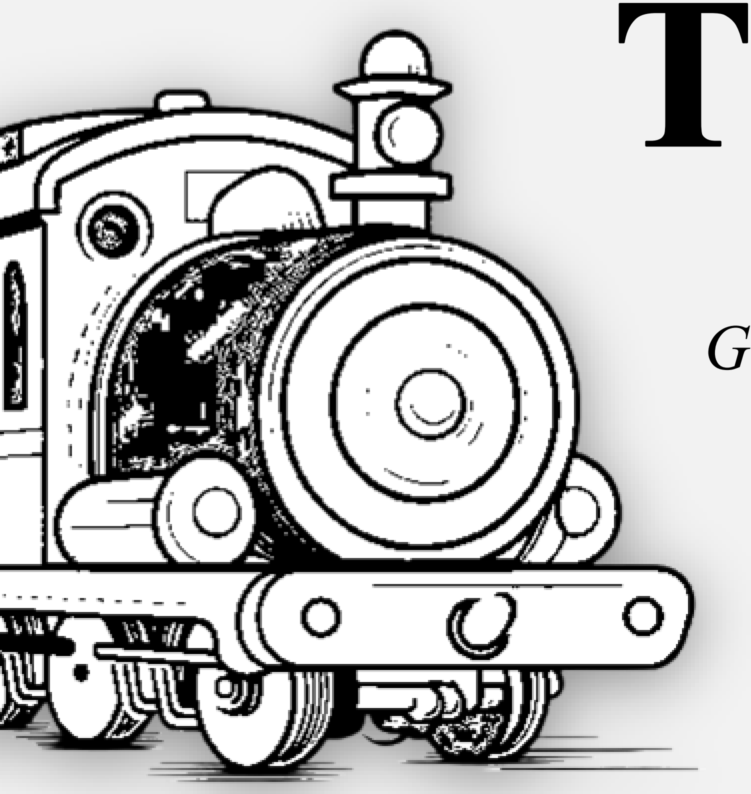
Ticket To Tech



Group Members: Carrie Houston, Makenzie Kadjeski, & Jeny Thomas

Problem Overview

➢Problem:
  - Finding Optimal Path to Classes

➢Algorithms Implemented in Python & C++:
  - Breadth First Search (BFS)
  - Depth First Search(DFS)
  - A* Search
  - Bidirectional Search

➢ The goal is to find the most efficient algorithm to solve this problem.

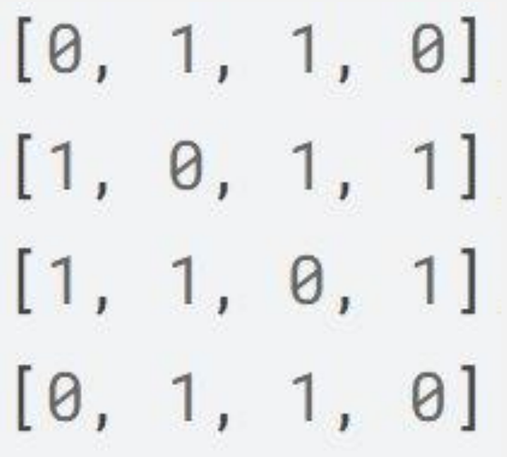
Objective & Research:

- Objective: Determine which algorithm is the most efficient for finding the optimal path between two places.
- Time complexity :
  - BFS, DFS, and A*: $O(|n|^2)$
  - Bidirectional: $O(|b|^{d/2})$
- Expected Results: Bidirectional will be the most efficient algorithm

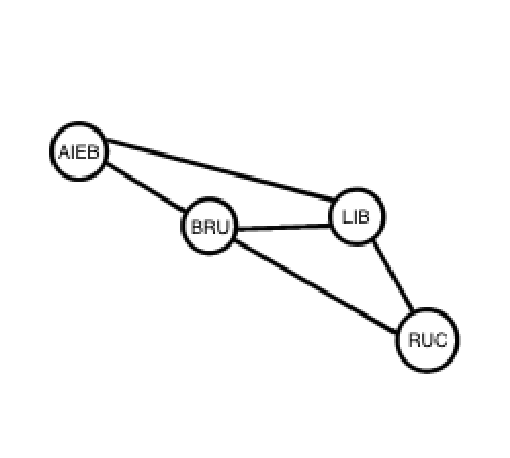
Description of Experiment:

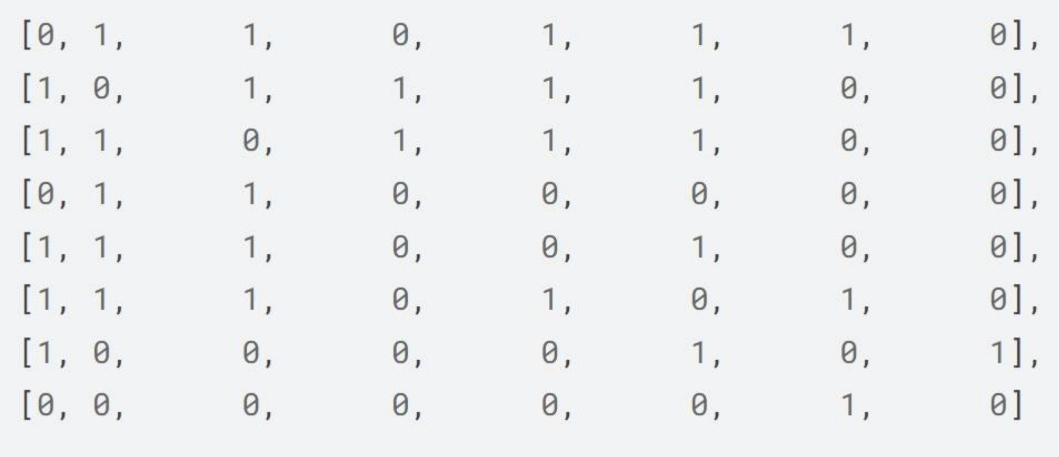
- Real World Datasets of Tennessee Tech’s Campus
  - Small, medium, & large unweighted graphs
- Tested code on Makenzie’s MacBook Pro five times for:
  - Each algorithm on each graph
  - Brute force on small and medium graphs
- Routes we tested
- Comparisons

Input Matrices and Graphs

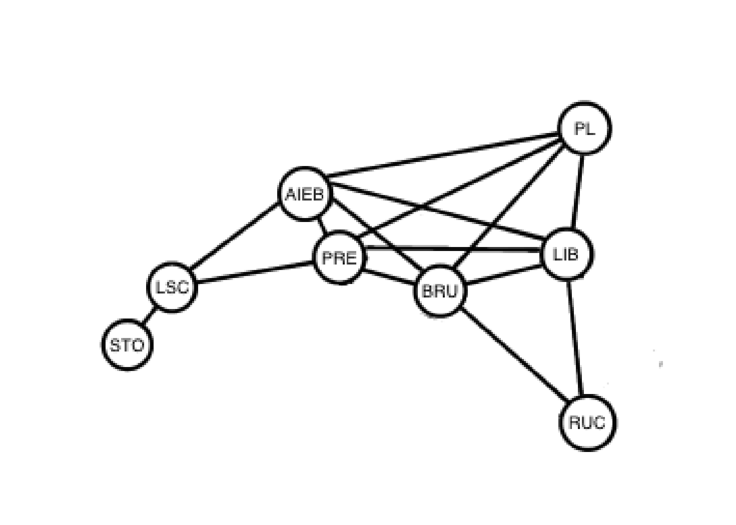


```
[0, 1, 1, 0]
[1, 0, 1, 1]
[1, 1, 0, 1]
[0, 1, 1, 0]
```

