Chapter 7

文件管理和外排序

在许多实际应用中，特别是数据处理时，都需要长期存储海量数据，这些数据通常以文件的方式组织并存储在外存。如何有效地管理这些数据，从而给使用者提供方便而高效的使用数据的方法称为文件管理。

在实际存取这些海量数据时，为了方便使用，往往以某种顺序排序后再存储在外存上，这种排序称为外部排序。在排序时由于一次不能将数据文件中的所有数据同时装入内存中进行，因此就必须研究如何对外存上的数据进行排序的技术。

7.1 文件的基本概念

1 文件的基本概念

⑴ 数据项(Item或field)：数据文件中最小的基本单位，反映实体某一方面的特征—属性的数据表示。

⑵ 记录(Record)：一个实体的所有数据项的集合。 用来标识一个记录的数据项集合(一个或多个)称为关键字项(Key)，关键字项的值称为关键字；能唯一标识一个记录的关键字称为主关键字(Primary Key)，其它的关键字称为次关键字(Secondary Key)。

文件存储在外存上，通常是以块(I/O读写的基本单位，称为物理记录)存取。

⑶ 文件(File)：大量性质相同的数据记录的集合。文件的所有记录是按某种排列顺序呈现在用户面前，这种排列顺序可以是按记录的关键字，也可以是按记录进入文件的先后等。则记录之间形成一种线性结构(逻辑上的)，称为文件的逻辑结构；文件在外存上的组织方式称为文件的物理结构。

基本的物理结构有：顺序结构，链接结构，索引结构 。

⑷ 文件的分类

⑴ 按记录类型，可分为操作系统文件和数据库文件：

① 操作系统文件(流式文件)： 连续的字符序列(串)的集合；

② 数据库文件： 有特定结构(所有记录的结构都相同)的数据记录的集合。

⑵ 按记录长度，可分为定长记录文件和不定长记录文件：

① 定长记录文件：文件中每个记录都有固定的数据项组成，每个数据项的长度都是固定的；

② 不定长记录文件：与定长记录文件相反。

2 文件的有关操作

文件是由大量记录组成的线性表，因此，对文件的操作主要是针对记录的，通常有：记录的检索、插入、删除、修改和排序，其中检索是最基本的操作。

⑴ 检索记录

根据用户的要求从文件中查找相应的记录。

① 查找下一个记录：找当前记录的下一个逻辑记录；

② 查找第k个记录：给出记录的逻辑序号，根据该序号查找相应的记录；

③ 按关键字查找：给出指定的关键字值，查找关键字值相同或满足条件的记录。

对数据库文件，有以下四种按关键字查找的方式：

- 简单匹配：查找关键字的值与给定的值相等的记录；
- 区域匹配：查找关键字的值在某个区域范围内的记录；
- 函数匹配：给出关键字的某个函数，查找符合条件的记录；
- 组合条件匹配：给出用布尔表达式表示的多个条件组合，查找符合条件的记录。

(2) 插入记录

将给定的记录插入到文件的指定位置。插入是首先要确定插入点的位置(检索记录)，然后才能插入。

(3) 删除记录

从文件中删除给定的记录。记录的删除有两种情况：

① 在文件中删除第k个记录；

② 在文件中删除符合条件的记录。

(4) 修改记录

对符合条件的记录，更改某些属性值。修改时首先要检索到所要修改的记录，然后才能修改。

(5) 记录排序

根据指定的关键字，对文件中的记录按关键字值的大小以非递减或非递增的方式重新排列(或存储)。

7.2 文件的组织方式

文件的组织方式指的是文件的物理结构。

7.2.1 顺序文件

记录按其在文件中的逻辑顺序依次进入存储介质。在顺序文件中，记录的逻辑顺序和存储顺序是一致的。

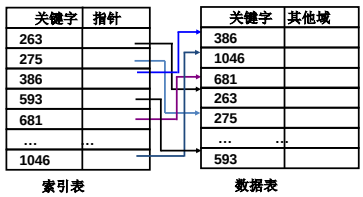
(1) 根据记录是否按关键字排序：可分为排序顺序文件和一般顺序文件；

(2) 根据逻辑上相邻的记录的物理位置关系：可分为连续顺序文件和链接顺序文件。

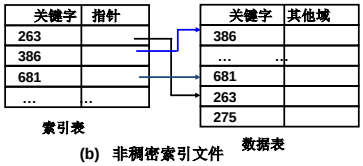
顺序文件类似于线性表的顺序存储结构，比较简单，适合于顺序存取的外存介质，但不适合随机处理。

