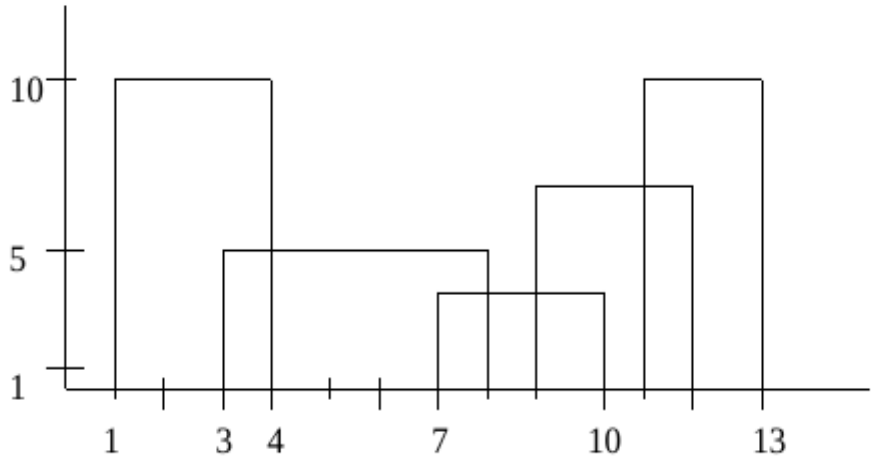
Algorítmica
Relación de Problemas: Divide y Vencerás

**Ejercicio 1**
Sea a[1..n], n≥1, un vector de enteros diferentes y ordenados crecientemente, tal que algunos de los valores pueden ser negativos. Diseñar un algoritmo que devuelva un índice natural k, 1≤ k ≤ n, talque a[k]=k, siempre que tal índice exista. El coste del algoritmo debeser O(log n) en el caso peor. Se pide diseñar el algoritmo dando todo lujo de detalles. Justificar que funciona correctamente y el coste. Repetir el problema pero para un vector de n enteros posiblemente repetidos y ordenado decrecientemente.

**Ejercicio 2**
Se está jugando un torneo entre n jugadores. Cada jugador juega una vez contra los n-1 jugadores restantes. No hay empates y los resultados de los partidos se anotan en una matriz. En general no se pueden ordenar los jugadores porque falla la transitividad, es decir, A le gana a B, B le gana a C pero C le gana a A. Estamos interesados en un orden más débil que definimos de la siguiente forma : en la secuencia ordenada que se obtiene P1,P2,..,Pn se cumple que P1 le ha ganado a P2, P2 le ha ganado a P3, etc. y Pn-1 le ha ganado a Pn. Diseñar un algoritmo que produzca esta secuencia ordenada con un coste de O(n.log n). La entrada del algoritmo es la matriz de resultados y el coste de acceder a una posición de la matriz es constante.

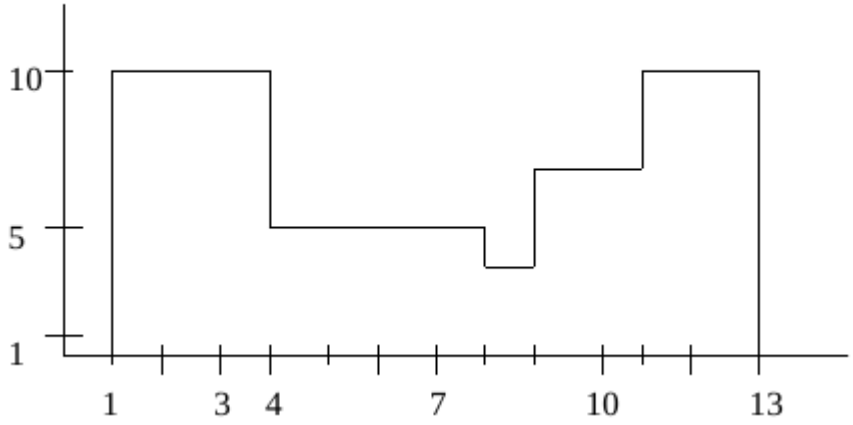
**Ejercicio 3**
Dada la localización exacta y la altura de varios edificios rectangulares de la ciudad, obtener la skyline, linea de recorte contra el cielo, que producen todos los edificios eliminando las líneas ocultas. Ejemplo : La entrada es una secuencia de edificios y un edificio se caracteriza por una tripleta de valores (xmin,xmax,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)




**Ejercicio 4**

Se dispone de un conjunto de ***n*** tornillos de diferente tamaño y sus correspondientes ***n*** tuercas, de forma que cada tornillo encaja perfectamente con una y sólo una tuerca. Dado un tornillo y una tuerca, uno es capaz de determinar si el tornillo es menor que la tuerca, mayor, o encaja exactamente. Sin embargo, no hay forma de comparar dos tornillos o dos tuercas entre ellos para decidir un orden. Se desea ordenar los dos conjuntos de forma que los elementos que ocupan la misma posición en los dos conjuntos emparejen entre sí

*IDEA: Se procederá mediante un algoritmo Divide y Vencerás de la siguiente forma. Representaremos los conjuntos de tuercas y tornillos mediante dos vectores de tamaño n que contienen las tuercas (tamaños) y los tornillos (tamaños). El algoritmo comienza escogiendo un tornillo aleatoriamente, este será el pivote para dividir los tuercas en tres conjuntos: S1 será el conjunto de tuercas que tienen un tamaño menor al tornillo ; S2: el conjunto que de tuercas que tienen igual tamaño igual al tornillo; y S3 el conjunto de tuercas que tienen tamaño mayor que la del tornillo. A continuación usando la tuerca que encaja perfectamente con el tornillo elegido, realizamos una partición similar en el conjunto de los tornillos dividiéndolo en tres conuntos (aquellos tornillos menores que la tuerca, el tornillo elegido inicialmente que encaja con la tuerca y los tornillos mayores que la tuerca). El algoritmo, de forma recursiva, aplica el mismo procedimiento a cada uno de los dos grupos que se han formado, es decir, el de los tornillos y tuercas menores que el par encontrado, y el de los mayores.*

