Project-3: `TrieMap`

Description

You will implement a data structure in your last project in this semester. We name this data stucture `TrieMap`. Essentially, it is a map, meaning that you will store key-value pairs. Also, your data structure should return the value that corresponds to a given key.

Tasks

- **Task-1:** Implement `TrieMap<T>` class by extending `TrieMapBase<T>` abstract class (Observe that you will also need to implement `Node<T>`). Here, T is the type of the keys to be stored.
- **Task-2:** Implement `public static boolean containsSubstr(String text, String key)`
  that returns true if `key` appears as a substring in `text` and false otherwise. You may assume that both strings are made up of a-z.
- **Task-3:** Implement `public static int wordCount(String book, String word)`
  that returns how many times `word` appears in the `book`. Each word in `book` is separated by a white space. You may assume that both strings are made up of a-z and white space.
- **Task-4:** Implement `public static String[] uniqueWords(String book)`
  that returns the unique words in a book. Each word in `book` is separated by a white space. You may assume that strings are made up of a-z.
- **Task-5:** Implement `public static String[][] autoComplete(String[] userHistory, String[] incompleteWords)`
  that suggests word completions based on the counts of the words written previously. More formally, given a string **S** of consecutive letters (**S** not containing any white space), autocomplete feature must recommend 3 most commonly written words that start with **S**. A very inefficient way of solving the problem is the following: Among all the strings in the user history, you will take those that start with S, sort them according to their frequencies (how many times they are written), and recommend the 3 most frequently written ones. But you will use Trie data structure to solve the problem efficiently. The first parameter above consists of the words written previously by the user. The second parameter is the list of strings your code will autocomplete. For each string in `incompleteWords`, you will recommend three words. Therefore, if the second parameter has length S, you will return a Sx3 array that contains the recommendations for each word to be autocompleted. You may assume that all strings are made up of a-z.

Notes

- **NOTICE:** When you ignore the values, TrieMap reduces to Trie. You will have to use that property.
- In TrieMap, keys are always strings, values must be generic.
- In TrieMap, N is not greater than 26 and not less than 2.
- In TrieMap, N and alphabet will always be consistent
- Do not change the class definition of TrieMap: `public class TrieMap<T> extends TrieMapBase<T>`
- You are expected to use Trie data structure in your solution to Task2-3-4-5. If not, you will get no credit.