ORDENACIÓN POR MEZCLAS: MERGESORT.

La idea básica de este método de ordenación es la mezcla(*merge*) de listas ya ordenadas. Este algoritmo sigue la estrategia típica de los algoritmos *divide y vence*. Los pasos que sigue se basan en *dividir* el problema de ordenar `n` elementos en dos subproblemas mas pequeños, de tamaño mitad, de tal forma que una vez ordenada cada mitad se mezclan para así resolver el problema original. Con mas detalle: *se ordena la primera mitad de la lista, se ordena la segunda mitad de la lista y una vez ordenadas su mezcla da lugar a una lista de elementos ordenada. A su vez, la ordenación de la sublista mitad sigue los mismos pasos, ordenar la primera mitad, ordenar la segunda mitad y mezclar*. La sucesiva división de la lista actual en dos hace que el problema (número de elementos) cada vez sea más pequeño; así hasta que la lista actual tiene 1 elemento y por tanto se considera ordenada, es el *caso base.* A partir de dos sublistas de un número mínimo de elementos, empiezan las mezclas de pares de sublistas ordenadas, dando cada vez lugar a sublistas ordenadas de cada vez mas elementos (el doble de la anterior), hasta alcanzar la lista completa.

EJEMPLO 1

Seguir la estrategia del algoritmo mergesort para ordenar la lista:

9 1 3 5 10 4 6

Las Figura 1 muestra las sucesivas divisiones que origina el algoritmo. Cada división se corresponde con una llamada recursiva, por lo que a la vez queda reflejado el árbol de llamadas recursivas.

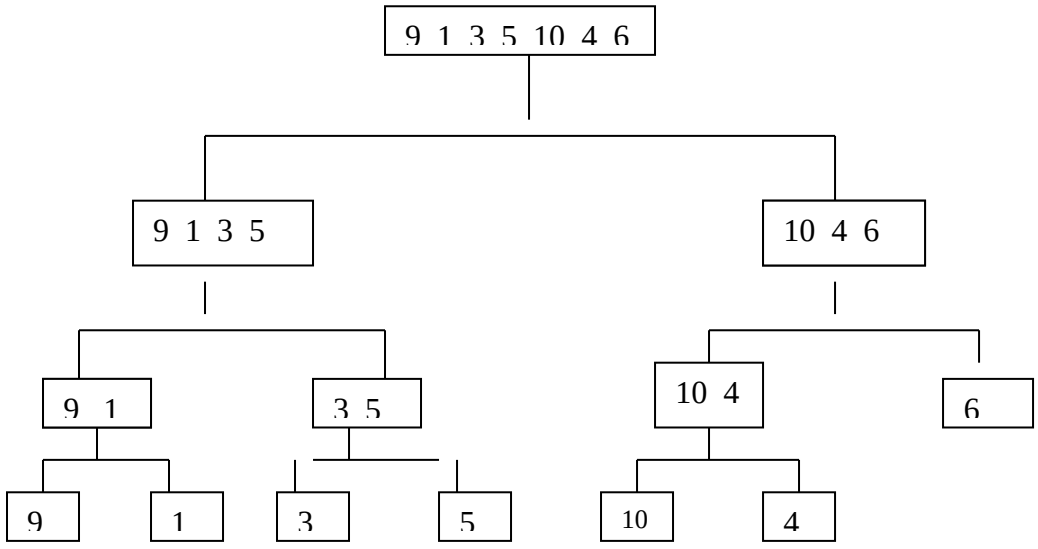


Figura 1. Sucesivas divisiones de una lista por algoritmo mergesort.

La mezcla comienza con las sublistas de un solo elemento, que dan lugar a otra sublista del doble de elementos ordenados. El proceso continúa hasta que se construye la lista ordenada. La Figura 2 muestra qué sublistas se mezclan hasta que el proceso se propaga a la raíz de las llamadas recursivas y la lista queda ordenada.

