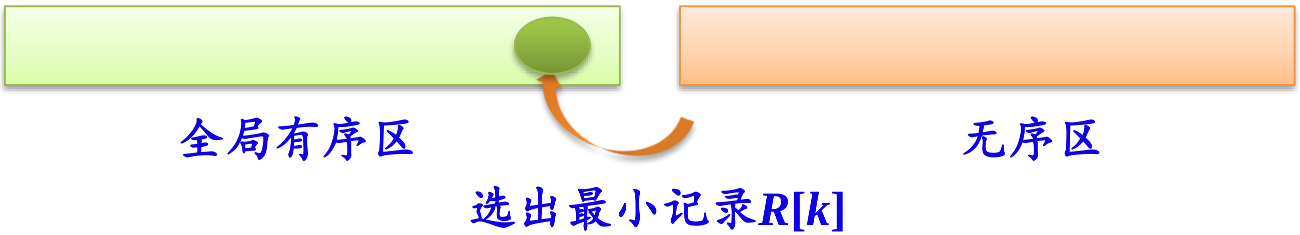
10.4 选择排序

基本思路



常见的选择排序方法：

（1）简单选择排序（或称直接选择排序）

（2）堆排序

10.4.1 简单选择排序

从$a[i..n-1]$中选出最小元素：

```
int Min(int a[],int n,int i)
{      int k=i,  j;                   //k保存最小元素的下标
       for (j=i+1;j<n;j++)
            if (a[j]<a[k])  k=j;
       return a[k];
}
```

简单选择　n个记录中找最小记录需要$n-1$次比较

基本思路

全局有序区　　　　　　　　无序区

$R[0]$ …… $R[i-1]$	$R[i]$ …… $R[n-1]$

采用简单选择方法选出最小记录

$R[0]$ ……… $R[i-1]$ $R[i]$	$R[i+1]$ …… $R[n-1]$

全局有序区　　　　　　　　无序区

初始时，全局有序区为空

$i=0 \sim n-2$，共经过$n-1$趟排序

简单选择排序算法

```
void SelectSort(RecType R[],int n)
{    int i,j,k;  RecType tmp;
     for (i=0;i<n-1;i++)                    //做第i趟排序
     {     k=i;
           for (j=i+1;j<n;j++)
                if (R[j].key<R[k].key)
                     k=j;
           if (k!=i)                        //R[i]⇔R[k]
           {     tmp=R[i];  R[i]=R[k];  R[k]=tmp;  }
     }
}
```

在$R[i..n-1]$中采用简单选择方法选出最小的$R[k]$

采用简单选择排序方法对(2，1，3，4，5) 进行排序

初始关键字	2	1	3	4	5
i=0	1	2	3	4	5
i=1	1	2	3	4	5
i=2	1	2	3	4	5
i=3	1	2	3	4	5

没有记录移动

任何情况下：都有做n−1趟！

算法分析

从i个记录中挑选最小记录需要比较$i-1$次。

第i（$i=0\sim n-2$）趟从$n-i$记录中挑选最小记录需要比较$n-i-1$。

对 n 个记录进行简单选择排序，所需进行的关键字的比较次数总计为：

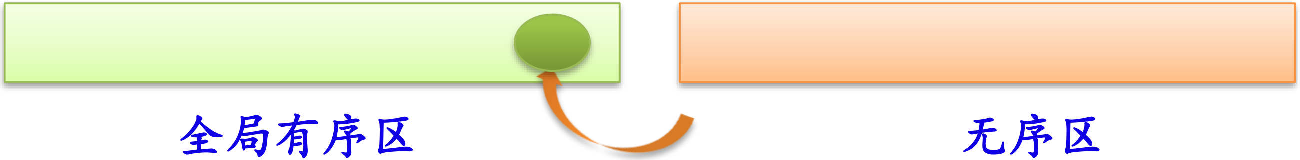
$$\sum_{i=0}^{n-2}(n-i-1)=\frac{n(n-1)}{2}$$

