1 对 HeapSort 的阐述和分析

在 `heapSort` 函数中，我使用将向量转换为堆，然后使用 `std::make_heap` 逐个将堆顶元素（最大元素）移到当前范围的末尾，并调整剩余元素使其仍为堆。最终，令向量 vec 中的元素按升序排列。关于 check 函数：我希望 check 函数实现检查向量是否按升序排列的功能，所以我使用了一个 for 循环，检查 vec[i-1] 是否小于 vec[i]，如果不是则返回 false，否则返回 true。然后我设计了四个函数来生成随机、有序、逆序、部分重复四种序列。对于测试流程，我首先生成一个随机序列，然后调用 `heapSort` 函数对其进行排序，然后调用 check 函数检查排序结果是否正确。同时记录排序时间和用 `std::make_heap` 进行排序的时间并比较。同理操作有序、逆序、部分重复四种序列。最后将结果输出到文件中。效率表格如下

	my heapsort time	`std::make_heap`
random sequence	0.0774519s	0.0726108s
ordered sequence	0.0305473s	0.0289308s
reverse sequence	0.035277s	0.0330002s
repetitive sequence	0.0496237s	0.0451618s

表 1: 堆排序与 `std::make_heap` 的时间对比

2 补充

经过理论计算，`heapSort` 的时间复杂度应该是 $O(n \log n)$，而 `std::sort_heap` 的时间复杂度也是 $O(n \log n)$。理论上两者的时间复杂度是相同的。但是在实际测试中，我发现 `std::sort_heap` 的时间要比 `heapSort` 的时间要短，猜测可能是因为 `std::sort_heap` 是 STL 中的一个函数，它的实现可能比我自己写的 `heapSort` 更加高效。