Universidad de Chile
Facultad de Ciencias Físicas y Matemáticas
Departamento de Ciencias de la Computacion

Informe Tarea 2

Distancia de edición

Integrante: Martin Araya
Profesor: Patricio Poblete
Auxiliares: Daniela Campos
Sven Reisenegger
Ayudantes: Bruno Rodríguez
Cristian Palma
Gabriel Chandía
Tomas Vallejos
Fecha de realización: 20 de octubre de 2018
Fecha de entrega: 20 de octubre de 2018
Santiago, Chile

1. Descripción del Problema

A partir del problema de la distancia de Levenshtein (calcular la mínima distancia de edición entre dos strings) se formula esta tarea, que consiste en calcular la mínima cantidad de pasos para convertir completamente un documento de texto en otro, y además, mostrar los cambios que fueron necesarios para completar la operación en la consola, como casos bases ocupare un texto con coherencia en nuestro lenguaje, y luego uno con letras elegidas al azar, siendo además los casos bordes los siguientes:

- Dos archivos vacíos
- Pasar de un archivo con texto a uno vacío
- Pasar de un archivo vacío a uno con texto

2. Descripción de la solución

La solución escogida consiste en un algoritmo de programación dinámica, donde se ocupa una matriz para la resolución de los subproblemas correspondientes, donde, simultáneamente tenemos una segunda matriz que se encarga de determinar todos los pasos que corresponden a cada uno de los cambios realizados, usando la siguiente nomenclatura:

- 'n': no se hizo nada en la linea correspondiente del texto
- 'c': se cambio la linea en el texto objetivo por la linea correspondiente del texto base
- 'i': se inserto la linea correspondiente del texto base en la posición del texto objetivo
- 'd': se elimino la linea correspondiente del texto objetivo

El código donde esta matriz se ve en funcionamiento es el siguiente, dividido en 2 partes

2.1. Inicialización de las matrices

Una vez recibidos los arreglos de string del texto objetivo (de aquí en adelante llamado A) y el texto base (que llamaremos B) se crea una matriz de enteros del largo de los arreglos mas uno, representando este uno el caso de un arreglo vacío.

Luego, se procede a llenar estas matriz en el caso de transformar un texto A vacío a uno B con texto y viceversa, siendo el numero de operaciones igual al largo de los textos A y B; simultáneamente, se llena la matriz de String con el numero de operaciones correspondientes ('i' o 'd') en cada caso

2.2. Calculo de los valores de las matrices

Luego de que las matrices están inicializadas, se procede a realizar el calculo a partir del algoritmo de mínima distancia de edición, que se ilustra en la figura 1, la implementación en el caso de la matriz de strings es bastante engorrosa, sin embargo, cubre todos los casos posibles, ya

que no solo nos sirve saber cual es el valor mínimo de operaciones, sino que también necesitamos saber específicamente que acción se tomo anteriormente, siendo la prioridad utilizada inserción-eliminación-cambio, además, esto no afecta el numero de pasos necesarios.

La función $d(i, j)$ se puede calcular recursivamente de la siguiente manera:

$$d(i, 0) = i$$

$$d(0, j) = j$$

$$d(i, j) = \min \begin{cases} d(i-1, j) + 1 \\ d(i, j-1) + 1 \\ d(i-1, j-1) + [a_i \neq b_j] \end{cases}$$

donde la notación $[P]$ toma el valor 1 si P es verdadero, y toma el valor 0 si P es falso.

