Práctica 3. SCMC. Programación dinámica.

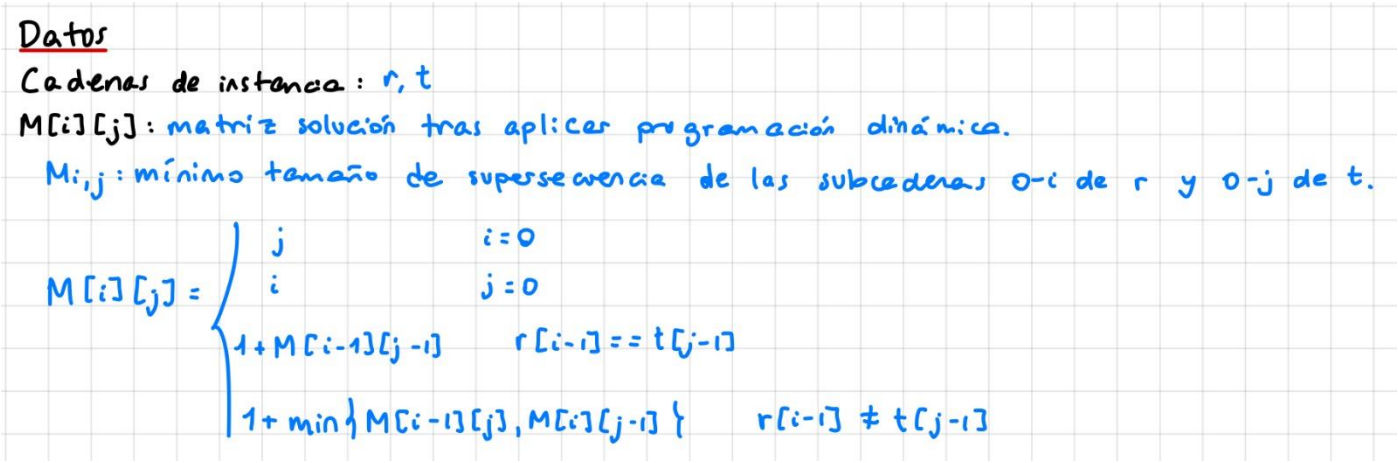
María Peinado Toledo. Doble Grado Ingeniería Informática y Matemáticas.

Vamos a tratar con el problema de la supersecuencia común más corta (SCMC). En resumen, diremos que la cadena s es supersecuencia común de las cadenas r y t si s >> r y s >> t. Para encontrar una supersecuencia común de ambas bastaría con la cadena que se obtiene al concatenarlas, pero nosotros estamos interesados en encontrar una supersecuencia común los más corta posible.

Veamos la clase SCMC.java utilizada para resolver el problema:

```java
public class SCMC {

    private String r, t; // Las dos cadenas de la instancia
    private String sigma; // El alfabeto de la instancia
    private int m[][]; // la matriz para resolucion por Prog. Dinamica
    private static Random rnd = new Random(); // generador de aleatorio

    /**
     * Crea una instancia del problema
     * @param sigma : alfabeto para la instancia
     * @param r     : primera cadena
     * @param t     : segunda cadena
     */
    public SCMC(String sigma, String r, String t) {
        this.r = r;
        this.t = t;
        this.sigma = sigma;
        m = new int[1+r.length()][1+t.length()];

    }

    /**
     * Crea una instancia aleatoria del problema
     * @param longMax  : longitud maxima de las cadenas
     * @param tamSigma : tamaño del alfabeto
     */
    public SCMC(int longMax, int tamSigma) {
        this.sigma = Utils.alfabeto(tamSigma);
        r = Utils.cadenaAleatoria(rnd.nextInt(1+longMax),sigma);
        t = Utils.cadenaAleatoria(rnd.nextInt(1+longMax),sigma);
        m = new int[1+r.length()][1+t.length()];

    }

    public String r(){
        return r;
    }

    public String t(){
        return t;
    }

    public String sigma(){
        return sigma;
    }

    public int m(int i, int j){
        if (i<m.length && j<m[0].length) {
            return m[i][j];
        } else {
            return -1;
        }
    }
```

