

数据结构与算法

Data Structures and Algorithms

第六部分 排序

数据结构

一、线性表

- (一) 线性表的基本概念
- (二) 线性表的实现
- (三) 线性表的应用

二、栈、队列和数组

- (一) 栈和队列的基本概念
- (二) 栈和队列的顺序存储结构
- (三) 栈和队列的链式存储结构
- (四) 多维数组的存储
- (五) 特殊矩阵的压缩存储
- (六) 栈、队列和数组的应用

三、树与二叉树

- (一) 树的基本概念
- (二) 二叉树
- (三) 树、森林
- (四) 树与二叉树的应用

四、图

- (一) 图的基本概念
- (二) 图的存储及基本操作
- (三) 图的遍历
- (四) 图的基本应用

数据结构

五、查找

- (一) 查找的基本概念
- (二) 顺序查找法
- (三) 分块查找法
- (四) 折半查找法
- (五) B树及其基本操作,
B+树的基本概念
- (六) 散列(Hash)表
- (七) 字符串模式匹配
- (八) 查找算法分析及应用

六、排序

- (一) 排序的基本概念
- (二) 插入排序
- (三) 起泡排序
- (四) 简单选择排序
- (五) 希尔排序
- (六) 快速排序
- (七) 堆排序
- (八) 二路归并排序
- (九) 基数排序
- (十) 外部排序

算 法

回顾：内部分类

(一) 排序的基本概念

(二) 插入排序算法

直接插入排序；折半插入排序；希尔排序(Shell sort)

