城市学院论文封皮黑字u

**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**

**本科毕业设计（论文）任务书**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 题目 | 基于连通图的深度优先算法的应用分析及实现 | | | | | |
| 姓名 | 陈润鹏 | 专业 | 计算机科学与技术 | | 学号 | 19001013 |
| 1.毕业设计（论文）课题的主要任务（说明：课题背景、意义和培养目标，工程设计类课题应有技术经济分析要求）：  排序是数据处理中最常用的运算之一 , 排序操作将一组无序的数据序列按某种次序重新排列，从而得到一组有序的数据序列，以方便用户查找，提高检索数据效率。比较关键字和移动记录是实现排序算法的两个基本操作，在经典排序算法中，基数排序是一种非比较型排序算法，利用“分配”和“收集”两种基本操作实现排序的算法。 | | | | | | |
| 2.课题的具体工作内容（原始数据、技术要求、案例分析资料、工作要求等。说明：工科类“技术要求”要有指标参数的具体要求。）：  　　本课题主要详细分析最高位优先基数排序算法的基本思想，算法复杂度，稳定性等特性，并用 C 或者 C++ 高级语言实现该算法，在此基础上，通过与其他排序算法的比较，研究分析该算法在实际应用中能解决的现实问题。   　　基数排序将关键字看作是以某个正整数为基的数，然后依次按关键字的各个字位分别对记录进行排序。本文主要从以下几个方面分析研究最高位优先基数排序算法：   　　1 ）熟悉数据结构，C或C++编程环境。   　　2 ）分析最高位优先基数排序算法的基本思想，算法结构、算法特性。   　　3 ）与其他常用排序算法的综合比较。   　　4 ）算法模块设计。   　　5 ）利用C或C++编程语言实现该算法。   　　6 ）分析研究该算法的具体应用。 | | | | | | |
| 3.课题完成后提交的书面材料要求（论文字数，图纸规格、数量，实物样品，外文翻译字数等）：  (1)毕业论文的字数15000字以上。   (2)毕业设计(论文)用A4纸打印，统一正规装订。   (3)完成课题相关的外文技术资料翻译，外文字符需达到20000字符以上。   (4)主要提交的材料有毕业设计任务书，毕业设计论文(电子版)。   (5)提交源代码。 | | | | | | |
| 4.主要参考文献（指导教师应向学生列出明确的参考文献（含外文文献）一般不少于5篇，不包括词典、手册。学生论文最终引用参考文献不限于任务书所列文献，须达到要求篇数）：   1. 谭浩强 . C语言程序设计（第四版） [M].北京：清华大学出版社，2010. 2. 严蔚敏.吴伟明.数据结构（C语言版）[M].北京：清华大学出版社，2007. 3. 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]. 4. 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. 5. Colloquium on Computing,Communication, Control, and Management (CCCM2010) Volume 1[C];2010年. 6. 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. | | | | | | |
| 要求完成日期： 2023年 05月 27日  指导教师（签名）： 王梅 | | | | 接受任务日期：2023年 01月 04日  学生（签名）： 陈润鹏 | | |

**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。为了研究相同长的或者不同长度的数据，对他们进行排序，引出了一种排序算法，最高位优先基数排序。将大量同类型但是各个元素直接有差别的数据整理成为有序的组合，这个就叫排序算法，本文在对比大量其他主流排序算法的基础上，对基数排序进行了深度的探讨，并且通过分析和实现。结果显示该算法在生活中一些地方带来了便捷和高效，为一些特定情况下的排序问题提供了解决方案。

关 键 词：高位优先算法；数据结构；排序

**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. In order to study the data of the same length or different lengths and sort them, a sorting algorithm, the highest order first radix sort sorting, is introduced. A large number of data of the same type but directly different from each element are sorted into orderly combinations, which is called sorting algorithm. Based on the comparison with a large number of other mainstream sorting algorithm, this paper makes an in-depth discussion on radix sort, and through analysis and implementation. The results show that this algorithm has brought convenience and efficiency in some areas of life, providing solutions for sorting problems in specific situations.

