

## 什么是快速排序?

快速排序 (Quick Sort) 是一个十分经典的排序算法, 但不同于希尔排序、选择排序的思想, 快速排序的思想其实来源于冒泡排序, 他们都属于交换类的排序思想, 不断通过比较和移动来实现排序。

## ★ 回顾冒泡排序

那么, 作为引入, 我们先回顾一下冒泡排序的思想: 基本思想就是交换, 两两比较相邻的元素, 如果反序则进行交换, 知道整个数组 / 序列没有反序为止。

假设我们有一个数组, 现在要进行升序排序:  
(下面是冒泡排序的初级版本)

```
#include <stdio.h>
void BubbleSort0(int arr[], int sz) {
    for (int i = 0; i < sz; i++) {
        for (int j = 0; j < sz - i - 1; j++) {
            if (arr[j] > arr[j + 1]) {
                int tmp = arr[j];
                arr[j] = arr[j + 1];
                arr[j + 1] = tmp;
            }
        }
    }
}

int main() {
    int arr[10] = { 5, 3, 6, 9, 1, 7, 2, 10, 8, 4 };
    //我们相对arr数组的元素进行排序(升序)
    int sz = sizeof(arr) / sizeof(arr[0]);
    BubbleSort0(arr, sz);
    for (int i = 0; i < sz; i++) {
        printf("%d ", arr[i]);
    }
    return 0;
}
```

```
int main() {
    int arr[10] = { 5, 3, 6, 9, 1, 7, 2, 10, 8, 4 }; 1 2 3 4 5 6 7 8 9 10
```

我们可以分析, 双重for循环结束条件如何控制, 在这儿有排序趟数, 和每一趟的比较次数, 考虑 3 2 1, 这是一个数组, 那我们想排为 1 2 3, 显然先交换 3 和 2 变为 2 3 1, 再 3 和 1 交换, 变为 2 1 3, 一次 (一趟) 排序我们找到一个最大的元素放在数组末尾, 到第二次的话第一次最大的元素就不需要进行比较, 则第二趟比较 2 和 1, 并交换即可。那么显然排序三个数字需要两趟可以排好, 且考虑 N 个数字是要进行比较 N - 1 次比较, 而 N 个数字共需要 N - 1 趟来进行排序。

但是不难发现，当数组完全逆序时，是最贴合上述次次比较且交换的情况的，这也是冒泡排序时间复杂度最坏的情况，但是把那个不是所有的序列是完全逆序的，考虑这样一种情况：

**当序列已经是完全有序的情况呢？**如：1 2 3 4 5；或者在第一趟排完后序列就有完全序了，如：5 1 2 3 4，一趟后变为：1 2 3 4 5，这意味着后续的训话在做无用功，我们必然要考虑改进：

```
//改进后的冒泡排序
void BubbleSort1(int arr[], int sz) {
    int flag = 1;
    for (int i = 0; i < sz && flag; i++) {
        for (int j = 0; j < sz - i - 1; j++) {
            flag = 0;
            if (arr[j] > arr[j + 1]) {
                int tmp = arr[j];
                arr[j] = arr[j + 1];
                arr[j + 1] = tmp;
                flag = 1; //有数据交换，则代表不有序
            }
        }
    }
}
```

通过flag标记就可以控制当这一趟比较未发生交换时，那么整个数组就是有序的，&& flag 就不满足，跳出循环，排序完成。

考虑冒泡排序的时间复杂度，关于最差性能，即所有趟分别比较 (n-1)、(n-2)、……3、2、1次，等差数列求和为  $n(n-1)/2$  次，那么时间复杂度显然是  $O(n^2)$ 。当然，最好情况就是完全有序，只进行比较 n - 1次，那时间复杂度为  $O(n)$ 。

## 解析C语言qsort快速排序函数

qsort函数是C语言标准库提供的排序函数，对数据进行排序，且qsort可以实现排序任意类型的数据。

在官方的函数细节文档[qsort](#)我们可以看到qsort函数的描述。

```
void qsort (void* base, size_t num, size_t size,
            int (*compar)(const void*,const void*));
```

- 【base】指向待排序数据的起始位置
- 【num】base数据里的元素个数
- 【size】数据里每一个元素的大小（字节）
- 【compar】`int (*compar)(const void*,const void*)` 这很显然是一个函数指针，这是一个比较函数。compar指针指向比较两个元素的函数。

当p1指向的元素大于p2，那么返回大于0的数字；

当p1指向的元素小于p2，那么返回小于0的数字；

如果相等，返回0；

base设计为void\* 类型代表我并不知道待排序数据的类型，void\* 指针很宽容，可以接收不同类型的

指针，其实给了我们很多自由度，也便于实现排序任意类型的数据。

那么我们还是使用上面的例子：{ 5,3,6,9,1,7,2,10,8,4 }，来使用qsort函数排序整形数组。

```
qsort(_Base: arr, _NumOfElements: sz, _SizeOfElements: sizeof(arr[0]), _CompareFunction: Cmpint);
```

```
void print(int arr[], int sz) {
    for (int i = 0; i < sz; i++) {
        printf("%d ", arr[i]);
    }
}

int Cmpint(const void* e1, const void* e2) {
    return *(int*)e1 - *(int*)e2;
}

void Test_qsort() {
    int arr[10] = { 5, 3, 6, 9, 1, 7, 2, 10, 8, 4 };
    //我们相对arr数组的元素进行排序(升序)
    int sz = sizeof(arr) / sizeof(arr[0]);
    // 在本例中我们实现比较两个整形
    qsort(arr, sz, sizeof(arr[0]), Cmpint);

    print(arr, sz);
}

int main() {
    Test_qsort();
    return 0;
}
```

