城市学院论文封皮黑字

**本 科 毕 业 设 计 论 文**



|  |  |
| --- | --- |
| **题 目：** | 最高位优先基数排序算法的应用分析与实现 |
| **姓 名：** | 陈润鹏 |
| **学 号：** | 19001013 |
| **所属学院：** | 计算机系 |
| **专业班级：** | 计科192 |
| **指导老师：** | 王梅 |

**完成日期 2022 年 6月**

摘 要

在社会的高速发展中，计算机已经在人们的生活中息息相关，它的普及使得我们日常的生活更加方便快捷。计算机能够对大量的数据进行处理，最常见的是排序功能，排序算法有冒泡排序、选择排序、插入排序、快速排序，堆排序、基数排序等等。那么针对待排序数据长度过长、待排序的数据距离相差过大、数据类型是字符型等情况进行排序，最高位优先基数排序算法有很大的优势。

本文在阐述大量主流排序算法的基础上，对最高位优先基数排序算法进行了更加深度的分析与探讨。针对生活中常见的一些应用场景进行分析，并且对其中几个十分典型的场景应用算法进行了实现。同时，由于最高位优先基数排序算法和最低位优先基数排序算法都属于基数排序。所以，本文在相关地方对上述的两种算法进行了分析比较，结果显示最高位优先基数排序算法在一些特定的生活应用中更便捷、更高效。

关键词：排序；最高位优先算法；数据结构

ABSTRACT

In the rapid development of society, computers have become closely related to people's lives, and their popularity has made our daily lives more convenient and efficient. The computer can process a large amount of data, the most common is the sorting function. The most common sorting algorithm include bubble sort, selection sorting, insertion sort, quick sorting, heapsort, radix sort, and so on. The highest order first radix sort sorting algorithm has great advantages for sorting when the length of data to be sorted is too long, the distance between data to be sorted is too large, and the data type is character type.

On the basis of expounding a large number of mainstream sorting algorithm, this paper makes a more in-depth analysis and discussion of the highest order first radix sort sorting algorithm. Analyze some common application scenarios in daily life, and implement algorithms in several very typical scenarios. At the same time, because the highest order first radix sort algorithm and the lowest order first radix sort algorithm belong to radix sort. Therefore, this paper analyzes and compares the above two algorithms in relevant places, and the results show that the highest order first radix sort algorithm is more convenient and efficient in some specific life applications.

**KEYWORDS:** High priority algorithm; Data structure; sort

目 录

[1 绪论 7](#_Toc136125418)

[1.1 课题背景及意义 7](#_Toc136125419)

[1.2 本课题研究主要内容 7](#_Toc136125420)

[1.3 本课题国内外研究现状 8](#_Toc136125421)

[2 排序的基本概念和算法 9](#_Toc136125422)

[2.1 排序的概念 9](#_Toc136125423)

[2.1.1 时间复杂度 9](#_Toc136125424)

[2.1.2 空间复杂度 9](#_Toc136125425)

[2.1.3 稳定性分析 9](#_Toc136125426)

[2.2 排序的两大主流算法思想 10](#_Toc136125427)

[2.2.1 比较类排序 10](#_Toc136125428)

[2.2.2 非比较类排序 11](#_Toc136125429)

[2.2.3 比较类排序和非比较类排序特点之间的比较 11](#_Toc136125430)

[2.3 最高位优先基数排序的思想及其算法 13](#_Toc136125431)

[2.3.1 最高位优先基数排序的具体思想 13](#_Toc136125432)

[2.3.2 算法实现 14](#_Toc136125433)

[3 最高位优先基数排序算法的应用分析 17](#_Toc136125434)

[3.1 数据库索引 17](#_Toc136125435)

[3.2 垃圾邮件过滤器 18](#_Toc136125436)

[3.3 编码和解码 19](#_Toc136125437)

[3.4 文本搜索 19](#_Toc136125438)

[3.5 电话簿排序 19](#_Toc136125439)

[4 最高位优先基数排序算法的应用实现 21](#_Toc136125440)

[4.1 解决扑克牌排序问题 21](#_Toc136125441)

[4.2 解决电话号码排序问题 26](#_Toc136125442)

[4.3 解决单词表排序问题 30](#_Toc136125443)

[结论 35](#_Toc136125444)

[致谢 36](#_Toc136125445)

[参考文献 37](#_Toc136125446)

# 绪论

## 课题背景及意义

自第一台计算机诞生以来，计算机产业的飞速发展已远远超出人们对它的预料。快到了几秒就能生产出一台微型计算机，产量猛增，价格低廉，这就使得它的应用范围迅速扩展。如今，计算机已深入到人类社会的各个领域。与此相应的数据也在不断的增加，为了查找方便，通常希望计算机中的表是按关键字有序的。因此，排序就显得尤为重要。

在大量排序算法中，有好多解决不同问题的重要算法，对于不同数据处理的方式存在不同的排序算法。冒泡排序算法经常被用作向学生介绍排序算法和算法复杂性的教学工具；快速排序在相互比较的排序算法中效率算是很优的，常用于金融中对金融数据进行排序，如股价、交易量和市场指数；堆排序用于游戏中对游戏对象的进行排序和管理游戏状态；基数排序，是通过比较各个元素关键字的大小进行排序，既然是数据有起始位置和结束位置，那么就分为最高位优先基数排序和最低位优先基数排序。

数据结构里面的算法可以运用到生活中，研究最高位优先基数算法可以更加了解这个算法。在这个计算机普遍的网络时代 ，最高位基数优先算法的运用可以解决生活中一些特定问题，比如：将大量无序的手机号码进行排序或者是将大量单词按照首字母顺序排序等等。

## 本课题研究主要内容

本文通过对十大主流排序的时间复杂度，空间复杂度，稳定性的三大特点进行了简明的横向分析与对比，分析了比较类排序和非比较类排序，并且对于基数排序的背景下，对最高位优先(MSD)排序和最低位优先(LSD)排序在同一个案例中进行纵向比较，进而突出了最高位优先基数排序处理某些情况下的问题所展示的优越性。

本文对最高位优先算法在一些计算机领域和生活中的应用进行了分析，并且在一些十分具有代表性的问题下，对该算法进行了深入的探究，通过流程图的方式进行了直观的分析与讨论，并且通过代码实现与理论分析相结合的方式进行实现，对于已经实现的应用场景下，总结归纳最高位优先基数算法处理该类型问题相对于其他算法的优越性。

## 本课题国内外研究现状

国内外对MSD基数排序算法进行了研究。一些关键研究主题包括：

1）性能优化：研究人员探索了各种技术来优化MSD基数排序算法的性能，特别是对于大型数据集和具有大范围值的数据集。其中一些技术包括并行实现、缓存感知算法和硬件加速。

2）内存效率：MSD基数排序算法的空间复杂度为O（n+k），其中n是元素的数量，k是每个数字的值范围。研究人员已经研究了降低算法内存需求的方法，例如使用压缩数据结构或利用输入数据的稀疏性。

3）适应特定应用：MSD基数排序算法可以通过修改分区和排序策略来适应特定类型的数据或应用。例如，研究人员开发了用于排序字符串、浮点数和其他专用数据类型的算法变体。

4）理论性质分析：研究人员分析了MSD基数排序算法的理论性质，如其时间和空间复杂性、稳定性，以及与其他排序算法的比较。其中一些研究侧重于开发算法性能的下限，或在某些假设下证明其最优性。

总体而言，MSD基数排序算法是一种强大而通用的排序算法，一直是国内外广泛研究的主题。随着研究人员探索优化其性能、降低内存需求并使其适应新应用的新方法，其研究内容不断发展。

# 排序的基本概念和算法

## 排序的概念

排序是指将杂乱无章的数据元素，通过一定的方法按关键字顺序排列的过程；其目的是将一组“无序”的记录序列调整为“有序”的记录序列。排序分为内部排序和外部排序；若整个排序过程不需要访问外存便能完成，则称为内部排序；反之，则称为外部排序。