Vemos que el objeto se puede crear de dos maneras distintas. En la primera se proporcionan las cadenas y el alfabeto y en la segunda se generar aleatoriamente. Ahora sí, pasemos al algoritmo que nos resuelva el problema utilizando programación dinámica. Para ello vamos a seguir el siguiente esquema:



-Si i=0, si j=0
Entonces estamos intentado conseguir la longitud de una supersecuencia entre una cadena vacía y otra, luego la mínima longitud será la longitud de la no-vacía (i ó j).
-Si r[i-1]==t[j-1]
En el caso que la letra anterior coincida, bastaría con añadirle esa letra a la supersecuencia luego tenemos 1+M[i-1][j-1].
-Si r[i-1]/=t[j-1]
Entonces la longitud mínima se obtendrá al comparar la longitud mínima si tomamos (i-1) y (j) o la longitud mínima al tomar (i) y (j-1).

Creamos un método para crear la matriz que nos dará la solución

```
public void solucionaPD(){
    int rTam=r.length();
    int tTam=t.length();
    for(int i=0;i<=rTam;i++) {
        for(int j=0;j<=tTam;j++) {
            if(i==0) {
                m[i][j]=j;
            }else if(j==0) {
                m[i][j]=i;
            }else if (r.charAt(i-1)==t.charAt(j-1)) {
                m[i][j]=1+m[i-1][j-1];
            }else {
                m[i][j]=1+Math.min(m[i-1][j], m[i][j-1]);
            }
        }
    }
}
```

```
/**
 * @return : devuelve la longitud de la solucion
 *           a la instancia, es decir, la longitud
 *           de la supersecuencia comun más corta de @r y @t
 *           a partir de la tabla obtenida por Prog Dinamica
 */
public int longitudDeSolucionPD(){
    return m(r.length(),t.length());
}

/**
 * @return Devuelve una solucion optima de la instancia, es decir
 *         una supersecuencia comun mas corta de @r y @t
 */
```

```java
public String unaSolucionPD(){
    int tam = m[r.length()][t.length()];
    char[] res = new char[tam];
    int i = r.length(), j = t.length();
    while(i>0&&j>0) {
        if(r.charAt(i-1)==t.charAt(j-1)) {
            res[--tam]=r.charAt(i-1);
            i--;
            j--;
        }else if (m[i-1][j]<m[i][j-1]) {
            res[--tam]=r.charAt(i-1);
            i--;
        }else {
            res[--tam]=t.charAt(j-1);
            j--;
        }
    }

    while(i>0) {
        res[--tam]=r.charAt(i-1);
        i--;
    }
    while(j>0) {
        res[--tam]=t.charAt(j-1);
        j--;
    }

    return new String(res);

}

// representacion como String de la instancia
public String toString(){
    return "Sigma="+Utils.entreComillas(sigma)
            +", r="+Utils.entreComillas(r)
            +", s="+Utils.entreComillas(t);
}



// Obtiene una solucion al problema por "fuerza bruta"
public String unaSolucionFB() {
    int l = Math.max(r.length(),t.length());
    String res = null;
    for(l=Math.max(r.length(),t.length()); res==null; l++)
      res = unaSolucionFB("",l);
    return res;
}

// método auxiliar recursivo
private String unaSolucionFB(String s, int l) {
    String str = null;
    if(l==0) {
        if(Utils.esSupersecuencia(s,r) && Utils.esSupersecuencia(s,t))
            str = s;
    }
    else
        for(int i=0; i<sigma.length(); i++) {
            str = unaSolucionFB(s+sigma.charAt(i),l-1);
            if(str!=null) break;
        }
    return str;
}
```

```
public class Prueba {

    public static void main(String[] args) {
        SCMC scmc = new SCMC("abc","aab","acbaa");
        scmc.solucionaPD();
        System.out.println("La longitud de la SCMC es "+scmc.longitudDeSolucionPD());
        System.out.println("Una SCMC es "+scmc.unaSolucionPD());

    }

}
```

Además, para las cadenas “aab” y “acbaa” la tabla que obtenemos es la siguiente:

0	1	2	3	4	5
1	1	2	3	4	5
2	2	3	4	4	5
3	3	4	4	5	6

La solución de la longitud mínima de supersecuencia de las subcadenas de 0-i de r y 0-j de t se encuentra en la celda naranja.

