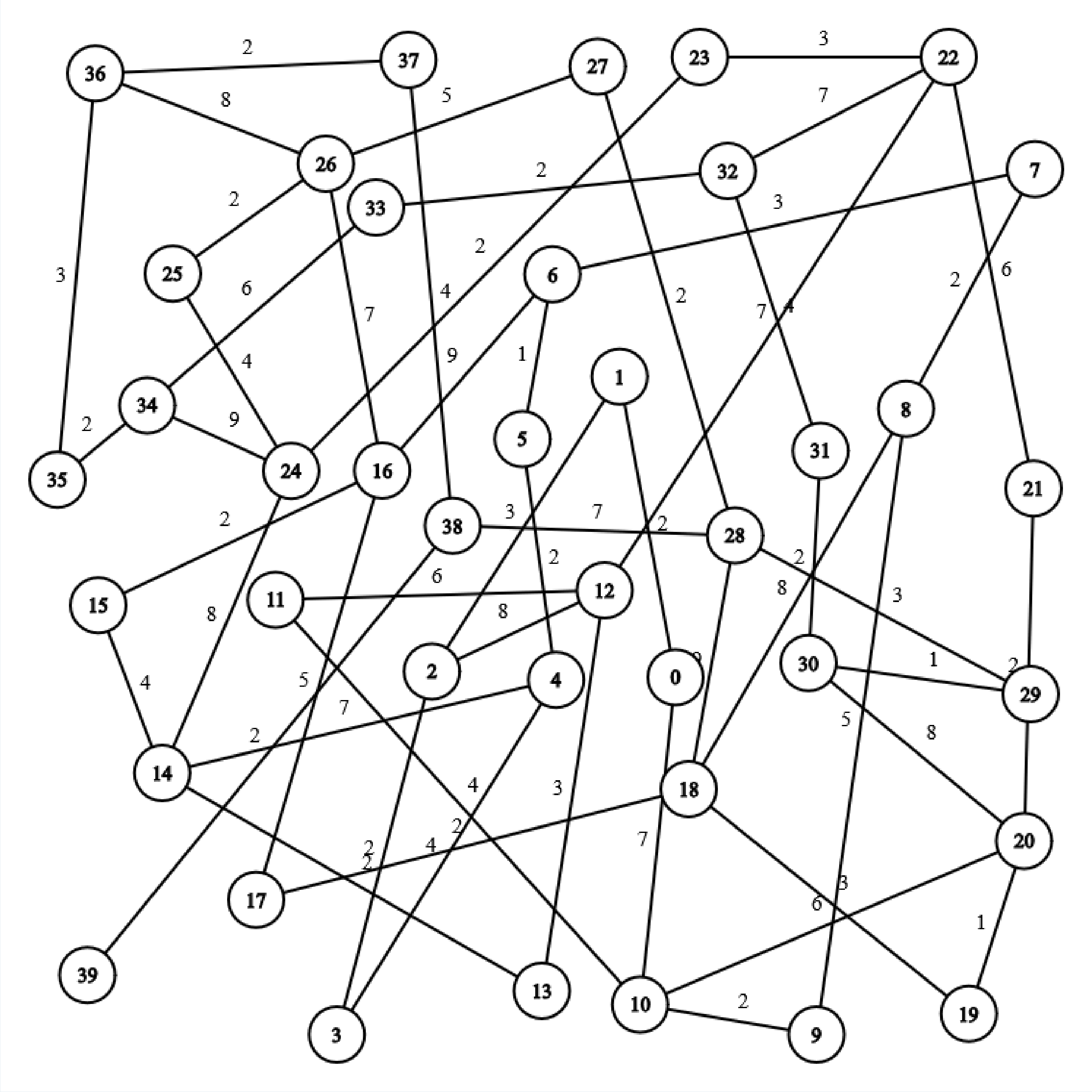
Dijkstra		Brute Force
Greedy algorithm that uses min heap priority	Approach	Combanitorial exhaustive search, typically using recursion
Has a time complexity of O((V + E) log V) with binary heap	Time Complexity	Worst-case tme complxity O(V!)
High scalability due to its quasi-linear behaviour	Scalability	Poor scalability due exponential behaviour.
Highly reliable with non-negative edge weight.	Practicality	it is impractical due to its non-polynomial runtime.

