

# [C/C++ 基础知识] 一篇就让你彻底搞懂qsort快速排序的文章

原创 Eastmount 2015-10-11 05:45:39 9251 收藏 2

版权

分类专栏: C/C++ 基础知识 文章标签: C语言 快速排序 sort 结构快排 分治法



## Python+TensorFlow人工智能

该专栏为人工智能入门专栏，采用Python3和TensorFlow实现人工智能相关算法。前期介绍安装流程、基础语法、神经网络、可视化等，中间讲解CNN、RNN、LSTM等代码，后续复现图像处理...



Eastmount

¥9.90

订阅博主

最近在做LeetCode的题目、面试和笔试后发现经常考察快速排序的知识。通过这篇文章介绍，能让你彻底的了解和学习快排，主要从一下三个部分进行介绍：

一.C语言实现qsort快速排序

二.快速排序的原理及手写快排源码

三.LeetCode关于Two Sum的快排实现

参考文献：

《算法分析与设计》关于分治法那章内容

如何利用C语言中的qsort库函数实现快速排序 - by:stpeace

白话经典算法系列之六 快速排序 快速搞定 - by:MoreWindows

<https://leetcode.com/problems/two-sum/>

[leetcode] Two Sum by:阳光岛主

## 一. C语言实现qsort快速排序

这段介绍参考百度百科，编译器函数库自带的快速排序函数qsort。使用qsort()排序并用 bsearch()搜索是一个比较常用的组合，使用方便快捷。

头文件：stdlib.h

用法：void qsort(void \*base, int nelem, int width, int (\*fcmp)(const void \*, const void \*));

参数：1 待排序数组首地址

2 数组中待排序元素数量

3 各元素的占用空间大小

4 指向函数的指针，用于确定排序的顺序

### 整数快排

```
int cmp1(const void *a, const void *b)
```

```
{
```

```
    return *(int*)a - *(int*)b;
```

```
}
```

```
qsort(num, len, sizeof(int), cmp1);
```

```
#include <stdio.h>
```

```
#include <stdlib.h>
```

```
//整数从小到大排序
```

```
int cmp1(const void *a, const void *b)
```

```
{
```

```
    return *(int*)a - *(int*)b;
```

```
}
```

```
//整数从大到小排序
```

```
int cmp2(const void *a, const void *b)
```

```
{
```

```

        return *(int*)b - *(int*)a;
    }

int main()
{
    int num[8] = {4,11,2,-3,15,4,0,7};
    int len = 8;
    int i;

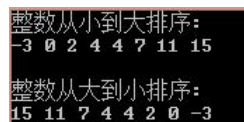
    printf("整数从小到大排序:\n");
    qsort(num, len, sizeof(int), cmp1);
    for(i = 0; i < len; i++)
    {
        printf("%d ", num[i]);
    }
    printf("\n\n");

    printf("整数从大到小排序:\n");
    qsort(num, len, sizeof(int), cmp2);
    for(i = 0; i < len; i++)
    {
        printf("%d ", num[i]);
    }
    printf("\n");

    system("PAUSE");
    return 0;
}

```

运行结果如下图所示:



```

整数从小到大排序:
-3 0 2 4 4 7 11 15
整数从大到小排序:
15 11 7 4 4 2 0 -3

```

## 二维数组快排

```

int cmp(const void *a, const void *b)
{
    return ((int*)a)[0] - ((int*)b)[0];
}

qsort(num, len, sizeof(int)*2, cmp);

#include <stdio.h>
#include <stdlib.h>

//二维数组按照num[i][0]从小到大排序
int cmp(const void *a, const void *b)
{
    return ((int*)a)[0] - ((int*)b)[0];
}

int main()
{
    int num[6][2] = {
        4, 1,
        11, 2,
        2, 3,
        -3, 4,
        15, 5,
        0, 7,
    }
}

```

```

    };

    int len = 6;

    int i;

    printf("排序前:\n");
    for(i = 0; i < len; i++)
    {
        printf("%4d %4d\n", num[i][0], num[i][1]);
    }

    printf("\n");

    qsort(num, len, sizeof(int)*2, cmp);

    printf("排序后:\n");
    for(i = 0; i < len; i++)
    {
        printf("%4d %4d\n", num[i][0], num[i][1]);
    }
    printf("\n");

    system("PAUSE");
    return 0;
}