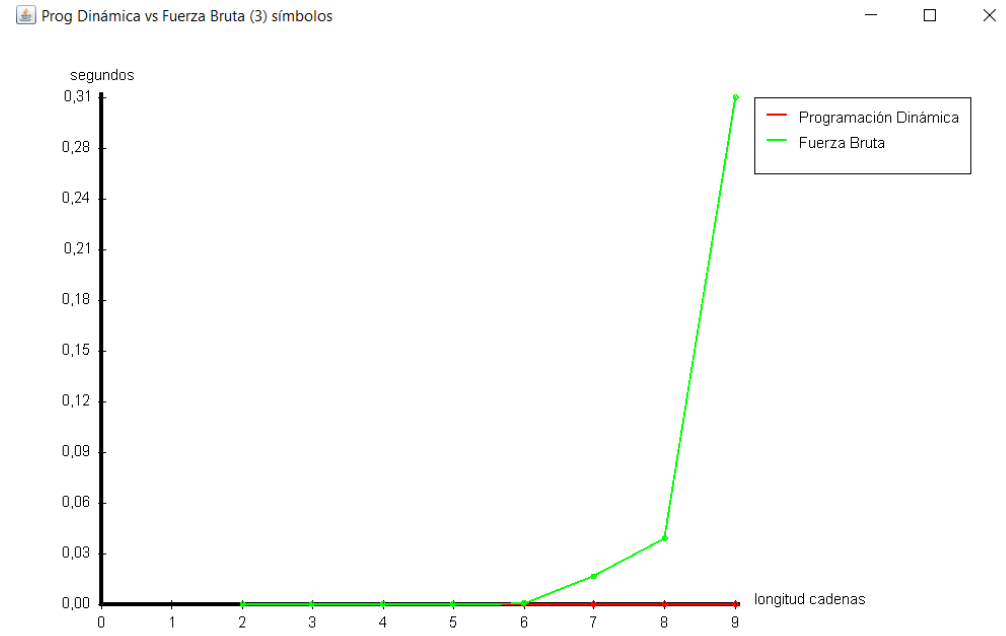
La consola muestra por pantalla lo siguiente:

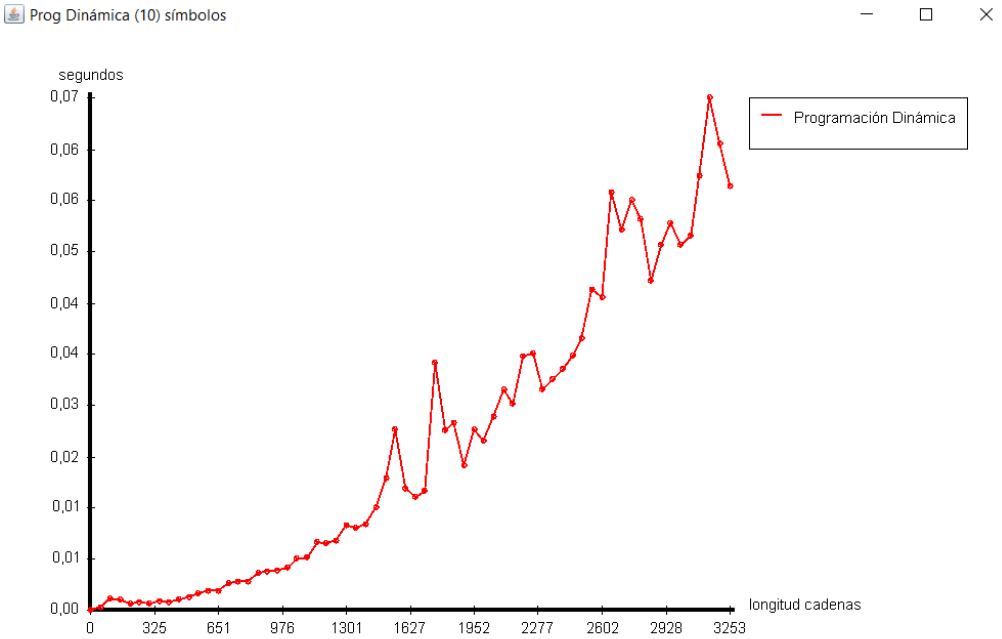
```
Console
<terminated> Prueba [Java Application]
La longitud de la SCMC es 6
Una SCMC es aacbaa
```