移动记录的次数，正序为最小值 0，反序为最大值$3(n-1)$。

简单选择排序的最好、最坏和平均时间复杂度为$O(n^2)$。

10.4.2 堆排序

基本思路



选出最小记录$R[k]$

采用堆方法选出最小记录：**堆排序算法**

1、堆的定义

一个序列$R[1..n]$，关键字分别为k_1、k_2、…、k_n。

该序列满足如下性质（简称为堆性质）：

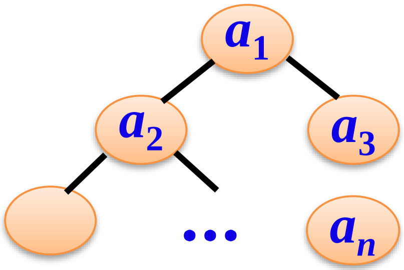
❶ $k_i \leqslant k_{2i}$ 且$k_i \leqslant k_{2i+1}$

或 ❷ $k_i \geqslant k_{2i}$ 且$k_i \geqslant k_{2i+1}$ （$1 \leqslant i \leqslant \lfloor n/2 \rfloor$）

满足第❶种情况的堆称为小根堆，满足第❷种情况的堆称为大根堆。下面讨论的堆是大根堆。

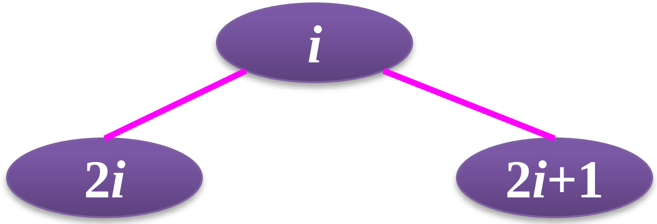
❶ 将序列$a_1\ a_2\ \dots\ a_n$看成是一颗完全二叉树

$a_1\ a_2\ \dots\ a_n$



完全二叉树

层序编号方式：



左孩子　　右孩子

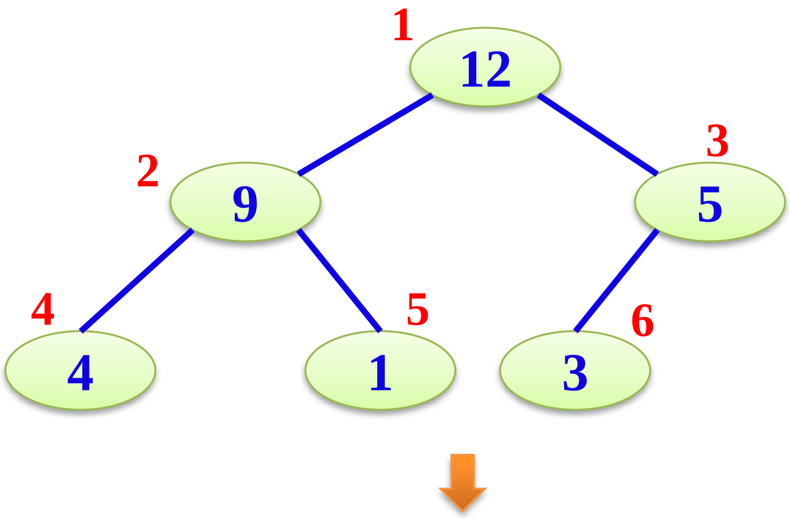
大根堆：对应的完全二叉树中，任意一个节点的关键字都大于或等于它的孩子节点的关键字。

