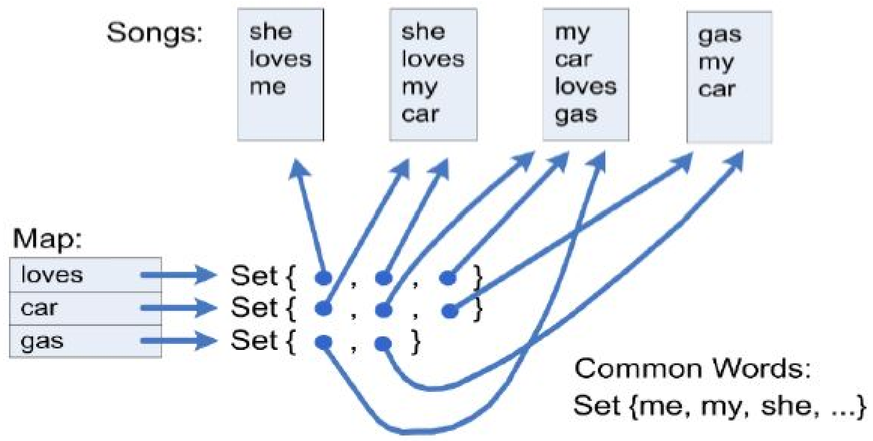
TreeMaps and TreeSets



Objectives: Using set operations.

In this programming project you will be building a compound data structure using sets and maps. The main data structure will be a map of individual lyrics words into sets of the songs that contain those words. A small example is shown above. In this example the word "loves" maps to the first 3 of the 4 songs. "Car" maps to the last 3 of the 4 songs, and "gas" maps to the last 2 songs.

To save on space, some very common words are ignored, and not placed in the map. The set of common words includes "me", "my", and "she", and thus those words were not included in the map. Parts 7 & 8 are going to be writing a new class named SearchByLyricsWords. In part 7 you are just writing the constructor that builds the data structures. In part 8 you will be writing the search method that uses this data structure.

Start the constructor for SearchByLyricsWords

- As with the other searches, the constructor will take a SongCollection as its argument and build the data structure.

Task 1: Building the set of common words

- In your constructor for SearchByLyricsWords create a set containing the common words. You can find a file containing this list of words in the course directory for Part7. You can choose how to incorporate this into your program. I recommend just putting the raw data as a string within your code, but if you choose instead to read in the data file, be sure to submit that data file with your program.

Task 2: Building the Map of Sets

- Create a TreeMap of Strings to TreeSets of Songs.
- And now for every unique word in every song, add that word and song to the map.
- The words will be the keys for the map, and the values in the map will be TreeSets of all the songs containing that word.
- Common words should be ignored.
- Also all words should be converted to lowercase and any punctuation or numbers should be ignored.
- Finally, do not include any single letters as search terms.
- For example the lyrics "We don't need no education" should be broken into the terms: "we", "don", "t", "need", "no" and "education". Of those terms "t" should be ignored because it is a single letter, and "we" and "no" should be ignored because they are in the set of common words. (Hint: I used `String.split("[^a-zA-Z]+")` to split up the lyrics string. The argument is a regular expression that specifies the delimiter. In this case any non-alphabetic characters are considered delimiters.) Regular expressions are an entire topic in themselves. Don't get creative with the regular expression (you can shoot yourself in the foot here).
- For testing, add a main() to your search class, similar to those in the previous assignments. As before the first command line argument is the name of the song file. For part 7, your testing routine should just create the song collection and then call your constructor for SearchByLyricsWords.

Task 3:Statistics Gathering & Analysis

- You are now going to add some statistics gathering code to verify it built the data structure correctly.
- Create a method statistics() and call it from main() after you build your data structure.
  1. Print the number of keys in the map (expect 31385)
  2. Iterate through your map and calculate and print the total number of Song references stored in all the sets in your map. This will be large! This value will be our data structure's size "N" because this is how many things our data structure is keeping track of for us.
  3. Based on the number of keys, calculate the space used just by the map.
  4. Based on the number of Song references, calculate the total space used by all the sets.
  5. Now based on these 2 values, calculate the total space used by our compound data structure.
  6. Finally, express the space usage as a function of N for this example.

What to turn in: Turn in your SearchByLyricsWords.java and your Part7.pdf. Your class must work with my base code.

Written part:

1. Using allSongs.txt, turn in your statistics and analysis from Task 3.
2. Any incomplete parts, or any known bugs in your code.

Electronic submission:

- Submit and your write up .pdf file and SearchByLyricsWords.java.
- Do not in any way combine, compress, zip, tar or jar your files!

Grading:

Set of common words & exclusion of common words from the map	30 points
Building the lyrics words map	40 points
Statistics and analysis from Task 3	30 points

Points may be deducted for inadequate commenting or poor style

I expect your programs to be properly commented and use good style. Points may be deducted for egregious disregard of these matters. Every method in every class must have complete documentation including authorship. A full an complete revision history should be at the top of every class file!

Extra credit: (20 points) Add an additional analysis method top10words() to your class that figures out the 10 most common words in your map. Add a command line argument "-top10words" that enables this extra analysis. Print your results. Explain how your code works. Finally analyze its asymptotic run time.