```

运行结果如下图所示，其中num[i][0]和num[i][1]同时移动。

```

排序前:
 4      1
11      2
 2      3
-3      4
15      5
 0      ?

排序后:
-3      4
 0      7
 2      3
 4      1
11      2
15      5

```

By:Eastmount CSDN

### 字符串快排

```

int cmp(const void *a, const void *b)
{
    return *(char*)a - *(char*)b;
}

qsort(str, sum, sizeof(char)*10, cmp);

```

```

#include <stdio.h>
#include <stdlib.h>

int cmp(const void *a, const void *b)
{
    return *(char*)a - *(char*)b;
}

int main()
{
    char str[7][10] = {
        "Monday",
        "Tuesday",
        "Wednesday",

```

```

        "Thursday",          "Friday",
        "Saturday",
        "Sunday"
    };
    int sum = 7;
    int i;

    printf("排序前:\n");
    for(i = 0; i < sum; i++)
    {
        printf("%s\n", str[i]);
    }
    printf("\n");

    qsort(str, sum, sizeof(char)*10, cmp);

    printf("排序后:\n");
    for(i = 0; i < sum; i++)
    {
        printf("%s\n", str[i]);
    }
    printf("\n");

    system("PAUSE");
    return 0;
}

```

同int类型一样，输出如下所示：



```

排序前:
Monday
Tuesday
Wednesday
Thursday
Friday
Saturday
Sunday
排序后:
Friday
Monday
Sunday
Saturday
Thursday
Tuesday
Wednesday
By:Eastmount CSDN

```

### double快排

```

int cmp(const void *a, const void *b)
{
    return *(double*)a > *(double*)b ? 1 : -1;
}

qsort(num, sum, sizeof(double), cmp);

#include <stdio.h>
#include <stdlib.h>

int cmp(const void *a, const void *b)
{
    return *(double*)a > *(double*)b ? 1 : -1;
}

int main()
{

```

```

double num[7] = {
    12.2324,
    -4.3457,
    0.00000,
    5.64375,
    1.25879,
    0.00001,
    7.85435
};
int sum = 7;
int i;

printf("排序前:\n");
for(i = 0; i < sum; i++)
{
    printf("%10f\n", num[i]);
}
printf("\n");

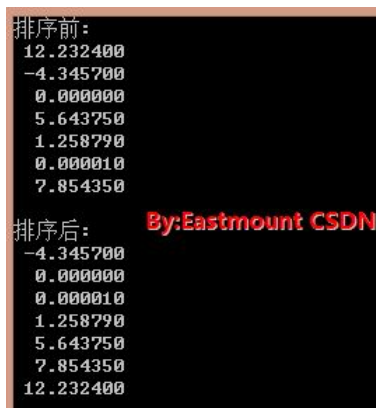
qsort(num, sum, sizeof(double), cmp);

printf("排序后:\n");
for(i = 0; i < sum; i++)
{
    printf("%10f\n", num[i]);
}
printf("\n");

system("PAUSE");
return 0;
}

```

输出如下图所示:



```

排序前:
12.232400
-4.345700
0.000000
5.643750
1.258790
0.000010
7.854350

排序后:
-4.345700
0.000000
0.000010
1.258790
5.643750
7.854350
12.232400

```

### 结构体排序

```

int compare1(const void *a,const void *b)
{
    return ((Student*)a)->score - ((Student*)b)->score;
}

qsort(s, N, sizeof(Student), compare1);

```

```

#include <stdio.h>
#include <stdlib.h>
#define N 6

typedef struct

```

```

{
    char name[15];
    int score;

}Student;

int compare1(const void *a,const void *b)
{
    return ((Student*)a)->score - ((Student*)b)->score;
}

int compare2(const void *a,const void *b)
{
    return *(((Student*)a)->name) - *(((Student*)b)->name);
}

int main()
{
    Student s[N] =
    {
        "Zhang San", 94,
        "Li Si",      80,
        "You",        94,
        "I",          100,
        "He",         72,
        "She",        60
    };

    int i;
    printf("按照分数排序:\n");
    qsort(s, N, sizeof(Student), compare1);
    for(i = 0; i < N; i++)
    {
        printf("%-15s : %d\n", s[i].name, s[i].score);
    }
    printf("\n");

    qsort(s, N, sizeof(Student), compare2);
    printf("按照姓名排序:\n");
    for(i = 0; i < N; i++)
    {
        printf("%-15s : %d\n", s[i].name, s[i].score);
    }
    system("PAUSE");
    return 0;
}

