- Administrative
    - Jason's fantastic team
    - Jason Nguyen (github - jasonneyugn, Joseph Cardillo, Diksha Kushwa
    - https://github.com/jasonneyugn/project2cop3530.git
    - https://youtu.be/HQejVx-3P8o
      Jason
- Extended and Refined Proposal [Suggested 2 Pages]
    - Problem: The goal of this project was to compare and visualize the shortest path problem on a very large dataset, in this case the global network of airports. The problem we originally had was to just find the shortest distance from point A to point B, but that was changed to connected airports. We also originally had just Dijkstra's algorithm, which is very costly on larger datasets, so we contrasted this with the A* algorithm that uses heuristics.
    - Motivation: This is a very modern and important problem in computer science with things focused around GPS and routing. This is where shortest path algorithms shine because the real world is much more convoluted than any simulated data. By using the dataset that consists of every airport it helps us visualize the algorithms and the way they work. This is one very important part of the project, being able to visualize these processes because the allow us to see how heuristics affect runtime.
    - Features implemented -
      Interactive world map using SFML
      Mercator projection of parsed data of map lat/lon into 2D coordinates.
      Randomized edge generation to connect graph
      Zooming and panning
      Path visualization (green lines for shortest path)
      Dynamic runtime execution and analysis
      Clickable airports with displayed routes
    - Description of data
      The data used was the airports.dat and routes.dat from the OpenFlights database. It includes over 10,000 airports with several tens of thousands of routes, which is why we had to include some more random ones. I had to parse the files for their ID, name, lat/log etc and each route/airport was visualized on the map of the project. During the implementation of this "graph" map, the graph was obviously sparse because some nodes did not have any routes, so we had to randomize some edges and create new ones. Using the haversine formula, I ensured each graph was connected at least once.
    - Tools/Languages/APIs/Libraries used
      Language: C++ (20)
      Graphics Library: SFML

    - Algorithms implemented
      Implemented Dijkstra's Algorithm and A* search algorithm. The A* search uses a heuristic, which helps the short path run faster than the Dijkstra. This is because the Dijkstra visits more nodes its search and the prioritization of the heuristic helps the A* be much more efficient.
    - Distribution of Responsibility and Roles: Diksha - zilch, Jason - coding, video, github, report (extended proposal, analysis) , idea,
    - Jason
- Analysis [Suggested 1.5 Pages]
    - Any changes the group made after the proposal? The rationale behind the changes.
      There was a lot of changes that were required to make the idea work without fully changing the base idea and breaking the rules. If we didn't generate any new edges, the graph would be sparse and unconnected, not allowing us to test any paths out. I also did not have any other algorithm planned out with Dijkstra's, only a rough idea of testing some others. However, with the A* search it is a good contrast to the other algorithm because it is much more efficient in the way it travels. I also had to add features such as zooming and dragging, which were not originally planned to make interacting with the map much easier in general.
    - Big O worst case time complexity analysis of the major functions/features you implemented
      The major functions include the routeload function, the Dijkstra algorithm, the A* search algorithm, and the haversine formula. The haversine formula is a constant formula with trig so it's just O(1). To parse the routing files, we just had to go through every single part, so O(N) time complexity where n is every piece of data. For the main algorithms, the Dijkstra and A* algorithms are actually both O((V+E)logV) time complexity.
      Jason
- Reflection [Suggested 1-1.5 Page]
- joseph
    - As a group, how was the overall experience for the project?

As a group our experience was engaging and extremely beneficial for learning the process of version control and group oriented programming. This project required the application of several different areas of computer science like data structures, algorithms, and graphics, all combined into a single practical application. Seeing the algorithms in use and working together like Dijkstra's and A* in real time gave us a satisfying sense of accomplishment. While we had many issues with debugging and version control, we felt our skills have been strengthened and that we are leaving this project more confident for future collaborative efforts.

One of the largest challenges our group faced was regarding the complexity of the OpenFlights datasets. We had initially assumed them to require less logic for parsing and did not

notice how careful we needed to be to avoid incorrect behaviors or results. Another issue was the SFML for visualization because in my experience the user interface for applications can often be challenging especially when trying to make something so complex user friendly. One hurdle we had noticed about debugging the logic regarding the latitude and longitude was that the pixel coordinate project proved to be more challenging than expected. The math used seemed simple in theory but became increasingly complex due to small round errors. Additionally, ensuring that the graph remained connected through any route generations was frustrating and sometimes difficult to debug.

If we were to start the project again we would definitely start by planning our workflow more clearly. If we had started by breaking it down into sections then from there using a drawing or some kind of workflow software we could have potentially cut down on time used coding and spent that time more efficiently working on other parts of the project. Another issue with defining our architecture as we go was that it led to multiple redesigns costing much needed time and resources. Also the version control practices we had used could have been more effective with better comments, more commits, more frequent review etc.. This is something that I especially struggled with because I have previously created messes of branches and merges that caused me to have to restart lots of code, but this time we had made sure to avoid this problem but unfortunately this shows by leaning towards the opposite side of not making enough small frequent commits. The comments made with each commit could have been slightly improved on top of that. More importantly the comments in our code could benefit from more explicit explanations. Besides these issues our group would have greatly benefited from some improved communication and could have started earlier to avoid the time crunch.

    - Did you have any challenges? If so, describe.
      My main challenge was the restrictions of the first Project 2a. We were not allowed to change the general idea of the project we were going to do, but the dataset chosen wasn't big enough and we only chose one algorithm.
    - If you were to start once again as a group, any changes you would make to the project and/or workflow?
      I would divide the work more specifically so group members would not be able to slack off. I would also probably work alone.
    - Comment on what each of the members learned through this process.
      Jason - During the entire process, it was evident how important understanding key ideas is. As I was coding the first quarter of the project, it was taking very long because I couldn't figure out the algorithm. However, once I implemented the first lines and the Dijkstra algorithm, it was so easy to do the A* algorithm as well generate new edges.
- References

https://www.geeksforgeeks.org/cpp/c-program-for-dijkstras-shortest-path-algorithm-greedy-algo-7/

https://www.geeksforgeeks.org/dsa/haversine-formula-to-find-distance-between-two-points-on-a-sphere/
https://openflights.org/data.html