

# 数据结构

## Ch11 外部排序

北京邮电大学 计算机学院

# 第11章 外部排序

- 引言
- 11.1 外存信息的存取
- 11.2 外部排序的基本方法—归并排序法
- 11.3 多路平衡归并
- 11.4 置换-选择排序
- 11.5 最佳归并树

## [外部排序的应用对象]

保存在外存储器上的信息量很大的数据记录文件。

## [外排序与内排序的差别]

内部排序充分利用内存可以随机存取的特点，如

希尔排序中，相隔 $d_i$ 的记录关键字可作比较；

堆排序中，完全二叉树中的父 $R[i]$ 与子 $R[2i]$ 、 $R[2i+1]$ 可比；

快速排序中，需正向和逆向访问记录序列

外存信息的定位和存取受其物理特性的限制

## [外部排序的实现手段]

在排序过程中，进行多次内外存之间的数据交换

# 11.1 外部信息的存取

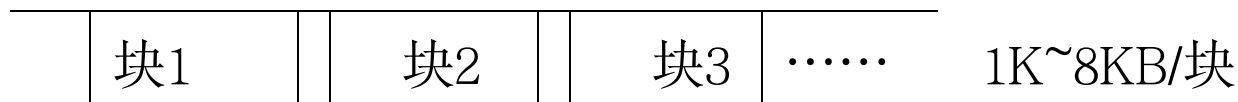
## 11.1.1 磁带信息的存取

属于**顺序存取设备**，只适用于存储顺序文件

### [工作原理]

磁带不是连续运转设备，而是启停设备

I/O操作以块为单位

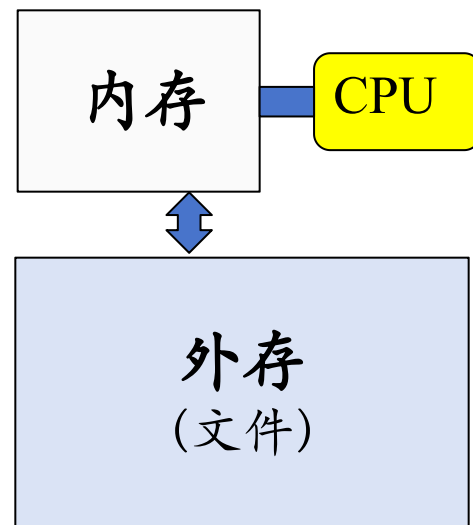


IBG(块间隙)

读写一块信息的时间 $T_{I/O} = t_a + nt_w$

$t_a$ ：用于磁头定位在物理块起始位置的时间

$t_w$ ：传输一个字符的时间



## 11.1.2 磁盘信息的存取

属于**直接存取设备**，适用于各类存储形式的文件

### [工作原理]

磁道：盘片上的同心圆

柱面：各盘面上半径相同的磁道合在一起

扇区：每个磁道等分为若干扇区(块)

读取信息的过程：

柱面号 移动磁头

盘面号(磁道号) 选定所用磁头

扇区/块号 磁盘一直高速旋转，磁头等待开始位置

读写一块信息的时间 $T_{I/O} = t_{seek} + t_{la} + nt_w$

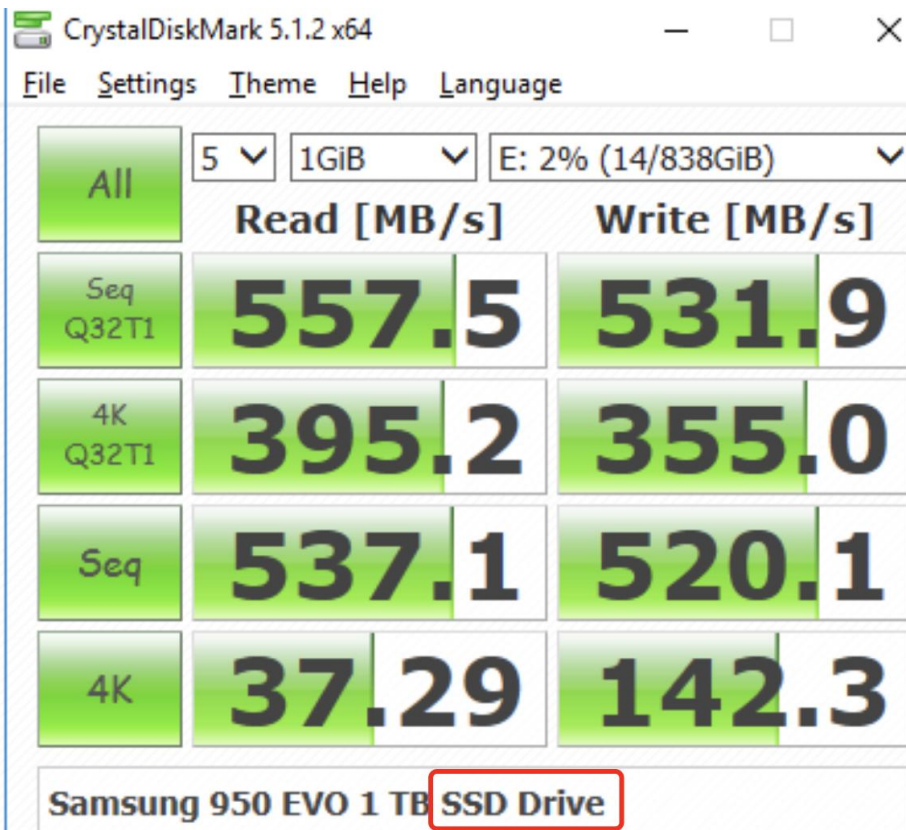
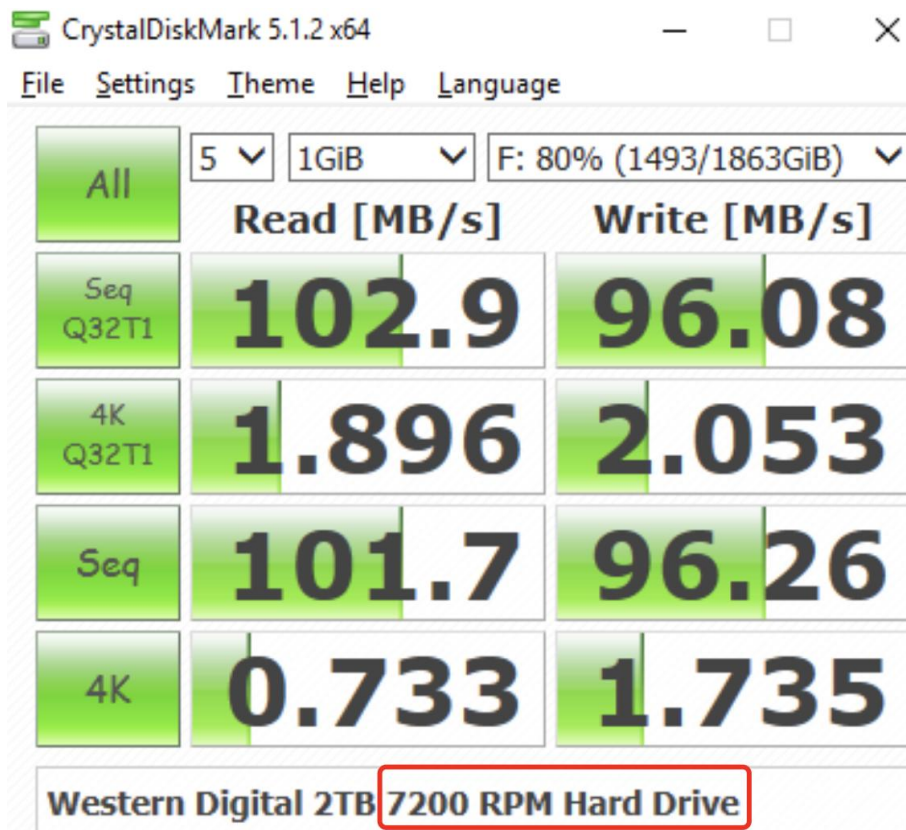
$t_{seek}$ ：磁头定位到磁道的时间