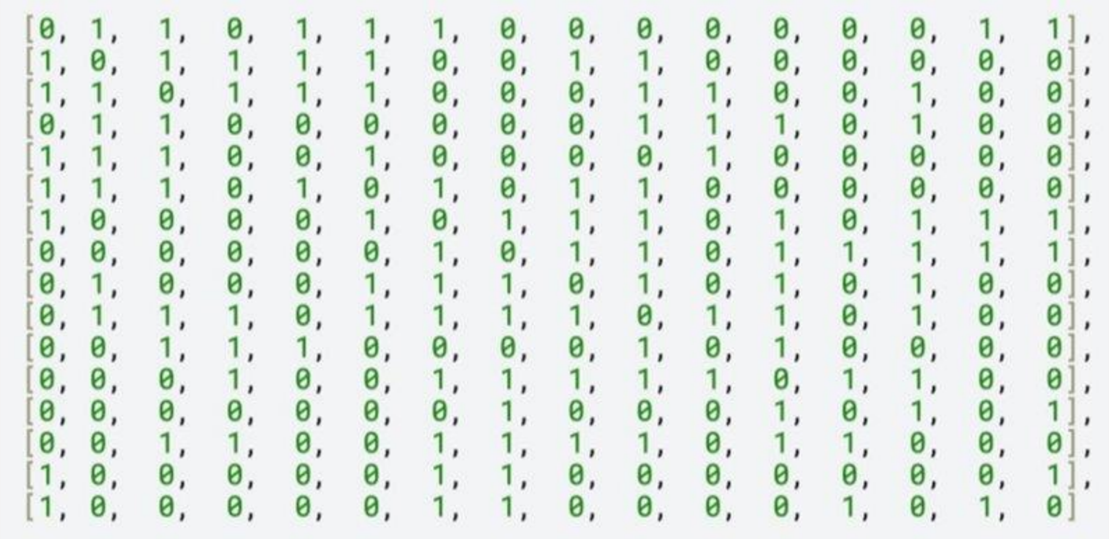




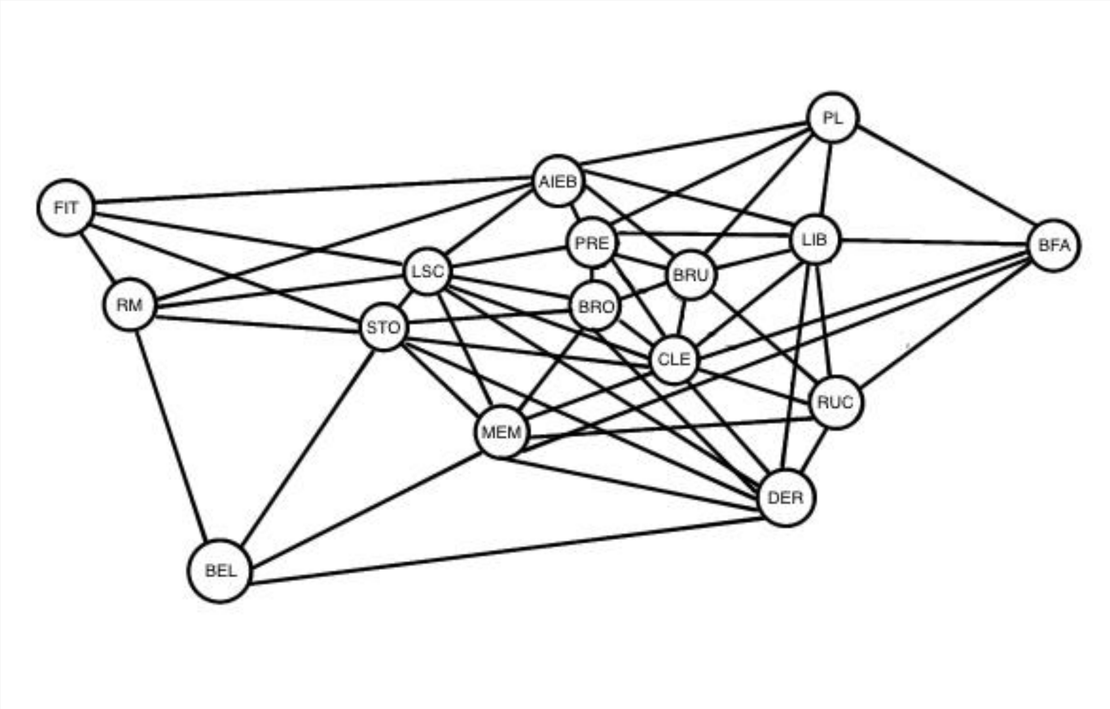
```
[0, 1,      1,      0,      1,      1,      1,      0],
[1, 0,      1,      1,      1,      1,      0,      0],
[1, 1,      0,      1,      1,      1,      0,      0],
[0, 1,      1,      0,      0,      0,      0,      0],
[1, 1,      1,      0,      0,      1,      0,      0],
[1, 1,      1,      0,      1,      0,      1,      0],
[1, 0,      0,      0,      0,      1,      0,      1],
[0, 0,      0,      0,      0,      0,      1,      0]
```



