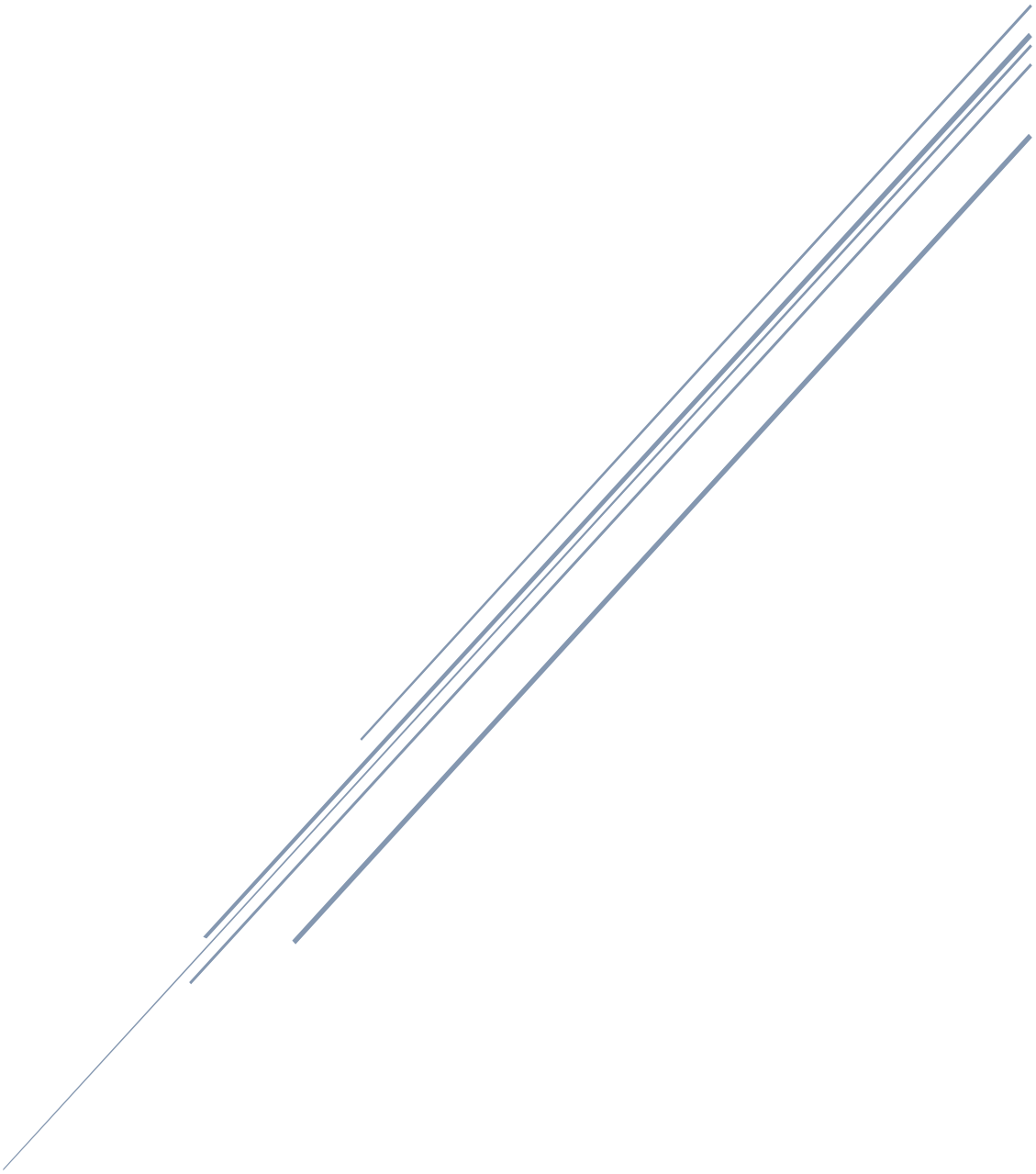
# MEDICIÓN DE TIEMPOS Y ANÁLISIS DE COMPLEJIDADES

P0, P1.1, P1.2. Algoritmia 19-20



Sofía Martín Rodríguez
UO258355

# CONTENIDO

# PRÁCTICA 0

## POTENCIA DE LAS CPUS

### TAREA 1

1. *Información sobre la CPU.*

Procesador: Intel(R) Core(TM) i7-4750HQ CPU @ 2.00GHz
Memoria instalada (RAM): 16.0 GB (15.9GB utilizable).

