COMMONGROUND: CS32 FINAL PROJECT

Background

CommonGround was created to automatically generate playlists for groups of friends. Imagine being able to easily and quickly create a playlist of songs that everyone in a group would enjoy, from your family in the car to your friends at a party to your computer science final project group at your house.

To this end, we created our own similarity algorithm using metrics from Spotify, LastFm, GraceNote, the Billboard Hot 100, DigitalDreamDoor, and track production data. We also extensively user tested our webapp to make sure it satisfied our users and made adjustments based on their feedback.

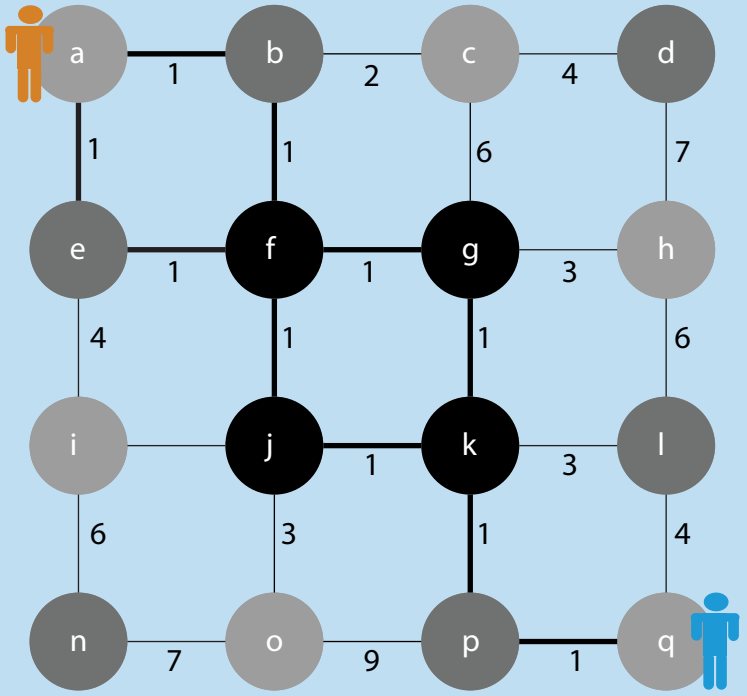
Building Up the Program

Populating the Database

We built our own database of music from scratch. Initially, we populated it with Billboard's Hot 100 songs from every year since 1960 and DDD's top 100 artists in every genre. We then used Spotify to gather the top songs for these artists. Then, using Spotify, LastFm, and Gracenote, we amassed similar songs to the songs in our database.

Algorithm Pseudocode

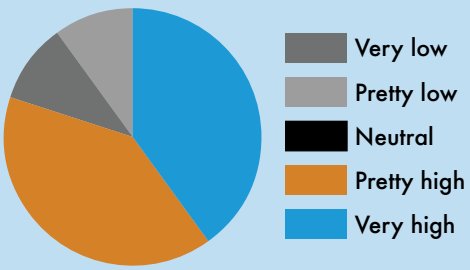
1. Find the songs that are most popular among the group of users.
2. Use these songs to generate similar songs by grouping them into genres and:
   a. Run Dijkstra's algorithm for each song pair.
   b. Take the center songs of each Dijkstra's path.
   c. If there are few enough songs, take other songs along the path.
   d. Recursively call the method a random number of times, collecting the centers of these paths.
3. Combine the correct number of known and unknown songs to create a playlist, ensuring no repeat songs and various artists.
4. If there is not enough user data to work with, populate the playlist with a variety of songs in the selected genres.



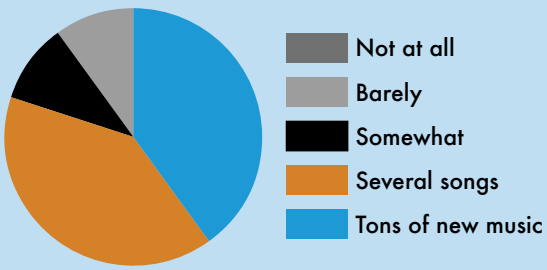
User 1 enjoys song "a" and User 2 enjoys song "q". The lines between the songs represent "similarities" between them, so for example, song "p" is similar to songs "o", "k", and "q". Using these similarities, we can form graphs between songs. Each similarity has an edge weight, with more similar songs having a lower edge weight, so we can find the shortest path between nodes. By just looking at the graph, we can intuit that User 1 and User 2 would likely enjoy songs "f", "g", "j", and "k". Using Dijkstra's algorithm (with various paths **bolded**), we can take the center nodes from these paths to find songs that users would both enjoy. Our algorithm, entitled **Center of Mass**, performs this calculation recursively, taking the centers of the paths and calling the algorithm on these nodes repeatedly.

User Testing Results

Overall playlist enjoyment:



Helped discover new music:



Changes made:

- Clarity regarding "familiarity"
- Easier navigation of genres, with ability to search by event
- Minimized repetition of artists in playlists
- Visual aesthetics sharpened
- Redistributed weight of selected genres to ensure that every genre is represented in a playlist

Developers: Calder Hoover, Eric Kong, Keiichiro Kinoshita, and Lena Renshaw. Special thanks to our mentor TA Rafael Reyes, Professor John Janotti, and the Head TAs.