SPEED OF SEARCH ALGORITHMS

Soumyadeep Das

PROBLEM STATEMENT

What is the fastest graph-searching algorithm?

WHY THE RESEARCHER CHOSE THIS PROJECT & BENEFITS

The researcher chose this project due to his interest in computer science. He has been coding since he was 8, and finding out about how GPS's work simply amazed him. This project will benefit many people, as many people use a wide variety of GPSs to commute from place to place, such as to work. Improvement to these algorithms will help everyone.

BACKGROUND INFORMATION

Algorithm: A process or set of rules to be followed in calculations or other problem-solving operations, especially by a computer.

Python: an interpreted, object-oriented, high-level programming language with dynamic semantics

Algorithmic Complexity: a measure of how long an algorithm would take to complete given an input of size n

Graph (graph theory): models pairwise relations between objects/nodes. Graphs consist of vertices which are connected by edges.

Graph Traversal: process of visiting (or checking) each vertex in a graph

Leaf node: node with no children/successors

A*: the "gold standard" of shortest path finding. Used by Google Maps to find routes.

Big O Notation: notation used for complexity which drops all coefficients

Informed/Uninformed Search: Algorithm which does/doesn't use a heuristic.

Heuristic: Technique designed for solving a problem more quickly when classic methods are invalid/too slow

BACKGROUND INFORMATION

Algorithms used:

- A*
- Bidirectional
- BFS
- Dijkstra's
- DFS
- Greedy Best-First Search
- Ant Colony Optimization (removed)
- IDA* (removed

- Ant Colony Optimization and IDA* were removed.
- Ant Colony Optimization couldn't find a correct path.
- IDA* used recursion.

In Python, the max recursion limit is 1000 (for a good reason). When the graph has almost that number or above, Python crashes. (This is to prevent Python from using up infinite RAM).

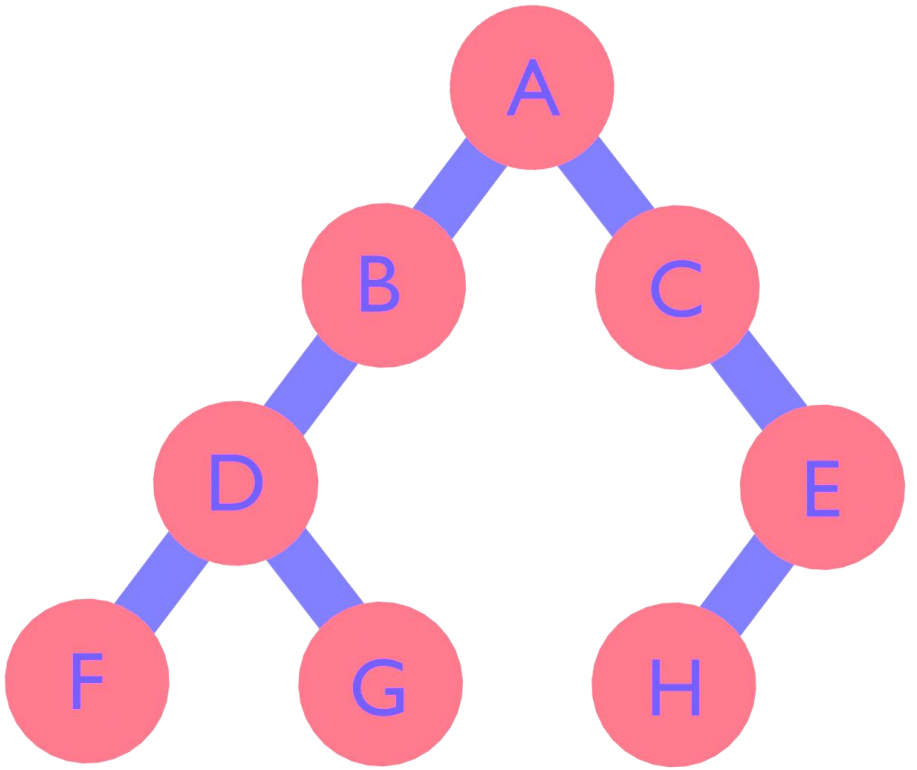
BACKGROUND INFORMATION

Depth-First Search (DFS, recursive):

Starts at node N, go up 1 level if it's a leaf node. Then run DFS on all the child nodes. This causes DFS to go all the way to the bottom of the tree before going to the next sibling.

