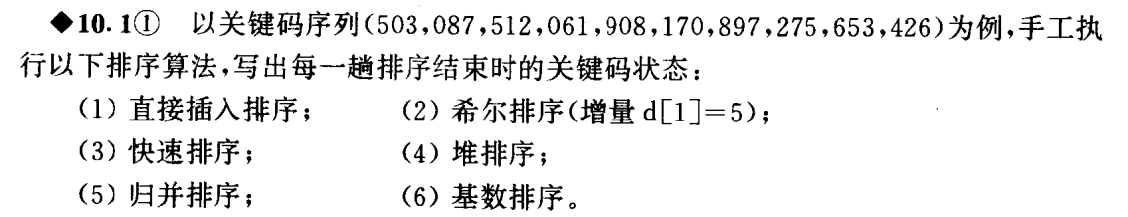
**10.1**



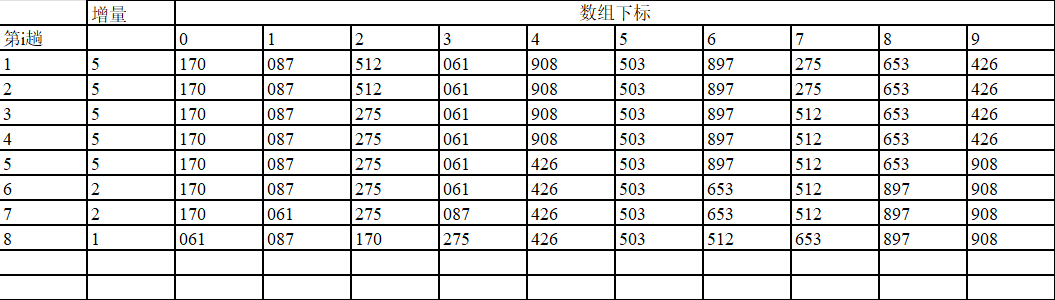
503 087 512 061 908 170 897 275 653 426

10

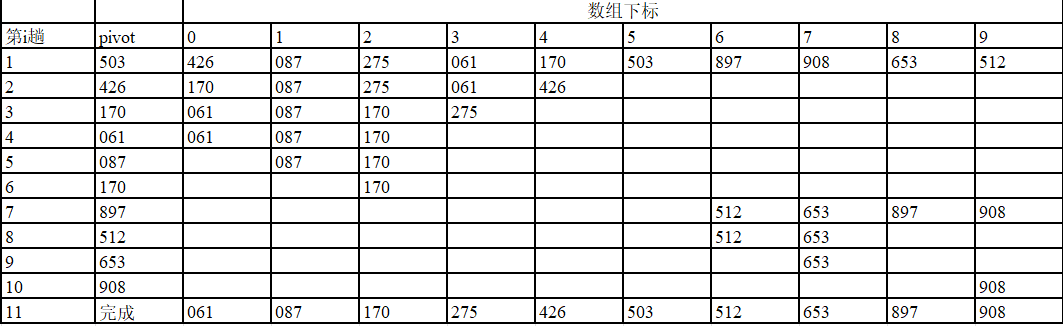
（1）直接插入排序



（2）希尔排序（增量d[l]=5）

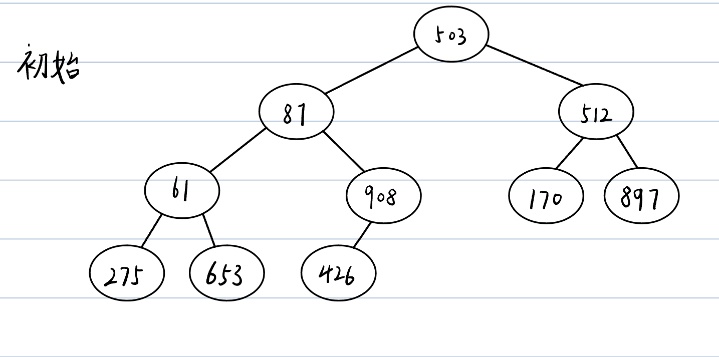


