

# 数据结构常见的八大排序算法（详细整理）



LeeLom (/u/3e74cab31591) [+ 关注](#)

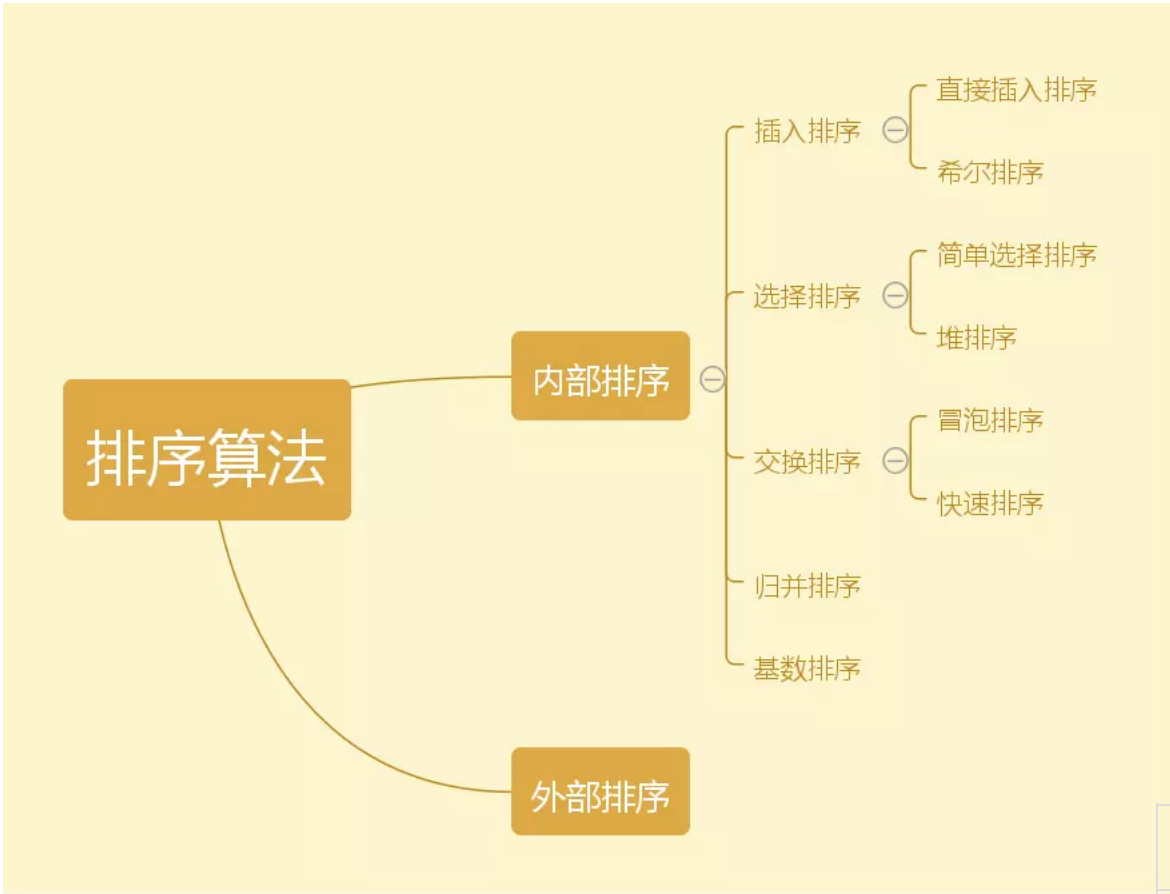
2016.08.13 13:22\* 字数 2175 阅读 54329 评论 39 喜欢 568 赞赏 2

(/u/3e74cab31591)

## 前言

八大排序，三大查找是《数据结构》当中非常基础的知识点，在这里为了复习顺带总结了一下常见的八种排序算法。

常见的八大排序算法，他们之间关系如下：



排序算法.png

他们的性能比较：

各种常用排序算法						
类别	排序方法	时间复杂度			空间复杂度	稳定性
		平均情况	最好情况	最坏情况	辅助存储	
插入排序	直接插入	$O(n^2)$	$O(n)$	$O(n^2)$	$O(1)$	稳定
	shell排序	$O(n^{1.3})$	$O(n)$	$O(n^2)$	$O(1)$	不稳定
选择排序	直接选择	$O(n^2)$	$O(n^2)$	$O(n^2)$	$O(1)$	不稳定
	堆排序	$O(n\log_2n)$	$O(n\log_2n)$	$O(n\log_2n)$	$O(1)$	不稳定
交换排序	冒泡排序	$O(n^2)$	$O(n)$	$O(n^2)$	$O(1)$	稳定
	快速排序	$O(n\log_2n)$	$O(n\log_2n)$	$O(n^2)$	$O(n\log_2n)$	不稳定
归并排序		$O(n\log_2n)$	$O(n\log_2n)$	$O(n\log_2n)$	$O(1)$	稳定
基数排序		$O(d(r+n))$	$O(d(n+rd))$	$O(d(r+n))$	$O(rd+n)$	稳定