**KEY WORDS:** High priority algorithm; Data structure; sort

目 录

[1绪论 1](#_Toc15680)

[1.1课题背景 1](#_Toc21635)

[1.2本课题研究意义 1](#_Toc15159)

[1.3本课题研究方法 1](#_Toc29137)

[2排序的基本概念和搜索算法](#_Toc32083)

[2.1排序的概念 3](#_Toc16012)

[2.2 排序的性质 4](#_Toc14998)

[2.3排序的两大主流算法思想 4](#_Toc1671)

[2.3.1比较类排序 4](#_Toc9466)

[2.3.2非比较类排序 6](#_Toc15230)

[2.3.3比较类排序和非比较类排序特点之间的比较 8](#_Toc30796)

[2.4最高位优先基数排序的思想及其算法 10](#_Toc32516)

[2.5最高位优先基数排序在生活中的的思想及案例说明](#_Toc14565)

[12](#_Toc14565)

[3最高位优先基数排序的应用分析与实现](#_Toc20697)

[3.1最高位优先基数排序算法解决扑克牌花色和面值排序问题 15](#_Toc26918)

[3.2最高位优先基数排序算法解决电话号码排序问题……………………18](#_Toc11505)

[3.3最高位优先基数排序算法解决不同长度字符串问题排序问题…… 20](#_Toc29670)

[结 论 29](#_Toc18229)

[致 谢 30](#_Toc32572)

[参考文献 31](#_Toc20723)

1绪论

1.1课题背景

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。

1.2本课题研究意义

数据结构里面的算法可以运用到生活中，研究最高位优先技术算法可以更加了解这个算法。使最高位优先算法灵活的运用到生活的各个领域当中去。在这个计算机普遍，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，或者是将大量单词通过a,b,c…的最高位开始的整齐顺序来完成有规律的排列整合

1.3本课题研究方法

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。   
例子：原数组73, 28, 93, 43, 55, 14, 22, 65, 39, 81   
MSD: 首先按第十位分组，得到14, 22, 28, 39, 43, 55,65,73, 81, 93，第二次对于十位相同的数进行个位的分组，得到14, 22, 28, 39, 43, 55, 65, 73, 81, 93。   
  下面我们描述MSD法的一种易于并行化的实现算法概述 可以看出,MSD法实际上是按某个关键字位对当前得到的各个子序列分别进行排序，而且各个子序列的排序过程可以独立地进行假设待排序的n个记录存放在数组R[1,…,n]中,排序过程中出现的各个子序列仍然依次存放在数组R[1,…,n]中为了对某个子序列排序,我们需要知道该子序列在数组R[1,,n]中的起始位置,该子序列的长度即子序列中记录的个数以及对该子序列进行排序时所用的关键字位为此我们用一个循环队列Q[0,…,n-1]中存放各个子序列在数组R[1,…,n]中的起始位置,子序列的长度和对该子序列进行排序时所用的关键字位在开始排序时,仅有一个子序列即给定的记录序列,因而初始时队列Q中仅存放给定记录序列的起始位置1,长度n,排序关键字位1在按关键字位j对一个子序列排序时,我们采用下列方法:首先计数kji=t(0≤t≤r-1)的个数,将其记录在数组NUM[0,…,-1]的第1个单元NUM[t]中,并计算关键字位kji=t的第一个记录在该子序列中的起始位置,并用数组 POS[0,…,r-1]的第t个单元 POS[t]记录若该子序列在数组R[1,…,n]中的起始位置为 Start,显然有 POS[0]= Start,   
POS[t]= POS[t- 1]+NUM[t- 1],(0 ≤t≤r-1)   
  然后依次扫描该子序列,若当前记录R的第j位关键字kji=t,则将R:放入辅助数组B[1,…,n]的第 POS[t]个位置,然后置POS[t]=POS[t]+1待按关键字位j对各个子序列排好序后,再将数组B写回数组R,这时该子序列已按第j位排好序了,对上述方法的一个改进是,在按关键字位j对一个子序列排序时，首先检查该子序列的第j位关键字是否全相同,若全相同,则在j<d时继续对该子序列的第j+1位关键字进行排序,否则用上述方法对该子序列的第j位关键字排序。

2排序的基本概念和最高位优先基数排序算法

2.1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，因为数据元素过多,不能同时放在内存中根据排序过程的要求，不能在内存和外存之间移动数据的排序。

2.2 排序的性质

### 1、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)

   时间复杂度O(1)：算法中语句执行次数为一个常数，则时间复杂度为O(1),

### 2、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)

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)