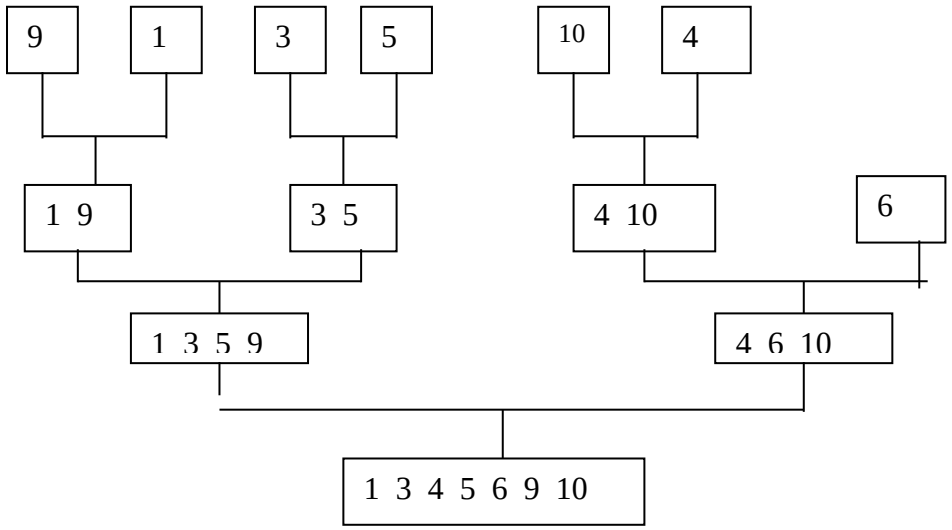


Figura 2 Progresivas mezclas de sublistas, de arriba abajo.

Algoritmo mergesort

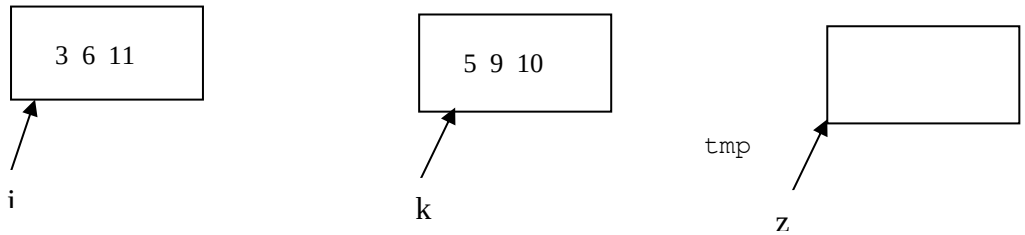
Este algoritmo de ordenación se diseña fácilmente con ayuda de las llamadas recursivas para dividir las listas en dos mitades; posteriormente se invoca al método de mezcla de dos listas ordenadas. La delimitación de las dos listas se hace con tres índices: `primero`, `central` y `ultimo`. Estos apuntan a los elementos del array de igual significado que los identificadores. Así si se tiene una lista de 10 elementos los valores de los índices:

```
primero = 0; ultimo = 9;  central = (primero+ultimo)/2 = 4
```

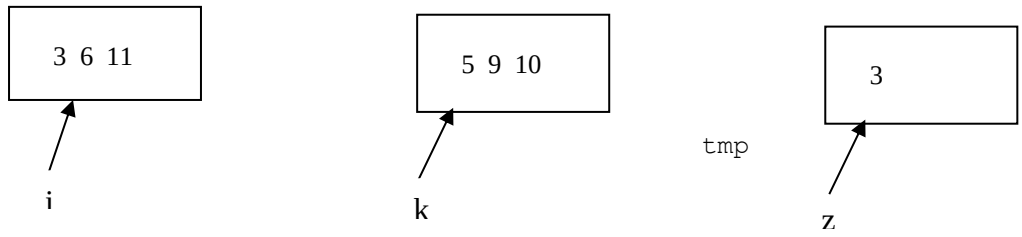
La primera sublista comprende los elementos $a_0 \ldots a_4$; y la segunda los elementos siguientes $a_{4+1} \ldots a_9$. Los pasos del algoritmo para el array `a` :

```
mergesort(a, primero, ultimo)
  si (primero < ultimo) Entonces
    central = (primero+ultimo)/2
    mergesort(a, primero, central);  ordena primera mitad
    mergesort(a, central+1, ultimo); ordena segunda mitad
    mezcla(a, primero, central, ultimo); fusiona las dos sublista
  fin_si
fin
```