7.2.2 索引文件

索引技术是组织大型数据库的一种重要技术，索引是记录和记录存储地址之间的对照表。索引结构(称为索引文件)由索引表和数据表两部分，如图7-1所示。



(a) 稠密索引文件



(b) 非稠密索引文件

图7-1 索引结构的基本形式

- 数据表：存储实际的数据记录；
- 索引表：存储记录的关键字和记录(存储)地址之间的对照表，每个元素称为一个索引项。

如果数据文件中的每一个记录都有一个索引项，这种索引称为稠密索引，否则，称为非稠密索引。

对于非稠密索引，通常将文件记录划分为若干块，块内记录可以无序，但块间必须有序。若块内记录是有序的，称为索引顺序文件，否则称为索引非顺序文件。对于索引非顺序文件，只需对每一块建立一个索引项。

对于稠密索引，可以根据索引项直接查找到记录的位置。若在索引表中采用顺序查找，查找时间复杂度为$O(n)$；若采用折半查找，查找时间复杂度为$O(\log_2 n)$。

对于稠密索引，索引项数目与数据表中记录数相同，当索引表很大时，检索记录需多次访问外存。

对于非稠密索引，查找的基本思想是：

首先根据索引找到记录所在块，再将该块读入到内存，然后再在块内顺序查找。

平均查找长度由两部分组成：块地址的平均查找长度L_b，块内记录的平均查找长度L_w，即$ASL_{bs}=L_b+L_w$

若将长度为n的文件分为b块，每块内有s个记录，则$b=n/s$。设每块的查找概率为$1/b$，块内每个的记录查找概率为$1/b$，则采用顺序查找方法时有：

$$ASL_{bs}=L_b+L_w=(b+1)/2+(s+1)/2=(n/s+s)+1$$

显然，当$s=n^{1/2}$时，ASL_{bs}的值达到最小；

若在索引表中采用折半查找方法时有：

$$ASL_{bs}=L_b+L_w=\log_2(n/s+1)+s/2$$

如果文件中记录数很庞大，对非稠密索引而言，索引也很大，可以将索引表再分块，建立索引的索引，形成树形结构的多级索引，如后面将要介绍的ISAM文件和VSAM文件。

7.2.3 ISAM文件

ISAM(Indexed Sequential Access Method，顺序索引存取方法)，是专为磁盘存取设计的一种文件组织方式，采用静态索引结构，是一种三级索引结构的顺序文件。图7-2是一个磁盘组的结构图。

ISAM文件由基本文件、磁道索引、柱面索引和主索引组成。

基本文件按关键字的值顺序存放，首先集中存放在同一柱面上，然后再顺序存放在相邻柱面上；对于同一柱面，则按盘面的次序顺序存放。

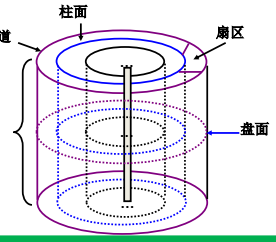
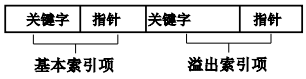


图7-2 一个磁盘组结构形式

在每个柱面上，还开辟了一个溢出区，存放从该柱面的磁道上溢出的记录。同一磁道上溢出的记录通常由指针相链接。

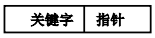
ISAM文件为每个磁道建立一个索引项，相同柱面的磁道索引项组成一个索引表，称为磁道索引，由基本索引项和溢出索引项组成，其结构是：



