ALGORITMIA

PRÁCTICA 5

Héctor Lavandeira Fernández
UO277303 | UNIVERSIDAD DE OVIEDO – CURSO 2021/22

Características principales del ordenador:

Procesador:	i5-8250U
Memoria RAM:	8GB

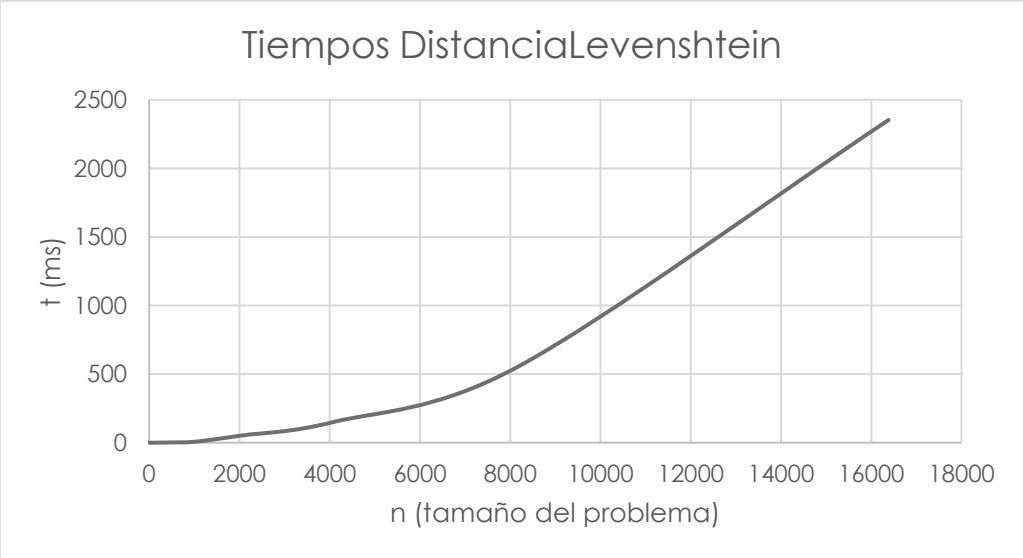
Mediciones de tiempo del algoritmo:

n	*tiempo (ms)*	*repeticiones*	*t / repeticiones (ms)*
1	63	1000000	0,000063
2	94	1000000	0,000094
4	149	1000000	0,000149
8	470	1000000	0,00047
16	1089	1000000	0,001089
32	3212	1000000	0,003212
64	12430	1000000	0,01243
128	114	1000	0,114
256	630	1000	0,63
512	2165	1000	2,165
1024	7567	1000	7,567
2048	522	10	52,2
4096	1519	10	151,9
8192	5575	10	557,5
16384	23530	10	2353

Después de tomar el último valor, salta una excepción indicando que no hay memoria:

```
Exception in thread "main" java.lang.OutOfMemoryError: Java heap space
        at p5.DistanciaLevenshtein.<init>(DistanciaLevenshtein.java:13)
        at p5.DistanciaLevenshteinTiempos.main(DistanciaLevenshteinTiempos.java:17)
```

Representación de los tiempos medidos:



Análisis de la complejidad del algoritmo

```
public int calcularDistancia() {
    for (int i = 1; i < matriz.length; i++) {
        for (int j = 1; j < matriz[0].length; j++) {
            if (cad1[i - 1] == cad2[j - 1]) {
                matriz[i][j] = matriz[i - 1][j - 1];
            } else {
                matriz[i][j] = 1 + min(matriz[i - 1][j - 1], matriz[i - 1][j], matriz[i][j - 1]);
            }
        }
    }
    return matriz[cad1.length][cad2.length];
}
```

Analizando la implementación del algoritmo, podemos observar como **siempre** se recorre toda la matriz, por lo que la complejidad será $O(n^2)$, ya que ambas cadenas tienen la misma longitud n. Al medir tiempos y representarlos en una gráfica, se respeta dicha complejidad.

Esto podemos verificarlo utilizando la fórmula $t_2 = f(n_2)/f(n_1) \cdot t_1$. Utilizando los valores $n_1 = 8192$, $n_2 = 16384$ y $t_1 = 557{,}5$ ms:

$$t_2 = (16384)^2 / (8192)^2 \cdot 557{,}5 = 4 \cdot 557{,}5 = 2230 \text{ ms}$$

Comparando con el valor obtenido para n = 16384 en la tabla de mediciones:

2230 ms (valor calculado) ≃ 2353 ms (valor obtenido)

Resultados del algoritmo con distintas cadenas

```
Cadena 1: ABRACADABRA
Cadena 2: BARCAZAS
         0       1       2       3       4       5       6       7       8
         1       1       1       2       3       4       5       6       7
         2       1       2       2       3       4       5       6       7
         3       2       2       2       3       4       5       6       7
         4       3       2       3       3       3       4       5       6
         5       4       3       3       3       4       4       5       6
         6       5       4       4       4       3       4       4       5
         7       6       5       5       5       4       4       5       5
         8       7       6       6       6       5       5       4       5
         9       8       7       7       7       6       6       5       5
         10      9       8       7       8       7       7       6       6
         11      10      9       8       8       8       8       7       7
Distancia de Levenshtein = 7

Cadena 1: ELEFANTE
Cadena 2: RELEVANTE
        0       1       2       3       4       5       6       7       8       9
        1       1       1       2       3       4       5       6       7       8
        2       2       2       1       2       3       4       5       6       7
        3       3       2       2       1       2       3       4       5       6
        4       4       3       3       2       2       3       4       5       6
        5       5       4       4       3       3       2       3       4       5
        6       6       5       5       4       4       3       2       3       4
        7       7       6       6       5       5       4       3       2       3
        8       8       7       7       6       6       5       4       3       2
Distancia de Levenshtein = 2
```

Cadena 1: ABRACADABRA
Cadena 2: RELEVANTE

0	1	2	3	4	5	6	7	8	9
1	1	2	3	4	5	5	6	7	8
2	2	2	3	4	5	6	6	7	8
3	2	3	3	4	5	6	7	7	8
4	3	3	4	4	5	5	6	7	8
5	4	4	4	5	5	6	6	7	8
6	5	5	5	5	6	5	6	7	8
7	6	6	6	6	6	6	6	7	8
8	7	7	7	7	7	6	7	7	8
9	8	8	8	8	8	7	7	8	8
10	9	9	9	9	9	8	8	8	9
11	10	10	10	10	10	9	9	9	9

Distancia de Levenshtein = 9