1.– Sea A una matriz cuadrada *n x n*, conteniendo la siguiente información en cada fila *i*

$$A[i,j] = \begin{cases} 1 & \text{para } 1 \le j \le k_i \le n \\ 0 & \text{para } k_i < j \le n \end{cases}$$

Construya un algoritmo "***Divide y Vencerás***" que ordene las filas de la matriz según el número de unos que tienen, sin pasar de una complejidad *O(n log n)*. No considere la complejidad que supone intercambiar filas.

2.– Construya un algoritmo "***Divide y Vencerás***" que multiplique dos polinomios de grado *n*, con una **complejidad estrictamente menor que cuadrática.** Calcule la complejidad del algoritmo desarrollado.

3.– Sean A y B dos vectores ordenados de *n* elementos. Diseñe un algoritmo "***Divide y Vencerás***" de complejidad menor estricta que lineal que calcule la mediana del vector ordenado que resultaría de mezclar A y B.

4.– Un método de ordenación es ESTABLE si el orden relativo entre elementos iguales se mantiene después de la ordenación. Compruebe la estabilidad de los algoritmos SHELL, HEAPSORT, QUICKSORT y MGSNatFILE.

5.– Busque contraejemplos que justifiquen todos los controles que se usan en el algoritmo QUICKSORT (≤ en vez de <, al revés, etc.).

6.– Para obtener el elemento pivote x en el algoritmo QUICKSORT pueden emplearse técnicas que impidan elegir el peor elemento. ¿Qué orden de complejidad deben tener éstas como máximo, para que la complejidad del QUICKSORT no varíe? Idear una forma de hacerlo.

7.– Construya una versión iterativa del QUICKSORT.

8.– ¿Qué inconvenientes tendría hacer una versión recursiva del algoritmo MGSNatFILE?

9.– Construya un algoritmo que localice el lugar *k* que ocupará un determinado elemento *x* de un vector si éste se ordena. Basarse para ello en la parte "Divide" del QUICKSORT. Estudie su complejidad en el mejor, en el peor de los casos y en media, y sacar conclusiones.

10.– Construya un algoritmo de ordenación "in situ" basado en la mezcla natural. (Debe mezclar dos tramos naturales de un array sin usar memoria auxilar que pase de orden constante). Inténtese que la complejidad total sea < $O(n^2)$.

11.– Disponemos de un vector A de *N* elementos enteros ordenados en sentido creciente estricto, i.e., no hay elementos repetidos. Se pretende encontrar una posición i, tal que A[i] = i. Diseñe un algoritmo "***Divide y Vencerás***" que devuelva dicha posición o retorne el valor 0 cuando no exista ninguna.

12.– Calcule el orden de complejidad para los casos peor, mejor y medio. En cualquier caso estas complejidades deben ser menores que la obtenida por un algoritmo de búsqueda exhaustiva.

13.– Disponemos de un vector A de *N* enteros ordenados en sentido decreciente, no necesariamente estricto. Construya una función "***Divide y Vencerás***" tal que si existe una posición i, tal que A[i] = i, devuelva dicha posición o retorne el valor 0 cuando no exista ninguna. La complejidad ha de ser menor estricta que la lineal.

14.– 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, al menos uno en sentido estricto. Construya una **función recursiva** que decida en **tiempo logarítmico** si hay un índice i tal que

$$X[i] = Y[i]$$

Construya una **versión iterativa** de dicha función.

15.– Construya un procedimiento que seleccione los *K* menores elementos de un vector de *N* elementos dado. La complejidad del procedimiento debe ser lineal.

16.– Construya un procedimiento que seleccione el *K*-ésimo menor elemento de un vector de *N* elementos dado. La complejidad del procedimiento debe ser lineal.

17.– Tenemos un vector de *N* elementos, y pretendemos comprobar si existe algún elemento en él que se repita al menos *N/2* veces. Diseñe un algoritmo que realice dicha comprobación, y devuelva, en su caso, cuál es el elemento repetido. Si existen dos elementos, devolverá cualquiera de ellos. El orden de complejidad del algoritmo, en el caso medio, no debe ser superior al lineal si el elemento existe.

Sobre el algoritmo diseñado, calcule la complejidad en los casos mejor y peor. Así mismo, calcule la complejidad en media sobre la hipótesis de que existen exactamente *N/2* elementos iguales, y los otros *N/2* son todos distintos.

No se permiten sobre los elementos del vector comparaciones de orden, sólo de igualdad.

18.– Escriba un procedimiento "***Divide y Vencerás***" que mezcle dos árboles binarios con ***estructura de montículo***, generando otro con la misma estructura.