### 时间复杂度

时间复杂度可以认为是对排序数据的总的操作次数。反映当n变化时，操作次数呈现什么规律。常见的时间复杂度有：常数阶O(1),对数阶O(log2n),线性阶O(n), 线性对数阶O(nlog2n),平方阶O(n2)。

### 空间复杂度

空间复杂度是指算法在计算机内执行时所需存储空间的度量，它也是问题规模n的函数。

空间复杂度O(1)：当一个算法的空间复杂度为一个常量，即不随被处理数据量n的大小而改变时，可表示为O(1)。

空间复杂度O(log2N)：当一个算法的空间复杂度与以2为底的n的对数成正比时，可表示为O(log2n)ax=N，则x=logaN。

空间复杂度O(n)：当一个算法的空间复杂度与n成线性比例关系时，可表示为0(n)。

### 稳定性分析

假设在待排序的记录序列中，存在多个具有相同关键字的记录，若经过排序这些记录相对次序保持不变，即原序列r[i]=r[j]且r[i]在r[j]前面，而排序之后的序列中r[i]仍然在r[j]前面则称这种排序是稳定的，否则称为不稳定的。

在生活中的一些场景下比如给学生考试成绩进行排序时候，如果两个学生的分数一样该怎么排名，如果取上次考试排名靠前的人，排在本次排名的前面，这样就保证了考试排名的公平性，这就是稳定性的优势所在，相反如果不关注这块内容，则排序是混乱的，是缺少公平性。由此可见，排序过程中稳定性是有必要的。

## 排序的两大主流算法思想

十种常见排序算法可以分为两大类：

1）比较类排序：通过比较来决定元素间的相对次序，由于其时间复杂度不能突破O(nlog2n)，因此也称为非线性时间比较类排序。

2）非比较类排序：不通过比较来决定元素间的相对次序，它可以突破基于比较排序的时间下界，以线性时间运行，因此也称为线性时间非比较类排序。

### 比较类排序

1)冒泡排序（Bubble Sort）

冒泡排序是一种简单的排序算法。它重复地走访过要排序的数列，一次比较两个元素，如果它们的顺序错误就把它们交换过来。走访数列的工作是重复地进行直到没有再需要交换，也就是说该数列已经排序完成。（比较相邻两个元素，每两个都进行比较，然后交换）。

2)选择排序（Selection Sort）

首先在未排序序列中找到最小（大）元素，存放到排序序列的起始位置，然后，再从剩余未排序元素中继续寻找最小（大）元素，然后放到已排序序列的末尾。以此类推，直到所有元素均排序完毕。

3）插入排序（Insertion Sort）

工作原理是通过构建有序序列，对于未排序数据，在已排序序列中从后向前扫描，找到相应位置并插入。

4）希尔排序（Shell Sort）

把记录下标的一定增量分组，对每组使用插入排序算法，随着增量逐渐减少，每组包含的元素越来越多，当增量减至1时，所有元素被分为一组，算法终止。

5）快速排序（Quick Sort）

从冒泡排序算法演变而来的，实际上是在冒泡排序基础上的递归分治法。快速排序在每一轮挑选一个基准元素，并让其他比它大的元素移动到数列一边，比它小的元素移动到数列的另一边，从而把数列拆解成了两个部分。

6）归并排序（ Merge Sort）

建立在归并操作上的一种有效的排序算法，归并排序对序列的元素进行逐层折半分组，然后从最小分组开始比较排序，合并成一个大的分组，逐层进行，最终所有的元素都是有序的。

7）堆排序（Heap Sort）

利用二叉树的概念来排序的选择排序算法，分为两种：

（1）升序排序：利用二叉树建立大堆，进行排序。

（2）降序排序：利用二叉树建立小堆，进行排序。

### 非比较类排序

1)计数排序（Count Sort）

一种非基于比较的排序算法，其核心在于将输入的数据值转化为键存储在额外开辟的数组空间中以达到排序的效果。

2)桶排序（Bucket Sort）

桶排序是计数排序算法的升级版，将数据分到有限数量的桶子里，然后对每个桶再分别进行排序。

3)基数排序（Radix Sort）

将待排序序列的每个元素统一为同样位数长度的元素，位数较短的通过补0达到长度一致，然后从最低位或从最高位开始，依次进行稳定的计数排序，最终形成有序的序列。

基数排序主要是针对整数的排序，由于整数也可以表示字符串或和特定格式的浮点数，因此能用整数表达的其他数据类型也能用基数排序。

基数排序，既可以从高位优先进行排序（简称MSD），也可以从低位优先进行排序（简称LSD）。

### 比较类排序和非比较类排序特点之间的比较

常见的快速排序、归并排序、堆排序、冒泡排序等属于比较排序，在排序的最终结果里，元素之间的次序依赖于他们之间的比较。每个数据都必须和其他数比较，才能确定自己的位置。

1）在冒泡排序之类的排序中，问题规模为n，又因为需要比较n次，所以平均时间复杂度为O(n²)。

2）在归并排序、快速排序之类的排序中，问题规模通过分治法消减为log2N次，所以时间复杂度平均O(nlog2n)。

3）比较排序的优势是，适用于各种规模的数据，也不在乎数据的分布，都能进行排序。可以说，比较排序适用于一切需要排序的情况。

非比较排序是通过确定每个元素之前，应该有多少个元素来排序。针对数组arr，计算arr[i]之前有多少个元素，唯一确定了arr[i]在排序后数组中的位置，非比较排序只要确定每个元素之前的已有的元素个数即可，所有一次遍历即可解决。因此时间复杂度是O(n)常见的非比较排序有三种：桶排序、计数排序、计数排序。它们的时间复杂度都是O(n)，因为这些排序算法的时间复杂度都是线性的，所以也把这类排序算法称为线性排序。之所以能够做到线性的时间复杂度，主要原因是这几个算法是非基于比较的排序算法，不涉及元素之间的比较操作。这几种排序算法的时间复杂度虽然很低，但是对要排序的数据要求比较苛刻，所以关键是要知道这些排序算法的适用场景，非比较排序时间复杂度底，但由于非比较排序需要占用空间来确定唯一位置。所以对数据规模和数据分布有一定的要求。

排序算法的时间复杂度，空间复杂度，稳定性的比较汇总如表2-1所示。

表2-1 排序算法时间复杂度，空间复杂度，稳定性的比较



## 最高位优先基数排序的思想及其算法

最高位优先(Most Significant Digit first)法，简称MSD法：先按k1排序分组，同一组中记录，关键码k1相等，再对各组按k2排序分成子组，之后对后面的关键码继续这样的排序分组，直到按最次位关键码kd对各子组排序后。再将各组连接起来，便得到一个有序序列。

### 最高位优先基数排序的具体思想

基数排序的思路是：是优先级高的先排序，然后收集；再按照次高优先级排序，然后再收集；依次类推，直到最低优先级的那一位。

基数排序可以说是扩展了的桶排序，比如当待排序列在一个很大的范围内0到999999，那么用桶排序是很浪费空间的。而基数排序把每个排序码拆成由d个排序码，比如任何一个6位数（不满六位前面补0）拆成6个排序码，分别是个位的，十位的，百位的…排序时，分6次完成，每次在同一位上的数字进行排序。

最高位优先基数排序的过程：

1)使用条件

基数排序对要排序的数据是有要求的，需要可以分割出独立的“位”来比较，而且位之间有递进的关系，如果 a 数据的高位比 b 数据大，那剩下的低位就不用比较了。除此之外，每一位的数据范围不能太大，要可以用线性排序算法来排序，否则，基数排序的时间复杂度就无法做到 O(n\*k) 了。

2)性能分析

基数排序是一个稳定的排序算法。当输入的元素后，时间复杂度是O(n\*k)，空间复杂度是O(n+k)，其排序速度快于任何比较排序算法。

3)实际应用

假设有10万个手机号码，希望这10万个手机号码从小到大排序，那应该怎么排序。不能用桶排序，因为手机号码有11位，范围太大。这个时候可以使用基数排序。

针对这个问题，假设要比较两个手机号码a、b的大小，如果在前面几位中，a手机号码已经比b手机号码大了，那后面几位号码就不用比较了。这里按照每位来排序的排序算法必须是稳定的，否则这个实现思路就是不正确的。因为如果是非稳定的排序算法，那最好一次排序只会考虑最高位的大小顺序，完全不管其他位的大小关系。

