**PRACTICA 0**

**Nota: las mediciones de este apartado se realizaron con un Mac, que cuenta con un M1**

TAREA 1

1 Modelo procesador: Chip M1

2 Memoria RAM: 8 GB

**3 Indice medio de operaciones enteras y reales por unidad de tiempo**

Texto, Tabla

Descripción generada automáticamente

El índice medio será de (154+126)/2= 140

4 **Anota el tiempo que ha tardado en finalizar**.

Tiempo Benchmarking1 = 212 s

5 Í**ndice (aproximado) de operaciones enteras/reales que necesitó el programa**

Índice de operaciones reales/enteras = 212 \* 143 = 30313

TAREA 2

1 y 2

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| # | CPU | milisegundos | SC MIX (avg) | Operaciones (aprox) |
| 1 | I7-4700U | 285 | 71.3 | 20320,5 |
| 2 | I3-3220 | 267 | 83.3 | 22241,1 |
| 3 | I5-4590 | 219 | 98.1 | 21483,9 |
| 4 | I7-4790 | 207 | 107 | 22149 |
| 5 | Pentium Gold G5400 | 215 | 104 | 22360 |
| 6 | I7-6700K | 303 | 121 | 36.663 |
| 7 | AMD-Ryzen7 4700U | 296 | 105 | 31.080 |

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

Conclusión: no, porque en función de cada CPU el resultado del tiempo del algoritmo varía.

ACTIVIDAD 2

-Secuencial

P-alto = 213

P-adaptable = 212

P-bajo = 984

-Paralela

P-alto = 387

P-adaptable = 319

P-bajo = 322

**1. ¿Qué plan de energía crees que es el más adecuado para realizar mediciones?**

El más adecuado es el adaptable después de varias ejecuciones.

**2.** Si tuvieses que realizar la medición de un experimento muy largo, **¿podrías utilizar el ordenador para por ejemplo ver un vídeo de YouTube?**

No se debería, porque la medición ha de ser monótona, y en caso de ejecutar otros programas, el resultado de la medición podría verse afectado.

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

Si porque así se obtiene una media de tiempo más fiable.

**Practica 1.1**

**Nota: las mediciones de este apartado se realizaron con mi ordenador personal, que cuenta con un i7-6700k**

**¿Cuántos años más podremos seguir utilizando esta forma de contar?**

Podremos usar esto hasta el año 292471208,678.

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

Que no es una medida fiable

**2. ¿A partir de qué tamaño de problema (n) empezamos a obtener tiempos fiables?**

130^6

**3 y 5**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Nota: el t en milisegundos | | |  |  |
| *n* | *t suma sin rellenar matriz* | *t máximo sin rellenar matriz* | *T suma con rellenar matriz* | *T máximo con rellenar matriz* |
|  |  |
| 10 | 0 | 0 | 0 | 0 |
| 30 | 0 | 0 | 0 | 0 |
| 90 | 0 | 0 | 0 | 0 |
| 270 | 0 | 0 | 0 | 0 |
| 810 | 0 | 0 | 0 | 0 |
| 2430 | 0 | 0 | 0 | 0 |
| 7290 | 0 | 0 | 0 | 0 |
| 21870 | 0 | 0 | 0 | 0 |
| 65610 | 1 | 1 | 3 | 2 |
| 196830 | 1 | 2 | 5 | 5 |
| 590490 | 1 | 1 | 9 | 65 |
| 1771470 | 0 | 0 | 22 | 19 |
| 5314410 | 2 | 2 | 70 | 81 |
| 15943230 | 6 | 4 | 208 | 222 |
| 47829690 | 17 | 16 | 645 | 630 |
| 143489070 | 55 | 49 | 1904 | 1842 |

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

Que el tiempo aumenta de forma lineal.

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

Sí que son los esperados.

**6 y 7**

|  |  |  |
| --- | --- | --- |
| n | T diagonal 1 | T diagonal 2 |
| 3 | 0 | 0 |
| 6 | 0 | 0 |
| 12 | 0 | 0 |
| 24 | 0 | 0 |
| 48 | 0 | 0 |
| 96 | 0 | 0 |
| 192 | 1 | 1 |
| 384 | 2 | 3 |
| 768 | 1 | 1 |
| 1536 | 1 | 2 |
| 3072 | 3 | 7 |
| 6144 | 12 | 24 |
| 12288 | 46 | 97 |

A partir del 24576 se queda sin espacio en la memoria.