Node Map

vertices = 30

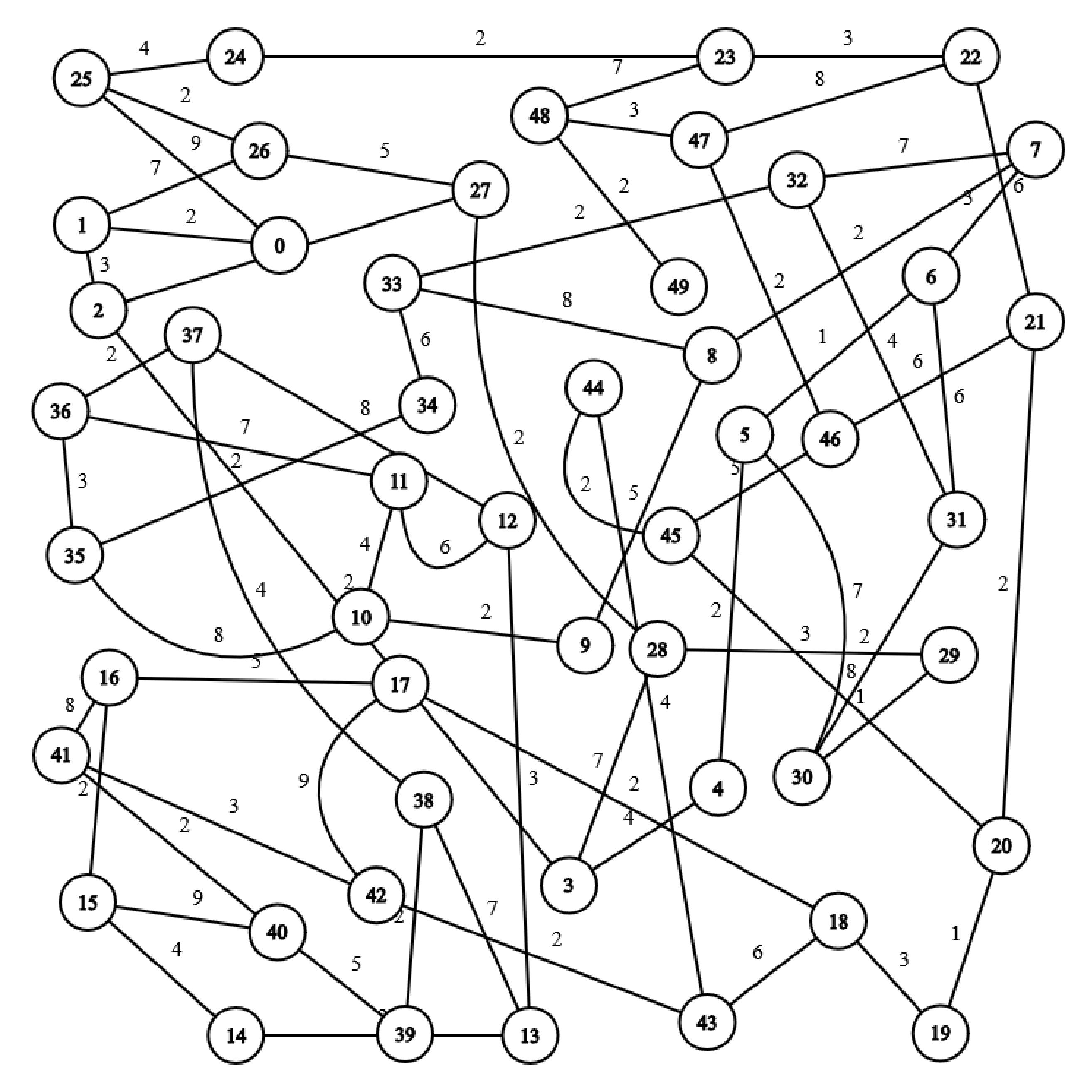


vertices = 40

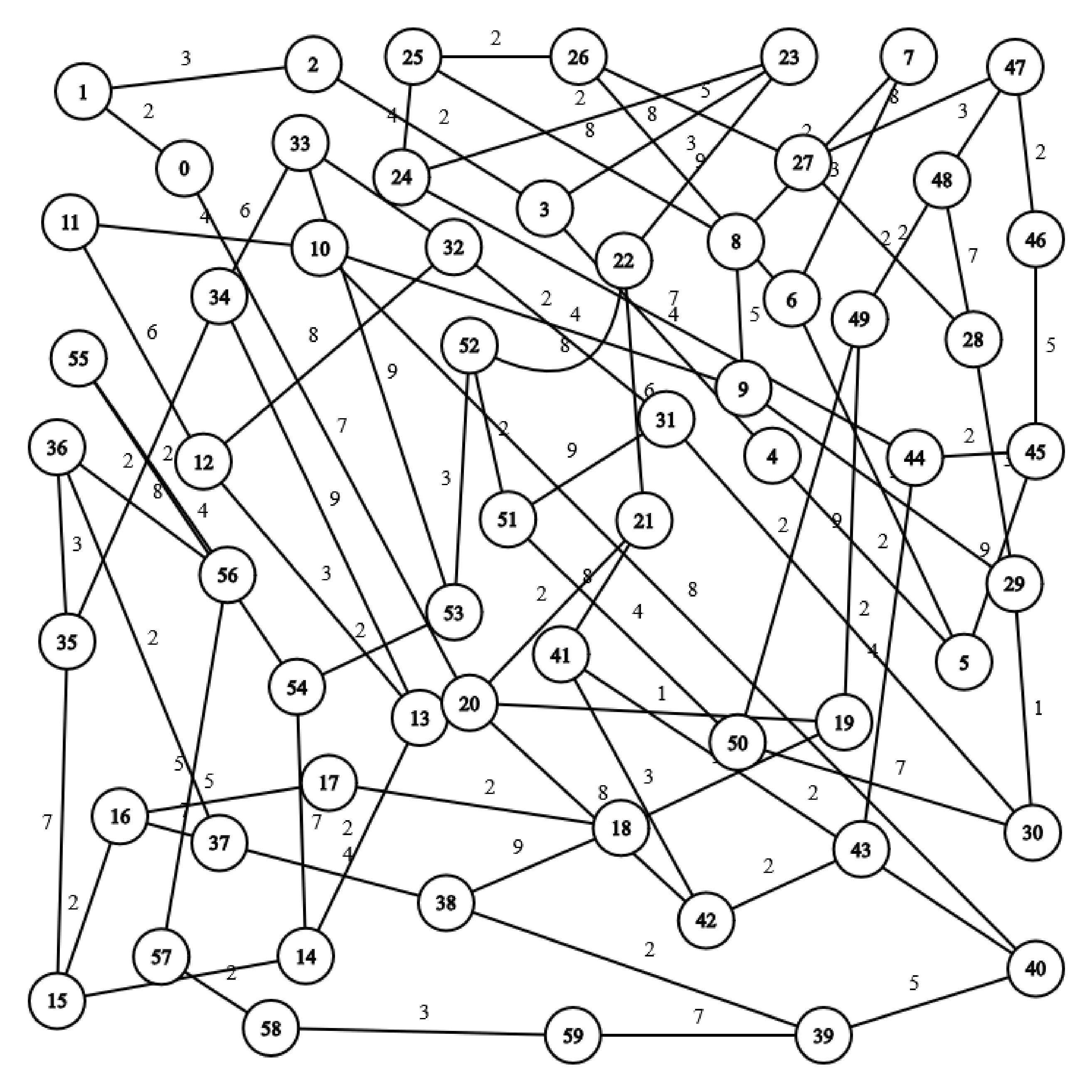


Node Map

vertices = 50



vertices = 60



Sample Output from Coding

vertices = 30

```
Enter source node (0-29): 0
Enter destination node (0-29): 29

[ Dijkstra ]
Shortest distance = 18
Path: [0, 10, 28, 29]
Elapsed time: 2635 microsecs

[ Brute Force ]
Shortest distance = 18
Path: [0, 10, 28, 29]
Elapsed time: 7777 microsecs
```