Breadth-First Search (BFS):

Create a queue that has the source node. Pop the first node from the queue and add its successors into the queue. Repeat while destination isn't found, or queue still has elements. This causes BFS to finish the first layer before going onto the next layer in a tree.



BACKGROUND INFORMATION

Variables:

- Independent:
  - Sorting Algorithm
- Dependent:
  - Time taken
  - Cost of the path
- Constants:
  - Graph used for each trial.

HYPOTHESIS STATEMENT

- If the researcher searched for the shortest path between two randomly chosen nodes, BFS will have the fastest time compared to any other search algorithm.
- According to research, BFS has an algorithmic complexity of O(V+E), which is higher than Dijkstra's, O(V^2) and Bidirectional, O(b^(d/2)). BFS also doesn't use a costly heuristic, which Greedy and A* use. Also, since the graphs will be large, the distance to the goal node may be very high, so the BFS algorithm is very likely to outcompete DFS (since it goes breadth-first).

MATERIALS

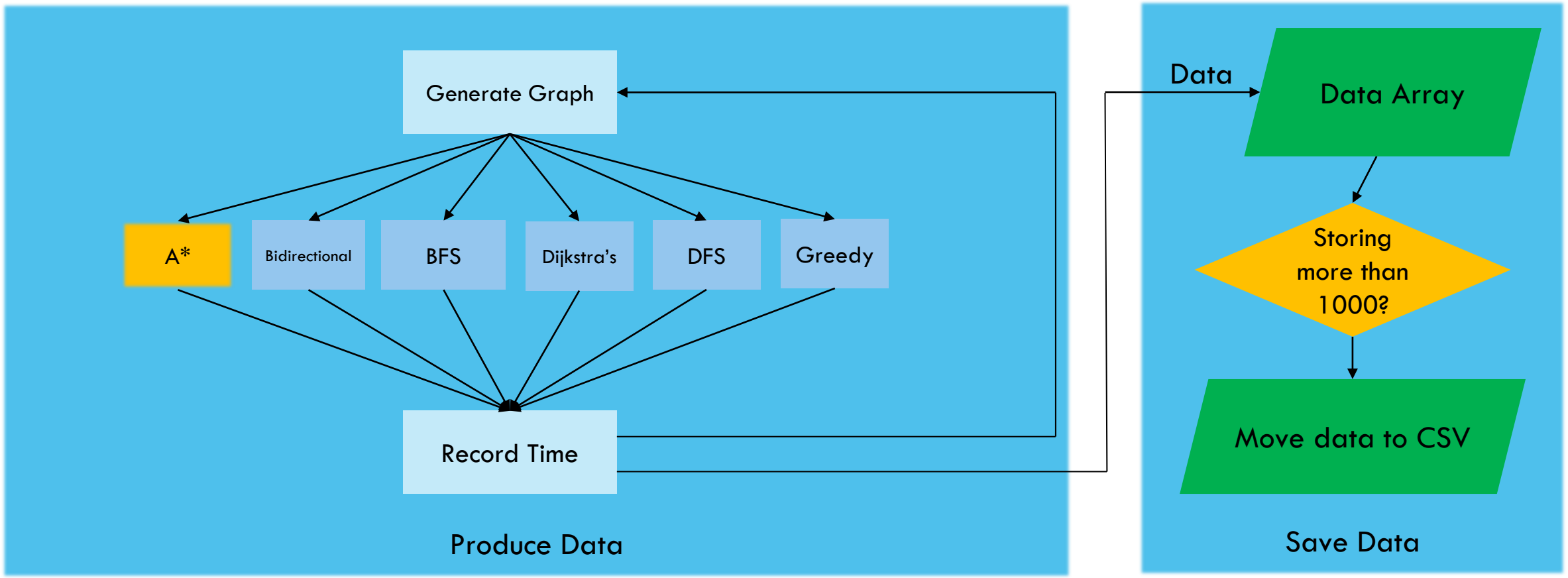
The code was altered to be compatible with how I stored my graphs.

