Ing. de Sistemas e Informática
Estructuras de Datos y Algoritmos
Jorge Mario Londoño P.

Taller 2

Análisis de algoritmos

Ejercicios a desarrollar

1. ADT Matriz (1 punto)
Definir un tipo de data abstracto Matriz que permita representar matrices de reales de dimensión NxM. Definir las operaciones suma, multiplicación y eliminación gaussiana para el ADT Matriz. Especificar claramente los argumentos y el tipo de retorno de cada operación.

2. Implementación del ADT (1 punto)
Dar una implementación del ADT Matriz que implemente las tres operaciones definidas en el ADT. Definir un constructor por defecto que inicialice la matriz de zeros y un constructor que acepte una matriz de double para inicializar un valor partircular del ADT.
Adicionalmente, incluir como funciones de biblioteca:

- `Matriz generarMatrizIdentidad(int N)` : Retorna la matriz identidad cuadrada de dimensiones NxN (elementos de la diagonal son 1).
- `Matriz generarMatrizAleatoria(int N, int M)` : Retorna una matriz de dimension NxM cuyos elementos son valores pseudo-aleatorios.

3. Pruebas unitarias (1 punto)
Implementar en el main las pruebas unitarias de las operaciones del ADT. Considerar al menos un caso para comprobar el correcto funcionamiento de cada una de las operaciones. Nota: Comparar números en punto flotante verificando que su diferencia en valor absoluto sea inferior a un margen de tolerancia: $|x-y| < TOL$.

4. Análisis de las operaciones del ADT Matriz
Considerando como modelo de costo las operaciones a) Accesos al arreglo, b) sumas/restas, y c) multiplicaciones/divisiones, estimar el costo indicando la función tilde y el orden de crecimiento para

- La operación suma de matrices (1 punto)
- La operación producto de matrices (1 punto)
- La operación eliminación gaussiana en una matriz (1 punto)

5. Análisis experimental de las operaciones del ADT matriz

Considerando matrices cuadradas (NxN) para facilitar la descripción del tamaño de la entrada, implementar las siguientes funciones de biblioteca:

- `double medirSuma(int N)` : Genera un par de matrices aleatorias de tamaño NxN. Mide el tiempo necesario para sumarlas y lo retorna. (1 punto)
- `double medirProducto(int N)` : Genera un par de matrices aleatorias de tamaño NxN. Mide el tiempo necesario para multiplicarlas y lo retorna. (1 punto)
- `double medirElimGauss(int N)` : Genera una matriz aleatorias de tamaño Nx(N+1). Mide el tiempo necesario para hacer la eliminación gausiana y lo retorna. (1 punto)

Nota: La medición de tiempos no debe incluir la generación de la matriz.

b. Ejecutar las pruebas con valores de N=100, 200, 400, 800, 1600. Repetir la prueba de cada operación 10 veces con cada valor de N y registrar los tiempos máximo, mínimo y promedio en una tabla separada por cada operación. (1 punto)

c. Obtener una gráfica de los tiempos promedio de cada operación en función de N. Obtener la curva de mejor ajuste para cada una de las gráficas. Comprobar si el modelo corresponde con el obtenido en el punto 4. (1 punto)

Entregables

Remitir el código fuente de la solución implementada y la hoja de cálculo con los resultados de las pruebas (se aceptan Word, Excel, OpenOffice, LibreOffice, PDF). Nombrar el archivo comprimido `Taller3-<Nombre1>-<Nombre2>…` (`.zip .rar .7z` o `.tgz`). Para estandarizar la forma de invocar el programa, ubicar el método `main` en la clase `Taller3`.

En caso de utilizar estructuras de las bibliotecas del texto (`algs4.jar`) **no** anexar la biblioteca.

Grupos máximo de 3 personas.

Referencias

Sobre el algoritmo de multiplicación de matrices:

https://en.wikipedia.org/wiki/Matrix_multiplication_algorithm

Sobre el método de eliminación gaussiana

http://mathworld.wolfram.com/GaussianElimination.html

http://www.math-cs.gordon.edu/courses/ma342/handouts/gauss.pdf