链滴

算法 - 排序之快速排序

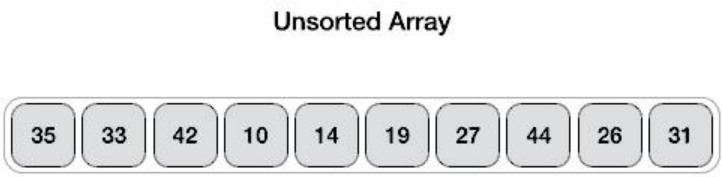
作者：amoslam
原文链接：https://ld246.com/article/1578199198233
来源网站：链滴

概述

快速排序采用了分治策略。就是在一个数组中取一个基准数字，把小的数放基准的左边，大的数放基的右边。 基准左边和右边分别是新的序列。在新的序列中再取一个基准数字，小的放左边，大的放边。

动图演示

图片来源：百度



优缺点

优点

极快，数据移动少

缺点

不稳定

复杂度

时间

1. 最好情况：O(nlog2(n))
2. 平均情况：O(nlog2(n))
3. 最坏情况：O(n^2)

空间

O(log2(n))

使用场景

适用于数组长度比较大的情况，对于小数组，快速排序比插入排序慢

代码

```
import java.util.Arrays;
```

```java
/**
 * ================================
 * 作者：amos lam
 * 时间：2020年1月5日下午12:37:50
 * 备注：算法 - 排序之快速排序
 * ================================
 */
public class QuickSort {

    public static void main(String[] args) {

        int[] arr = new int[] { 32, 26, 83, 55, 42, 67, 73, 92, 60 };
        quickSort(arr, 0, arr.length - 1);
        System.out.println(Arrays.toString(arr));
    }

    public static void quickSort(int[] arr, int low, int high) {

        if (low < high) {

            int temp = quickPartition(arr, low, high);
            quickSort(arr, low, temp - 1);
            quickSort(arr, temp + 1, high);
        }
    }

    public static int quickPartition(int[] arr, int low, int high) {

        int key = arr[low];
        while (low < high) {

            while (low < high && arr[high] >= key) {

                high--;
            }
            arr[low] = arr[high];
            while (low < high && arr[low] <= key) {

                low++;
            }
            arr[high] = arr[low];
        }
        arr[low] = key;
        return low;
    }
}
```