◆ 基本索引项：关键字域存放该磁道上的最大关键字；指针域存放该磁道的首地址。

◆ 溢出索引项：是为插入记录设置的。关键字域存放该磁道上溢出的记录的最大关键字；指针域存放溢出记录链表的头指针。

在磁道索引的基础上，又为文件所占用的柱面建立一个柱面索引，其结构是：



关键字域存放该柱面上的最大关键字；指针域指向该柱面的第1个磁道索引项。

当柱面索引很大时，柱面索引本身占用很多磁道，又可为柱面索引建立一个主索引。则ISAM文件的三级索引结构如图7-3所示。

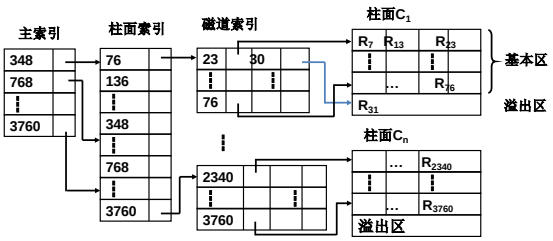


图7-3 ISAM文件结构示意图

1 ISAM文件的检索

根据关键字查找时，首先从主索引中查找记录所在的柱面索引块的位置；再从柱面索引块中查找磁道索引块的位置；然后再从磁道索引块中查找出该记录所在的磁道位置；最后从磁道中顺序查找要检索的记录。

2 记录的插入

首先根据待插入记录的关键字查找到相应位置；然后将该磁道中插入位置及以后的记录后移一个位置(若溢出，将该磁道中最后一个记录存入同一柱面的溢出区，并修改磁道索引)；最后将记录插入到相应位置。

3 记录的删除

只需找到要删除的记录，对其做删除标记，不移动记录。当经过多次插入和删除操作后，基本区有大量被删除的记录，而溢出区也可能有大量记录，则周期性地整理ISAM文件，形成一个新的ISAM文件。

4 ISAM文件的特点

- 优点：节省存储空间，查找速度快；
- 缺点：处理删除记录复杂，多次删除后存储空间的利用率和存取效率降低，需定期整理ISAM文件。

7.2.4 VSAM文件

VSAM(Virtual Storage Access Method，虚拟存取方法)，也是一种索引顺序文件组织方式，利用OS的虚拟存储器功能，采用的是基于B+树的动态索引结构。

文件的存取不是以柱面、磁道等物理空间为存取单位，而是以逻辑空间——控制区间(Control Interval)和控制区域(Control Range)为存取单位。

一个VSAM文件由索引集、顺序集和数据集组成，如图7-4所示。

文件的记录都存放在数据集中，数据集又分成多个控制区间；VSAM进行I/O操作的基本单位是控制区间，由一组连续的存储单元组成，同一文件的控制区间大小相同；

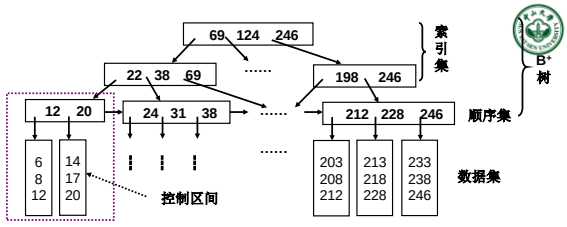


图7-4 VSAM文件结构示意图

每个控制区间存放一个或多个逻辑记录，记录是按关键字值顺序存放在控制区间的前端，尾端存放记录的控制信息和控制区间的控制信息，如图7-5所示。

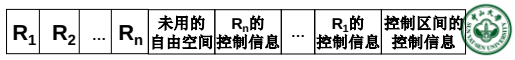


图7-5 控制区间的结构

顺序集是由B+树索引结构的叶子结点组成。每个结点存放若干个相邻控制区间的索引项，每个索引项存放一个控制区间中记录的最大关键字值和指向该控制区间的指针。顺序集中的每个结点及与它所对应的全部控制区间组成一个控制区域。

顺序集中的结点之间按顺序链接成一个链表，每个结点又在其上层建立索引，并逐层向上按B+树的形式建立多级索引。则顺序集中的每一个结点就是B+树的叶子结点；在顺序集之上的索引部分称为索引集。