vertices = 40

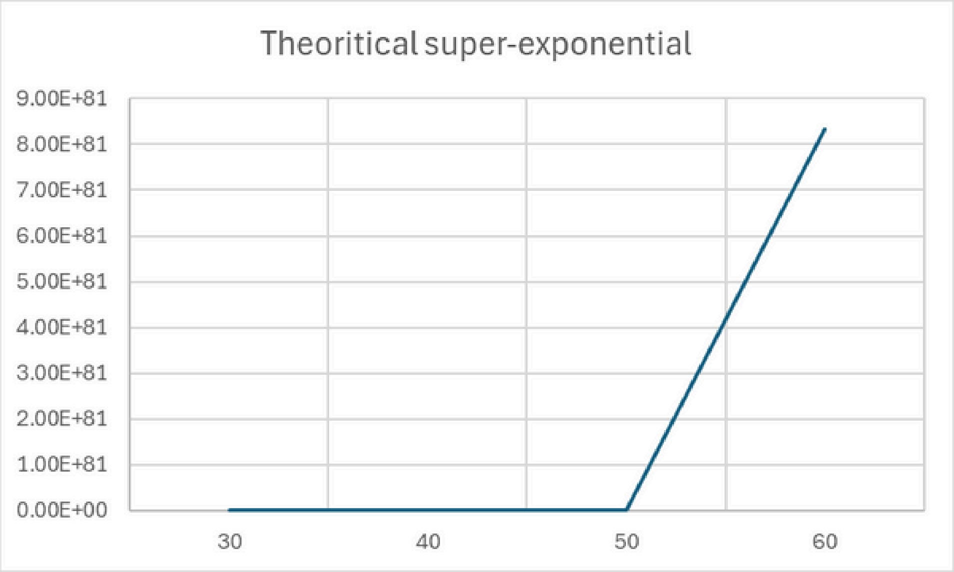
```
Enter source node (0-39): 0
Enter destination node (0-39): 39

[ Dijkstra ]
Shortest distance = 34
Path: [0, 10, 20, 30, 29, 28, 38, 39]
Elapsed time: 2788 microsecs

[ Brute Force ]
Shortest distance = 34
Path: [0, 10, 20, 30, 29, 28, 38, 39]
Elapsed time: 11064 microsecs
```

vertices = 50

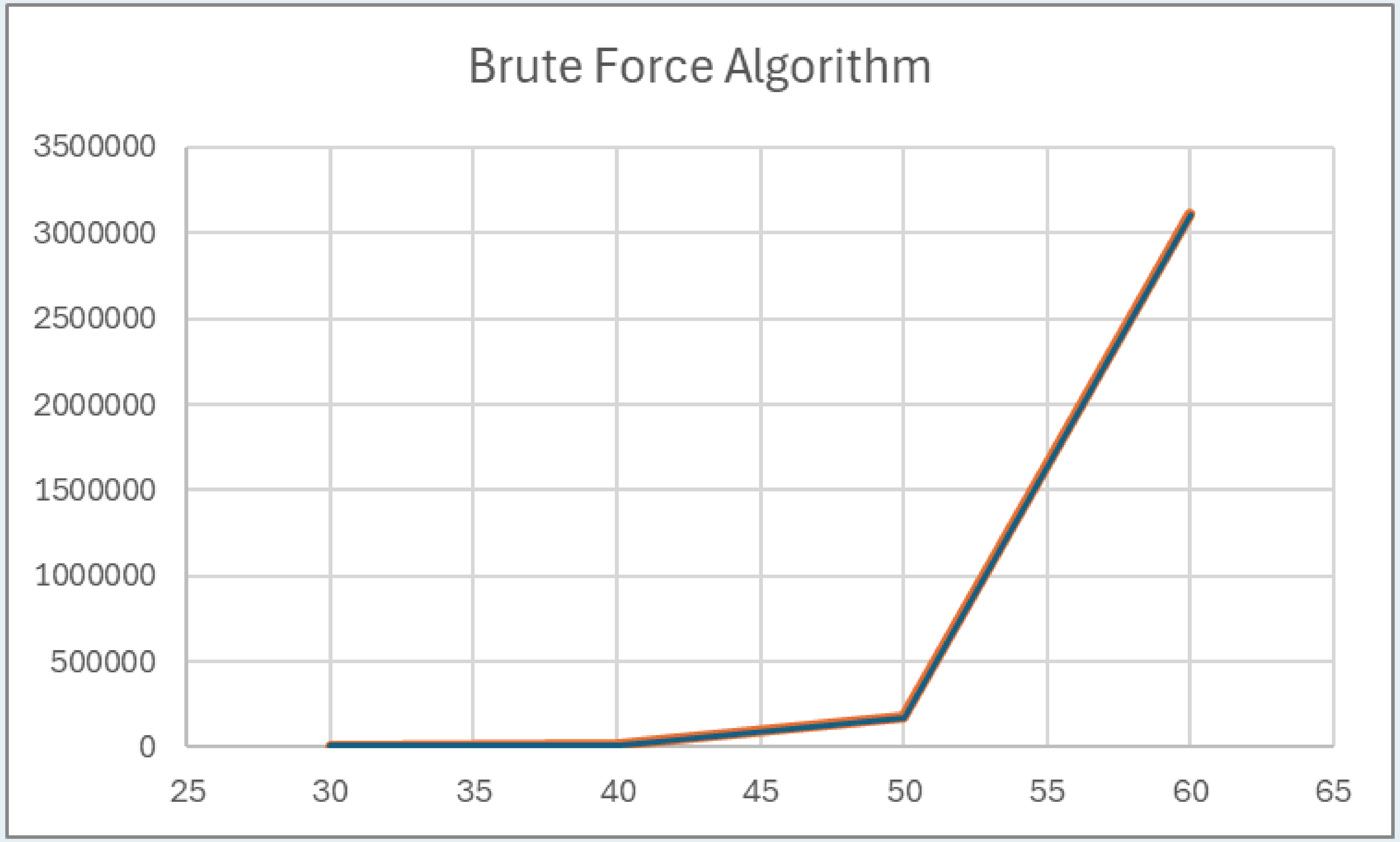
```
Enter source node (0-49): 0
Enter destination node (0-49): 49

[ Dijkstra ]
Shortest distance = 24
Path: [0, 25, 24, 23, 48, 49]
Elapsed time: 2888 microsecs

[ Brute Force ]
Shortest distance = 24
Path: [0, 25, 24, 23, 48, 49]
Elapsed time: 171752 microsecs
```

