UNIVERSIDAD DE GRANADA

E.T.S. DE INGENIERÍAS INFORMÁTICA
y DE TELECOMUNICACIÓN

Departamento de Ciencias de la
Computación e Inteligencia Artificial

Algorítmica

Guión de Prácticas

Práctica 2: Algoritmos Divide y Vencerás

Curso 2019-2020

Grado en Informática

1. Normativa

El objetivo de esta práctica es que el estudiante aprecie la utilidad de la técnica "divide y vencerás"(DyV) para resolver problemas de forma diferente a otras alternativas más sencillas o directas, como los métodos de fuerza bruta. Para ello cada equipo deberá realizar lo siguiente:

1. **Problema común:** Un problema a resolver por todos los equipos.
2. **Problema asignado:** Cada equipo tendrá que resolver un problema de la relación que se asignará de forma aleatoria.

Cada equipo tendrá que entregar:

1. Una memoria en la que se incluya el análisis teórico, empírico e híbrido de los problemas a resolver tanto en su versión por fuerza bruta (no DyV) como en su versión DyV. Debe también realizarse el estudio del umbral a partir del cual es más conveniente usar una versión u otra. La memoria no debe contener código, pero sí debe explicar las ideas generales de los algoritmos propuestos.
2. El código, comentado y ordenado, elaborado para resolver ambos problemas en sus dos versiones (fuerza bruta y DyV). El código se debe entregar indicando cómo debe ser compilado.
3. En la memoria se tiene que describir un caso de ejecución. Esta descripción debe incluir los resultados de los pasos intermedios mediante capturas de pantalla de la ejecución del algoritmo.
4. Una presentación (debería ser un resumen de la memoria), para la defensa de la práctica.

2. Problema común

Todos los equipos deberán resolver el siguiente problema:

Traspuesta de una matriz. Dada una matriz cuadrada de tamaño $n = 2^k$, diseñar el algoritmo que devuelva la traspuesta de dicha matriz. Se debe tener en cuenta que n es el número de elementos de la matriz.

3. Problema a asignar

Generadores: Los problemas 3.1, 3.2, 3.3 y 3.4 dependen de la disposición de los datos y no de los datos en sí, por lo que se proporcionan generadores de números para dichos problemas. En cada problema se indica el nombre del generador. Los generadores están publicados en Prado.

3.1. Máximo y mínimo de un vector

Diseñar, analizar la eficiencia e implementar un algoritmo que devuelva el valor máximo y el mínimo de un vector.

Nombre generador: `genera-vector.cpp`

3.2. Mezclando *k* vectores ordenados

Se tienen k vectores ordenados (de menor a mayor), cada uno con n elementos, y queremos combinarlos en un único vector ordenado (con kn elementos). Una posible alternativa por fuerza bruta consiste en, utilizando un algoritmo clásico, mezclar los dos primeros vectores, posteriormente mezclar el resultado con el tercero, y así sucesivamente. Diseñe, analice la eficiencia e implemente un algoritmo de mezcla.

Nombre generador: `genera-mezclavectores.cpp`

3.3. Serie unimodal de números

Sea un vector v de números de tamaño n, todos distintos, de forma que existe un índice p (que no es ni el primero ni el último) tal que a la izquierda de p los números están ordenados de forma creciente y a la derecha de p están ordenados de forma decreciente; es decir $\forall i, j \leq p, i < j \Rightarrow v[i] < v[j]$ y $\forall i, j \geq p, i < j \Rightarrow v[i] > v[j]$ (de forma que el máximo se encuentra en la posición p). Diseñe, analice la eficiencia e implemente un algoritmo que nos permita determinar p.

Nombre generador: `genera-unimodal.cpp`

3.4. Eliminar elementos repetidos

Dado un vector de n elementos, de los cuales algunos pueden estar duplicados, el problema es obtener otro vector donde todos los elementos duplica-

dos hayan sido eliminados. Diseñar, analizar la eficiencia e implementar un algoritmo para esta tarea.

Nombre generador: `genera-duplicados.cpp`