|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Algoritmos de ordenamiento | Burbuja | Quicksort | Shellsort | Inserción | Radix |
| Descripción | La idea general es intercambiar números pequeños al frente comparando e intercambiando con elementos adyacentes. Este procedimiento seguirá así hasta que haya ordenado todas las casillas del vector | Utilizar un pivote y ordena los elementos del arreglo según el pivote. | compara elementos separados por un espacio de varias posiciones. Esto permite que un elemento haga "pasos más grandes" hacia su posición esperada. | Toma uno por uno los elementos y avanza hacia su posición con respecto a los anteriormente ordenados hasta recorrer todo el arreglo | ordena datos procesando sus elementos de forma individual, según la posición que ocupan dentro del dato |
| Característica | Tomar el elemento mayor y compararlo posición por posición | División por un pivote | Compara e intercambia los elementos de un arreglo |  | agrupar una serie de números dependiendo del número que tengan en las unidades que se traten. |
| Ventajas | * Fácil implementación * No requiere de memoria adicional * Este método es fácil de comprender, programar y es el más extendido. * En un código reducido se realiza el ordenamiento | * No requiere memoria adicional * Su ejecución es rápida * Requiere de pocos recursos en comparación a otros métodos de ordenamiento | * Es un método eficiente para conjuntos de elementos menores a 1000 * 5 veces más rápido que bubble sort. * No requiere memoria adicional | * Fácil implementación * Requerimiento mínimo de memoria * muestra un buen rendimiento cuando se trata de una pequeña lista * es particularmente útil solo cuando se ordena una lista de pocos elementos | * es particularmente eficiente cuando se tratan con grandes grupos de números cortos * es estable, preservando la orden de elementos iguales * tiempo de ordenar cada elemento es constante, ya que no se hacen comparaciones entre elementos. |
| Desventajas | * Realiza numerosas comparaciones * Puede llegar a ser lento * es uno de los menos eficientes y por ello, normalmente, se aprende su técnica, pero no se utiliza. | * Trabaja con recursividad * Su implementación puede llegar a ser complicada * Un simple error en la implementación puede pasar sin detección, lo que provocaría un rendimiento pésimo | * Realiza numerosas comparaciones e intercambios. * No es estable ya que se puede perder el orden relativo inicial con facilidad | * Realiza numerosas comparaciones * La ordenación de la inserción no trata bien con una lista enorme | * no funciona tan bien cuando los números son muy largos |