TIEMPOS DE EJECUCIÓN

Diego Beltrán Fernández Prada. Programación II

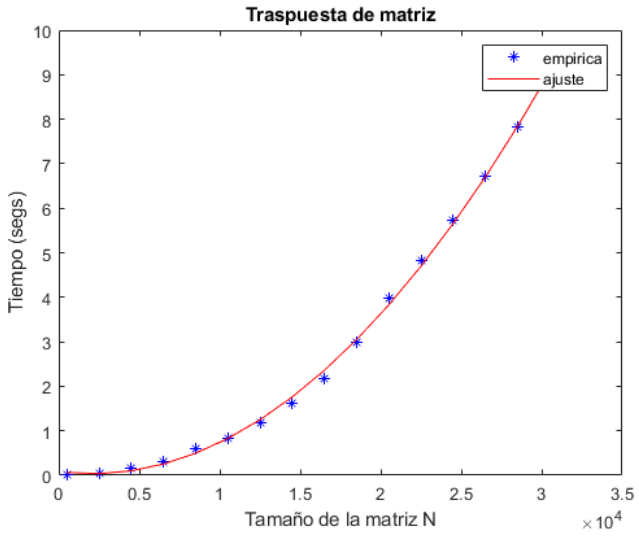
Introducción:

El propósito de este informe es abordar los resultados experimentales obtenidos en la práctica de Tiempos de ejecución. El objetivo último es evaluar el aumento de coste computacional cuando el número de cálculos a realizar en dos operaciones con un TAD de matrices crece. Las operaciones evaluadas han sido: trasposición de matrices y multiplicación de matrices.

TRASPOSICIÓN DE MATRICES

Para la trasposición de matrices añadiremos al TAD "matrizdinamica" utilizado en prácticas anteriores, dos funciones de tipo void, que nos permitirán llevar a cabo dicha trasposición. En primer lugar, añadiremos la función void inicializar (matrizD *matrix) que pasada la función matrix, la inicializará dándole valores aleatorios en las diferentes posiciones de la matriz. El valor que tomará cada posición viene dado por la siguiente expresión: element = 10.0 * (TELEMENTO) rand() / RAND_MAX;

A continuación, añadimos la función void trasp(matrizD * result, matrizD m1), la cual almacenará la matriz traspuesta de m1 en la matriz result. Para lograr una correcta implementación de esta función, se debe en primer lugar comprobar que las dimensiones de la matriz result son adecuadas para almacenar la traspuesta de m1 (las filas de result tienen que ser igual a las columnas de m1 y las columnas de result igual a las filas de m1). La implementación de esta función está definida utilizando dos bucles de tipo for. Hipótesis: Consecuentemente, la complejidad de esta función será $O(n^2)$. A continuación, se muestra la gráfica de los tiempos experimentales obtenidos para comprobar dicha hipótesis. El rango de valores analizados es: tamaño inicial:500, tamaño máximo: 30500, paso:2000. Estos valores fueron escogidos con el objetivo de visualizar tiempos inferiores a 1 segundo y superiores a 9 segundos.



Los tiempos experimentales se aproximan con gran exactitud a los tiempos teóricos de una función cuadrática, por lo que podemos confirmas nuestra hipótesis inicial: la trasposición de matrices tiene una complejidad cuadrática.

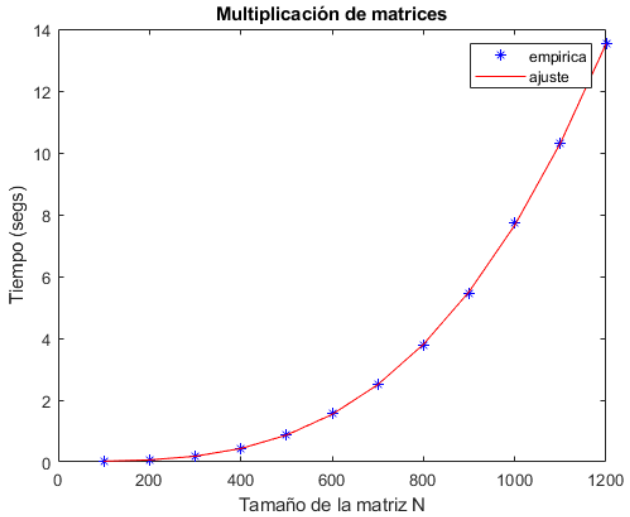
En el guion de la práctica se planteaba una pregunta en relación con el tipo de variable a utilizar: short o long. En el código mantenemos las variables short a excepción de la función void crear, donde se puede dar el caso que el valor de la variable i, tome valores por encima del rango máximo que las variables tipo short pueden hacer. Por ello dicha variable será de tipo long.

MULTIPLICACIÓN DE MATRICES

Para una correcta implementación al TAD matrizdinamica de una función que nos permita multiplicar matrices, añadiremos la función:
void mult(matrizD *result, matrizD m1, matrizD m2)

Esta función almacenará en la matriz result, el resultado de la multiplicación de las matrices m1 y m2. Para una correcta implementación de esta función, se debe hacer la siguiente comprobación: las dimensiones de las matrices permiten la multiplicación matricial (numero de columnas de m1 coinciden con las filas de m2). Para acceder a los valores de cada una de las matrices se utiliza la función recuperar. Finalmente se llamará a la función asignar para almacenar el valor correspondiente en la posición (i,j) de la matriz result. La implementación de la función void mult, cuenta con tres bucles enlazados de tipo for. Hipótesis: La complejidad de esta función es $O(n^3)$.

A continuación, se muestra la gráfica de los tiempos experimentales obtenidos para comprobar dicha hipótesis. El rango de valores seleccionados es: tamaño inicial: 100, tamaño máximo: 1200, paso:100. Este rango permite visualizar multiplicaciones con un tiempo de cálculo inferior a 1 segundo y superiores a 10 segundos.



Los tiempos experimentales se ajustan una vez más, con gran exactitud, a los tiempos teóricos, en este caso de una función cúbica. Por ello, se confirma la hipótesis planteada: la multiplicación de matrices corresponde a una función de complejidad cúbica.