按照每一位来排序，我们可以用基数排序，如果要排序的数据有k位，那就最多需要k次基数排序，例如手机号码排序，k 最大值就是 11。

### 算法分析

最高位优先法通常是一个递归的过程：

先根据最高位关键码K1排序，得到若干对象组，对象组中每个对象都有相同关键码K1。再分别对每组中对象根据关键码K2进行排序，按K2值的不同，再分成若干个更小的子组，每个子组中的对象具有相同的K1和K2值。依此重复，直到对关键码Kd完成排序为止。最后，把所有子组中的对象依次连接起来，就得到一个有序的对象序列。分配排序的基本思想就是进行多次的桶式排序。基数排序过程无须比较关键字，而是通过“分配”和“收集”过程来实现排序。

例如：扑克牌中52 张牌，可按花色和面值分成两个字段，其大小关系为：

花色：梅花< 方块< 红心< 黑心   
面值：2 < 3 < 4 < 5 < 6 < 7 < 8 < 9 < 10 < J < Q < K < A

若对扑克牌按花色、面值进行升序排序，即可得到正确序列：

即两张牌，若花色不同，不论面值怎样，花色低的那张牌小于花色高的，只有在同花色情况下，大小关系才由面值的大小确定。这就是多关键码排序。为得到排序结果，我们讨论两种排序方法。  
方法一：先对花色排序，将其分为4个组，即梅花组、方块组、红心组、黑心组。再对每个组分别按面值进行排序，最后将4个组连接起来即可。流程如图2-2所示：

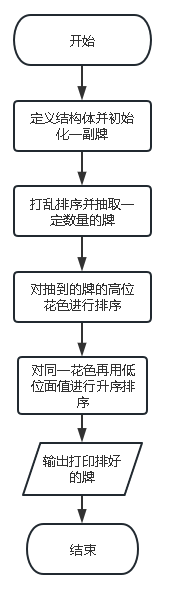


图2-2 按高位排乱序扑克牌的流程图

方法二：先按13个面值给出13个编号组(2 号，3 号，...，A 号)，将牌按面值依次放入对应的编号组，分成13 堆。再按花色给出4个编号组(梅花、方块、红心、黑心)，将2号组中牌取出分别放入对应花色组，再将3 号组中牌取出分别放入对应花色组，……，这样，4个花色组中均按面值有序，然后，将4个花色组依次连接起来即可。流程如图2-3所示：

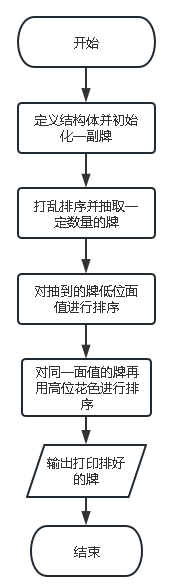


图2-3 按低位排乱序扑克牌的流程图

多关键码排序按照从最主位关键码到最次位关键码或从最次位到最主位关键码的顺序逐次排序，分两种方法：

1）最高位优先(Most Significant Digit first)法在上述案例中的思想总结：

（1）先按花色排序分组，将序列分成若干子序列，同一组序列的记录中，关键码花色相等。

（2）再对各组按面值排序分成子组，之后，对后面的关键码继续这样的排序分组，直到按最次位关键码有序。

（3）再将各组连接起来，便得到一个有序序列。扑克牌按花色、面值排序中介绍的方法一即是MSD法。

2）最低位优先(Least Significant Digit first)法在上述案例中的思想总结：

（1） 先从扑克牌面值开始排序，再对相同面值不同花色的扑克牌进行花色排序。

（2）最后将各个子序列连接起来，便可得到一个有序的序列， 扑克牌按花色、面值排序中介绍的方法二即是LSD法。

LSD基数排序的最大问题是，它从差异最小的数字开始也就是数字的最低位开始。如果可以从最低位有效数字开始，则第一遍将对排序整个范围帮助不明显，此后的每一遍将对更高一位进行处理。

MSD基数排序的想法是将所有具有相等值的数字划分到各自的存储桶中，然后对所有存储桶执行相同的操作，直到对数组进行排序为止。自然建议使用递归算法，但这也意味着我们现在可以对可变长度的数据进行排序，而不必触摸所有数字即可获得排序数组。这使得MSD基数的分类大大加快并且更加有用。

LSD和MSD都是字符串数组排序算法，它们基于所谓的键索引计数而不是基于比较。因此，与传统的快速排序或归并排序相比，LSD和MSD具有不同的运行时间。

# 最高位优先基数排序算法的应用分析

## 数据库索引

当程序从数据库中快速检索数据时，需要使用索引。使用最高基数排序算法排有效数字可以高效准确地构建索引，实现快速的数据检索。通过高位数字对数据进行排序，该算法可以在索引中快速定位所需的数据。这对于具有许多记录的大型数据库尤其重要，因为传统的搜索方法可能太慢或不准确。因此，使用索引并考虑使用最高位优先基数算法是很重要的。

键索引计数法是MSD的基础。其允许通过一定的键值来对字符串进行排序，一个很好的例子就是分组——每个学生对应一个小组，将他们按照小组分开。在这个问题中，学生姓名就是字符串，而小组的组号就是对应的键值，需要将相同小组的同学分到一起，学生分组后的结果如图3-1所示。

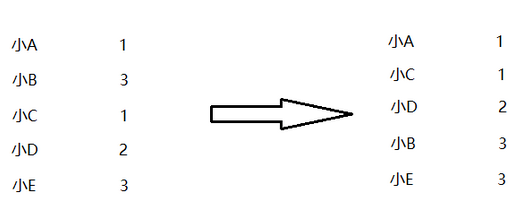


图3-1 将学生分组后的结果

为了完成上述的功能，先统计每个键值的个数并计入一个Count数组中， Count[n + 1]中统计键值为n的字符串的个数，在上图中Count[2] = 2，Count[3] = 1，Count[4] = 2；在统计完成后，需要将频数转换为索引，这是至关重要的一步，在这里用Count[n + 1]来统计键值n而不是Count[n]。使Count[n + 1] += Count[n]，这样Count[0] = 0，Count[1] = 0，Count[2] = 2，Count[3] = 3；可以发现这样变换了以后Count[n]正是下一个键值为n的字符串的索引下标。最后只需要根据索引将字符串存入预先准备的字符串数组及可以完成分组。

MSD的核心思想是分治算法，将大问题分为小问题来解决，其思想与快速排序类似，具体做法是先对最高位的字符进行排序，将排序后的字符串进行分组——最高位相同的在一组；在对同一组的进行MSD排序，不过此时以第二位字符进行排序，直到排完最低位，算法结束。MSD算法的排序过程如图3-2所示。

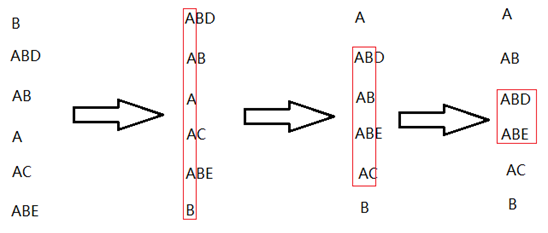


图3-2 MSD算法的排序过程

思想讲起来总是很简单，不过当中的一些细节确实我们需要注意的。 一个显而易见的问题是怎么处理结尾字符的问题，因为MSD运行字符的长度不同，那么总会有字符串先结束，这时候就需要对这些字符串进行处理。

## 垃圾邮件过滤器

垃圾邮件过滤器常常需要分析电子邮件的内容并确定其是否为垃圾邮件。使用最高位优先基数算法可以有效地减少误报率和漏报率。

信头有很多的特征可以用来作为判断是否垃圾邮件。比如群发中的垃圾邮件往往在“To：”域中包含大量的收件人，或者“From：”域中的域名与“Received：”域中的域名不相同。该方法在避免分析邮件正文的前提之下，多数时候能取得比较好的效果。