1. A* search, www.geeksforgeeks.org/a-search-algorithm/
2. Bidirectional search, www.geeksforgeeks.org/bidirectional-search/
3. BFS, www.geeksforgeeks.org/breadth-first-search-or-bfs-for-a-graph/
4. Dijkstra's Algorithm, www.geeksforgeeks.org/dijkstras-shortest-path-algorithm-greedy-algo-7
5. DFS, www.geeksforgeeks.org/depth-first-search-or-dfs-for-a-graph/
6. Greedy Search, www.programiz.com/dsa/greedy-algorithm

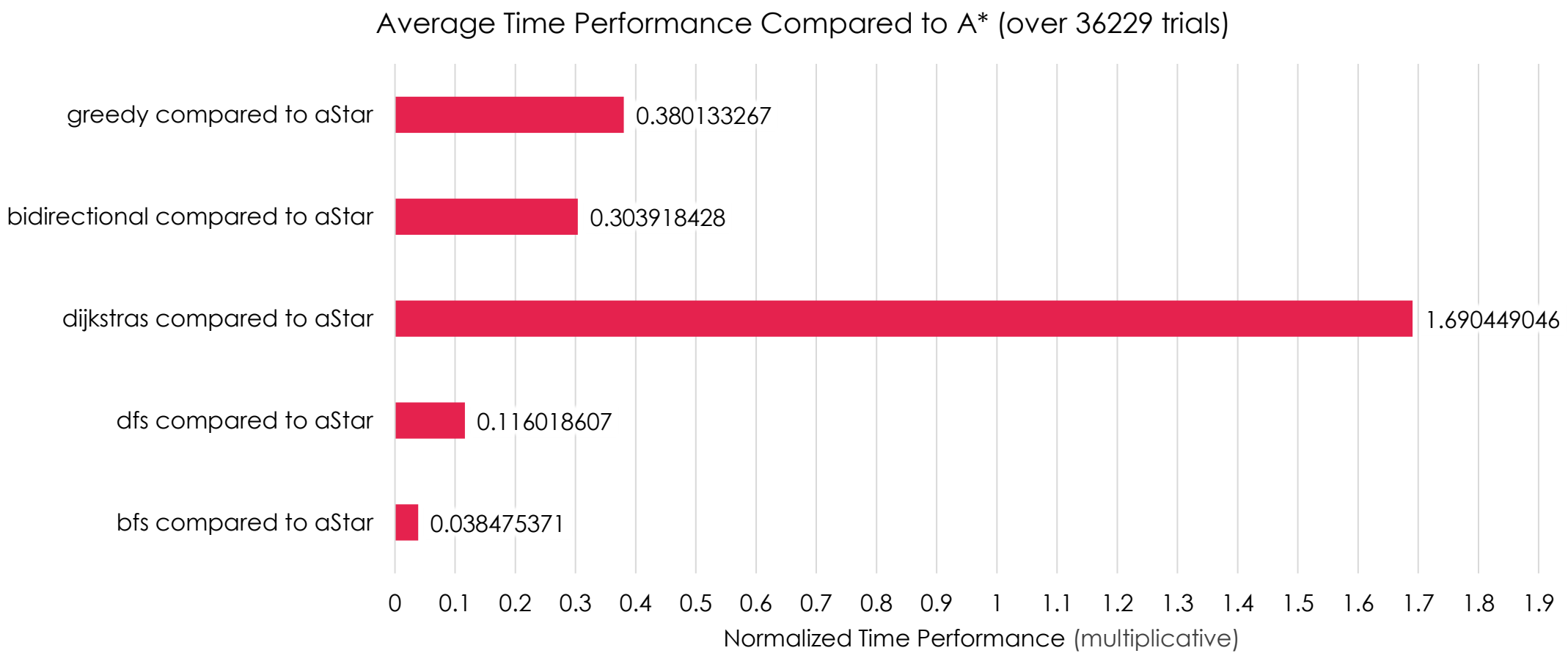
PROCEDURE

1. Generate random graph with 1024 random vertices on a 400x400 plane, where vertices within 24 length between them are connected.
2. Run algorithm on graph and measure time and cost. (-1 for invalid paths)
3. Dump data to csv
4. Repeat steps 2-3 for each algorithm
5. Repeat steps 1-4 for each trial (36229 trials in total)
6. Calculate average times and costs for each algorithm
7. Create bar graph for each time and cost.

