FACULTAD DE CIENCIAS FÍSICAS Y MATEMÁTICAS UNIVERSIDAD DE CHILE

Tarea 2: Diff

Autor: Vanessa Gaete
Correo: naan.u.285@gmail.com

Profesor: Jeremy Barbay
Auxiliares: Cristóbal Muñoz
Daniela Campos
Sven Reisenegger
Bernardo Subercaseaux
Curso: CC3001

Fecha de entrega: 19 de Octubre de 2018
Santiago, Chile

Índice de Contenidos

Lista de Figuras

1. Descripción del problema

El problema a resolver en esta tarea consiste en crear un programa en Java que entregue las instrucciones para modificar las líneas de un primer texto y tranformarlo así en un segundo texto ingresado. Las instrucciones deben seguir la forma "Número de línea, operación, nueva línea", donde las operciones son: "c" para cambiar la frase, "i" para insertar una nueva y "d" para borrarla. Esto debe realizarse en la cantidad mínima de pasos para que así el algortimo sea lo más optimal posible. Para lograr esto último se utilizará la función llamada "distancia de edición (o de Levenshtein)", que calcula el número mínimo de pasos para transformar un texto en otro.

Para realizar este código se supuso que el usuario ingresaría siempre un archivo ".txt"válido. Los casos bases corresponden a: *Si ambos textos tienen 1 linea y estas son distintas entonces la cantidad de pasos necesarios es 1 y la instrucción es cambiar la línea 1 del primer texto por la primera del segundo. *Si ambos tienen 1 linea y son iguales, la cantidad de pasos es cero y no hay que hacer nada. *Si el primer texto es vacio hay que hacer tantas operaciones como líneas tiene el segundo texto y la instruccion es que hay que cambiar cada linea de i por la correspondiente en j. *Si el segundo texto es vacio, hay que borrar todas las lineas del primero, con lo que se hacen tantas operaciones como líneas tenga este.

2. Descripcion de la solución

Lo primero que hace el programa es leer los textos dados en el input con scanner y se guardan como archivos. A cada archivo se le cuentan sus líneas y se guardan en un string con cada línea del texto separada por "para luego hacer un arreglo con ellas. Estos arreglos serán ingresados en la función diff para obtener los solicitado en la tarea.

El algoritmo de la funcion diff consiste principalmente en crear dos matrices, una dp[m][n] de enteros y otra cp[m][n] de strings, con m la cantidad de lineas del primer texto y n la del segundo. Ambas matrices se van creando paralelamente.

La función de dp[m][n] es guardar el número de pasos mínimos a seguir para cada combinación de textos de largo desde 0 a m y 0 a n.

Por otro lado la matriz cp[m][n] va guardando en sus casillas las instrucciones a seguir óptimas a seguir para cada combinación de textos de largo desde 0 a m y 0 a n. Esta matriz se obtiene con la ayuda de dp[m][n], pues es esa matriz la que contiene la información sobre la cantidad de pasos mínimos, y si cp[m][n] no derivara de ella la cantidad de instrucciones sería mayor de la que se solicitan.Así la casilla [i][j] en dp tendrá el número de pasos óptimo para cambiar un texto de largo i a un texto de largo j y en cp tendrá las instrucciones.

Primero las matrices son rellenadas con los casos bases (que ya fueron mencionados pero se colocarán de nuevo para que sea más simple de comprender el algortimo), que serían:

*Si ambos textos tienen 1 linea y estas son distintas entonces la cantidad de pasos necesarios es 1 y la instrucción es cambiar la línea 1 del primer texto por la primera del segundo.

```
1                   if(i==0 && j==0 && ! TA[i].equals(TB[j])){
2                       dp[i][j] = 1;
3                       cp[i][j]=i+1 +",c,"+TB[j]+'\n';
4                   }
```

*Si ambos tienen 1 linea y son iguales, la cantidad de pasos es cero y no hay que hacer nada.

```
1                   else if(i==0 && j==0 && TA[i].equals(TB[j])){
2                       dp[i][j] = 0;
3                       cp[i][j]="";
4                   }
```

