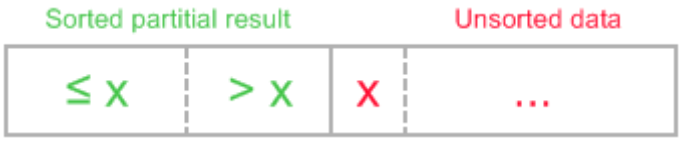
Insertion Sort

El género de inserción pertenece a los algoritmos de clasificación O (n^2). A diferencia de muchos algoritmos de clasificación con complejidad cuadrática, en realidad se aplica en la práctica para clasificar pequeñas matrices de datos. Por ejemplo, se usa para mejorar la rutina de quicksort . Algunas fuentes notan que las personas usan el mismo algoritmo para ordenar elementos, por ejemplo, la mano de cartas.

Algoritmo

El algoritmo de ordenación de inserción se asemeja al tipo de selección . La matriz es imaginaria dividida en dos partes, **una ordenada** y **una sin clasificar** . Al principio, la **parte ordenada** contiene el **primer elemento**de la matriz y la **no ordenada** contiene el resto. En cada paso, el algoritmo toma el **primer elemento** en la **parte no ordenada** y lo **inserta** en el lugar correcto del **ordenado.** Cuando la **parte no ordenada** se **vacía** , el algoritmo se *detiene* . Incompleto, el paso del algoritmo de ordenación de inserción se ve así:

