T9

Laasyasree Desu

Fall 2023

1 Introduction

This assignment involved the implementation of system similar in structure to the T9. The T9 was a predictive system used in mobile phones. The numbers on the keypad corresponded to different letters of the alphabet and by keeping track of all possible words the system made suggestions or choices for the user based on the numbers typed in. The implementation in this assignment used a given list of the most common Swedish words, except for words with "q" and "w". Therefore there were a total of 27 letters used (three letters per number).

2 Trie

The first step of the implementation of the system was to initialise a tree structure called "trie". The structure had 27 branches, one for each character (letter). Each node had a next value with leaves having their next value set to null. The nodes also have a boolean value that is set to true when a leaf is reached and there is a valid word, otherwise it is set to false. A section of the code can be seen below:

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

3 Methods

3.1 Character to code

The next part of the assignment involved implementing a method that takes a character and returns the code. This was implemented with ASCII code, which meant that a-ö corresponded to 0-26. To create this method, the switch function was used. A section of the code can be seen below:

```
switch (c) {
   case 'a':
       return 0;
   case 'b':
       return 1;
   //all the way to "ö"
   case 'ö':
       return 26;
   default:
       return -1;
```

3.2 Code to character

The second method was to implement the opposite of above, which meant to take in a code and return the character. This was done using the same switch function but checking for the code instead of the character.

3.3 Key to Index

Another method to implement was to take in a key and return an index between 0-8. This done by simply subtracting 1 from the key.

```
private static int retindex(char key) {
        return key - 1;
```

3.4 Insert

The next method involved adding words to populate the tree. The method starts at the root of tree and uses a for loop to iterate through the characters of the word to be added. To receive the character, the charAt method is used and the character to code with the getCode method. If the branch in tree corresponding to the given code is empty (null) then a new node is created to hold the character. Once the word has traversed and the final branch is reached the boolean value "valid" is set to true, marking the end of the word. A section of the code can be seen below:

```
 for (int cIndex = 0; cIndex < word.length(); cIndex++) {
    char a = word.charAt(cIndex);
    int cd = getCode(a);

    if (node.next[code] == null) {
    node.next[cd] = new Node();
    }
    node = node.next[cd];
        }
node.valid = true;
```

3.5 Search

The search method involved taking a sequence of keys and finding all the possible words that match the sequence. This was implemented by using a method decode that takes in the sequence, an empty string, and an empty arraylist as parameters. The method first checks if the sequence is empty, as that would mean that it has already been processed. If its not empty, the first key of the sequence is taken and the range of branches it corresponds to are iterated thorugh. While the branch exists (not equal to null), the corresponding letters are added to the empty string. The method is called recursively and while the boolean value for the word is true it is added to arraylist. A section of the code can be seen below:

```
if (seq.isEmpty()) {
    if (node.valid) {
    words.add(curr);
    }
    return;
    }
```