性能比较.png

下面，利用Python分别将他们进行实现。

直接插入排序

- 算法思想：



直接插入排序.gif

直接插入排序的核心思想就是：将数组中的所有元素依次跟前面已经排好的元素相比较，如果选择的元素比已排序的元素小，则交换，直到全部元素都比较过。因此，从上面的描述中我们可以发现，直接插入排序可以用两个循环完成：

1. 第一层循环：遍历待比较的所有数组元素
2. 第二层循环：将本轮选择的元素(selected)与已经排好序的元素(ordered)相比较。

如果：selected > ordered，那么将二者交换

• 代码实现

```
#直接插入排序
def insert_sort(L):
    #遍历数组中的所有元素，其中0号索引元素默认已排序，因此从1开始
    for x in range(1,len(L)):
        #将该元素与已排序好的前序数组依次比较，如果该元素小，则交换
        #range(x-1,-1,-1):从x-1倒序循环到0
        for i in range(x-1,-1,-1):
            #判断：如果符合条件则交换
            if L[i] > L[i+1]:
                temp = L[i+1]
                L[i+1] = L[i]
                L[i] = temp
```

希尔排序

• 算法思想：

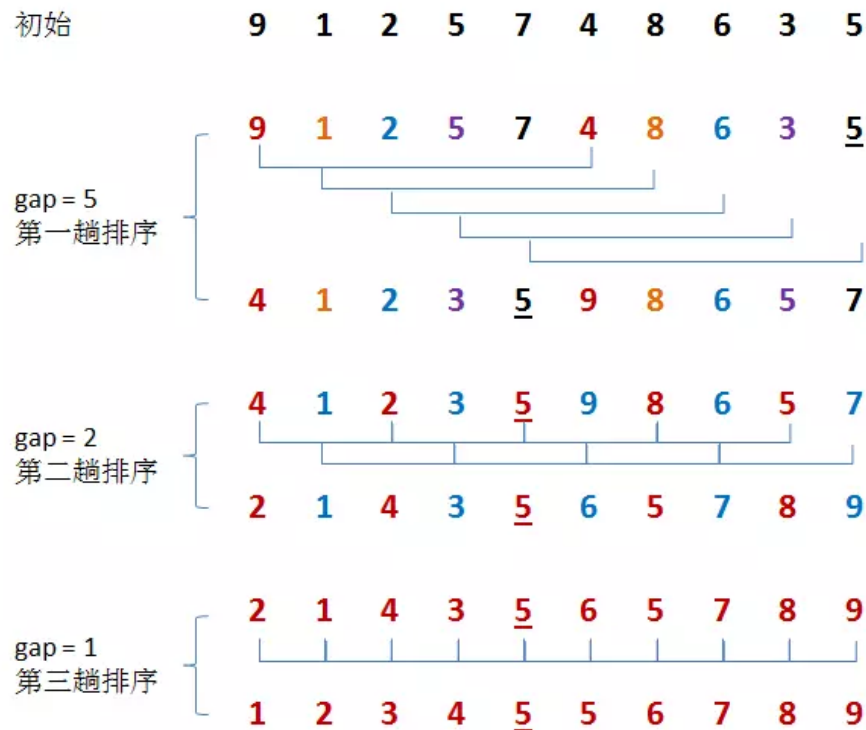


图-希尔排序示例图 Victor Zhang

希尔排序.png

希尔排序的算法思想：将待排序数组按照步长gap进行分组，然后将每组的元素利用直接插入排序的方法进行排序；每次将gap折半减小，循环上述操作；当gap=1时，利用直接插入，完成排序。

