**Ejercicio 1:**

**Tras medir los tiempos, rellenar la tabla:**

*TABLA1 (tiempos en milisegundos y SIN\_OPTIMIZACIÓN):*

**Pondremos “FdT” para tiempos superiores al minuto**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **n** | **t Bucle1** | **t Bucle2** | **t Bucle3** | **t Bucle4** |
| **100** | 0,0094 | 0,297 | 1,1 | 1,41 |
| **200** | 0,0203 | 1,157 | 4,85 | 10,9 |
| **400** | 0,0406 | 5,421 | 20,61 | 94 |
| **800** | 0,1 | 25,141 | 90,76 | 687 |
| **1600** | 0,2125 | 99,781 | 395,5 | 5422 |
| **3200** | 0,4453 | 482,218 | 1713,54 | 43372 |
| **6400** | 0,9562 | 1828 | 7459 | FdT |
| **12800** | 2,1438 | 8467 | 29931 | FdT |
| **25600** | 4,2986 | 36453 | FdT | FdT |
| **51200** | 9,0894 | FdT | FdT | FdT |

**Razone si los diferentes tiempos obtenidos concuerdan con lo esperado, según la complejidad temporal estudiada para los cuatro casos.**

Los resultados obtenidos concuerdan con lo esperado ya que el algoritmo del programa:

* Bucle1 es de complejidad casi lineal (), es decir, si en cada iteración del bucle principal se duplica el tamaño del problema, el tiempo de ejecución de una iteración a la siguiente se ve multiplicado por un factor de .
* Bucle2 y Bucle3 son de complejidad casi cuadrática (), es decir, si en cada iteración del bucle principal se duplica el tamaño del problema, el tiempo de ejecución de una iteración a la siguiente se ve multiplicado por un factor de .
* Bucle4 es de complejidad cúbica (), es decir, si en cada iteración del bucle principal se duplica el tamaño del problema, el tiempo de ejecución de una iteración a la siguiente se ve multiplicado por un factor de .

Los tiempos obtenidos para todos los programas cumplen con el factor calculado para cada uno de ellos, pero no dejan de ser aproximaciones en milisegundos, así que el prescindir de un número de decimales entre mediciones tiene el inconveniente de no dar un resultado exacto de aplicar el factor dicho antes comparando con los resultados recogidos en la tabla.

(Lo mismo para el resto de tablas siguientes, son aproximaciones de los tiempos de ejecución en milisegundos)

**Ejercicio 2:**

**Implementar tres nuevas clases Bucle5, Bucle6 y Bucle7, que simulen algoritmos iterativos con una complejidad , y respectivamente.**

**Tras implementar las clases, mida sus tiempos de ejecución y rellene la tabla:**

*TABLA2 (tiempos en milisegundos y SIN\_OPTIMIZACIÓN):*

**Pondremos “FdT” para tiempos superiores al minuto**

|  |  |  |  |
| --- | --- | --- | --- |
| **n** | **t Bucle5** | **t Bucle6** | **t Bucle7** |
| **100** | 6,3 | 78 | 657 |
| **200** | 28,2 | 672 | 10172 |
| **400** | 141 | 5968 | FdT |
| **800** | 688 | 52180 | FdT |
| **1600** | 3203 | FdT | FdT |
| **3200** | 15061 | FdT | FdT |
| **6400** | FdT | FdT | FdT |

**Razone si los diferentes tiempos obtenidos concuerdan con lo esperado, según la complejidad temporal de los tres casos.**

Los resultados obtenidos concuerdan con lo esperado ya que el algoritmo del programa:

* Bucle5 es de complejidad , es decir, si en cada iteración del bucle principal se duplica el tamaño del problema, el tiempo de ejecución de una iteración a la siguiente se ve multiplicado por un factor de .
* Bucle6 es de complejidad , es decir, si en cada iteración del bucle principal se duplica el tamaño del problema, el tiempo de ejecución de una iteración a la siguiente se ve multiplicado por un factor de .
* Bucle7 es de complejidad , es decir, si en cada iteración del bucle principal se duplica el tamaño del problema, el tiempo de ejecución de una iteración a la siguiente se ve multiplicado por un factor de .