Input Matrices and Graphs



```
[0, 1,  1,  0,  1,  1,  1,  0,  0,  0,  0,  0,  0,  0,  1,  1],
[1, 0,  1,  1,  1,  1,  0,  0,  1,  1,  0,  0,  0,  0,  0,  0],
[1, 1,  0,  1,  1,  1,  0,  0,  0,  1,  1,  0,  0,  1,  0,  0],
[0, 1,  1,  0,  0,  0,  0,  0,  0,  1,  1,  1,  0,  1,  0,  0],
[1, 1,  1,  0,  0,  1,  0,  0,  0,  0,  1,  0,  0,  0,  0,  0],
[1, 1,  1,  0,  1,  0,  1,  0,  1,  1,  0,  0,  0,  0,  0,  0],
[1, 0,  0,  0,  0,  1,  0,  1,  1,  1,  0,  1,  0,  1,  1,  1],
[0, 0,  0,  0,  0,  0,  1,  0,  1,  1,  0,  1,  1,  1,  1,  1],
[0, 1,  0,  0,  0,  1,  1,  1,  0,  1,  0,  1,  0,  1,  0,  0],
[0, 1,  1,  1,  0,  1,  1,  1,  1,  0,  1,  1,  0,  1,  0,  0],
[0, 0,  1,  1,  1,  0,  0,  0,  0,  1,  0,  1,  0,  0,  0,  0],
[0, 0,  0,  1,  0,  0,  1,  1,  1,  1,  1,  0,  1,  1,  0,  0],
[0, 0,  0,  0,  0,  0,  0,  1,  0,  0,  0,  1,  0,  1,  0,  1],
[0, 0,  1,  1,  0,  0,  1,  1,  1,  1,  0,  1,  1,  0,  0,  0],
[1, 0,  0,  0,  0,  0,  1,  1,  0,  0,  0,  0,  0,  0,  0,  1],
[1, 0,  0,  0,  0,  0,  1,  1,  0,  0,  0,  0,  1,  0,  1,  0]
```