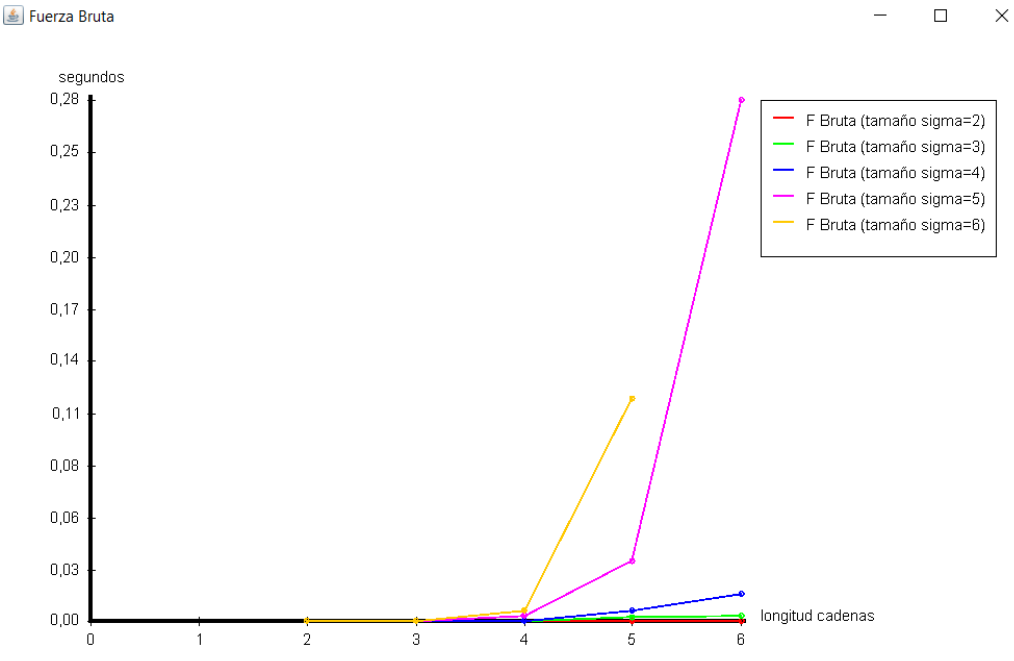
Y tras probar la clase TestCorrección.java la consola muestra por pantalla lo siguiente:

```
Console
<terminated> TestsCorreccion (1) [Java Application]
Test de unaSolucion correcto
Test de longitudDeSolucionPD correcto
```

Adjunto las gráficas resultado de la práctica:


Prog Dinámica vs Fuerza Bruta (3) símbolos
segundos
0,31
0,28
0,24
0,21
0,18
0,15
0,12
0,09
0,06
0,03
0,00
0
1
2
3
4
5
6
7
8
9
longitud cadenas
Programación Dinámica
Fuerza Bruta


Prog Dinámica (10) símbolos
segundos
0,07
0,06
0,06
0,05
0,04
0,04
0,03
0,02
0,01
0,01
0,00
0
325
651
976
1301
1627
1952
2277
2602
2928
3253
longitud cadenas
Programación Dinámica


Fuerza Bruta
segundos
0,28
0,25
0,23
0,20
0,17
0,14
0,11
0,08
0,06
0,03
0,00
0
1
2
3
4
5
6
longitud cadenas
F Bruta (tamaño sigma=2)
F Bruta (tamaño sigma=3)
F Bruta (tamaño sigma=4)
F Bruta (tamaño sigma=5)
F Bruta (tamaño sigma=6)