Insert algorithm (Insertion Sort)

Sort insertion is a sorting algorithm based on in-place comparison. Here, a sub-list is always maintained in sorted form. Inserting is inserting an element into the sorted list of children. The element is inserted in the appropriate position so that the list is still in order.

What is sort insertion (Insertion Sort)?

Sort insertion is a sorting algorithm based on in-place comparison. Here, a sub-list is always maintained in sorted form. Inserting is inserting an element into the sorted list of children. The element is inserted in the appropriate position so that the list is still in order.

With the array data structure, we imagine that: the array consists of two parts: a sorted list of children and the other part are unordered elements. The insertion sort algorithm will perform a continuous search through that array, and unordered elements will be moved and inserted into the appropriate position in the sub-list (of the same array).

This algorithm is not suitable for use with large data sets when the case complexity is worst and the average case is ? (n2) where n is the number of elements.

How to solve the insertion arrangement?

For example, we have an array of unordered elements:

The insertion sort algorithm compares the first two elements:

The algorithm found that both 14 and 33 were in ascending order. Now, 14 is in the sorted list of children.

The insertion algorithm continues to move to the next element and compares 33 and 27.

And see that 33 is not in the right position.

Algorithm to insert and swap positions of 33 and 27. Also check all elements in the sorted list. Here, we see that in this sub-list only one element 14 and 27 is greater than 14. Therefore the sub-list remains the same after the swap.

Now in the list of children we have two values ??14 and 27. Continue to compare 33 with 10.

These two values ??are not in order.

So we swap them.

Swapping leads to 27 and 10 in no order.

So we also exchange them.

We see again that 14 and 10 are not in order.

And we continue to swap these two numbers. Finally, after the third loop we have 4 elements.

The above process will continue until all unordered values ??are sorted into the sorted child list.

Next we explore the programming aspect of insertion sorting algorithm.

Insert algorithm (Insertion Sort)

From the above illustration, we have a general picture of the insertion arrangement algorithm, from which we will have the basic steps in the algorithm as follows:

```
B??c 1 : Ki?m tra n?u ph?n t? ??u tiên ?ã ???c s?p x?p. tr? v? 1 B??c 2
: L?y ph?n t? k? ti?p B??c 3 : So sánh v?i t?t c? ph?n t?
trong danh sách con ?ã qua s?p x?p B??c 4 : D?ch chuy?n t?t c? ph?n t?
trong danh sách con mà l?n h?n giá tr? ?? ???c s?p x?p B??c 5
: Chèn giá tr? ?ó B??c 6 : L?p l?i cho t?i khi danh sách ???c s?p x?p
```

Sample algorithm for arranging foam

```
B ? t  ?? u h à m insertionSort (  A  :  m ? ng ph ? n t ?
) int  holePosition  int  valueToInsert  for  i  = 1  t ?
i length ( A )  th ? c hi ? n : /* ch?n m?t giá tr? ??
```

```
 chèn */  valueToInsert  =  A [ i ]  holePosition  =  i  /*xác ??nh v?
 trí cho ph?n t? ???
c chèn */ while  holePosition  > 0  v à  A [ holePosition - 1 ] >  valueToInsert
? c hi ?
 n :  A [ holePosition ] =  A [ holePosition - 1 ]  holePosition  =  holePositio
? t th ú c  while /* chèn giá tr? t?i v?
 trí trên */  A [ holePosition ] =  valueToInsert k ? t th ú c  for  K ?
 t th ú c h à m
```

According to Tutorialspoint

Previous lesson: Bubble Sort (Bubble Sort)

Next lesson: Selection sort algorithm (Selection Sort)