**Ejercicio 3:**

**Tras medir los tiempos, rellenar la tabla:**

*TABLA3 (tiempos en milisegundos y SIN\_OPTIMIZACIÓN):*

**Pondremos “FdT” para tiempos superiores al minuto**

|  |  |  |  |
| --- | --- | --- | --- |
| **n** | **t Bucle1(t1)** | **t Bucle2(t2)** | **t1/t2** |
| **100** | 0,0094 | 0,297 | 0,031649832 |
| **200** | 0,0203 | 1,157 | 0,017545376 |
| **400** | 0,0406 | 5,421 | 0,007489393 |
| **800** | 0,1 | 25,141 | 0,003977567 |
| **1600** | 0,2125 | 99,781 | 0,002129664 |
| **3200** | 0,4453 | 482,218 | 0,000923441 |
| **6400** | 0,9562 | 1828 | 0,000523085 |
| **12800** | 2,1438 | 8467 | 0,000253195 |
| **25600** | 4,2986 | 36453 | 0,000117922 |
| **51200** | 9,0894 | FdT | #¡VALOR! |

**Razone si los diferentes tiempos y su cociente concuerda con lo esperado**

Sabemos que el algoritmo del programa Bucle1 es de una complejidad más eficiente que la del algoritmo del programa Bucle2, lo que se traduce en que Bucle1 tiene un tiempo de ejecución inferior que el de Bucle2 para cualquier tamaño de problema n.

La afirmación se respalda también de la columna t1/t2 de la tabla. Dicha columna guarda valores menores que 1, lo que indica que en la división t1/t2 el denominador es mayor que el numerador, es decir, que los tiempos de Bucle2 son superiores que los de Bucle1 reforzando la primera afirmación.

**Ejercicio 4:**

**Tras medir los tiempos, rellenar la tabla:**

*TABLA4 (tiempos en milisegundos y SIN\_OPTIMIZACIÓN):*

**Pondremos “FdT” para tiempos superiores al minuto**

|  |  |  |  |
| --- | --- | --- | --- |
| **n** | **t Bucle3(t3)** | **t Bucle2(t2)** | **t3/t2** |
| **100** | 1,1 | 0,297 | 3,7037037 |
| **200** | 4,85 | 1,157 | 4,1918755 |
| **400** | 20,61 | 5,421 | 3,8018816 |
| **800** | 90,76 | 25,141 | 3,6100394 |
| **1600** | 395,5 | 99,781 | 3,9636805 |
| **3200** | 1713,54 | 482,218 | 3,5534551 |
| **6400** | 7459 | 1828 | 4,0804158 |
| **12800** | 29931 | 8467 | 3,5350183 |
| **25600** | FdT | 36453 | #¡VALOR! |
| **51200** | FdT | FdT | #¡VALOR! |

**Razone si los diferentes tiempos y su cociente concuerda con lo esperado.**

Sabemos que, aunque ambos programas cuentan con un algoritmo de la misma complejidad, por tener misma complejidad no va a implicar que tengan un mismo tiempo de ejecución para un mismo tamaño de problema n.

Al observar los resultados recogidos en la columna t3/t2 de la tabla podemos ver que son resultados > 1, es decir, que el numerador de la división es mayor que el denominador de este, lo que se traduce en que el Bucle2 es más eficiente que el Bucle3 pese a tener la misma complejidad.

**Ejercicio 5:**

**Tras medir los tiempos, rellenar la tabla:**

*TABLA5 (tiempos en milisegundos y SIN\_OPTIMIZACIÓN):*

**Pondremos “FdT” para tiempos superiores al minuto**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **n** | **Bucle4(PY) t41** | **Bucle4(JA\_SIN) t42** | **Bucle4(JA\_CON) t43** | **t42/t41** | **t43/t42** |
| **200** | 7,8 | 10,9 | 0,125 | 1,397435897 | 0,011468 |
| **400** | 390 | 94 | 0,63 | 0,241025641 | 0,006702 |
| **800** | 3171 | 687 | 4,7 | 0,216650899 | 0,006841 |
| **1600** | 26512 | 5422 | 28,1 | 0,204511165 | 0,005183 |
| **3200** | FdT | 43372 | 156 | #¡VALOR! | 0,003597 |
| **6400** | FdT | FdT | 1000 | #¡VALOR! | #¡VALOR! |

**Razone si los diferentes tiempos y sus cocientes concuerda con lo esperado.**

Al comparar los tiempos del Bucle4 en Python con los tiempos en Java sin optimización nos damos cuenta que para un tamaño de problema n lo suficientemente pequeño, Python es más rápido que java sin optimizaciones, pero a medida que el tamaño n va creciendo el rendimiento de su versión en java es superior tanto con optimización como sin ella.

Esto viene reforzado por los valores obtenidos en la columna t42/t41 en los que, si el valor recogido es > 1 significa que el numerador es mayor que el denominador, y si es < 1 significa que el denominador es mayor que el numerador.

También observamos que la versión de Java con optimización es la más rápida de las tres, siendo esta afirmación respaldada por los valores de la columna t43/t42 los cuales nos indican que los tiempos de la versión optimizada son menores que la versión sin optimizar, y comparando los tiempos obtenidos de la versión de java optimizada con la versión en Python.