2. *Buscar información en userbenchmark.*

https://cpu.userbenchmark.com/SpeedTest/3709/IntelR-CoreTM-i7-4750HQ-CPU---200GHz

3. *Número máximo de operaciones enteras por unidad de tiempo.*

**Min**: 50.0
**Avg**: 1-Core 81.3
**Max**: 93.6

4. *Tiempo que tarda en finalizar el programa Benchmarking1.*

n=1048576**Time=333

5. *Número aproximado de operaciones enteras que necesitó el programa.*

(333*10E-3) * 93.6 = 31,1688 ~ 32 operaciones.

### TAREA 2

| *CPU* | Milisegundos | 1-Core (Max) | Operaciones |
|---|---|---|---|
| *I7-4790* | 216 | 118 | 25 |
| *I5-7600* | 205 | 126 | 26 |
| *I3-3220* | 267 | 90.6 | 24 |
| *I7-4750* | 333 | 93.6 | 32 |
| *I5-2500* | 221 | 120 | 27 |

### CONCLUSIONES

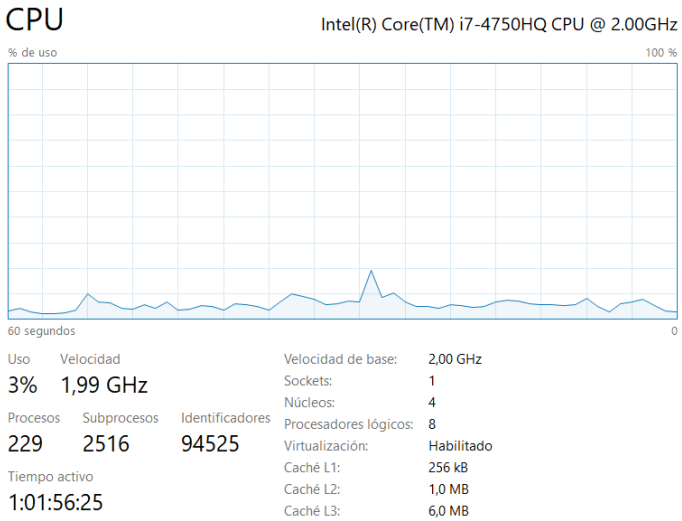
*¿Crees que podrías mezclar valores de diferentes CPUs en un estudio analítico de los tiempos de ejecución de un algoritmo?*

No, dado que cada CPU tiene unas características propias y las mediciones del rendimiento del algoritmo son diferentes en cada una de ellas. La medición de tiempos debería realizarse siempre en el mismo entorno controlado para poder obtener resultados fiables en cuanto a tiempos de ejecución. Estos tiempos dependen también del rendimiento que tenga la CPU donde se prueba, por eso siempre debería ser la misma.