Figura 1: Descripcion del calculo de una casilla de la matriz

Código 1: Calculo de las casillas de ambas matrices

```
for(int i=1;i<=B.length;i++){
    for(int j=1;j<=A.length;j++){
        distancia[i][j] = min(distancia[i-1][j]+1,
                distancia[i][j-1]+1,
                distancia[i-1][j-1] + ((Objects.equals(B[i - 1], A[j - 1]))?0:1));
        if (B[i - 1].equals(A[j - 1])){
            if(distancia[i-1][j]+1<=distancia[i-1][j-1]){
                if(distancia[i-1][j]+1<=distancia[i][j-1]+1){
                    operaciones[i][j]=operaciones[i-1][j].concat("i");
                }
                else{
                    if(distancia[i][j-1]+1<=distancia[i-1][j-1]){
                        operaciones[i][j]=operaciones[i][j-1].concat("d");
                    }
                    else{
                        operaciones[i][j]=operaciones[i-1][j-1].concat("n");
                    }
                }
            }
            else{
                if(distancia[i][j-1]+1<=distancia[i-1][j-1]){
                    operaciones[i][j]=operaciones[i][j-1].concat("d");
                }
                else {
                    operaciones[i][j]=operaciones[i-1][j-1].concat("n");
                }
            }
        }
        else{
            if(distancia[i-1][j]+1<=distancia[i-1][j-1]+1){
```

```
31                  if(distancia[i-1][j]+1<=distancia[i][j-1]+1){
32                      operaciones[i][j]=operaciones[i-1][j].concat("i");
33                  }
34                  else{
35                      if(distancia[i][j-1]+1<=distancia[i-1][j-1]+1){
36                          operaciones[i][j]=operaciones[i][j-1].concat("d");
37                      }
38                      else{
39                          operaciones[i][j]=operaciones[i-1][j-1].concat("c");
40                      }
41                  }
42              }
43              else{
44                  if(distancia[i][j-1]+1<=distancia[i-1][j-1]+1){
45                      operaciones[i][j]=operaciones[i][j-1].concat("d");
46                  }
47                  else {
48                      operaciones[i][j]=operaciones[i-1][j-1].concat("c");
49                  }
50              }
51          }
52      }
53  }
```

Finalmente, tenemos el resto del código del Main, donde tenemos la lectura de archivo, que se realizo mediante un Arraylist, ya que no conocemos a priori el tamaño de los textos, que luego se pasa a una lista de String, para ser luego ingresada a la función, y, mediante la lista de caracteres que nos da la función, mostramos en la consola los cambios realizados (solo si estos fueron de la forma cambio o inserción)

3. Resultados

Como se menciono en la sección numero 1, tenemos 3 casos de borde que usamos, además del caso de ejemplo que esta presente en el enunciado de la tarea, los resultados de esto se presentan a continuación

3.1. Texto de ejemplo

Como lo indica la tarea, tenemos dos textos de la siguiente forma

Código 2: texto A

```
Que linda en la rama
la fruta se ve
si lanzo un piedra
tendr que caer
```

Código 3: texto B

```
Que linda en la rama
las frutas se ven
ojala no me atrapen
si lanzo una piedra
```

y como resultado tenemos

Código 4: Resultado Primer ejemplo

```
A
B
2,c, las frutas se ven
3,c, ojala no me atrapen
4,c, si lanzo una piedra

Process finished with exit code 0
```

3.2. Ambos textos vacíos

En este caso de ejemplo, tenemos tanto el texto A como el texto B vacío, por ende el resultado esperado es que no se imprima nada en la consola, puesto que no es necesario hacer cambios, se omitirán los textos, pero como resultado en la consola obtuvimos:

Código 5: Resultado Segundo ejemplo

```
A
B

Process finished with exit code 0
```

3.3. Texto A vacío

En este ejemplo, el texto objetivo esta vacío, y el texto objetivo es el mismo que en el primer ejemplo, por ende, intuitivamente podemos decir que es necesario insertar todas las lineas del texto base en el texto de referencia, por ende tenemos:

Código 6: Texto B

```
Que linda en la rama
las frutas se ven
ojala no me atrapen
si lanzo una piedra
```

y como resultado en la consola obtuvimos:

Código 7: Resultado tercer ejemplo

```
A
B
1,i, Que linda en la rama
2,i, las frutas se ven
3,i, ojala no me atrapen
4,i, si lanzo una piedra

Process finished with exit code 0
```

3.4. Texto B vacío

En este ultimo ejemplo, tenemos un texto base vacío y un texto objetivo con el contenido del primer ejemplo, en este caso, la intuición nos dice que es necesario eliminar todas las lineas del texto objetivo para conseguir el texto base como resultado, por ende tenemos:

Código 8: Texto A

```
Que linda en la rama
la fruta se ve
si lanzo un piedra
tendr que caer
```

y como resultado final tenemos:

Código 9: Resultados cuarto ejemplo

```
A
B
1,d
2,d
3,d
4,d

Process finished with exit code 0
```