```

代码输出如下图所示:

```

按照分数排序:
She      : 60
He       : 72
Li Si    : 80
You      : 94
Zhang San : 94
I        : 100

按照姓名排序:
He       : 72
I        : 100
Li Si    : 80
She      : 60
You      : 94
Zhang San : 94

```

## 二. 快速排序原理及手写快排源码

由于MoreWindows写的白话文系列太好，所以这部分主要参考他的文章和《算法设计与分析》关于分治法那章内容。强烈推荐大家阅读系列文章，原文链接：

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

分治法的思想将一个大问题分割成小规模问题各个击破，分而治之，常用递归解决。常用的包括二分查找、归并排序和快速排序等应用。其中思想如下所示：

```
divide-and-conquer(P)
{
    if(|P|<=n0) adhoc(p);          //解决小规模问题
    divide P into smaller subinstances P1,P2...Pk;
    for(i=1; i<=k; i++) {
        yi = divide-and-conquer(Pi); //分解
    }
    return merge(y1,y2...yk);      //合并子算法
}
```

关于快速排序，由于它的时间复杂度在 $O(N \cdot \log N)$ 效率较高，经常使用。该算法的基本思想：

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

MoreWindows对快速排序作了进一步的说明：**挖坑填数+分治法**

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	5	7	8	8	6	0	4	2	8	3	7	3	4	8	8	5

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 $a[8]$ ，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	5	7	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	5	7	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：

**分治法：**

```

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);
    }
}

```

### 挖坑填数：

```

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++;
        }

        // 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;
}

```

PS：去一些大公司面试，经常会让你手写快排、二叉树非递归遍历、链表翻转等

## 三. LeetCode关于Two Sum的快排实现

下面讲一个经典的题目。做过LeetCode的肯定做个这个题目，LeetCode的第一题Two Sum，求两个数的和。  
链接：<https://leetcode.com/problems/two-sum/>

Given an array of integers, find two numbers such that they add up to a specific target number.

The function twoSum should return indices of the two numbers such that they add up to the target, where index1 must be less than index2. Please note that your returned answers (both index1 and index2) are not zero-based.

You may assume that each input would have exactly one solution.

Input: numbers={2, 7, 11, 15}, target=9



Output: index1=1, index2=2

找到两个数字和为target，并输出两个值的下标。由于数组可能是无序状态，故需要排序后输出即可。但你需要注意以下几点：

1.首先想到的方法是两层循环遍历普通排序，时间复杂度 $O(N^2)$ 会TLE

2.输出下标不是从0开始，故i+1即可

3.输出时负数需要判断小的在前

易错：Input: [-1,-2,-3,-4,-5] -8 Output: [5,3] Expected: [3,5]

4.错误 Input: [3,2,4] 6 Output: [1,3] Expected: [2,3] 输出原始位置，所以使用快速排序怎样解决呢？方案一采用结构，方案二采用二维数组调用qsort：

(1)先定义结构NODE存储<值,下标>在进行qsort排序

(2)再一层循环前后两个指针移动遍历求和,时间复杂度 $O(N \log N)$

由于结果唯一,可以通过左右下标移动实现,如果sum < target 移动左下标

5.通过Hash实现时间复杂度 $O(n)$

代码如下：

```
// 定义结构 因为需要数组值和下标同时排序
typedef struct NODE {
    int value;
    int pos;
}node;

// 调用cmp排序实现结构排序
int cmp(const void *a, const void *b)
{
    return ((node *)a)->value - ((node *)b)->value;
}

int* twoSum(int* nums, int numsSize, int target) {
    int *result;
    int i,j;
    int sum;
    node *numNode;
    result=(int*)malloc(sizeof(int)*2);
    numNode=(node*)malloc(sizeof(node)*numsSize);

    for(i=0; i<numsSize; i++) {
        numNode[i].value=nums[i];
        numNode[i].pos=i;
    }

    qsort(numNode,numsSize,sizeof(node),cmp);

    for(i=0,j=numsSize-1; i<j; ) {
        sum=numNode[i].value+numNode[j].value;
        if(sum==target) {
            if(numNode[i].pos<numNode[j].pos) { // 小在前
                result[0]=numNode[i].pos+1;
                result[1]=numNode[j].pos+1;
                break;
            }
            else {
                result[0]=numNode[j].pos+1;
                result[1]=numNode[i].pos+1;
                break;
            }
        }
    }
}
```

```

        else if(sum<target) { //左移                i++;
        }
        else { //右移
            j--;
        }
    }
    return result;
}

