项目说明文档

数据结构课程设计

——八种排序排序算法的比较案例

作 者 姓 名： 陈君

学 号： 2250420

指 导 教 师： 张颖

学院、 专业： 软件学院 软件工程

同济大学

Tongji Universit

# 1 分析

## 1.1 背景分析

假设您是一个软件工程师，负责开发一个需要对大量数据进行排序的应用程序。这个应用程序需要高效地排序数据，以确保用户获得快速的响应时间。然而，您可能会面临到选择合适的排序算法的挑战，因为不同的算法在不同情况下表现不同。在这种情况下，您需要比较八种排序算法，以确定哪种算法最适合您的应用程序。

## 1.2 功能分析

本项目要求能够通过随机函数产生10000个随机数，用快速排序，直接插入排序，冒泡排序，选择排序的排序方法排序，并统计每种排序所花费的排序时间和交换次数。其中，随机数的个数由用户定义，系统产生随机书。并且显示他们的比较次数。

# 2 设计

## 2.1 数据结构设计

如上功能分析所述，在这个比较案例中，需要一个数据结构来存储10000个随机数，同时要能够十分方便的对这些数进行交换，所以使用线性表这个数据结构是最合适的，有如下的优点：1、通过数组的下标可以直接确定随机数的位置。2、这种数据结构所占用的内存比较小。3、方便对数进行交换，时间复杂度是常数1。

## 2.2 类结构设计

设计一个List类存放10000个随机数，然后里面可以调用8种排序方法，对类里面的随机数进行排序

## 2.3 成员与操作设计

List类（List）

class List

{

public:

List();//默认析构函数

List(int n);//析构函数

void Bouble\_sort();//冒泡排序

void Selection\_sort();//选择排序

void Insertion\_sort();//直接插入排序

void Shell\_sort();//希尔排序

void Insertion\_sort(int i, int j);

void QKsort();//调用递归函数的函数

void Merge\_sort();

void QKsort(int start, int end);

void Heap\_sort();//堆排序

void Merge\_sort(int start, int end);//归并排序

void Radix\_sort();//基数排序

void sort\_two\_list(int start1, int start2, int end1, int end2);//对两个已经排好序的数列进行合并

int maxbit(int n);

void put();

void ajust\_Max(int i);//用于构造最大堆

int get\_list\_size() { return list.size(); }

int get\_time() { return time; }

int get\_num() { return num; }

void puttemp();

void reset\_num() { num = 0; }

private:

void swap(int a, int b);

vector<int> list;

vector<int> temp;

double time;//记录运行时间

int num;//记录交换时间

};

## 2.4 系统设计

主程序先调用initialzeio函数初始化io界面然后再声明一个List对象产生10000个随机数数据，然后使用一个switch语句，跳转到八种算法和一个退出的选项，方便用户进行比较。

# 3 实现

### 3.1 List::List()//默认析构函数

{

list = { 0 };

num = 0;

time = 0;

