Medición experimental de tiempos de ejecución en programas

Xiana Carrera Alonso

27 de abril de 2020

Introducción

En este informe se presenta un análisis de los tiempos de ejecución de las operaciones de trasposición y multiplicación con matrices, así como de su crecimiento en relación con los tamaños de las últimas.

Empleando el TAD matrizdinamica y la librería <time.h> se han medido los tiempos empleados en realizar dichas operaciones, que después se han representado frente al número de filas de las correspondientes matrices, con el fin de estimar su complejidad computacional.

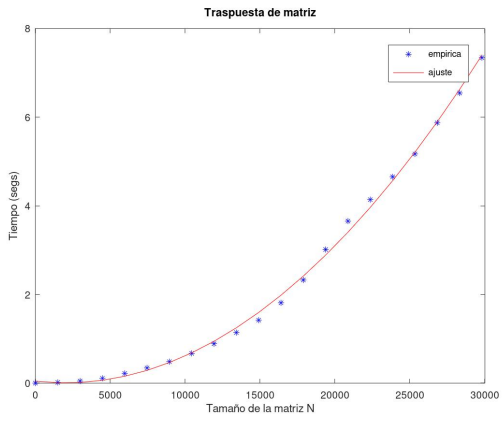
Cambios en el TAD

Se han añadido tres funciones nuevas al TAD: *inicializar*, *trasp* y *mult*, que dan valores aleatorios, trasponen y multiplican matrices, respectivamente, siempre que existan y tengan las dimensiones adecuadas.

Como el número de filas y columnas de una matriz puede alcanzar valores muy grandes (del orden de 30.000) se ha tenido que cambiar el tipo de una variable de short a long en la función *crear*. Esto se debe a que en el bucle en el que se inicializan los datos a 0 se compara la mencionada variable con el producto del número de filas con el número de columnas, que puede llegar a tomar valores superiores a 32.767, el límite para un short. No obstante, como el producto será, como mucho, de 900.000.000, entra en el rango de representación de un long (2.147.483.647).

Trasposición

Se muestra a continuación la gráfica obtenida:



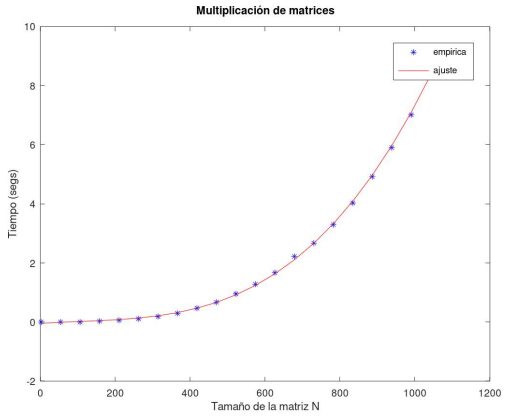
Se puede apreciar que los datos obtenidos experimentalmente se ajustan a una función polinómica de grado 2. Esto se debe a la existencia de dos bucles *for* anidados, donde uno recorre las filas de la matriz (n iteraciones) y otro las columnas ($\frac{n}{2}$ iteraciones). Por tanto, como se puede despreciar el resto de las operaciones, se tiene en conjunto $n * \frac{n}{2}$, es decir, un coste cuadrático (el coeficiente $\frac{1}{2}$ no es significativo).

En la figura aparecen 20 datos, con tamaños que empiezan en 2 (con un cálculo prácticamente instantáneo) y terminan en 29.802 (que tarda 7,3440 s), con el objetivo de no sobrepasar el límite de 30.000 filas. Se ha elegido un paso de 1.490, ya que es lo suficientemente pequeño como para tener un número de datos adecuado para obtener una estimación lo más precisa posible y lo suficientemente grande como para no sobresaturar la gráfica de puntos.

Podemos observar que aunque nos acerquemos al límite impuesto de 30.000 filas, no se llegan a alcanzar los 8 s, por lo que todos los tamaños permitidos tardarían un tiempo "razonable", si consideramos como tal aquel que está por debajo de los 10 s.

Multiplicación

Se muestra a continuación la gráfica obtenida:



En este caso, los resultados experimentales se aproximan a una función polinómica de grado 3, pues hay 3 bucles anidados: uno que recorre las filas de la matriz que almacenará el resultado (n iteraciones), otro que recorre sus columnas ($\frac{n}{2}$ iteraciones) y otro que recorre a la vez las columnnas de la primera matriz y las filas de la segunda (n iteraciones). Por lo tanto, como el resto de operaciones son despreciables, se tiene un coste de orden $n * \frac{n}{2} * n$, esto es, cúbico (de nuevo, el coeficiente $\frac{1}{2}$ no es significativo).

En la figura se pueden ver 20 puntos, que van desde un tamaño 2 (cuyo cálculo es imperceptible) hasta 1.042 (que tarda 8,5150 s). Se ha decidido marcar un máximo de 1.042 porque, al ser el crecimiento cúbico en lugar de cuadrático, los tiempos aumentan mucho más rápido. El paso es de 52, mucho más pequeño que en la trasposición para que haya suficientes datos como para poder apreciar bien los resultados, pero sin un exceso de información en la representación.

En la multiplicación, los tamaños con tiempos "razonables" son mucho menores. Así, con 1.100 filas el programa ya alcanza los 10 s, incrementándose esta cifra cada vez más rápido.

Conclusión

A través de estos dos ejemplos se ha podido comprobar el funcionamiento del cronometraje de un programa en C empleando la librería <time.h>, así como analizar su representación y su correspondencia con el estudio teórico del coste computacional.