1 设计思路

本次实现了基于最大堆的堆排序算法。主要包含两个核心函数：

1. heapify：用于维护最大堆性质的核心函数
2. heapsort：主排序函数

1.1 核心函数实现

- heapify 函数实现了”下沉”操作，确保以某个节点为根的子树满足最大堆性质
- heapsort 函数分两个阶段：
  1. 建堆阶段：自底向上构建最大堆
  2. 排序阶段：反复取出堆顶元素并重建堆

2 测试方案

测试程序设计了四种不同的输入序列：

- 随机序列：使用随机数生成器生成
- 有序序列：已经排好序的序列
- 逆序序列：完全逆序的序列
- 部分重复序列：有限范围内的随机数，确保存在重复元素

每种序列的测试规模均为 100 万个元素。

3 性能对比

序列类型	my heapsort time(ms)	std::sort_heap(ms)
随机序列	83	81
有序序列	51	33
逆序序列	50	37
部分重复序列	90	80

表 1: 不同序列类型的排序性能对比

4 复杂度分析

4.1 时间复杂度

- heapify 操作：O(log n)
- 建堆阶段：O(n)
- 排序阶段：O(n log n)

整体时间复杂度为 O(n log n)。

4.2 与 std::sort_heap 的效率差异分析

从测试结果可以看出：

1. 在随机序列和部分重复序列上，我的实现与标准库性能接近
2. 在有序和逆序序列上，标准库实现显著优于我的实现
3. 性能差异的主要原因：
    - 我的实现使用递归的 heapify，而标准库使用迭代实现