Kevin Zhou, Beth Fineberg
Cale Wolf, Evan Matthews

Spotigraph Project Report

Spotigraph is an application that can make connections between artists on Spotify based on songs that they have collaborated on. We have created a graphical representation of a subset of all artists on Spotify, with the vertices of our graph representing artists and the edges of our graph representing a song that two artists worked on. When acquiring our data, we wanted to be able to directly access the Spotify API and use all artists on the platform. However, because of grading restrictions, we needed to create a subset of those artists as a CSV file, and we chose to do so using a python script. There are some limitations, as our script makes no distinction between popular artists and artists that have less than 1,000 followers. For this reason, we have a large collection of uncommon artists while not having big stars, (Taylor Swift, for example,) in our subset.

We used BFS to traverse a given graph, creating a web outwards from a specific artist so that we can find a specific artist in O(n) time. However, since we used a hashmap implementation to store our vertices, we can find a specific vertex in O(1) time. We used this to our advantage in our Dijkstra's algorithm implementation, where our edge weights for our songs are the song length. Spotigraph can create a playlist of songs going from one artist to another in the shortest listening time. Throughout our testing with various artists, it seems that the artists existent in our subset and Spotify are very interconnected. We have not found a playlist from one artist to another that is longer than four songs. This may change if we have more artists in our data set, but with our CSV containing over 6,500 artists, we thought that it would be a fair representation of a section of Spotify. For our landmark path, our goal was to see how interesting the connections could get. Our favorite connection is Jacob Sartorius through Jacob Collier to

Ariana Grande. We chose to use Dijkstra's algorithm as our shortest path algorithm since it made sense for us to listen to the least amount of Jacob Sartorius's music as possible.

Through this entire project, we have learned much about interacting with APIs, even if that was not the main goal of this project. We found it to be unfortunate that we could not use the entire Spotify API by directly making calls, but also quite thankful because we found cURL with C++ to be cumbersome. The parsing of the CSV file created by the python script was likely easier than trying to create a graph implementation solely through API calls. We also learned the value of working with a team through GitHub and how to coordinate branches and version control. Pair programming was valuable to members of our team because of the way we would catch each other's bugs. With learning related to graphs, we are very proud of our implementations of these algorithms and how fast they can perform. C++ has multiple data structures that we used in tandem to our advantage, which was difficult at first but manageable with our previous experience in the class. We hope that, in tandem with our creative concept, we will be able to use our experience from this project to continue it outside of CS 225.