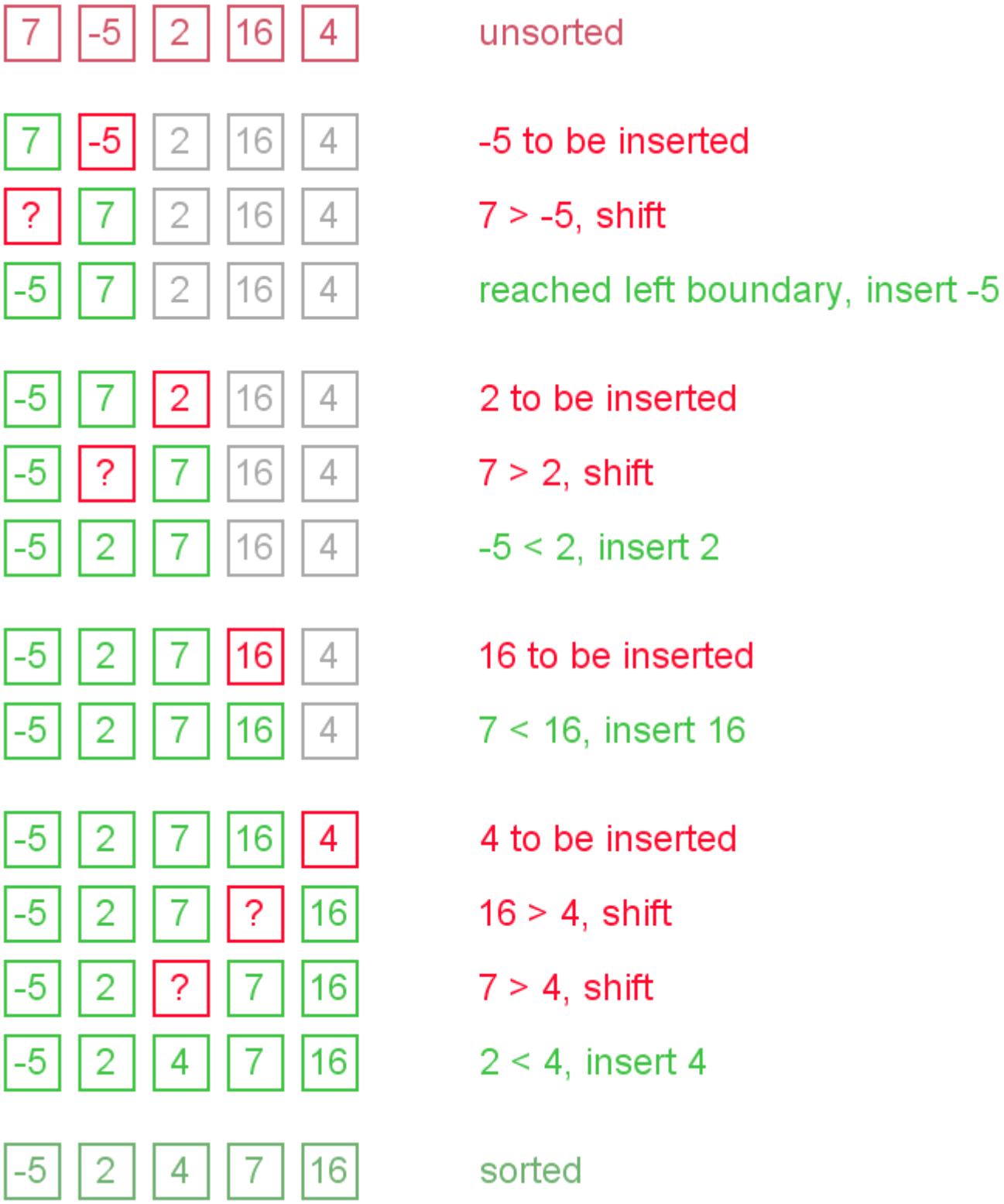


se convierte

La idea del boceto fue publicada originalmente aquí .

Veamos un ejemplo de rutina de ordenación por inserción para aclarar la idea del algoritmo.

Ejemplo. Clasifique {7, -5, 2, 16, 4} usando la ordenación por inserción.

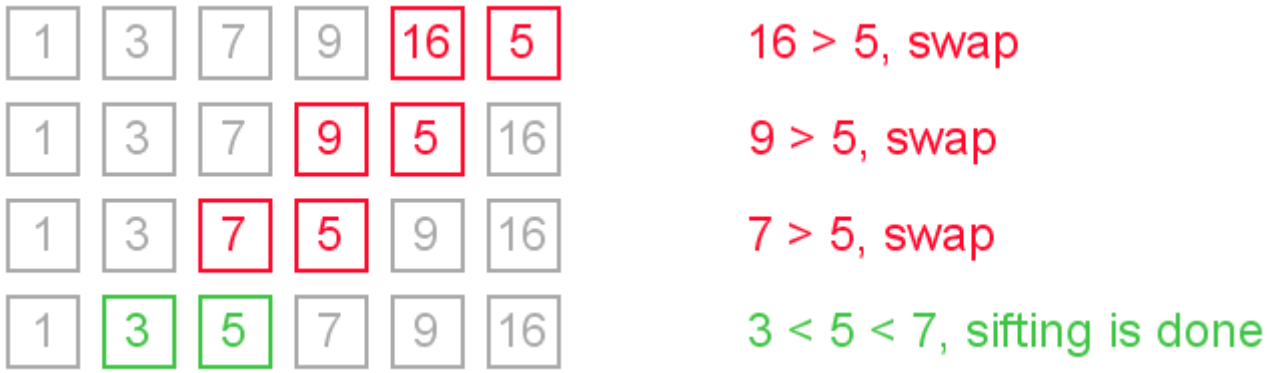


Las ideas de inserción

La operación principal del algoritmo es la **inserción** . La tarea es insertar un valor en la parte ordenada de la matriz. Veamos las variantes de cómo podemos hacerlo.

"Tamizar" usando swaps

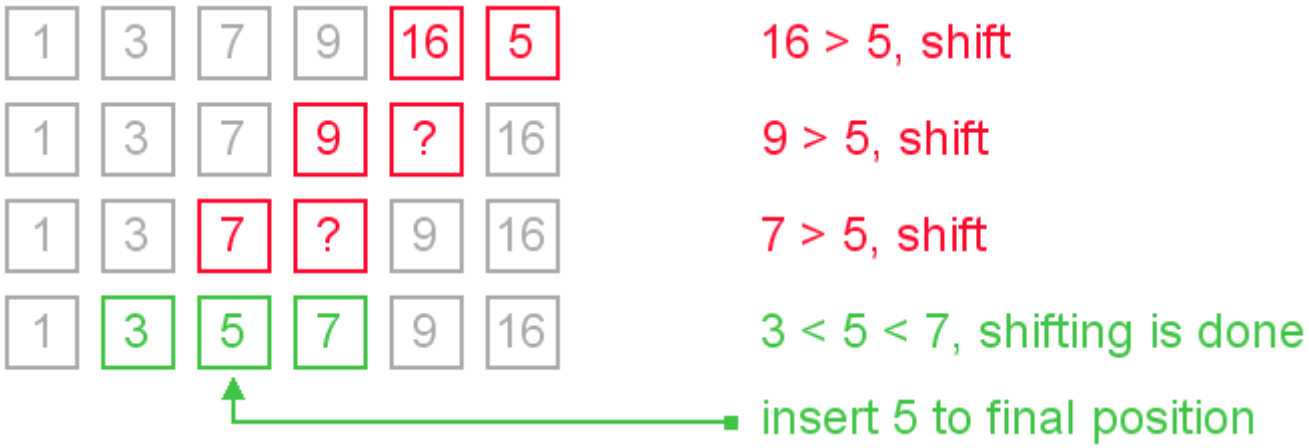
La forma más sencilla de insertar el siguiente elemento en la parte ordenada es filtrarla hasta que ocupe la posición correcta. Inicialmente, el elemento permanece justo después de la parte ordenada. En cada paso, el algoritmo compara el elemento con uno anterior y, si permanecen en orden inverso, lo intercambian. Veamos una ilustración.