```

当然你也可以调用unordered\_map实现哈希表O(n)算法：

```

class Solution {
public:
    vector<int> twoSum(vector<int>& nums, int target) {
        vector<int> vec;
        unordered_map<int,int> map;
        for(int i=0; i<nums.size(); i++){
            if(map.find(target-nums[i]) != map.end()){
                vec.push_back(map[target-nums[i]]+1);
                vec.push_back(i+1);
                return vec;
            }
            map[nums[i]]=i;
        }
    }
};

```

希望文章对你有所帮助吧！如果有错误或不足之处，还请海涵。最后分析一个今天的故事：

今天去一个小公司面试（百度百科没有该公司词条），那个Leader感觉很不尊重人，一方面总是不屑的摇头与嘲笑人，一方面又总和旁边的人聊天，还不断打断人说话。确实我普通话带有西南口音，但是我就不明白了，难道这能影响到我写程序吗？虽然他直言不讳的说我编程语言不说精通，连熟悉都达不到，但是我前面都说了我不精通任何东西，只是一般。让我给他讲知识图谱，他说要用通俗的话，那我就举例子乔布斯、苹果等，他又说没难度，那我就讲神经网络，他说他不会编程。我不知道怎么说了~他居然还说百度可能仅仅是通过传统的搜索引擎现在返回一个值，如姚明的身高，而没有真正去做这个知识图谱，我不知道怎么吐槽。

仅仅面试了10分钟，我就想走了，后面的问题我都说不知道，问我Python的优势，我说**脚本语言、开发快、语言简洁，面向对象**。他非说是面向过程的，和我争论怎么是面向对象的，我说举个例子建房子，他又说那是老师教的，我让你讲实例。好吧！我承认我比较弱，但是Python是面向对象这种常识都不知道，我只能呵呵了~

然后他说你这也不会，那也不会，那就简单讲讲你认为自己最优秀的地方。我就说我最大的优点就是坚持，不论10年，5年也好，我能做出东西来；同时以后还是想会贵州当老师。他最后说我们要的人还是有要求的，明确告诉你不会给你offer，我当时也回了句：我也不会来的。

最后离开的时候我给HR说我就是很不爽他，就没心思答了，而且我表达能力也不好，但还是非常感谢你的推荐。最后当得知他是国内最好的大学毕业的时候，我也非常震惊。我脾气只有这么好了，二十几年的光阴里没怎么生过气，当时确实生气。只恨脸皮薄，当时没损他几句，但那又显得不够大度。

其实我想说：面试本身就是一个相互学习的过程，确实很多基础知识我都不记得了，可能也不善言表，但是你是一个面试官不尊重面试人员，那我就是看不起你。不论你赚再多的钱，再有本事，学校再厉害，我就是看不起，**做人似乎比工作更重要吧！**我也大大小小参加了不少的面试，阿里、360、百度都有，各种大牛也遇到过，马云、吴恩达、刘知远的视频和讲座也看过不少，清华北大也都有同学，但是没见过这么不尊重人的，面试过程中又嘲讽又聊天的，各种秀英语，还吐槽我普通话。

先做人，后做事吧！看着比我年轻几岁，这种秀优越感只能理解为不成熟。

如果他们公司都是这样招人的，那早晚也会毁在他们手上的，虽然他们公司现在有很大的潜力；如果我还在他说下干活，遇到个东西整天吐槽你做得不够好，给他代码他又说不会，我也早晚会辞职的。总之，以后如果我走到当面试官那一步了，**最起码要尊重面试人员，因为你是再寻财，而不是秀优越感！**而且我都不知道哪里来的优越感，学

校吗？好吧！但我还是看不起你这个人~

当然面试也让我知道了自己还存在很多不足，尤其是底层的算法很多都忘了，最近扎扎实实看书和做LeetCode吧！自己技术还是不够精通，非常致命，包括同步异步通讯，线程进程代码等~但还是感谢那个热心的HR吧！以后也尽量别参加这种没有笔试，通过猎头直接去小公司面试了。

(By: Eastmount 2015-10-11 清晨6点 <http://blog.csdn.net/eastmount/>)