

# 实验一

姓名：李思净

学号：1820201063

班级：07152002

CPU 核数	8
CPU 主频	3.40
内存	16 GB
Cache 大小	640 KB

## 一、实验目的和内容

### 一、实验目的

了解程序设计语言的发展历史，了解不同程序设计语言的各自特点；感受编译执行和解释执行两种不同的执行方式，初步体验语言对编译器设计的影响，为后续编译程序的设计和开发奠定良好的基础。

### 二、实验内容

分别使用 Java, C/C++, Python, Haskell, Assembly 五种不同的程序设计语言实现数组的快速排序，并输出不同语言之间排序所需要用到的时长，并比较不同语言之间的差异。

## 二、实现的具体过程和步骤

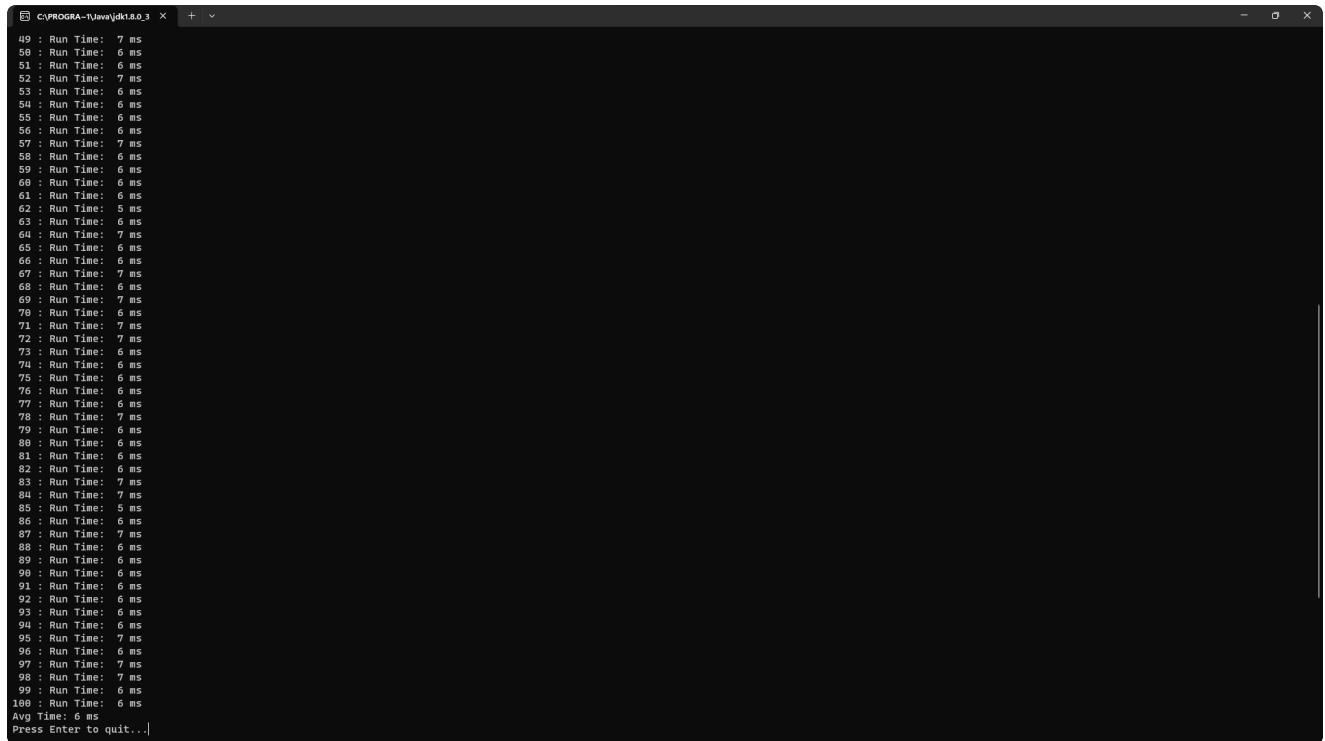
本次实验使用的快速排序算法是经典的基于分治思想的算法，具体实现的过程如下：

1. 定义一个 partition 函数，该函数将数组划分成两部分，小于枢轴的放在左边，大于枢轴的放在右边，并返回枢轴的索引。
2. 定义一个 quicksort 函数，该函数递归地调用 partition 函数，将数组不断划分为更小的子数组，并对每个子数组进行排序。

