**一、实验题目**

排序算法

**二、实验目的：**

1、掌握起泡排序、快速排序、希尔排序、堆排序、基数排序算法以及程序实现。

2、比较起泡排序、快速排序、希尔排序、堆排序、基数排序算法运行时间。

**三、实验设备与环境**

微型计算机、macOS 系列操作系统 、Pycharm系列软件

**四、实验内容**

随机产生10000个数据，进行起泡排序、快速排序、希尔排序、堆排序、基数排序，并比较各种算法的排序时间。

**五、概要设计（思路、算法、步骤等）**

**思路：**

本次实验要求随机产生 10000 个数据，进行冒泡排序、快速排序、希尔排序、堆排序、基数排序，并比较各种算法的排序时间。按照题目要求编写好这五个排序方法的函数，记录开始时间和结束时间，用结束时间减去开始时间即为排序时间。

**算法：**

**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。这个算法的名字由来是因为越小的元素会经由交换慢慢“浮”到数列的顶端。

算法描述：

1. 比较相邻的元素。如果第一个比第二个大，就交换它们两个；
2. 对每一对相邻元素作同样的工作，从开始第一对到结尾的最后一对，这样在最后的元素应该会是最大的数；
3. 针对所有的元素重复以上的步骤，除了最后一个；
4. 重复步骤1~3，直到排序完成

**快速排序（Quick Sort）**

快速排序的基本思想是通过一趟排序将待排记录分隔成独立的两部分，其中一部分记录的关键字均比另一部分的关键字小，则可分别对这两部分记录继续进行排序，以达到整个序列有序。

算法描述：

快速排序使用分治法来把一个串（list）分为两个子串（sub-lists）。具体算法描述如下：

从数列中挑出一个元素，称为 “基准”（pivot）；

重新排序数列，所有元素比基准值小的摆放在基准前面，所有元素比基准值大的摆在基准的后面（相同的数可以到任一边）。在这个分区退出之后，该基准就处于数列的中间位置。这个称为分区（partition）操作；

递归地把小于基准值元素的子数列和大于基准值元素的子数列排序。

**希尔排序（Shell Sort）**

希尔排序是一种插入排序的改进版本，也被称为缩小增量排序。它通过将待排序的元素分组，对每个分组进行插入排序，逐步减小分组的大小，最终完成整个序列的排序。希尔排序的基本思想是通过比较距离较远的元素进行交换，从而快速将较小的元素移动到序列的前部分。

算法描述：

1. 首先，选择一个增量序列，通常是一系列递减的整数。
2. 按照增量序列对待排序的元素进行分组。将相隔增量距离的元素放在同一组中。
3. 对每个分组内的元素进行插入排序，以使分组内的元素逐渐有序。
4. 不断缩小增量，重复步骤2和步骤3，直到增量减至1。
5. 最后，使用增量为1的一次插入排序，将整个序列排序。

**堆排序（Heap Sort）**

堆排序（Heap sort）是指利用堆这种数据结构所设计的一种排序算法。堆积是一个近似完全二叉树的结构，并同时满足堆积的性质：即子结点的键值或索引总是小于（或者大于）它的父节点。

算法描述：

将初始待排序关键字序列(R1,R2….Rn)构建成大顶堆，此堆为初始的无序区；

将堆顶元素R[1]与最后一个元素R[n]交换，此时得到新的无序区(R1,R2,……Rn-1)和新的有序区(Rn),且满足R[1,2…n-1]<=R[n]；

由于交换后新的堆顶R[1]可能违反堆的性质，因此需要对当前无序区(R1,R2,……Rn-1)调整为新堆，然后再次将R[1]与无序区最后一个元素交换，得到新的无序区(R1,R2….Rn-2)和新的有序区(Rn-1,Rn)。不断重复此过程直到有序区的元素个数为n-1，则整个排序过程完成。