$t_{la}$ ：等待时间



HDD

## 11.1.2 磁盘信息的存取

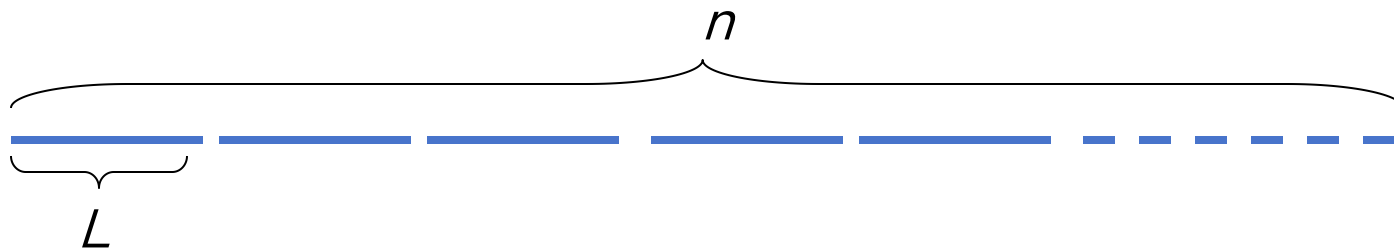


## 11.2 外部排序的基本方法—归并排序法

### [步骤]

#### 1) 生成若干初始归并串/顺串(文件预处理)

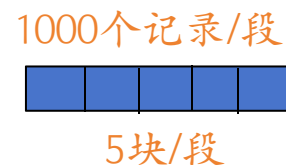
把含有 $n$ 个记录的文件，按内存缓冲区大小分成若干长度为 $L$ 的子文件/段；  
分别调入内存用有效的内排序方法排序后送回外存；



#### 2) 多路合并

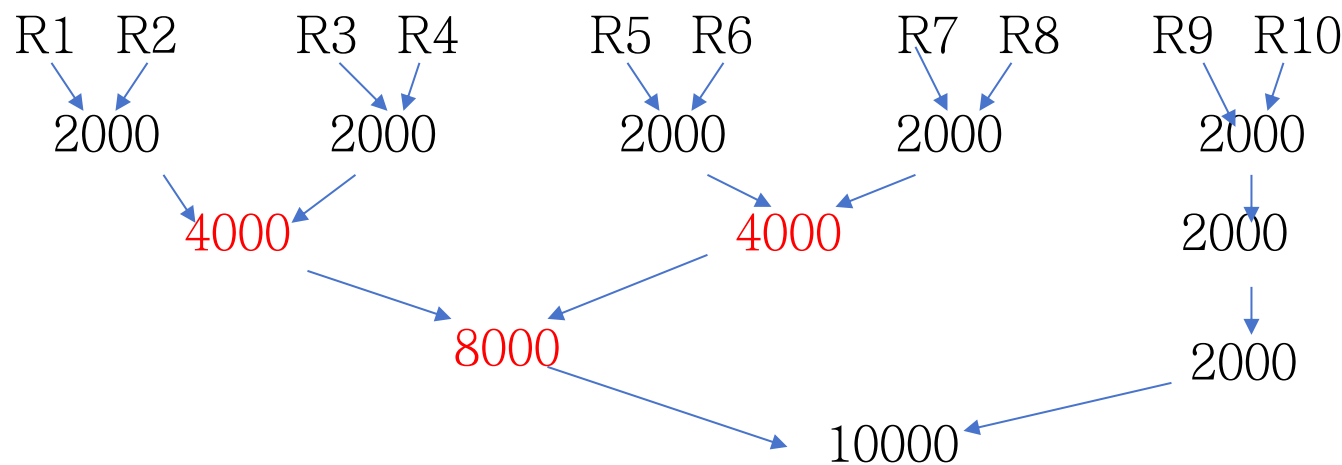
对初始归并串逐趟合并，直至最后在外存上得到整个有序文件为止

**[例]**某文件共10000个记录，设每个物理块可以容纳200个记录，内存缓冲区可以容纳5个物理块（即内存缓冲区可以容纳1000个记录）



1) 经过10次内排序后得到10个初始归并段R1~R10

2) 采用两路归并，需四趟可以得到排好序的文件



外排序总时间:

内排序:  $10t_{is}$

$t_{is}$  是对一个初始归并段进行内部排序所需时间

I/O操作:  $100t_{IO}$  (初始) +  $(100+80+80+100)t_{IO}$  (归并) =  $460t_{IO}$ , 和归并趟数有关