*Si el primer texto es vacio hay que hacer tantas operaciones como líneas tiene el segundo texto y la instruccion es que hay que cambiar cada linea de i por la correspondiente en j.

```
1                    else if(i == 0){
2                       dp[i][j] = j;
3                       cp[i][j] = cp[i][j - 1];
4                       cp[i][j] += j + ",i," + TB[j] + '\n';
5
6                   }
```

*Si el segundo texto es vacio, hay que borrar todas las lineas del primero, con lo que se hacen tantas operaciones como líneas tenga este.

```
1                   else if(j == 0){
2                       dp[i][j] = i;
3                       cp[i][j] = cp[i-1][j];
4                       cp[i][j] += i+",d"+'\n';
5
6                   }
```

Luego con estos casos bases se comienza a rellenar el resto de la matriz con recursiones:

*En el caso que la línea "i" del primer texto y la linea "j" sean iguales entonces no se hace nada y las instrucciones y pasos seran los mismos de cp[i-1][j-1] y dp[i-1][j-1] respectivamente.

```
                else if(TA[i].equals(TB[j])){
                    dp[i][j] = dp[i-1][j-1];
                    cp[i][j]=cp[i-1][j-1];
```

* En otro caso se escoje la casilla circundante de dp con menor número de pasos y se le suma uno, porque se realiza una operación en este proceso. Y dependiendo de cuál sea ese mínimo será, la instrucción que se arrojará. Así, si la casilla con el mínimo es dp[i][j-1] entonces la instruccion será insertar en la línea j del texto 1 la línea correspondiente de j, esto se suma a las instrucciones que venían de pasos anteriores. Si la casilla con el minimo es dp[i-1][j], se borra la casilla i del texto uno y se suma a las instrucciones que venían de pasos anteriores. Y por último si el minimo es dp[i-1][j-1] se cambia la línea i del primer texto por la linea j del otro.

```
                else{
                    dp[i][j] = 1 +  Math.min(Math.min(dp[i][j-1], dp[i-1][j]),dp[i-1][j-1])
    ;
                    // debemos encontrar el minimo de las posibilidades que tenmos
                    if (Math.min(Math.min(dp[i][j-1], dp[i-1][j]),dp[i-1][j-1])==dp[i][j
    -1]){
                        cp[i][j]=cp[i][j-1]+j+",i,"+TB[j]+'\n';
                    }
                    else if (Math.min(Math.min(dp[i][j-1], dp[i-1][j]),dp[i-1][j-1])==dp[i
    -1][j]){
                        cp[i][j]=cp[i-1][j]+i+",d\n";
                    }
                    else{
                        cp[i][j]=cp[i-1][j-1]+i+",c,"+TB[j]+'\n';
                    }
                }
```

3. Resultados

Se intentaron probar los casos más representativos y algunos de afeed, los casos fueron los siquientes:

*Texto 1:
brp
brp
*Texto 2:
trm
lqd
*Resultado:
1,c,trm
2,c,lqd

*Texto 1:
Que linda en la rama
la fruta se ve
si lanzo una piedra
tendra que caer
*Texto 2:
Que linda en la rama
las frutas se ven
ojala que no me atrapen
si lanzo una piedra
*Resultado:
1,c,trm
2,c,lqd

Texto 1:
Texto2:
Que linda en la rama
las frutas se ven
ojala que no me atrapen
si lanzo una piedra
Resultado:
1,i,Que linda en la rama
2,i,las frutas se ven
3,i,ojala que no me atrapen
4,i,si lanzo una piedra

*Texto 1:
Que linda en la rama
la fruta se ve
si lanzo una piedra
tendra que caer
*Texto2:
Resultado: 1,d
2,d
3,d
4,d

*Texto1:
*Texto2:
Resultado:

*Texto1:
dck
*Texto2:
ale
Resultado:
1,c,ale

4. Código Fuente

```
import java.util.*;
import java.io.*;
import java.io.IOException;
import java.io.FileNotFoundException;

public class Main{

     public static String diff(String[] TA, String[] TB){
        int m = TA.length;
        int n = TB.length;

        int[][] dp = new int[m][n];
        String[][] cp = new String[m][n];

         for (int i=0; i < m ; i++) {
             for (int j=0; j< n ; j++) {
                 if(i==0 && j==0 && ! TA[i].equals(TB[j])){
                     dp[i][j] = 1;
                     cp[i][j]=i+1 +",c,"+TB[j]+'\n';
                 }
                 // no hay nada en el primer string, por lo que los unicos cambios
                 // que nos faltan son los que no hemos hecho en T, es decir j
                 else if(i==0 && j==0 && TA[i].equals(TB[j])){
                     dp[i][j] = 0;
                     cp[i][j]="";
                 }
                  else if(i == 0){
                     dp[i][j] = j;
                     cp[i][j] = cp[i][j - 1];
                     cp[i][j] += j + ",i," + TB[j] + '\n';

                 }
                 // no hay nada en el segundo string, por lo que los unicos cambios
                 // que nos faltan son los que no hemos hecho en S, es decir i
                 else if(j == 0){
                     dp[i][j] = i;
                     cp[i][j] = cp[i-1][j];
                     cp[i][j] += i+",d"+'\n';

                 }
                 // si son los mismo caracteres no debemos hacer ningun cambio
                 else if(TA[i].equals(TB[j])){
                     dp[i][j] = dp[i-1][j-1];
                     cp[i][j]=cp[i-1][j-1];
                     // no debemos hacer cambios, pero puede ser que ya hayamos
                     // hecho cambios antes de llegar a este punto,
                     //por lo que debenos considerar las operaciones anteriores
                 }

                 // cualquier otro caso, debemos si o si hacer una operacion
                 else{
                     dp[i][j] = 1 +  Math.min(Math.min(dp[i][j-1], dp[i-1][j]),dp[i-1][j-1])
    ;
                     // debemos encontrar el minimo de las posibilidades que tenmos
                     if (Math.min(Math.min(dp[i][j-1], dp[i-1][j]),dp[i-1][j-1])==dp[i][j
    -1]){
                          cp[i][j]=cp[i][j-1]+j+",i,"+TB[j]+'\n';
                     }
                     else if (Math.min(Math.min(dp[i][j-1], dp[i-1][j]),dp[i-1][j-1])==dp[i
    -1][j]){
                         cp[i][j]=cp[i-1][j]+i+",d\n";
                     }
                     else{
```

```
                        cp[i][j]=cp[i-1][j-1]+i+",c,"+TB[j]+'\n';
                    }
                }
            }

        }

        return cp[m-1][n-1];
    }

    public static void main(String[] args)  throws FileNotFoundException, IOException {

        Scanner scanner = new Scanner(System.in);

        String filename1 = scanner.nextLine();
        String filename2 = scanner.nextLine();

        File file1 = new File(filename1); //filename es el nombre del archivo
        File file2 = new File(filename2); //filename es el nombre del archivo

            Scanner A = new Scanner(file1);
            Scanner B = new Scanner(file2);

            int countB = 0;
            String lineasb= "";
            while (B.hasNextLine()) {
                lineasb +="-"+B.nextLine();
                countB+=1;
            }


            int countA = 0;
            String lineasa= "";
            while (A.hasNextLine()) {
                lineasa +="-"+A.nextLine();
                countA+=1;
            }

            String[] TA = lineasa.split("-");

            String[] TB = lineasb.split("-");

            System.out.println(diff(TA,TB));


    }

    /*
        String largos[] = scanner.nextLine().split(" ");
        int largoA = Integer.parseInt(largos[0]);
        int largoB = Integer.parseInt(largos[1]);

        System.out.println("largoA = " + largoA + " largoB = " + largoB);

        String[] TA = new String[largoA];
        String[] TB = new String[largoB];

        for (int i = 0; i < largoA; ++i) {
            TA[i] = scanner.nextLine();
        }

        for (int i = 0; i < largoB; ++i) {
            TB[i] = scanner.nextLine();
        }

        System.out.println(diff(TA, TB));
```

```
127
128
129     }*/
130 }
```