### 3、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。

2.3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：

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(nlogn)，因此也称为非线性时间比较类排序。

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。

2.3.1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，没两个都进行比较，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）**

是把记录按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）**

是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）**

是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）**

是利用二叉堆的概念来排序的选择排序算法，分为两种：

升序排序：利用最大堆进行排序

降序排序：利用最小堆进行排序

。

2.3.2 非比较类排序

**8、计数排序（Count Sort）**

是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

**9、桶排序（Bucket Sort）**

桶排序（Bucket sort）是计数排序算法法的升级版，将数据分到有限数量的桶子里，然后每个桶再分别排序

**10、基数排序（Radix Sort）**

是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）

2.3.3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、冒泡排序等术语**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。每个数都必须和其他数比较，才能确定自己的位置。**

* 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)。
* 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logN次，所以时间复杂度平均O(nlogn)。
* **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，则唯一确定了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低，但是对要排序的数据要求比较苛刻，所以我们**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**。

2.4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是**：

基数排序的思路是：是按照低位先排序，然后收集；再按照高位排序，然后再收集；依次类推 ，直到最高位。有时候有些属性是有优先级顺序的，先按低优先级排序，再按高优先级排序。最后的次序就是高优先级高的在前，高优先级相同的低优先级高的在前。

**基数排序可以说是扩展了的桶式排序，比如当待排序列在一个很大的范围内，比如0到999999内，那么用桶式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，每次按第i个排序码来排。

基数排序一般借助桶式排序，每次按第N位排序时，采用桶式排序。对于如何安排每次落入同一个桶中的数据有两种安排方法：

* 顺序存储：每次使用桶式排序，放入r个桶中，相同时增加计数。
* 链式存储：每个桶通过一个静态队列来跟踪。

步骤：

* 得到数组中的最大数，并取得位数
* arr为原始数据，从最低位开始取每个为组成radix数组；
* 对radix进行基数排序（利用计数排序适用于小范围数的特点）；

**最高位优先基数排序排序的过程：**

# 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) 了。

# 性能分析

计数排序是一个稳定的排序算法。当输入的元素是 n 个 0到 k 之间的整数时，时间复杂度是O(n+k)，空间复杂度也是O(n+k)，其排序速度快于任何比较排序算法。当k不是很大并且序列比较集中时，计数排序是一个很有效的排序算法。

# 实际应用

假设我们有10万个手机号码，希望这10万个手机号码从小到大排序，那应该怎么排序呢？

分析：

* 不能用桶排序和计数排序，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，那号码几位就不用看了。
* 借助稳定排序算法，我们可以先按照最后一位来排序手机号码，然后，再按照倒数第二位来重新排序，以此类推，最后按照第一位重新排序。经过11次排序之后，手机号码就有序了。
* 注意，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，我们可以用桶排序或者计数排序，它们的时间复杂度可以做到O(n)。如果要排序的数据有k位，那我们就需要k次桶排序或者计数排序，总的时间复杂度是O(k\*n)。当k不大的时候，比如手机号码排序的例子，k 最大就是 11，所以基数排序的时间复杂度就近似于 O(n)。

**算法实现**：

（1）MSD法实现

最高位优先法通常是一个递归的过程：

<1>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。

<2>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。

<3>依此重复，直到对关键码Kd完成排序为止。

<4>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。

**2.5最高位优先基数排序在生活中的的思想及案例说明**

分配排序的基本思想：说白了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。它们的时间复杂度可达到线性阶：O(n)。

**实例:**

　　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1**：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。  
**方法2**：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。

**两种多关键码排序方法：**

　　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：

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

　　1）先按k1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k1相等。

　　2）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。

　　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。

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

　　1) 先从kd 开始排序，再对kd-1进行排序，依次重复，直到按k1排序分组分成最小的子序列后。

　　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。如果我们可以从最高有效数字开始，则第一遍将对排序整个范围大有帮助，此后的每一遍将仅处理细节。

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。自然地，这建议使用递归算法，但这也意味着我们现在可以对可变长度的项目进行排序，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。

如《算法》一书中所读，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合并排序相比，LSD和MSD具有不同的运行时间