上面实现了升序，那么根据compar的特点，我们交换e1和e2，就可以实现降序排序：

```
int Cmpint(const void* e1, const void* e2) {
    //return *(int*)e1 - *(int*)e2;
    return *(int*)e2 - *(int*)e1;
}
```

```
void print(int arr[], int sz) {
    for (int i = 0; i < sz; i++) {
        printf("%d ", arr[i]);
    }
}

int Cmpint(const void* e1, const void* e2) {
    //return *(int*)e1 - *(int*)e2;
    return *(int*)e2 - *(int*)e1;
}

void Test_qsort() {
    int arr[10] = { 5, 3, 6, 9, 1, 7, 2, 10, 8, 4 };
    //我们相对arr数组的元素进行排序(升序)
    int sz = sizeof(arr) / sizeof(arr[0]);
    // 在本例中我们实现比较两个整形
    qsort(arr, sz, sizeof(arr[0]), Cmpint);

    print(arr, sz);
}

int main() {
    Test_qsort();
    return 0;
}
```

Microsoft Visual Studio  
10 9 8 7 6 5 4 3 2 1  
E:\Github\_Gitee\C\So  
按任意键关闭此窗口.

CSDN @期邈云汉

那么其实如果要排序结构体类型，原理也是相同的，只不过多了一些结构体的使用方法：

```
struct Stu {
    char name[10];
    int age;
};

CmpStruct(const void* e1, const void* e2) {
    //假设按姓名排序
    //return strcmp(((struct Stu*)e1)->name, ((struct Stu*)e2)->name);
    //假设按年龄排序
    return ((struct Stu*)e1)->age - ((struct Stu*)e2)->age;
}

void test_qsort2() {
    struct Stu s[] = { {"wang", 20}, {"liu", 18} };
    int sz = sizeof(s) / sizeof(s[0]);
    qsort(s, sz, sizeof(s[0]), CmpStruct);

    printf("%s %d\n", s[0].name, s[0].age);
    printf("%s %d\n", s[1].name, s[1].age);
}

int main() {
    //Test_qsort();
    test_qsort2();
    return 0;
}
```

```
liu 18
wang 20
```

CSDN @期邈云汉

这样也可以使用qsort函数实现结构体排序。


考虑将qsort函数特点，我们自己来实现一个可以排序任意类型的BubbleSort冒泡排序算法。

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
void print(int arr[], int sz) {
    for (int i = 0; i < sz; i++) {
        printf("%d ", arr[i]);
    }
}
int Cmpint(const void* e1, const void* e2) {
    return *(int*)e1 - *(int*)e2; //升序
    //return *(int*)e2 - *(int*)e1; //降序
}
struct Stu {
    char name[10];
    int age;
};
CmpStruct(const void* e1, const void* e2) {
    //假设按姓名排序
    //return strcmp(((struct Stu*)e1)->name, ((struct Stu*)e2)->name);
    //假设按年龄排序
    return ((struct Stu*)e1)->age - ((struct Stu*)e2)->age;
}
void Swap(char* buf1, char* buf2, int size) {
    for (int i = 0; i < size; i++) {
        char tmp = *buf1;
        *buf1 = *buf2;
        *buf2 = tmp;
        buf1++;
        buf2++;
    }
}
void Bubble_sort(void* base, size_t sz, size_t size, int (*cmp)(const void* e1, const void* e2)) {
    for (int i = 0; i < sz; i++) {
        for (int j = 0; j < sz - i - 1; j++) {
            if (cmp((char*)base + j * size, (char*)base + (j + 1) * size) > 0) {
                //返回大于零,那就要交换
                Swap((char*)base + j * size, (char*)base + (j + 1) * size, size);
            }
        }
    }
}
void test3() {
    int arr[10] = { 5, 3, 6, 9, 1, 7, 2, 10, 8, 4 };
}
```

```

//我们相对arr数组的元素进行排序(升序)
int sz = sizeof(arr) / sizeof(arr[0]);
Bubble_sort(arr, sz, 4, Cmpint);
print(arr, sz);
printf("\n");
//测试排序结构体
struct Stu s[] = { {"wang",20}, {"liu",18} };
sz = sizeof(s) / sizeof(s[0]);
Bubble_sort(s, sz, sizeof(s[0]), CmpStruct);
printf("%s %d\n", s[0].name, s[0].age);
printf("%s %d\n", s[1].name, s[1].age);
}
int main() {
    //Test_qsort();
    //test_qsort2();
    test3();
    return 0;
}

```

 Microsoft Visual Studio 调试控制台

```

1 2 3 4 5 6 7 8 9 10
liu 18
wang 20

```

CSDN @期邈云汉