**基数排序（Radix Sort）**

基数排序是按照低位先排序，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。

算法描述：

取得数组中的最大数，并取得位数；

nums为原始数组，从最低位开始取每个位组成radix数组；

对radix进行计数排序（利用计数排序适用于小范围数的特点）；

**六、详细设计（核心代码、算法流程图等）**

UI界面：

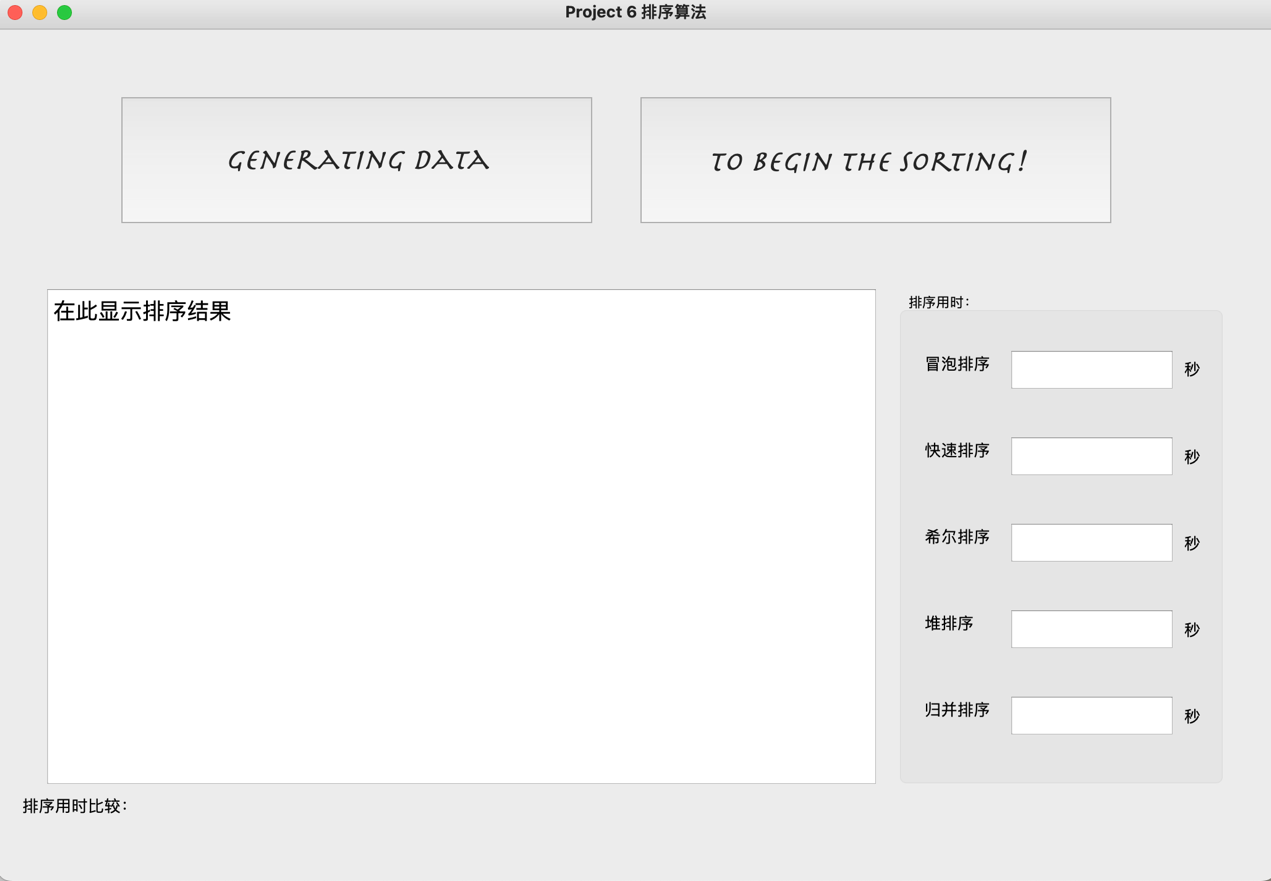
*# Form implementation generated from reading ui file 'sorting.ui'  
#  
# Created by: PyQt6 UI code generator 6.1.0  
#  
# WARNING: Any manual changes made to this file will be lost when pyuic6 is  
# run again. Do not edit this file unless you know what you are doing.*from PyQt6 import QtCore, QtGui, QtWidgets  
  