该算法自动识别表头的“To”并且从该位置从前往后进行检索遇到一个‘@’符号就进行自动给他设置的变量从零开始加加，直到遇到字符‘，’。如果变量大于一定值就自动识别为垃圾邮件进而过滤掉如图3-3所示：

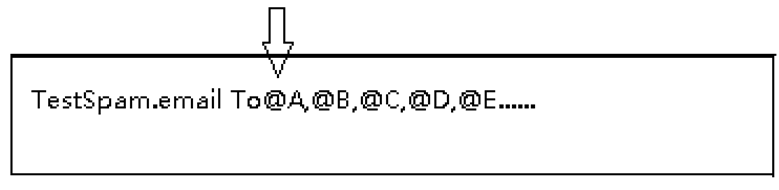


图3-3 模拟电脑通过MSD算法统计邮件的@过程

在垃圾邮件过滤器的上下文中，该算法可以用于有效且准确地分析电子邮件的内容，并确定它是否是垃圾邮件。通过根据数字对电子邮件内容进行排序，该算法可以快速识别垃圾邮件中常见的模式和特征。

总的来说，最高位优先基数排序算法对于各种应用程序是一个有价值的工具，它能够根据数字有效地对数据进行排序，这使它成为分析和处理大型数据集的强大算法。

## 编码和解码

将字符按一定的变法格式转化为字节的方式称为编码，将字节按一定编码格式沾化为字符的方式称为解码。编码和解码通常需要对数据进行排序，以便在传输和存储期间更快地访问数据。最高位优先基数算法可以将数据按照字母或数字的顺序排序，并且这种排序方法易于编码和解码，如图3-4所示：

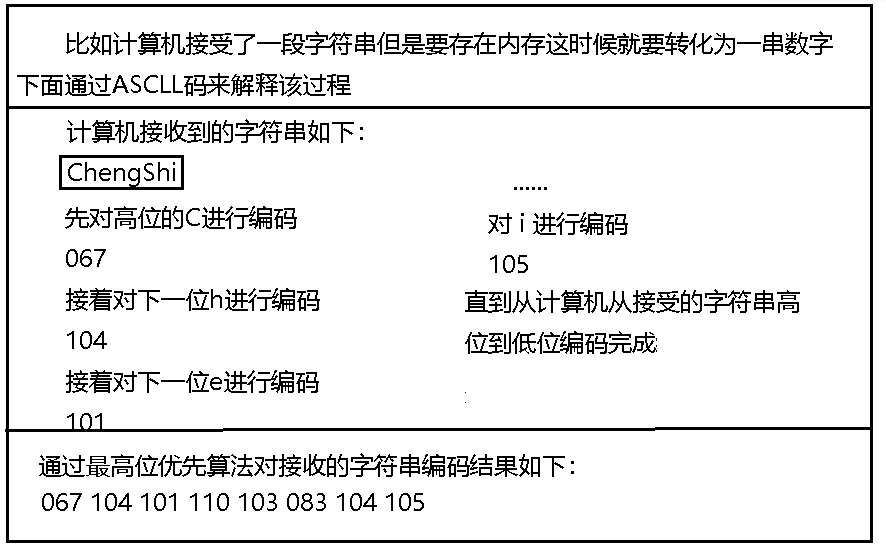


图3-4 模拟计算机内部对字符信息转化编码的过程

在对数据进行编码和解码的情况下，根据数据的数字对数据进行排序对于在传输和存储期间更快、更高效的访问是有用的。通过以这种方式对数据进行排序，该算法可以快速定位所需的数据，并减少编码和解码所需的时间和资源。

## 文本搜索

在文本编辑器中搜索单词或者在搜索引擎中搜索关键字时，需要使用排序算法。最高位优先基数算法可以帮助搜索引擎更快地找到相关的网页。如图3-5所示：

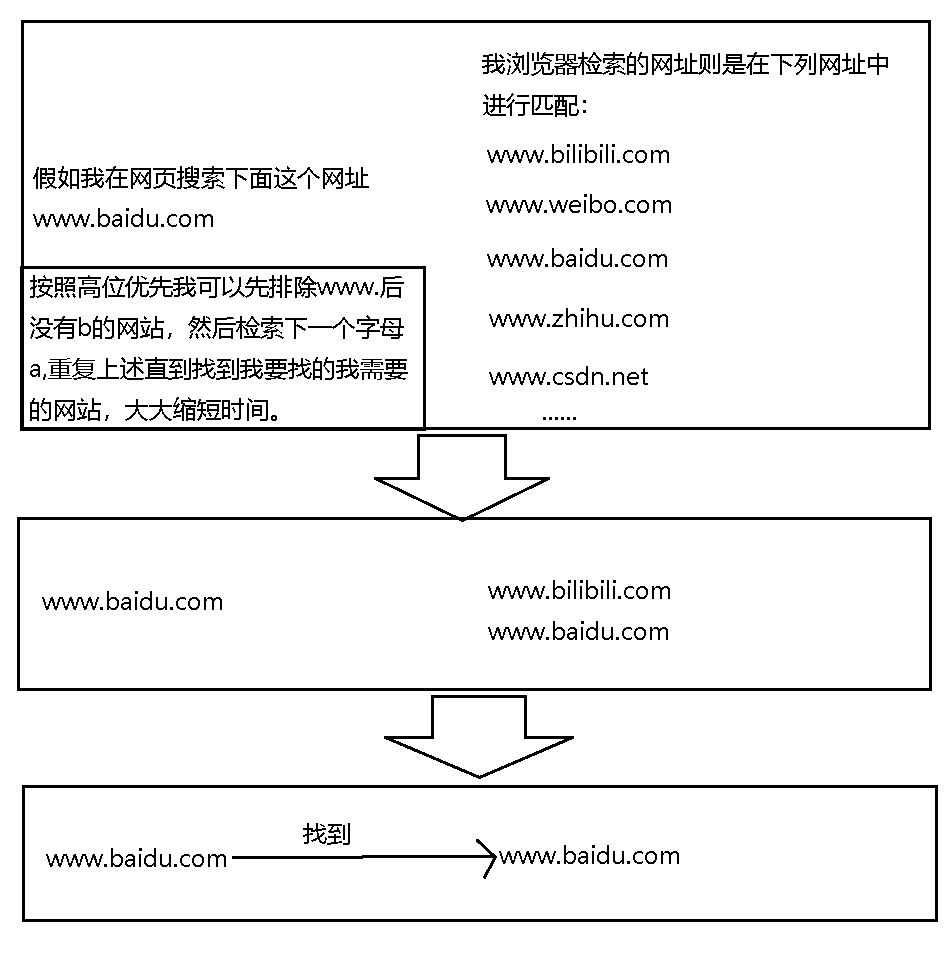


图3-5 模拟高位优先算法找网页的过程

在文本搜索的上下文中，根据文本的数字对文本进行排序对于在搜索过程中更快、更高效地访问非常有用。最高优先基数排序算法可用于根据字母或数字对文本进行排序，从而更容易搜索特定的单词或短语。通过这种方式对文本进行排序，该算法可以快速定位所需的文本，并减少搜索所需的时间和资源。

## 电话簿排序

电话号码经常按照区域和前缀进行排列，以便更快地查找人员的电话号码。最高位优先基数算法可以帮助电话簿更有效地排序。

在电话簿排序的背景下，根据电话号码的数字对其进行排序有助于在搜索过程中更快、更高效地访问。最高优先基数排序算法可用于根据电话号码的区号和前缀对其进行排序，从而更容易定位所需的电话号码。通过以这种方式对电话号码进行排序，该算法可以快速识别电话号码中常见的模式和特征，并减少搜索所需的时间和资源。

在电话簿排序的背景下，该算法可以根据电话号码的区号和前缀快速识别相关电话号码，从而有助于提高电话号码检索的速度和准确性

总的来说，最高优先基数排序算法是一种有价值的工具，适用于各种应用，包括数据库索引、垃圾邮件过滤、数据编码和解码以及文本搜索等领域。它能够根据数字有效地对数据进行排序，这使它成为分析和处理大型数据集的强大算法。