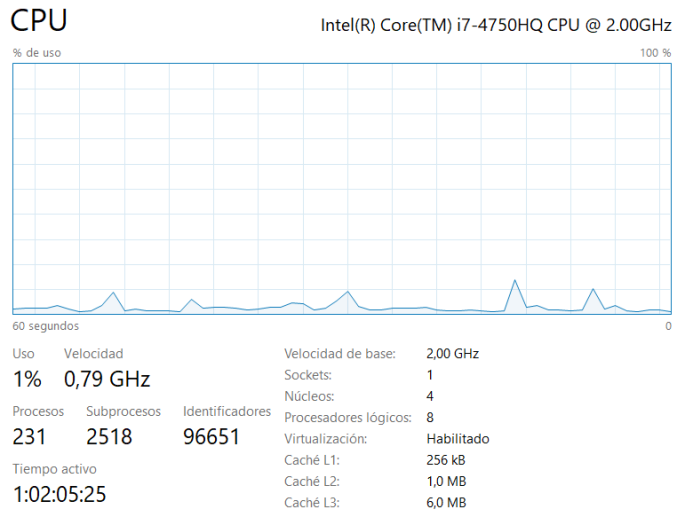
## INFLUENCIA DEL SO

### TAREA 1

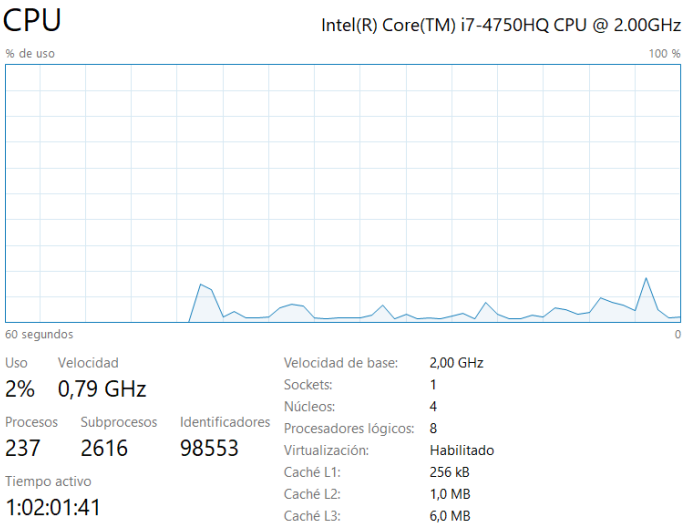
*Equilibrado*




*Economizador*




*Alto rendimiento*



### TAREA 2

*¿Qué plan de energía es el más adecuado para realizar mediciones?*

Alto rendimiento, dado que se aprovechan al máximo todo el potencial que ofrece la CPU.

*Durante la medición de un experimento muy largo, ¿podrías usar el ordenador para ver por ejemplo un vídeo de Youtube mientras tanto?*

No, el uso del navegador es una actividad que consume muchos recursos, sobre todo durante reproducciones de vídeo/audio. El rendimiento de la CPU no estaría al 100% para ejecutar el algoritmo con plena capacidad.

*¿Crees conveniente realizar varias mediciones simultáneamente en el mismo ordenador?*

No creo que sea conveniente ya que acaparas más recursos cuantas más mediciones simultáneas hagas.

## PRÁCTICA 1.1

### TOMAS DE TIEMPOS DE EJECUCIÓN

1. *¿Cuánto tiempo más podremos usar esta forma de contar?*

   292.471.208.6775360

2. *¿Qué significa que el tiempo medido sea 0?*

   El tiempo que tarda es despreciable y no supone ninguna relevancia para medir la eficiencia del algoritmo.

3. *¿A partir de qué tamaño del problema empezamos a tener resultados fiables?*

   Se empiezan a tener resultados fiables a partir de los 100k elementos.

### CRECIMIENTO DEL TAMAÑO DEL PROBLEMA

1. *¿Qué pasa con el tiempo si el tamaño del problema se multiplica por 5?*

   El tamaño del problema aumenta por cinco.

2. *¿Los tiempos obtenidos son los que se esperaban de la complejidad lineal O(n)?*

   Con vector3 no se puede deducir la complejidad porque el tiempo es mayoritariamente 0, es decir, despreciable. Cogiendo los tiempos por encima de 50, se ve que la complejidad de vector 3 comienza a ser lineal.