vertices = 60

```
Enter source node (0-59): 0
Enter destination node (0-59): 59

[ Dijkstra ]
Shortest distance = 29
Path: [0, 20, 19, 18, 38, 39, 59]
Elapsed time: 3099 microsecs

[ Brute Force ]
Shortest distance = 29
Path: [0, 20, 19, 18, 38, 39, 59]
Elapsed time: 3158153 microsecs
```



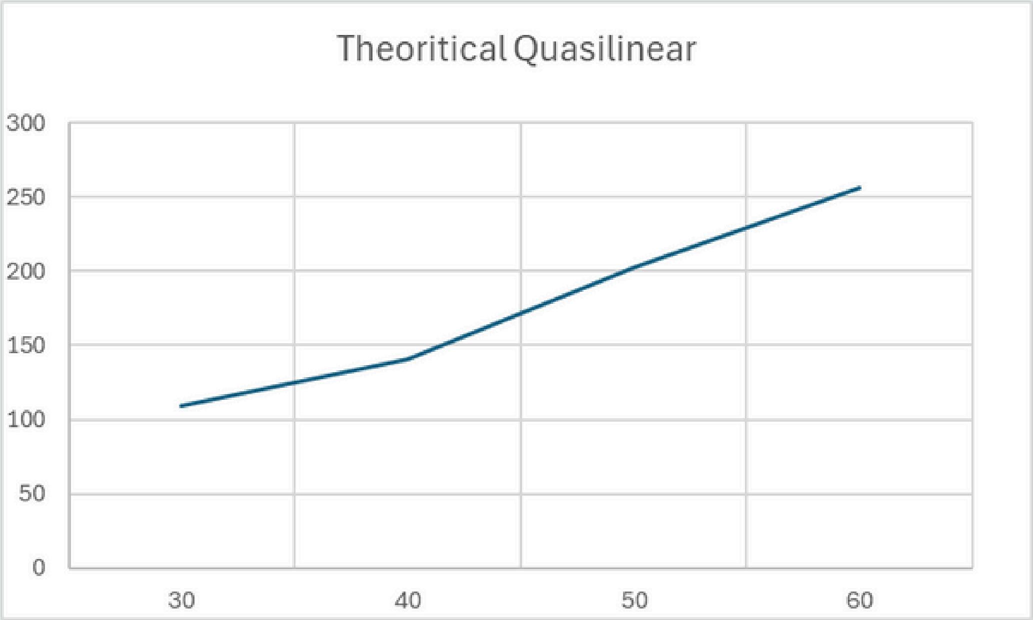
Brute Force Algorithm graph



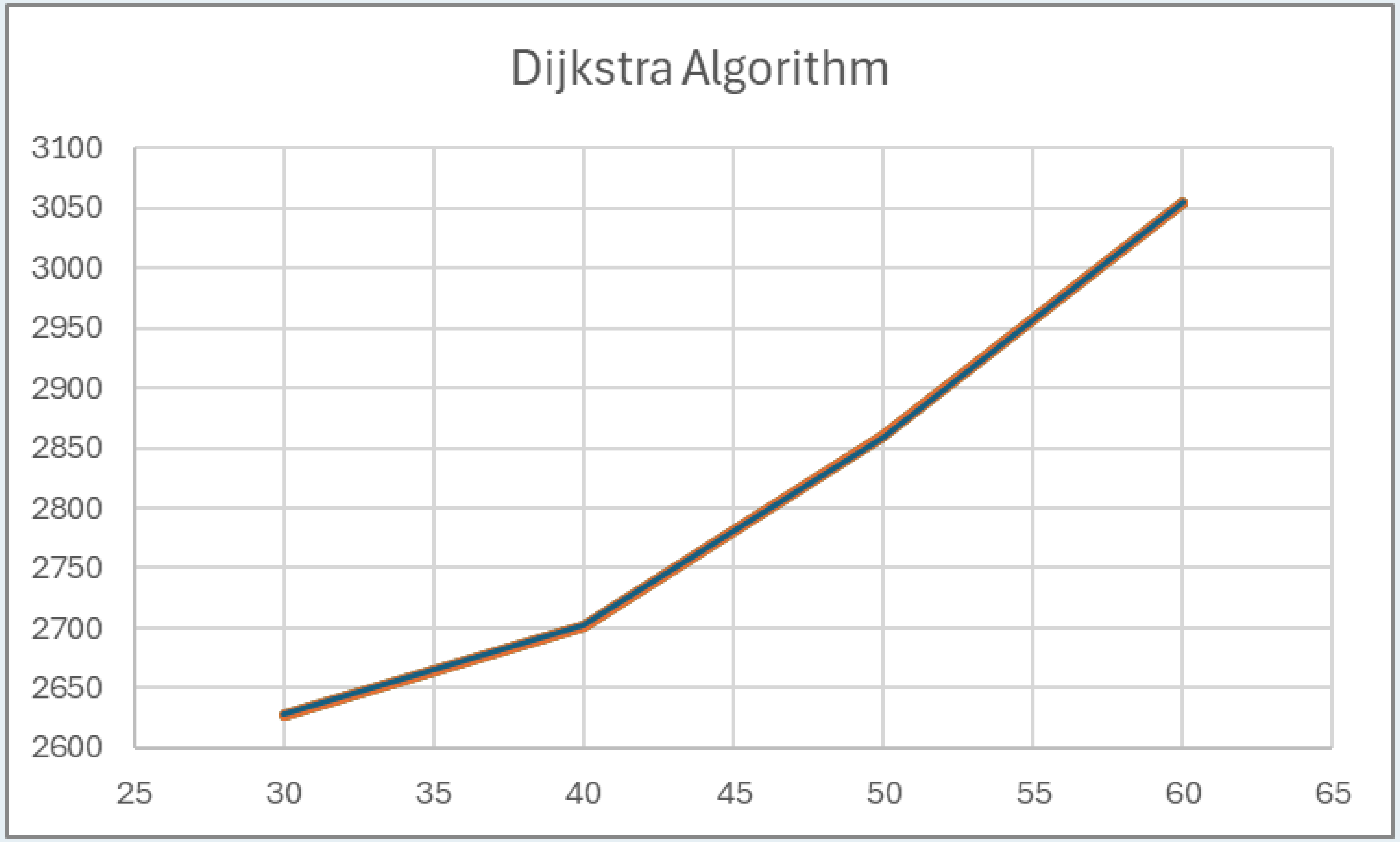
Vertices Size	Elapsed Time (microseconds)
30	8128
40	11163
50	177162
60	3108688

Time Complexity

V !



