1. A solution overview: use a block diagram an/or an algorithm you used.
2. State the space and time complexity of your solution.
3. A listing of the most important data structures you employed.

Most important data structures:

* HashMap - this data structure is what allowed me to map the sorted strings to all anagrams and quickly look up all anagrams

1. Sample input and outputs (as required above) in the form of a table.

Please note that for running this program, I used powerpuff and the associated dictionary.

|  |  |
| --- | --- |
| **Input** | **Output** |
| Anagrams for plekic: | pickle |
| Anagrams for diapers: | aspired, despair, diapers, praised |
| Anagrams for teardrop: | parroted, predator, prorated, teardrop |
| Anagrams for nameless: | lameness, maleness, nameless, salesmen |
| Anagrams for allergy: | allergy, gallery, largely, regally |
| Anagrams for deepak: | peaked |
| Anagrams for impressions: | impressions, permissions |
| Anagrams for restrain: | restrain, retrains, strainer, terrains, trainers |
| Anagrams for calligraphy: | calligraphy, graphically |
| Anagrams for nepal: | panel, penal, plane |
| Anagrams for stale: | least, slate, stale, steal, tales, teals |
| Anagrams for parliaments: | parliaments, paternalism |
| Anagrams for sucrose: | courses , sources, sucrose |
| Anagrams for persist: | persist, priests, spriest, sprites, stripes |
| Anagrams for disintegration: | disintegration, disorientating |

1. A discussion of the entire process.
2. A complete printout of the program(s) you wrote.

**Code:**

import java.io.BufferedReader;

import java.io.IOException;

import java.nio.charset.StandardCharsets;

import java.nio.file.FileSystems;

import java.nio.file.Files;

import java.nio.file.Path;

import java.util.ArrayList;

import java.util.HashMap;

public class Lab05 {

/\* Global variables \*/

public static HashMap<String, ArrayList<String>> dictionary = new HashMap<>();

public static int NUM\_CHARS = 256;

public static ArrayList<String> result = new ArrayList<>();

private static String testing[] = {"plekic", "diapers", "teardrop", "nameless", "allergy", "deepak", "impressions", "restrain", "calligraphy", "nepal", "stale", "parliaments", "sucrose", "persist", "disintegration"};

/\* Main method that runs algorithm \*/

public static void main(String args[]) {

//for all words, read word, sort, and insert into hashmap

readWords();

for(int i=0; i<testing.length; i++) {

//get a word and search for it

result = searchWord(testing[i]);

//print list of anagrams

if (result != null) {

printList(testing[i], result);

}

}

//exit/done

}

/\* Method that prints a row of the output table. \*/

private static void printList(String input, ArrayList<String> anagrams) {

System.out.print("Anagrams for " + input + ":\t\t\t");

for (int i = 0; i < anagrams.size(); i++) {

System.out.print(anagrams.get(i)+ "\t");

}

System.out.println();

}

/\* Method that searches for the word in the dictionary. \*/

public static ArrayList<String> searchWord(String word) {

ArrayList<String> returnList = dictionary.get(sortString(word));

return returnList;

}

/\* Method that reads from the file and loads the dictionary \*/

public static void readWords() {

Path path = FileSystems.getDefault().getPath("/usr/share/dict/", "words");

try {

BufferedReader reader = Files.newBufferedReader(path, StandardCharsets.ISO\_8859\_1);

String line;

String word;

String sorted;

while ((line = reader.readLine()) != null) {

word = line.trim();

sorted = sortString(word);

if (dictionary.containsKey(sorted)) {

ArrayList<String> anagrams = dictionary.get(sorted);

anagrams.add(line);

dictionary.put(sorted, anagrams);

} else {

ArrayList<String> anagrams = new ArrayList<>();

anagrams.add(line);

dictionary.put(sorted, anagrams);

}

}

reader.close();

} catch (IOException e) {

e.printStackTrace();

}

}

/\* Method that sorts the strings in alphabetical order. \*/

private static String sortString(String unsortedInput) {

int[] charCounts = new int[NUM\_CHARS]; //array to count the number of characters

StringBuilder sortedInput = new StringBuilder();

for (int i = 0; i < unsortedInput.length(); i++) {

charCounts[(int) unsortedInput.charAt(i)]++;

}

for (int i = 0; i < NUM\_CHARS; i++) {

for (int j = 0; j < charCounts[i]; j++) {

sortedInput.append((char) i);

}

}

return sortedInput.toString();

}

}