最小关键字的记录一定是某个叶子节点！！！

❷ 如何判断一颗完全二叉树是否为大根堆

n=6

从编号为n/2=3的节点开始，逐一判断所有分支节点

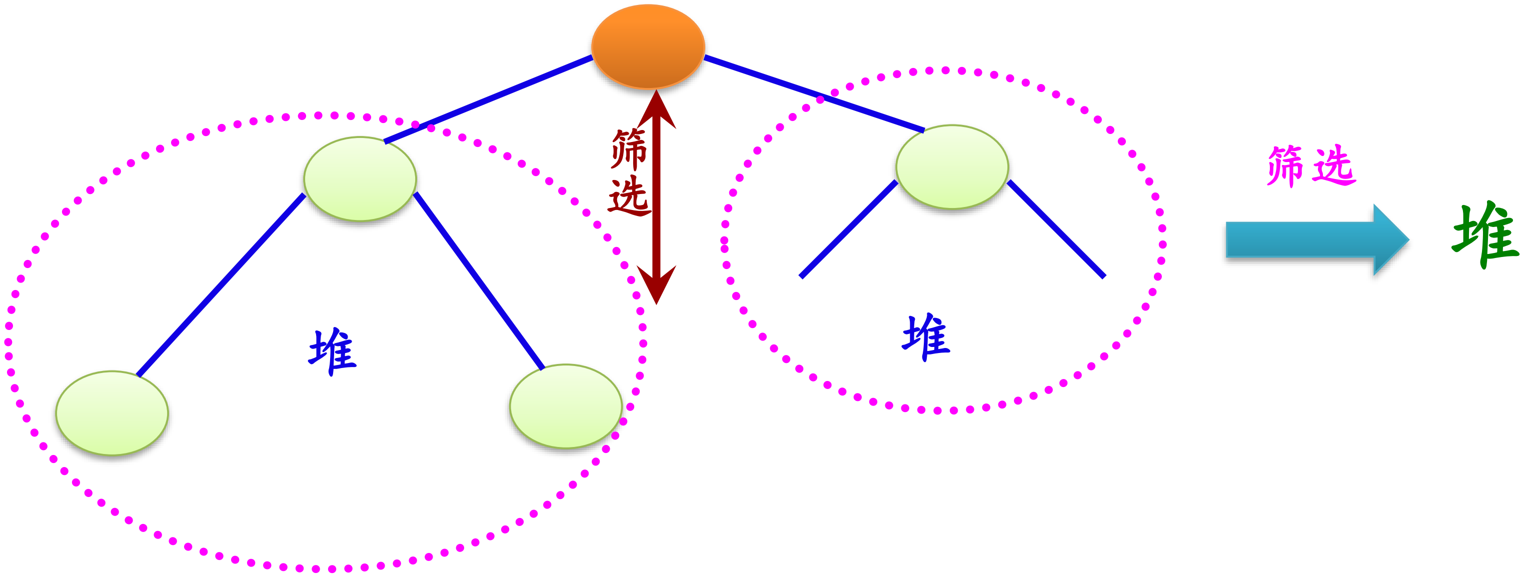


所有分支节点满足定义 ⇨ 为大根堆

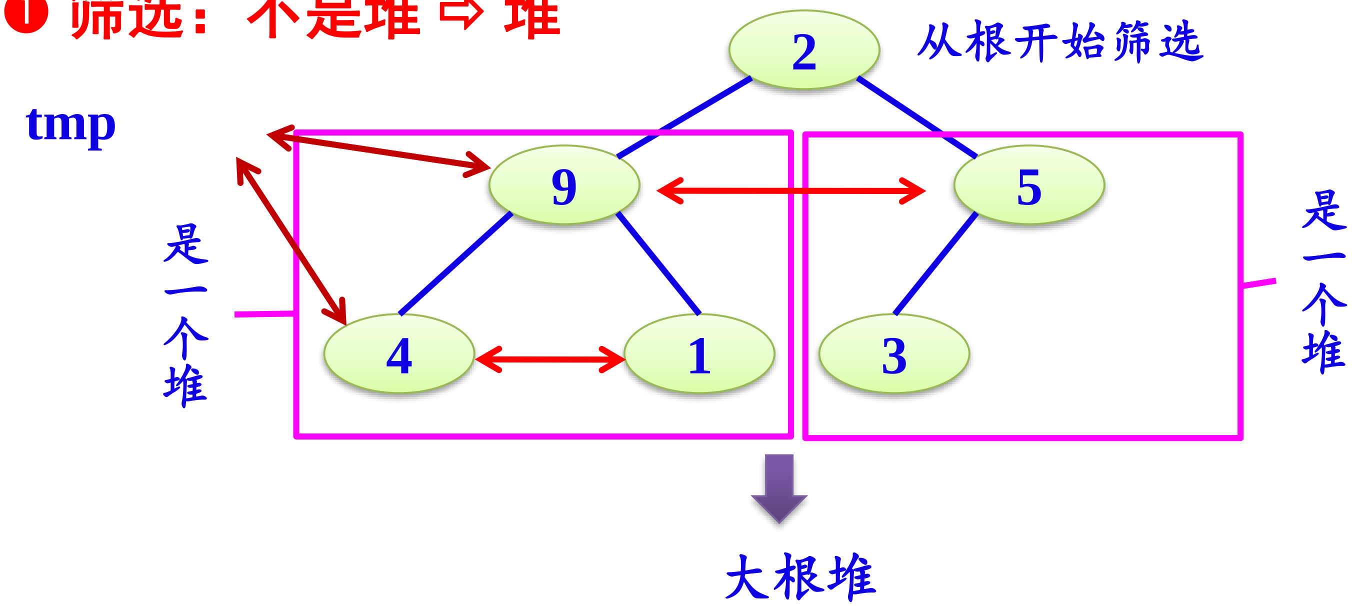
2、堆排序算法设计

堆排序的关键是构造堆，这里采用筛选算法建堆。

所谓“筛选”指的是，对一棵左/右子树均为堆的完全二叉树，“调整”根节点使整个二叉树也成为一个堆。