### TRABAJO PEDIDO

1. *Tabla 1. Métodos suma y máximo. ¿Cumplen los valores obtenidos con lo esperado?*

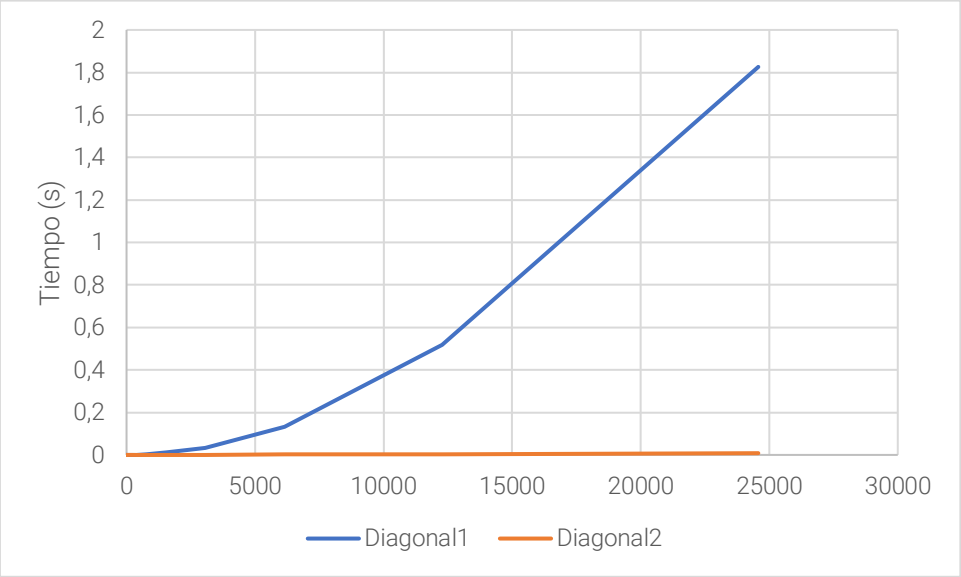
Sí, se trata de un algoritmo de complejidad lineal. Esto se puede observar también en la gráfica, donde se ve la evolución de ambos algoritmos mostrando el tiempo que tardan (en segundos) frente al tamaño del problema.

| nVeces | Tamaño | SUMA | MAXIMO |
|---|---|---|---|
| n= 100000000 | 10 | 2,96E-09 | 5,48E-09 |
| | 30 | 8,82E-09 | 1,358E-08 |
| | 90 | 2,619E-08 | 3,375E-08 |
| | 270 | 9,029E-08 | 9,807E-08 |
| | 810 | 2,7098E-07 | 2,561E-07 |
| n= 1000000 | 2430 | 8,07E-07 | 9,04E-07 |
| | 7290 | 2,436E-06 | 2,467E-06 |
| | 21870 | 7,289E-06 | 7,374E-06 |
| n= 100000 | 65610 | 0,0002179 | 0,0002725 |
| | 196830 | 0,0006633 | 0,0008148 |
| | 590490 | 0,0019588 | 0,0024507 |
| | 1771470 | 0,0063962 | 0,0078266 |
| n= 1000 | 5314410 | 0,01919 | 0,0193 |
| | 15943230 | 0,0584 | 0,05773 |
| | 47829690 | 0,19406 | 0,20299 |


2. *Tabla 2. Diagonal 1 y 2. ¿Cumplen los valores obtenidos con lo esperado?*

| nVeces | Tamaño | Diagonal1 | Diagonal2 |
|---|---|---|---|
| n= 100000000 | 3 | 1,212E-08 | 3,09E-09 |
| | 6 | 4,124E-08 | 6,95E-09 |
| | 12 | 1,3157E-07 | 1,115E-08 |
| | 24 | 4,4802E-07 | 2,08E-08 |
| n= 1000000 | 48 | 1,546E-06 | 4,2E-08 |

| | | | |
|---|---|---|---|
| | 96 | 6,266E-06 | 8,6E-08 |
| | 192 | 2,1977E-05 | 0,00000017 |
| | 384 | 8,0162E-05 | 6,23E-07 |
| n= 100000 | 768 | 0,0031092 | 0,0000123 |
| | 1536 | 0,0122981 | 0,0001748 |
| n= 1000 | 3072 | 0,03345 | 0,00051 |
| | 6144 | 0,13228 | 0,00136 |
| | 12288 | 0,51918 | 0,00312 |
| n= 10 | 24576 | 1,826 | 0,00813 |




