CS 182 Final Project Proposal
Vivek Jayaram, Samarth Singal, Saroj Kandel

Our goal is to build an automated DJ program that can create a mix of several songs. The ability to mix and mash songs together is highly desirable by listeners and is a feature of live parties, radio shows, and online recordings. While the ability to create these mashups takes a certain degree of human intelligence and creativity, the basic properties of good mashups are well defined and easily automated. A high level description of the project is as follows: We will have a song collection of *n* songs and look at the how well each song matches with each other. Then the algorithm will create the best mix of *m* songs where n and m can change.

The song library
The song library will consist of songs manually entered by the user. Ideally this would contain all songs in the universe, but due to copyright and performance concerns, this project will use several hundred songs that are generally used for party/DJ music. Each song will be manually marked with a mix in point and a mix out point. This will be stored in the metadata as a millisecond offset from the start. Ideally finding such points could be done automatically, but such signal processing is out of the scope of this course and would involve heavy machine learning. Another simplification is that songs usually have multiple points to mix out and mix in well, but for now we will just use a single point for each.

Pairwise matchups
The idea of this is to use a data structure similar to a factor graph in a constraint satisfaction problem. Each song has a 2 factors with all other songs. One factor represents how well the mix out point from song 1 matches with the mix in point of song 2, and the second factor is the mix out from song 2 with the mix in of song 1. This will be pre computed and stored. Unary factors will also be used to represent how good a single song is. It doesn't matter if there is a great sounding mix if it contains all songs that people don't want to hear. The mix should ideally contain "bangers" or popular songs, which will be taken into consideration with the weight of unary factors.

Determining how well songs mash
This will not be a major concern of the project due to the complicated nature of such a task. However this paper http://telecom.inescporto.pt/~mdavies/pdfs/DaviesEtAl13-ismir.pdf outlines a basic procedure for determining how well two songs mash together. In addition factors such as BPM and Key will be taken into consideration.

Creating the best mashup
Once the pairwise and unary factors are computed, the goal is to create the mashup that maximizes the pairwise and unary scores. While the state space can be quite large, the idea is to use a dynamic programming algorithm such as the Viterbi algorithm to find the optimal path. The details of the algorithm are here:

http://www.systems.caltech.edu/EE/Courses/EE127/EE127A/handout/ForneyViterbi.pdf This problem is particularly well suited to the Viterbi algorithm due to the statelessness nature of music; The song that mixes well next doesn't depend on songs you played earlier but only the current song.

Other features
After determining the best sequence of songs, some part of the program actually needs to play the songs. Because the songs are rarely exactly the same BPM, some amount of tempo shifting and alignment is necessary. There exist many libraries for tempo shifting and since we are given the millisecond offset of the mix-in and mix-out, it is easy to match the tempo and start playing them in sync.

Division of the work
This has not yet been decided but we will most likely split up the parts that are easily parallelizable. For example one person can work on pairwise mashability, another can work on the Viterbi algorithm, and another can work on the data structures and libraries used to represent and play back such a mashup.