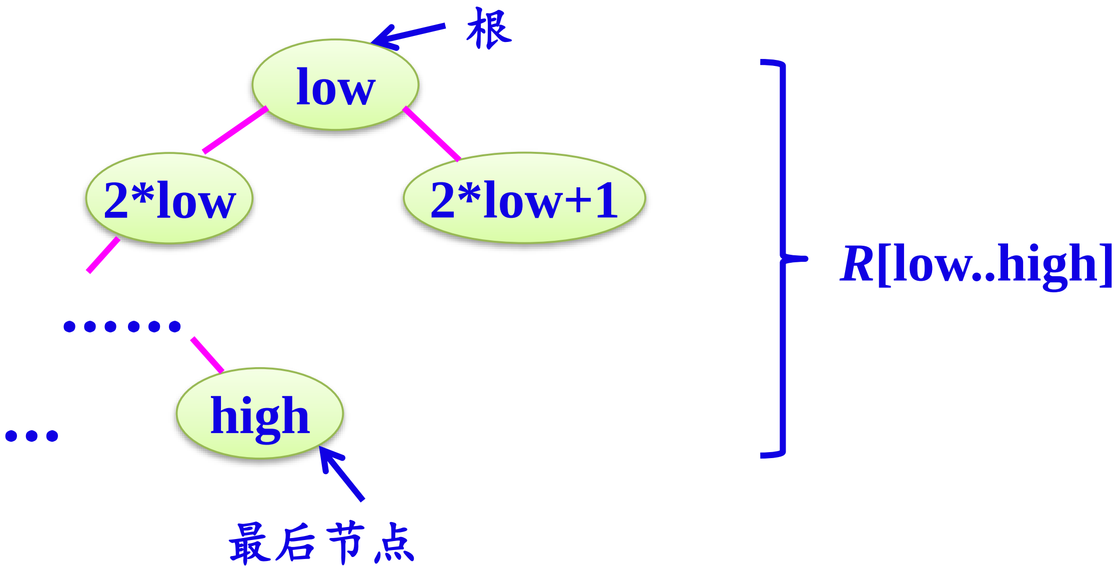
❶ 筛选：不是堆 ⇨ 堆



- 仅仅处理从根节点⇨某个叶子节点路径上的节点
- n个节点的完全二叉树高度为$\lceil \log_2(n+1) \rceil$
- 所有筛选的时间复杂度为$O(\log_2 n)$

筛选算法

sift(RecType R[], int low, int high)：*R*[low . . high]



