Week 0 Work Summary: 11/22/2020

All:

All of us together this week worked on actually writing up our team contract and project goals. We also individually researched and came together to a decision to use the Reddit dataset for our project.

We decided that we could take advantage of the sentiment data provided in the dataset. Each hyperlink from subreddit to target subreddit had a 1 or -1 (positive/negative sentiment). We started looking at ways we could take this data and convert it into a proper edge weight. We also started to look at how we could take this dataset and create a single edge, connected and directed graph.

Adish:
Started considering the structure of the data (subreddits connected to target subreddits). Then started researching the implementation of both BFS and DFS on this graph. Furthermore, I started looking into using the BFS traversal to find the shortest path between 2 subreddits within our graph.

Mike:
Researched the implementations of the strongly connected components. This includes both Tarjan's and Kosaraju's algorithm. Also looked into how to read in data in c++ from a csv file.

John:
Regarding the uncovered algorithms needed for our project, I researched how we can implement either the IDS or forced-directed graph drawing algorithms for our dataset. Furthermore, I looked into possible ways to parse the csv file in python since this part was not required to be done in c++.

Yash:
Researched how we can take a tsv dataset file and convert it to a csv file using python. Then looked into the shortest path implementations that we could use. This includes the covered algorithms from lecture (Dijkstras and Floyd-Warshalls).

Week 1 Work Summary: 11/29/2020

Adish:
My first job this week was creating the initial structure of our entire project directory. This meant creating the basic Makefile, importing all the given cs225 code for PNG, Graph and Edge, and finally creating our reddit projects' cpp and h file. I then worked on creating the initial BFS traversal implementation for our graph using what I had learned from the lecture. This was later altered and refined for the purpose of our project.

Mike:
I changed the Graph.cpp and Graph.h files to remove any unnecessary functions that were relevant to the lab but not our project. I started implementing an algorithm to find all the strongly connected components in a graph. My implementation will be using Tarjan's Algorithm as it requires traversing through the graph 1 time with DFS rather than 2 times when using Kosaraju's Algorithm. The runtime however, comes at the cost of using more memory.

Yash:
I wrote the reddit class constructor and a BFS implementation that returns the shortest path between two given vertices. I also collaborated on the iterative deepening search function. I wrote Kosaraju's implementation of Strongly Connected Components - required additional DFS utility and modification of constructor to create graph transpose and graph simultaneously.

John:
I looked into the tree traversal methods we covered in class. Using the idea of BFS & DFS, I started writing the algorithm for iterative deepening search that we chose to implement for one of the uncovered topics. This algorithm is very similar to DFS but has a limit to how deep we can traverse the tree, so it helps find if certain nodes are closely related or not.

Week 2 Work Summary: 12/06/2020

Adish:
This week I worked on a variety of things. My main priority was first creating test case graphs so each of the other members could use them to test their functions. This meant creating a small, medium, and 2 large example graphs that mimicked our entire dataset. I then taught the group how to actually write and test their code. I created the test cases for the BFS function as well. I then worked on trying to debug the BFS code in terms of some edge cases it was failing with 1 or no nodes in the graph. Finally, I started writing up the project report as well as started creating the final presentation for our deliverables.

Mike:
This week I fixed all the bugs that were in the StronglyCC function that was implemented with Tarjan's algorithm. I used the test cases that we came up with together to make sure that everything was working properly and had no memory problems.

Yash: Collaborated on creating 4 test CSV files for unit testing class functions. Wrote test cases for Kosaraju's SCC function. Wrote test cases for constructor - fixed minor error (double insertion of last edge from csv file) in constructor as a result of failed test. Fixed errors with IDS function caused by marking vertex as visited prematurely. Wrote python code for SCCs sanity check. Improved runtime for Kosaraju's SCCs, BFS, and IDS by making small changes in how vertices are stored temporarily - Using a vector as a stack for SCC, not

reversing the resultant path vector for BFS, and inserting vertices to path vector when path is found in IDS - cutting push/pop calls to vector by half. Added path printing functionality, prints paths of BFS and IDS functions in reverse as they return paths in reverse order. Added path sentiment calculator function - works almost identically to path print function.

John:

For this week, I added a public function DFS so that it could be used to traverse the entire given dataset, and renamed the DFS that was being used as a helper function for SCC. After adding the DFS function, I made a test case using a small dataset (small.csv) that we created and compared the output to the expected solution and made sure the function was correct.