3最高位优先基数排序算法在生活应用场景

1.数据库索引

当应用程序需要从数据库中快速检索数据时，需要使用索引。使用最高位优先技术算法可以更快速而准确地构建索引。

在数据库索引的背景下，最高有效数字基数排序算法可以用于高效、准确地构建索引，以实现快速数据检索。通过根据数据的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有用，因为传统的搜索方法可能太慢或不准确。因此，当应用程序需要从数据库中快速检索数据时，使用索引并考虑使用诸如最高位优先基数之类的算法是很重要的

2.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。使用最高位优先技术算法可以有效地减少误报率和漏报率。

在垃圾邮件过滤器的上下文中，该算法可以用于有效和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。这有助于降低误报率和漏报率，误报率和误报率是评估垃圾邮件过滤器有效性的重要指标。

总的来说，最高有效数字基数排序算法对于各种应用程序是一个有价值的工具，包括数据库索引和垃圾邮件过滤。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。

3.编码和解码

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。最高位优先技术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。

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。最高有效数字基数排序算法可用于根据字母或数字对数据进行排序，从而更容易对数据进行编码和解码。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。

4.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。最高位优先技术算法可以帮助搜索引擎更快地找到相关的网页。

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。最高有效数字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。

5.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。最高位优先技术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。最高有效数字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。

总的来说，最高有效数字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。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。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。

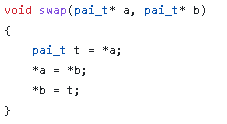
4最高位优先基数排序算法的应用分析与实现

4.1最高位优先基数排序算法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，你周围的朋友也一定接触过在线斗地主挖坑等等的纸牌游戏，在你畅玩游戏的同时有没有想到过扑克牌是怎样精确且有序的分发到若干玩家手中，这就是我现在所要提到的问题

有没有发现，扑克牌的顺序无非由两个内容来决定你的牌序，花色和面值花色分为,黑桃,红桃,方块,梅花。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排洗成由花色和面值规律的排序，如何将你从牌堆中抽到的十五张顺序混乱的扑克牌排成你满意的顺序，这一切都离不开今天谈论的排序最高位优先基数排序

下面我通过C语言来实现上面的思路



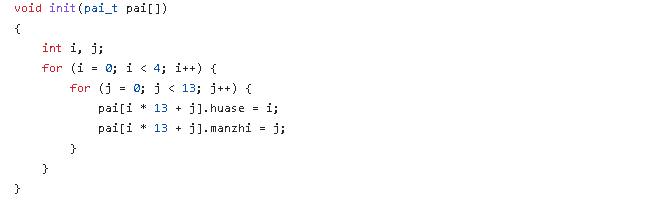
对下面要用到的地方进行传地址排序



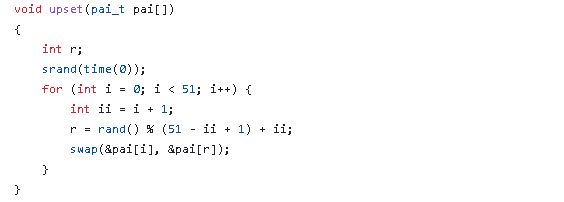
既然要排序，又因为扑克牌是由花色和面值来确定的那么创建一个pai的结构体, 其中包含两个整型变量 huase 和 manzhi，分别表示牌的花色和面值



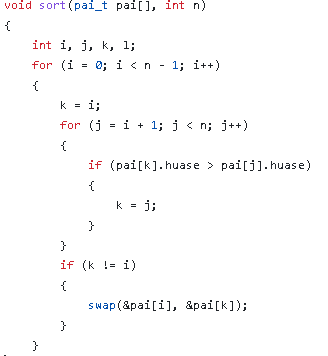
接下来设置两个字符串数组,目的就是针对以上数据进行排序

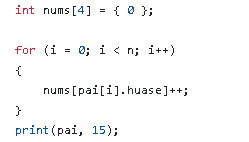


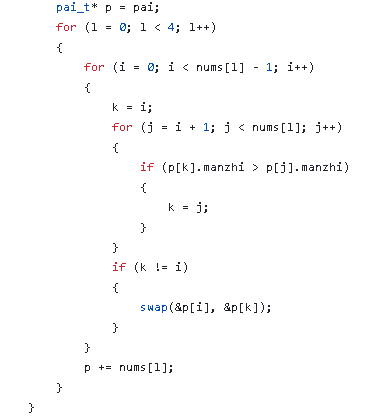
接着就是对扑克牌数组进行初始化,从而形成我需要排序的除了大小王的一组生活中常见扑克牌。