**¿cumplen los valores obtenidos con lo esperado?**

Sí porque la complejidad esperada es n^2.log(n)

**8**

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

A que Java es un lenguaje compilado, mientras que Python es un lenguaje interpretado.

1. **Independientemente de los tiempos concretos ¿existe alguna analogía en el comportamiento de las dos implementaciones?**

Crecen de forma similar, pero con tiempos claramente distintos.

**PRÁCTICA 1.2**

**Nota: las mediciones de este apartado se realizaron con mi ordenador personal, que cuenta con un i7-6700k**

3.

**Tabla 1**

|  |  |  |  |
| --- | --- | --- | --- |
| N | T bucle2 ns | T bucle 3 ns | T bucle2/ t bucle3 |
| 8 | 22,8 | 24 | 0,95 |
| 16 | 45,2 | 13,3 | 3,39849624 |
| 32 | 120,4 | 30,9 | 3,89644013 |
| 64 | 282 | 81,3 | 3,46863469 |
| 128 | 814,4 | 265,8 | 3,06395786 |
| 256 | 6450 | 4840 | 1,33264463 |
| 512 | 22410 | 16240 | 1,37992611 |
| 1024 | 88980 | 54360 | 1,63686534 |
| 2048 | 331870 | 183810 | 1,80550569 |
| 4096 | 1305060 | 679850 | 1,91962933 |
| Complejidad | N^2 | N^2 |  |

Aclaración:

Como el algoritmo utilizado a partir de la repetición 128 llevaba demasiado tiempo, la solución planteada en los algoritmos fue introducir un total de 2,5 millones de repeticiones en un inicio, pero al llegar al n = 128, estas repeticiones se redujesen a 100.000. De esta forma obtenemos la media de tiempo esperada, que tiende a 2.

Todos los tiempos son expresados en nanosegundos.

**Tabla 2**

|  |  |  |  |
| --- | --- | --- | --- |
| N | T bucle1 ns | T bucle 2 ns | T bucle1/ t bucle2 |
| 8 | 23,2 | 22,8 | 1,01754386 |
| 16 | 54 | 45,2 | 1,19469027 |
| 32 | 133,6 | 120,4 | 1,10963455 |
| 64 | 291,6 | 282 | 1,03404255 |
| 128 | 620 | 814,4 | 0,76129666 |
| 256 | 1470 | 6450 | 0,22790698 |
| 512 | 3150 | 22410 | 0,14056225 |
| 1024 | 6890 | 88980 | 0,07743313 |
| 2048 | 14940 | 331870 | 0,04501763 |
| 4096 | 33040 | 1305060 | 0,02531684 |
| Complejidad | n log n | N^2 |  |

Aclaración:

Como el algoritmo utilizado a partir de la repetición 128 llevaba demasiado tiempo, la solución planteada en los algoritmos fue introducir un total de 2,5 millones de repeticiones en un inicio, pero al llegar al n = 128, estas repeticiones se redujesen a 100.000. De esta forma obtenemos la media de tiempo esperada, que tiende a 0.

Todos los tiempos son expresados en nanosegundos.

**Tabla 3**

|  |  |  |  |
| --- | --- | --- | --- |
| N | T bucle4 ms | T bucle 5 ms | T incognita |
| 8 | 0,0005 | 0,00072857 | 1,01754386 |
| 16 | 0,0024 | 0,00541429 | 1,19469027 |
| 32 | 0,0048 | 0,05598571 | 1,10963455 |
| 64 | 0,00694 | 0,50015714 | 1,03404255 |
| 128 | 3 | 4 | 0,76129666 |
| 256 | 47 | 39 | 0,22790698 |
| 512 | 657 | 369 | 0,14056225 |
| 1024 | 9030 | 3203 | 0,07743313 |
| 2048 | 128048 | 27843 | 0,04501763 |
| 4096 | 1899936 | 243453 | 0,02531684 |
| Complejidad | O(n^4 ) | O(n^3 log n) | O(n^3) |

Aclaración:

Como los algoritmos realizados a partir de la n = 128 eran demasiado extensos, a partir de ese valor de n el número de repeticiones pasa a ser 1. Todos los tiempos son expresados en milisegundos.