在VSAM文件上既可以按B+树的方式实现记录的查找，又可以利用顺序集索引实现记录顺序查找。

VSAM文件中没有溢出区，解决方法是留出空间：

- 每个控制区间中留出空间；
- 每个控制区域留出空的控制空间，并在顺序集的索引中指出。

1 记录的插入

首先根据待插入记录的关键字查找到相应的位置：

- 若该控制区间有可用空间：将关键字大于待插入记录的关键字的记录全部后移一个位置，在空出的位置存放待插入记录；
- 若控制区间没有可用空间：利用同一控制区域的一个空白控制空间进行区间分裂，将近一半记录移到新的控制区间中，并修改顺序集中相应的索引，插入新的记录；

- 若控制区域中没有空白控制空间：则开辟一个新的控制区域，进行控制区间域分裂和相应的顺序集中的结点分裂。也可按B^+树的分裂方法进行。

2 记录的删除

先找到要删除的记录，然后将同一控制区间中比删除记录关键字大的所有记录逐个前移，覆盖要删除的记录。当一个控制区间的记录全部删除后，需修改顺序集中相应的索引项。

3 VSAM文件的特点

(1) 优点

- 能动态地分配和释放空间；

- 能保持较高的查询效率，无论是查询原有的还是后插入的记录，都有相同的查询速度；
- 能保持较高的存储利用率(平均75%)；
- 永远不需定期整理文件或对文件进行再组织。

(2) 缺点

- 为保证具有较好的索引结构，在插入或删除时索引结构本身也在变化；
- 控制信息和索引占用空间较多，因此，VSAM文件通常比较庞大。

基于B^+树的VSAM文件通常被作为大型索引顺序文件的标准。

7.2.5 散列文件

散列文件(直接存取文件)：利用散列存储方式组织的文件。类似散列表，即根据文件中记录关键字的特点，设计一个散列函数和冲突处理方法，将记录散列到存储介质上。

在散列文件中，磁盘上的记录是成组存放的，若干个记录组成一个存储单位，称为桶(Bucket)，同一个桶中的记录都是同义词(关键字的角度)。

设一个桶中能存放m个记录，当桶中已有m个同义词的记录时，要存放第m+1个同义词就“溢出”。冲突处理方法一般是拉链法。

检索记录时，先根据给定值求出散列桶地址，将基桶的记录读入内存进行顺序查找，若找到关键字等于给定值的记录，则查找成功；否则，依次读入各溢出桶中的记录继续进行查找。

在散列文件中删除记录，是对记录加删除标记。

散列文件的特点

(1) 优点

- 文件随机存取，记录不需进行排序；
- 插入、删除方便，存取速度快；
- 不需要索引区，节省存储空间。

(2) 缺点

- 不能进行顺序存取，只能按关键字随机存取；
- 检索方式仅限于简单查询。

7.2.6 多关键字文件

数据库文件常常是多关键字文件，多关键字文件的特点是不仅可以对主关键字进行各种查询，而且可以对次关键字进行各种查询。因此，对多关键字文件除了可按前面的方法组织主关键字索引外，还需要建立各个次关键字的索引。由于建立次关键字的索引的结构不同，多关键字文件有多重表文件和倒排文件。

1 多重表文件

多重表文件(Multilist Files)的特点是：记录按主关键字的顺序构成一个串联文件(物理上的)，并建立主关键字索引(称为主索引)；对每个次关键字都建立次关键字索引(称为次索引)，所有具有同一次关键字值的记录构成一个链表(逻辑上的)。

主索引一般是非稠密索引，其索引项一般有两项：主关键字值、头指针。

次索引一般是稠密索引，其索引项一般有三项：次关键字值、头指针、链表长度。头指针指向数据文件中具有该次关键字值的第1个记录，在数据文件中为各个次关键字增加一个指针域，指向具有相同次关键字值的下一个记录的地址。