Calcule el orden de complejidad en el caso medio considerando como tamaño del problema el número total de nodos entre los dos árboles.

Calcule el orden de complejidad en el peor caso considerando como tamaño del problema la suma de las profundidades de los árboles. Supóngase en este caso que los dos árboles tienen la misma altura y son completos.

El algoritmo debe tener la mínima complejidad posible.

NOTA: Un árbol binario tiene ***estructura de montículo*** si todo nodo es *mayor* que sus hijos izquierdo y derecho.

19.– El rey Midas se hizo famoso por su avaricia, que le llevó en cierta ocasión a proponerles a los sabios del reino el siguiente juego:

Tenemos una bolsa de *N* monedas de oro entre las que sabemos que hay una falsa, cuyo peso es menor que las demás. Teniendo presente que únicamente disponemos de una balanza de **DOS PLATOS** perfectamente equilibrada para pesarlas, diseñe una estrategia Divide y Vencerás segura, que encuentre la moneda falsa en el menor número de pesadas

NOTA: Se ha adoptar una estrategia del tipo ***Divide y Vencerás***, aunque el número de pesadas se reduce por una leve observación. Estudie pues bien la subdivisión del problema.

¿Es realmente óptima la estrategia adoptada?

20.– Se ha de organizar el calendario de un campeonato entre *N* jugadores. Cada uno ha de jugar exactamente una vez contra cada adversario. Además, cada jugador ha de jugar exactamente un partido diario, con la excepción posible de un sólo día en el que no jugará.

a) Si *N* es potencia de dos, construya un algoritmo que permita terminar el campeonato en *N - 1* días.

b) Siendo *N* cualquiera, construya un algoritmo que planifique el campeonato en *N - 1* días si *N* es par, o en *N* días si *N* es impar y mayor que 1.

21.– Un vector de enteros de dimensión impar, decimos que es "*centradito*" si sus tres elementos centrales luego de ordenarlo son consecutivos. Desarrolle una estrategia para decidir **en tiempo lineal** si un vector dado es "*centradito*".

22.– Dadas dos matrices A y B de tamaño *2k* x *2k*, pueden multiplicarse aplicando la descomposición siguiente:

$$\begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix}\begin{pmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{pmatrix} = \begin{pmatrix} A_{11}B_{11} + A_{12}B_{21} & A_{11}B_{12} + A_{12}B_{22} \\ A_{21}B_{11} + A_{22}B_{21} & A_{21}B_{12} + A_{22}B_{22} \end{pmatrix}$$

siendo A_{ij} y B_{ij} matrices cuadradas de orden *k*.

Se pide:

a) Aplique dicha idea para desarrollar un algoritmo "***Divide y Vencerás***", para multiplicar matrices cuadradas de dimensión 2^n.

b) Idem para matrices cuadradas de dimensión $n = m\ 2^k$, con *m* y *k* naturales.

INDICACIONES Y NOTAS:

i) En ambos apartados debe cuidarse el gasto de memoria.

ii) En el apartado b) particione las matrices originales en *2k* x *2k* matrices de tamaño *m* x *m*. Aplique el método desarrollado en el apartado a) para multiplicar las matrices originales *n* x *n*, y el estándar para multiplicar los pares de submatrices *m* x *m* requeridos.

23.– Dada una matriz cuadrada de tamaño 2^n, con $n>0$. Diseñe utilizando ***divide y vencerás***, un algoritmo para rotarla 90º en el sentido de las agujas del reloj. Por ejemplo:

$$\begin{pmatrix} a & e & i & m \\ b & f & j & n \\ c & g & k & \tilde{n} \\ d & h & l & o \end{pmatrix}$$ se transforma mediante un giro de 90º en $$\begin{pmatrix} d & c & b & a \\ h & g & f & e \\ l & k & j & i \\ o & \tilde{n} & n & m \end{pmatrix}$$

24.– Diseñe un algoritmo basado en la técnica ***Divide y Vencerás*** que calcule la traspuesta de una matriz de dimensión 2^n.

25.– Diseñe un algoritmo basado en la técnica ***Divide y Vencerás*** que construya un árbol binario **casi-completo** a partir de un vector.

26.– Diseñe un algoritmo "***Divide y Vencerás***" para multiplicar dos enteros grandes.

27.– Dada una secuencia de enteros $E_1, E_2,..., E_n$. Se trata de encontrar una subsecuencia $E_j, E_{j+1},..., E_{j+k}$, para algunos j y k que verifican $1 \le j \le n$ y $1 \le j+k \le n$ tal que la suma

$$\sum_{j \le l \le k+j} E_l$$

sea **máxima**. Construya un algoritmo "***Divide y Vencerás***" que devuelva la **suma de la subsecuencia de suma máxima** con a lo sumo complejidad $O(n \log n)$.