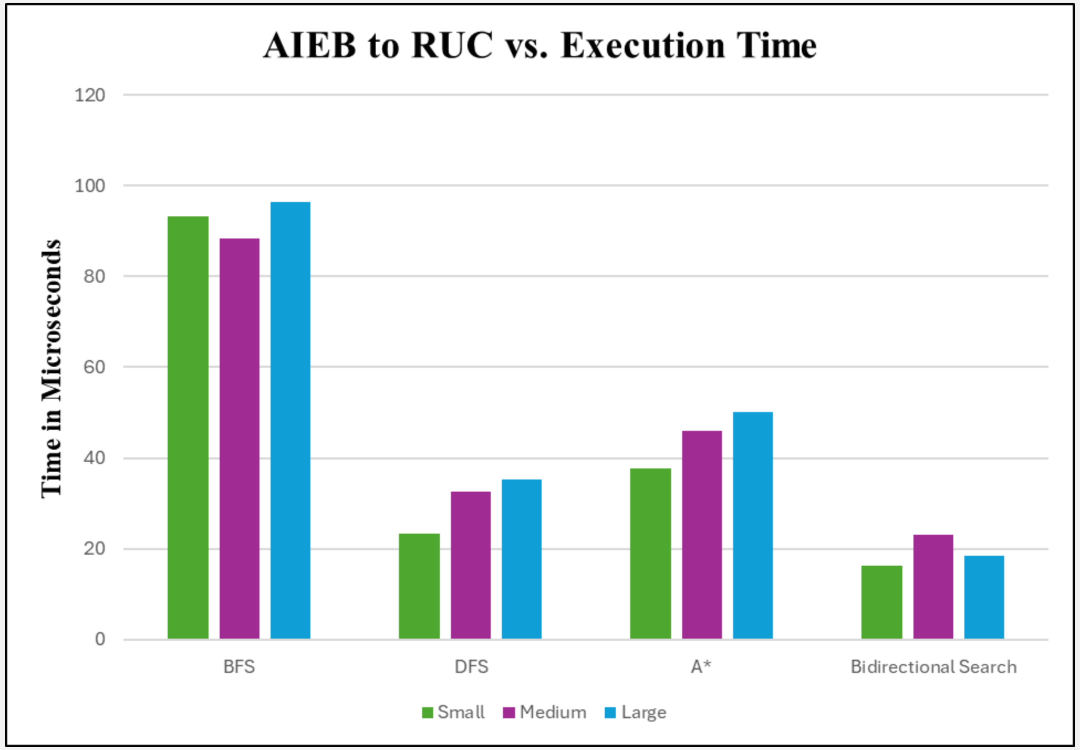


Results:

- Execution times ranked from lowest to highest:
    - Bidirectional
    - DFS
    - A*
    - BFS