***Ejercicio 5***

Dado un vector de números enteros positivos y negativos, encontrar el subvector de casillas contiguas que contenga la secuencia en orden creciente no monótono de longitud máxima. Por ejemplo, dado el vector {2, -1, 3, 3, 8, 7, 6, -2, 5}, la secuencia creciente no monótona de longitud máxima es {-1, 3, 3, 8}. En caso de que haya dos secuencias distintas con el mismo tamaño, bastaría con obtener una de ellas.

*IDEA: Dado un vector, se divide éste en dos de igual tamaño, en caso de que su longitud sea par, o de tamaños con diferencia de 1, en caso de que sea impar. De forma recursiva, el algoritmo aborda cada uno de los subvectores. Así se procede hasta que el vector sea de tamaño 1 ó 2, donde*

*el resultado es trivial ya que para el tamaño 1 la solución es el propio vector y para tamaño 2 basta con comprobar si el primer elemento es menor o igual que el segundo, en cuyo caso la secuencia es el vector completo, o bien sucede lo contrario, donde cualquiera de los dos subvectores (el que contiene al primer elemento o el que contiene al segundo) serían soluciones válidas. Una vez obtenida la solución para las dos mitades, el algoritmo procede a combinar las soluciones de la siguiente forma. En este caso, se sabe que la solución del vector que contiene las dos mitades será, o bien la secuencia de longitud máxima entre las devueltas en las dos mitades, o bien una secuencia que contiene tanto elementos de la parte izquierda como elementos de la parte derecha. Este segundo caso sólo se podrá dar cuando el último elemento del vector izquierdo sea igual o menor que el primer elemento del vector derecho.*
*En tal caso, para comprobar esta secuencia, podemos recorrer el subvector izquierdo partiendo desde la última casilla y retrocediendo hasta el principio siempre que el nuevo elemento que vamos analizando en cada paso sea menor o igual que el elemento anterior. Del mismo modo, podemos recorrer la parte derecha comenzando desde el principio y avanzando hasta el final siempre que el nuevo elemento analizado sea mayor o igual que el elemento anterior. Como resultado de esto, y tras unir las dos subcadenas obtenidas, tendremos una subcadena central que contiene elementos tanto de la parte derecha como de la parte izquierda. Bastará con comparar su longitud con el máximo de las subcadenas obtenidas en las dos mitades para decidir cuál es la solución al problema.*

***Ejercicio 6***
Dados n enteros cualesquiera a1,a2,...,an, necesitamos encontrar el valor de la expresión:

$$max_{0 \leqslant i \leqslant j < n} \left\{ \sum_{k=i}^{j} a_k \right\}$$

que calcula el máximo de las sumas parciales de elementos consecutivos. Como ejemplo, dados los 6 enteros (–2,11,–4,13,–5,–2) la solución al problema es 20 (suma de a1 hasta a3).
Deseamos implementar un algoritmo Divide y Vencerás de complejidad nlogn que resuelva el problema. ¿Existe algún otro algoritmo que lo resuelva en menor tiempo?

***IDEA:*** *Se divide el problema en tres subproblemas más pequeños, sobre cuyas soluciones construiremos la solución total. En este caso la subsecuencia de suma máxima puede encontrarse en uno de tres lugares. O está en la primera mitad del vector, o en la segunda, o bien contiene al punto medio del vector y se encuentra en ambas mitades. Las tres soluciones se combinan mediante el cálculo de su máximo para obtener la suma pedida. Los dos primeros casos pueden resolverse recursivamente. Respecto al tercero, podemos calcular la subsecuencia de suma máxima de la primera mitad, (desde el último al primero), obteniendo la sum máxima, y la subsecuencia de suma máxima de la segunda mitad que contenga al primer elemento mirando hasta el último, obteniendo la mejor secuencia. Estas dos secuencias pueden concatenarse para construir la subsecuencia de suma máxima que contiene al elemento central de vector.*

**Ejercicio 7**

Diseñar un algoritmo eficiente para el problema de la selección múltiple : Dado un vector A[0..n-1] de elementos, n>0, y un vector j[0..p-1] de enteros, p>0, donde $0 \leq j[0] < j[1] < ... < j[p] \leq n-1$, hay que hallar los elementos j[0]-ésimo, j[1]-ésimo, ..., j[p-1]-ésimo del vector A si éste estuviera ordenado. La solución propuesta debe ser más "eficiente" que la evidente de ordenar el vector A o que la de iterar un algoritmo de selección p veces, una vez con cada valor j[i] dado.

Ejemplo :
El vector A contiene los siguientes elementos A[0]=8, A[1]=14, A[2]=5, A[3]=7, A[4]=3, A[5]=1, A[6]=25 y A[7]=2. El vector j, qué está ordenado crecientemente, contiene los índices de los elementos de A que se quieren obtener si A estuviera ordenado. Así, j[0]=2, j[1]=5 y j[2]=7. El resultado nos ha de dar el valor de A que ocuparía la posición 2 si A estuviera ordenado, o sea Res[0]=3, y el valor de A que ocuparía la posición 5 si A estuviera ordenado, o sea Res[1]=8 y el valor de A que ocuparía la posición 7 si A estuviera ordenado, o sea Res[2]=25.