Práctica Nro. 1

Optimización de algoritmos secuenciales

Esta práctica cuenta con una entrega obligatoria que los alumnos deberán entregar en grupos de dos personas.

1. Analizar el algoritmo matrices.c que resuelve la multiplicación de matrices cuadradas de N*N, ¿dónde cree se producen demoras? ¿cómo se podría optimizar el código? Optimizar el código y comparar los tiempos probando con diferentes tamaños de matrices.

2. Analizar los algoritmo SumMulMatrices.c y SumMulMatricesOpt.c que resuelven la siguiente operación (A*B) + (C*D) donde A, B, C y D son matrices cuadradas de N*N, Comparar los tiempos probando con diferentes tamaños de matrices, ¿Cuál es más rápido? ¿Por qué?

3. Describir brevemente cómo funciona el algoritmo multBloques.c que resuelve la multiplicación de matrices cuadradas de N*N utilizando la técnica por bloques. Ejecutar el algoritmo utilizando distintos tamaños de matrices y distintos tamaño de bloques, comparar los tiempos con respecto a la multiplicación de matrices optimizada del ejercicio 1. Según el tamaño de las matrices y de bloque elegido ¿Cuál es más rápido? ¿Por qué? ¿Cuál sería el tamaño de bloque óptimo para un determinado tamaño de matriz?

4. Analizar el algoritmo triangular.c que resuelve la multiplicación de una matriz cuadrada por una matriz triangular inferior ambas de N*N, ¿cómo se podría optimizar el código? Optimizar el código y comparar los tiempos probando con diferentes tamaños de matrices.

5. El algoritmo fib.c resuelve la serie de Fibonacci para un numero N dado utilizando dos métodos, el método recursivo y el método iterativo. Analizar los tiempos de ambos métodos ¿Cuál es más rápido? ¿Por qué?

6. El algoritmo funcion.c resuelve para un x dado la siguiente sumatoria:

$$\sum_{i=0}^{100\,000\,000} 2 * \frac{x^3 + 3x^2 + 3x + 2}{x^2 + 1} - i$$

El algoritmo compara dos formas de resolverla. ¿Cuál de las dos formas es más rápida? ¿Por qué?

7. El algoritmo instrucciones.c compara el tiempo de ejecución de las operaciones básicas suma (+), resta (-), multiplicación (*) y división (/) para dos operandos dados x e y. ¿Qué análisis se puede hacer de cada operación? ¿Qué ocurre si x e y son potencias de 2?

8. En función del ejercicio 7 analizar el algoritmo instrucciones2.c que resuelve la una operación binaria con dos operaciones distintas.

9. Analizar el algoritmo iterstruc.c que resuelve una multiplicación de matrices utilizando dos estructuras de control distintas. ¿Cuál de las dos estructuras de control tiende a acelerar el cómputo?

10. Dado un vector de N elementos de números reales distintos de 0, realizar la reducción por cociente consecutivo. Ejemplo:

500	**10**	**6**	**3**	**60**	**2**	**18**	**3**
500/10 = 50		6/3 = 2		60/2 = 30		18/3 = 6	
50		**2**		**30**		**6**	
50/2 = 25				30/6 = 5			
25				**5**			
25/5 = 5							
5							

Utilizar vectores con N potencias de 2 y se debe minimizar el espacio de almacenamiento.

11. Analizar el algoritmo modularidad.c que resuelve el rango sobre el promedio de los elementos de un vector de tamaño N. ¿Cómo puede optimizarse?

Pautas:

En todos los ejercicios de matrices probar con tamaños de matriz potencias de 2 (32, 64, 128, 256, 512, 1024, 2048) etc.

Compilar en Linux gcc:

gcc –o salidaEjecutable archivoFuente.c

Ejecutar:

Ejercicio 1:

./matrices N

Ejercicio 2:

./SumMulMatrices N
./SumMulMatricesOpt N

Ejercicio 3:

./multBloques r B [0|1]

r: cantidad de bloques por lado de la matriz
B: tamaño de bloque
[0|1]: = 1 o 0 para imprimir o no las matrices en pantalla

Ejemplo:

*2 bloques de 512 elementos da una matriz de N=2*512=1024, sin mostrar en pantalla:*

./multBloques 2 512 0

Ejercicio 4:

./triangular N

Ejercicio 5:

./fib N

Probar con N entre 0 y 50.

Ejercicio 6:

./funcion

Ejercicio 7:

./instrucciones

Ejercicio 8:

./instrucciones2

Ejercicio 9:

Compilar con la opción –O3

./iterstruct N R
N: tamaño de la matriz
R: cantidad de repeticiones

Ejercicio 11:

./ modularidad N
N tamaño de vector