# 实验一 排序算法

李远航 PB20000137

### 一、实验内容

- 排序n个元素，元素为随机生成的0到$2^{15}-1$之间的整数，n的取值为：$2^3, 2^6, 2^9, 2^{12}, 2^{15}, 2^{18}$
- 实现以下算法：堆排序，快速排序， 归并排序，计数排序

### 二、实验要求

- 编程要求
  - C/C++，排序算法要自己实现，不能直接调用qsort()等解决
- 目录格式
  - 实验需建立根文件夹，文件夹名称为：`编号-姓名-学号-project1`，在根文件夹下需包括实验报告和 `eX1` 子文件夹。实验报告命名为 `编号-姓名-学号-project1.pdf`， `eX1` 子文件夹又包含3个子文件夹：
    - input文件夹：存放输入数据
    - src文件夹：源程序
    - output文件夹：输出数据
  - input：
    - 输入文件中每行一个随机数据，总行数大于等于$2^{15}$
    - 顺序读取n个数据，进行排序
    - Example：用快速排序对$2^9$个元素进行排序，其随机数据的输入文件路径为 `编号-姓 名-学号-project1/ex1/input/input.txt`，顺序读取前$2^9$个元素进行排序
- 实验报告
  - 必须包含实验内容、实验设备和环境、实验方法和步骤、实验结果与分析
  - 截图：
    - 四个排序算法$n=2^3$时排序结果的截图
    - 任一排序算法六个输入规模运行时间的截图
  - 根据不同输入规模时记录的数据，画出各算法在不同输入规模下的运行时间曲线图。比较你的曲线是否与课本中的算法渐进性能是否相同， 若否，为什么，给出分析
  - 比较不同的排序算法的时间曲线，分析在不同输入规模下哪个更占优势

### 三、实验设备和环境

- `g++ (Ubuntu 9.4.0-1ubuntu1~20.04.1) 9.4.0`
- `wsl2`
- `Intel Core i5-10200H @ 8x 2.4GHz`

### 四、实验方法和步骤

- **Heap Sort**
  - 首先写出从堆顶开始调整元素的算法
  - 排序的步骤为先利用调整算法将原数据调整成堆，再逐次交换堆顶元素和最后一个元素，重复调整堆的操作
- **Quick Sort**

    - 核心是选取基准数，根据大小比较，将数据分组，接着递归调用
- **Merge Sort**
    - 将排序划分成一个个子问题，递归进行求解
- **Counting Sort**
    - 核心是求出每个元素出现的次数，求出比它小的元素的个数，个数即为该元素应该在的位置，同时需要注意相同的元素的情况
- 随机数的生成
    - 使用 `random` 库
- 算法运行时间
    - 在程序运行前后使用 `clock_t`
- 实验结果分析
    - 使用 `matplotlib` 绘图
- 实验步骤已写成 `py` 脚本

```
python ex1.py
```

**五、具体实现**

见代码文件

**六、实验结果分析**

- $2^3$时输出截图

ex1 > input > input_3.txt

```
8072
13274
1125
23654
315
2105
8160
20579
```

ex1 > output > countingsort > result_3.txt

```
315 1125 2105 8072 8160 13274 20579 23654
```

ex1 > output > heapsort > result_3.txt

```
315 1125 2105 8072 8160 13274 20579 23654
```

ex1 > output > mergesort > result_3.txt

```
315 1125 2105 8072 8160 13274 20579 23654
```

