QuickSort

Que es?

Es un método de ordenamiento de listas, el cual, debido a su algoritmo, el tiempo en el que se ejecuta es corto y es el método más rápido de ordenamiento, como que el de burbuja y otros más.

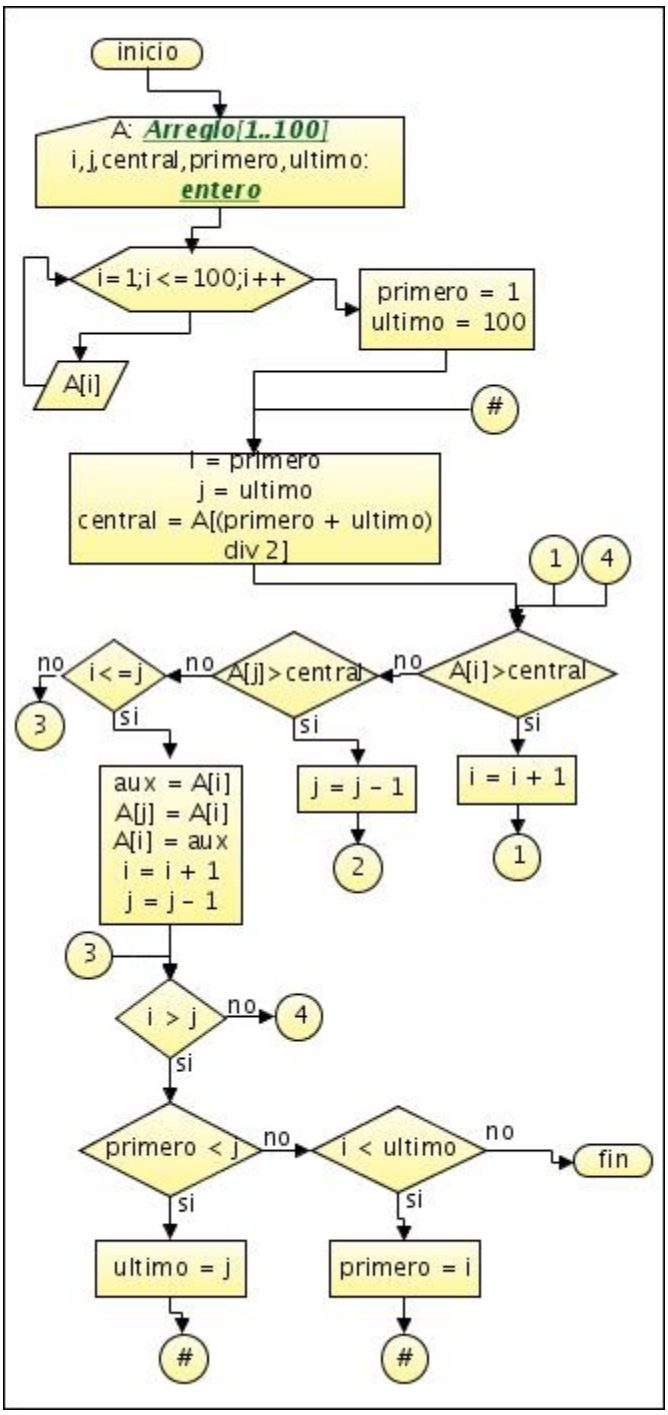
Codigo

```
public class QuickSortClass {
   public static void quickSort(int[] vector, int izquierda, int derecha) {
      int pivote = vector[izquierda];
      int i = izquierda;
      int j = derecha;
      int auxIntercambio;
      while (i < j) {
         while (vector[i] <= pivote && i < j) {
            i++;
         }
         while (vector[j] > pivote) {
            j--;
         }
         if (i < j) {
            auxIntercambio = vector[i];
            vector[i] = vector[j];
            vector[j] = auxIntercambio;
         }
      }
      vector[izquierda] = vector[j];
      vector[j] = pivote;
      if (izquierda < j - 1) {
         quickSort(vector, izquierda, j - 1);
      }
      if (j + 1 < derecha) {
         quickSort(vector, j + 1, derecha);
      }
   }

   public static void main(String[] args) {
      int[] numeros = new int[40];
      Random rnd = new Random();
      System.out.println("Vector desordenado");
      for (int i = 0; i < numeros.length; i++) {
         numeros[i] = rnd.nextInt(50);
         System.out.print(numeros[i] + " ");
      }
      QuickSortClass.quickSort(numeros, 0, numeros.length - 1);
      System.out.println("\nVector Ordenado");
      for (int n : numeros) {
         System.out.print(n + " ");
      }

   }
}
```

Diagrama de flujo



Como funciona?

- Elegir un elemento de la lista de elementos a ordenar, al que llamaremos pivote.
- Mover los demás elementos de la lista a cada lado del pivote, de manera que a un lado queden todos los menores que él, y al otro los mayores. En este momento, el pivote ocupa exactamente el lugar que le corresponderá en la lista ordenada.
- La lista queda separada en dos sublistas, una formada por los elementos a la izquierda del pivote, y otra por los elementos a su derecha.
- Repetir este proceso de forma recursiva para cada sublista mientras éstas contengan más de un elemento. Una vez terminado este proceso todos los elementos estarán ordenados. Como se puede suponer, la eficiencia del algoritmo depende de la posición en la que termine el pivote elegido.
- En el mejor caso, el pivote termina en el centro de la lista, dividiéndola en dos sublistas de igual tamaño. En este caso, el orden de complejidad del algoritmo es $O(n \cdot \log n)$.
- En el peor caso, el pivote termina en un extremo de la lista. El orden de complejidad del algoritmo es entonces de $0(n^2)$. El peor caso dependerá de la implementación del algoritmo, aunque habitualmente ocurre en listas que se encuentran ordenadas, o casi ordenadas.
- En el caso promedio, el orden es $O(n \cdot \log n)$.