（3）快速排序



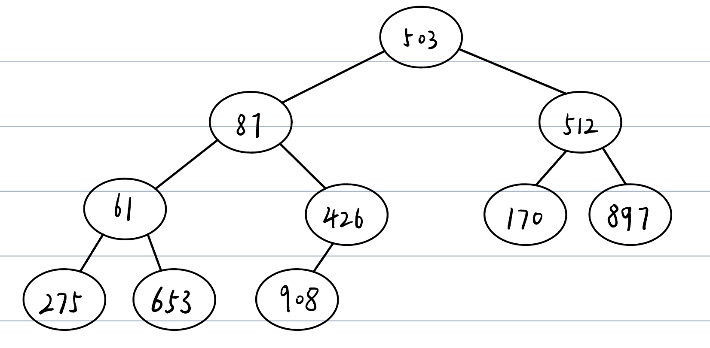
（4）堆排序

初始二叉树如图：n=10

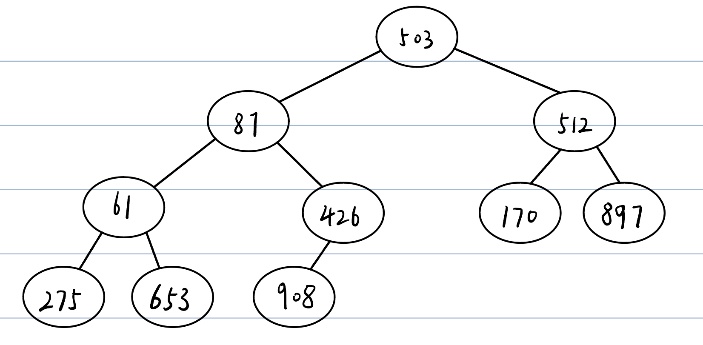


先自底向上将该序列调整为堆

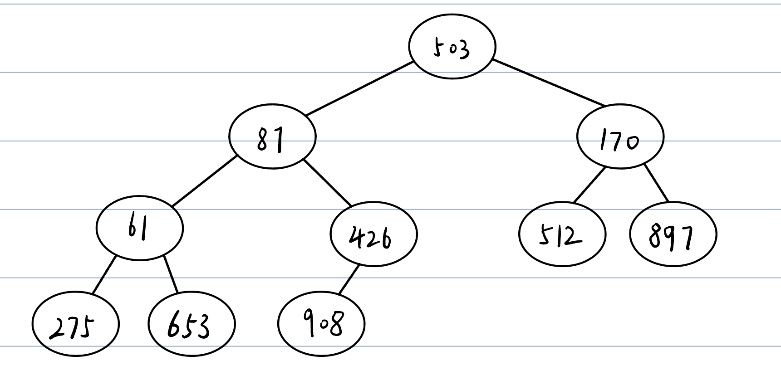
定位最后一个非叶子节点i=(n-2)/2=4, a[4] = 908，从其开始向下调整得：