同样的：从上面的描述中我们可以发现：希尔排序的总体实现应该由三个循环完成：

- 1. 第一层循环：将gap依次折半，对序列进行分组，直到gap=1

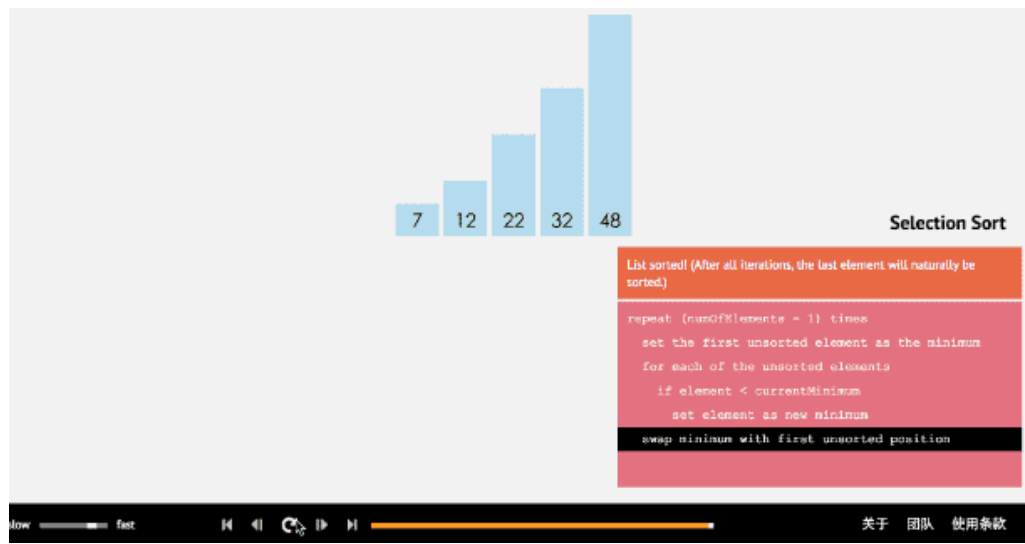
2. 第二、三层循环：也即直接插入排序所需要的两次循环。具体描述见上。

• 代码实现：

```
#希尔排序
def insert_shell(L):
    #初始化gap值，此处利用序列长度的一般为其赋值
    gap = (int)(len(L)/2)
    #第一层循环：依次改变gap值对列表进行分组
    while (gap >= 1):
        #下面：利用直接插入排序的思想对分组数据进行排序
        #range(gap, len(L)):从gap开始
        for x in range(gap, len(L)):
            #range(x-gap, -1, -gap):从x-gap开始与选定元素开始倒序比较，每个比较元素之间间隔gap
            for i in range(x-gap, -1, -gap):
                #如果该组当中两个元素满足交换条件，则进行交换
                if L[i] > L[i+gap]:
                    temp = L[i+gap]
                    L[i+gap] = L[i]
                    L[i] = temp
        #while循环条件折半
        gap = (int)(gap/2)
```

## 简单选择排序

• 算法思想



简单选择排序.gif

简单选择排序的基本思想：比较+交换。

1. 从待排序序列中，找到关键字最小的元素；
  2. 如果最小元素不是待排序序列的第一个元素，将其和第一个元素互换；
  3. 从余下的 N - 1 个元素中，找出关键字最小的元素，重复(1)、(2)步，直到排序结束。
- 因此我们可以发现，简单选择排序也是通过两层循环实现。
- 第一层循环：依次遍历序列当中的每一个元素

+

🔖

❤️

🔗

第二层循环：将遍历得到的当前元素依次与余下的元素进行比较，符合最小元素的条件，则交换。

- 代码实现

```
# 简单选择排序
def select_sort(L):
    #依次遍历序列中的每一个元素
    for x in range(0, len(L)):
        #将当前位置的元素定义此轮循环当中的最小值
        minimum = L[x]
        #将该元素与剩下的元素依次比较寻找最小元素
        for i in range(x+1, len(L)):
            if L[i] < minimum:
                temp = L[i];
                L[i] = minimum;
                minimum = temp
        #将比较后得到的真正最小值赋值给当前位置
        L[x] = minimum
```

## 堆排序

- 堆的概念

堆：本质是一种数组对象。特别重要的一点性质：**任意的叶子节点小于（或大于）它所有的父节点**。对此，又分为大顶堆和小顶堆，大顶堆要求节点的元素都要大于其孩子，小顶堆要求节点元素都小于其左右孩子，两者对左右孩子的大小关系不做任何要求。