对于任何次关键字的查询，都应首先查找对应的索引，然后顺着相应指针所指的方向查找属于本链表的记录。

多重表文件的特点

(1) 优点

易于构造和修改、查询方便。

(2) 缺点

插入和删除一个记录时，需要修改多个次关键字的指针(在链表中插入或删除记录)，同时还要修改各索引中的有关信息。

2 倒排文件

倒排文件又称逆转表文件。与多重表文件类似，可以处理多关键字查询。其差别是：

- 多重表文件：将具有相同关键字值的记录链接在一起，在数据文件中设有与各个关键字对应的指针域；
- 倒排文件：将具有相同关键字值的记录的地址收集在一起，并保存到相应的次关键字的索引项中，在数据文件中不设置对应的指针域。

次索引是次关键字倒排表，倒排表由次关键字值、记录指针(地址)，索引中保持次关键字的逻辑顺序。

倒排表文件的特点

(1) 优点

检索速度快，插入和删除操作比多重表文件简单。当插入一个记录时，只要将记录存入数据文件，并将其存储地址加入各倒排表中；删除也很方便。

(2) 缺点

倒排表维护比较困难。在同一索引表中，不同关键字值的记录数目不同，同一倒排表中的各项长度不等。

7.3 外存信息的存取

- 常用的外存储器分类：
  - 顺序存取的设备（如磁带）；
  - 随机存取的设备（如磁盘）。

常用的外存储器是磁表面存储器，信息记录在一薄层磁性材料的表面上，这层材料附着于载体表面，随着载体作高速旋转或直线运动，在运动过程中用磁头进行读或写。

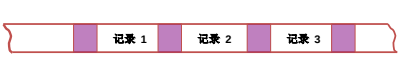
外存信息的存取特点，决定了外部排序的策略选择。

- 磁带信息的存取

磁带存储器的工作原理和磁带录音机一样，不同之处在于它存储的是数字信息而不是模拟信息。

磁带上的信息在横向分布、纵向分布以及首尾标志等都一定的格式。

- 以1/2英寸九道的磁带为例，每一横排就可表示一个字符（8位+1个校验位）。
- 纵向按区进行存储，区的长度不固定，但有一个范围，例如2到4096字节，相邻区之间有一定长度的间隔（IBG， Inter Block Gap），作为磁带起停之用。



- 在磁带上读写一块信息所需的时间由两部分组成：

$$T_{I/O} = t_a + n\ t_w$$

  - t_a为延迟时间，即读/写头到达传输信息所在物理快起始位置所需时间；
  - t_w为传输一个字符的时间。

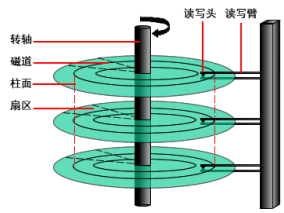
显然，延迟时间和信息在磁带上的位置、当前读/写头所在的位置有关。

所以磁带便宜、可反复使用、是一种顺序存取设备，但查找费时、速度慢（尤其是查找末端记录时）。

- 磁盘信息的存取

磁盘是一种直接存取的存储设备(DASD）。

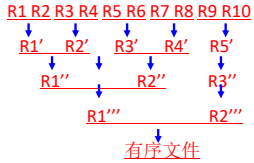
- 页块的读写时间：$T_{I/O} = t_{seek} + t_{latency} + n\ t_{wm}$
  - t_{seek}：寻道时间
  - $t_{latency}$：等待时间
  - t_{wm}：传输时间



7.4 外部排序的方法

- 外部排序基本上由两个相对独立的阶段组成：
  - 首先，按可用内存大小，将外存上含n个记录的文件分成若干长度为 l 的子文件或段(segment)，依次读入内存并利用有效的内部排序方法对它们进行排序，并将排序后得到的有序子文件重新写入外存，通常称这些有序子文件为归并段或顺串(run)；
  - 然后，对这些归并段进行逐躺归并，使归并段（有序的子文件）逐渐由小至大，直到得到整个有序文件为止。

