Ejercicios. Divide y vencerás

1. Escribir un pseudocódigo que resuelva el problema de la búsqueda binaria. Identificar su ecuación de recurrencia y a partir de ella su orden de complejidad.

2. Escribir un pseudocódigo que resuelva el problema de la búsqueda ternaria. Identificar su ecuación de recurrencia y a partir de ella su orden de complejidad.

3. Dado el siguiente vector, vamos a realizar sobre él una búsqueda binaria, para ver si se encuentra el número 11. Escribir las llamadas a la función buscar con sus parámetros, sabiendo que la primera es *Buscar(vector, 0, 11)*.

1	3	4	6	8	9	10	11	13	15	16	17

4. Dado el siguiente vector, vamos a realizar sobre él una búsqueda binaria, para ver si se encuentra el número 5. Escribir las llamadas a la función buscar con sus parámetros, sabiendo que la primera es *Buscar(vector, 0, 11)*.

1	3	4	6	8	9	10	11	13	15	16	17

5. Escribir un pseudocódigo que resuelva el problema de la mediana de dos vectores ordenados. Identificar su ecuación de recurrencia y a partir de ella su orden de complejidad.

6. Dados los siguientes dos vectores, queremos calcular su mediana. Escribir las llamadas a la función mediana con sus parámetros, y para cada una de ellas calcular la posición media de cada subvector. La primera llamada es *Mediana(x, 0, 7, y, 0, 7)*.

1	5	7	8	11	19	22	23

2	3	10	13	14	16	18	26

7. Escribir un pseudocódigo que resuelva el problema de la búsqueda de un elemento coincidente con su posición. Identificar su ecuación de recurrencia y a partir de ella su orden de complejidad.

8. Dado el siguiente vector, vamos buscar si existe algún elemento cuyo valor coincida con su posición (suponemos que las posiciones empiezan en 0, al igual que en C). Escribir las llamadas a la función buscar con sus parámetros, sabiendo que la primera es *Buscar(vector, 0, 11)*.

-4	-2	-1	1	2	3	5	7	9	10	13	14

9. Escribir un pseudocódigo que resuelva el problema de la transpuesta de una matriz cuadrada de nxn elementos (siendo n potencia de 2). Identificar su ecuación de recurrencia y a partir de ella su orden de complejidad.

10. Escribir un pseudocódigo que resuelva el problema de la subsecuencia de suma máxima. Identificar su ecuación de recurrencia y a partir de ella su orden de complejidad.

11. Dado el siguiente vector, queremos buscar la subsecuencia de suma máxima. Escribir las llamadas a la función suma_maxima con sus parámetros, así como los valores que estas devuelven para encontrar la solución final.

3	-1	5	1	-2	1	5	-7	1	-1	2	4

12. Vamos a ordenar un vector de números enteros mediante el algoritmo QUICKSORT. El vector a ordenar es el compuesto por los siguientes valores

8	3	7	11	1	5	2	10	4	6	9

Teniendo en cuenta que el algoritmo siempre realiza la llamada a ordenar la subtabla izquierda antes que la llamada a ordenar la subtabla derecha, ¿cuáles de los siguientes vectores son pasos intermedios en la ordenación?

a)

8	3	7	6	1	5	2	10	4	11	9

b)

4	3	7	6	1	5	2	8	10	9	11

c)

4	3	2	6	1	5	7	8	10	9	11

d)

4	3	7	6	1	5	2	8	10	11	9

e)

1	2	3	4	6	5	7	8	10	11	9

f)

1	3	2	4	6	5	7	8	10	11	9

g)

1	3	2	4	5	6	7	8	9	10	11

13. Vamos a ordenar un vector de números mediante el algoritmo MERGESORT. El vector a ordenar es el compuesto por los siguientes valores

8	3	7	11	1	5	2	10	4	6	9

¿Cuántas llamadas a la función mergesort se realizan? Enuncia todas ellas, con sus parámetros inicio y fin, en el orden en que son llamadas durante la ejecución del algoritmo. Ten en cuenta que la primera llamada es mergesort(tabla, 0, 10).

14. Escribir un pseudocódigo para el algoritmo mergesort. Identificar su ecuación de recurrencia y a partir de ella su orden de complejidad.

15. Escribir un pseudocódigo para el algoritmo quicksort. Analizar su complejidad en los casos peor y mejor.

16. Dado un vector C de tamaño n y un entero S, diseñar un algoritmo de complejidad Θ(nlogn) que determine si existen o no dos elementos de C tales que su suma sea exactamente S. (Puedes ayudarte del algoritmo de la búsqueda binaria).

17. Dos amigos matan el tiempo de espera en la cola del cine jugando a un juego muy sencillo: uno de ellos piensa un número natural positivo entre 1 y un N fijo y el otro debe adivinarlo preguntando solamente si es menor o igual que otros números que va diciendo. Diseñar un algoritmo para adivinar el número y estudiar su complejidad.

18. Sea a[1..n] un vector de enteros, con n potencia de 2. Un elemento x se denomina elemento mayoritario de a si aparece en el vector más de n/2 veces. Escribe un algoritmo capaz de decidir si un vector dado contiene un elemento mayoritario (no puede haber más de uno) y calcula su tiempo de ejecución (debes conseguir un tiempo de ejecución menor que el que se obtendría ordenando el vector y después comprobando si hay un elemento mayoritario).

19. Dado un vector V de tamaño n (con n potencia de 2), utiliza la técnica divide y vencerás para escribir un algoritmo que encuentre el máximo y el mínimo de V realizando 3n/2-2 comparaciones entre sus elementos. Calcula su tiempo de ejecución para demostrarlo. (Nota: el caso base será n=2).

20. En una librería van a analizar las preferencias de sus clientes para poder estudiar las similitudes con otros clientes y hacerles recomendaciones personalizadas. Por ello, han pedido a sus clientes que ordenen por orden de preferencia un conjunto de libros que ya han leído. Para cada par de clientes, necesitan estudiar cómo de similares son sus rankings. Una de las formas de calcular la similitud entre dos rankings es contar el número de inversiones que existen. Dos libros i y j están invertidos en las preferencias de A y B si el usuario A prefiere el libro i antes que el j (aparece antes en su ranking) mientras que el usuario B prefiere el libro j antes que el i. Cuantas menos inversiones existan entre dos rankings, más similares son las preferencias de esos clientes. Para facilitar este proceso, podemos suponer que el ranking de uno de los clientes siempre es 1, 2, …, n (cualquier par de rankings se puede transformar para que se cumpla esto). De esta forma, podemos trabajar sólo con el otro ranking, y existe una inversión cada vez que un número más pequeño está situado a la derecha de un número más grande.
Escribir un pseudocódigo que resuelva este problema.