利用堆排序，就是基于大顶堆或者小顶堆的一种排序方法。下面，我们通过大顶堆来实现。

- 基本思想：

堆排序可以按照以下步骤来完成：

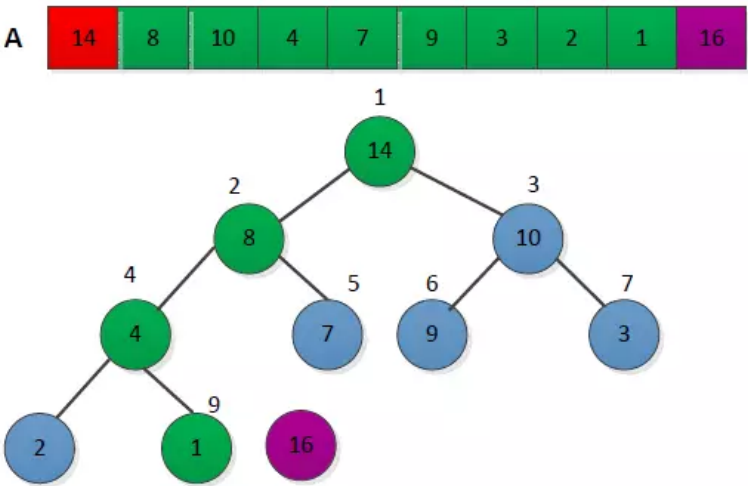
1. 首先将序列构建称为大顶堆；

（这样满足了大顶堆那条性质：位于根节点的元素一定是当前序列的最大值）



构建大顶堆.png

- 2. 取出当前大顶堆的根节点，将其与序列末尾元素进行交换；  
（此时：序列末尾的元素为已排序的最大值；由于交换了元素，当前位于根节点的堆并不一定满足大顶堆的性质）
- 3. 对交换后的n-1个序列元素进行调整，使其满足大顶堆的性质；



Paste\_Image.png

- 4. 重复2.3步骤，直至堆中只有1个元素为止
- 代码实现：

+

🔖

❤️

🔗

```

#-----堆排序-----
#*****获取左右叶子节点*****
def LEFT(i):
    return 2*i + 1
def RIGHT(i):
    return 2*i + 2
#***** 调整大顶堆 *****
#L:待调整序列 length: 序列长度 i:需要调整的结点
def adjust_max_heap(L,length,i):
    #定义一个int值保存当前序列最大值的下标
    largest = i
    #执行循环操作: 两个任务: 1 寻找最大值的下标; 2.最大值与父节点交换
    while (1):
    #获得序列左右叶子节点的下标
        left,right = LEFT(i),RIGHT(i)
    #当左叶子节点的下标小于序列长度 并且 左叶子节点的值大于父节点时, 将左叶子节点的下标赋值给largest
        if (left < length) and (L[left] > L[i]):
            largest = left
            print('左叶子节点')
        else:
            largest = i
    #当右叶子节点的下标小于序列长度 并且 右叶子节点的值大于父节点时, 将右叶子节点的下标值赋值给largest
        if (right < length) and (L[right] > L[largest]):
            largest = right
            print('右叶子节点')
    #如果largest不等于i 说明当前的父节点不是最大值, 需要交换值
        if (largest != i):
            temp = L[i]
            L[i] = L[largest]
            L[largest] = temp
            i = largest
            print(largest)
            continue
        else:
            break
#***** 建立大顶堆 *****
def build_max_heap(L):
    length = len(L)
    for x in range((int)((length-1)/2),-1,-1):
        adjust_max_heap(L,length,x)
#***** 堆排序 *****
def heap_sort(L):
    #先建立大顶堆, 保证最大值位于根节点; 并且父节点的值大于叶子结点
    build_max_heap(L)
    #i: 当前堆中序列的长度. 初始化为序列的长度
    i = len(L)
    #执行循环: 1. 每次取出堆顶元素置于序列的最后(len-1,len-2,len-3...)
    # 2. 调整堆, 使其继续满足大顶堆的性质, 注意实时修改堆中序列的长度
    while (i > 0):
        temp = L[i-1]
        L[i-1] = L[0]
        L[0] = temp
    #堆中序列长度减1
    i = i-1
    #调整大顶堆
    adjust_max_heap(L,i,0)

```

## 冒泡排序

- 基本思想



### 冒泡排序.gif

冒泡排序思路比较简单：

1. 将序列当中的左右元素，依次比较，保证右边的元素始终大于左边的元素；  
（第一轮结束后，序列最后一个元素一定是当前序列的最大值；）
2. 对序列当中剩下的n-1个元素再次执行步骤1。
3. 对于长度为n的序列，一共需要执行n-1轮比较  
（利用while循环可以减少执行次数）