- 例：一文件含10000记录，通过10次内部排序得到10个初始归并段R1…R10，其中每一段都含有1000个记录。然后作两两归并：



- 由10个初始归并段到一个有序文件，共进行了4趟归并，每一趟都从m个归并段得到ceil(m/2)个归并段。这种归并方法称为2-路平衡归并。

- 外存上信息的读/写是以“物理块”为单位进行的，假设每个物理块可以容纳200个记录，则每一趟归并需进行50次“读”和50次“写”，4趟归并加上内部排序时所需进行的读/写使得在外排序中总共需进行500次读/写。

- 外部排序所需总的时间
  = 内部排序（产生初始归并段）所需的时间($m \times t_{IS}$)
  + 外存信息读写的时间($d \times t_{IO}$)+内部归并所需的时间($s \times ut_{mg}$)
- 其中：
  t_{IS} 是为得到一个初始归并段进行内部排序所需时间的均值；
  t_{IO} 是进行一次外存读/写时间的均值；
  ut_{mg} 是对u个记录进行内部归并所需时间；
  m 为经过内部排序之后得到的初始归并段的个数；
  s 为归并的趟数；
  d 为总的读/写次数。
- 于是上例进行外排序所需总的时间为：$10\,t_{IS} + 500\,t_{IO} + 4\text{x}10000\,t_{mg}$
  显然t_{IO}较t_{mg}要大的多，因此提高外排序的效率应主要着眼于减少外存信息读写的次数d。

7.5 多路平衡归并的实现

- 对于2路归并，令两个归并段上有u个记录，每得到归并后的一个记录，仅需一次比较即可，因此得到含u个记录的归并段需进行$u-1$次比较。
- 对于k路归并，令u个记录分布在k个归并段上，显然，归并后的第一个记录应是k个归并段中关键字最小的记录，这需要进行$k-1$次比较，得到u个记录的归并段，共需
- $(u-1)(k-1)$次比较。

- 由此，对n个记录的文件进行外排序时，在内部归并过程中进行的总的比较次数为$s(k-1)(n-1)$。假设所得初始归并段为m个，则归并过程中进行比较的总的时间为：
  $\text{floor}(\log_k m)(k-1)(n-1)t_{mg} = \text{floor}(\log_2 m/\log_2 k)(k-1)(n-1)t_{mg}$
- 由于$(k-1)/\log 2k$ 随 k 的增长而增长，这将抵消由于增大k而减少外存信息读写时间所得效益。

7.6 置换-选择排序

- 归并的趟数不仅和k成反比，也和m成正比，因此减少m是减少s的另一条途径。这里m是外部文件经过内部排序之后得到的初始归并段的个数，$m=\text{ceil}(n/l)$。

  若要减少m，就需要增加l，但是内存的容量有限，利用上一章内排序的方法无法做到，所以必须探索新的排序方法。

- 置换-选择排序(Replacement-Selection Sorting)是在树形选择排序的基础上得来的，它的特点是：在整个排序（得到所有初始归并段）的过程中，选择最小（或最大）关键字和输入、输出交叉或平行进行。

- 假设初始待排文件为输入文件FI，初始归并段文件为输出文件FO，内存工作区为WA，FO和WA的初始状态为空，并设内存工作区WA的容量为w个记录，则置换-选择排序的操作过程为：
  1. 从FI输入w个记录到工作区WA；
  2. 从WA中选出其中关键字最小值的记录，记为MINIMAX记录；
  3. 将MINIMAX记录输出到FO中去；
  4. 若FI不空，则从FI输入下一个记录到WA中；
  5. 从WA中所有关键字比MINIMAX记录的关键字大的记录中选出最小关键字记录，作为新的MINIMAX记录；
  6. 重复3-5，直至在WA中选不出新的MINIMAX记录为止，由此得到一个初始归并段，输出一个归并段的结束标志到FO中去；
  7. 重复2-6，直至WA为空，由此得到全部初始归并段。