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。



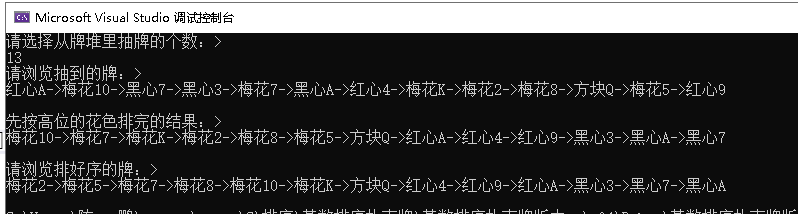




Sort() 函数是一个排序函数,用于将牌按花色和面值排序。函数的参数为指向牌数组的指针和牌的数量。函数内部使用两个循环将牌按花色和面值排序。首先按花色排序,然后按面值排序。具体来说，函数使用两个循环遍历牌数组,外层循环变量为 i,内层循环变量为 j。外层循环从第一张牌开始遍历到倒数第二张牌，内层循环从外层循环变量加一开始遍历到最后一张牌。在内层循环中,如果当前牌的花色比第 k 张牌的花色小,则将 k 更新为当前牌的下标。内层循环结束后，如果 k 不等于 i,则交换第 i 张牌和第 k 张牌的位置。这样就完成了按花色排序的过程。接着，函数使用一个指针 p 指向牌数组的起始位置,然后使用一个循环遍历四种花色的牌。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。

函数总结就是如下：

这个代码实现了一个扑克牌的排序和洗牌功能。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\_t pai[])：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\_t pai[])：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\_t pai[], 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\_t pai[], 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，然后按面值排序。

 所以上述步骤是一个基数排序的思想一个典型案例，先对最高位(花色)进行排序分桶，然后对面值进行整理，最终完成对扑克牌的总体排序

在通过上述对于扑克牌的最高位优先基数排序的应用分析与实现之后是否对该算法有了初步的认识,既然这样来看看下一步，在生活中的其他应用

4.2 最高位优先基数排序算法解决电话号码排序问题

在生活中我们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,供消费者来挑选,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,那么这虽然这是数字,有人就会说,写个希尔排序或者是快速排序不就行了,但是有没有想过这样排序反复的比较就显得特别麻烦,可以看看下面对于最高位优先基数排序算法是一种常用的排序算法，它可以用于对电话号码、身份证号码等字符串进行排序。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以下我所总结的优势：

1. 时间复杂度低：最高位优先基数排序算法的时间复杂度为O(dn)，其中d为的手机号最大位数也就是0到9一共为10，n为手机号数字字符串的个数，也就是手机号的位数。虽然时间复杂度比较高，但是在实际应用中，由于d通常比较小，因此该算法在手机号排序的领域来讲,效率还是比较高的。

2. 稳定性好：最高位优先基数排序算法是一种稳定的排序算法，它可以保证相同数字的顺序不会改变。这在某些场景下非常重要，就如电话簿中需要按照电话号码进行排序时，就需要使用稳定的排序算法。

3. 适用范围广：最高位优先基数排序算法不仅可以用于数字的排序，还可以用于字符串的排序。例如，可以使用该算法对电话号码、身份证号码等字符串进行排序。

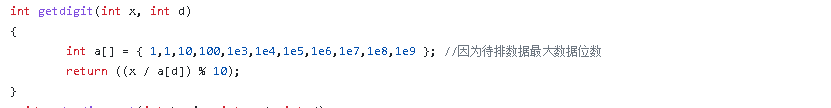
因此，最高位优先基数排序算法在数字方面排序具有较大的优势,特别是在需要对电话号码、身份证号码等字符串进行排序时,该算法可以发挥出其优势，提高排序的效率和稳定性。

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。

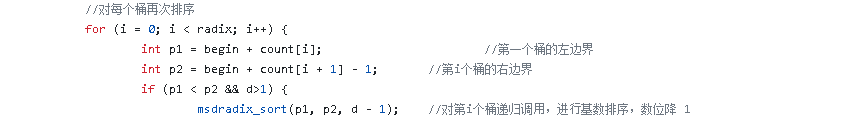
总的来说，getdigit()函数是一个用于获取一个整数的某一位数字的函数，它的实现比较简单，只需要使用数组和取模运算即可。



