项目名称：外部排序算法的实现

项目背景

外部排序：待排序的记录数量大，无法一次性载入内存，整个排序过程必须借用外存分批读入内存。外部排序基本上由两个相对独立的阶段组成：

第一阶段：按可用内存大小,利用内部排序方法，构造若干个记录的有序子序列写入外存，通常称这些记录的有序子序列为 “归并段”或“顺串”；

第二阶段：归并排序，通过“归并”，逐步扩大记录的有序子序列的长度，直至外存中整个记录序列按关键字有序为止。

项目主要功能

本项目要实现外部排序算法。排序函数读入一个待排序的文件“unsorted.txt”（文件内容是随机生成的2×10^7个数字），然后通过外部排序算法对其排序,最终写入到自己任意命名的文件当中，要求：

1. 使用尽量少的内存来完成排序，使用内存越少则分数越高。
2. 磁盘空间使用量不做限制，但建议不要超过 500MB。
3. 运行时间不做限制，但建议不要超过 10 分钟。
4. 生成的待排序数据和排序后的数据中，数字之间用换行符分隔。

具体来说，外部排序算法的实现分为以下几步：

1. 模拟分批读入数据完成第一阶段排序的过程，在内存中排序产生若干初始归并段。
2. 确定 k，根据“最佳归并树”或其他方法，确定归并方案。
3. 使用 k-路平衡归并完成若干趟归并，最终使得文件内数据有序。

项目实现任务

本项目需要实现两个任务：

任务一：完成 external_sort.cpp 里的外部排序函数。附件中的 main_without_score.cpp 可以用来帮助你调试和测试自己写的外部排序函数。

任务二：将自行实现的外部排序算法和附件中的文件一起按要求手动编译成可执行文件，完成评测。

附件中的文件说明：

1. main.o：该文件包含除了外部排序算法以外的所有实现。
2. header.h：该文件包含外部排序算法的函数声明，以及一个 reader 的声明。
3. reader.cpp：该文件包含从 ifstream 读入一个整数的实现。

将 main.o、header.h、reader.cpp、外部排序的实现（即 external_sort.cpp）放在同一目录下，参考以下编译命令：

```
g++ external_sort.cpp -c
g++ reader.cpp -c
g++ main.o reader.o external_sort.o -o sort.exe
```

运行 sort.exe，程序会自动生成待排序文件，并执行外部排序算法进行排序，排序过程中会对排序占用的内存进行监控，记录瞬时最大占用内存，排序完成后对结果文件进行检验，并根据最大内存使用量 M（以 Byte 为单位）进行打分，打分公式为：

$$\min(\frac{2\times 10^7\times 40}{M},100)$$

打分结果将记录在 score.data 文件中，该文件存储内容已加密，请勿随意更改导致分数丢失！

项目评分要点：

1. 将提供的辅助文件和用 C++实现的外部排序算法编译成可执行文件并成功运行（60 分，具体得分按照 score.data **中的得分按比例折算**）。
2. 在外部排序过程中，给出如下情况下的排序效率（内存使用情况，总的执行时间）对比：
   1) 直接使用内部排序算法产生初始归并段，根据指定 k 值进行 k 路归并完成排序。（10 分）

2) 使用置换选择算法（加入败者树结构）产生初始归并段，根据指定 k 值和产生的初始归并段构造最佳归并树完成排序。（15 分）

3) 使用置换选择算法（加入败者树结构）产生初始归并段，使用不同 k 值构造最佳归并树完成排序，k = 2,3,4,5,6……。给出不同的 k 值的排序效率。（15 分）