筛选或调整算法：

```
void sift(RecType R[],int low,int high) //调整堆的算法
{     int i=low, j=2*i;                  //R[j]是R[i]的左孩子
      RecType tmp=R[i];
      while (j<=high)
      {
          if (j<high && R[j].key<R[j+1].key) j++;
          if   (tmp.key<R[j].key)         //双亲小
          {      R[i]=R[j];               //将R[j]调整到双亲节点位置上
                 i=j;                     //修改i和j值,以便继续向下筛选
                 j=2*i;
          }
          else break;                     //双亲大：不再调整
      }
      R[i]=tmp;
}
```

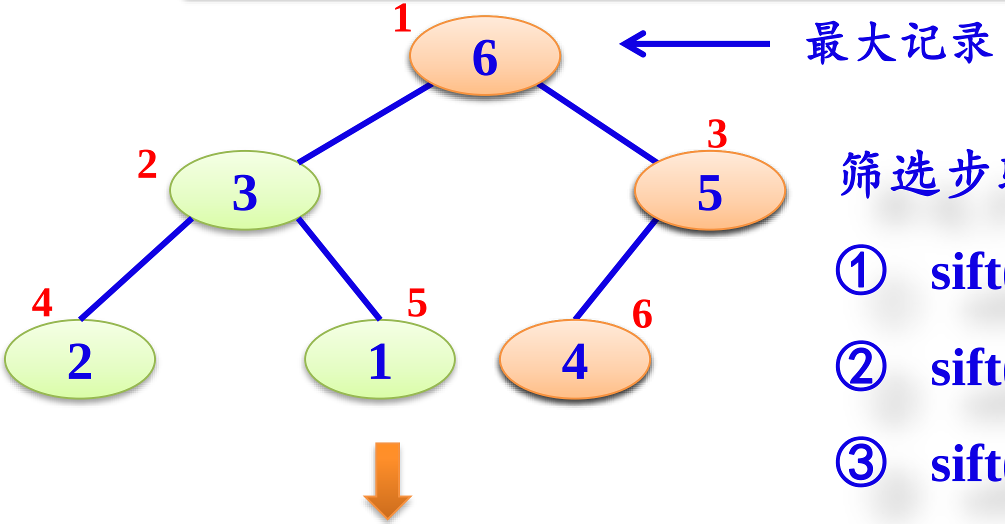
j 指向大孩子

❷ 一颗完全二叉树 ⇨ 初始堆

例如，序列：（4，3，5，2，1，6）n=6

从编号为n/2=3的节点开始，逐一筛选

```
for (i=n/2;i>=1;i--)     //循环建立初始堆
      sift(R,i,n);
```



筛选步骤：

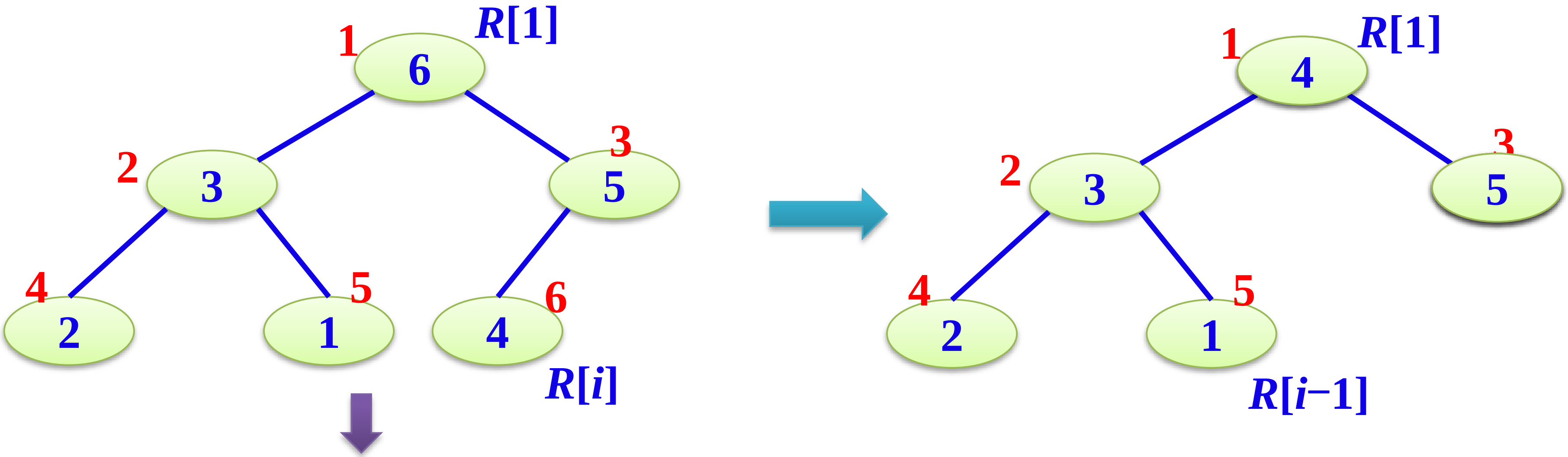
① sift(R,3,6)

