# 快速排序:

快速排序，顾名思义，是一种速度快，效率高的排序算法。

快排原理：

在要排的数（比如数组array）中选择一个中心值key（比如array[0]），通过一趟排序将数组array分成两部分，其中以key为中心，key右边都比key大，key左边的都key小，然后对这两部分分别重复这个过程，直到整个有序。

整个快排的过程就简化为了一趟排序的过程，然后递归调用就行了。

一趟排序的方法：

假设要排的数组为：array[8] ={ 5 2 8 9 2 3 4 9 }

选择 key = 5， 开始时 i=0，j=7

index 0 1 2 3 4 5 6 7

开始： 5 2 8 9 2 3 4 9

i j

第一次找 5 2 8 9 2 3 4 9

i j

交换： 5 2 4 9 2 3 8 9

i j

第二次找 5 2 4 9 2 3 8 9

i j

交换： 5 2 4 3 2 9 8 9

i j

第三次找 5 2 4 3 2 9 8 9

ij

调整key： 2 2 4 3 5 9 8 9

### 第1步：找基准值

所谓的基准值，顾名思义就是以它为基准进行比大小。通常来说，我们选取数组的第一个数为基准值。在数组array里基准值就是5.

### 第2步：比大小

先从数组的最右边开始往左边找比基准值小的第一个数，然后从数组的最左边开始往右找比基准值大的第一个数。那么为什么要这么找呢？因为现在我们要把数组从小到大排序，所以要找出比基准值小的数放到基准值的左边，找出比基准值的数放在基准值的右边。 那么在数组array里，从左往右找，第一个比5大的数就是8，我们把它的坐标记为i；从右往左找，第一个比5小的数就是4，我们把它的坐标记为j。

### 第3步：交换

找到之后，如果这个时候i<j，那么就交换这两个数，因为i=2,j=6，符合条件，将4和8进行交换。现在数组变成了int[]a={5,2 ,4 ,9 ,2 ,3 ,8 ,9}; 如果j>=i，那么不做处理 为什么还要判断i和j的大小呢？就像第二步说的，我们要找出比基准值小的数放到基准值的左边，找出比基准值的数放在基准值的右边。所以如果i<j的话，交换就达到目的了，如果i>=j，比基准值小的数就在基准值的左边，比基准值大的数已经在基准值的右边了，这时候就没必要交换了。

### 第4步：继续查找

在i<j的前提下，继续向右查找比基准值大的数，往左找比基准值小的数，然后交换。 在我们的例子中，9和3、符合交换的条件，所以数组就变成了 int[]a={5,2 ,4 ,3,2 ,9,8 ,9};这时候i=3,j=5

### 第5步：交换基准值到合适的位置

当查找继续进行，这时候i=j=4，已经不满足继续查找和交换的条件了，那么我们应该怎么办呢？答案就是把array[4]和基准值交换，因为i=j的时候说明已经到了一个分界线的位置，分界线左边的数比基准值小，分界线右边的数比基准值大，而这个分界线就是我们的基准值，所以把它和array[i]交换，这时候数组就变成： int[]a={2,2 ,4 ,3,5 ,9,8 ,9};

### 第6步：重复

对基准值左右两边的两个子数组重复前面五个步骤直至排序完成。

|  |
| --- |
| import java.util.Arrays;  public class QuickSort {   public static void main(String[] arrayrgs) {   //创建数组  int[] array = new int[10];  for(int i = 0;i<array.length;i++){  array[i] = (int)(Math.random()\*100);  }  System.out.println(Arrays.toString(array));   //调用排序  quickSort(array, 0 , array.length-1);   System.out.println(Arrays.toString(array));   }       public static void quickSort(int[] array, int start, int end) {   //1,找到递归算法的出口   if( start > end) {   return;   }   //2, 存储开始和末尾的位置  int i = start;   int j = end;   //3,存储基准值  int key = array[start];   //4，完成一趟排序   while( i< j) {   //4.1 ，从右往左找到第一个小于key的数   //要想降序只需要将> 修改成<  while(i<j && array[j] > key){   j--;   }   // 4.2 从左往右找到第一个大于key的数   //要想降序只需要将<= 修改成 >=  while( i<j && array[i] <= key) {   i++;   }   //4.3 交换   if(i<j) {   int tmp = array[i];   array[i] = array[j];   array[j] = tmp;   }   }   // 4.4，调整key的位置   int p = array[i];   array[i] = array[start];   array[start] = p;   System.out.println(i);  //5, 对key左边的数快排   quickSort(array, start, i-1 );   //6, 对key右边的数快排   quickSort(array, i+1, end);     }  } |