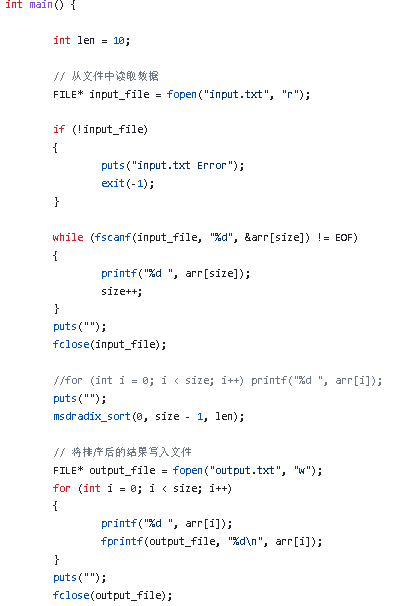
然后，遍历整个数组arr，将每个数字按照当前位数的权值进行分组，并将每组中数字出现的次数记录在计数数组count中。接着，对计数数组count进行前缀和操作，以便确定每个数字在排序后的数组中的位置。然后，从后往前遍历数组arr，将每个数字按照当前位数的权值分配到对应的位置上，并将计数数组count中对应数字的计数减1。最后，将排序后的数组存储在桶数组bucket中，并将其复制回原数组arr中。

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



所以上述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，然后对我的下一位进行整理，然后处理次高位，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。

为了让读者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

如有以下数据来模拟手机号码(现实中手机号是11位，这里用3位来模拟即可)

278,109,063,064,930,589,184,505,269,083

按最高位排序后，原关键字序列分解为下列子序列

{063,064,008,083}, {109,184}, {278,269}, {589,505}, {930}

按次高位分别对，上述各子序列排序后得到.

{008},{063, 064},{083}, {109},{184},{269}, {278},{505}, {589}, {930}

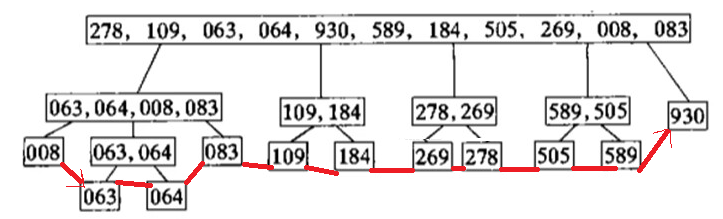
按最低位分别对上述各子序列排序后得到.

{008}, {063}, {064}, {083},{109}, {184},{269},{278}, {505}, {589}, {930},

最后再将所有的子序列依次联接在一起就得到排好序的序列.

008, 063, 064, 083, 109, 184, 269, 278. 505. 589, 930

下面是直观的图片分析：



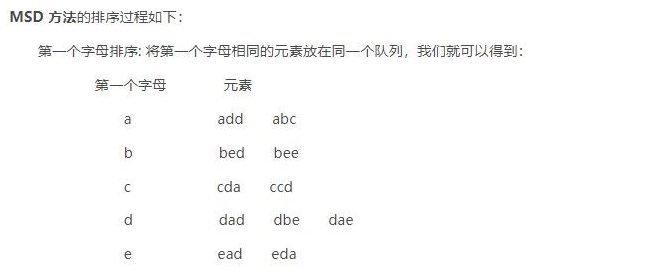
最高位优先基数算法排序算法是一种高效、稳定的排序算法，它可以用于对等长数字进行排序。那么在此情境下相对于最低位优先基数算法,高位优先就体现了它优秀的一面，既然我最高位已经排出来高低次序了那么我就不用在一步一步的去比较比较低的位数那么在实际状况中就省去了一大笔时间,在这反面大家可能没什么感觉,那么再举一个例子,如果是入口大普查呢,假如计算机要处理一个小区的所有入口的身份证信息，再按最高位优先基数排序来排列身份证号码，这可足足有十八位！！！那么如果从低位开始整理要整整递归十八次,这还是一个小区的入口普查，那么如果我放到全区，全县甚至是全省,计算机对如此庞大的数据源进行处理,那么高位优先对于身份证号这个关键字的排序的高效性便是不言而喻的。

