

# 快速排序

分类 **编程技术**

快速排序由于排序效率在同为O(N\*logN)的几种排序方法中效率较高，因此经常被采用，再加上快速排序思想----分治法也确实实用，因此很多软件公司的笔试面试，包括像腾讯，微软等知名IT公司都喜欢考这个，还有大大小小的程序方面的考试如软考，考研中也常常出现快速排序的身影。

总的说来，要直接默写出快速排序还是有一定难度的，因为本人就自己的理解对快速排序作了下白话解释，希望对大家理解有帮助，达到快速排序，快速搞定。

快速排序是C.R.A.Hoare于1962年提出的一种划分交换排序。它采用了一种分治的策略，通常称其为分治法(Divide-and-ConquerMethod)。

该方法的基本思想是：

1. 先从数列中取出一个数作为基准数。
2. 分区过程，将比这个数大的数全放到它的右边，小于或等于它的数全放到它的左边。
3. 再对左右区间重复第二步，直到各区间只有一个数。

虽然快速排序称为分治法，但分治法这三个字显然无法很好的概括快速排序的全部步骤。因此我的对快速排序作了进一步的说明：挖坑填数+分治法：

先来看实例吧，定义下面再给出（最好能用自己的话来总结定义，这样对实现代码会有帮助）。

以一个数组作为示例，取区间第一个数为基准数。

0	1	2	3	4	5	6	7	8	9
72	6	57	88	60	42	83	73	48	85

初始时，i = 0; j = 9; X = a[i] = 72

由于已经将 a[0] 中的数保存到 X 中，可以理解成在数组 a[0] 上挖了个坑，可以将其它数据填充到这儿来。

从j开始向前找一个比X小或等于X的数。当j=8，符合条件，将a[8]挖出再填到上一个坑a[0]中。a[0]=a[8]; i++; 这样一个坑a[0]就被搞定了，但又形成了一个新坑

## 教程列表

- ADO 教程
- Ajax 教程
- Android 教
- Angular2
- AngularJS
- AppML 教
- ASP 教程
- ASP.NET
- Bootstrap
- Bootstrap4
- Bootstrap5
- C 教程
- C# 教程
- C++ 教程
- CSS 参考
- CSS 教程
- CSS3 教程
- Django 教
- Docker 教
- DTD 教程
- ECharts 教
- Eclipse 教
- Firebug 教
- Font
- Foundation
- Git 教程
- Go 语言教
- Google 地
- Highcharts
- HTML
- HTML 参考
- HTML 字符
- HTML 教程
- HTTP 教程
- ionic 教程
- iOS 教程
- Java 教程
- JavaScript
- Javascript
- jQuery
- jQuery
- jQuery UI
- jQuery 教
- JSON 教程
- JSP 教程
- Julia 教程
- Kotlin 教程
- Linux 教程
- Lua 教程
- Markdown
- Matplotlib
- Maven 教
- Memcached
- MongoDB
- MySQL 教
- Node.js 教
- NumPy 教
- Pandas 教
- Perl 教程
- PHP 教程
- PostgreSQL
- Python 3
- Python 基
- R 教程
- RDF 教程
- React 教程
- Redis 教程
- RSS 教程
- Ruby 教程
- Rust 教程
- Sass 教程
- Scala 教程
- SciPy 教程
- Servlet 教
- SOAP 教程
- SQL 教程
- SQLite 教
- SVG 教
- SVN 教程
- Swift 教程
- TCP/IP
- TypeScript
- VBScript
- Vue.js 教
- Vue3 教程
- W3C 教程

a[8], 这怎么办了? 简单, 再找数字来填a[8]这个坑。这次从i开始向后找一个大于X的数, 当i=3, 符合条件, 将a[3]挖出再填到上一个坑中a[8]=a[3]; j--;

数组变为:

0	1	2	3	4	5	6	7	8	9
48	6	57	88	60	42	83	73	88	85

i = 3; j = 7; X=72

再重复上面的步骤, 先从后向前找, 再从前向后找。

从j开始向前找, 当j=5, 符合条件, 将a[5]挖出填到上一个坑中, a[3] = a[5]; i++;