  
class Ui\_Form(object):  
 def setupUi(self, Form):  
 Form.setObjectName("Form")  
 Form.resize(1034, 689)  
 self.verticalLayoutWidget = QtWidgets.QWidget(Form)  
 self.verticalLayoutWidget.setGeometry(QtCore.QRect(40, 210, 671, 401))  
 self.verticalLayoutWidget.setObjectName("verticalLayoutWidget")  
 self.verticalLayout = QtWidgets.QVBoxLayout(self.verticalLayoutWidget)  
 self.verticalLayout.setContentsMargins(0, 0, 0, 0)  
 self.verticalLayout.setObjectName("verticalLayout")  
 self.textEdit = QtWidgets.QTextEdit(self.verticalLayoutWidget)  
 font = QtGui.QFont()  
 font.setFamily("Arial")  
 font.setPointSize(18)  
 font.setItalic(False)  
 self.textEdit.setFont(font)  
 self.textEdit.setObjectName("textEdit")  
 self.verticalLayout.addWidget(self.textEdit)  
 self.beginsort = QtWidgets.QPushButton(Form)  
 self.beginsort.setGeometry(QtCore.QRect(520, 50, 381, 111))  
 font = QtGui.QFont()  
 font.setFamily("Herculanum")  
 font.setPointSize(24)  
 font.setItalic(True)  
 self.beginsort.setFont(font)  
 self.beginsort.setObjectName("beginsort")  
 self.groupBox = QtWidgets.QGroupBox(Form)  
 self.groupBox.setGeometry(QtCore.QRect(730, 210, 261, 401))  
 self.groupBox.setObjectName("groupBox")  
 self.bubble = QtWidgets.QLabel(self.groupBox)  
 self.bubble.setGeometry(QtCore.QRect(20, 50, 61, 20))  
 self.bubble.setObjectName("bubble")  
 self.quick = QtWidgets.QLabel(self.groupBox)  
 self.quick.setGeometry(QtCore.QRect(20, 120, 61, 20))  
 self.quick.setObjectName("quick")  
 self.shell = QtWidgets.QLabel(self.groupBox)  
 self.shell.setGeometry(QtCore.QRect(20, 190, 61, 20))  
 self.shell.setObjectName("shell")  
 self.heap = QtWidgets.QLabel(self.groupBox)  
 self.heap.setGeometry(QtCore.QRect(20, 260, 61, 20))  
 self.heap.setObjectName("heap")  
 self.radix = QtWidgets.QLabel(self.groupBox)  
 self.radix.setGeometry(QtCore.QRect(20, 330, 61, 20))  
 self.radix.setObjectName("radix")  
 self.bubblesort = QtWidgets.QTextEdit(self.groupBox)  
 self.bubblesort.setGeometry(QtCore.QRect(90, 50, 131, 31))  
 self.bubblesort.setObjectName("bubblesort")  
 self.quicksort = QtWidgets.QTextEdit(self.groupBox)  
 self.quicksort.setGeometry(QtCore.QRect(90, 120, 131, 31))  
 self.quicksort.setObjectName("quicksort")  
 self.shellsort = QtWidgets.QTextEdit(self.groupBox)  
 self.shellsort.setGeometry(QtCore.QRect(90, 190, 131, 31))  
 self.shellsort.setObjectName("shellsort")  
 self.heapsort = QtWidgets.QTextEdit(self.groupBox)  
 self.heapsort.setGeometry(QtCore.QRect(90, 260, 131, 31))  
 self.heapsort.setObjectName("heapsort")  
 self.radixsort = QtWidgets.QTextEdit(self.groupBox)  
 self.radixsort.setGeometry(QtCore.QRect(90, 330, 131, 31))  
 self.radixsort.setObjectName("radixsort")  
 self.bubble\_2 = QtWidgets.QLabel(self.groupBox)  
 self.bubble\_2.setGeometry(QtCore.QRect(230, 49, 20, 31))  
 self.bubble\_2.setObjectName("bubble\_2")  
 self.bubble\_3 = QtWidgets.QLabel(self.groupBox)  
 self.bubble\_3.setGeometry(QtCore.QRect(230, 120, 20, 31))  
 self.bubble\_3.setObjectName("bubble\_3")  
 self.bubble\_4 = QtWidgets.QLabel(self.groupBox)  
 self.bubble\_4.setGeometry(QtCore.QRect(230, 190, 20, 31))  
 self.bubble\_4.setObjectName("bubble\_4")  
 self.bubble\_5 = QtWidgets.QLabel(self.groupBox)  
 self.bubble\_5.setGeometry(QtCore.QRect(230, 260, 20, 31))  
 self.bubble\_5.setObjectName("bubble\_5")  
 self.bubble\_6 = QtWidgets.QLabel(self.groupBox)  
 self.bubble\_6.setGeometry(QtCore.QRect(230, 330, 20, 31))  
 self.bubble\_6.setObjectName("bubble\_6")  
 self.label = QtWidgets.QLabel(Form)  
 self.label.setGeometry(QtCore.QRect(20, 620, 141, 16))  
 self.label.setObjectName("label")  
 self.comparetime = QtWidgets.QLabel(Form)  
 self.comparetime.setGeometry(QtCore.QRect(60, 640, 731, 21))  
 font = QtGui.QFont()  
 font.setPointSize(18)  
 self.comparetime.setFont(font)  
 self.comparetime.setText("")  
 self.comparetime.setObjectName("comparetime")  
 self.generatedata = QtWidgets.QPushButton(Form)  
 self.generatedata.setGeometry(QtCore.QRect(100, 50, 381, 111))  
 font = QtGui.QFont()  
 font.setFamily("Herculanum")  
 font.setPointSize(24)  
 font.setItalic(True)  
 self.generatedata.setFont(font)  
 self.generatedata.setObjectName("generatedata")  
  