ex1 > output > quicksort > result_3.txt

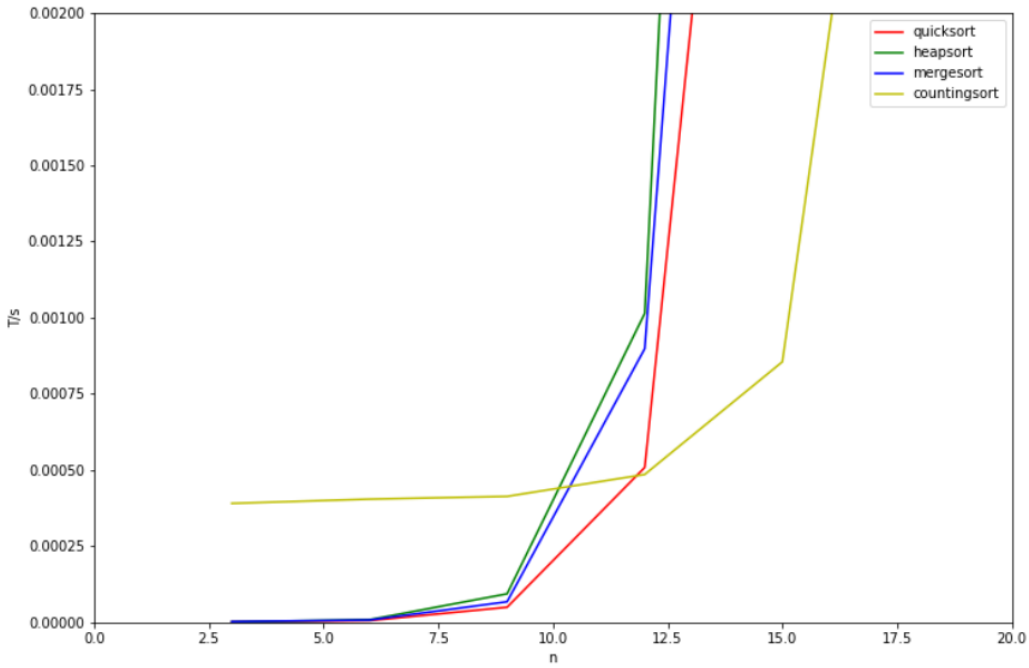
```
315 1125 2105 8072 8160 13274 20579 23654
```

- **Quick Sort**运行时间截图，依次为不同数据规模

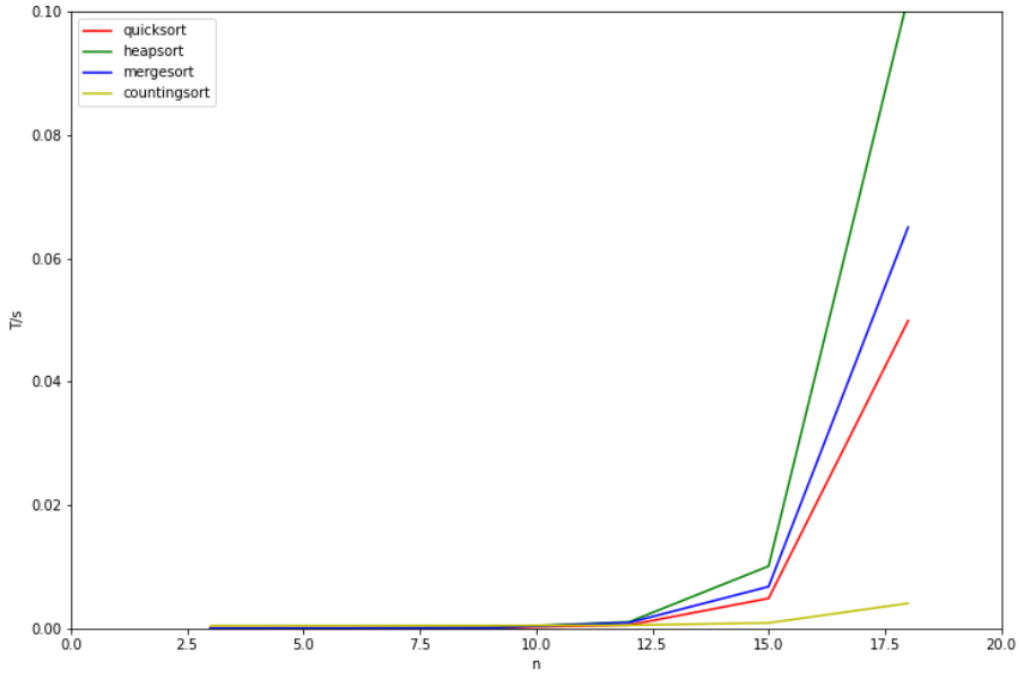
ex1 > output > mergesort > time.txt

```
0.000002s
0.000007s
0.000067s
0.000898s
0.006741s
0.065035s
```