3. 以 100 次为循环，生成 100000 个不重复的乱序数组，并调用 quicksort 函数对其进行排序，并把排序所用的时间输出。
4. 最后输出 100 次排序所用的平均时常。

### 三、运行效果截图

Java



```
C:\PROGRAM-Files\Java\jdk1.8.0_91>
49 : Run Time: 7 ms
50 : Run Time: 6 ms
51 : Run Time: 6 ms
52 : Run Time: 7 ms
53 : Run Time: 6 ms
54 : Run Time: 6 ms
55 : Run Time: 6 ms
56 : Run Time: 6 ms
57 : Run Time: 7 ms
58 : Run Time: 6 ms
59 : Run Time: 6 ms
60 : Run Time: 6 ms
61 : Run Time: 6 ms
62 : Run Time: 5 ms
63 : Run Time: 6 ms
64 : Run Time: 7 ms
65 : Run Time: 6 ms
66 : Run Time: 6 ms
67 : Run Time: 7 ms
68 : Run Time: 6 ms
69 : Run Time: 7 ms
70 : Run Time: 6 ms
71 : Run Time: 7 ms
72 : Run Time: 7 ms
73 : Run Time: 6 ms
74 : Run Time: 6 ms
75 : Run Time: 6 ms
76 : Run Time: 6 ms
77 : Run Time: 6 ms
78 : Run Time: 7 ms
79 : Run Time: 6 ms
80 : Run Time: 6 ms
81 : Run Time: 6 ms
82 : Run Time: 6 ms
83 : Run Time: 7 ms
84 : Run Time: 7 ms
85 : Run Time: 5 ms
86 : Run Time: 6 ms
87 : Run Time: 7 ms
88 : Run Time: 6 ms
89 : Run Time: 6 ms
90 : Run Time: 6 ms
91 : Run Time: 6 ms
92 : Run Time: 6 ms
93 : Run Time: 6 ms
94 : Run Time: 6 ms
95 : Run Time: 7 ms
96 : Run Time: 6 ms
97 : Run Time: 7 ms
98 : Run Time: 7 ms
99 : Run Time: 6 ms
100 : Run Time: 6 ms
Avg Time: 6 ms
Press Enter to quit...
```

C/C++

```
Python Library\大三\编译原理与... x + -
49: Run Time: 9 ms
50: Run Time: 9 ms
51: Run Time: 9 ms
52: Run Time: 8 ms
53: Run Time: 9 ms
54: Run Time: 9 ms
55: Run Time: 9 ms
56: Run Time: 9 ms
57: Run Time: 9 ms
58: Run Time: 9 ms
59: Run Time: 9 ms
60: Run Time: 8 ms
61: Run Time: 8 ms
62: Run Time: 8 ms
63: Run Time: 8 ms
64: Run Time: 9 ms
65: Run Time: 9 ms
66: Run Time: 9 ms
67: Run Time: 9 ms
68: Run Time: 7 ms
69: Run Time: 9 ms
70: Run Time: 8 ms
71: Run Time: 8 ms
72: Run Time: 9 ms
73: Run Time: 8 ms
74: Run Time: 9 ms
75: Run Time: 9 ms
76: Run Time: 8 ms
77: Run Time: 8 ms
78: Run Time: 8 ms
79: Run Time: 8 ms
80: Run Time: 8 ms
81: Run Time: 8 ms
82: Run Time: 8 ms
83: Run Time: 8 ms
84: Run Time: 8 ms
85: Run Time: 8 ms
86: Run Time: 9 ms
87: Run Time: 8 ms
88: Run Time: 9 ms
89: Run Time: 8 ms
90: Run Time: 9 ms
91: Run Time: 9 ms
92: Run Time: 8 ms
93: Run Time: 9 ms
94: Run Time: 9 ms
95: Run Time: 9 ms
96: Run Time: 9 ms
97: Run Time: 9 ms
98: Run Time: 9 ms
99: Run Time: 9 ms
100: Run Time: 9 ms
Avg Time: 8.57 ms
Press any key to continue . . . |
```