28.– **Par de puntos más cercano**. Se tienen n puntos en el plano. Diseñe un algoritmo "***Divide y Vencerás***" que devuelva el par de puntos cuya distancia entre ambos es **mínima**.

29.– Dadas **dos cadenas ordenadas** x, de n símbolos, e y, de exactamente **dos** símbolos, se desea obtener un índice $j \in N$ tal que $0 < j < n$, $x_j = y_1$, $x_{j+1} = y_2$, o bien $j = 0$, si no existe tal j.

Siguiendo la estrategia "***Divide y Vencerás***", desarrolle algoritmos para resolver el problema propuesto en los siguientes casos:

a) Ninguna de las dos cadenas contienen símbolos repetidos.

b) Las dos cadenas pueden contener símbolos repetidos.

En ambos casos la complejidad de los algoritmos (que ha calcularse) deberá ser **logarítmica** en el peor de los casos.

Ejemplos:

a) x ="A D E F H P S T W", y ="H P", $\rightarrow j = 5$; y ="H T", $\rightarrow j = 0$.

b) x ="A D D D D R R S T W", y ="D D", $\rightarrow j = 2$; y ="R T", $\rightarrow j = 0$.

Nótese que con y ="D D", otras soluciones correctas son $j = 3$ o $j = 4$.

30.– Se consideran un vector v de N elementos y un entero positivo k. Diseñe un algoritmo que **devuelva los *k* elementos centrales** de v, luego de ordenarlo. La complejidad del algoritmo desarrollado debe ser lineal en el peor de los casos. La complejidad no es necesario que se calcule analíticamente, pero debe justificarse.

31.– Dadas dos secuencias ordenadas de n y m enteros, respectivamente. Diseñe un algoritmo ***Divide y Vencerás***, eficiente, que calcule el *k-ésimo* elemento de la mezcla resultante de las dos secuencias dadas. Calcule la complejidad del algoritmo desarrollado.

32.– Dadas dos secuencias ordenadas de n y m enteros, respectivamente. Diseñe un algoritmo ***Divide y Vencerás***, eficiente, que calcule la *mediana* de la mezcla resultante de las dos secuencias dadas. Calcule la complejidad del algoritmo desarrollado.

33.– Se tiene un vector v de números enteros distintos, con pesos asociados $p_1, \ldots, p_n$. Los pesos son valores no negativos y verifican que $\sum_{i=1}^{n} p_i = 1$. Se define la ***mediana ponderada*** del vector v como el valor $v[m]$, $1 \le m \le n$, tal que

$$\sum_{v[i]<v[m]} p_i < \frac{1}{2} \quad \text{y} \quad \sum_{v[i]\le v[m]} p_i \ge \frac{1}{2}$$

Por ejemplo, para $n = 5$, $v = [4, 2, 9, 3, 7]$ y $p = [0{,}15, 0{,}2, 0{,}3, 0{,}1, 0{,}25]$, la media ponderada es $v[5] = 7$ porque

$$\sum_{v[i]<7} p_i = p_1 + p_2 + p_4 = 0{,}15 + 0{,}2 + 0{,}1 = 0{,}45 < \frac{1}{2}, \text{ y}$$

$$\sum_{v[i]\le 7} p_i = p_1 + p_2 + p_4 + p_5 = 0{,}15 + 0{,}2 + 0{,}1 + 0{,}25 = 0{,}7 \ge \frac{1}{2}$$

Diseñe un algoritmo de tipo ***Divide y Vencerás*** que encuentre la mediana ponderada en un tiempo lineal en media. (Obsérvese que v puede no estar ordenado.) ¿Cómo se puede conseguir un coste lineal en tiempo en el caso peor?

34.– En una habitación oscura se tienen dos cajones en uno de los cuales hay n tornillos de varios tamaños, y en el otro las correspondientes n tuercas. Es necesario emparejar cada tornillo con su tuerca correspondiente, pero debido a la oscuridad no se pueden comparar tornillos con tornillos ni tuercas con tuercas, y la única comparación posible es la de intentar enroscar una tuerca en un tornillo para comprobar si es demasiado grande, demasiado pequeña, o se ajusta perfectamente al tornillo. Desarrolle un algoritmo ***Divide y Vencerás*** para emparejar los tornillos con las tuercas, que use O($n \log n$) comparaciones en término medio.