在通过上述对于手机号最高位优先基数排序的应用分析与实现之后,是否对该算法有了进一步的认识呢。

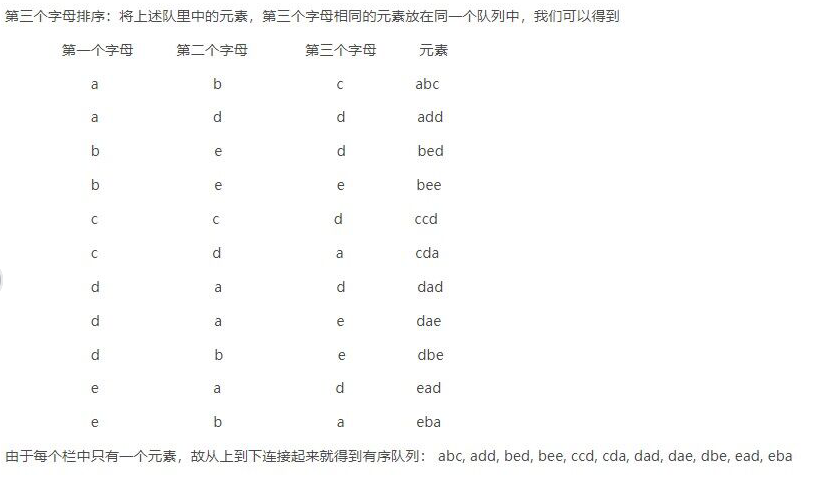
4.3 最高位优先基数排序算法解决不同长度字符串问题排序问题

在现如今的学校中，相必大学对于英语的重视程度是非常高的,就一般高校吧，对于大一大二的学生来说，一周好几节英语课是躲不开的事情,同学们肯定也会接触背单词这件事情，那么翻到英语书后面的单词表,我们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,但是有没有发现对于这两个单词,第二个字母就不一样了，就拿我举的例子来说,我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,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，从首位开始找我要对应的字母,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,最终一定会找到想查找的对应的单词并且找到它所对应的汉语翻译。

那么听完上面的阐述，相必各位读者有了清楚了解，没错，这对于大量无规则的单词通过一种排序，进而形成规则的序列的排序方法跟最高位优先基数排序有着紧密的联系。有没有发现，我们检索单词的时候也是从最高位开始检索的，估计没有人背单词是倒着背的，正因如此，高位优先排序便十分的贴切的去解决大量的不同长度的字符串的排序问题。







注意上面例子(一般而言关键字是26)，但由于i只排上面这五个字母使用关键字是五。

之前介绍过最低位优先基数排序算法,请读者们仔细想一想,低位优先算法适合这种环境下的排序问题吗？

答案很明显,不适合。

因为低位优先算法,必须在稳定的情况下才可以对待排序的数据处理，

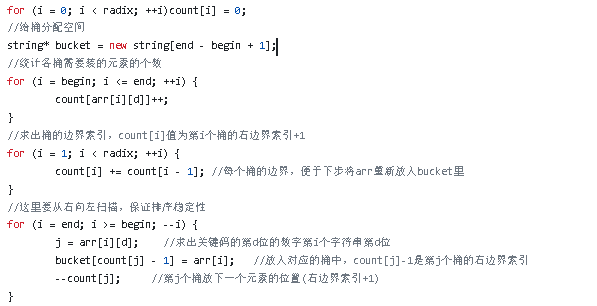
但是高位优先算法,可以稳定处理也可以不稳定处理,并且高位优先可以处理随机字符串。

下面我通过C语言来实现上面的思路

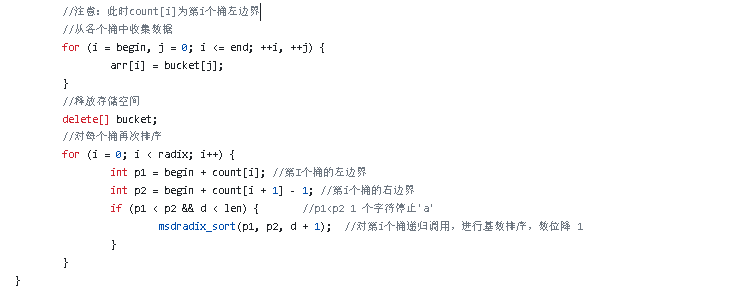


代码行的const int radix=127；将恒定整数变量基数初始化为值127。此变量用作代码中实现的MSD基数排序算法的基数。基数决定了排序过程中使用的桶数，并根据输入数据中的值范围进行选择。在这种情况下，使用基数127，这是对ASCII字符进行排序的常见选择。MSD基数排序算法是一种稳定高效的排序算法，用于对等长字符串进行排序。