 self.retranslateUi(Form)  
 QtCore.QMetaObject.connectSlotsByName(Form)  
  
 def retranslateUi(self, Form):  
 \_translate = QtCore.QCoreApplication.translate  
 Form.setWindowTitle(\_translate("From", "Project 6 排序算法"))  
 self.textEdit.setHtml(\_translate("Form", "<!DOCTYPE HTML PUBLIC \"-//W3C//DTD HTML 4.0//EN\" \"http://www.w3.org/TR/REC-html40/strict.dtd\">\n"  
"<html><head><meta name=\"qrichtext\" content=\"1\" /><style type=\"text/css\">\n"  
"p, li { white-space: pre-wrap; }\n"  
"</style></head><body style=\" font-family:\'Arial\'; font-size:18pt; font-weight:400; font-style:normal;\">\n"  
"<p style=\" margin-top:0px; margin-bottom:0px; margin-left:0px; margin-right:0px; -qt-block-indent:0; text-indent:0px;\">在此显示排序结果</p></body></html>"))  
 self.beginsort.setText(\_translate("Form", "To begin the sorting！"))  
 self.groupBox.setTitle(\_translate("Form", "排序用时："))  
 self.bubble.setText(\_translate("Form", "冒泡排序"))  
 self.quick.setText(\_translate("Form", "快速排序"))  
 self.shell.setText(\_translate("Form", "希尔排序"))  
 self.heap.setText(\_translate("Form", "堆排序"))  
 self.radix.setText(\_translate("Form", "归并排序"))  
 self.bubble\_2.setText(\_translate("Form", "秒"))  
 self.bubble\_3.setText(\_translate("Form", "秒"))  
 self.bubble\_4.setText(\_translate("Form", "秒"))  
 self.bubble\_5.setText(\_translate("Form", "秒"))  
 self.bubble\_6.setText(\_translate("Form", "秒"))  
 self.label.setText(\_translate("Form", "排序用时比较："))  
 self.generatedata.setText(\_translate("Form", "Generating data"))  
  
  
if \_\_name\_\_ == "\_\_main\_\_":  
 import sys  
 app = QtWidgets.QApplication(sys.argv)  
 Form = QtWidgets.QWidget()  
 ui = Ui\_Form()  
 ui.setupUi(Form)  
 Form.show()  
 sys.exit(app.exec())

主要函数：

import random  
import time  
import copy  
import sys  
import sys  
  
  
from PyQt6.QtCore import pyqtSignal, Qt  
from PyQt6.QtGui import QImage, QPixmap  
from PyQt6.QtWidgets import QMainWindow, QFileDialog, QMessageBox, QApplication, QGraphicsScene, QGraphicsPixmapItem  
from PyQt6.QtWidgets import QPushButton  
  
from sorting import Ui\_Form as u  
sys.setrecursionlimit(20000)*# 解除递归使用次数限制  
  
# 冒泡排序*def bubble\_sort(nums):  
 lens = len(nums)  
 for i in range(lens - 1):  
 for j in range(lens - 1 - i):  
 if nums[j] > nums[j+1]:  
 nums[j], nums[j + 1] = nums[j + 1], nums[j]  
  
*# 快速排序*def quick\_sort(nums):  
 lens = len(nums)  
 if len(nums) <= 1:  
 return nums  
 p = nums[lens // 2]  
 l = [x for x in nums if x < p]  
 m = [x for x in nums if x == p]  
 r = [x for x in nums if x > p]  
 return quick\_sort(l) + m + quick\_sort(r)  
  
*# 希尔排序*def shell\_sort(nums):  
 lens = len(nums)  
 gap = lens // 2  
 while gap > 0:  
 for i in range(gap, lens):  
 t = nums[i]  
 j = i  
 while j >= gap and nums[j - gap] > t:  
 nums[j] = nums[j - gap]  
 j -= gap  
 nums[j] = t  
 gap //= 2  
  
