Lab 11 T9

Weichen Chai

Fall 2022

Introduction

In this assignment, we are asked to work with T9, which is an implementation of predictive texting usually used in cellphones. We will be looking into how to implement a Trie structure and the advantage and disadvantages.

Background

In order to implement T9, we use a sequence on numbers on a 9 digit numberpad, with each number holding three letters. When pressing the numbers in sequence, the program will make predictions for words that contain a variation of those letters.

Tasks

Trie

The Trie will contain Swedish special characters, removing q and w, but instead containing 'a' to '¨o', making it exactly 3 letters per number. The table looks something like the following:

The number on the left represents number on a keypad of an old fashioned phone, the letters n the right represents the letters that each number is associated with.

1 = a b c
2 = d e f
3 = g h i
4 = j k l
5 = m n o
6 = p r s
7 = t u v
8 = x y z
9 = a ¨a ¨o

The node given to us is the following, for each node we will offer 27 possibilities as there are 27 letters that we can choose from. Boolean word will be used later to indicate whether the word has ended, where if Boolean returns false, there is more letters, if the Boolean returns true, then there will be no more letter left in the word.

```
public class Node {
    public Node[] next;
    public boolean word;
    public Node() {
        next = new Node[27];
        word = false;
    }
}
```

code, index and key

In this task we are first required to implement a method that every letter corresponds to one number, from 0 to 26, this implementation is rather simple, since characters can be found on the ASCII table, the values of A to to P can first be found, as P is the letter before Q which we do not include. From the table, we notice that A is 97, and P is 112, therefore, we have, if the character is bigger or equal to 97, and less or equal to 112, we subtract 97 from it to receive the code ranging from 0 to 26. Furthermore, in order to include all letters, we take else if the character entered is R (the letter directly after Q) to V(letter directly before W), their numbers in ASCII table are 114 to 118, and we subtract 98 to receive their respective code number. At last we take X (letter after W) to Z, which is 120 to 122, and subtract 98 from it to get the code number. For the three Swedish letters, through looking at the ASCII table, we notice that they are 228, 229, and 246 respectively. So we can just make three else if functions that define them.

In order to connect the code returned to indices of integer 1 to 9, we take 27 divided by 3, which is 3 letters per number, and the indexes are just key - 1 , as indexes start with 0, and keys start with 1. In order to find the key, we notice that when entering a character, we get the code, which should be divided by three, and added by one.

```
public static int keyCode(char c){
    return code(c)/3 + 1;
}
```

adding words

When adding, start from the root and go down the tree, if we find an empty branch, we initialize a node. We use the Boolean word that we have, and

when the letter of the word is not the last, it will return false, but when it reaches the last letter in the word, it will return true, indicating that the word is done.

```
if(next[p] == null){
    next[p] = new Node();
}
if (s.length() == 1) { //
    next[p].word = true;
 ...
 }
```

search

Searching for a word is a bit more difficult, as for the keys that is used, we have to consider three alternatives, namely, when (k-1)*3, (k-1)*3 + 1 and (k-1)*3 + 2 branches. After we check for indices, we need to know whether there is a node there, if there is, then append the letter.

```
public void lookup(StringBuilder sb, String s) {
...
            if (next[b1] != null) {
                StringBuilder t = new StringBuilder(sb.toString());
                t.append(indexToChar(b1));


                if (s.length() == 1) {
                    if (next[b1].word == true) {
                    System.out.println(t.toString());
                } else
                next[b1].lookup(t, s.substring(1));
            }
        ...
        // The codes are the same for b2 and b3
```

We repeat the same process for (k-1)*3 + 1 and (k-1)*3 + 2 respectively, the implementation above shows for (k-1)*3. The implementations are more or less similar.

Vanligaste orden i svenska

Using the codes given above, we are able to type a selected word given to us from the assignment text file, by typing in their keys the words that correspond to it shows up.