# 最高位优先基数排序算法的应用实现

## 解决扑克牌排序问题

在网络快速发展的今天，人们的休闲娱乐方式随着科技的发展也变得丰富起来了，与此同时网络游戏在年轻人之间十分流行，人们接触过在线斗地主挖坑等等的网络纸牌游戏，在畅玩游戏的同时扑克牌是怎样精确且有序的分发到若干玩家手中？接下来通过最高位优先基数算法对玩家拿到的牌进行分析，并模拟出游戏内的功能。

扑克牌的顺序由两个内容来决定牌序，花色和面值花色分为，梅花，方块，红心，黑心。面值包括从2 -> 3 ->4 ->5 ->6 ->7 ->8 ->9 ->10 ->J ->Q -> K ->A。那么不算大小王的情况下如何将一副杂乱的扑克牌洗成由花色和面值有规律的序列，如何从牌堆中抽到的十五张顺序混乱的扑克牌排成玩家满意的顺序，这一切都离不开排序最高位优先基数排序。

实现了一个扑克牌的排序和洗牌功能，按花色排序流程图如图4-1所示：

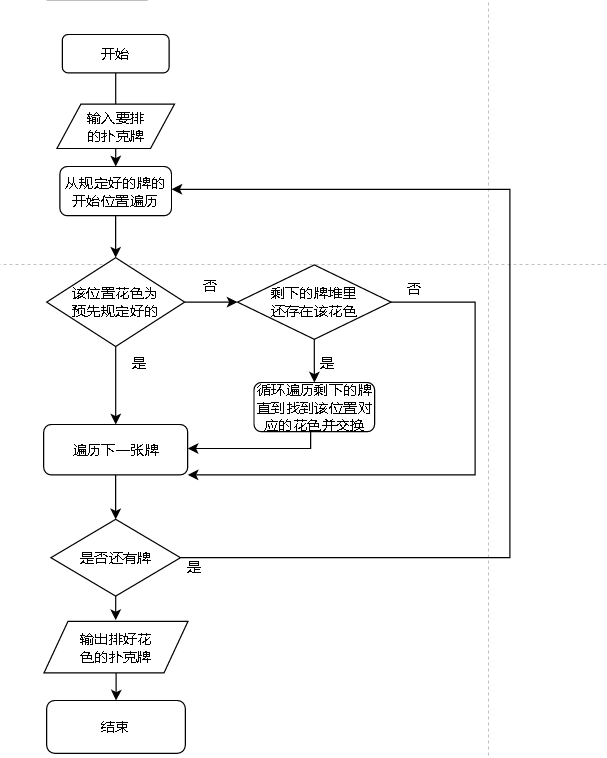


图4-1 对扑克牌花色进行排序流程图

按照花色排完后被分成了四组内容分别对应，梅花，方块，红心，黑心，接着对每种花色内部进行面值排序。按面值排序流程图如图4-2所示：

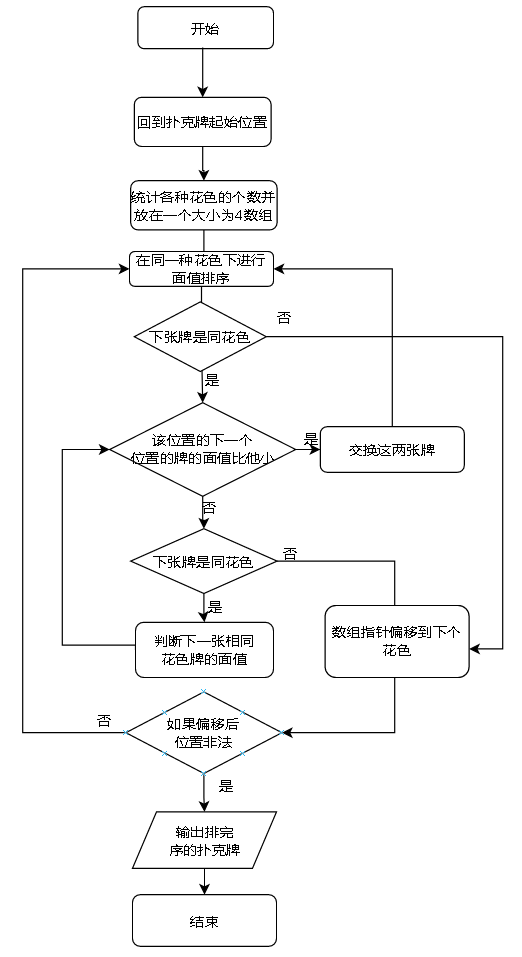
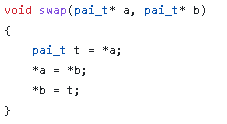


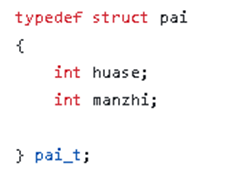
图4-2 按面值排扑克牌流程图

以下是每个函数的作用和结构：  
void swap(pai\_t\*a，pai\_t\*b)：交换两个牌的函数，参数为指向牌的指针。函数内部定义了一个临时变量t，用于交换两个牌的值。  
void init(pai\_tpai[])：初始化牌的函数，参数为指向牌数组的指针。函数内部使用两个循环将所有的牌按花色和面值排列。  
void upset(pai\_tpai[])：洗牌函数，参数为指向牌数组的指针。函数内部使用随机数生成器打乱牌的顺序。  
void print(pai\_tpai[]，int n)：输出牌的函数，参数为指向牌数组的指针和牌的数量。函数内部使用循环将牌按顺序输出。  
void sort(pai\_tpai[]，int n)：排序函数，参数为指向牌数组的指针和牌的数量。函数内部使用两个循环将牌按花色和面值排序。首先按花色排序，然后按面值下面排序。通过C语言来实现上面的思路

Swap()函数对下面要用到的地方进行传地址排序。



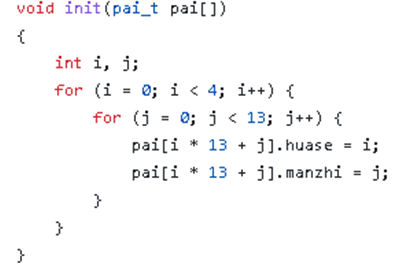
既然要排序，又因为扑克牌是由花色和面值来确定的那么创建一个pai的结构体， 其中包含两个整型变量huase和manzhi，分别表示牌的花色和面值。



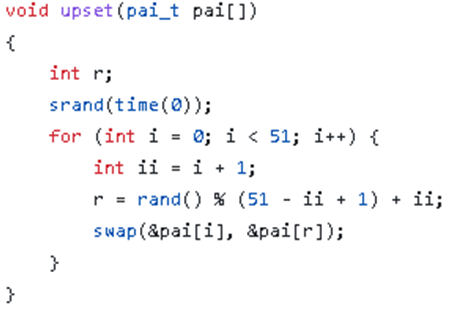
接下来设置两个字符串数组，目的就是针对以上数据进行排序。



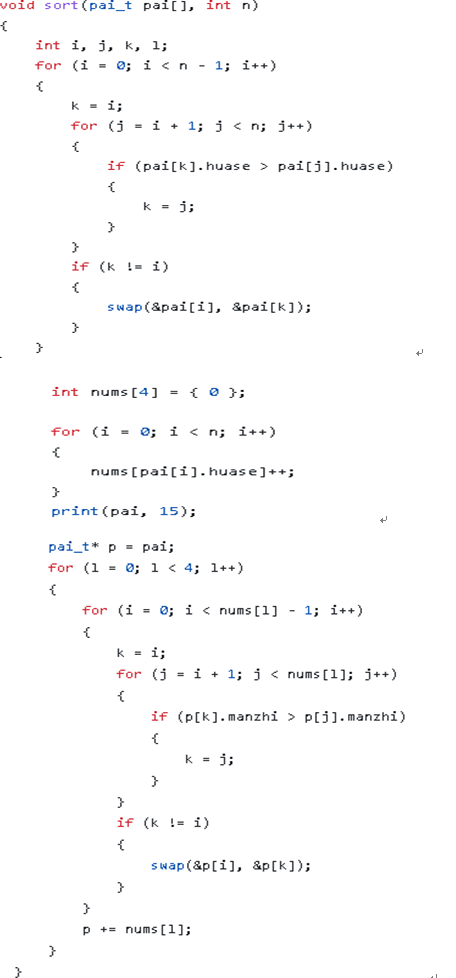
然后就是对扑克牌数组进行初始化，从而形成我需要排序的除了大小王的一组生活中常见扑克牌。