*# 堆排序  
## 堆*def heap(nums, n, i):  
 largest = i  
 l = 2 \* i + 1 *# 左子树* r = 2 \* i + 2 *# 右子树* if l < n and nums[l] > nums[largest]: *# 比较根结点和左子树* largest = l  
 if r < n and nums[r] > nums[largest]:  
 largest = r  
 if largest != i:  
 nums[i], nums[largest] = nums[largest], nums[i]  
 heap(nums, n, largest)  
*## 堆排序*def heap\_sort(nums):  
 n = len(nums)  
 *# 建立最大堆* for i in range(n // 2 - 1, -1, -1): *# 用于从最后一个非叶子节点开始，逐个向上调整子树，以构建最大堆。* heap(nums, n, i)  
 *# 逐个提取最大值，并进行堆调整* for i in range(n - 1, 0, -1):  
 nums[0], nums[i] = nums[i], nums[0] *# 将当前最大值移到数组堆最后面* heap(nums, i, 0) *# 调整堆  
  
# 基数排序*def counting\_sort(nums, e):  
 lens = len(nums)  
 output = [0] \* lens *# 输出数组* cnt = [0] \* 10 *#计数数组* for i in range(lens):  
 index = nums[i] // e  
 cnt[index % 10] += 1 *# 统计每个数字出现的次数* for i in range(1, 10):  
 cnt[i] += cnt[i - 1] *# 将计数数组转换为每个数字在输出数组中的位置  
  
 # 构建排序后的输出数组* i = lens - 1  
 while i >= 0:  
 index = nums[i] // e  
 output[cnt[index % 10] - 1] = nums[i]  
 cnt[index % 10] -= 1  
 i -= 1  
  
 for i in range(lens):  
 nums[i] = output[i] *# 将排序后的数组复制回原始数组*def radix\_sort(nums):  
 maxi = max(nums) *# 获取数组中的最大值* e = 1 *# 从个位开始* while maxi // e > 0:  
 counting\_sort(nums, e)  
 e \*= 10  
  
class MySort(QMainWindow, u):  
 signal = pyqtSignal(str)  
 def \_\_init\_\_(self, parent = None):  
 super(MySort, self).\_\_init\_\_(parent)  
 self.setupUi(self)  
 self.connecter()  
  
 def connecter(self): *#把函数与对应按钮相连接* self.generatedata.clicked.connect(self.generate)  
 self.beginsort.clicked.connect(self.sortall)  
  
 def generate(self):*# 生成排序数据* i = 0  
 self.nums = []  
 while (i < 10000):  
 self.nums.append(random.randint(0, 10000))  
 i = i + 1  
 strr = ', '.join([str(element) for element in self.nums]) *# 去掉[]* self.textEdit.setText(strr)  
 self.bubblesort.clear()  
 self.quicksort.clear()  
 self.shellsort.clear()  
 self.heapsort.clear()  
 self.radixsort.clear()  
 self.comparetime.clear()  
  
  
 def sortall(self):  
 times = {}  
 *# 冒泡排序* nums = self.nums  
 nums1 = copy.deepcopy(nums) *# 深拷贝* start = time.perf\_counter() *# 开始时间* bubble\_sort(nums1) *# 算法函数* end = time.perf\_counter() *# 结束时间* time1 = (end - start)  
 times['冒泡排序'] = time1  
 strr = ', '.join([str(element) for element in nums1]) *# 去掉[]* self.bubblesort.setText(str(time1)[:10])  
 self.textEdit.setText(str(strr))  
  
 *# 快速排序* nums2 = copy.deepcopy(nums)  
 start = time.perf\_counter() *# 开始时间* quick\_sort(nums2) *# 算法函数* end = time.perf\_counter() *# 结束时间* time2 = (end - start)  
 times['快速排序'] = time2  
 self.quicksort.setText(str(time2)[:10])  
  
 *# 希尔排序* nums3 = copy.deepcopy(nums)  
 start = time.perf\_counter() *# 开始时间* shell\_sort(nums3) *# 算法函数* end = time.perf\_counter() *# 结束时间* time3 = (end - start)  
 times['希尔排序'] = time3  
 self.shellsort.setText(str(time3)[:10])  
  
 *# 堆排序* nums4 = copy.deepcopy(nums)  
 start = time.perf\_counter() *# 开始时间* heap\_sort(nums4) *# 算法函数* end = time.perf\_counter() *# 结束时间* time4 = (end - start)  
 times['堆排序'] = time4  
 self.heapsort.setText(str(time4)[:10])  
  