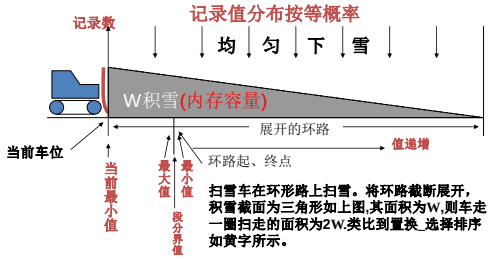
- 例如：初始文件含24个记录，关键字分别为：
  51, 49, 39, 46, 38, 29, 14, 61, 15, 30, 1, 48, 52, 3, 63, 27, 4, 13, 89, 24, 46, 58, 33, 76
  假设内存工作区可容纳6个记录，则用内排序的方法可以得到4个初始段：
  - RUN1：29, 38, 39, 46, 49, 51
  - RUN2： 1, 14, 15, 30, 48, 61
  - RUN3： 3, 4, 13, 27, 52, 63
  - RUN3：24, 33, 46, 58, 76, 89

  而用置换-选择排序，则可求得如下3个初始归并段：
  - RUN1：29, 38, 39, 46, 49, 51, 61
  - RUN2： 1, 3, 14, 15, 27, 30, 48, 52, 63, 89
  - RUN3： 4, 13, 24, 33, 46, 58, 76

- 置换-选择排序的过程：

FO	WA	FI
		51, 49, 39, 46, 38, 29, 14, 61, 15, 30, 1, 48, 52, 3, 63, 27, 4, 13, 89, 24, 46, 58, 33, 76
	51, 49, 39, 46, 38, 29	14, 61, 15, 30, 1, 48, 52, 3, 63, 27, 4, 13, 89, 24, 46, 58, 33, 76
29	51, 49, 39, 46, 38, 14	61, 15, 30, 1, 48, 52, 3, 63, 27, 4, 13, 89, 24, 46, 58, 33, 76
29, 38	51, 49, 39, 46, 61, 14	15, 30, 1, 48, 52, 3, 63, 27, 4, 13, 89, 24, 46, 58, 33, 76
29, 38, 39	51, 49, 15, 46, 61, 14	30, 1, 48, 52, 3, 63, 27, 4, 13, 89, 24, 46, 58, 33, 76
29, 38, 39, 46	51, 49, 15, 30, 61, 14	1, 48, 52, 3, 63, 27, 4, 13, 89, 24, 46, 58, 33, 76
29, 38, 39, 46, 49	51, 1, 15, 30, 61, 14	48, 52, 3, 63, 27, 4, 13, 89, 24, 46, 58, 33, 76
29, 38, 39, 46, 49, 51	48, 1, 15, 30, 61, 14	52, 3, 63, 27, 4, 13, 89, 24, 46, 58, 33, 76
29, 38, 39, 46, 49, 51, 61	48, 1, 15, 30, 52, 14	3, 63, 27, 4, 13, 89, 24, 46, 58, 33, 76
...	...	...

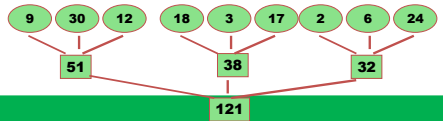
- 可以证明，利用置换-选择排序，初始归并段的平均长度可达内存允许尺寸w的二倍。



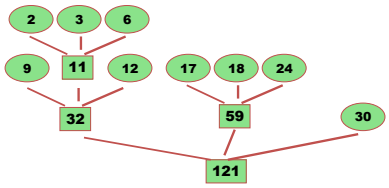
7.7 最佳归并树

- 由置换-选择生成所得的初始归并段，其各段长度不等。
- 假如由置换-选择得到9个初始归并段，其长度分别为：9, 30, 12, 18, 3, 17, 2, 6, 24。作3-路平衡归并（如下图），假设每个记录占一个物理块，则两趟归并所需对外存进行的读/写次数为：

(9+30+12+18+3+17+2+6+24)x2x2=484



- 考虑下图的归并过程，仅需对外存进行446次读/写，这棵归并树为最佳归并树(11+32+59+121)x2，为所有归并策略中所需读/写次数最小的方案。



- 存在有m个叶节点的带权路径长度最短的k叉树，称为霍夫曼树(Huffman)。
- 对长度不等的m个初始段，以其长度为权，构造一棵霍夫曼树作为归并树，便可使得在进行外部归并时所需对外存进行的读/写次数达到最小。