\*代码实现

```
#冒泡排序
def bubble_sort(L):
    length = len(L)
    #序列长度为length，需要执行length-1轮交换
    for x in range(1,length):
        #对于每一轮交换，都将序列当中的左右元素进行比较
        #每轮交换当中，由于序列最后的元素一定是最大的，因此每轮循环到序列未排序的位置即可
        for i in range(0,length-x):
            if L[i] > L[i+1]:
                temp = L[i]
                L[i] = L[i+1]
                L[i+1] = temp
```

## 快速排序

- 算法思想：





## 快速排序.gif

快速排序的基本思想：**挖坑填数+分治法**

1. 从序列当中选择一个基准数(pivot)

在这里我们选择序列当中第一个数最为基准数

2. 将序列当中的所有数依次遍历，比基准数大的位于其右侧，比基准数小的位于其左侧

3. 重复步骤1.2，直到所有子集当中只有一个元素为止。

用伪代码描述如下：

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

• 代码实现：

```
#快速排序
#L：待排序的序列；start排序的开始index,end序列末尾的index
#对于长度为length的序列：start = 0;end = length-1
def quick_sort(L,start,end):
    if start < end:
        i , j , pivot = start , end , L[start]
        while i < j:
            #从右开始向左寻找第一个小于pivot的值
            while (i < j) and (L[j] >= pivot):
                j = j-1
            #将小于pivot的值移到左边
            if (i < j):
                L[i] = L[j]
                i = i+1
            #从左开始向右寻找第一个大于pivot的值
            while (i < j) and (L[i] < pivot):
                i = i+1
            #将大于pivot的值移到右边
            if (i < j):
                L[j] = L[i]
                j = j-1
        #循环结束后，说明 i=j，此时左边的值全都小于pivot，右边的值全都大于pivot
        #pivot的位置移动正确，那么此时只需对左右两侧的序列调用此函数进一步排序即可
        #递归调用函数：依次对左侧序列：从0 ~ i-1//右侧序列：从i+1 ~ end
        L[i] = pivot
        #左侧序列继续排序
        quick_sort(L,start,i-1)
        #右侧序列继续排序
        quick_sort(L,i+1,end)
```

## 归并排序

- 算法思想：

归并排序.gif

1. 归并排序是建立在归并操作上的一种有效的排序算法，该算法是采用分治法的一个典型的应用。它的基本操作是：将已有的子序列合并，达到完全有序的序列；即先使每个子序列有序，再使子序列段间有序。

2. 归并排序其实要做两件事：

- 分解----将序列每次折半拆分
  - 合并----将划分后的序列段两两排序合并
- 因此，归并排序实际上就是两个操作，拆分+合并

3. 如何合并？

L[first...mid]为第一段，L[mid+1...last]为第二段，并且两端已经有序，现在我们要将两端合成达到L[first...last]并且也有序。

- 首先依次从第一段与第二段中取出元素比较，将较小的元素赋值给temp[]
- 重复执行上一步，当某一段赋值结束，则将另一段剩下的元素赋值给temp[]
- 此时将temp[]中的元素复制给L[]，则得到的L[first...last]有序

4. 如何分解？

在这里，我们采用递归的方法，首先将待排序列分成A,B两组；然后重复对A、B序列分组；直到分组后组内只有一个元素，此时我们认为组内所有元素有序，则分组结束。

- 代码实现



```
# 归并排序
#这是合并的函数
# 将序列L[first...mid]与序列L[mid+1...last]进行合并
def mergearray(L,first,mid,last,temp):
#对i,j,k分别进行赋值
    i,j,k = first,mid+1,0
#当左右两边都有数时进行比较，取较小的数
    while (i <= mid) and (j <= last):
        if L[i] <= L[j]:
            temp[k] = L[i]
            i = i+1
            k = k+1
        else:
            temp[k] = L[j]
            j = j+1
            k = k+1
#如果左边序列还有数
    while (i <= mid):
        temp[k] = L[i]
        i = i+1
        k = k+1
#如果右边序列还有数
    while (j <= last):
        temp[k] = L[j]
        j = j+1
        k = k+1
#将temp当中该段有序元素赋值给L待排序列使之部分有序
    for x in range(0,k):
        L[first+x] = temp[x]
# 这是分组的函数
def merge_sort(L,first,last,temp):
    if first < last:
        mid = (int)((first + last) / 2)
#使左边序列有序
        merge_sort(L,first,mid,temp)
#使右边序列有序
        merge_sort(L,mid+1,last,temp)
#将两个有序序列合并
        mergearray(L,first,mid,last,temp)
# 归并排序的函数
def merge_sort_array(L):
#声明一个长度为len(L)的空列表
    temp = len(L)*[None]
#调用归并排序
    merge_sort(L,0,len(L)-1,temp)
```