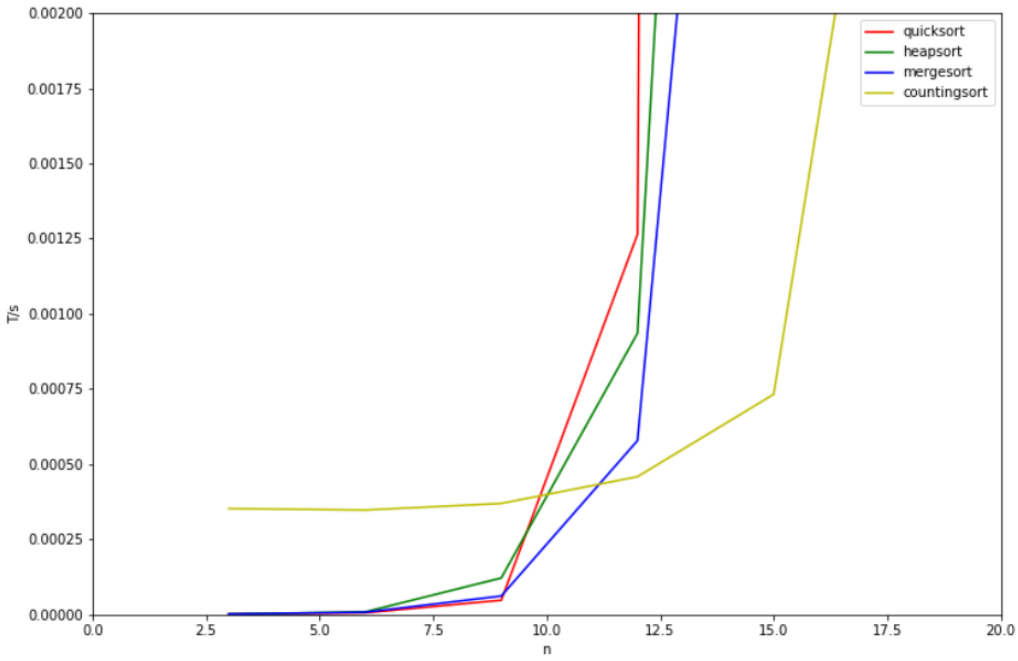
- 各算法在不同输入规模下的运行时间曲线图(其中$n$表示数据规模$2^n$)
  - 数据规模较小时

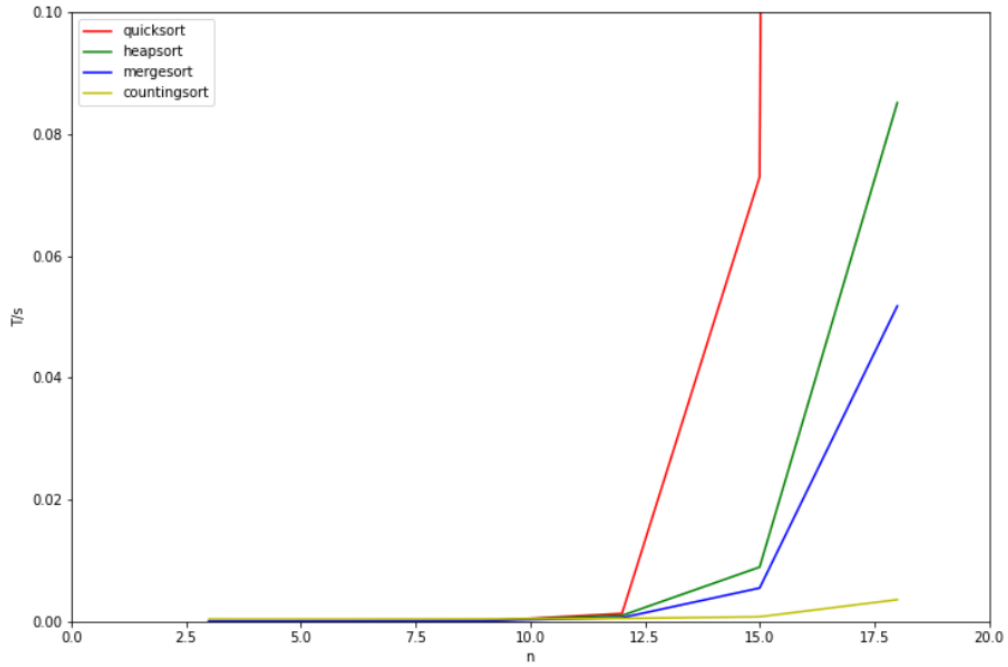



  - 数据规模较大时




  - 数据分布较密集时




- 实验结果分析
  - 运行时间图像和课本中的算法渐进性能基本相同
    - 快速排序、堆排序、归并排序的平均复杂度为$O(nlog(n))$，三者图像基本相同，而在运行时间上有细微的差别
    - 计数排序的时间复杂度为线性，在数据规模较小时，运行时间较长，但在数据规模大时，具有很好的效果
  - 实际应用
    - 综合考虑，已知数据规模较小时，应该使用归并排序
      - 在数据规模较小时，且数字分布跨度大时，使用快速排序算法较为优势
      - 如果数据规模较小，但数据分布紧密，快速排序运行时间较长，使用归并排序可以获得更好的性能
    - 当数据规模较大，使用计数排序较为优势

## 七、实验反思

- 算法在不同数据规模，不同数据分布下的性能不完全相同
- 图表可以直观地用来分析算法的性能