upset()函数是一个洗牌函数，用于打乱牌的顺序。函数的参数为指向牌数组的指针。函数内部使用srand()和rand()函数生成随机数，然后使用swap函数交换两个牌的位置，从而打乱牌的顺序。具体来说，函数使用一个循环遍历牌数组，每次循环生成一个随机数r，然后将当前牌和第r张牌交换位置。循环次数为牌的数量减一，因为最后一张牌不需要交换。



sort()函数是一个排序函数，用于将牌按花色和面值排序。函数的参数为指向牌数组的指针和牌的数量。函数内部使用两个循环将牌按花色和面值排序。首先按花色排序，然后按面值排序。具体来说，函数使用两个循环遍历牌数组，外层循环变量为i，内层循环变量为j。外层循环从第一张牌开始遍历到倒数第二张牌，内层循环从外层循环变量加一开始遍历到最后一张牌。在内层循环中，如果当前牌的花色比第k张牌的花色小，则将k更新为当前牌的下标。内层循环结束后，如果k不等于i，则交换第i张牌和第k张牌的位置。这样就完成了按花色排序的过程。接着，函数使用一个指针p指向牌数组的起始位置，然后使用一个循环遍历四种花色的牌。循环变量为i，从零开始遍历到三。在循环内部，函数使用两个循环将当前花色的牌按面值排序。外层循环变量为i，内层循环变量为j。外层循环从第一张牌开始遍历到倒数第二张牌，内层循环从外层循环变量加一开始遍历到最后一张牌。在内层循环中，如果当前牌的面值比第k张牌的面值小，则将k更新为当前牌的下标。内层循环结束后，如果k不等于i，则交换第i张牌和第k张牌的位置。这样就完成了按面值排序的过程。最后，函数调用print函数输出排序后的前15张牌。函数内部定义了四个整型变量i、j、k和l，分别用于循环计数和记录当前花色的牌的数量。



排序完成后扑克牌的结果如图4-3所示：

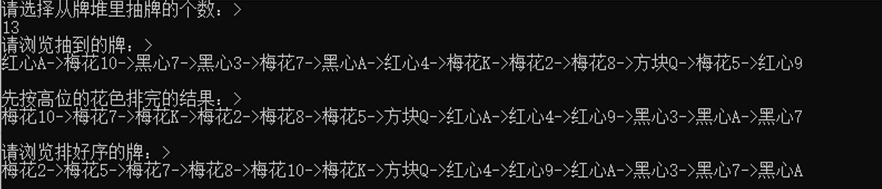


图4-3 按高位优先排完扑克牌的结果

所以上述步骤是一个基数排序的思想对于多关键字分析的一个典型案例，先对最高位(花色)进行排序分桶，然后对最低位(面值)进行整理，最终完成对扑克牌的总体排序。这样排序体现了该算法对于多关键字的排序，解决了其他一些算法对于多关键字类型数据排序无从下手的问题。

## 解决电话号码排序问题

在生活中在移动营业厅办理注册手机号码的时候，都会看到手机业务员拿着几张密密麻麻写满手机号的纸张，提供给消费者来挑选，这时候如果对于所提供的纸张留意的朋友会不会惊奇的发现，在大量的手机号里面，所有号码都是有规律整齐排放的，比如137\*\*\*\*7861，137\*\*\*\*7862，137\*\*\*\*7863以此类推，排列这些手机号码的数字，写一个希尔排序或者是快速排序就行，但是这些排序算法反复的比较手机号数字就显得特别麻烦，但是使用高位优先基数排序算法它可以用于对电话号码、身份证号码等这类问题进行排序。与普通排序算法相比，最高位优先基数排序算法在数字方面排序具有更加高效的优势。提高排序的效率和稳定性。

通过高位优先的排序思想解决大量手机号的排序算法，解决此类问题，不至于挑选手机号时候让顾客眼花缭乱。

所以上述问题实质是一个基数排序的思想的又一个典型案例，先对手机号码最高位进行排序分桶和收集，然后对下一位进行整理，由于这种处理方法是相同的，所有这使程序不得不对上述的重复性动作进行递归操作，最终直到切分的内容是一个单位或者零个单位时候就是我们递归的终止，从而完成我们对电话号码的总体排序。

为了更加直观的理解该算法对于手机号码的排序，接下来便对它进行直观分析。如图4-4所示：

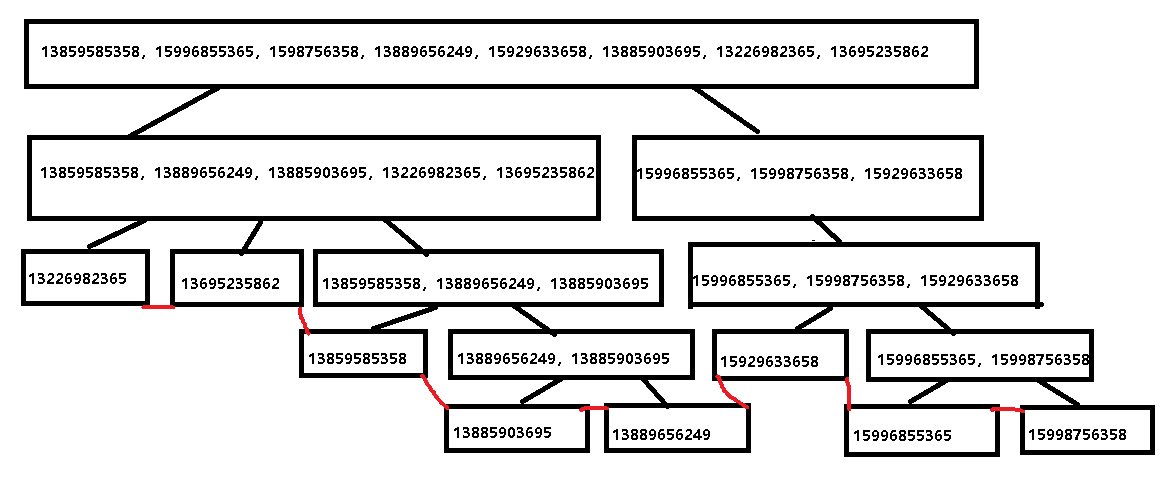


图4-4 按高位优先排电话号码过程直观图

手机号排序的最终结果，如图4-5所示：

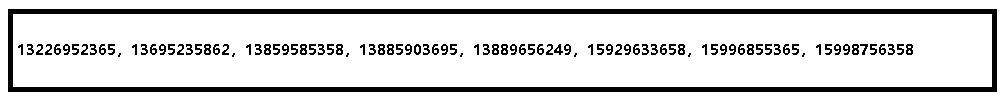


图4-5按高位优先排序最终排好的预期图

流程图如4-6所示：

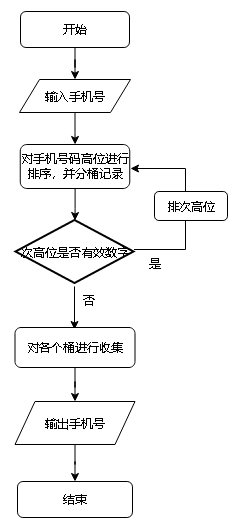


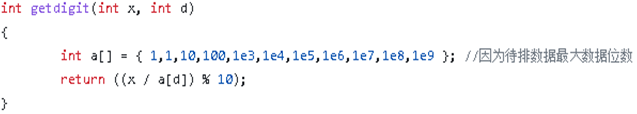
图4-6 按高位排抽到的乱序手机号的流程图

下面我通过C语言来实现上面的思路。



arr[MAXN]是一个数组，它的作用是存储待排序的整数。在给定的代码中，arr[MAXN]被定义为全局变量，其大小为MAXN，即数组中最多可以存储MAXN个整数。在msdradix\_sort()函数中，会使用arr[MAXN]数组中的元素进行排序。具体实现中，msdradix\_sort()函数会将arr[MAXN]数组中的元素按照当前位数的数字进行分组，并将它们存储在桶数组bucket中。