## 基数排序

- 算法思想



### 基数排序.gif

---

1. 基数排序：通过序列中各个元素的值，对排序的N个元素进行若干趟的“分配”与“收集”来实现排序。

**分配：**我们将 $L[i]$ 中的元素取出，首先确定其个位上的数字，根据该数字分配到与之序号相同的桶中

**收集：**当序列中所有的元素都分配到对应的桶中，再按照顺序依次将桶中的元素收集形成新的一个待排序列 $L'$

对新形成的序列 $L'$ 重复执行分配和收集元素中的十位、百位...直到分配完该序列中的最高位，则排序结束

2. 根据上述“基数排序”的展示，我们可以清楚的看到整个实现的过程

- 代码实现



```

*****基数排序*****
#确定排序的次数
#排序的顺序跟序列中最大数的位数相关
def radix_sort_nums(L):
    maxNum = L[0]
    #寻找序列中的最大数
    for x in L:
        if maxNum < x:
            maxNum = x
    #确定序列中的最大元素的位数
    times = 0
    while (maxNum > 0):
        maxNum = (int)(maxNum/10)
        times = times+1
    return times
#找到num从低到高第pos位的数据
def get_num_pos(num,pos):
    return ((int)(num/(10**((pos-1)))))%10
#基数排序
def radix_sort(L):
    count = 10*[None]      #存放各个桶的数据统计个数
    bucket = len(L)*[None] #暂时存放排序结果
    #从低位到高位依次执行循环
    for pos in range(1,radix_sort_nums(L)+1):
        #置空各个桶的数据统计
        for x in range(0,10):
            count[x] = 0
        #统计当前该位(个位, 十位, 百位....)的元素数目
        for x in range(0,len(L)):
            #统计各个桶将要装进去的元素个数
            j = get_num_pos(int(L[x]),pos)
            count[j] = count[j]+1
        #count[i]表示第i个桶的右边界索引
        for x in range(1,10):
            count[x] = count[x] + count[x-1]
        #将数据依次装入桶中
        for x in range(len(L)-1,-1,-1):
            #求出元素第K位的数字
            j = get_num_pos(L[x],pos)
            #放入对应的桶中, count[j]-1是第j个桶的右边界索引
            bucket[count[j]-1] = L[x]
            #对应桶的装入数据索引-1
            count[j] = count[j]-1
        # 将已分配好的桶中数据再倒出来, 此时已是对应当前位数有序的表
        for x in range(0,len(L)):
            L[x] = bucket[x]

```

## 后记

写完之后运行了一下时间比较：

- 1w个数据时：



```
直接插入排序:11.615608
希尔排序:13.012008
简单选择排序:3.645136000000001
堆排序:0.09587900000000005
冒泡排序:6.687218999999999
#*****
快速排序:9.99999974752427e-07
#快速排序有误:实际上并未执行
#RecursionError: maximum recursion depth exceeded in comparison
#*****
归并排序:0.05638299999999674
基数排序:0.08150400000000246
```

- 10w个数据时:

```
直接插入排序:1233.581131
希尔排序:1409.8012320000003
简单选择排序:466.66974500000015
堆排序:1.20367200000000969
冒泡排序:751.274449
#*****
快速排序:1.0000003385357559e-06
#快速排序有误:实际上并未执行
#RecursionError: maximum recursion depth exceeded in comparison
#*****
归并排序:0.82622300000000272
基数排序:1.1162899999999354
```

从运行结果上来看，堆排序、归并排序、基数排序真的快。

对于快速排序迭代深度超过的问题，可以将考虑将快排通过非递归的方式进行实现。

## 参考资料

- 数据结构可视化: visualgo (<https://link.jianshu.com?t=http://zh.visualgo.net/>)
- 希尔排序介绍: 希尔排序 (<https://link.jianshu.com?t=http://www.cnblogs.com/jingmoxukong/p/4303279.html>)
- 堆排序: (<https://link.jianshu.com?t=http://www.cnblogs.com/Anker/archive/2013/01/23/2873422.html>) 《算法导论》读书笔记之第6章 堆排序 (<https://link.jianshu.com?t=http://www.cnblogs.com/Anker/archive/2013/01/23/2873422.html>)
- 博客园: 静默虚空 (<https://link.jianshu.com?t=http://home.cnblogs.com/u/jingmoxukong/>)
- 博客: vincent-cws (<https://link.jianshu.com?t=http://blog.chinaunix.net/uid/21457204.html>)