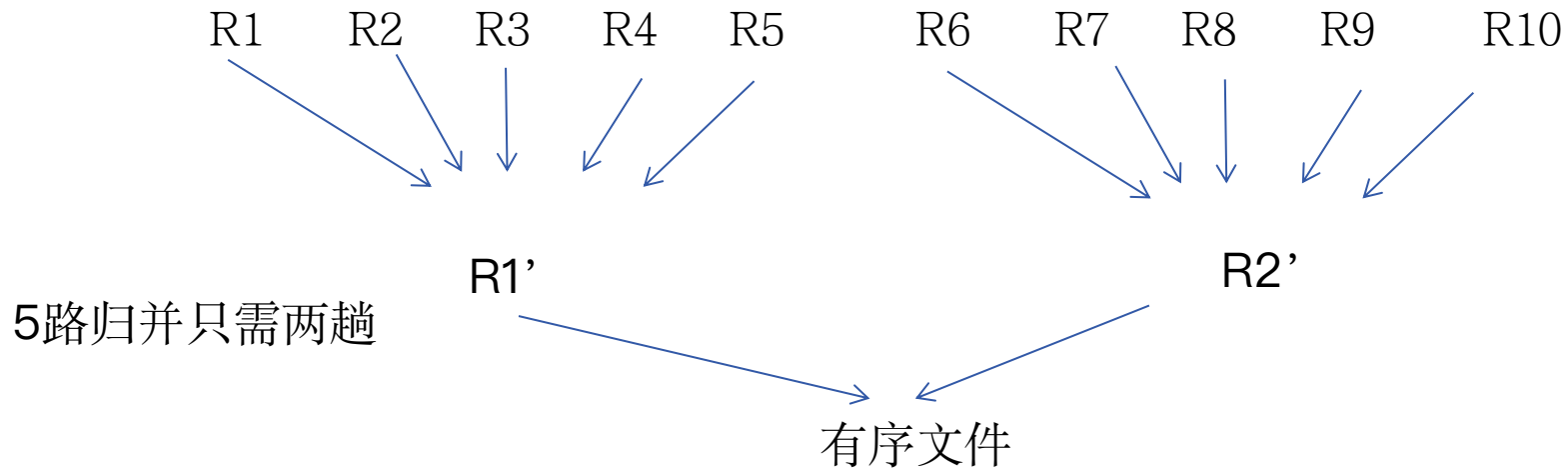
内部归并:  $(10000+8000+8000+10000)t_{mg} = 46000 t_{mg}$

第2趟归并, 共读写8000记录, 读写各40块



## [提高外排序效率的途径]

提高外排的效率应主要着眼于减少外存信息读写的次数。



减少合并趟数 $s$ ，以减少I/O次数

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

途径1: 扩大初始归并段长度，从而减少初始归并段个数 $m$

途径2: 进行多路( $k$ 路)归并， $k$ 个记录中选最小如何减少比较次数?

## 11.3 多路(k路)平衡归并

[通过简单比较进行归并存在的问题]

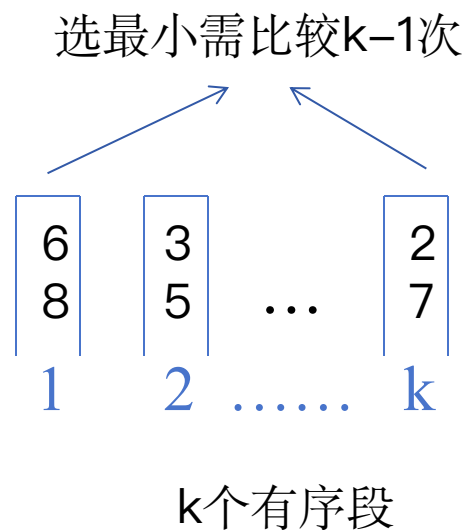
设记录总数为 $n$ ，初始归并段的个数为 $m$ ，

从 $k$ 个记录中选择一个关键字最小的记录时的比较次数为 $c$

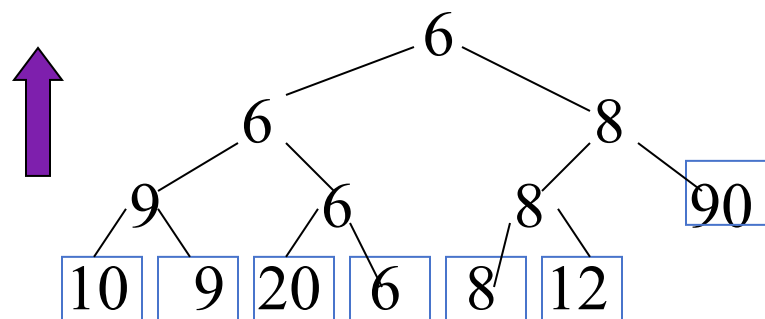
矛盾  $\begin{cases} k \uparrow, s \downarrow, \text{可减少I/O次数;} \\ k \uparrow, c=(k-1) \uparrow, \text{归并效率} \downarrow \end{cases}$

[解决矛盾的方法]

——利用“败者树”选关键字最小的记录

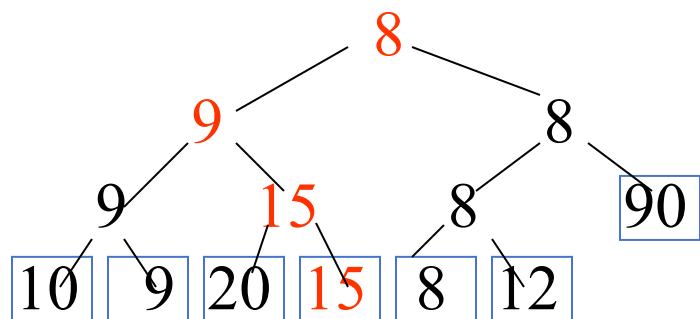


## [胜者树(树形选择排序)的缺陷]



10	9	20	6	8	12	90
14	22	24	15	16	17	92
26	38	30	25	50	18	97
:	:	:	:	:	:	:

7路合并胜者树

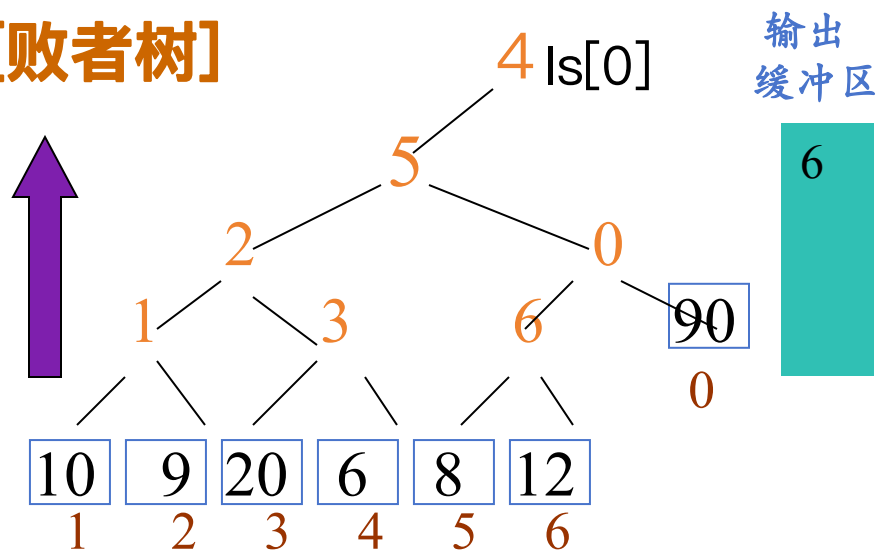


10	9	20	*	8	12	90
14	22	24	15	16	17	92
26	38	30	25	50	18	97
:	:	:	:	:	:	:

重构后的胜者树

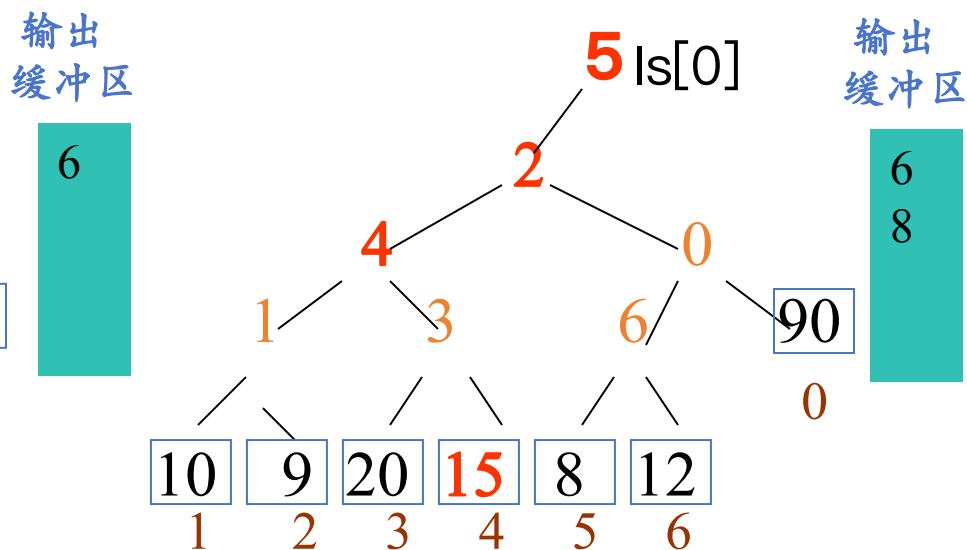
将左图第一轮最小值6取走后，重构胜者树时，需要和兄弟结点比较，然后从根结点至新补入记录的叶结点的路径上的所有结点都必须更新

## [败者树]



10	9	20	6	8	12	90
14	22	24	15	16	11	92
26	38	30	25	50	18	97
:	:	:	:	:	:	:
:	:	:	:	:	:	:

7路合并败者树



10	9	20	*	8	12	90
14	22	24	15	16	11	92
26	38	30	25	50	18	97
:	:	:	:	:	:	:
:	:	:	:	:	:	:


重构后的败者树

败者树的特点: 记录败者, 胜者参加下一轮比赛

败者树的优点: 重构时和父结点比较, 修改结点较少。

**[数据结构]** (依据: 败者树为完全二叉树)

主:  $b[0..k]$

$b[0..k-1]$ —— $k$ 个叶结点  , 存放 $k$ 个输入归并段中当前参加归并的记录

$b[k]$ ——虚拟记录, 用于建败者树  
该关键字取可能的最小值minkey

辅:  $ls[0..k-1]$  ——不含叶结点的败者树

存放最后胜出的编号 ( $ls[0]$ ) 以及所记录的败者编号

**[处理步骤]**

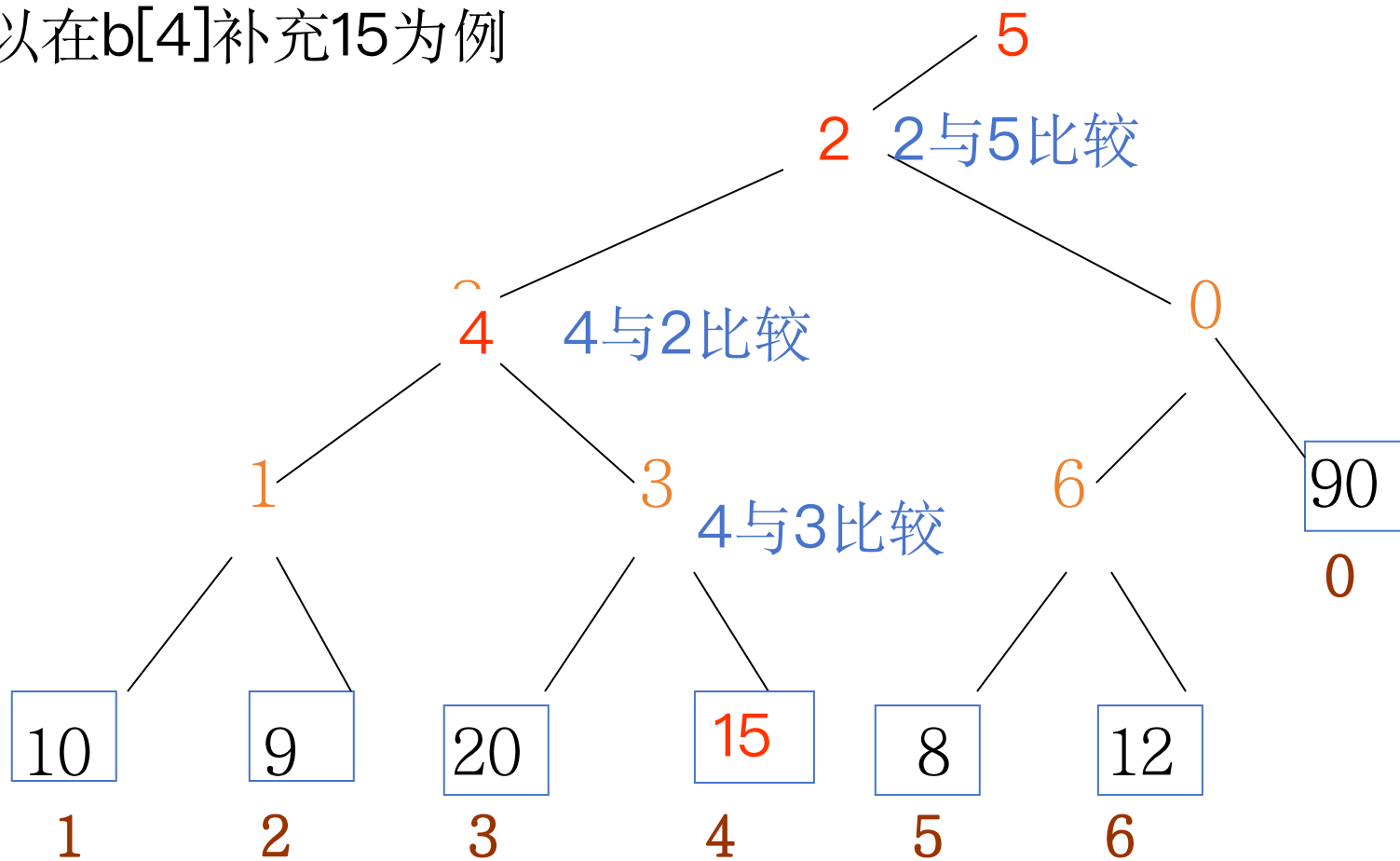
1)建败者树 $ls[0..k-1]$

2)重复直至 $k$ 路归并完毕

2.1)将 $b[ls[0]]$ 写至输出归并段

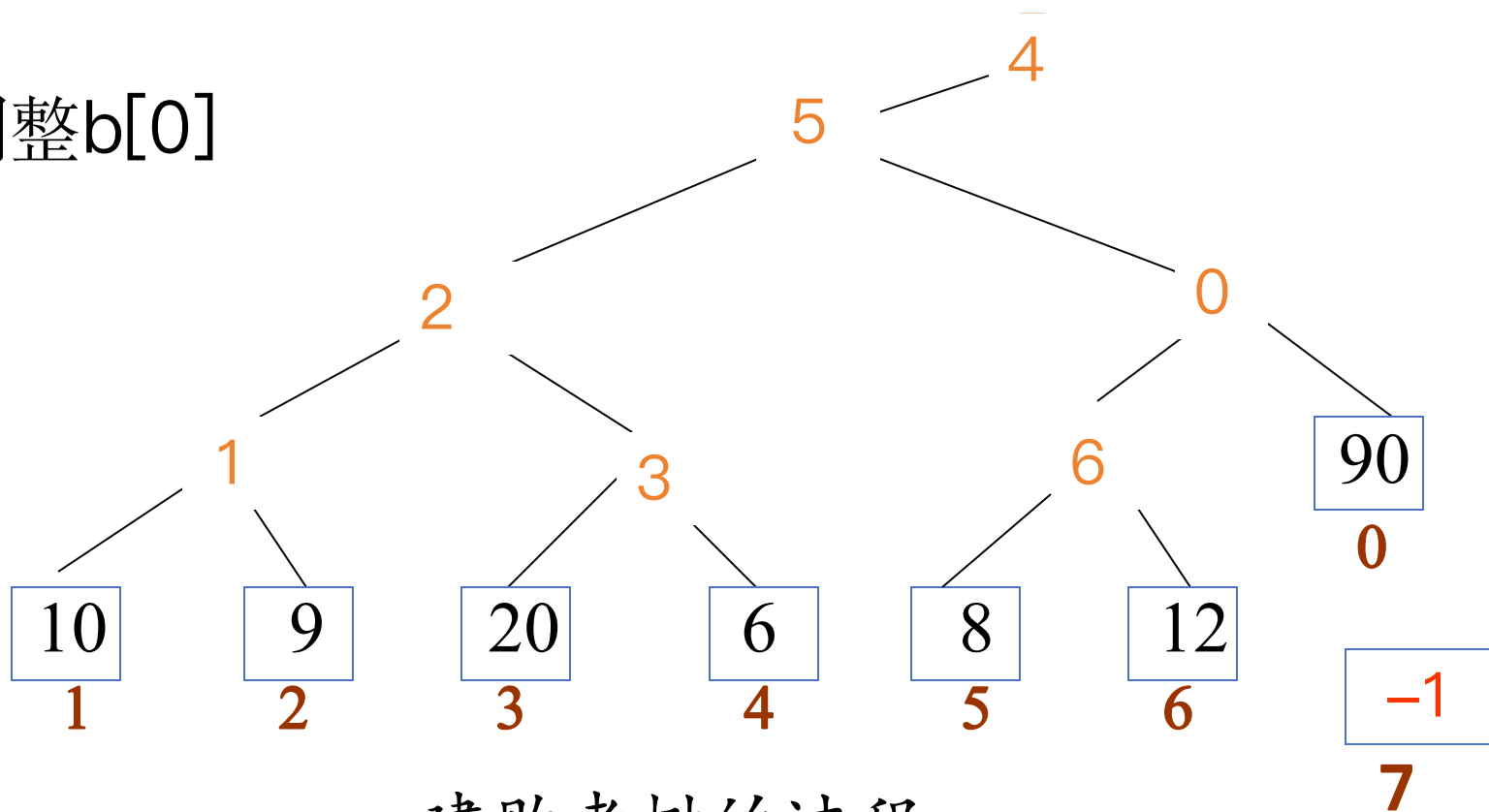
2.2)补充记录(某归并段变空时,补 $\infty$ ), 调整败者树

## 以在b[4]补充15为例



调整败者树的方法: 将新补充的结点与其双亲结点比较, 败者留在该双亲结点, 胜者继续向上直至树根的双亲

调整b[0]