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。

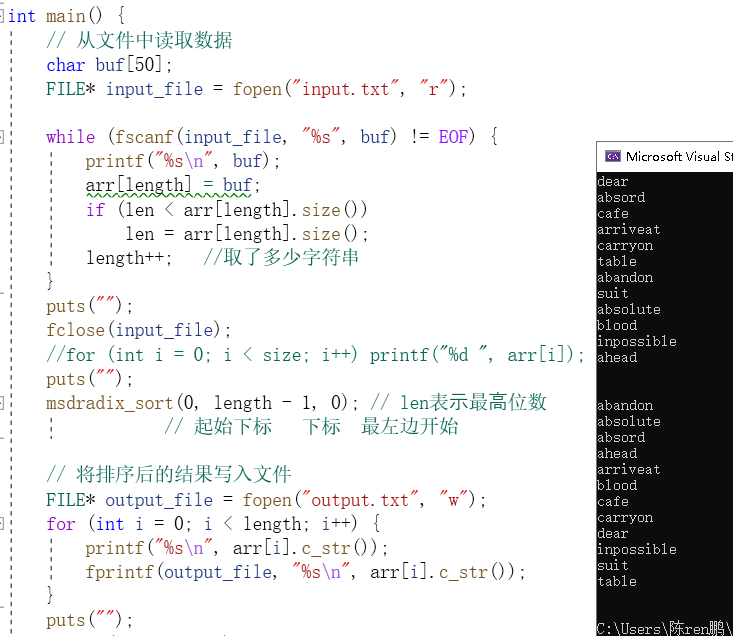


接下来初始化count数组，以存储每个bucket中的元素数量。对字符串数组进行排序。具体来说，它初始化计数数组，为桶数组分配内存，并根据当前数字d计算每个桶中的元素数量。这是最高有效数字基数排序算法中的关键步骤，该算法用于根据字符串或整数的数字对其进行排序。for循环从头到尾迭代数组中的元素，对于每个元素，它会根据字符串的第d个字符递增相应bucket的计数。此步骤对于确定分拣过程中将放置在每个桶中的元素数量是必要的。随后的for循环计算每个bucket的右边界，并将元素分类为各自的元素



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。



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。

致 谢

时光荏苒，如白驹过隙，转眼间就到了大学生涯结束的时刻。毕业论文的完成也随之接近了尾声，在次期间完成的艰辛和遇到的困难，在老师，同学，朋友的帮助和鼓励支持下也顺顺利利的完成了。在此聊表心意十分感谢，这个论文过程中提过的建议和方法，使得进度更加快，整体完成度有所提高。

这个毕业设计和论文也是第一次来进行，开始的时候懵懵懂懂总会出一些小小的差错，从选题开题再到设计完成论文一系列的程序。就像一个小白的一样慢慢摸索，相应的在此期间的问题也浮现出来比如什么格式不懂呀没有理解透彻或者设计过程中遇到的难题，出现的bug之类总是干扰着自己进度的前行。但是在老师的帮助和指导提的意见和详细的讲解，这些问题也逐个的被解决被攻克，自己心头里的石头也就悄然无声的落地了。十分感谢我们学校的老师对自己的帮助和提供的建议使得自己在面对这些疑难和问题面前充满着信心。

在整个的过程中，自己有过好多次失败，就是在很小的细节里没有注意使得最后的结果不得人意。但是在我们学校老师的耐心解决一点一点的解决了问题使得自己的进程也逐步顺利的进行，也使得后续我自己在设计和编写的流程中更加的得心应手，所有很感谢老师无私的教诲和帮助，就像蜡烛一样点燃自己照亮别人。

回首整个大学生涯，自己过的还算是充实的，参加过学生会社联认识了精气神和干劲十足的学长学姐使得自己的生活变的更加的充实和美好也学会运用一些软件的技能。对自己的提升还是十分的明显也十分感谢他们的帮助和包容，这使得自己变得更好，向好的方向继续前进。

最后十分感谢在这四年的岁月中，老师们，同学们，朋友们对自己的帮助也希望他们在以后的道路上一路高歌猛进。

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.