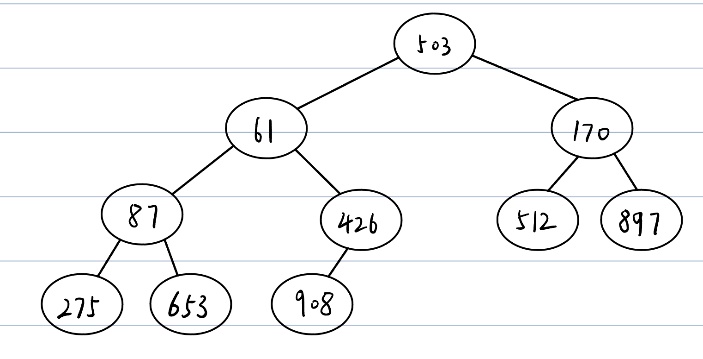
再从a[3]=61开始向下调整得：无需交换



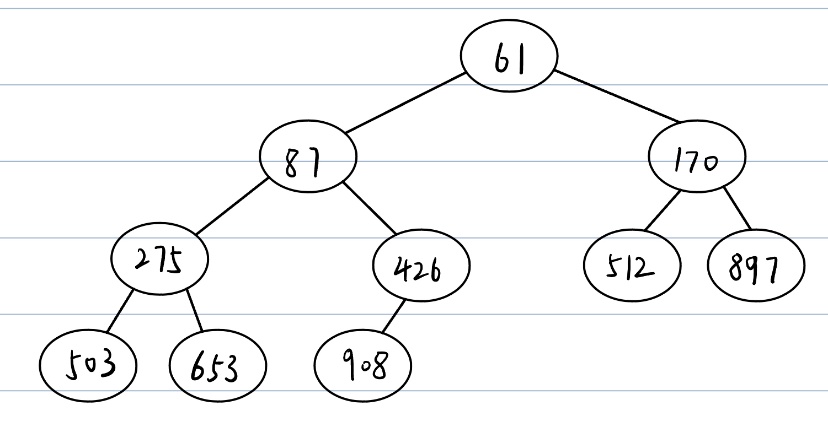
再从a[2]=512开始向下调整得：



再从a[1]=87开始向下调整得：

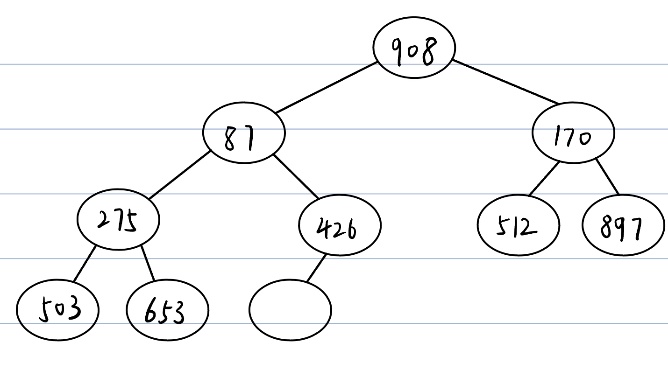


再从a[0]=503开始向下调整得：完成建堆操作

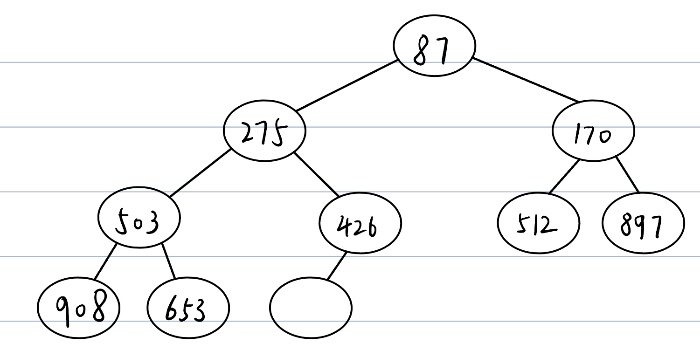


现在开始输出堆顶并调整

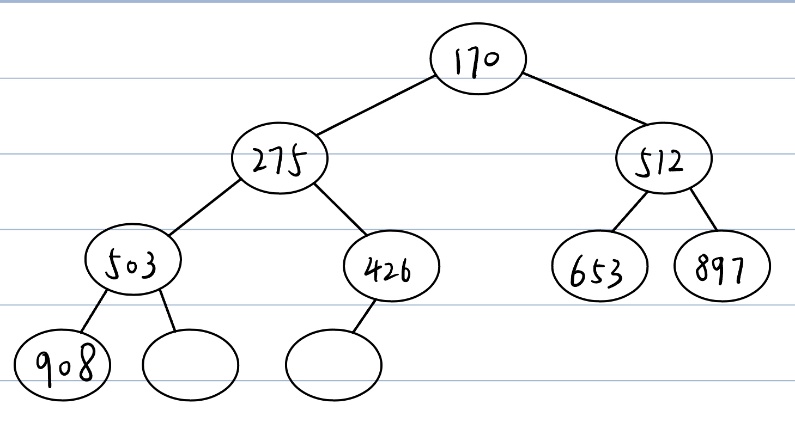
输出61，交换堆顶和908：



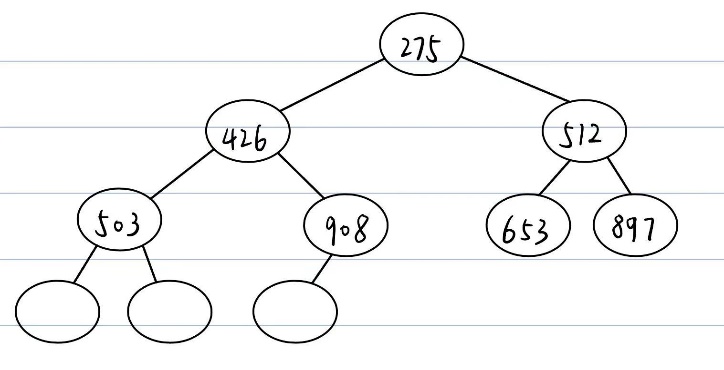
调整后得:



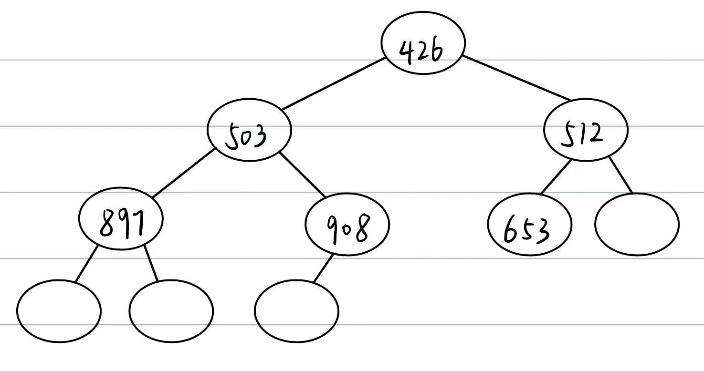
输出87，交换堆顶和653，调整后得：



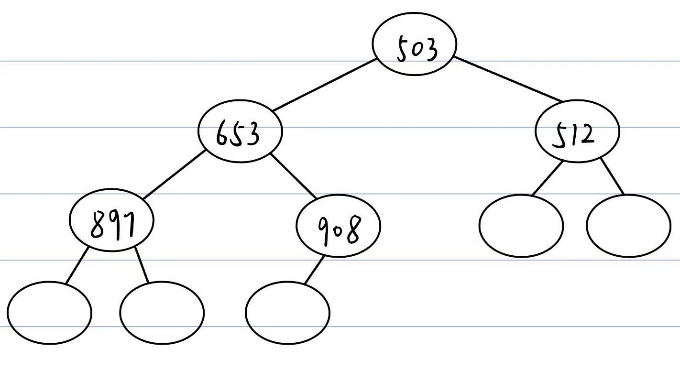
输出170，交换堆顶和908，调整后得：



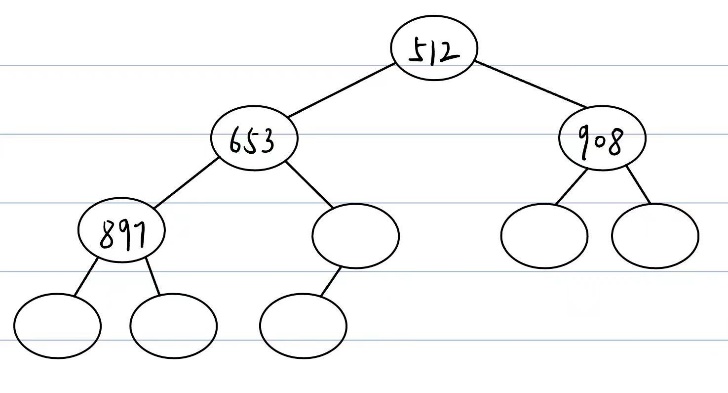
输出275，交换堆顶和897，调整后得：



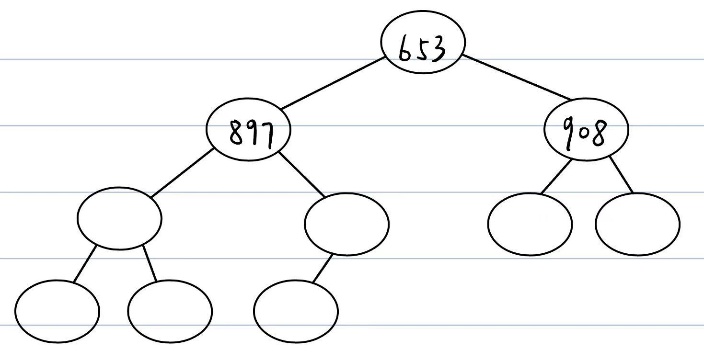
输出426，交换堆顶和653，调整后得：



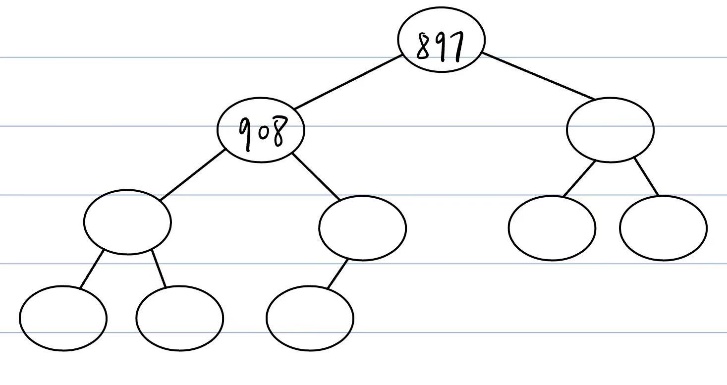
输出503，交换堆顶和908，调整后得：



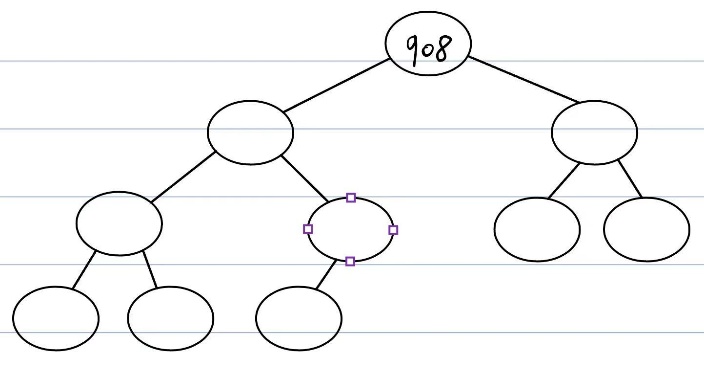
输出512，交换堆顶和897，调整后得：



输出653，交换堆顶和908，调整后得：