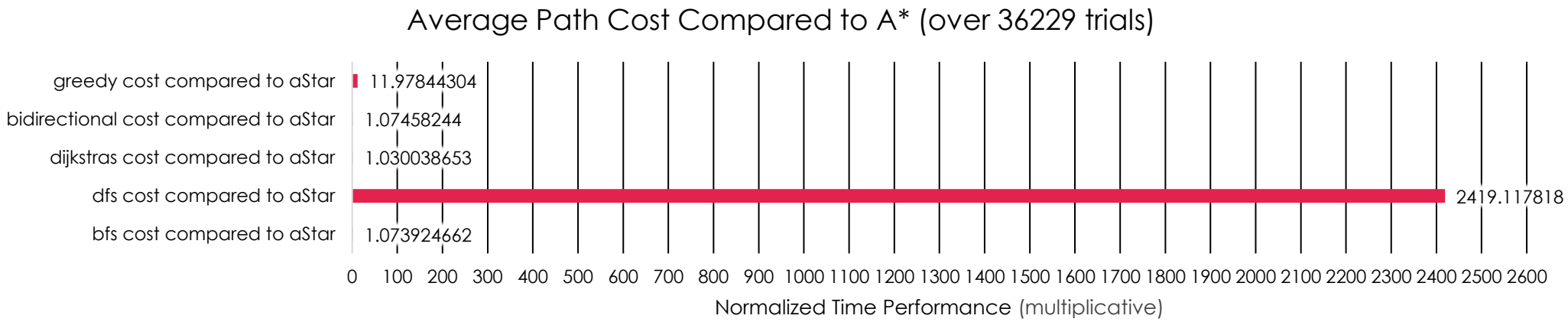
HIGH-LEVEL DIAGRAM

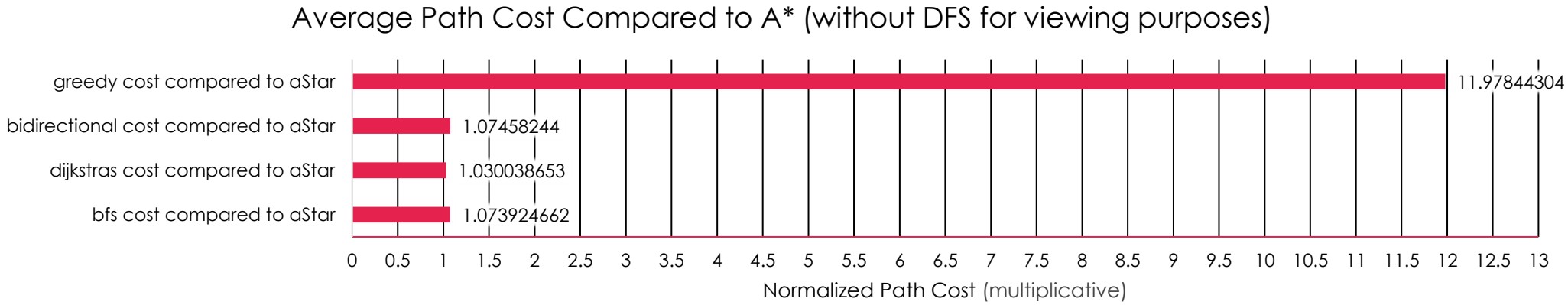


TIME COMPARISON CHART

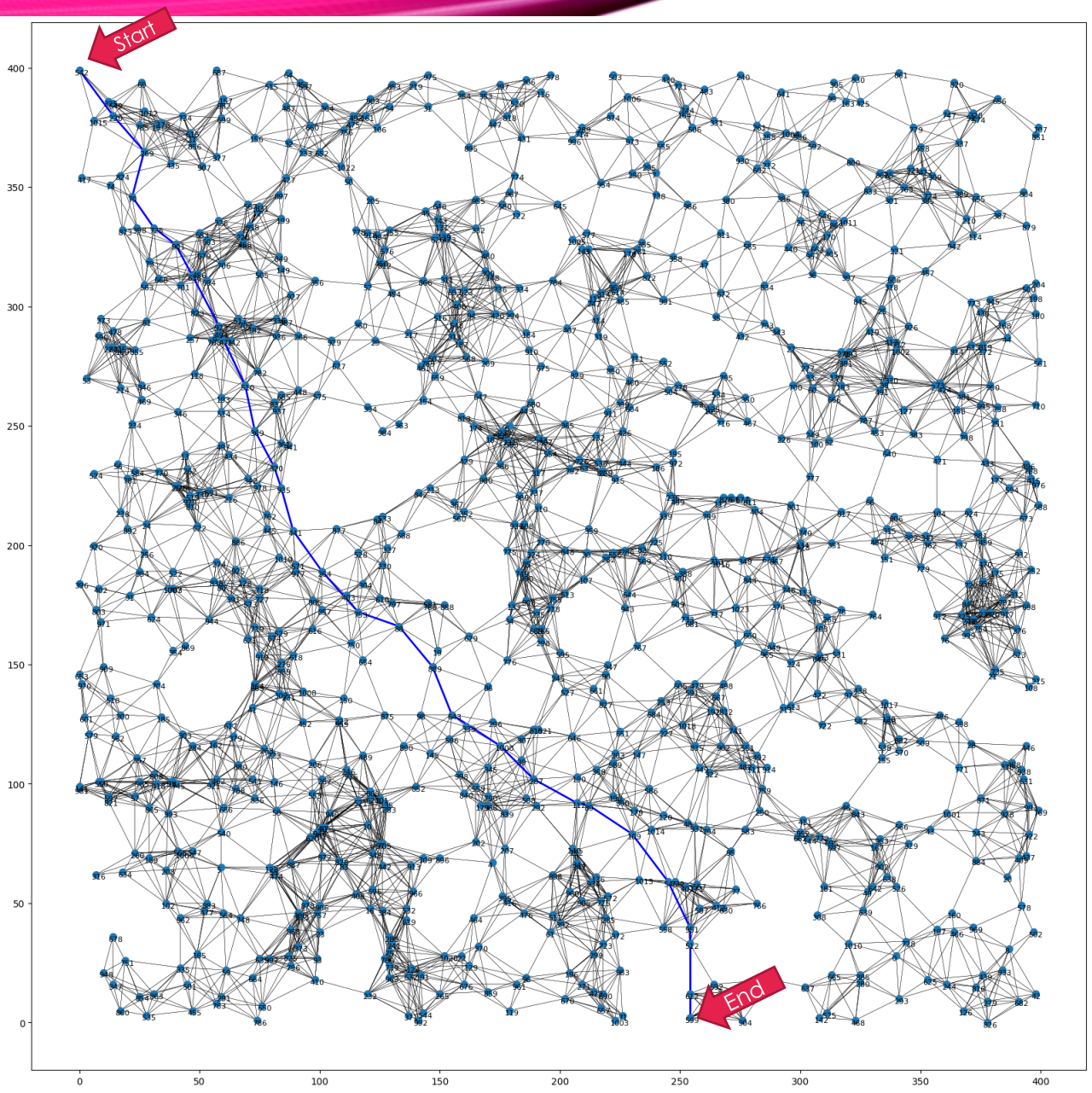


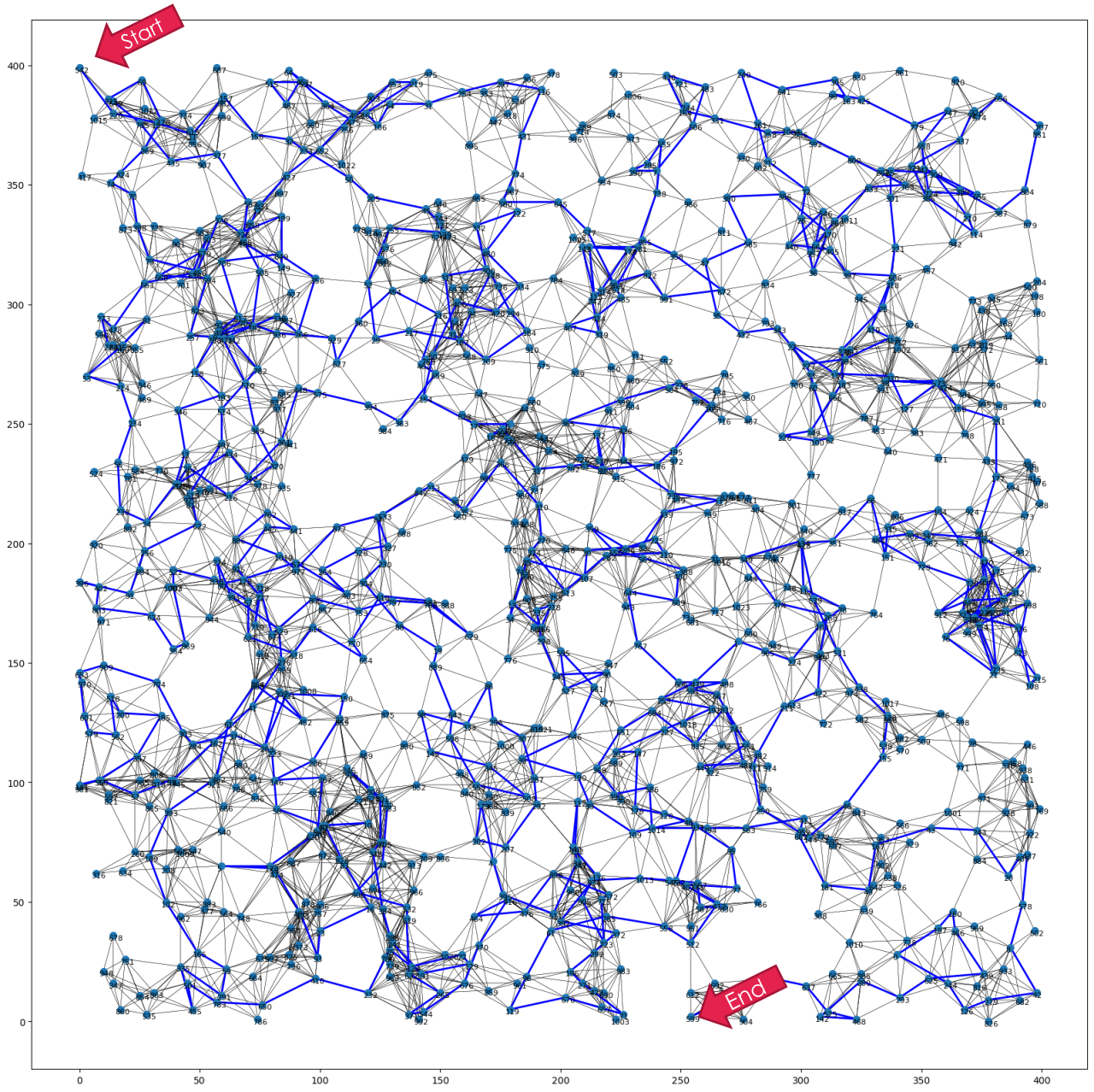
DATA CHARTS


Average Path Cost Compared to A* (over 36229 trials)
greedy cost compared to aStar
bidirectional cost compared to aStar
dijkstras cost compared to aStar
dfs cost compared to aStar
bfs cost compared to aStar
11.97844304
1.07458244
1.030038653
2419.117818
1.073924662
0 100 200 300 400 500 600 700 800 900 1000 1100 1200 1300 1400 1500 1600 1700 1800 1900 2000 2100 2200 2300 2400 2500 2600
Normalized Time Performance (multiplicative)


Average Path Cost Compared to A* (without DFS for viewing purposes)
greedy cost compared to aStar
bidirectional cost compared to aStar
dijkstras cost compared to aStar
bfs cost compared to aStar
11.97844304
1.07458244
1.030038653
1.073924662
0 0.5 1 1.5 2 2.5 3 3.5 4 4.5 5 5.5 6 6.5 7 7.5 8 8.5 9 9.5 10 10.5 11 11.5 12 12.5 13
Normalized Path Cost (multiplicative)

EXAMPLE GRAPHS





CONCLUSION

- The research question was, “What is the fastest graph searching algorithm?”
- The hypothesis was: If the researcher searched for the shortest path between two randomly chosen nodes, BFS will have the fastest time compared to any other search algorithm.
- According to research, BFS has the best factors compared to the other algorithms, in time complexity and time needed to calculate the heuristic (which BFS doesn’t have).