## Python

```
Windows PowerShell x + -
49 : Run Time: 209.32 ms
50 : Run Time: 187.42 ms
51 : Run Time: 198.26 ms
52 : Run Time: 200.75 ms
53 : Run Time: 188.66 ms
54 : Run Time: 211.49 ms
55 : Run Time: 201.50 ms
56 : Run Time: 218.96 ms
57 : Run Time: 188.19 ms
58 : Run Time: 193.40 ms
59 : Run Time: 190.99 ms
60 : Run Time: 184.04 ms
61 : Run Time: 202.99 ms
62 : Run Time: 205.37 ms
63 : Run Time: 194.37 ms
64 : Run Time: 188.15 ms
65 : Run Time: 196.40 ms
66 : Run Time: 197.37 ms
67 : Run Time: 185.43 ms
68 : Run Time: 180.67 ms
69 : Run Time: 193.40 ms
70 : Run Time: 199.39 ms
71 : Run Time: 195.40 ms
72 : Run Time: 193.37 ms
73 : Run Time: 182.94 ms
74 : Run Time: 189.43 ms
75 : Run Time: 192.38 ms
76 : Run Time: 191.41 ms
77 : Run Time: 200.00 ms
78 : Run Time: 185.48 ms
79 : Run Time: 182.45 ms
80 : Run Time: 185.40 ms
81 : Run Time: 187.42 ms
82 : Run Time: 194.99 ms
83 : Run Time: 185.06 ms
84 : Run Time: 197.40 ms
85 : Run Time: 205.38 ms
86 : Run Time: 192.21 ms
87 : Run Time: 185.44 ms
88 : Run Time: 190.57 ms
89 : Run Time: 183.18 ms
90 : Run Time: 183.43 ms
91 : Run Time: 189.01 ms
92 : Run Time: 206.36 ms
93 : Run Time: 197.35 ms
94 : Run Time: 207.37 ms
95 : Run Time: 206.92 ms
96 : Run Time: 232.28 ms
97 : Run Time: 211.36 ms
98 : Run Time: 188.88 ms
99 : Run Time: 183.23 ms
100 : Run Time: 185.50 ms
Avg Time: 194.8012 ms
PS E:\Main_Library\大三\编译原理与设计\作业\07152002-1820201063-李思净-Lab1\Python> |
```

## Haskell

```
Windows PowerShell
58 : Run Time : 90.0ms
59 : Run Time : 60.0ms
60 : Run Time : 70.0ms
61 : Run Time : 80.0ms
62 : Run Time : 110.0ms
63 : Run Time : 100.0ms
64 : Run Time : 1510.0ms
65 : Run Time : 60.0ms
66 : Run Time : 70.0ms
67 : Run Time : 120.0ms
68 : Run Time : 40.0ms
69 : Run Time : 90.0ms
70 : Run Time : 130.0ms
71 : Run Time : 90.0ms
72 : Run Time : 70.0ms
73 : Run Time : 80.0ms
74 : Run Time : 60.0ms
75 : Run Time : 70.0ms
76 : Run Time : 90.0ms
77 : Run Time : 80.0ms
78 : Run Time : 60.0ms
79 : Run Time : 70.0ms
80 : Run Time : 70.0ms
81 : Run Time : 90.0ms
82 : Run Time : 70.0ms
83 : Run Time : 40.0ms
84 : Run Time : 90.0ms
85 : Run Time : 90.0ms
86 : Run Time : 50.0ms
87 : Run Time : 80.0ms
88 : Run Time : 70.0ms
89 : Run Time : 80.0ms
90 : Run Time : 60.0ms
91 : Run Time : 70.0ms
92 : Run Time : 60.0ms
93 : Run Time : 60.0ms
94 : Run Time : 90.0ms
95 : Run Time : 70.0ms
96 : Run Time : 80.0ms
97 : Run Time : 70.0ms
98 : Run Time : 90.0ms
99 : Run Time : 80.0ms
100 : Run Time : 70.0ms
```