输出897，交换堆顶和908，调整后得：



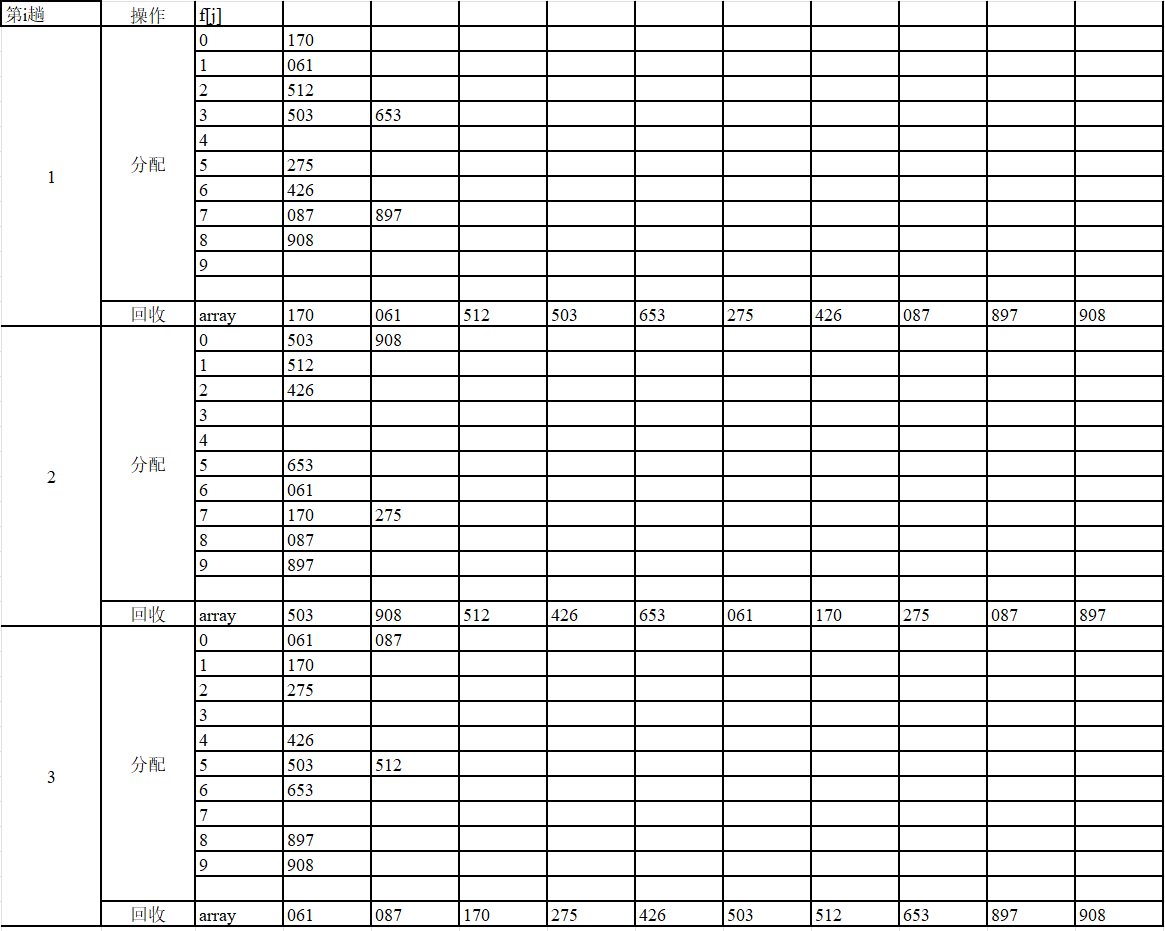
输出908

最终输出序列为061 087 170 275 426 503 512 653 897 908

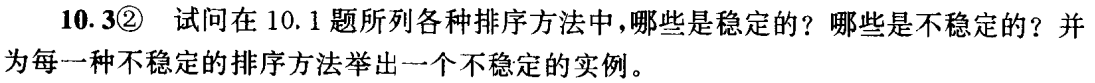
（5）归并排序



（6）基数排序



**10.3**



（1）稳定的：直接插入排序、归并排序、基数排序

不稳定的：希尔排序、快速排序、堆排序

（2）不稳定实例

希尔排序（假设初始增量为2）：2 2 1 8 9 5 ，其中第一个2会被调整到第二个2 的后面。

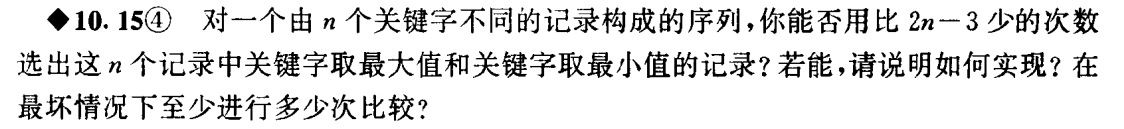
快速排序（以第一个元素为标杆）：3 9 2 5 2，第二个2会被放到3的位置上，即放在了第一个2的前面。