Este enfoque escribe elemento tamizado a posición temporal muchas veces. La siguiente implementación elimina esas escrituras innecesarias.

Cambio en lugar de intercambio

Podemos modificar el algoritmo anterior, por lo que escribirá el elemento cernido solo en la posición correcta final. Veamos una ilustración.



Es la modificación más utilizada del tipo de inserción.

Usando búsqueda binaria

Es razonable usar el algoritmo de búsqueda binaria para encontrar un lugar adecuado para la inserción. Esta variante de la clase de inserción se denomina **clasificación de inserción binaria** . Después de encontrar la posición para la inserción, el algoritmo cambia la parte de la matriz e inserta el elemento. Esta versión tiene un número menor de comparaciones, pero la complejidad promedio general sigue siendo O (n^2). Desde un punto de vista práctico, esta mejora no es muy importante, porque el ordenamiento por inserción se usa en conjuntos de datos bastante pequeños.

Análisis de complejidad

La complejidad general de la clase de inserción es O (n^2) en promedio, independientemente del método de inserción. En las matrices casi ordenadas, la ordenación por inserción muestra un mejor rendimiento, hasta O (n) en caso de aplicar ordenación por inserción a una matriz ordenada. El número de escrituras es O (n^2) en promedio, pero el número de comparaciones puede variar según el algoritmo de inserción. Es O (n^2) cuando se usan métodos de cambio o intercambio y O (n log n) para la ordenación de inserción binaria.

Desde el punto de vista de la aplicación práctica, una complejidad promedio del tipo de inserción no es tan importante. Como se mencionó anteriormente, la ordenación por inserción se aplica a conjuntos de datos bastante pequeños (de 8 a 12 elementos). Por lo tanto, antes que nada, se debe considerar un "rendimiento práctico". En la práctica, la ordenación por inserción supera a la mayoría de los algoritmos de clasificación cuadráticos, como el tipo de selección o el tipo de burbuja .

Propiedades de ordenación de inserción

- adaptativo (el rendimiento se adapta al orden inicial de los elementos);
- estable (la ordenación por inserción conserva el orden relativo de los mismos elementos);
- in situ (requiere una cantidad constante de espacio adicional);
- en línea (se pueden agregar nuevos elementos durante el ordenamiento).

Fragmentos de código

Mostramos la idea de inserción con cambios en la implementación de Java y la idea de inserción utilizando swaps en el fragmento de código de C ++.

Implementación de Java

```
void insertionSort ( int [] arr) {
      int i, j, newValue;
      para (i = 1; i <arr.length; i ++) {
            newValue = arr [i];
            j = i;
            while (j> 0 && arr [j - 1]> newValue) {
                  arr [j] = arr [j - 1];
                  j--;
            }
            arr [j] = newValue;
      }
}
```

Implementación C ++

```
void insertionSort ( int arr [], int length ) {
      int i , j , tmp ;
      para ( i = 1; i < longitud ; i ++) {
            j = i ;
            while ( j > 0 && arr [ j - 1]> arr [ j ]) {
                  tmp = arr [ j ];
                  arr [ j ] = arr [ j - 1];
                  arr [ j - 1] = tmp ;
                  j -;
            }
      }
}
```