## Assembly

```
Windows PowerShell
49: Run Time: 4 ms
50: Run Time: 4 ms
51: Run Time: 9 ms
52: Run Time: 3 ms
53: Run Time: 5 ms
54: Run Time: 3 ms
55: Run Time: 3 ms
56: Run Time: 4 ms
57: Run Time: 4 ms
58: Run Time: 4 ms
59: Run Time: 4 ms
60: Run Time: 5 ms
61: Run Time: 4 ms
62: Run Time: 3 ms
63: Run Time: 5 ms
64: Run Time: 3 ms
65: Run Time: 5 ms
66: Run Time: 5 ms
67: Run Time: 4 ms
68: Run Time: 5 ms
69: Run Time: 4 ms
70: Run Time: 4 ms
71: Run Time: 4 ms
72: Run Time: 4 ms
73: Run Time: 4 ms
74: Run Time: 5 ms
75: Run Time: 3 ms
76: Run Time: 5 ms
77: Run Time: 4 ms
78: Run Time: 4 ms
79: Run Time: 4 ms
80: Run Time: 4 ms
81: Run Time: 5 ms
82: Run Time: 4 ms
83: Run Time: 4 ms
84: Run Time: 5 ms
85: Run Time: 4 ms
86: Run Time: 4 ms
87: Run Time: 4 ms
88: Run Time: 4 ms
89: Run Time: 5 ms
90: Run Time: 3 ms
91: Run Time: 6 ms
92: Run Time: 4 ms
93: Run Time: 5 ms
94: Run Time: 4 ms
95: Run Time: 5 ms
96: Run Time: 4 ms
97: Run Time: 4 ms
98: Run Time: 5 ms
99: Run Time: 6 ms
100: Run Time: 4 ms
Avg Time: 4.46 ms
Press any key to continue . . .
```

## 四、语言易用性和程序规模对比分析

### 1. Java

易用性：相较于其它的程序，Java 是一种面向对象的语言，具有良好的可读性以及适合维护，更适合作为开发语言进行开发。

程序规模：因为 Java 语言需要进行定义类以及定义方法，并且需要更多的语句进行类型声明，因此规模相较大部分的语言来说规模较大。

2. C/C++

易用性：和 Java 比起来，拥有更加强大的功能，可以开发更多的应用、操作系统等等；但相对而言会较 Java 需要掌握更多的知识点。

程序规模：与 Java 语言类似，在编写代码的过程中需要更多的精力作为代码的测试以及调试。

3. Python

易用性：Python 相较于其它的编程语言，拥有更加简洁、直观的代码风格，拥有许多的内置函数以及内置库，编程的效率较高。

程序规模：Python 可以开发许许多多的应用，但程序规模相对较小，需要开发大型的应用的时候会受到性能的影响。

4. Haskell

易用性：语法和结构较为简单，但需要一定的数学和函数式编程基础。在 Haskell 中，可以使用高级类型系统和函数组合来编写更为清晰和模块化的代码。

程序规模：Haskell也可以用于开发大型和复杂的应用程序。Haskell的高级类型系统和纯函数式编程模式可以帮助开发者编写更安全、高效、可维护的代码，使得代码的规模和复杂度可以更好地控制。但是，Haskell在某些情况下的性能可能比其他语言低，需要进行更多的优化和调试。

5. Assembly

易用性：Assembly是一种底层编程语言，它与计算机硬件紧密相关。Assembly的代码通常是直接操作硬件和寄存器的机器指令，需要更高的技术水平和经验。虽然Assembly的效率很高，但在开发大型应用程序时，它的开发速度和易用性都相对较低。

程序规模：Assembly可以用于编写高效的底层代码，但是在处理大规模的数据和业务逻辑时，需要更多的时间和精力来进行代码的编写和调试。同时，由于需要直接操作硬件和寄存器，Assembly的代码可读性和可维护性也较低。

五、程序运行性能对比分析

语言	平均耗时(ms)
Java	6

C/C++	8.57
Python	194
Haskell	70
Assembly	4.46

从上面的结果可以看出，Java 和 C/C++ 以及 Assembly 在算法上的性能差不多，编译运行的方式也差不多；而 Python 在算法的性能上，与 Java, C/C++, 以及 Assembly 差别很大，运行了 194 毫秒，由此可以预见，当处理大数据的时候，需要用到诸如 Numpy 之类的库进行计算。

## 六、实验心得体会

通过这次的实验，我了解了不同语言之间的差异性、编译执行以及解释执行之间的差距。并在性能测试后，充分的了解了编程语言对程序开发的效率、运行的效率和程序规模的影响。

总的来说，本次实验让我对编程语言的选择有了更深入的了解，也让我体验到了不同语言在实现同一个算法时的差异和优缺点。