从i开始向后找, 当i=5时, 由于i=j退出。

此时, i = j = 5, 而a[5]刚好又是上次挖的坑, 因此将X填入a[5]。

数组变为:

0	1	2	3	4	5	6	7	8	9
48	6	57	42	60	72	83	73	88	85

可以看出a[5]前面的数字都小于它, a[5]后面的数字都大于它。因此再对a[0...4]和a[6...9]这二个子区间重复上述步骤就可以了。

对挖坑填数进行总结:

1. i = L; j = R; 将基准数挖出形成第一个坑a[i]。
2. j--由后向前找比它小的数, 找到后挖出此数填前一个坑a[i]中。
3. i++由前向后找比它大的数, 找到后也挖出此数填到前一个坑a[j]中。
4. 再重复执行2, 3二步, 直到i=j, 将基准数填入a[i]中。

照着这个总结很容易实现挖坑填数的代码:

```
int AdjustArray(int s[], int l, int r) //返回调整后基准数的位置
{
    int i = l, j = r;
    int x = s[l]; //s[l]即s[i]就是第一个坑
    while (i < j)
    {
        // 从右向左找小于x的数来填s[i]
        while(i < j && s[j] >= x)
            j--;
        if(i < j)
        {
            s[i] = s[j]; //将s[j]填到s[i]中, s[j]就形成了一个
            //新的坑
            i++;
        }
    }
}
```

Web	WSDL 教	XLink 教程
XML DOM	XML	XML 教程
XPath 教程	XQuery 教	XSLFO 教
XSLT 教程	数据结构	正则表达式
测验	浏览器	网站品质
网站建设指	网站服务器	设计模式

```

    }

    // 从左向右找大于或等于x的数来填s[j]
    while(i < j && s[i] < x)
        i++;
    if(i < j)
    {
        s[j] = s[i]; //将s[i]填到s[j]中，s[i]就形成了一个
新的坑
        j--;
    }
}
//退出时，i等于j。将x填到这个坑中。
s[i] = x;

return i;
}

```

再写分治法的代码：

```

void quick_sort1(int s[], int l, int r)
{
    if (l < r)
    {
        int i = AdjustArray(s, l, r); //先成挖坑填数法调整s[]
        quick_sort1(s, l, i - 1); // 递归调用
        quick_sort1(s, i + 1, r);
    }
}

```

这样的代码显然不够简洁，对其组合整理下：

```

//快速排序
void quick_sort(int s[], int l, int r)
{
    if (l < r)
    {
        //Swap(s[l], s[(l + r) / 2]); //将中间的这个数和第一个
数交换 参见注1
        int i = l, j = r, x = s[l];
        while (i < j)
        {
            while(i < j && s[j] >= x) // 从右向左找第一个小于x
的数
                j--;
            if(i < j)
                s[i++] = s[j];

            while(i < j && s[i] < x) // 从左向右找第一个大于等
于x的数
                i++;
            if(i < j)
                s[j--] = s[i];
        }
        s[i] = x;
    }
}

```



```
        quick_sort(s, l, i - 1); // 递归调用
        quick_sort(s, i + 1, r);
    }
}
```

快速排序还有很多改进版本，如随机选择基准数，区间内数据较少时直接用另的方法排序以减小递归深度。有兴趣的筒子可以再深入的研究下。

注1，有的书上是以中间的数作为基准数的，要实现这个方便非常方便，直接将中间的数和第一个数进行交换就可以了。

作者: MoreWindows

原文: <https://blog.csdn.net/morewindows/article/details/6684558>

← 归并排序的实现

Mac OS SSH 使用 PEM 文件登录 →



2 篇笔记



写笔记



C#版本的，我来提供吧。

42

```
namespace{
    class Program{
        static void QuickSort(int[] dataArray,int left
,int right){
            if(left < right){
                int x=dataArray;
                int i=left;
                int j=right;
                while(true && i<j)
                {
                    while(true && i<j){
                        if(dataArray[j]<=x){
                            dataArray[i]=dataArray[j];
                            break;
                        }else{
                            j--;
                        }
                    }
                    while(true && i<j){
                        if(dataArray[i]>x){
                            dataArray[j]=dataArray[i];
                            break;
                        }//if结束
                    }else{
                        i++;
                    }
                }//while结束
            }//第一个while结束

            //跳出循环，现在i==j了，i是中间位。
```



```

        dataArray[i]=x;
        QuickSort(dataArray,left, i-1);
        QuickSort(dataArray,i+1,right);
    }
}
static void Main(string[] args){
    int[] data=new int[]{72,6,57,88,60,42,83,7
3,48,85};

    Console.WriteLine("快速排序的顺序为: ");
    QuickSort(data,0,data.Length-1);
    foreach(var item in data){
        Console.Write(item + " ");
    }
    Console.ReadLine();
}
}
}