建败者树的过程

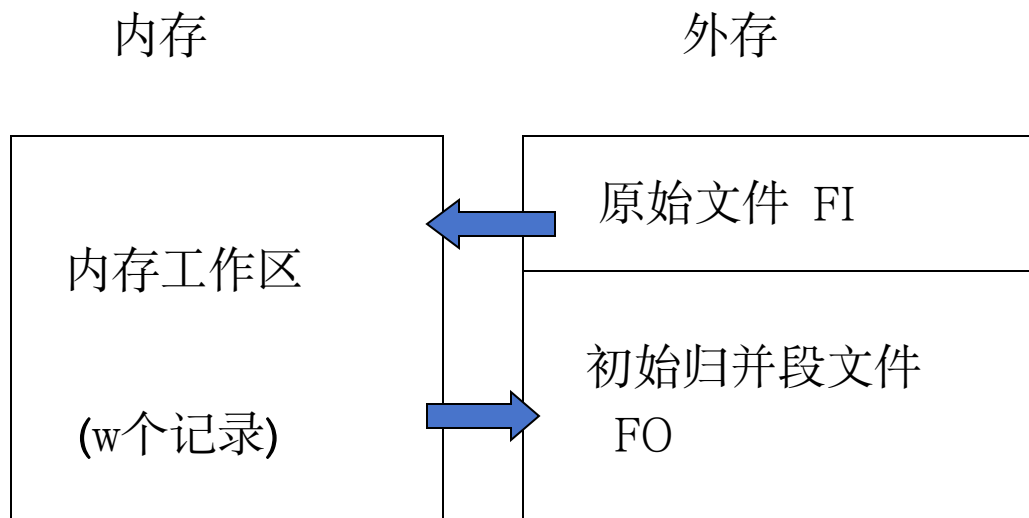
调整败者树的方法: 将调整的结点与其双亲结点比较, 败者留在该双亲结点, 胜者继续向上

## 11.4 置换-选择排序

### [目标]

扩大初始归并段长度，突破内存工作区容量（设 $w$ 个记录）的限制。（即在内存工作区容量的限制下，获得尽可能长的初始归并段）

### [置换-选择排序原理]



为简化问题，设每个物理块存放一个记录

在构造初始归并段时，选择最小（或最大）关键字和输入、输出交叉或平行进行

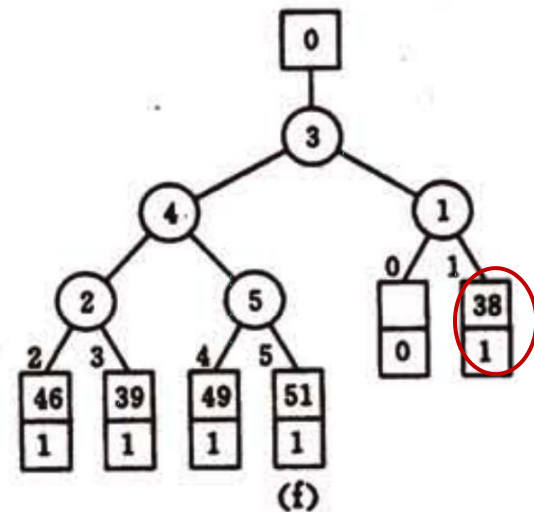
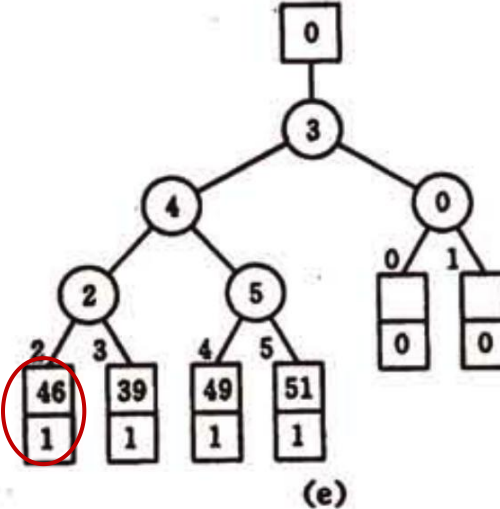
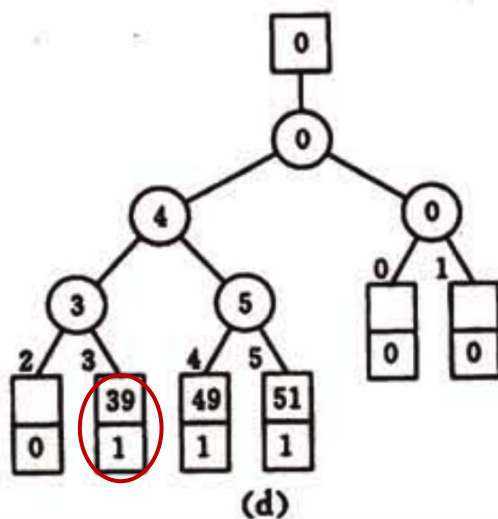
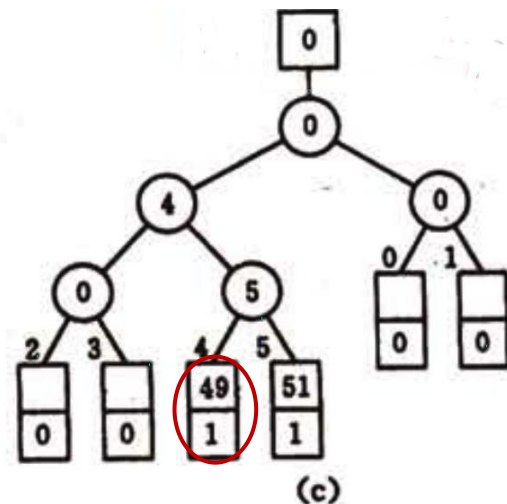
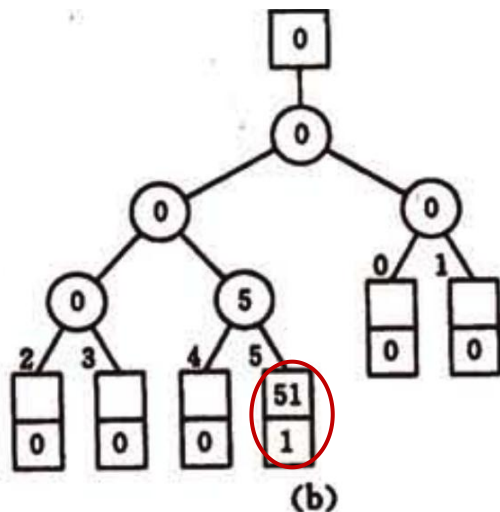
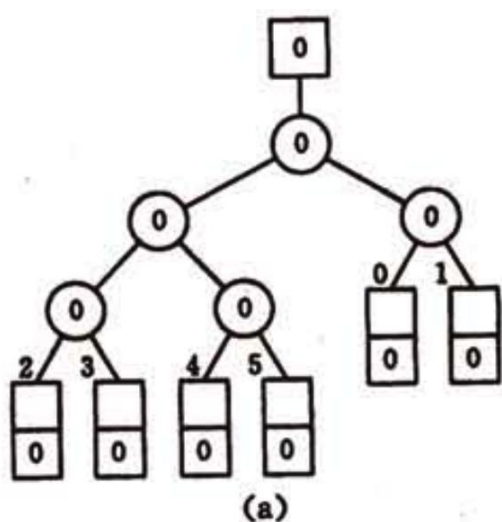


## [示例]

FO	WA(4个记录)	FI
空	空	36,21,33,87,23,7,62,16, 54,43,29,...
空	36,21,33,87	23,7,62,16,54, 43,29,...
21	36,23,33,87	7,62,16,54, 43,29,...
21,23	36,7,33,87	62,16,54,43,29,...
21,23,33	36,7,62,87	16,54,43,29,...
21,23,33,36	16,7,62,87	54,43,29,...
21,23,33,36,62	16,7,54,87	43,29,...
21,23,33,36,62,87	16,7,54,43	29,...
21,23,33,36,62,87  7	16,29,54,43	...
...		

