Divide y Vencerás

Problema 1: Algoritmos de ordenación: Mergesort

Implementa el algoritmo de ordenación Mergesort sobre vectores de enteros de tamaño n.

- Implementar en primer lugar el algoritmo de ordenación de inserción, para utilizarlo cuando el tamaño del caso sea menor o igual que el umbral.
- ¿Cuál es la eficiencia del algoritmo mergesort implementado?
- ¿Cuál es el mejor umbral?

NOTA: Para medir tiempos de ejecución utiliza la biblioteca time.h del lenguaje de programación C.

Problema 2: Búsqueda ternaria

Implementar el algoritmo de la búsqueda ternaria sobre un vector de enteros ordenado de tamaño n.

Dicho algoritmo es similar a la búsqueda binaria, pero en vez de dividir el vector en dos partes iguales y decidir si se continúa la búsqueda por la parte izquierda o por la derecha, este nuevo algoritmo divide el vector en tres partes iguales y decide si tiene que buscar en la parte izquierda, en la central o en la derecha.

- Implementar en primer lugar el algoritmo de búsqueda secuencial, para utilizarlo cuando el tamaño del caso sea menor o igual que el umbral.
- ¿Cuál es la eficiencia del algoritmo de búsqueda ternaria implementado?
- ¿Cuál es el mejor umbral?

Problema 3: Elemento Mayoritario de un Conjunto de Datos

Dado un conjunto C de n elementos (no necesariamente ordenables) se dice que un elemento x es mayoritario en C cuando el número de veces que aparece x en C es estrictamente mayor que n/2. Dado un conjunto de elementos se desea saber si un conjunto contiene un elemento mayoritario y devuelva tal elemento cuando exista.

- ¿Cómo sería el algoritmo clásico que se podría diseñar para resolver este problema? ¿Cuál sería su eficiencia?
- Diseñar e implementar un algoritmo basado en la técnica Divide y Vencerás para solucionar este problema. ¿Cuál sería su eficiencia?

Problema 4: Suma de los elementos de un vector, excepto el mayor y el menor

Sea T un vector desordenado de números enteros de tamaño n. Se desea calcular la suma de todos sus elementos a excepción del mayor y del menor. Por ejemplo, para el siguiente vector de 8 elementos la suma es 30.

4	1	8	7	2	6	9	3

- Diseñar e implementar el algoritmo clásico.
- Diseñar e implementar el Divide y Vencerás.
- ¿Cuál es el mejor umbral?

Problema 5: Calcular el número de elementos que tiene una matriz entre dos valores dados.

En una matriz nxn, con n potencia de 2, se almacenan una serie de números enteros positivos de forma desordenada. Se desean conocer, cuántos números hay en la matriz cuyo valor está en el intervalo cerrado [a, b], siendo a y b dos valores pasados como parámetros. Por ejemplo, para la siguiente matriz, si a=5 y b=8, el resultado debe ser 7.

$$\begin{pmatrix} 4 & 8 & 5 & 6 \\ 6 & 12 & 4 & 1 \\ 3 & 1 & 3 & 5 \\ 5 & 13 & 6 & 2 \end{pmatrix}$$

- Diseñar e implementar el algoritmo clásico.
- Diseñar e implementar el Divide y Vencerás.

Problema 6: Carreras de líquidos

Se ha celebrado una carrera de transporte de líquidos. En dicha carrera todos los participantes parten con un recipiente encima de la cabeza que tiene una capacidad C, la misma para todos. Cuando cada uno de los participantes va llegando a la meta, entrega su recipiente y se anota el dorsal y la cantidad de líquido que han conseguido transportar. Por ejemplo, si hay 10 participantes y se mide en mililitros, los valores anotados podrían ser los siguientes (dorsal y cantidad entregada):

07	03	10	01	04	06	08	02	05	09
575	842	27	135	970	10	1000	624	468	580

Para los premios, se ha de conocer el dorsal del corredor que ocupa la posición k (entre 1 y los n participantes) en función de la cantidad de líquido entregada. Diseñar un algoritmo Divide y Vencerás para resolver este problema.

NOTA: No se puede hacer una ordenación total de los participantes en función de la cantidad de líquido entregada.

Problema 7

Un heap o montículo es un árbol binario en el que cada nodo es mayor que sus hijos, si existen. Diseña un algoritmo DyV que dado un árbol binario determine si es o no un heap.

Problema 8

Sean *X* e *Y* dos vectores de enteros sin elementos repetidos de dimensión *N*. Se sabe que *X* está ordenado crecientemente y que *Y* lo está decrecientemente. Diseña un algoritmo DyV que devuelva en tiempo logarítmico, si existe, el índice *i* tal que X[i] = Y[i]

Problema 9

Dado un conjunto de puntos P en el plano, diseñar un algoritmo basado en la técnica Divide y Vencerás para hallar el par de puntos más cercanos entre sí.

NOTA: La distancia euclídea entre dos puntos i y j es $\text{sqrt}[(x_i-x_j)^2+(y_i-y_j)^2]$

Problema 10

Dados un vector de elementos distintos entre sí, ordenado en orden creciente y un rango numérico [A..B], diseña un algoritmo Divide y Vencerás que devuelva cuántos números del vector están en el rango dado.

Problema 11

Implementa un algoritmo **Divide y Vencerás** que calcule la siguiente función aplicada sobre dos vectores de enteros de igual tamaño:

$$f(U,V) = \begin{cases} -1 & si\ \forall\ i\ \in [0,n-1],\ \ U[i] = V[i] \\ el\ menor\ i & si\ \exists\ i\ \in [0,n-1]\ tal\ que\ U[i] \neq V[i] \end{cases}$$

Problema 12

Un mínimo local de un array A[0...n-1] es un elemento A(k) que satisface que A(k-1) ≥ A(k) ≤ A(k+1). Como mínimo, el vector tiene 3 elementos y A(0) ≥ A(1) y A(n-2) ≤ A(n-1). Estas condiciones garantizan la existencia de algún mínimo local.

Diseña un algoritmo basado en la técnica DyV que encuentre algún mínimo local de A que sea más rápido que el evidente de O(n) en el peor caso.

Problema 13

Dados una serie de valores almacenados en un vector ***v*** cuyo perfil se ajusta al de una curva cóncava, siendo estrictamente decrecientes hasta un determinado valor a partir del cual son estrictamente crecientes, se pide implementar un algoritmo basado en la técnica Divide y Vencerás que calcule el valor mínimo del vector.

Problema 14

Dado un vector no ordenado de n valores enteros, diseña un algoritmo basado en la técnica Divide y Vencerás que devuelva la menor de las diferencias, en valor absoluto, entre todo par de elementos. La complejidad de dicho algoritmo debe mejorar el $O(n^2)$ del algoritmo clásico. Ejemplo. Para el vector [17, 10, 14, 37, 24, 1, 8, 5] devuelve una diferencia de 2 (10-8).

$$d = \min | v_i - v_j | \text{ para todos } 1 \leq i, j \leq n , i \neq j$$