```

阿拉蕾 11个月前 (06-07)



java 白话解释版，我来提供吧。

48

//公众号：一只快活的野指针

```

public class Quick {
    public static void main(String[] args) {
        int[] arr={8,4,5,7,1,3,6}; //直接复制数组
        quick_sort(arr,0,arr.length-1);
        print(arr);
    }

    private static int get_mid(int arr[],int left,int
right){
        int pivot=arr[left]; //自定义排序中心轴，这里把arr
r[left]存到pivot中去，此时arr[left]为空。pivot相当于一个
中间量

        while(left<right){ //当left与right指针相遇的时候
退出循环，双指针遍历结束

            while(arr[right]>=pivot && left<right) rig
ht--; //right指针从右往左遍历，当arr[right]>=pivot，即满足
以pivot为中轴，小放左，大放右的条件时，right指针继续往右遍
历。当arr[right]<pivot的时候，把当前值arr[right]赋给空置
arr[left]，此时arr[right]成了空值。

            arr[left]=arr[right];

            while(arr[left]<=pivot && left<right) left
++; //到left指针从左往右遍历，当arr[left]<=pivot，即满足以
pivot为中轴，小放左，大放右的条件时，left指针继续往左遍历。
当arr[left]>pivot的时候，把当前值arr[left]赋给空置arr[ri
ght]，此时arr[left]成了空值。

            arr[right]=arr[left];
        }

        //经历了上面的循环实现了pivot为中轴，小放左，大放
右的格局

        arr[left]=pivot; //最后把存放在pivot值放回数组空a
rr[left]中

        return left; //返回中轴所在的下标位置。
    }
}

```



```
private static void quick_sort(int[] arr,int left
,int right){
    if(left<right){
        /*将arr[left..right]均分为两部分arr[left..mid]和a
rr[mid+1..right]
        *,以pivot为中轴，小放左，大放右。这是第一步。*/
        int mid =get_mid(arr,left,right);//接收中轴
所在的下标位置。
        quick_sort(arr,left,mid-1);//递归地对arr[le
ft..mid]进行快速排序，使得左子序列有序
        quick_sort(arr,mid+1,right);//递归地对arr[m
id+1..right]进行快速排序，使得左子序列有序
    }
}

public static void print(int arr[])//封装函打印函数
{
    for(int k=0;k<arr.length;k++)
    {
        System.out.print(arr[k]+" ");
    }
}
}
```

一只可爱的野指针 11个月前 (06-20)

<div>在线实例</div> <ul style="list-style-type: none"><li>· HTML 实例</li><li>· CSS 实例</li><li>· JavaScript 实例</li><li>· Ajax 实例</li><li>· jQuery 实例</li><li>· XML 实例</li><li>· Java 实例</li></ul>	<div>字符集&amp;工具</div> <ul style="list-style-type: none"><li>· HTML 字符集设置</li><li>· HTML ASCII 字符集</li><li>· HTML ISO-8859-1</li><li>· PNG/JPEG 图片压缩</li><li>· HTML 拾色器</li><li>· JSON 格式化工具</li><li>· 随机数生成器</li></ul>	<div>最新更新</div> <ul style="list-style-type: none"><li>· MIME 类型</li><li>· Julia 字典和集合</li><li>· Julia 流程控制</li><li>· Julia 函数</li><li>· Python 简单的银...</li><li>· Chrome 浏览器无...</li><li>· Vue3 组合式 API</li></ul>	<div>站点信息</div> <ul style="list-style-type: none"><li>· 意见反馈</li><li>· 免责声明</li><li>· 关于我们</li><li>· 文章归档</li></ul>
<div>关注微信</div> <div></div>			<div> ★</div>