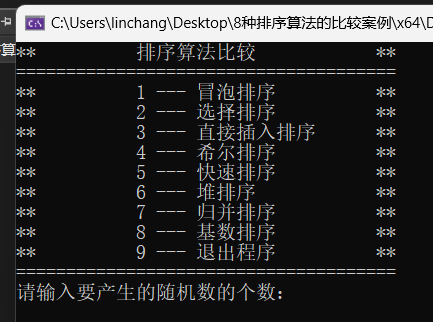
## }

默认的析构函数。

## 3.2 void setup ()//初始化io界面

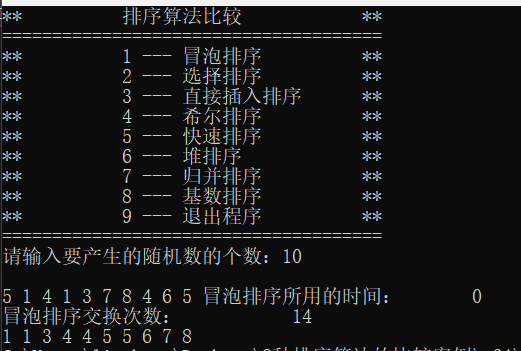
Cout输出题目中展示的画面

## 3.2.1 void setup ()功能截屏示例



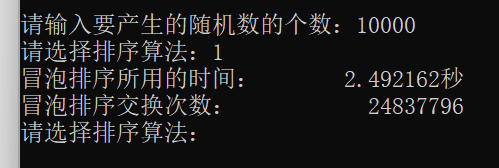
## 3.3 List(int n);//构造函数

具体实现：使用一个for循环，做n次，每次将随机数pushback进类里面的vector容器。同时设置交换次数和时间为0；



## 3.4 void List::Bouble\_sort()//冒泡排序功能

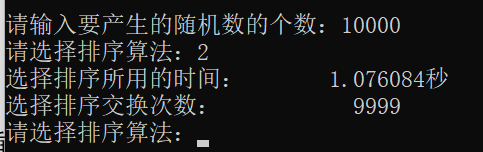
具体实现：声明一个temp来存储list，两个嵌套的for循环，第一层从0开始到list的大小-1为止，第二层从list.size()-1开始到i+1结束，如果list[j] < list[j+1]，就把这两个位置上的值进行交换。这样一次第二层的for循环下来，list的第一个元素就是最小的了



## 3.5void List::Selection\_sort()//选择排序

具体实现：声明一个temp来存储list，两个嵌套的for循环，第一层从0开始到list的大小-1为止，第二层从i开始到list.size()结束，每次搜索从i开始的最小的数值，找到后将这个位置的数值和i位置的数据进行交换

## 3.5.1截图展示



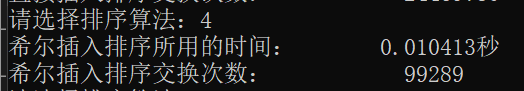
## 3.6 void List::Insertion\_Sort()//直接插入排序

具体实现：两层for循环，i从第二个数开始一直到数组的最后一个，每次拿着i位置的数和i前面的数从第一个进行比较，如果i位置的数小于j位置的数，就将i位置的数插入到j位置，j位置后的数顺势往右移动一位。

## 3.6.1截图展示

## 3.7void List::Shell\_sort()//希尔排序

具体实现：记录按下标的一定增量分组，对每组使用直接插入排序算法排序；随着增量逐渐减少，每组包含的关键词越来越多，当增量减至1时，整个文件恰被分成一组，算法便终止。分组大小从1/2\*数组大小开始，到1结束，每个组内进行插入排序。

截图展示：

## 3.8 void List::QKsort(int start,int end)//快速排序

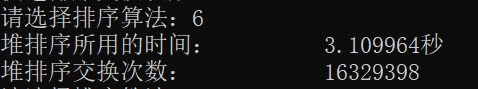
具体实现：在待排序的数列中，我们首先要找一个数字作为基准数（这只是个专用名词）。为了方便，我们一般选择第 1 个数字作为基准数（其实选择第几个并没有关系）。接下来我们需要把这个待排序的数列中小于基准数的元素移动到待排序的数列的左边，把大于基准数的元素移动到待排序的数列的右边。这时，左右两个分区的元素就相对有序了；接着把两个分区的元素分别按照上面两种方法继续对每个分区找出基准数，然后移动，直到各个分区只有一个数时为止。

3.8.1截图实例：



## 3.8void Heap\_sort();//堆排序

具体实现：对于给定的数列，进行一次最大堆的构造，可以将这个堆的最大值调整到数列的第一个位置，然后将最大值和最后一个位置交换，就完成了对最大值的排序，然后对前n-1个数在构造最大堆，可以完成对第二大的数的排序，以此类推，直到带构造的数为1为止。

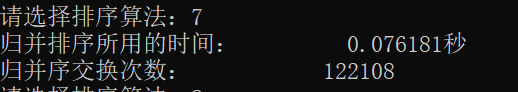


## 3.9void Merge\_sort();//归并排序

总体思想：先将数组从上向下逐渐分解若干组直到组内元素为1停止，然后将这些小的组从下向上逐渐排序成升序的数组

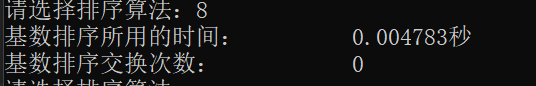
具体实现：递归调用void List::Merge\_sort(int start,int end)，这个函数内调Merge\_sort(start, (start + end) / 2 - 1);Merge\_sort((start + end) / 2, end);来获取排好序的两个相对小的组，然后再调用sort\_two\_list(start, (start + end) / 2 - 1, (start + end) / 2, end);将这两个小的组排序成升序的一个大的组。sort\_two\_list(start, (start + end) / 2 - 1, (start + end) / 2, end);实现方法类似于两个多项式的相加的算法。

