Process function:

For the process function, the first time is to input each word of every song into an array which has a linear time complexity of O(T), where T is the total number of words.

Then we run radix sort for the id using radix sort of each word this takes O(TN), where N is the largest id number. We do sorting for id first because we want the smaller id to appear first for the same word.

Next we do sorting alphabetically using radix sort which runs in O(TM) time where M is the length of the longest word.

Lastly we put the sorted words followed by their id into another textfile(sorted_words.txt), this take O(T) time

the total time complexity is T+TN+TM+T = 2T+TN+TM

In big O notation we drop the constants and the less significant terms , we end up with a runtime of O(TM)

Collate function:

For the collate function,the first time is to input each word of every song into an array which has a linear time

complexity of O(T), where n is the total number of words.

Then we compare each word to see if there is any repeating elements , if so we concatenate their id together. this

takes O(TM) time as they would compare each letter for each word.

Then we write it back into a textfile (collated_ids.txt), this takes a linear time of O(T).

the total time complexity is T+TM+T = 2T+TM

In big O notation we drop the constants and the less significant terms , we end up with a runtime of O(TM)

Lookup function:

For this lookup function we first append the query words into a array which takes time of O(q) where q is the number of words in the query file

Next we input the words from the collated_ids file into an array , this takes time of O(T)

Then for each query word is searched from the collated_ids using binary search, this takes

O(Mlog U) where U is the number of lines in collated_file, M is the length of the longest word in any song)

Then the id is written into a file, this takes time O(P), where P is the total number of IDs in the output

the total time complexity is T+(q x MlogU)+P

In big O notation we drop the constants and the less significant terms , we end up with a runtime of O(q x MlogU+P)