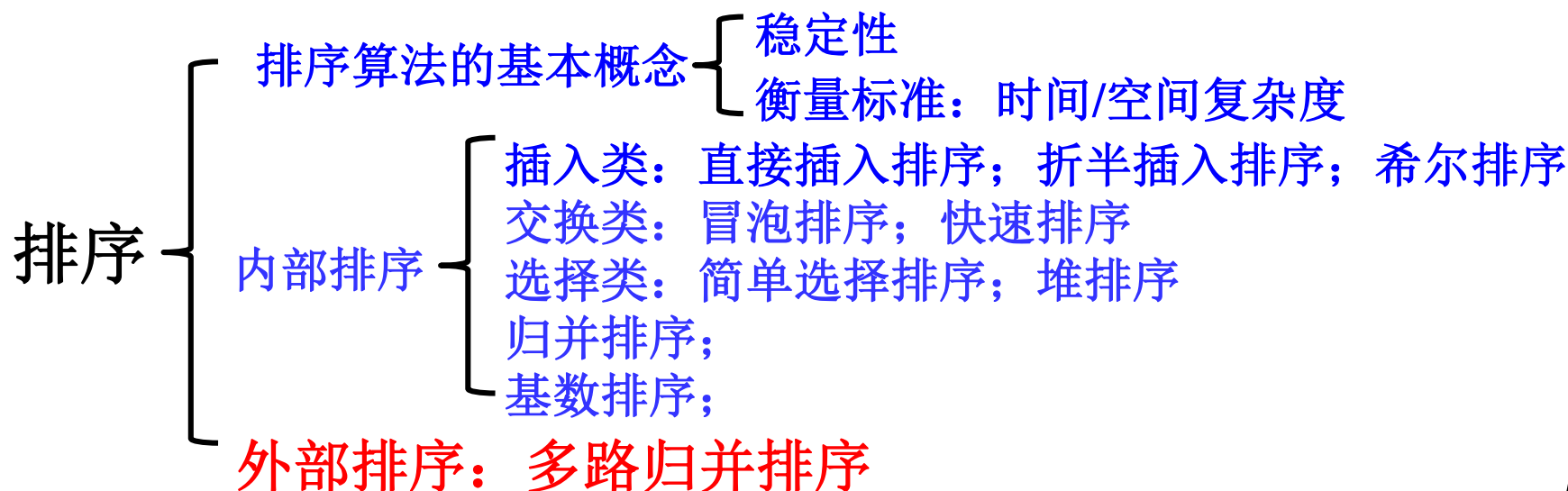
(三) 交换类排序算法：冒泡排序；快速排序

(四) 选择类排序算法：简单选择排序；堆排序

(五) 归并类排序算法：2路归并排序

(六) 基数排序 (七) 外部排序 (八) 各种排序算法的比较

(九) 排序算法的应用



主要内容

7.1	磁盘文件的归并排序
7.2	磁带文件的归并排序

归并方法：首先将文件中的数据输入到内存，采用内部分类方法进行分类（归并段），然后将有序段写回外存；对多归并段进行多遍归并，最后形成一个有序序列。

7.1 磁盘文件的归并分类



磁盘信息的存取

磁盘：是一个扁平的圆盘，盘面上有许多称为磁道的圆圈，信息就记载在磁道上。它是一种直接存取的存储设备（DASD）。

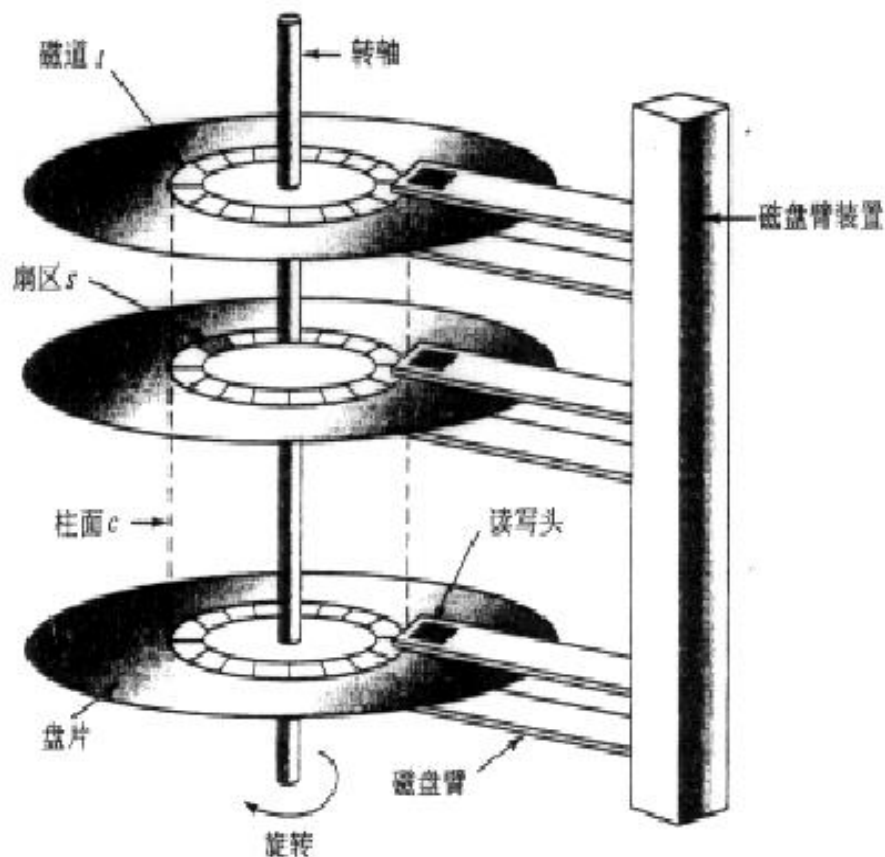
磁盘的工作原理：盘片装在一个主轴上，并绕主轴高速旋转，当磁道在读/写头下通过时，便可进行信息的读/写。读/写信息的功能由磁盘驱动器执行。

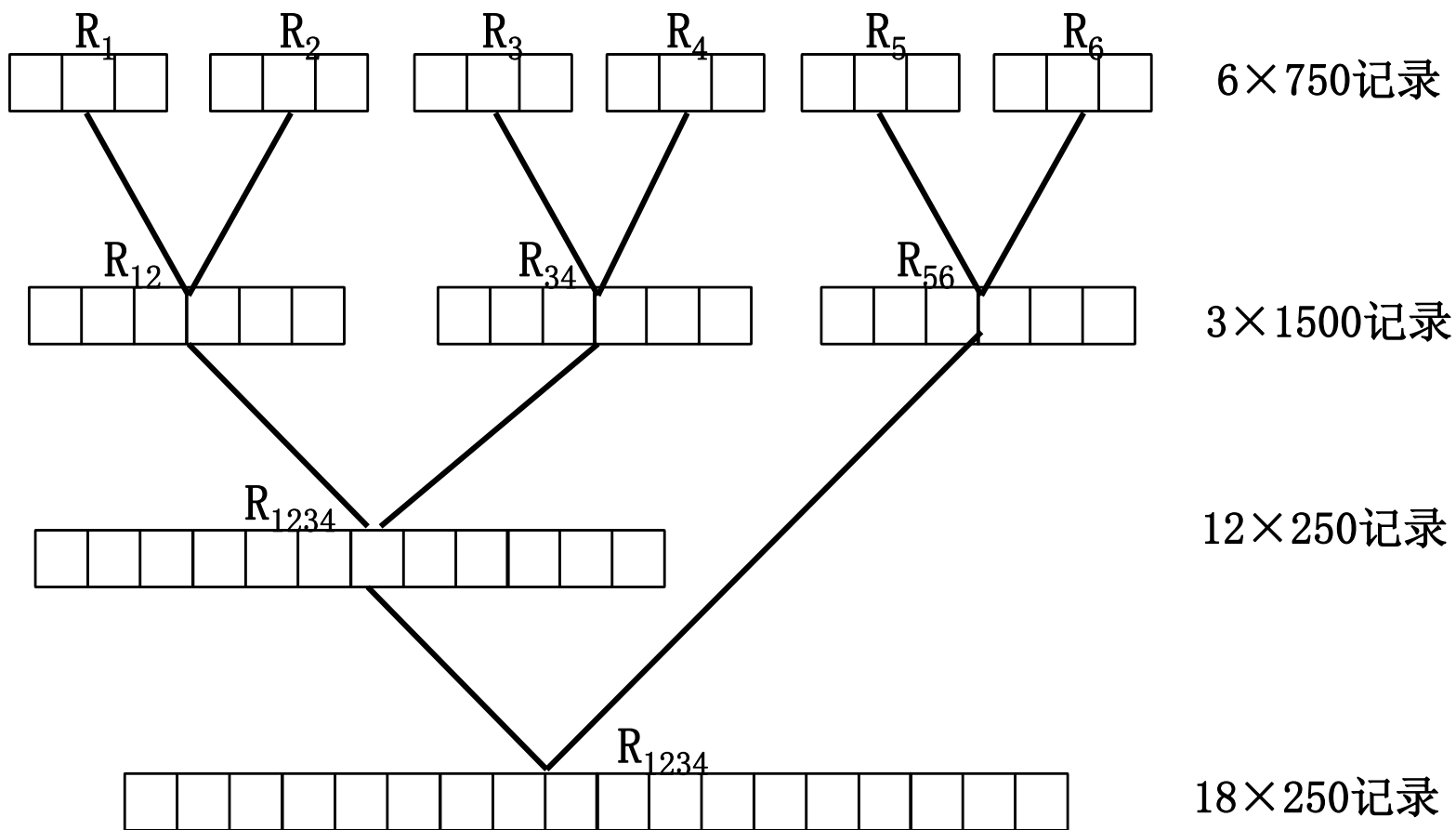
固定头盘：固定头盘的每一磁道上都有独立的磁头，这些磁头固定不动，专负责读/写某一磁道上的信息。