## BENCHMARKING

1. *¿A qué se deben las diferencias de tiempos en la ejecución entre uno y otro programa?*

Mayoritariamente se deben a que Python es un lenguaje interpretado, mientras que Java es compilado. Esto significa que Java tarda menos tiempo en ejecutar los algoritmos porque …….

2. *Independientemente de los tiempos concretos, ¿Existe alguna analogía en el comportamiento de las dos implementaciones?*

Ambas se corresponden con las complejidades asociadas, aunque Python sea un total de 160 veces más lento que Java.

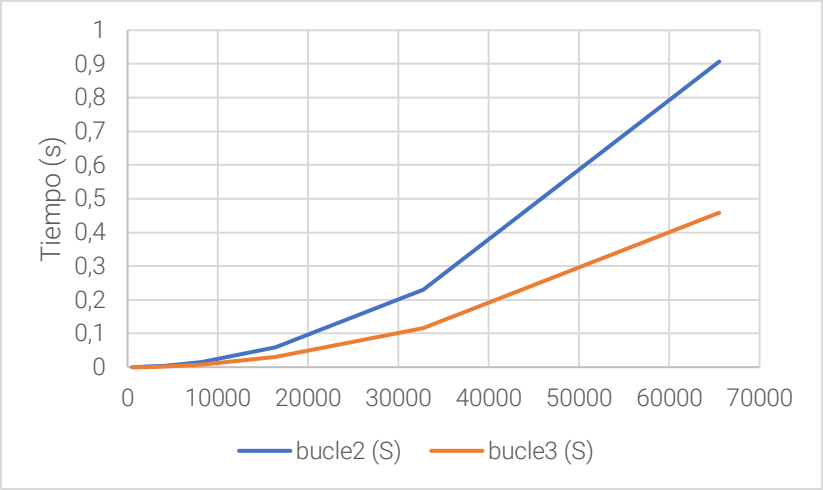
# PRÁCTICA 1.2

## TRABAJO PEDIDO

1. *Tabla 1. Dos algoritmos con misma complejidad.*

| nVeces | tamaño | bucle2 (S) | bucle3 (S) | bucle2/bucle3 |
|---|---|---|---|---|

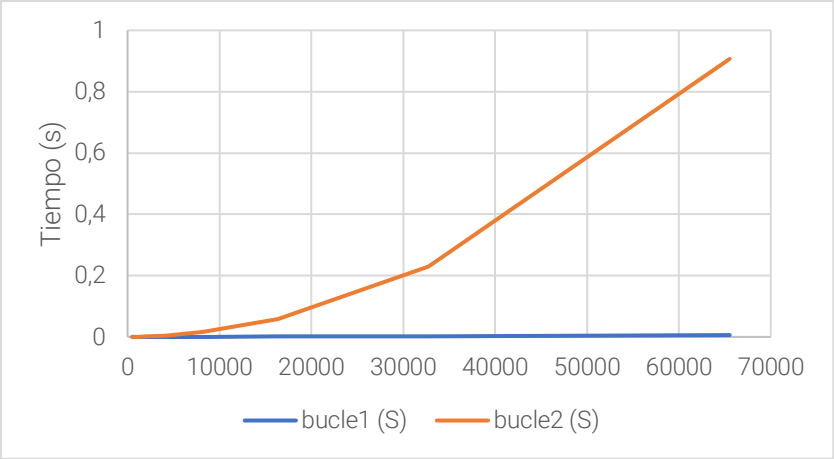
| | | | | |
|---|---|---|---|---|
| n= 1000 | 512 | 0,00016 | 0,0001 | 1,6 |
| | 1024 | 0,00045 | 0,00028 | 1,607 |
| | 2048 | 0,00132 | 0,00077 | 1,714 |
| | 4096 | 0,00437 | 0,00238 | 1,836 |
| | 8192 | 0,01552 | 0,00822 | 1,888 |
| | 16384 | 0,05883 | 0,03022 | 1,947 |
| | 32768 | 0,23024 | 0,11594 | 1,986 |
| | 65536 | 0,90687 | 0,45848 | 1,978 |
| | | | Media = | 1,813 |