4. Código Fuente

Código 10: Código final

```
import java.io.File;
import java.io.FileNotFoundException;
import java.util.ArrayList;
import java.util.Objects;
import java.util.Scanner;

public class Tarea2 {
    private static int min(int a,int b, int c) { return Math.min(a,Math.min(b,c));}

    private static char[] diferenciar(String[] A, String[] B){
        int [][] distancia = new int[B.length+1][A.length+1];
        String [][] operaciones = new String[B.length+1][A.length+1];
        for(int i = 0; i<=B.length;i++){
            for(int j=0; j<=A.length;j++){
                operaciones[i][j]="";
            }
        }
        for(int i=1;i<=B.length;i++){
            distancia[i][0]=i;
            operaciones[i][0]=operaciones[i-1][0].concat("i");
        }
        for(int j=1;j<=A.length;j++){
            distancia[0][j]=j;
            operaciones[0][j]=operaciones[0][j-1].concat("d");
        }
        for(int i=1;i<=B.length;i++){
            for(int j=1;j<=A.length;j++){
                distancia[i][j] = min(distancia[i-1][j]+1,
                        distancia[i][j-1]+1,
                        distancia[i-1][j-1] + ((Objects.equals(B[i - 1], A[j - 1]))?0:1));
                if (B[i - 1].equals(A[j - 1])){
                    if(distancia[i-1][j]+1<=distancia[i-1][j-1]){
                        if(distancia[i-1][j]+1<=distancia[i][j-1]+1){
                            operaciones[i][j]=operaciones[i-1][j].concat("i");
                        }
                        else{
                            if(distancia[i][j-1]+1<=distancia[i-1][j-1]){
                                operaciones[i][j]=operaciones[i][j-1].concat("d");
                            }
                            else{
                                operaciones[i][j]=operaciones[i-1][j-1].concat("n");
                            }
                        }
                    }
                    else{
                        if(distancia[i][j-1]+1<=distancia[i-1][j-1]){
                            operaciones[i][j]=operaciones[i][j-1].concat("d");
```

```
48                      }
49                      else {
50                          operaciones[i][j]=operaciones[i-1][j-1].concat("n");
51                      }
52                  }
53              }
54              else{
55                  if(distancia[i-1][j]+1<=distancia[i-1][j-1]+1){
56                      if(distancia[i-1][j]+1<=distancia[i][j-1]+1){
57                          operaciones[i][j]=operaciones[i-1][j].concat("i");
58                      }
59                      else{
60                          if(distancia[i][j-1]+1<=distancia[i-1][j-1]+1){
61                              operaciones[i][j]=operaciones[i][j-1].concat("d");
62                          }
63                          else{
64                              operaciones[i][j]=operaciones[i-1][j-1].concat("c");
65                          }
66                      }
67                  }
68                  else{
69                      if(distancia[i][j-1]+1<=distancia[i-1][j-1]+1){
70                          operaciones[i][j]=operaciones[i][j-1].concat("d");
71                      }
72                      else {
73                          operaciones[i][j]=operaciones[i-1][j-1].concat("c");
74                      }
75                  }
76              }
77          }
78      }
79      return operaciones[B.length][A.length].toCharArray();
80  }
81  public static void main(String[] args){
82      Scanner sc = new Scanner(System.in);
83      String La;
84      String Lb;
85      ArrayList<String> s1 = new ArrayList<>();
86      ArrayList<String> s2 = new ArrayList<>();
87      La = sc.nextLine();
88      Lb = sc.nextLine();
89      File A = new File("C:\\Users\\Kawallala\\Google Drive\\Universidad\\4to
    Semestre\\Algoritmos\\"+La+".txt");
90      File B = new File("C:\\Users\\Kawallala\\Google Drive\\Universidad\\4to
    Semestre\\Algoritmos\\"+Lb+".txt");
91      sc.close();
92
93      try (Scanner sA = new Scanner(A)) {
94          try (Scanner sB = new Scanner(B)) {
95              while (sA.hasNextLine()) {
96                  s1.add(sA.nextLine());
97              }
```

```
98                  while (sB.hasNextLine()) {
99                      s2.add(sB.nextLine());
100                 }
101             } catch (FileNotFoundException e) {
102                 e.printStackTrace();
103             }
104         } catch (FileNotFoundException e) {
105             e.printStackTrace();
106         }
107
108         String[] alfa = new String[s1.size()];
109         String[] beta = new String[s2.size()];
110
111         for(int i =0 ; i<s1.size();i++){
112             alfa[i]=s1.get(i);
113         }
114         for(int j =0; j<s2.size();j++){
115             beta[j]=s2.get(j);
116         }
117         char[] cambios = diferenciar(alfa,beta);
118         for(int i = 0;i<cambios.length; i++){
119             if (cambios[i]=='c') {
120                 System.out.println((i + 1) + ",c, " + beta[i]);
121             }
122             else if(cambios[i]=='i'){
123                 System.out.println((i+1) + ",i, " + beta[i]);
124             }
125             else if(cambios[i]=='d'){
126                 System.out.println((i+1) +",d ");
127             }
128         }
129     }
130 }
```