ECE250 Project 1 Design Document- Vectors

Devon Miller-Junk 20763552 September 26, 2019

Project 1: Vectors

For the design of project 1, my plan was to use the vector data structure to make a vector of songs called a playlist. Each song would have a name and an artist. The playlist class would allow for certain operations to be performed on the vector of songs. This same data structure can be applied to a variety of different scenarios using playlists, and these functions can be replicated when using a vector of another data type instead of a song.

**Note:** To compile and run my program please use the following:

g++ -std=c++11 -o playlisttest playlisttest.cpp

./ playlisttest <inputFile >outputFile

1. **Song Struct**

The song struct I created contains two strings: one that contains the name of the song and the other that contains the name of the artist. I chose to separate the name of the song from the name of the artist in the design of the song struct to allow for easy separating between the two for if a programmer were to use this same class in the future.

The song struct has a copy constructor, a null constructor, and a constructor where you can pass in the values of all the variables. I created all these constructors since this allows for easy creation of whichever type of song the user would like. These constructors take time proportional to the time it takes to copy a string variable type, so proportional to the length of the string, which in the case of the null constructor is 0, so constant time. I did not overload the default destructor of the song struct since none is needed.

I also overloaded the equality and the equals operators for the song struct. The equality operator returns true if the songs have the same name and artist, otherwise it returns false. The equals operator makes the artist and name of the current song equal to that of the parameter song. These operations both take time dependant on the length of the strings (to compare the strings and to equate the strings), which is proportional to the length of the strings.

1. **Playlist Class**

The playlist class contains a vector of songs and uses the vector of songs as a playlist of music. Since the vector is resizable, then this allows for no size to be initially set for the vector. Since no assumption can be made as to the final size of the vector, I do not initially resize the vector to an appropriate size. This saves memory at the cost of future processing time.

The playlist class only uses a null constructor since by default, a playlist will only ever be created empty. The destructor for the playlist simply clears the vector, so that it can be deleted safely. I also overload the equals operator to set two playlists equal. The equals operator clears the first vector and then equates the two vectors, which takes linear time based off the length of the second vector.

The playlist class also contains a push function, remove function, and play function. The push function appends a song to the end of the vector if that song is not currently in the vector. If the song is in the vector, then this process takes linear time to search through the vector to find the song that is already in the vector. If the song is not already contained in the vector, then it takes linear time to search for the song in the vector anyways and then takes time to append to the vector. There are 2 situations to consider when appending to a vector. If the vector has reached its capacity, then the entire vector must be copied over to a new place to match the greater capacity that it needs. This process takes linear time to copy over the vector. If the vector is not at capacity, then it takes constant time to append to the vector. This results in linear time to append since it takes linear time to do search either way. The remove function can also have two situations. If removing from the end, then it takes constant time to remove from the vector. However, if not at the end then by removing a space from the vector, there is now an empty space, meaning that every subsequent index must be copied over to remove the empty space. This takes linear time, resulting in linear time to remove. The play function simply accesses a song from a given index, which takes constant time.

1. **References**

Ward, P. (2018, Fall). *ECE 150 Lectures*. *ECE 150 Lectures*. Waterloo.