排序完成后，将排序后的结果复制回arr[MAXN]数组中。因此，arr[MAXN]数组在最高位优先基数排序算法中扮演着重要的角色，它用于存储待排序的整数，并参与到排序的过程中。



getdigit()函数是一个用于获取一个整数的某一位数字的函数。它接受两个参数，分别是一个整数x和一个整数d，其中x表示待获取数字的整数，d表示待获取数字的位数。

在函数内部，首先定义了一个大小为11的数组a，用于存储10的幂次方。然后，根据待获取数字的位数d，从数组a中获取对应的幂次方，将其除以x，并对10取模，即可得到待获取数字的某一位数字。最后，将该数字作为函数的返回值返回。

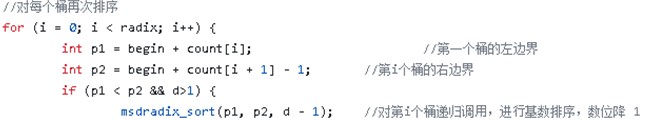
总的来说，getdigit()函数是一个用于获取一个整数的某一位数字的函数，它的实现比较简单，只需要使用数组和取模运算即可。



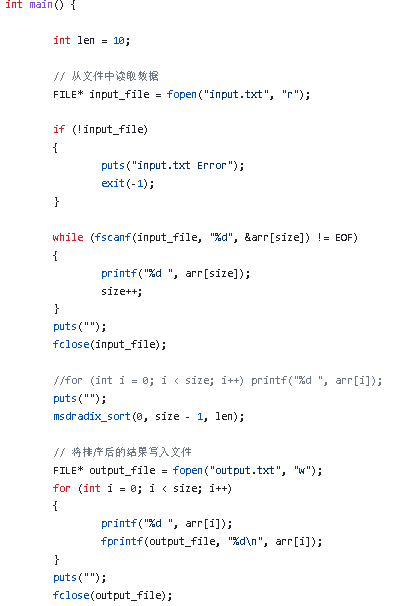
然后，遍历整个数组arr，将每个数字按照当前位数的权值进行分组，并将每组中数字出现的次数记录在计数数组count中。接着对计数数组count进行前缀和操作，以便确定每个数字在排序后的数组中的位置。然后，从后往前遍历数组arr，将每个数字按照当前位数的权值分配到对应的位置上，并将计数数组count中对应数字的计数减1。最后，将排序后的数组存储在桶数组bucket中，并将其复制回原数组arr中。

接下来，递归调用msdradix\_sort()函数，将当前位数的权值加1，以便对下一位数进行排序。这个过程会一直持续到所有位数都被排序完毕为止。

总的来说，msdradix\_sort()函数是一个使用最高位优先基数算法对数组进行排序的函数，它的实现思路比较巧妙，使用了桶排序和递归的方式进行排序。



既然有了该算法主题部分的实现那么，主函数遍迎刃而解，为了贴近现实生活中对电话号码的排序这里使用了文件操作，这样在使用该算法的过程中，方便使用者高效的对文件进行修改。



排序完成后手机号的结果如图4-7所示：

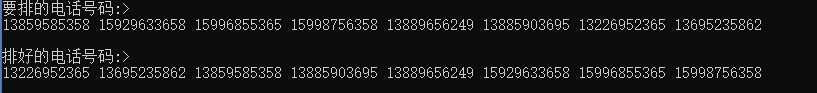


图4-7 手机号排序的结果

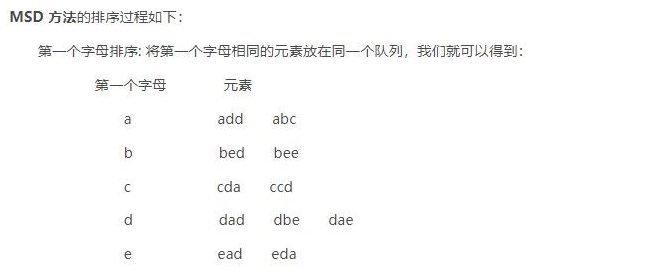
最高位优先基数算法排序算法是一种高效、稳定的排序算法，它可以用于对等长数字进行排序。那么在此情境下相对于最低位优先基数算法，高位优先就体现了优秀的一面，既然最高位已经排出来高低次序了那么就不用在一步一步的去比较低的位数，在实际状况中就节省了时间。

假如计算机要处理一个小区的所有人口的身份证信息，再按最高位优先基数排序来排列身份证号码，这可足足有十八位，那么如果从低位开始整理要整整递归十八次，这还是一个小区的人口普查，那么如果我放到全区，全县甚至是全省，计算机对如此庞大的数据源进行处理，那么高位优先对于身份证号这个关键字的排序的高效性便是不言而喻的。

## 解决单词表排序问题

在现如今的学校中，大学对于英语课程学习的重视程度是非常高的，同学肯定也会接触背单词这件事情，翻到英语书后面的单词表，会看到第一个开始肯定是从A开始的，例如abrupt ，abstraction ，acquisition… 等等这些单词。这些单词确实看起来很整齐，首字母从a开始到的下一个单词也是从a开始的，例如：abrupt和abstraction同样是a开头的字母，为什么abrupt排在前面，而abstraction才排到第二个位置？因为这样排出来可以更加快捷的找到要查找的单词，只要在含有大量单词的词汇书上，从首位开始找对应的字母，然后锁定符合首位的单词列表，再对次一位的字母位置找到想要找的字母，最后循环上述的过程，最终会找到要查找到对应的单词并找到它所对应的汉语翻译。

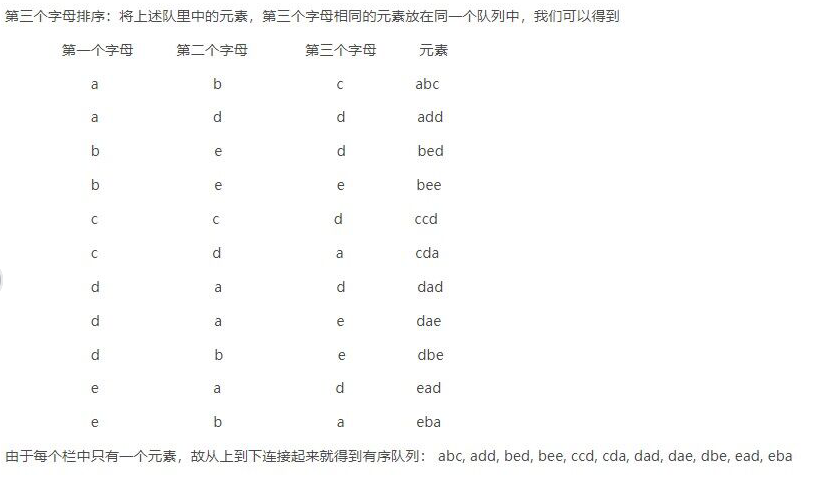
为了解决这字符串排序类问题，可以使用最高位优先基数排序算法。在检索单词的时候也是从最高位开始检索，正因如此，高位优先排序便十分的恰当的去解决大量的且不同长度的字符串的排序问题。虽然最低为优先基数算法同样可以解决大量字符串排序问题，但是高位优先算法从字符串的最高位开始进行数据间的对比，更加的节约排序所花费的时间 ，这一点是低位优先排序所做不到的。下面我们对含有a、b、c、d、e，5个字母的字符串进行排序，由于只排下面这五个字母，所以使用的关键字是5。过程如图4-8（a）（b）（c）所示：