截图展示



## 3.9void Radix\_sort()//基数排序

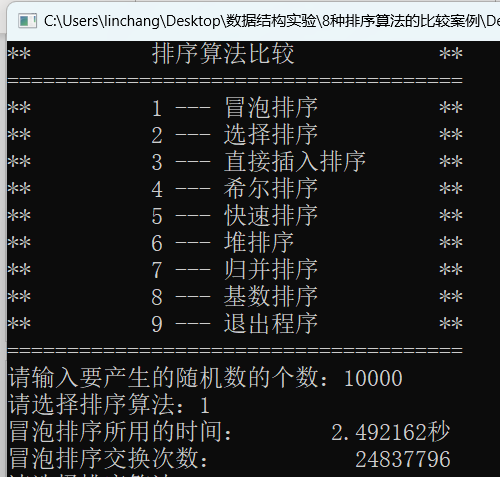
具体实现：设d为数组中最大数的位数，进行d次排序，从各位一直到d位，每次对特定的一个位进行升序排序，进行d次后就可以完成排序，每一位的升序排序可以用桶排序，用十个桶

截图展示

## 3.6 总体系统的实现

主程序首先调用setup函数，生成初始化界面，提示用户输入数字，然后用一个switch语句负责跳转到哪一个排序算法。随后按照上述的各种排序算法进行排序，最后输出运行时间和交换次数，当用户输入9时，exit（1）退出程序。

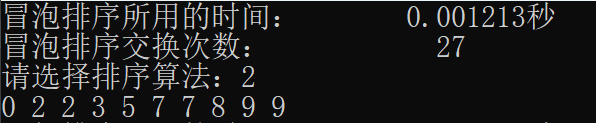
### }3.6.1 总体系统截屏示例



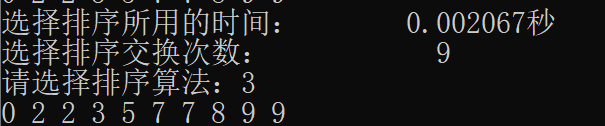
# 4 测试

## 4.1 系统总体功能测试功能测试

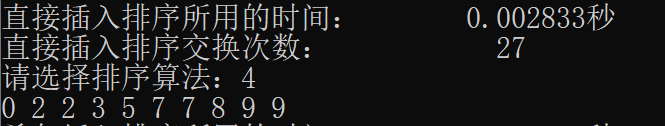
算法一的功能测试



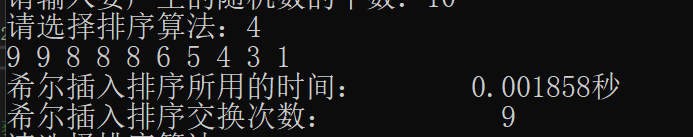
算法二的功能测试



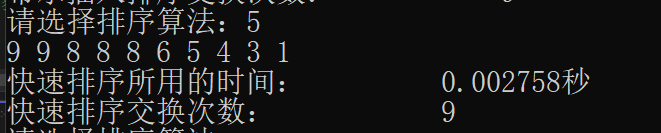
算法三的功能测试



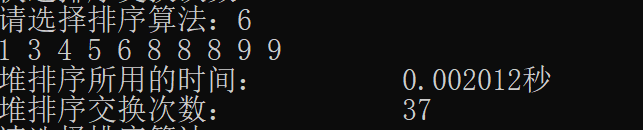
算法四的功能测试



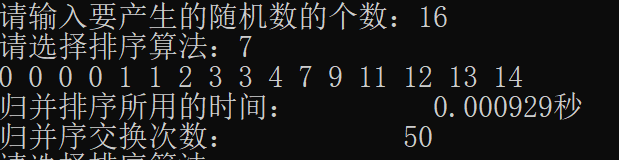
算法五的功能测试



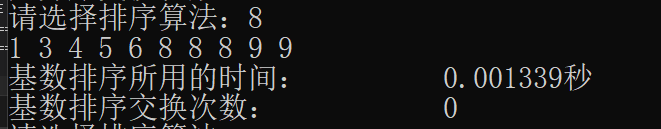
算法六的功能测试



算法七的功能测试



算法八的功能测试



4.2 边界条件测试

输入0。

4.2.1截屏展示