活动头盘：活动头盘的磁头是可以移动的。一个盘面上只有一个磁头，磁头装在一个动臂上，可以从该面上的一道移动到另一道。

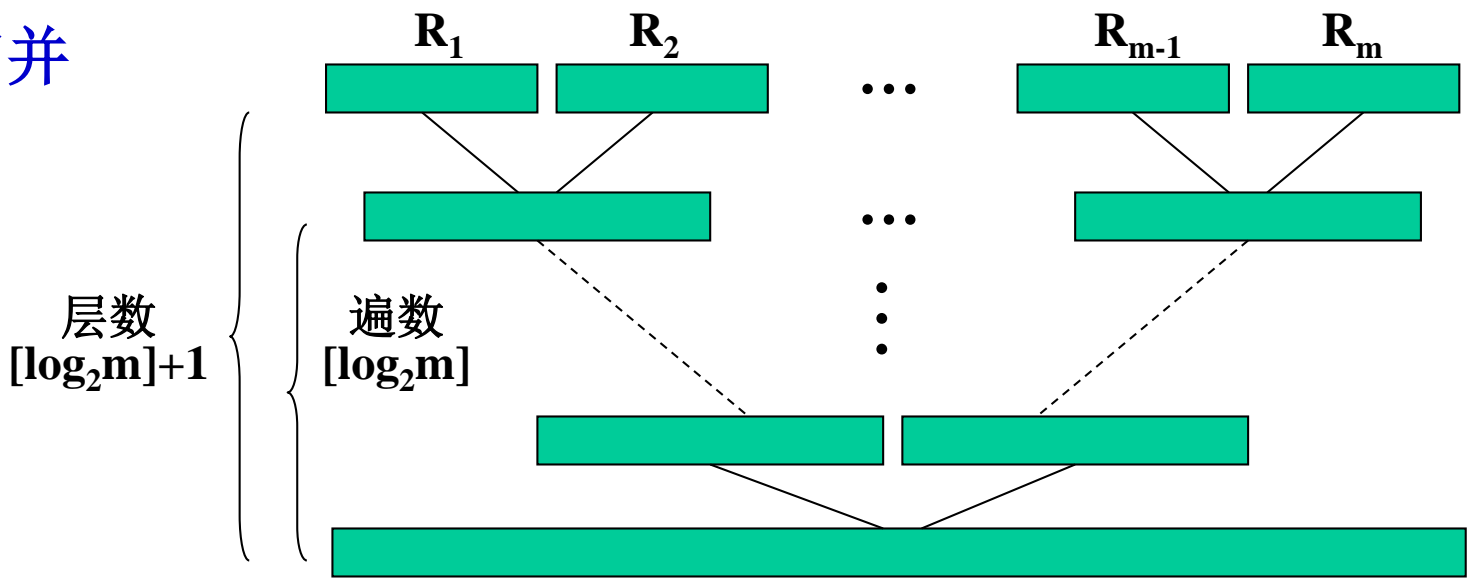
在磁盘上表明一个具体信息必须用一个**三维地址**：柱面号（确定读/写头的径向运动）、盘面号、块号（确定信息在盘片圆圈上的位置）。

磁盘结构：由磁盘驱动器、读、写磁头、活动臂、盘片（磁道、扇区）、旋转主轴构成。速度快、容量大、直接存取设备。



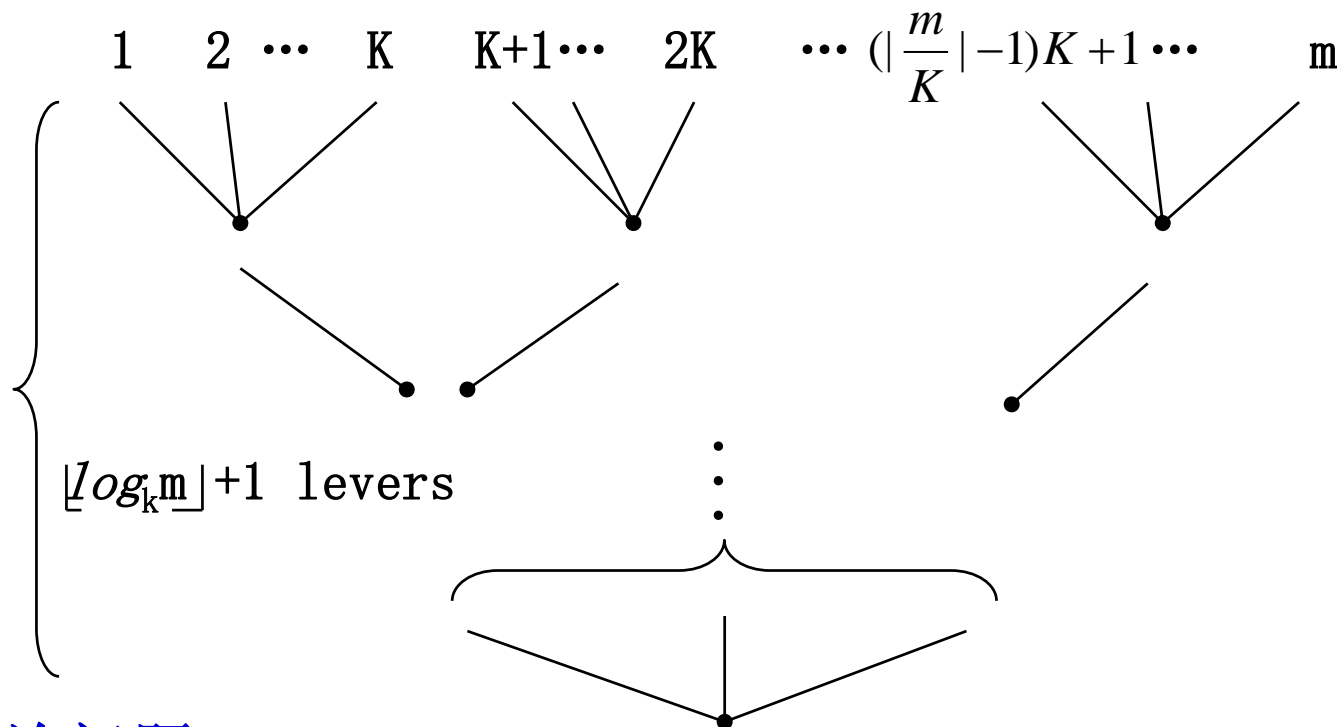


K路归并



M 个归并段的归并过程

$\log_k m$ 遍比较次数:



讨论问题:

- (1) 多路归并——减少归并遍数
- (2) 并行操作的缓冲区处理
——使输入、输出和CPU处理尽可能重叠
- (3) 初始归并段的生成（内排实现）

提高外排序效率的途径:

- ① 扩大初始归并段长度, 从而减少初始归并段个数 m
- ② 进行多路(k 路)归并减少合并趟数 s , 以减少I/O次数

$$s = \lceil \log_k m \rceil$$

(1) 多路归并——减少归并遍数

m 个初始段进行 2 路归并, 需要 $\log_2 m$ 遍归并;

m 个初始段, 采用 k 路归并, 需要 $\log_k m$ 遍归并。

显然, k 越大, 归并遍数越少, 可提高归并的效率。

在 k 路归并时, 从 k 个关键字中选择最小记录时, 要比较 $k-1$ 次。若记录总数为 n , 每遍要比较的次数为:

$$n \cdot (k-1) \lceil \log_2 m / \log_2 k \rceil$$

可以看出, 随着 k 增大, $(k-1)/\log_2 k$ 也增大, 当归并路数多时, CPU 处理的时间也随之增多。为此要选择好的分类方法, 以减少分类中比较次数。

选择树 (Selection tree) 或 败者树 (tree of loser)

分析:

第一次建立选择树的比较所花时间为:

$$O(k-1) = O(k)$$

而后每次重新建造选择树所需时间为:

$$O(\log_2 k)$$

n 个记录处理时间为初始建立选择树的时间加上 $n-1$ 次重新选择树的时间:

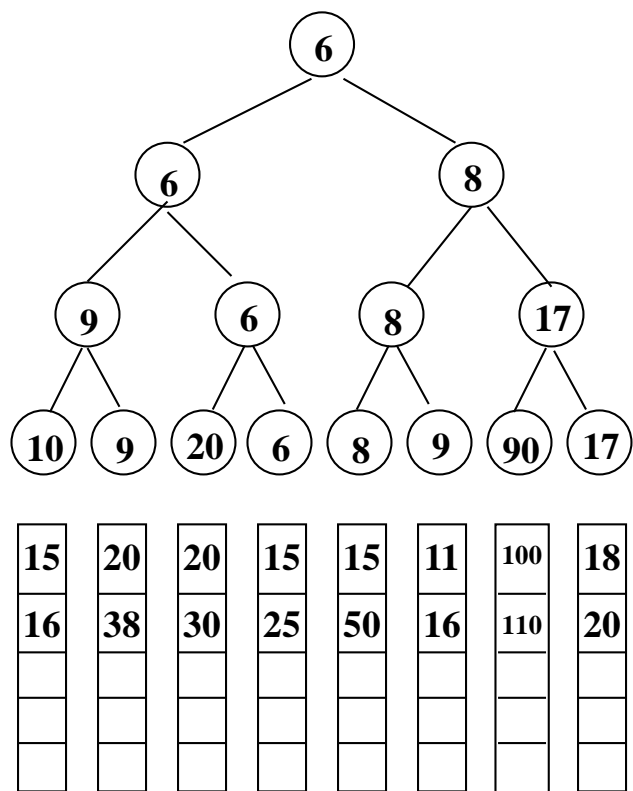
$$O((n-1) \cdot \log_2 k) + O(k) = O(n \cdot \log_2 k)$$

这就是 k 路归并一遍所需的 CPU 处理时间。

归并遍数为 $\log_k m$, 总时间为:

$$O(n \cdot \log_2 k \cdot \log_k m) = O(n \cdot \log_2 m)$$

(k 路归并 CPU 时间与 k 无关)



最佳归并树

将哈夫曼树进行拓展，不仅对2叉树，同样可形成3叉、4叉、...、 k 叉树，亦称为哈夫曼树，同样可求得带权路径长度最小。

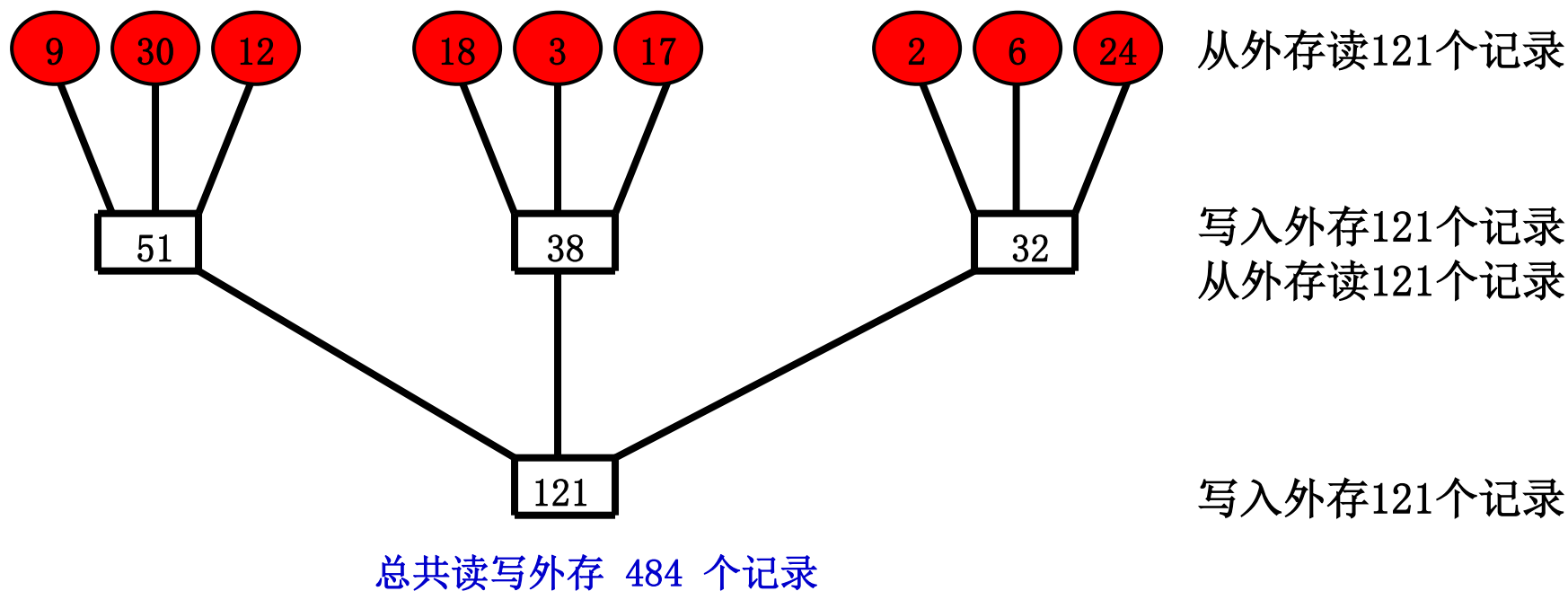
对长度不等的 m 个初始归并段，**构造哈夫曼树作为归并树**，可使在进行外部归并时所需要对外存进行的读写次数达到最小。