小礼物走一走，来简书关注我

赞赏支持





(/u/30ad1e56353e47d)

Python开发 (/nb/5516031)

举报文章 © 著作权归作者所有



LeeLom (/u/3e74cab31591)

写了 14230 字，被 481 人关注，获得了 738 个喜欢

(/u/3e74cab31591)

+ 关注

BYR / iOS / Student Blog: <https://leelom.github.io/>

喜欢 | 568



更多分享



下载简书 App ▶  
随时随地发现和创作内容



(/apps/redirect?utm\_source=note-bottom-click)

被以下专题收入，发现更多相似内容

+ 收入我的专题



首页投稿（暂停... (/c/bDHhpK?utm\_source=desktop&utm\_medium=notes-included-collection)



程序员 (/c/NEt52a?utm\_source=desktop&utm\_medium=notes-included-collection)



unity3D... (/c/6aa2d74ce7da?utm\_source=desktop&utm\_medium=notes-included-collection)



Android... (/c/b1fa46ec3b08?utm\_source=desktop&utm\_medium=notes-included-collection)



程序 (/c/4c142c369b49?utm\_source=desktop&utm\_medium=notes-included-collection)



实用\_学编程 (/c/2d853ec0bb01?utm\_source=desktop&utm\_medium=notes-included-collection)



Java Web (/c/695ba84e62fc?utm\_source=desktop&utm\_medium=notes-included-collection)




展开更多 ▾

(/p/df701d2c43c6?



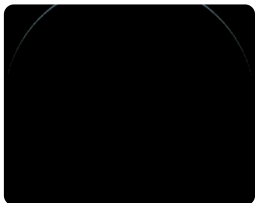
utm\_campaign=maleskine&utm\_content=note&utm\_medium=seo\_notes&utm\_source=recommend  
国际 C 语言混乱代码大赛结果，什么！二次元的世界！！ (/p/df701d2c43c...

第 24 届国际 C 语言混乱代码大赛结果出炉了，之前两位常在该比赛中拿奖的童鞋：浙大的侯启明和 Google 的 Don Yang 又拿奖了。侯启明这次是写了一个无整数的 MD5 程序，Don Yang 写了一个有海星图案的文...

 学习编程\_\_ (/u/16fe1b2bc5df?


utm\_campaign=maleskine&utm\_content=user&utm\_medium=seo\_notes&utm\_source=recommend

(/p/a6549fd7c951?



utm\_campaign=maleskine&utm\_content=note&utm\_medium=seo\_notes&utm\_source=recommend  
程序员必用的电脑桌面 (/p/a6549fd7c951?utm\_campaign=maleskine&ut...

声明：本文首发微信公众号【菜鸟要飞】，如有转载，请标明出处！最近发现了两套开源、神秘、科幻、有逼格的桌面，非常适合在座的各位程序员。NO1、Himawaripy Himawaripy是一个开源的使用 Python 3开...

 rookieflyhigher (/u/0851ecf312d1?


utm\_campaign=maleskine&utm\_content=user&utm\_medium=seo\_notes&utm\_source=recommend

(/p/11775b0ba33e?



utm\_campaign=maleskine&utm\_content=note&utm\_medium=seo\_notes&utm\_source=recommend  
月薪5k与月薪10k程序员的区别 (/p/11775b0ba33e?utm\_campaign=males...

同一个城市，同一个行业，同一个公司，为什么有的程序员可以拿到3万的薪水，而有的却只能拿到3千？这里我们首先排除一系列的杂的东西，比如裙带关系，我们只谈能力和技术，假设公司已经给到程序员能力...

 Android技术干货分享 (/u/06fd4cf1f427?

utm\_campaign=maleskine&utm\_content=user&utm\_medium=seo\_notes&utm\_source=recommend

(/p/626932d735eb?



utm\_campaign=maleskine&utm\_content=note&utm\_medium=seo\_notes&utm\_source=recommend  
程序员刚写完排序就被老板开除，网友笑傻：牛逼了，睡眠排序法？ (/p/62...