35.– En un jardín de infancia los niños se disponen a dormir la siesta después de comer. Para ello, el profesor les quita previamente los zapatos uno a uno y los guarda emparejados, por ejemplo, atando los cordones de cada par. Una vez terminada la siesta, el profesor debe poner de nuevo los zapatos a todos sus alumnos. El problema es que todos los zapatos son del mismo modelo y color, que es el del uniforme del jardín de infancia. Cuando intenta colocar un par de zapatos a un niño puede suceder que el par encaje perfectamente, que le queden grandes, o que le queden pequeños. El objetivo del profesor es que todos encajen perfectamente. Diseñe un algoritmo ***Divide y Vencerás*** que resuelva el problema suponiendo que cada par de zapatos solamente le sirve a un niño, que no use un número de comparaciones superior a O*(n log n)* en término medio. Calcule su complejidad.

36.– Se considera la clasificación de una maratón con miles de participantes, pero resulta que los datos de los participantes están ordenados por el dorsal con el que compitieron en la misma. Se desea conocer:

a) los cien primeros clasificados, no necesariamente ordenados, y
b) los diez primeros clasificados, éstos si ordenados.

Diseñe algoritmos eficientes que resuelvan estos problemas. La complejidad del algoritmo desarrollado debe ser lineal en el peor de los casos.

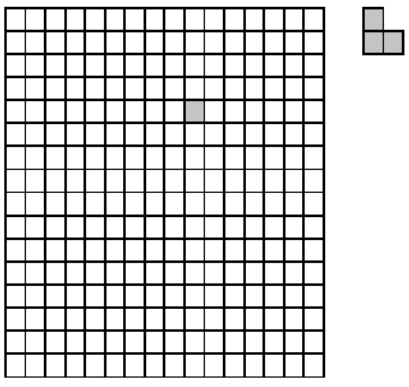
37.– Cálculo del *n*-ésimo término de la sucesión de Fibonacci.

a) Diseñe un algoritmo "***Divide y Vencerás***" que calcule x^n con un coste $O(n \log n)$, en términos del número de multiplicaciones efectuadas.

b) Sea F la matriz $\begin{pmatrix} 0 & 1 \\ 1 & 1 \end{pmatrix}$. Considere el producto del vector $(a\ b)$ por la matriz F. ¿Cuál es el resultado cuando a y b son dos términos consecutivos de la sucesión de Fibonacci.

c) Utilizando las ideas de los apartados anteriores, desarrolle un algoritmo "***Divide y Vencerás***" para calcular el *n*-ésimo término de la sucesión de Fibonacci. Calcule su coste en términos del número de operaciones aritméticas efectuadas.

38.– Tenemos un tablero cuadrado de dimensión 2^m (tiene, pues 2^{2m} celdas) y una tesela en forma de L, como puede observarse en la figura. Diseñe un algoritmo ***Divide y Vencerás***, que cubra todo el tablero utilizando teselas en forma de L como las de la figura, supuesto que ya tenemos cubierta una casilla.



39.– ***D.V. Loser*** es aficionado al juego y todas las noches va a jugar una partida de póquer. Como es un jugador responsable, para controlar sus pérdidas, anota lo que gana o pierde cada noche como un número entero de euros. Por ejemplo, en el último mes los resultados cosechados han sido los siguientes:

29 -7 14 21 30
-47 1 7 -39 23 -20 -36
-41 27 -34 7 48 35 -46
-16 32 18 5 -33 27 28
-22 1 -20 3 -42

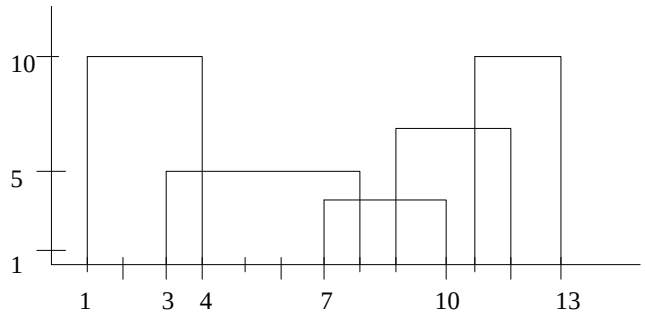
Se observa que empezó bien el mes con una ganancia de 29 euros, pero terminó con una pérdida de 42 euros. El beneficio total obtenido a lo largo de todo el mes es -47 euros.

Mirando esta información en retrospectiva, *D.V.* se da cuenta de que si hubiera empezado a jugar el día 16 y terminado el día 26, habría maximizado sus ganancias, obteniendo 105 euros, una diferencia de 155 euros con respecto a su situación actual.