最佳归并树中，并不只是只有度为 k 和0的结点，会有缺额。当初始归并段的数目不足时，需附加长度为0的虚段，按照哈夫曼树的构造原则，权为0的叶子结点应离树根最远。

问题：

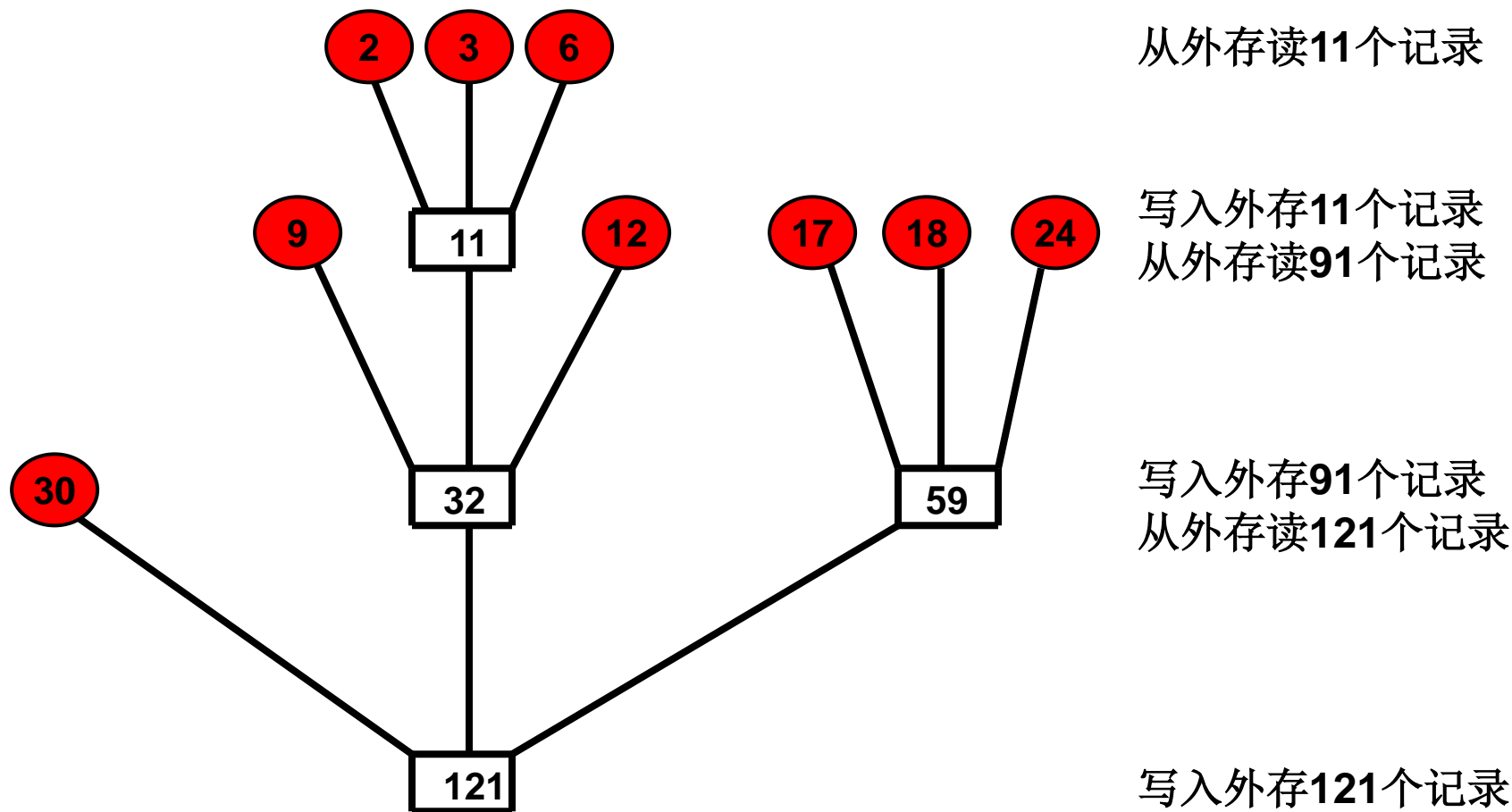
- **起因**：由于初始归并段通常不等长，进行归并时，长度不同的初始归并段归并的顺序不同，读写外存的总次数也不同。
- **目的**：减少读写外存的次数。