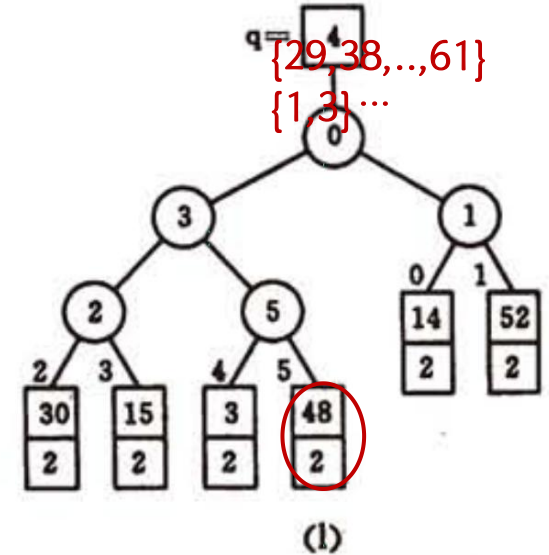
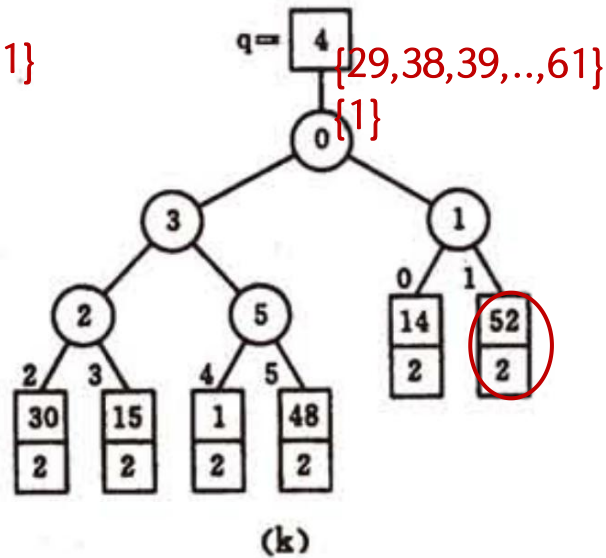
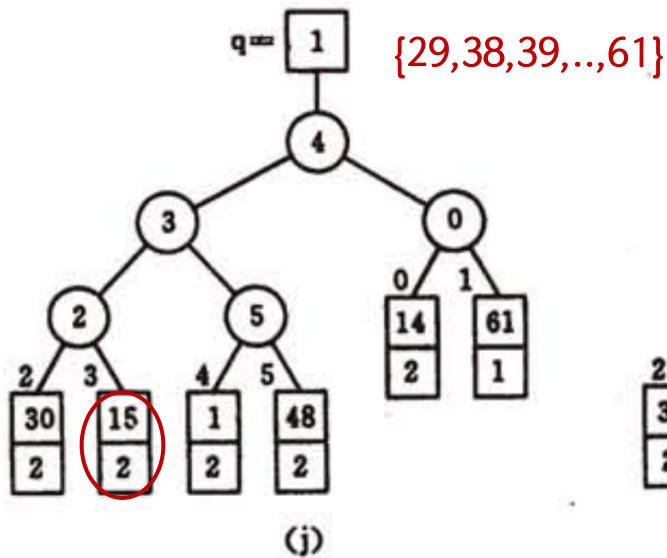
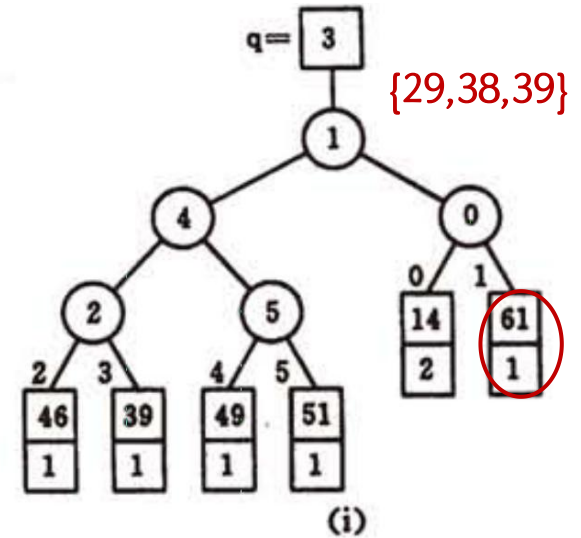
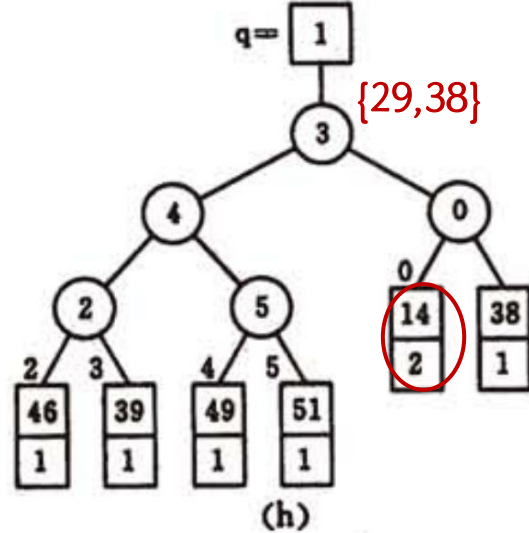
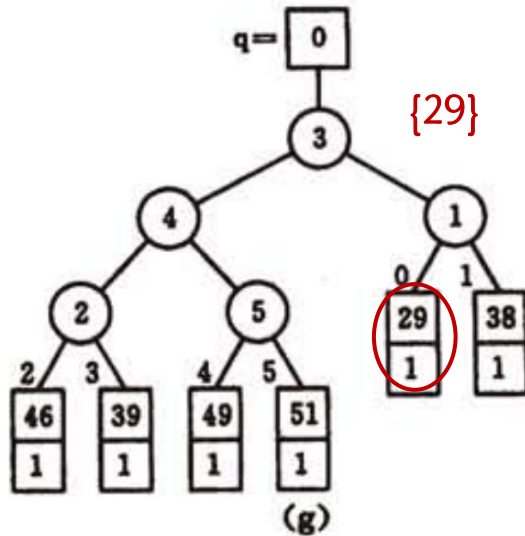
置换-选择排序可通过败者树实现 P<sub>303</sub>图11.6 WA=6

