预处理

外排序

大多数内排序算法都是利用了内存是直接访问的事实,读写一个数据是常量的时间。如果输入是在磁带上，磁带上的元素只能顺序访问。甚至数据是在磁盘上，效率还是下降，因为转动磁盘和移动磁头会产生延迟。

- 外排序模型
- 预处理
- 归并

预处理阶段

- 最简单的方法是按照内存的容量尽可能多地读入数据记录，然后在内存进行排序，排序的结果写入文件，形成一个已排序片段。
- 每次读入的记录数越小，形成的初始的已排序片段越多。而已排序片段越多，归并的次数也越多。
- 如果能够让每个初始的已排序片段包含更多的记录，就能减少排序时间。置换选择可以让我们在只能容纳p个记录的内存中生成平均长度为2p的初始的已排序片段。

置换选择

- 如何更有效地构造已排序片段
- 事实上，只要第一个元素被写到输出磁带上，它所用的内存空间就可以给别的元素使用。如果输入磁带上的下一个元素比刚刚输出的元素大，它能被放入这个已排序片段。

置换选择过程

1. 初始时，将M个元素读入内存，用一个优先队列存储这M个元素。
2. 执行一次取最小元素操作，把最小的元素写入输出磁带。
3. 从输入磁带读入下一个元素。
    1. 如果它比刚才写出去的元素大，则把它加入到优先级队列；
    2. 否则，它不可能进入当前的已排序片段。因为优先级队列比以前少了一个元素，该元素就被放于优先级队列的空余位置，
4. 继续这个过程，直到优先级队列的大小为0，此时该已排序片段结束。我们重新构建一个优先级队列，开始了一个新的已排序片段，此时用了所有存放在空余位置中的元素。

a[0]	a[1]	a[2]	输出
1	4	10	1
2	4	10	2
4	10	0	4
5	10	0	5
7	10	0	7
10	0	6	10
0	6	3	已排序片段结束
0	3	6	0
3	6	9	3
6	9	12	6
9	12		9
12			12
			已排序片段结束

已排序片段构建实例：

文件上的数据为1、4、10、2、0、5、7、6、3、9、12，

内存中能够容纳3个记录