【例7-5】9个初始归并段，记录数分别为9、30、12、18、3、17、2、6、24。如果进行3-路归并，请讨论在各种情况下的对外存的读写次数。



$$\text{读写磁盘次数} = \sum w_j \cdot l_j = (9+30+12+18+3+17+2+6+24) \cdot 2 = 242$$

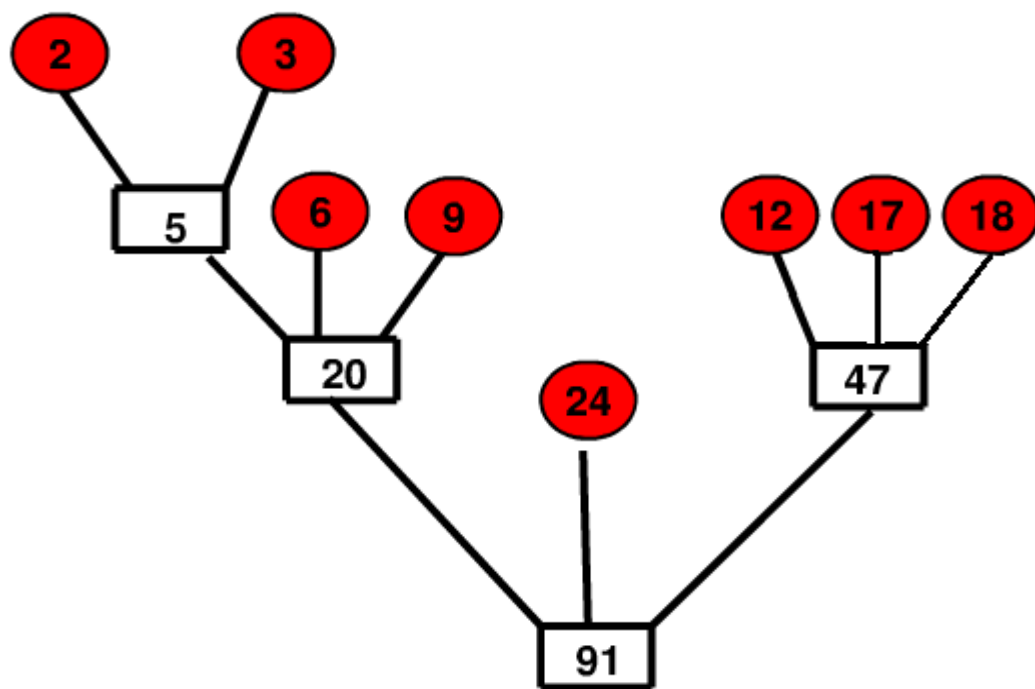
按照hafuman树的思想，记录少的段最先合并。不够时增加虚段。



总共读写外存 446 个记录

$$\text{读写磁盘次数} = \sum w_j \cdot l_j = (2+3+6) \cdot 3 + (9+12+17+18+24) \cdot 2 + 30 \cdot 1 = 223$$