（a）排第一个字母



（b）排第二个字母



（c）排第三个字母

图4-8 最高位优先基数算模拟排列字母序列

具体实现流程图如4-9所示：

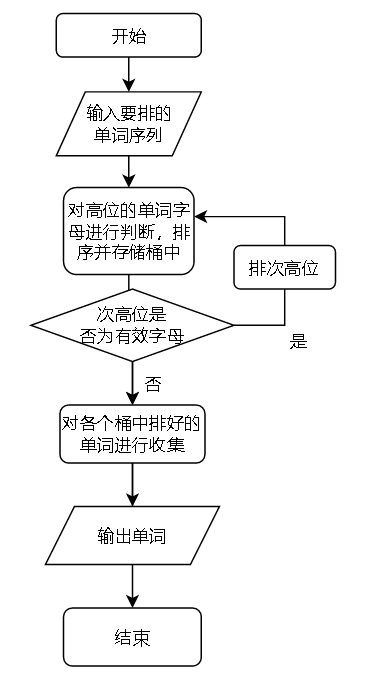
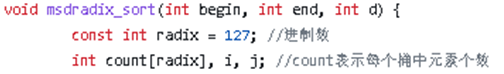


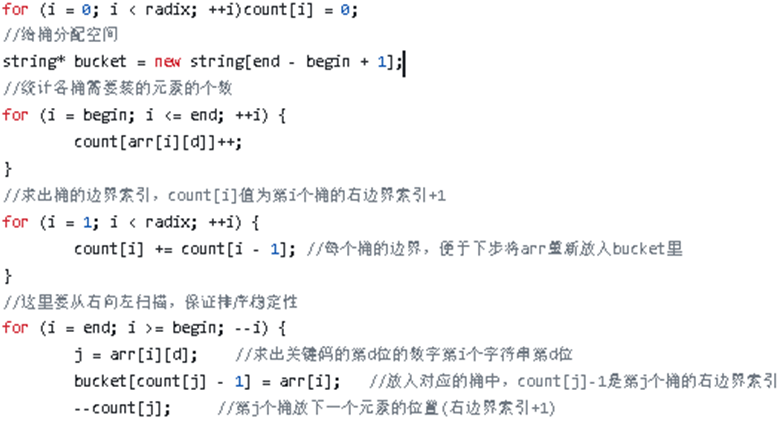
图4-9 按高位排抽到的乱序单词表的方法

下面通过C语言来实现上面的思路

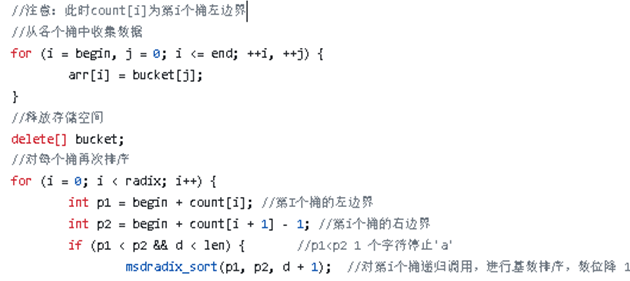


代码行的const int radix=127；将恒定整数变量基数初始化的值为127。此变量用作代码中实现的MSD基数排序算法的基数。基数决定了排序过程中使用的桶数，并根据输入数据中的值范围进行选择。在这种情况下，使用基数127，这是对ASCII字符进行排序的常见选择。MSD基数排序算法是一种稳定高效的排序算法，用于对等长字符串进行排序。

代码行int count[radix]，i，j；声明了三个整数变量：count、i和j。count是一个基数大小的整数数组，基数是代码中初始化为127的常量整数。这个数组用于在排序过程中存储每个bucket中的元素数量。int count[radix]；count是一个整数数组，用于在排序过程中存储每个bucket中的元素数量。



接下来初始化count数组，以存储每个bucket中的元素数量。对字符串数组进行排序。具体来说，它初始化计数数组，为桶数组分配内存，并根据当前数字d计算每个桶中的元素数量。这是最高有效数字基数排序算法中的关键步骤，该算法用于根据字符串或整数的数字对其进行排序。for循环从头到尾迭代数组中的元素，对于每个元素，它会根据字符串的第d个字符递增相应bucket的计数。此步骤对于确定分拣过程中将放置在每个桶中的元素数量是必要的。随后的for循环计算每个bucket的右边界，并将元素分类为各自的元素。



紧接着对字符串数组进行排序。具体来说，它从每个bucket中收集已排序的元素，并将它们存储回原始数组中。这是最高有效数字基数排序算法的关键一步，该算法用于根据字符串或整数的数字对其进行排序。for循环从头到尾迭代数组中的元素，对于每个元素，它将bucket数组中的相应元素分配给arr数组。这个步骤对于从每个bucket中收集已排序的元素并将它们存储回原始数组是有必要的。

随后的delete[]语句释放为bucket数组分配的内存。最后，如果有更多的数字要排序，函数会在每个bucket上递归调用自己。if语句检查是否有更多的位数要排序，以及当前bucket的左边界是否小于右边界。如果这两个条件都为真，则函数在当前bucket上递归调用自身，d参数递增1。此步骤对于根据下一个数字对每个bucket中的元素进行排序是必要的。

排序完成的英文单词结果如图4-10所示：：

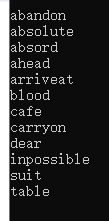
输入： 输出：

图4-10 英文单词排序结果

在主函数中通过对文件的处理方式对单词进行录入，然后以文件方式再输出。

在分析完上述所举的例子之外，可以理解最高位优先技术算法(MSD)是种基数排序算法，特别适用于处理字符串和数字的列表。该算法开始于最高位，从高到低依次进行排序，依据每个数字在各个位上的大小关系。它在现实生活中有许多应用。

结 论

本次对最高位优先基数排序算法对其应用和其思想有了深刻的了解，也使得自己的知识更加丰富，知道最高位优先基数算法的应用可以运用到我们实际的生活中去。因为之前也只是课堂上在数据结构课中对于排序的各个算法有了一些基本了解但是还是不够深刻的。也正是这次的灵活运用排序的基本算法最高位优先基数排序让自己更加深刻的了解这个算法的魅力和作用。能够灵活的实现和解决一些生活之中出现的一些难题。而且该问题的源码实现运用是结合了平时所学的基础知识来完成的，这不仅巩固了之前所学习的知识并且还对自己会有所提高。使得看待问题的角度和方式方法产生了细微的变化，对自己来说是比较好的有益于自己的。在进行这个系统完成的时候也出现了一些问题，但是在不懈的努力下还是被攻克了，也让自己学会在出现问题的时候要保持冷静然后慢慢一条一条的发现问题逐层递减来看待和处理，而不是焦头烂额的急躁这样反而对问题的解决是没有任何的帮助。在此次的毕业设计中我相对于以前的自己是进步的相信未来也会继续的进步提高，这次过程带给自己的是对以后的路程中要循序渐进，对自己以后的发展还是有很大的帮助的。

这次的毕业设计也参考了其他优秀同学他们的建议取长补短，取其精华弃其糟粕，使得自己的设计更加的尽善尽美。在我看来这次是非常好的经历，在不断的尝试中逐步的成长，自己也学会了很多在老师同学的帮助下我相信自己会越来越好，正是他们的鼓励和支持，使得自己的自信心增加。

参考文献

1. 严蔚敏，吴伟民，数据结构（C语音版）[M].北京：清华大学出版社，1998.
2. 陈小平.数据结构导论[M].北京：经济科学出版社，2000.
3. 李春葆.数据结构(C 语言篇) [ M] .北京:清华大学出版社， 2002
4. 王春森.系统设计师( 高级程序员) 教程[ M] .北京:清华大学出版社， 2001.
5. 谭浩强.C语言程序设计（第四版）[M].北京：清华大学出版社，2010.
6. 殷人昆，等.数据结构（用面向对象方法与C++描述）[M].北京：清华大学出版社，1999.
7. 黄水松，董红斌，数据结构与算法习题解析[M].北京：电子工业出版社，1996.
8. 朱建莉.刘宏强.常用排序算法综述[J].胜利 油田师范专科学校学报，2002年04期
9. 何宇晨.严晶涛等.排序算法的性能比较及基数排序在数据分类中的应用 [J].
10. 高涛陆丽娜C语言程序设计[M].西安：西安交通大学大学出版社，2007.
11. Colloquium on Computing，Communication， Control， and Management (CCCM2010) Volume 1[C];2010年.