快速排序不同主元选择策略比较的设计思路和测试说明

December 12, 2021

1. 项目设计思路:

    1. QuickSort.h的实现:

        I QuickSort.h中实现了分别用三数中值分割法和随机分割法两种分割策略进行的快速排序。其中三数中值分割法的实现主要参考了书中的代码，当数组的大小小于等于10时，本程序采用了快速排序。对于随机分割法，程序在排序的数组范围内随机地选取主元。为了控制变量以便进行两种策略的比较，随机分割法在其他部分尽量与三数中值分割法保持一致。

    2. main.cpp的实现:

        I **数据的生成**：用户输入进行排序的元素个数N后，程序将自动生成大小为N的vector。由于本程序使用的具体的模板类型是Int,因此，为了方便验证排序的结果,生成的数据为[0,N]中的整数(可以重复)。

        II **排序效率比较的实现**：根据项目要求，我们首先对vector中的数据进行多次生成。然后，将每次生成的数据分别用三数中值分割法和随机分割法两种分割策略进行快速排序，并分别记录两种策略下排序所用的平均时间，从而进行比较。当然，为了验证排序的正确性，程序对第一次排序前的数据和两种主元选择策略排序后的数据分别进行了输出。

2. 测试说明:

    1. 测试输入:

        I 运行 make 命令，自动编译 main.cpp 并生成可执行文件 test;

        II 运行 bash run 命令，**在同一行后输入一个正整数N，表示进行排序的元素的个数**。

    2. 测试输出:

I 测试输出第一次排序生成的数据及其分别用三数中值分割法和随机分割法两种分割策略进行快速排序后的数组，和两种策略下多次排序所用的平均时间。具体在输出中都有详细的说明。

3．测试结论：

1．经验证，无论N取什么值，两种主元选择策略下输出的排序后的数组都是有序的，这说明**用三数中值分割法和随机分割法进行的快速排序算法都是正确的**。

2．对于两种主元选择策略排序所用的平均时间T，经多次测试得如下平均结果(部分值)：

N	T(三数中值)/ms	T(随机)/ms
100	0.003	0.004
1000	0.06	0.08
10000	0.63	0.64
100000	6.7	6.8
1000000	76	75

纵向比较，我们测试出发现当N较小时， T(N)的增长速率为Ω(NlogN)。**对此一种可能的解释是**，由于N小于等于10时程序采用的是快速排序，因此N较小时进行排序时，快速排序次数所占比例较大，T(N)的增长速率会趋向于O(N^2)。而在N较大的时候 T(N)的增长速率则符合O(NlogN)，这与快速排序效率的理论分析是一致的，也体现了解释的合理性。

横向比较，结果表明，对于相同的N，三数中值分割法和随机分割法进行快速排序的平均时间几乎是相同的。这与课本分析的结论并不能完全吻合。**对此一种可能的解释是**，随着编译器的不断升级，随机数生成的时间开销不再庞大；此外，三数中值分割法虽然不用进行随机数的生成，但是仍需要进行多次比较。因此两种方法的耗时是没有显著差异的，甚至从某种程度上可能随机分割法是更为安全的。