Dijkstra Algorithm Graph

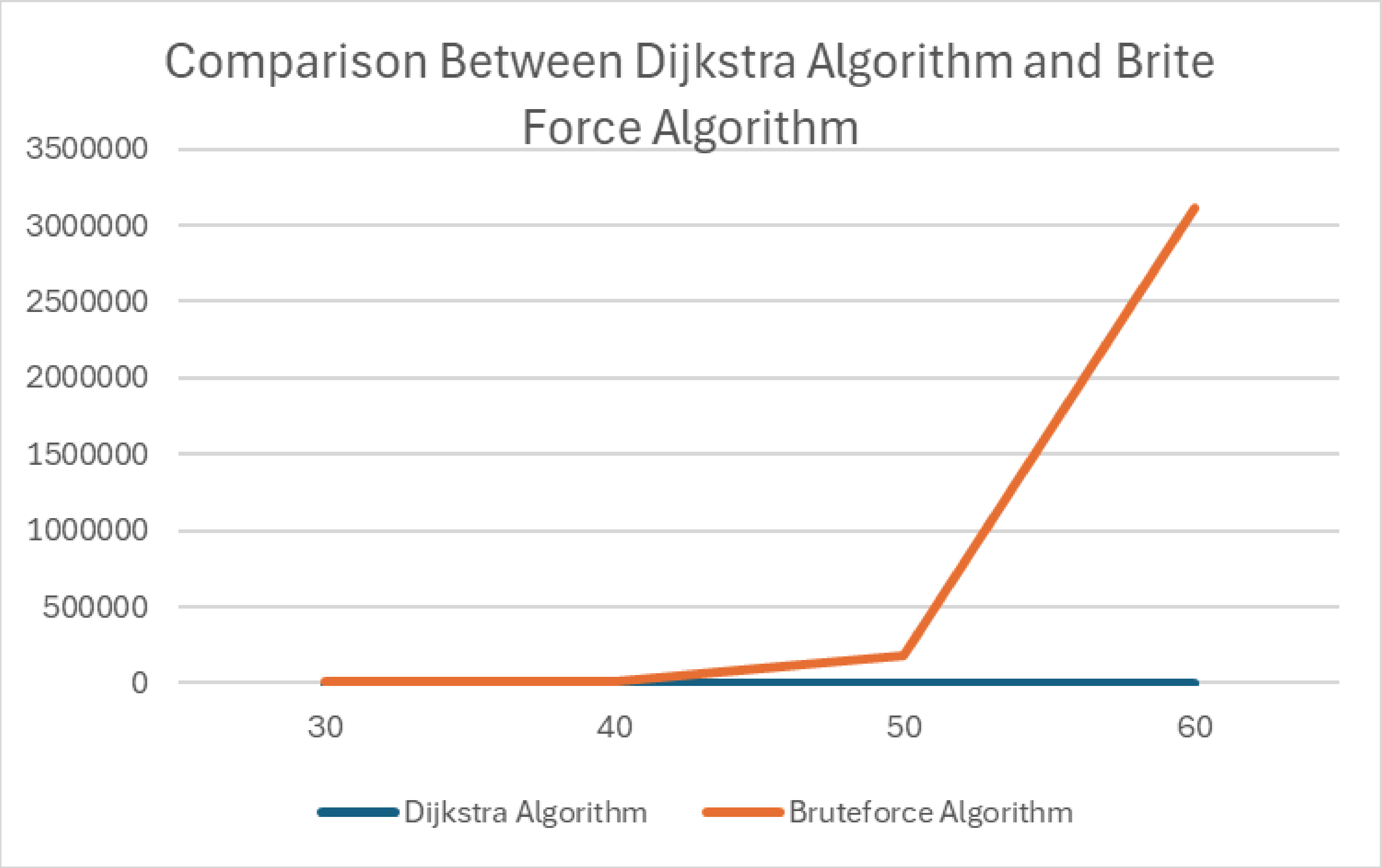


Vertices Size	Elapsed Time (microseconds)
30	2628
40	2702
50	2859
60	3055

Time Complexity

$(V + E) \operatorname{Log} V$

Comparison

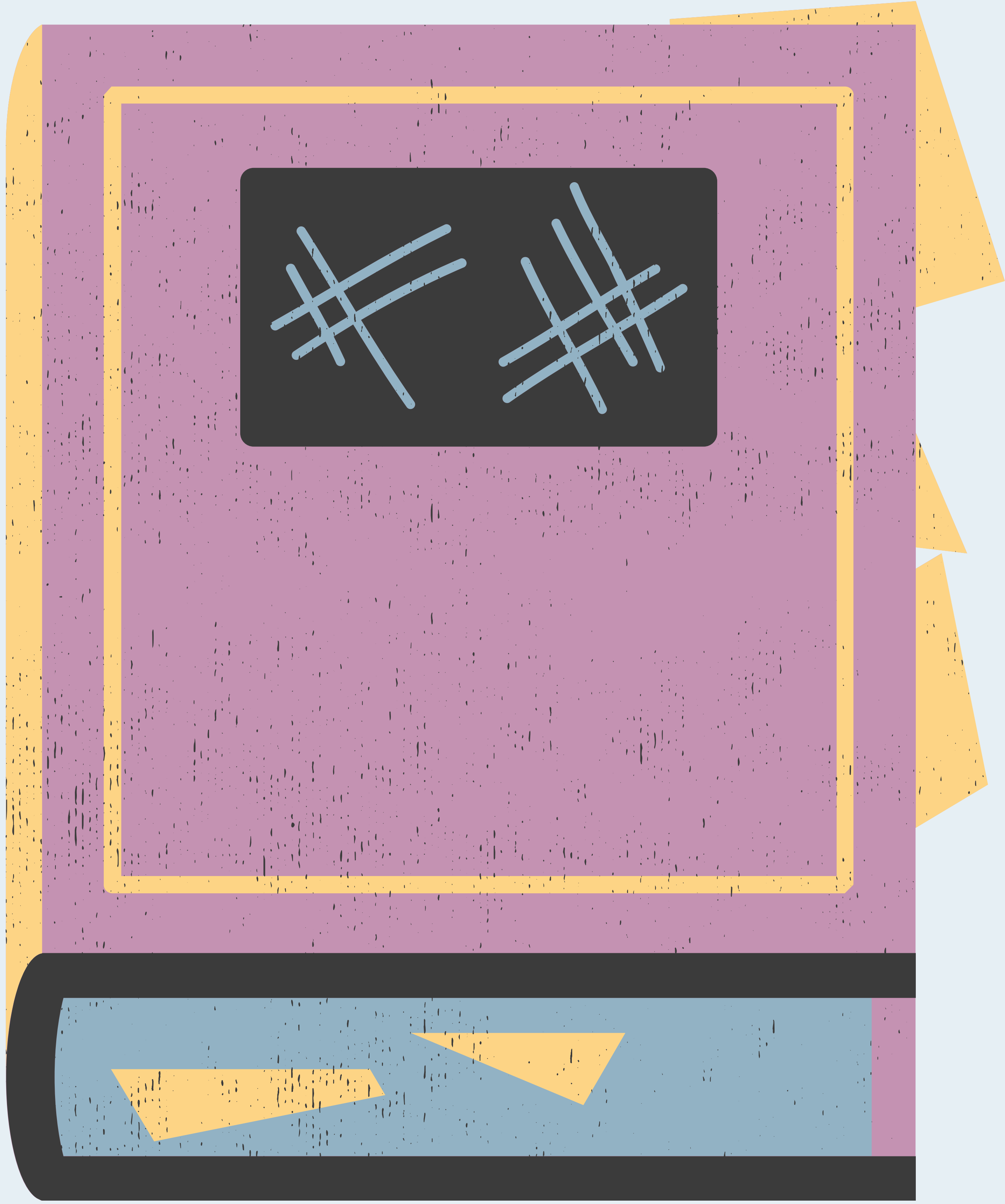


It's very difficult to compare these two algorithms on the same graph because

the brute force algorithm grows super-exponentially; values to increase extremely fast

Dijkstra's algorithm is quasilinear; scales much more slowly

Due to the brute force algorithm's huge values, the dijkstra growth looks almost flat by comparison. In reality, it is increasing too just at a much slower rate that's hard to see on the same graph



Conclusion

- *The brute force algorithm grows super-exponentially, causing values to increase extremely fast as input size increases.*
- *Dijkstra's algorithm grows quasilinear, making it significantly more efficient and scalable.*
- *This visual gap highlights the vast efficiency difference between the two approaches.*
- *Dijkstra's algorithm is far more practical for larger problem sizes, offering consistent performance and lower computational cost.*

REFERENCES

- Iqbal, M., Zhang, K., Iqbal, S., & Tariq, I. (2018). A fast and reliable Dijkstra algorithm for online shortest path. Int. J. Comput. Sci. Eng, 5(12), 24-27.

- Madkour, A., Aref, W. G., Rehman, F. U., Rahman, M. A., & Basalamah, S. (2017). A survey of shortest-path algorithms. arXiv preprint arXiv:1705.02044.

- Candra, A., Budiman, M. A., & Hartanto, K. (2020, July). Dijkstra's and a-star in finding the shortest path: A tutorial. In 2020 International Conference on Data Science, Artificial Intelligence, and Business Analytics (DATABIA) (pp. 28-32). IEEE.

CSC4202 **DESIGN AND ANALYSIS OF ALGORITHM**

THANK YOU

GROUP 3