【例7-6】8个初始归并段，记录数分别为2、3、6、9、12、17、18、24。如果进行3-路归并，请讨论在各种情况下的对外存的读写次数。



从外存读5个记录

写入外存5个记录
从外存读67个记录

写入外存67个记录
从外存读91个记录

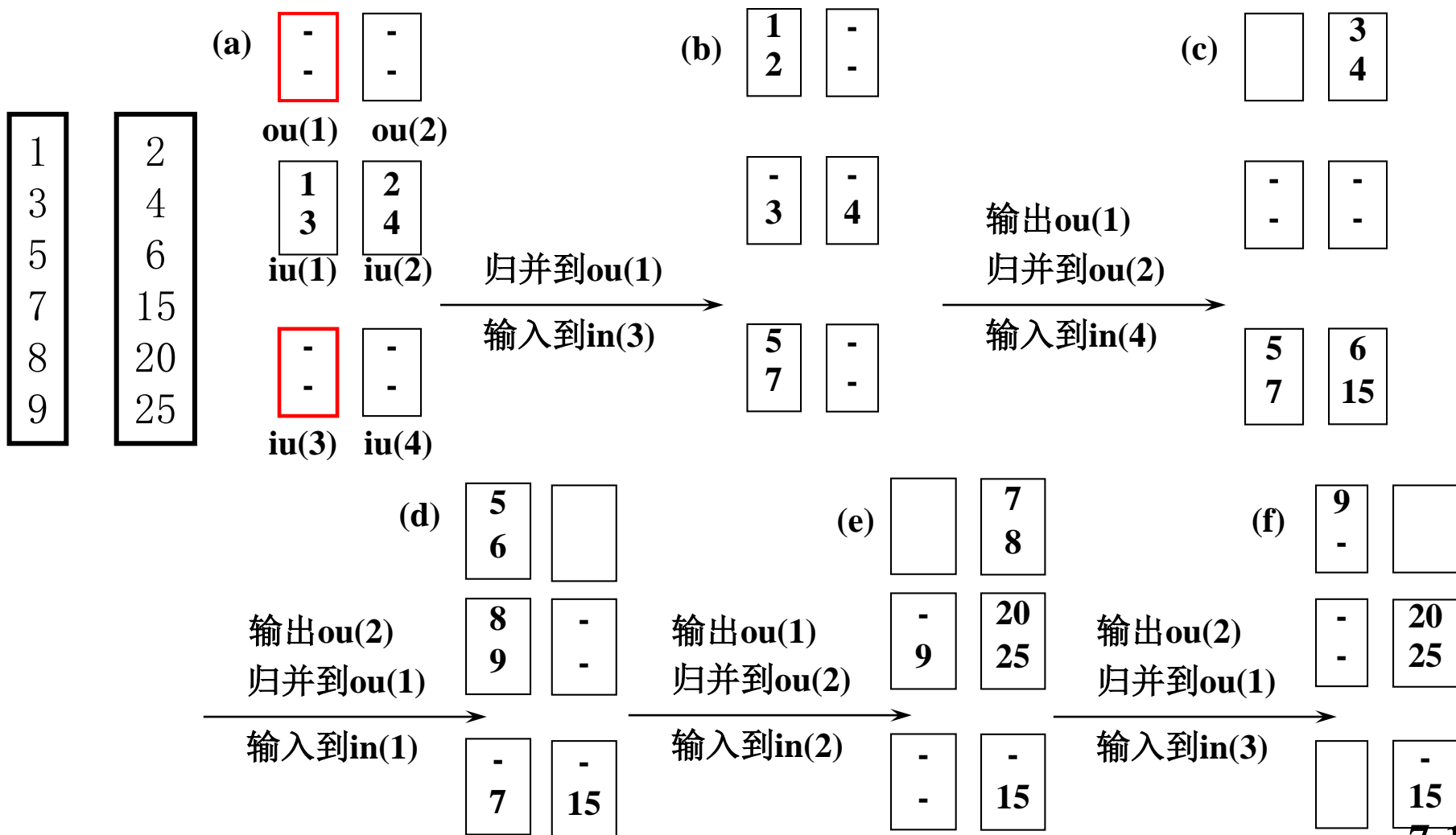
写入外存91个记录

共读写326个记录

$$\text{读写磁盘次数} = \sum w_j \cdot l_j = (2+3) \cdot 3 + (6+9+12+17+18) \cdot 2 + 24 \cdot 1 = 163$$

(2) 并行操作的缓冲区处理

对 k 个归并段进行 k 路归并至少需要 k 个输入和1个输出缓冲区，要使输入、输出和归并同时进行， $k+1$ 个缓冲区是不够的，需要 $2k$ 个输入缓冲区实现并行操作。



(3) 初始归并段的生成 置换-选择法

- (a) 初始归并段的长度≥缓冲区的长度
- (b) 任何内部分类算法都可作为生成初始归并段的算法
- (c) 例如：缓冲区的长度为4，输入序列为：

15 19 04 83 12 27 11 25 16 34 26 07 10 90 06 ...

新输入记录. key小于当前记录. key，等待下一个归并段

步	1	2	3	4	5	6	7	8	9	10	11	12	13	...
缓冲区内容	15	15	15	(11)	(11)	(11)	(11)	(11)	(11)	11	11	(06)
	19	19	19	19	25	(16)	(16)	(16)	(16)	16	16	16
	04	12	27	27	27	27	34	(26)	(26)	26	26	26
	83	83	83	83	83	83	83	83	(07)	10	90	90
输出结果	04 12 15 19 25 27 34 83 07 10 11 16 ...													
	R ₁							R ₂						

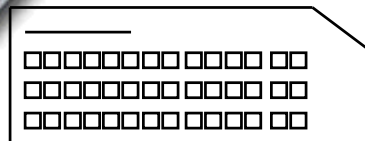
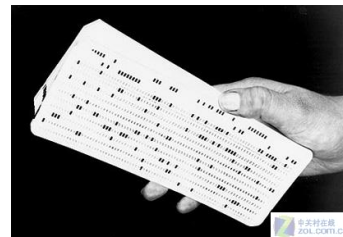
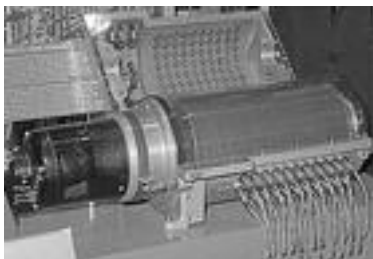
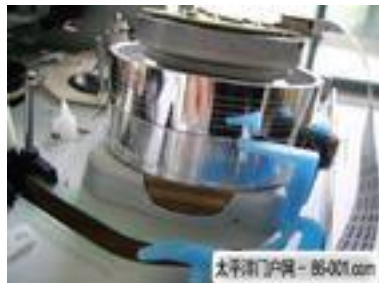
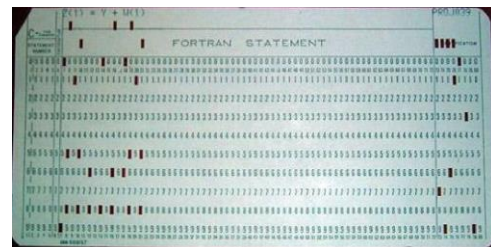
采用置换-选择法生成初始归并段的长度平均是缓冲区长度的两倍。

磁盘文件的归并分类小结：

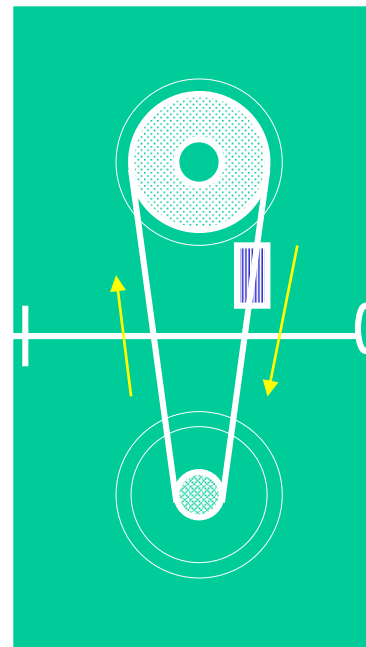
- (1) 多路归并—减少归并遍数（败者树、最佳归并树）
- (2) 并行操作的缓冲区处理—使输入、输出和CPU处理尽可能重叠（引入缓冲区）
- (3) 初始归并段的生成（置换-选择法）

