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 2018-2019

Doble Grado en Ingeniería Informática y Matemáticas/ADE

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 más eficiente que otras alternativas más sencillas o directas. 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 en la primera sesión de la Práctica 2.
3. **Tamaño problema**: Los problemas que están basados en el uso de un vector, el tamaño será $n = 1000$. El problema de la traspuesta de la matriz tendrán un tamaño $n = 2^k$ con $k = 8$.
4. **Generadores de números**: Los problemas `3.1`, `3.3`, `3.4` y `3.5` 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 nombres del generador. Los generadores están publicados en Prado.

Cada equipo tendrá que entregar:

1. Una memoria en la que se incluya el análisis téorico, empírico e híbrido de los problemas a resolver tanto en su versión sencilla (no DyV) como en su versión DyV. Debe también realizarse el estudio del umbral a partir del cual se debe usar el algoritmo de DyV.
2. El código elaborado para resolver ambos problemas. 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.

2 Problema común

Todos los equipos deberán resolver el siguiente problema:

Traspuesta de una matriz. Dada una matriz 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

3.1 Comparación de preferencias

Muchos sitios web intentan comparar las preferencias de dos usuarios para realizar sugerencias a partir de las preferencias de usuarios con gustos similares a los nuestros. Dado un ranking de n productos (p.ej. películas) mediante el cual los usuarios indicamos nuestras preferencias, un algoritmo puede medir la similitud de nuestras preferencias contando el número de inversiones: dos productos i y j están "invertidos" en las preferencias de A y B si el usuario A prefiere el producto i antes que el j, mientras que el usuario B prefiere el producto j antes que el i. Esto es, cuantas menos inversiones existan entre dos rankings, más similares serán las preferencias de los usuarios representados por esos rankings.

Por simplicidad podemos suponer que los productos se pueden identificar mediante enteros $1, \ldots, n$, y que uno de los rankings siempre es $1, \ldots, n$ (si no fuese así bastaría reenumerarlos) y el otro es $a_1, a_2, \ldots, a_n$, de forma que dos productos i y j están invertidos si $i < j$ pero $a_i > a_j$. De esta forma nuestra representación del problema será un vector de enteros v de tamaño n, de forma que $v[i] = a_i$, $i = 1, \ldots, n$.

Diseñar un algoritmo para medir la similitud entre dos rankings.

Nombre generador: `suffle.cpp`

3.2 Máximo y mínimo de un vector

Diseñar un algoritmo que devuelva el valor máximo y el mínimo de un vector.

3.3 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 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`

3.4 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 un algoritmo que permita determinar p.

Nombre generador: `genera-unimodal.cpp`

3.5 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 duplicados hayan sido eliminados. Comparar el algoritmo sencillo de la Práctica 1 con una implementación DyV de orden $O(n \log n)$.

Nombre generador: `genera-duplicados.cpp`