CONCLUSION

As seen in the normalized time comparison chart, BFS had a ratio of about 0.03:1 which is the lowest out of all the algorithms. **The researcher's hypothesis is supported by the data.**

- BFS doesn't use heuristics, which makes it very fast. Bidirectional, DFS and Dijkstra's are the only other ones not to use a heuristic. The heuristic involves taking a square root, which is an expensive operation.
- BFS has the lowest complexity, of O(V+E)

CONCLUSIONS

Errors:

- The heuristic problem can easily be fixed by not using the square root. However, there is a possibility that this may make no effect, as the heuristic is run with a super efficient module (NumPy).
- Another error is that since the code runs so fast, the times recorded may be off by some value. If this code is run on a device with less processing power (e.g., Raspberry Pi), those margins make less of an effect. Choosing higher node counts is harder, because trial and error is required to get the minimum length that keeps all the nodes connected, but at the same time not connecting too many to create an easy path.

CONCLUSION

- Pathfinding is very important in the area of AI. In this case, it is being used with physical paths, however there may be other items that are represented through graphs, such as a game tree, which is used in gamebots.
- The only downside is that such heuristics may not be applicable, reducing the effectiveness of informed (heuristic-based) searches.
- Google Maps, which is a very common application used to find routes to get from a source to a destination uses A*. Many other GPSs use A*.

SUGGESTIONS

One area for improvement is the coding language used. Python is an interpreted language, meaning that the code is read, then changed to some form which computers understand, and then executed.

A great alternative to Python is C++. C++ is a language that is compiled, meaning that the code is read, then changed to the computer form, but the main difference is that this is before the code is even ran! The code must only be compiled once to be ran many times. The machine code is saved, giving it much more speed than Python.

The only problem with C++ is how long it would take to develop the code. In Python it took about 10-15 hours, but in C++ it would likely 2-3x as much.

ADDITIONAL LINKS

GitHub Repo: https://github.com/python9160/science-fair-2223

main.py: https://github.com/python9160/science-fair-2223/blob/main/main.py

THANK YOU

BIBLIOGRAPHY

Works Cited

Belwariar, Rachit. "A* Search Algorithm." GeeksforGeeks, Sandeep Jain, 30 May 2022, www.geeksforgeeks.org/a-search-algorithm/. Accessed 7 Feb. 2023.

Depth First Search or DFS for a Graph. Sandeep Jain, 26 Dec. 2022, www.geeksforgeeks.org/depth-first-search-or-dfs-for-a-graph/. Accessed 7 Feb. 2023.

GeeksforGeeks, and Divyanshu Mehta. "Dijkstra's Shortest Path Algorithm | Greedy Algo-7." Edited by Pranav Singh Sambyal. GeeksforGeeks, Sandeep Jain, 27 Jan. 2023, www.geeksforgeeks.org/dijkstras-shortest-path-algorithm-greedy-algo-7/. Accessed 7 Feb. 2023.

GeeksforGeeks, and Neelam Pandey. "Iterative Deepening Search(IDS) or Iterative Deepening Depth First Search(IDDFS)." GeeksforGeeks, Sandeep Jain, 26 July 2022, www.geeksforgeeks.org/iterative-deepening-searchids-iterative-deepening-depth-first-searchiddfs/. Accessed 7 Feb. 2023.

GeeksforGeeks, et al. "Breadth First Search or BFS for a Graph." GeeksforGeeks, Sandeep Jain, 19 Jan. 2023, www.geeksforgeeks.org/breadth-first-search-or-bfs-for-a-graph/. Accessed 7 Feb. 2023.

Kumar, Atul. "Bidirectional Search." GeeksforGeeks, Sandeep Jain, www.geeksforgeeks.org/bidirectional-search/.

PratikBasu. "Introduction to Ant Colony Optimization." GeeksforGeeks, Sandeep Jain, 17 May 2020, www.geeksforgeeks.org/introduction-to-ant-colony-optimization/. Accessed 7 Feb. 2023.

Programiz. "Greedy Algorithm." Programiz, edited by Ranjit Bhatta, www.programiz.com/dsa/greedy-algorithm. Accessed 7 Feb. 2023.