7.2 磁带文件的归并分类

(外部) 存储设备——纸带、磁鼓、磁带、磁盘等

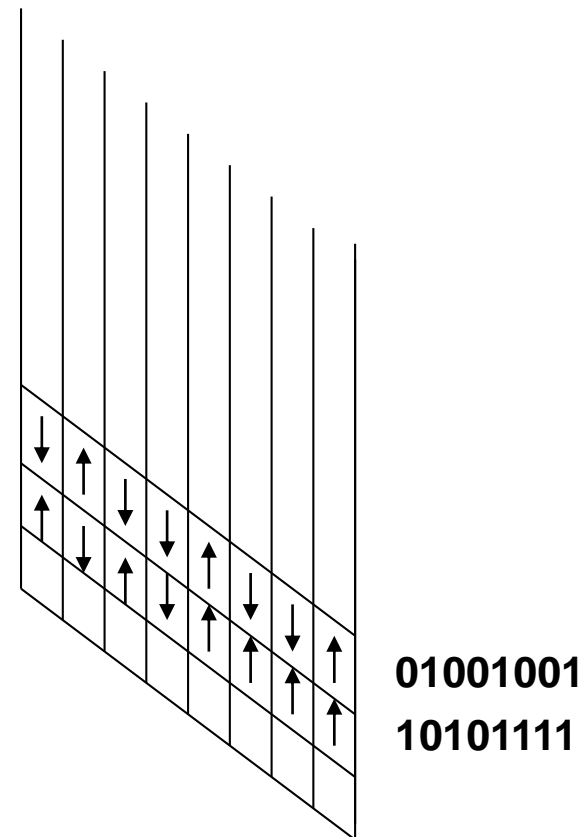


eNet



• 磁带信息的表示:

- ↑ 一种磁化方向、代表1
- ↓ 另一种磁化方向，代表0



与磁盘不同，磁带是顺序存储设备，读取信息块的时间与信息块的位置有关。研究磁带分类，需要了解信息块的分布。

k路平衡归并分类

磁带机数量: $2k$

输入: T_1, T_2, \dots, T_k 输出 \uparrow
 ↓ 输出: $T_{k+1}, T_{k+2}, \dots, T_{2k}$ 输入 \uparrow

磁带机	T_1	T_2	...	T_k
归并段	R_1	R_2	...	R_k
	R_{k+1}	R_{k+2}	...	R_{2k}

	R_{mk+1}

$T_1: R_1(1000), R_3(1000), R_5(1000)$
 $T_2: R_2(1000), R_4(1000), R_6(1000)$
 $T_3: \emptyset$
 $T_4: \emptyset$

$T_1: \emptyset$
 $T_2: \emptyset$
 $T_3: R_1(2000), R_3(2000)$
 $T_4: R_2(2000)$

$T_1: R_1(4000)$
 $T_2: R_2(2000)$
 $T_3: \emptyset$
 $T_4: \emptyset$

$T_1: \emptyset$
 $T_2: \emptyset$
 $T_3: R_1(6000)$
 $T_4: \emptyset$

以 $k=2$ 为例，用三台磁带机T1，T2，T3，假设初始归并段长度为L。
初始归并段的段数为34。过程如表1所示。

将上例递归过程从最后一步逆推，如表2所示。
每一步归并段总数排列成序列为：
1，2，3，5，8，13，21，34,... 刚好组成Fibonacci数列， $F_k=F_{k-1}+F_{k-2}$

K 路多阶段归并，可从2路归并扩充，对应 k 阶Fibonacci数列。

$$\begin{cases} F_n^{(k)} = 0 \\ F_n^{(k)} = 1 \\ F_n^{(k)} = F_{n-1}^{(k)} + F_{n-2}^{(k)} + \dots + F_{n-k}^{(k)} \end{cases}$$

结论：
 $K+1$ 台磁带机 k 路多阶段归并，
在 $n-j$ 步归并段的分布规则= \rangle

第 j 步归并断总数：

$$F_{G(j+k-1)}^{(k)} = t_1^j + t_2^j + \dots + t_k^j$$

其中 t_i^j 表示：

第 j 步中逻辑磁带上第 i 台磁带机。

$$\begin{cases} t_1^j = F_{j+k-21}^{(k)} \\ t_2^j = F_{j+k-3}^{(k)} + F_{j+k-2}^{(k)} \\ \dots \\ t_i^j = F_{j+k-i-1}^{(k)} + \dots + F_{j+k-2}^{(k)} \\ \dots \\ t_{k-1}^j = F_j^{(k)} + F_{j+1}^{(k)} + \dots + F_{j+k-2}^{(k)} \\ t_k^j = F_{j-1}^{(k)} + F_j^{(k)} + \dots + F_{j+k-2}^{(k)} \end{cases}$$

i遍后	t ₁	t ₂	t ₃
开始	13(1L)	21(L)	空
1	空	8(1L)	13(2L)
2	8(3L)	空	5(2L)
3	3(3L)	5(5L)	空
4	空	2(5L)	3(8L)
5	2(13L)	空	1(8L)
6	1(13L)	1(21L)	空
7	空	空	1(34L)



步	t ₁	t ₂	t ₃	总段数
n	0	0	1	1
n-1	1	1	0	2
n-2	2	0	1	3
n-3	0	2	3	5
n-4	3	5	0	8
n-5	8	0	5	13
n-6	0	8	13	21
n-7	13	21	0	34

【例7-7】 设有磁盘文件中记录的关键字分别为：

10,20,15,25,12,13,21,30,8,16,10

用置换-选择排序法产生初始归并段，问可产生几个初始归并段？每个初始归并段包含哪些记录（设工作区能容纳4个记录）。

解： 内存缓冲区可容纳4个记录，采用4路归并的置换-选择排序方法生成初始归并段，如表所示。

步	1	2	3	4	5	6	7	8	9	10	11
缓冲区内容	10	12	13	21	21	21	(16)	(16)	16	16	
	20	20	20	20	20	(8)	(8)	(8)			
	15	15	15	15	30	30	30	30			
	25	25	25	25	25	25	25	(10)	10		
输出结果	10	12	13	15	20	21	25	30	8	10	16
	 生成的第一个初始归并段							 第二个初始归并段			