 *# 基数排序* nums5 = copy.deepcopy(nums)  
 start = time.perf\_counter() *# 开始时间* radix\_sort(nums5) *# 算法函数* end = time.perf\_counter() *# 结束时间* time5 = (end - start)  
 times['基数排序'] = time5  
 self.radixsort.setText(str(time5)[:10])  
  
 *#比较用时* sortedtimes = sorted(times.keys(), key=lambda x: times[x], reverse=True)  
  
 *# 构建输出字符串* str1 = ""  
 for i, key in enumerate(sortedtimes):  
 str1 += key  
 if i < len(sortedtimes) - 1:  
 str1 += " > "  
  
 self.comparetime.setText(str1)  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 app = QApplication(sys.argv)  
 win = MySort()  
 win.show()  
 sys.exit(app.exec())

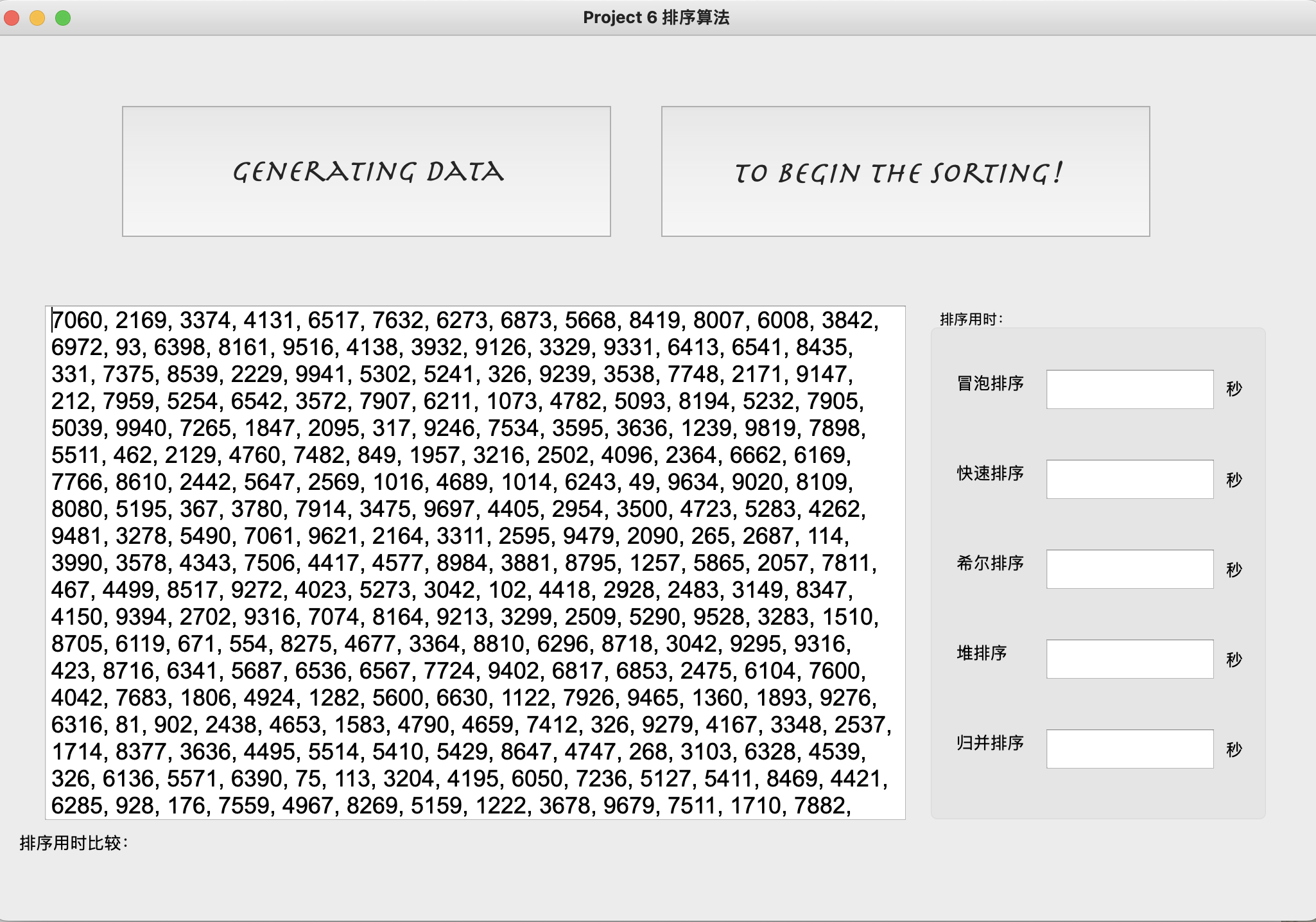
**七、测试结果及分析：**

UI界面：

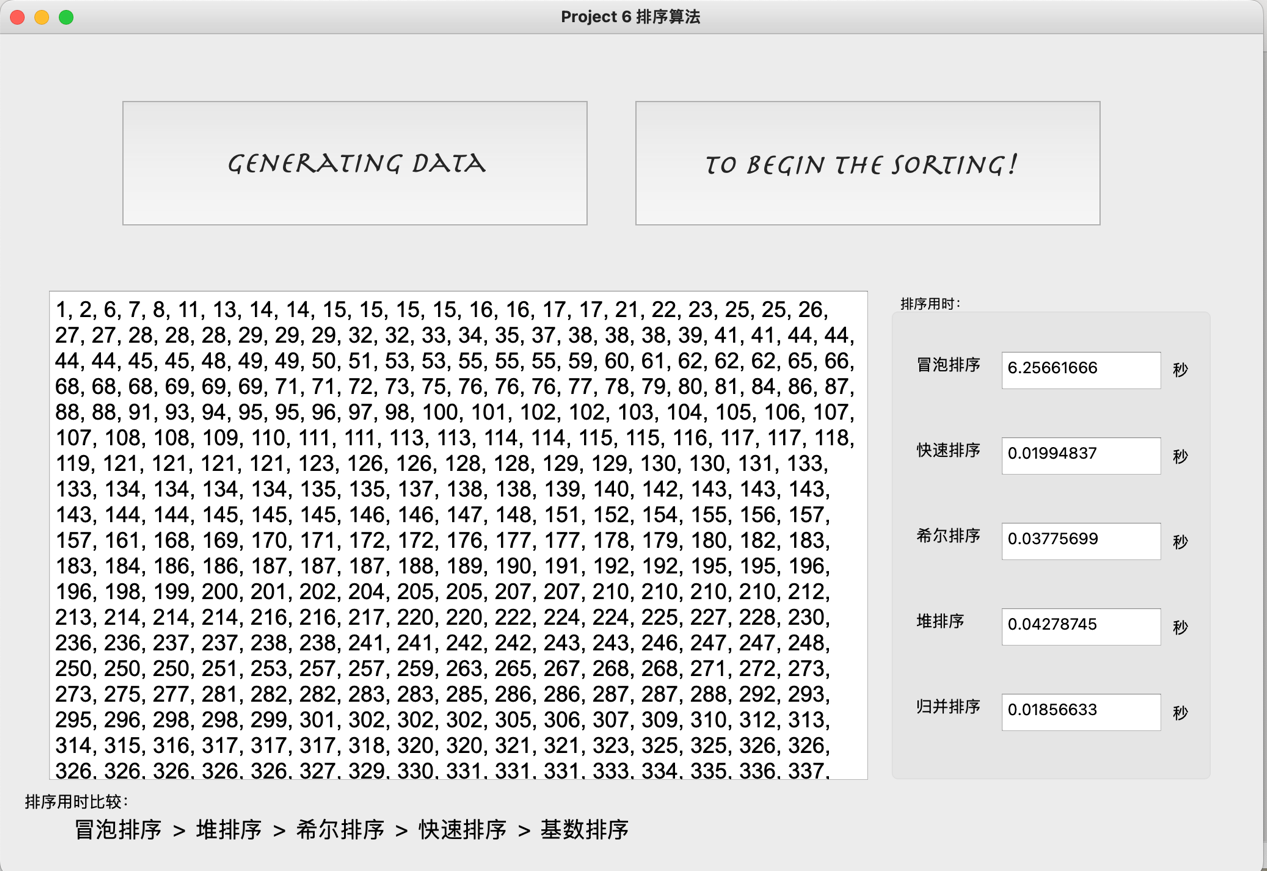


生成数据：

（显示窗口可以下拉）

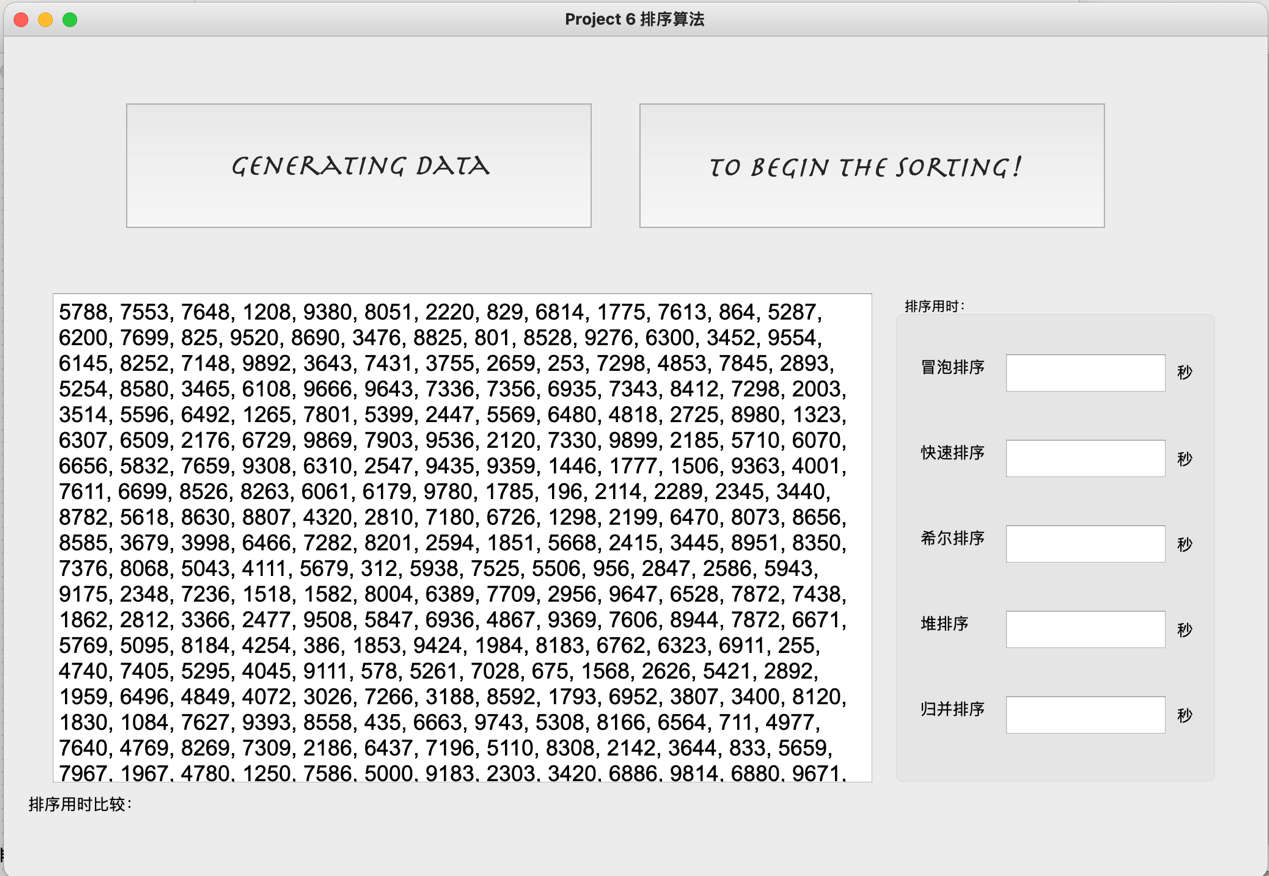


开始排序：



再次生成数据：

（显示值清空）



再次排序：



可见，冒泡排序所需时间最长，其次是堆排序，接着是希尔排序，然后是快速排序，运行速度最快的是基数排序。各种排序算法的时间复杂度如下表所示：



**八、总结**

通过本次实验，本次实验利用 Python 语言完成了冒泡排序、快速排序、希尔排序、堆排序、基数排序， 并对 10000 个数据利用上述排序方法进行排序，比较了几种排序方法的所消耗的时间，其结果见测试结果及分析。

五种排序方法运行时间从小到大分别为基数排序、快速排序、希尔排序、 堆排序、冒泡排序。源代码见 sort.py。

但是，需要注意的是，希尔排序和快速排序是不稳定的，希尔排序可能比快速排序快，也可能比其慢，还与其gap值选取有关。

本次实验将理论的排序算法加以代码实现，我不仅知道了五种排序算法所要花费的时间，还对这五种排序算法更加熟悉。

总的来说，本次实验给我带来了丰富的收获。我不仅更加深入地理解了排序算法，还进一步复习了PyQt工具的使用。我希望今后能够进一步利用这些知识和工具，进行更多有意义的实践。这次实验为我打开了新的可能性，让我在算法和可视化方面都有了进一步的提升。