- Most consistent algorithms:
    - BFS
    - Bidirectional

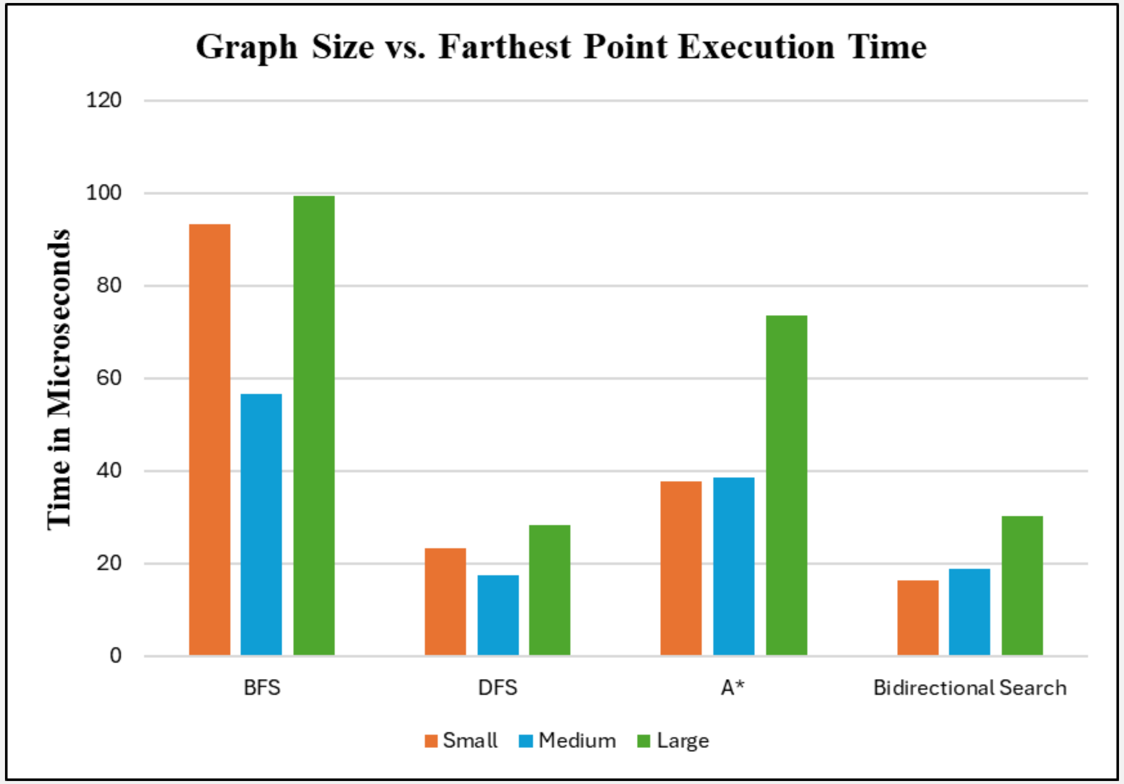
Figure 1 and Table 1



AIEB to RUC vs. Execution Time (μs)

Algorithm	Small	Medium	Large
BFS	93.2	88.4	96.4
DFS	23.4	32.6	35.4
A*	37.8	46	50.2
Bidirectional Search	16.4	23.2	18.4