堆排序：调整堆时，子节点的key值与父节点相等时，子节点会上升，如

原始数组: (5, A) (3, B) (5, C) (2, D) (3, E)

堆排序后: (2, D) (3, E) (3, B) (5, C) (5, A)

**10.15**



设序列为a1, a2, …, an

设置两个变量max和min，分别用来记录最大值和最小值。设置工作指针i。

1. 初始化max，min：

若n为奇数，令max=min=a1，i=2；

若n为偶数，比较a1和a2，max=较大者，min=较小者，i=3

2. 成对处理剩余元素：

每次取ai和ai+1：

比较两者；

将两者中较大者与max比较，若比max大，则替换max；

将两者中较小者与min比较，若比min小，则替换min。

直到处理完所有元素。

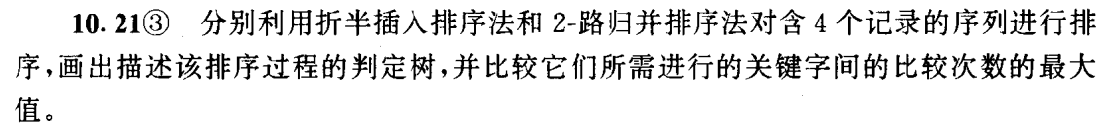
（处理每对数据时需要3次比较）

设n=2k或2k+1，

k≥2时，共需要3k = [n]×3/2次比较。由于n≥4，有[n]×3/2 ≤ 3n/2 ＜ 2n-3

k=1时，比较次数等于2n-3。

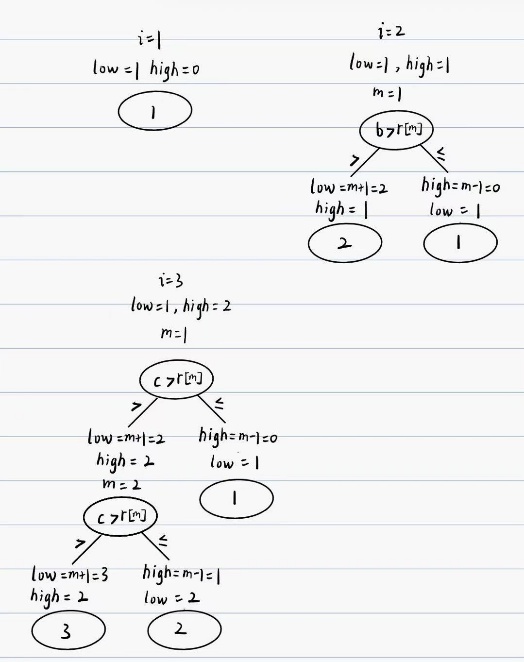
**10.21**

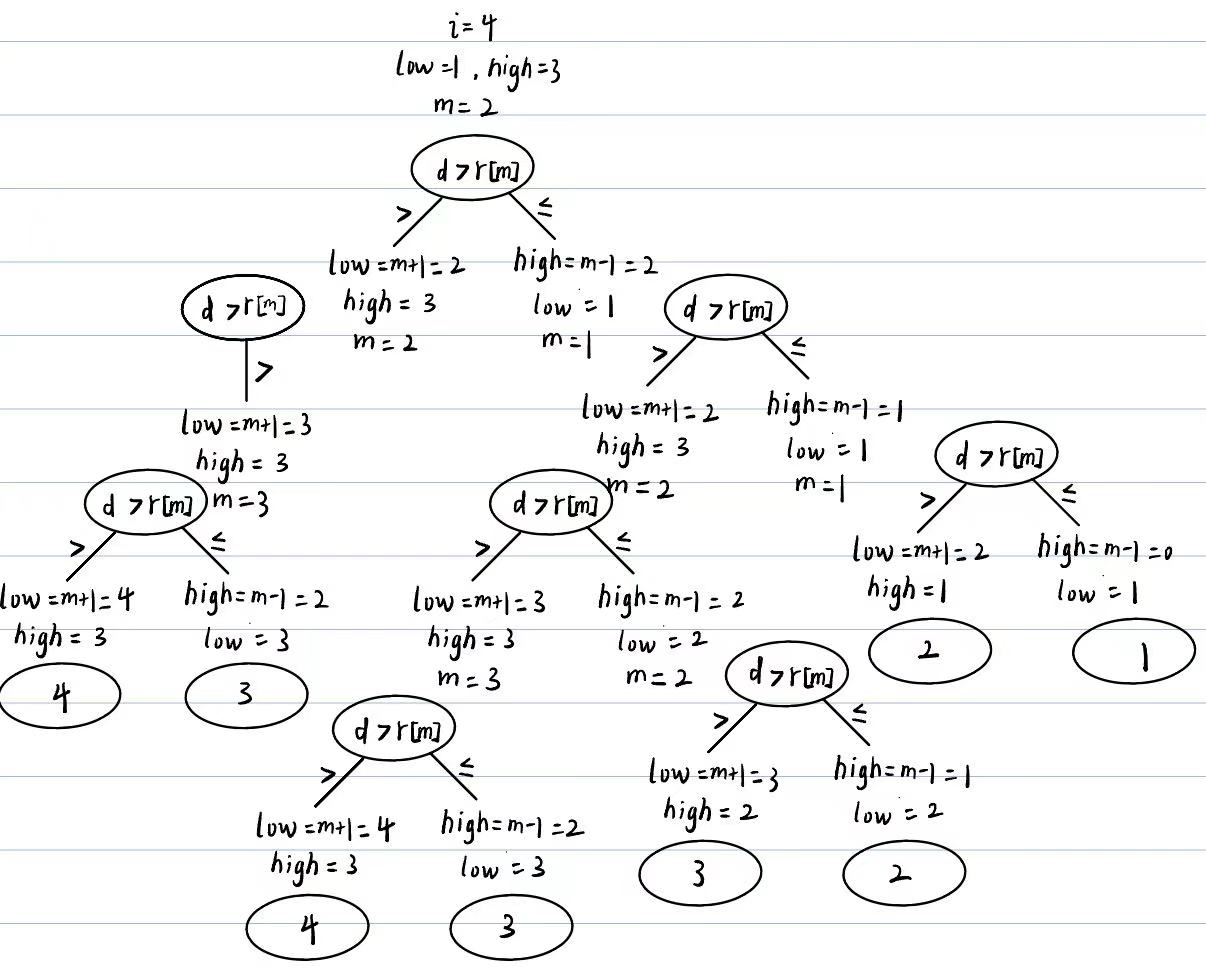


设记录序列为[a, b, c, d]

折半插入排序：

r[0]存储当前插入元素，r[1:4]=[a, b, c, d]

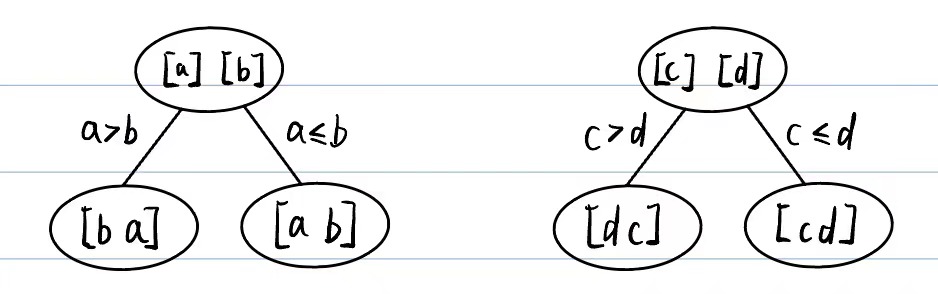




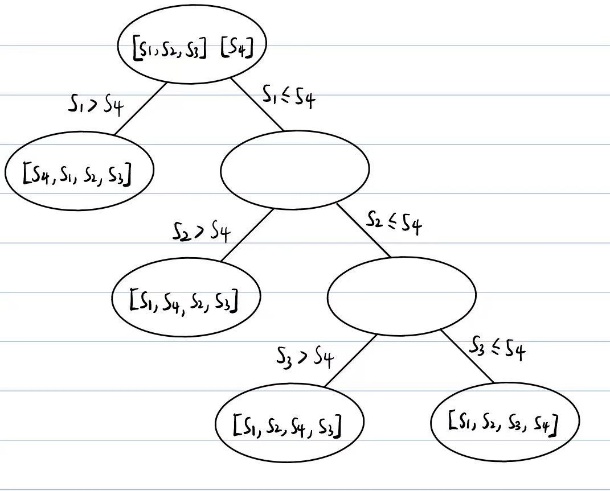
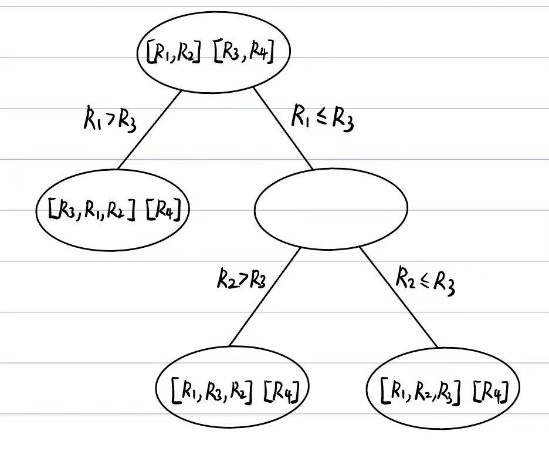
最多比较0+1+2+4=7次

