

实验一 插入排序与归并排序

May 2022

1 实验项目结构

- project
 - include
 - MySort.hpp
 - InsertionSort.hpp
 - MergeSort.hpp
 - data
 - sample
 - test
 - generate
 - performance
 - performance.cpp
 - main.cpp

project 文件夹包含了本实验的所有代码源程序以及测试数据。

include 文件夹中, MySort.hpp 头文件定义了 MySort 类, 以它为基类在 InsertionSort.hpp 和 MergeSort.hpp 中分别定义了两个子类: InsertionSort 和 MergeSort。你需要在本实验中完成这两个子类的算法实现。

data 文件夹中包含针对算法程序正确性测试以及性能测试所需的测试数据以及数据生成器。其中, sample 单独用于正确性的检测, 仅包含小规模测试集数据, test 用于性能测试, 包含部分大规模测试集数据。另外, generate 包含了测试数据的生成器。

performance 文件夹中包含了性能测试的代码。在完成正确性测试之后, 可以运行 performance.cpp 来观察两种算法的性能。

main.cpp 是正确性检测的主程序, 当你编写完 InsertionSort.hpp 或者 MergeSort.hpp 后, 可以编译运行 main.cpp, 选择你想要测试的算法。

2 实验内容

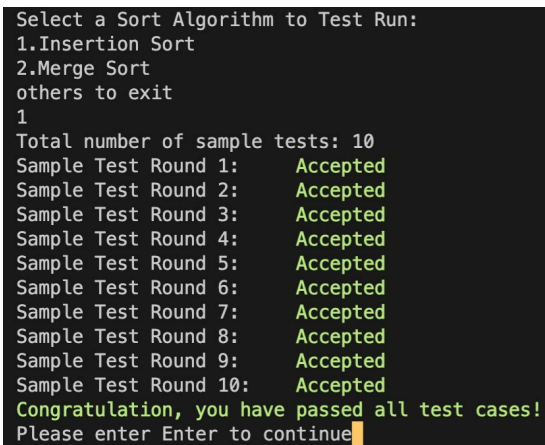
2.1 选择排序

根据插入排序的思想，完成 InsertionSort.hpp 中的代码。并通过编译运行 main.cpp 的正确性检测。

```
class InsertionSort: public MySort {
public:
    void mysort(std::vector<int>& nums) {
        // 请在这里完成你的代码
    }
};
```

具体来说，你需要对 nums 中的数字使用选择排序算法进行排序。mysort 并不需要返回任何内容，因为参数 nums 为引用传递，在函数内部对 nums 的修改会对调用该函数时传入的实参产生影响。

编写代码之后，你可以运行 main.cpp 进行测试，如果正确的话，会出现下图中的结果。



```
Select a Sort Algorithm to Test Run:
1.Insertion Sort
2.Merge Sort
others to exit
1
Total number of sample tests: 10
Sample Test Round 1: Accepted
Sample Test Round 2: Accepted
Sample Test Round 3: Accepted
Sample Test Round 4: Accepted
Sample Test Round 5: Accepted
Sample Test Round 6: Accepted
Sample Test Round 7: Accepted
Sample Test Round 8: Accepted
Sample Test Round 9: Accepted
Sample Test Round 10: Accepted
Congratulation, you have passed all test cases!
Please enter Enter to continue
```

2.2 归并排序

根据归并排序的思想，完成 MergeSort.hpp 中的代码。并通过运行 main.cpp 的正确性检测。

```
class MergeSort: public MySort {
public:
    void merge_sort_aux(std::vector<int>& nums, int left, int right) {
        //请在这里完成你的代码
    }
    void mysort(std::vector<int>& nums) {
        int n = nums.size();
        merge_sort_aux(nums, 0, n - 1);
    }
};
```

具体来说，你需要对 `nums` 中的数字使用归并排序算法进行排序。`mysort` 并不需要返回任何内容，因为参数 `nums` 为引用传递，在函数内部对 `nums` 的修改会对调用该函数时传入的实参产生影响。

`merge_sort_aux` 函数为归并排序辅助函数，你需要对 `nums` 数组闭区间 `[left, right]` 内的数值进行排序。

3 实验思考

1. 以第一组测试数据为例，分别观察选择排序和归并排序过程中数组元素的变化情况（需要逐步给出）。

- input

5

1 6 2 10 2

- output

1 2 2 6 10

2. 比较插入排序与归并排序的性能差距。
3. 【拓展题】尝试修改源代码，实现排序过程中「比较」操作的计数功能，并以该角度来对插入排序和归并排序进行对比。
4. 【拓展题】基于归并排序，求解给定数组的逆序对问题。
在数组中的两个数字，如果前面一个数字大于后面的数字，则这两个数字组成一个逆序对。
输入一个数组，求出这个数组中的逆序对的总数。

- input

4

7 5 6 4

- output

5

5. 【拓展题】基于归并排序中使用的分治思想，实现快速幂算法。

给定 a, b, p ，求解 $a^b \bmod p$ 。其中 $1 \leq a, b, p \leq 10^9$ 。

例如 $3^2 \bmod 5 = 4, 4^3 \bmod 9 = 1, 814817899^{637634896} \bmod 914938489 = 808091124$

提示：注意 `int` 乘法可能会溢出，需要使用 `long long`。