老板这是有毛病吧！刚写完排序就叫我直接走人，我TM嫌弃你这9k工资太低呢！此处不留爷，只有留爷处。看到网友全是666，小编也是不由自主的说卧槽。排序算法是我们编程中遇到的最多的算法。目前主流的...

+

🔖

❤️

🔗

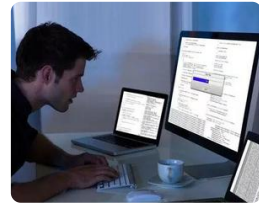




温柔的倾诉 (/u/467379856827?)

utm\_campaign=maleskine&amp;utm\_content=user&amp;utm\_medium=seo\_notes&amp;utm\_source=recommend

(/p/12e34d6ea8f6?)



utm\_campaign=maleskine&utm\_content=note&utm\_medium=seo\_notes&utm\_source=recommend  
**一个BAT老程序员的忠告！** (/p/12e34d6ea8f6?utm\_campaign=maleskine...

一、在中国，你千万不要因为学习技术就可以换来稳定的生活和高的薪水待遇，你更不要认为那些从事市场、运营的人，没有前途。不清楚你是不是知道，咱们中国有相当大的一部分软件公司，他们的软件开发...



程序员江湖\_陆小凤 (/u/9ab8d7b38c4e?)

utm\_campaign=maleskine&amp;utm\_content=user&amp;utm\_medium=seo\_notes&amp;utm\_source=recommend

**突来感想** (/p/bf9aee40ca8e?utm\_campaign=maleskine&utm\_content=n...

重温以前的习惯，却早已不习惯。



麻辣拌 (/u/9281768fff91?)

utm\_campaign=maleskine&amp;utm\_content=user&amp;utm\_medium=seo\_notes&amp;utm\_source=recommend

**梦** (/p/eab1f8b8f51f?utm\_campaign=maleskine&utm\_content=note&ut...

做梦了 上课：两个很胖的女生在演唱然后说自己的家庭经历，我清楚的看到较矮的那个女生胖到衣服不能遮住肚子，只穿一件内衣，胸很大，内衣为非常土的花纹。另一个女生衣服为红色的长款棉袄，能够遮住肚...



bigbiggirl0000 (/u/f071b13f601c?)

utm\_campaign=maleskine&amp;utm\_content=user&amp;utm\_medium=seo\_notes&amp;utm\_source=recommend

(/p/d436676a0606?)



utm\_campaign=maleskine&utm\_content=note&utm\_medium=seo\_notes&utm\_source=recommend  
**【體態肥胖·推薦健康調整飲食】** (/p/d436676a0606?utm\_campaign=mal...

【體態肥胖·推薦健康調整飲食】 1.高纖維食品 - 高纖維食物，如蔬菜，堅果，種子和漿果，增加飽腹感，不增加熱量。 2.潔淨瘦蛋白 - 每餐至少吃3-4盎司蛋白質的人往往易感到更加滿意，並且總體吃得也較少些。...



生命的别样年华 (/u/4b07a70edb6c?)

utm\_campaign=maleskine&amp;utm\_content=user&amp;utm\_medium=seo\_notes&amp;utm\_source=recommend

**姚和张，没必要上升到娱乐政治道德层面** (/p/e7f85b45308d?utm\_campaig...

【转】只是表明态度，伪愤青们请别为提高逼格而批判表明立场了 其实在拿张万年将军和姚贝娜的事大谈特谈 好好想想这根本是扯淡 有些伪愤青跑到一个娱乐因素占据绝对地位的微博和朋友圈 趾高气昂骂民族精...




Ran\_布拉德皮蛋 (/u/f1c0fb7bf368?)

utm\_campaign=maleskine&amp;utm\_content=user&amp;utm\_medium=seo\_notes&amp;utm\_source=recommend



抱歉，我不是你的洛神 (/p/b5a3645f13fe?utm\_campaign=maleskine&ut...

01 夏天的夜，总是黑的晚。晚上八点，千羽来到了预先约定的地方。她抬手看表，八点过了五分。滨江路上的灯已经全部亮起来了。她看到了斜对岸的文峰塔，整个塔身被一层柔和的橘色灯光所笼罩，远远望去...

 晴空下的雨燕 (/u/b5154758eee9?

utm\_campaign=maleskine&utm\_content=user&utm\_medium=seo\_notes&utm\_source=recommend

+

♥

🔗