Quick Sort (Quick Sort)

Quick Sort is a highly efficient algorithm and is based on dividing the array into smaller pieces.

What is Quick Sort?

Quick Sort is a highly efficient algorithm and is based on dividing the array into smaller pieces. The algorithm quickly divides the array into two parts by comparing each element of the array with an element called a Pivot element: an array of elements smaller than or equal to key and array elements The rest consists of larger or equal elements.

This split process continues until the length of the sub-arrays is equal to 1. The quick sorting algorithm appears to be quite effective for large data sets where the average case complexity and the case of especially O (nlogn) where n is the number of elements.

Techniques for selecting key elements in quick sort algorithms (Quick Sort)

The technique to select latch elements greatly affects the ability to fall into infinite loops for special cases. It is best to choose the pivot element located in the median of the list. Then, after log2 (n) the split time we will reach the size of the child arrays by 1.

Here are ways to select latch elements:

Select the top or bottom element as the latch element.

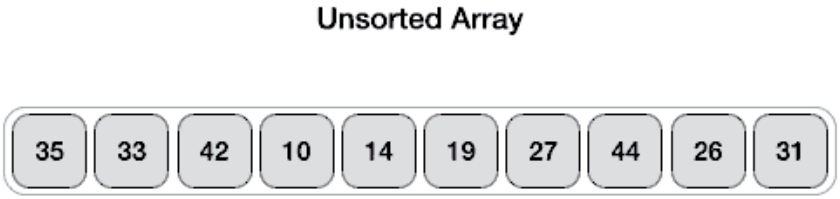
Select the middle element of the list as a latch element.

Select the middle element in the top three elements, centered and standing at the end as the latch element.

Select random elements as latch elements. However, this is very easy to lead to the possibility of falling into special circumstances.

Illustrate how to divide in a quick sort algorithm (Quick Sort)

The illustration below illustrates how to find the key element in the array. Here, we select the latch element at the bottom of the list.



The latch element divides the list into two parts. And using recursion, we find the key element for the child arrays until the list has only one element.

Quick-key element algorithm (Quick Sort)

Based on how to divide the list in the quick sort algorithm above, we can write an algorithm as below.

```
 B??c 1 : Ch?n ph?n t? ch?t là ph?n t? có ch? m?c cao nh?t (ph?n t? ? cu?
i danh sách) B??c 2 : Khai báo hai bi?n ?? tr? t?i bên trái và bên ph?i c?
a danh sách, ngo?i tr? ph?n t? ch?t B??c 3 : Bi?n bên trái tr? t?i m?
ng con bên trái B??c 4 : Bi?n bên ph?i tr? t?i m?ng con bên ph?i B??c 5
 : Khi giá tr? t?i bi?n bên trái là nh? h?n ph?n t? ch?t thì di chuy?
n sang ph?i B??c 6 : Khi giá tr? t?i bi?n bên ph?i là l?n h?n ph?n t? ch?
t thì di chuy?n sang trái B??c 7 : N?u không trong tr??ng h?p c? b??
c 5 và b??c 6 thì tráo ??i giá tr? bi?n trái và ph?i B??c 8 : N?u left ?
 right, thì ?ây chính là giá tr? ch?t m?i
```

Sample key algorithm in quick sort (Quick Sort)

From the steps above, we can deduce the sample code for the Quick Sort algorithm as follows:

```
 B ? t  ??
 u h à m partitionFunc ( left ,  right ,  pivot )  leftPointer  =  left  - 1  rig
? c hi ? n  while  A [++ leftPointer ]   pivot th ? c hi ? n  //không làm ?
i?u gì   k ?
 t th ú c  while while  rightPointer  > 0 &&  A [-- rightPointer ] >  pivot th
? c hi ? n  //không làm ?i?u gì   k ?
 t th ú c  while if  leftPointer  >=  rightPointer  break else Tr á o  ??
 i leftPointer , rightPointer k ? t th ú c  if  k ?
 t th ú c  while Tr á o  ?? i leftPointer , right  return  leftPointer K ?
 t th ú c h à m
```

Quick Sort (Quick Sort)

Using latch element algorithm recursively, we can end up with smaller sub-arrays. Then each of these sub-arrays can be processed with quick sort. Below, I use recursive algorithm for quick sort:

```
 B??c 1 : L?y ph?n t? ch?t là ph?n t? ? cu?i danh sách B??c 2 : Chia m?ng b
?i s? d?ng ph?n t? ch?t B??c 3 : S? d?ng s?p x?p nhanh m?t cách ?? qui v?
i m?ng con bên trái B??c 4 : S? d?ng s?p x?p nhanh m?t cách ?? qui v?i m?
ng con bên ph?i
```

Sample algorithm for Quick Sort (Quick Sort)

From the algorithm above, we can deduce the sample code for the algorithm to use recursion for quick sort as follows:

```
 B ? t  ??
 u h à m quickSort ( left ,  right ) if  right - left  = 0 return else  pivot  =
? t th ú c  if  K? t th ú c h à m
```

According to Tutorialspoint

Previous lesson: Shell Sort in data structure and algorithm

Next lesson: Graph data structure (Graph)