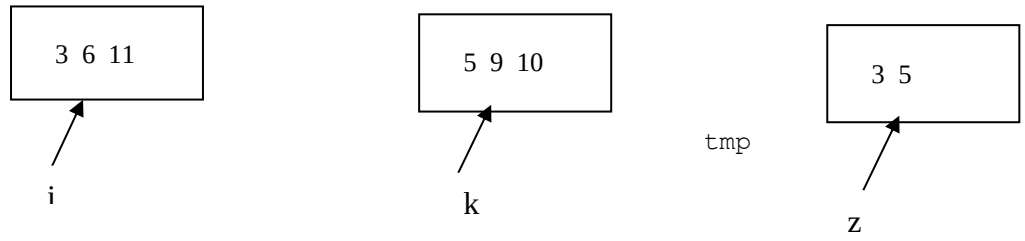
El algoritmo de mezcla utiliza un vector auxiliar, `tmp[]`, para realizar la fusión entre dos sublistas ordenadas, que se encuentran en el vector `a[]`, delimitadas por los índices `izda`, `medio`, `drcha`. A partir de estos índices se pueden recorrer las sublistas como se muestra en la Figura 5.8 con las variables `i`, `k`. En cada pasada del algoritmo de mezcla se compara `a[i]` y `a[k]`, el menor se copia en el vector auxiliar, `tmp[]`, y avanzan los índices de la sublista y del vector auxiliar. La secuencia de Figuras 5.8 muestra la mezcla de dos sublistas ordenadas.



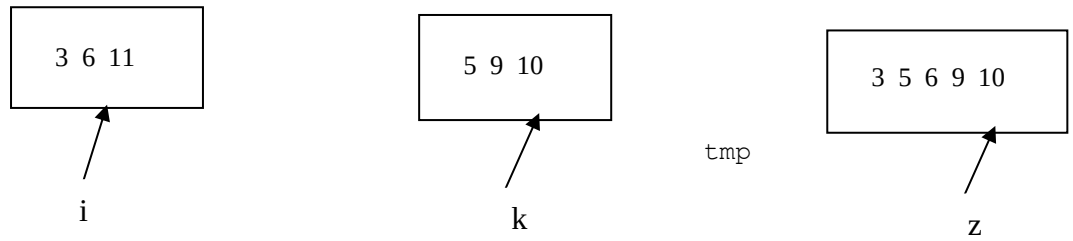
a) Punto de partida en la mezcla de dos sublistas ordenadas.



b) Primera pasada, se copia el elemento `a[i]` en `tmp[z]` y avanzan `i, z`.



c) Segunda pasada, se copia el elemento a[k] en tmp[z] y avanzan k, z.



d) Índices y vector auxiliar después de 5 pasadas.

Figura 3. Mezcla de sublistas ordenadas.

El algoritmo de mezcla es lineal, debido a que hay que realizar tantas pasadas como número de elementos, en cada pasada se realiza una comparación y una asignación (complejidad constante, `O(1)`). El número de pasadas que realiza el algoritmo mergesort es igual a la parte entera de $\log_2 n$. Se puede concluir que el tiempo de este algoritmo de ordenación es `O(n log n)`.

Codificación

El tipo de datos del array ha de ser cualquier tipo *comparable*. El método `mezcla()`, una vez realizada ésta, copia el array auxiliar `tmp[]` en el `a[]` con la utilidad `arraycopy()` de la clase `System`.

```
static void mergesort(double [] a, int primero,  int ultimo)
{
  int central;
  if (primero < ultimo)
  {
    central = (primero + ultimo)/2;
    mergesort(a, primero, central);
    mergesort(a, central+1, ultimo);
    mezcla(a, primero, central, ultimo);
  }
}
// mezcla de dos sublistas ordenadas
static void mezcla(double[] a, int izda, int medio, int drcha)
{
  double [] tmp = new double[a.length];
  int i, k, z;
  i = z = izda;
  k = medio + 1;
  // bucle para la mezcla, utiliza tmp[] como array auxiliar
  while (i <= medio && k <= drcha)
  {
    if (a[i] <= a[k])
      tmp[z++] = a[i++];
    else
      tmp[z++] = a[k++];
  }
  // se mueven elementos no mezclados de sublistas
  while (i <= medio)
    tmp[z++] = a[i++];
  while (k <= drcha)
    tmp[z++] = a[k++];
  // Copia de elementos de tmp[] al array a[]
  System.arraycopy(tmp, izda, a, izda, drcha-izda+1);
}
```

La llamada al método es: `mergesort(a, 0, a.length - 1)`.