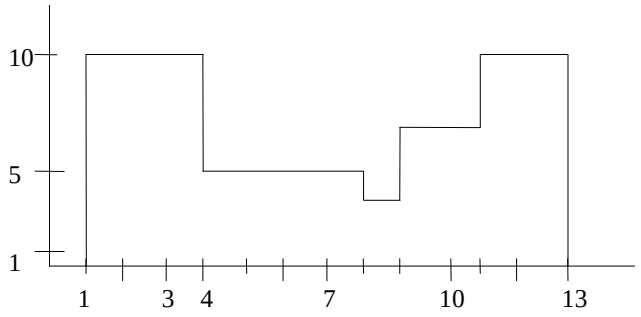
Dado un vector de ganancias/pérdidas, de longitud *N*, utilizando la técnica "***Divide y Vencerás***", se trata de devolver la secuencia de días (consecutivos) con los que se consigue **maximizar el beneficio total**, devolviendo los índices de comienzo y fin, y ese beneficio máximo. Calcule la complejidad del algoritmo desarrollado.

40.– Dada la localización exacta y la altura de varios edificios rectangulares de la ciudad, obtener la *skyline*, línea de recorte contra el cielo, que producen todos los edificios eliminando las líneas ocultas. *Ejemplo*: La entrada es una secuencia de edificios, donde un edificio se caracteriza por una tripleta de valores (x_{min},x_{max},h).

Una posible secuencia de entrada para los edificios de la siguiente figura podría ser : (7,10,3), (9,12,7), (1,4,10), (3,8,5) y (11,13,10).



Y ésta es la representación de la *skyline* de la salida que se debe producir:



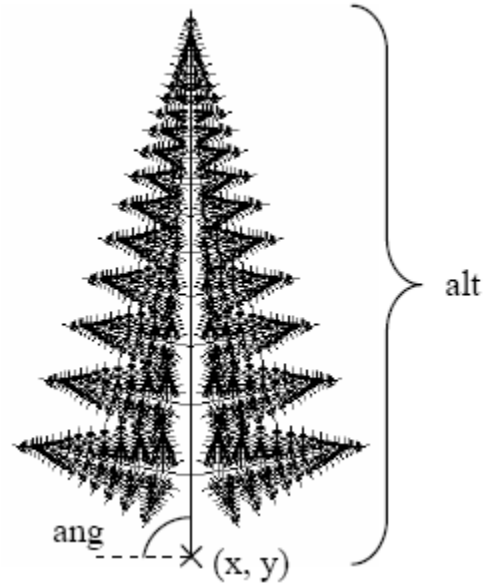
Su secuencia asociada es : (1,4,10), (4,8,5),(8,9,3), (9,11,7) y (11,13,10).

Diseñe un algoritmo ***Divide y Vencerás*** que resuelva el problema.

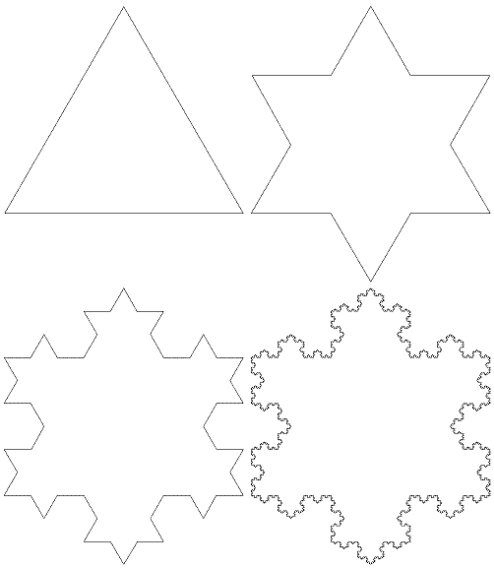
41.– Se considera la figura fractal "***helecho***" de la figura. Diseñe un algoritmo ***Divide y Vencerás*** que la dibuje. La cabecera de la función es de la forma:

void PintaHelecho (**float** x, **float** y, **float** ang, **float** alt)

que pinta un helecho de altura **alt**, cuyo tallo principal tiene la base en **(x, y)** y tiene ángulo **ang** respecto a la horizontal.



42.– Se considera la figura fractal "***copo de nieve***" que se muestra en la figura. Diseñe un algoritmo ***Divide y Vencerás*** que la dibuje.



43.– **Cierre Convexo**. Dado un conjunto de *n* puntos del plano. Diseñe un algoritmo ***Divide y Vencerás*** que compute el cierre convexo del mismo. Calcule la complejidad del algoritmo desarrollado.