2-路归并排序：

对两组最小子序列做归并，得到两个子序列[R1, R2][R3, R4]：



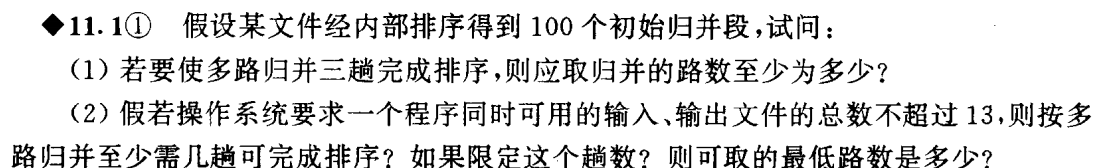
对[R1, R2][R3, R4]做归并：



最多比较1+1+2+3=7次。故两种算法的最多比较次数相同。

(Optional)

**11.1**



（1）由logk100 ≤ 3得k ≥ e^(1n(100) / 3)≈4.6

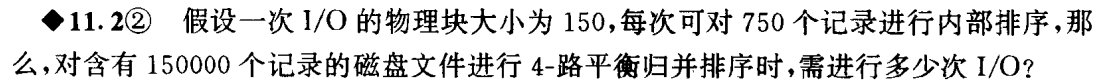
则归并路数至少为5。

（2）一次k路归并操作需要k个输入文件和1个输出文件，要求k+1≤13，则k≤12。设趟数为p，要求kp≥100。

若取p=1，显然k不满足约束。若取p=2，则10≤k≤12

因此，至少需要2趟完成排序。如果限定2趟，可取的最低路数为10路。

**11.2**



归并初始段数s=150000/750=200个

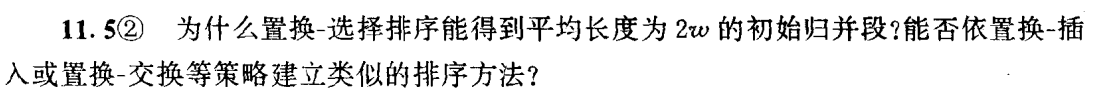
设趟数为p，4p≥200，则求得p≥3.8219，即p最小取4。

生成初始归并段的I/O次数为150000/150\*2=2000次

每趟归并排序的I/O次数也为150000/150\*2=2000次

则总I/O次数=2000+2000\*4=10000次

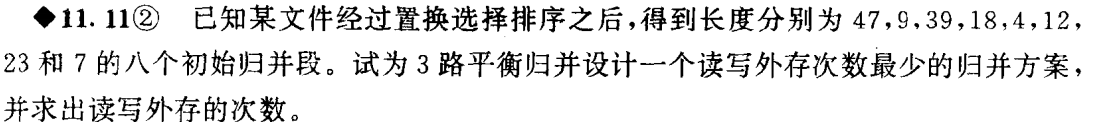
**11.5**



（1）工作区的初始容量长度为ω。设工作区的最小记录为r，假设输入序列为随机分布时，每次读入的新纪录有0.5的概率≥r，从而能被划分到该归并段，且对工作区初始的每个记录以及后来加入该归并段的记录都是如此，即有该归并段长度L = ω+0.5L，求得L=2ω。

（2）不可以。插入/交换策略均需维持工作区全序，会使工作区丧失筛选不同归并段记录数据的能力。

**11.11**



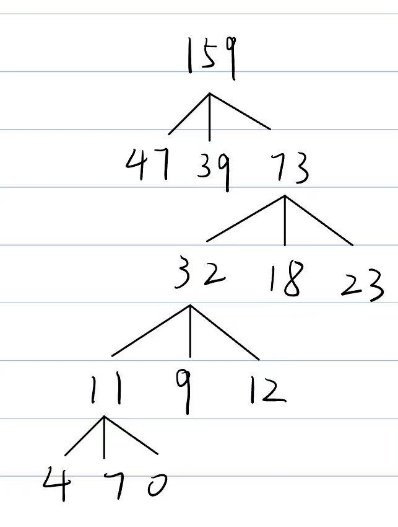
构造3路哈夫曼树​：

设3路哈夫曼树中度为0的结点个数为n0，度为3的结点个数为n3，则有n0+n3=3n3+1。

则n0=2n3+1，即叶子数n0应满足n0=2k+1。

由于只有8个初始归并段，所以需要添加一个长度为0的虚段。

构造出的3路哈夫曼树如图：

第一趟读入4、7、0三个段，写入11这个新段；

第二趟读入11、9、12三个段，写入32这个新段；

第三趟读入32、18、23三个段，写入73这个新段；

第三趟读入47、39、73三个段，写入159这个新段；

访问外存次数 = (11+32+73+159)\*2 = 550