La constante bucle2/bucle3 es un dato indicativo de cuál de los dos algoritmos es mejor en cuanto a rendimiento, dado que en complejidad se trata de dos algoritmos iguales de complejidad cuadrática O($n^2$). En este caso, el resultado es >1, con lo que bucle3 tiene un mejor rendimiento que bucle2. Tal y como se muestra en la gráfica, bucle3 tiene unos tiempos de ejecución menores con respecto a bucle2.

*2. Tabla 2. Dos algoritmos con distinta complejidad.*

| nVeces | tamaño | bucle1 (S) | nVeces2 | bucle2 (S) | bucle1/bucle2 |
|---|---|---|---|---|---|
| n= 100000 | 512 | 0,0000271 | n=1000 | 0,00016 | 0,169 |
| | 1024 | 0,0000605 | | 0,00045 | 0,134 |
| | 2048 | 0,0001299 | | 0,00132 | 0,098 |
| | 4096 | 0,0002901 | | 0,00437 | 0,066 |
| | 8192 | 0,0006002 | | 0,01552 | 0,039 |
| | 16384 | 0,0012931 | | 0,05883 | 0,022 |
| | 32768 | 0,0027344 | | 0,23024 | 0,012 |
| | 65536 | 0,0059239 | | 0,90687 | 0,007 |
| | | | | Media = | 0,042 |




En este caso, el algoritmo bucle1 posee una complejidad logarítmica algo peculiar, nlogn, mientras que bucle2 es de complejidad cuadrática $O(n^2)$. Según lo que nos dice la teoría, la complejidad nlogn es una complejidad quasi lineal, con lo que será mejor en rendimiento en comparación con la cuadrática. Tal y como refleja la constante bucle1/bucle2, el resultado es <1, con lo que esto demuestra que el algoritmo bucle1 es mejor. Mirando la gráfica, se ve la diferencia considerable entre los tiempos de ejecución de cada uno de los algoritmos, y vemos que bucle1 se mantiene casi lineal, y de hecho nunca llega a obtener el valor de 1. Nos queda claro que el rendimiento de bucle1 no es superable en este caso por bucle2.

*3. Tabla 3. Complejidad del resto de los algoritmos.*

| nVeces | tamaño | bucle4 (S) | bucle5 (S) | nVeces2 | tamaño2 | incógnita (S) |
|---|---|---|---|---|---|---|
| n= 1 | 16 | 0,01 | 0 | n= 10 | 256 | 0,002 |
| | 32 | 0,04 | 0,01 | | 512 | 0,022 |
| | 64 | 0,34 | 0,05 | | 1024 | 0,123 |
| | 128 | 0,19 | 0,06 | | 2048 | 0,673 |
| | 256 | 2,01 | 0,6 | | 4096 | 3,997 |
| | 512 | 23,19 | 5,12 | | 8192 | 26,79 |
| | 1024 | 416,51 | 46,61 | | 16384 | 193,364 |
| | 2048 | 4871,27 | 412,96 | n=1 | 32768 | 1381,63 |
| | | | | | Complejidad: | O(n^3) |

Bucle4 y bucle5 son dos algoritmos cuyo tiempo de ejecución es muy grande en comparación con incógnita. Bucle4 posee una complejidad $O(n^4)$ mientras que bucle5 es $O(n^3 \log n)$ e incógnita $O(n^3)$.