Documentation of Algorithm Analysis and Design

Project for summer ‘19

“Text Auto-complete”

Antony Samir. // 20171702015

Caroline Talaat.// 20171701074

Lydia George. // 20171701080

Problem Definition:

Autocomplete is pervasive in modern applications. As the user types, the program predicts the complete query (typically a word or phrase) that the user intends to type.
Autocomplete allows the program to predict the value. When a user starts to type in a field, the program should display options to fill in the field, based on earlier typed values.

Some application on the project:

1- Search Engines.

2- Text AutoComplete

3- Database Queries.

4- Cell phones use it to speed up text input.

Input:

The input file contains the following:

1. The number of queries N.
2. N queries, each consists of a query string Q. Each query is in separate line.

The input query contains the following:

1- String from user.

2- The choice either using “Edit-Distance” or “Prefix” algorithms.

Output:

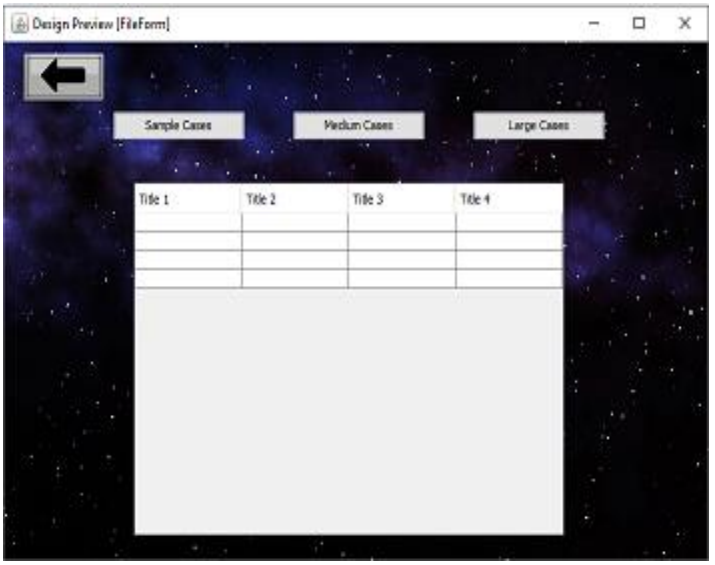
The output for file and query search contains the following:

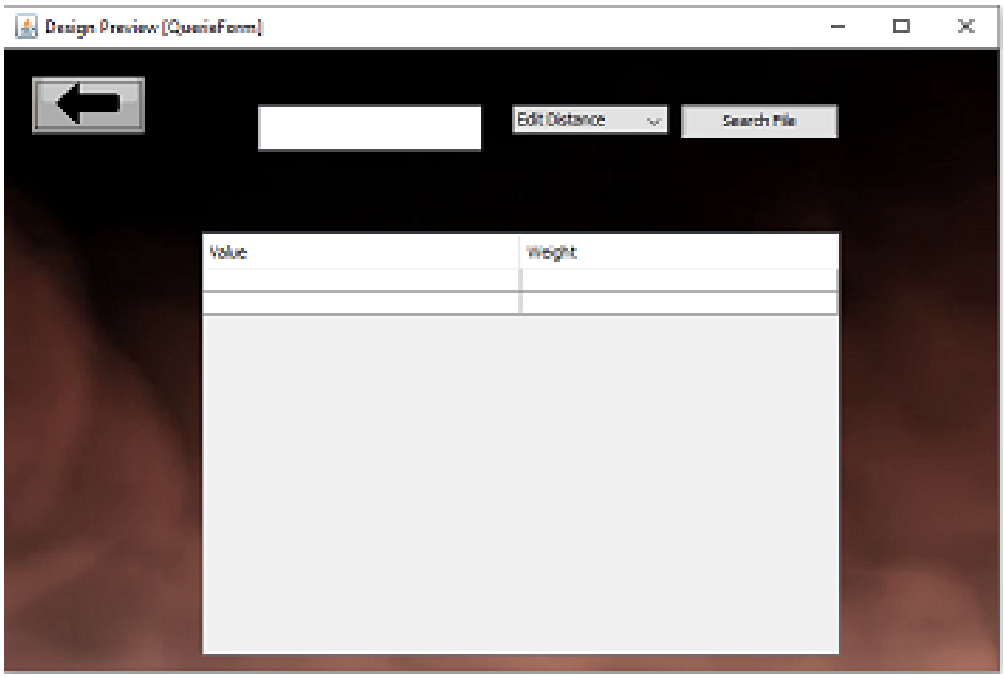
1. The number of elements Q after using prefix or edit distance.
2. The value and the weight for each query output.

Programming Language and IDE:

Java, NetBeans.

Using a powerful gui to make it easy for user :)





Running Code:

Edit Distance Function :-

```
package strings;
public class EditDistance {
   static int min(int x, int y, int z) { //Θ(1)
      if (x <= y && x <= z) { //Θ(1)
         return x; //Θ(1)
      }
      if (y <= x && y <= z) { //Θ(1)
         return y; //Θ(1)
      } else {
         return z; //Θ(1)
      }
   }

   public static int editDist(String str1, String str2, int m, int n) { //O(m*n)
      int dp[][] = new int[m + 1][n + 2]; //O(1)

      for (int i = 0; i <= m; i++) { //Θ(m)
         for (int j = 0; j <= n; j++) { //Θ(n)
            if (i == 0) { //Θ(1)
               dp[i][j] = j; //Θ(1)
            } else if (j == 0) { //Θ(1)
               dp[i][j] = i; //Θ(1)
            } else if (str1.charAt(i - 1) == str2.charAt(j - 1)) { //Θ(1)
               dp[i][j] = dp[i - 1][j - 1]; //Θ(1)
            } else {
               dp[i][j] = 1 + min(dp[i][j - 1], dp[i - 1][j], dp[i - 1][j - 1]); //Θ(1)
               // Insert //Remove //Replace
            }
         }
      }
      return dp[m][n];
   }
}
```

Prefix Function :-

```
package strings;

import java.io.IOException;
import java.util.Vector;

public class Prefix
{

    public boolean startsWith(String data, String prefix)
   {
        Boolean flag = data.startsWith(prefix);
        return flag;
    }

}
```