{ 51,49,39,46,38,29,14,61,15,30,1,48,52,3,63,27,4,... }



段号不同，段大为败；段号相同，key大为败

{ 51,49,39,46,38,29,14,61,15,30,1,48,52,3,63,27,4,... }



## [置换-选择排序的效果]

- 所得初始排序段的长度不等
- 当输入文件记录的关键字大小随机分布时，初始归并段的平均长度为内存工作区大小 $w$ 的两倍

## [缓冲区及并行操作]

使输入、输出和内部归并三种操作同时进行，也能提高总的外部排序时间。但应综合考虑缓冲区数目、工作区大小等参数的设置。

内存



# 11.5 最佳归并树

## [最佳归并树的引入]

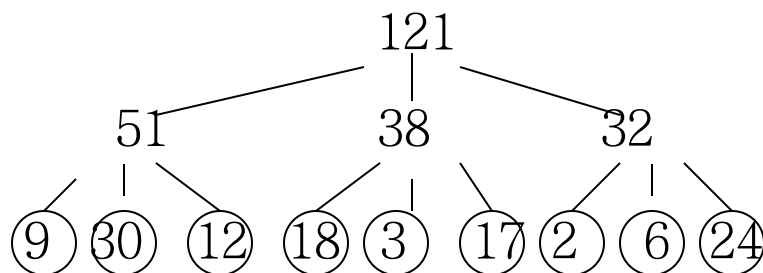
进行k路归并时，初始归并段的不同搭配，会导致归并过程中I/O的次数不同。

设有9个初始归并段，记录个数(长度)分别为：

9, 30, 12, 18, 3, 17, 2, 6, 24

3-路归并

方案一

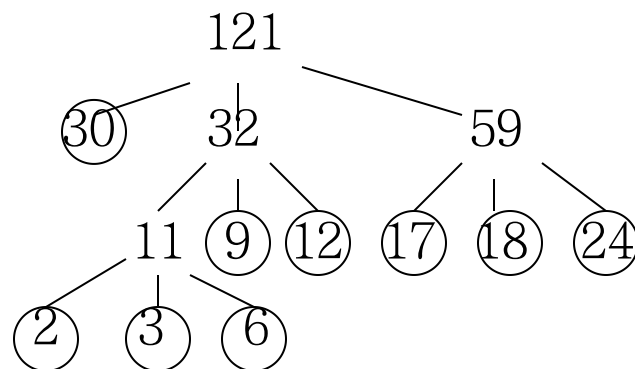


$$(9+30+12+18+3+17+2+6+24) \times 2 \\ + (51+38+32) \times 2 \\ = 484$$

x2因为读写各1次

所有叶结点加权外通路长度的2倍

方案二



$$(2+3+6) \times 2 + (11+9+12+17+18+24) \times 2 + \\ (30+32+59) \times 2 \\ = [(2+3+6) \times 3 + (9+12+17+18+24) \\ \times 2 + 30 \times 1] \times 2 \\ = 446$$

## [最佳归并树的构造]

最佳归并树即 $k$ 叉(阶)哈夫曼树。

设初始归并段为 $m$ 个，进行 $k$ -路归并

1) 若  $(m-1) \bmod (k-1) \neq 0$

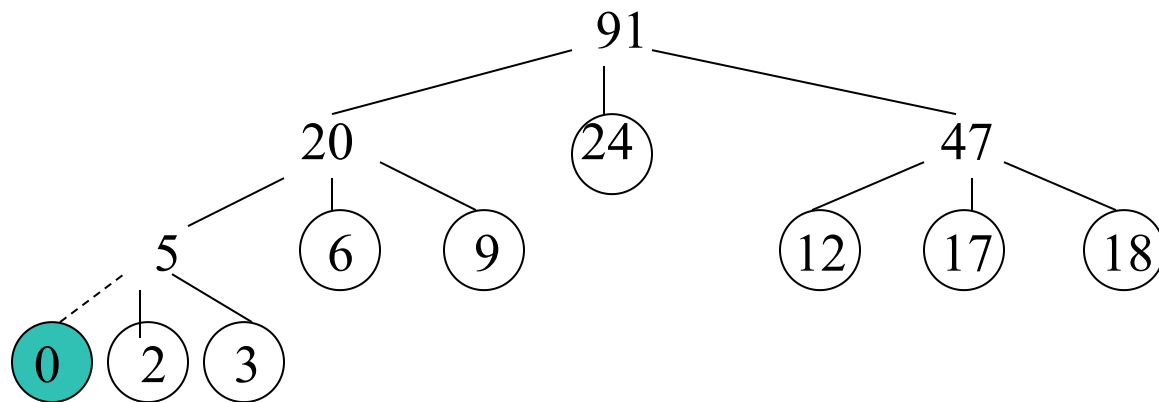
则需附加  $(k-1) - ((m-1) \bmod (k-1))$  个长度为0的虚段，以使每次归并都可以对应 $k$ 个段

2) 按照哈夫曼树的构造原则（权值越小的结点离根结点越远）构造最佳归并树。

## [示例]

长度分别为2, 3, 6, 9, 12, 17, 18, 24的8个初始归并段进行3路归并, 求最佳归并树。

【解】  $m=8$   $k=3$ , 则  $(m-1) \bmod (k-1) = 1$



I/O次数:

$$[(2+3) \times 3 + (6+9+12+17+18) \times 2 + 24 \times 1] \times 2 = 326$$

# 本章学习要点

- 了解外排序需要两个基本步骤
  - 生成初始归并段
  - 多趟归并
- 了解常规外排序时内外存交换次数的计算方法
- 理解提高外排序效率的两个手段是减少初始归并段个数（利用置换-选择排序）和增加归并路数（进行多路归并）
- 掌握k叉哈夫曼树的概念和最佳归并树的构造方法