Figure 2 & Table 2

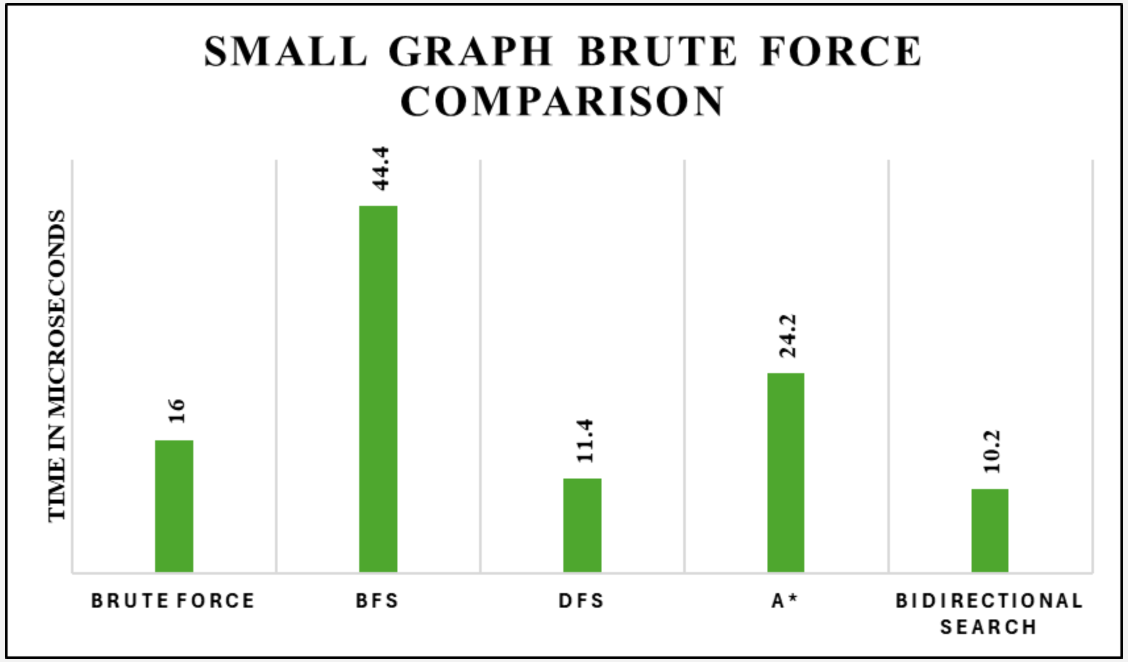


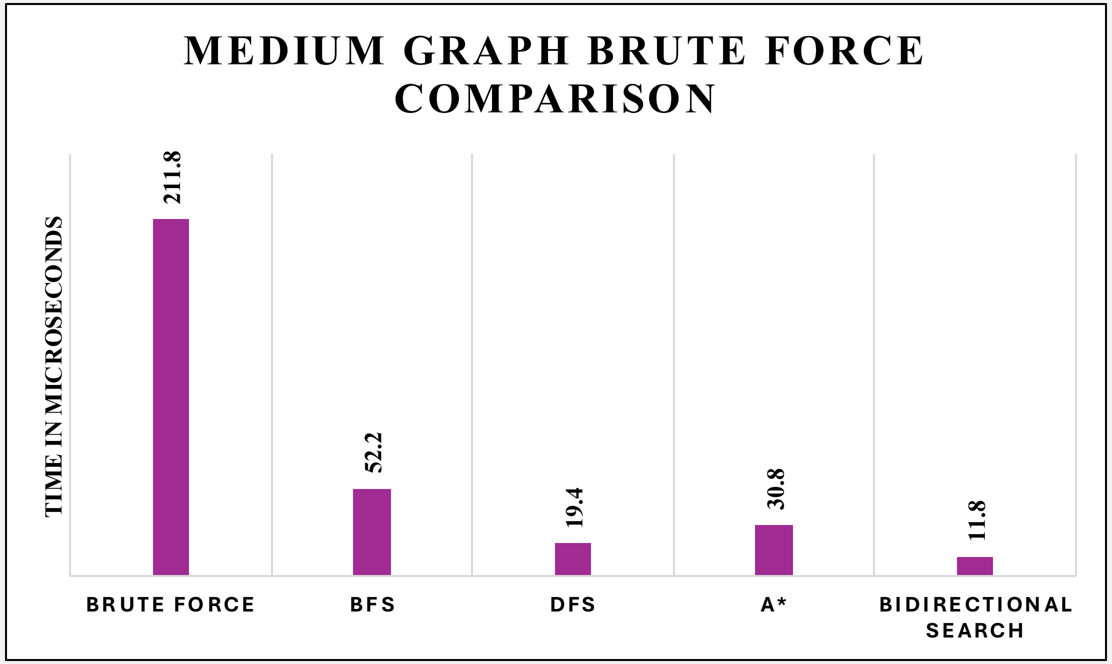
Graph Size vs. Farthest Point Execution Time (µs)			
Algorithm	**Small**	**Medium**	**Large**
BFS	93.2	56.6	99.4
DFS	23.4	17.6	28.2
A*	37.8	38.6	73.6
Bidirectional Search	16.4	18.8	30.2

Table 3:

Average of Brute Force Method vs. Path Planning Algorithms		
Algorithm	**Small**	**Medium**
Brute Force	16 µs	211.8 µs
BFS	44.4 µs	52.2 µs
DFS	11.4 µs	19.4 µs
A*	24.2 µs	30.8 µs
Bidirectional Search	10.2 µs	11.8 µs

Figures 3 & 4


SMALL GRAPH BRUTE FORCE COMPARISON
TIME IN MICROSECONDS
16
44.4
11.4
24.2
10.2
BRUTE FORCE
BFS
DFS
A*
BIDIRECTIONAL SEARCH


MEDIUM GRAPH BRUTE FORCE COMPARISON
TIME IN MICROSECONDS
211.8
52.2
19.4
30.8
11.8
BRUTE FORCE
BFS
DFS
A*
BIDIRECTIONAL SEARCH

Interpretation Summary:

- Ran the program 5 times per input

- Larger number of vertices → longer execution times

- Bidirectional was most efficient when searching for the path between the furthest points; BFS was least efficient

- Brute force is most efficient with small input sizes

Limitations & Next Steps:

- Limited input size, graph structure, limited time
- Larger input size, sparser graph, implement a weighted graph, measure different outputs

QR Code:

References:

- GeeksforGeeks. (2008). GeeksforGeeks. https://www.geeksforgeeks.org/ -used in objective/research
- Kumar, Rajesh. (2024). DataCamp. The A* Algorithm: A Complete Guide | DataCamp
- Levitin, A. (2012). *Introduction to the design & analysis of algorithms*. Pearson. -used in objective/research
- OpenAI. (2025). *ChatGPT* [Large language model]. OpenAI. https://chat.openai.com -experimentation
- Radian, C. (2025). *Section 3.5 DFS_BFS Presentation*. -used in objective/research