② sift(R,2,6)

③ sift(R,1,6)

初始堆：（6，3，5，2，1，4）

❸ 最大记录归位



4，3，5，2，1，6

最大记录6归位

再对R[1..i−1]的记录进行筛选

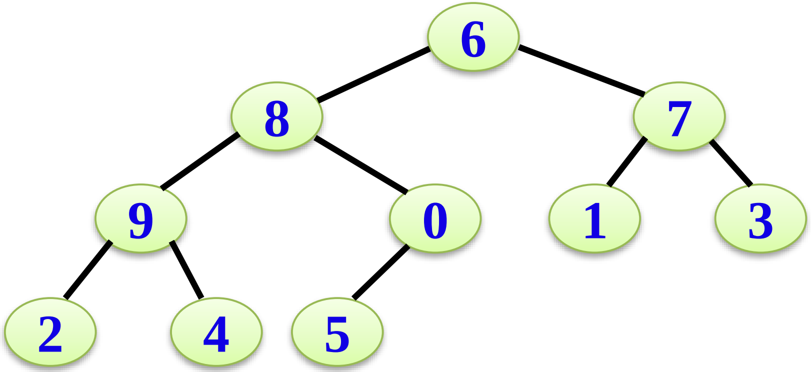
堆排序算法：

```
void HeapSort(RecType R[],int n)
{      int i;  RecType tmp;
       for (i=n/2;i>=1;i--)       //循环建立初始堆
           sift(R,i,n);
       for (i=n; i>=2; i--)       //进行n-1次循环,完成推排序
       {
            tmp=R[1];             //R[1] ⇔ R[i]
            R[1]=R[i];  R[i]=tmp;
            sift(R,1,i-1);        //筛选R[1]节点,得到i-1个节点的堆
       }
}
```

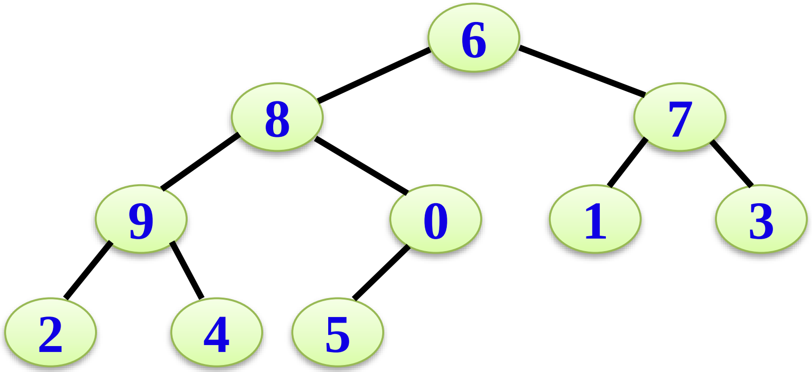
【例 10-5】设待排序的表有 10 个记录，其关键字分别为{6,8,7,9,0,1,3,2,4,5}。说明采用堆排序方法进行排序的过程。

排序序列：6，8，7，9，0，1，3，2，4，5

看成是一棵完全二叉树



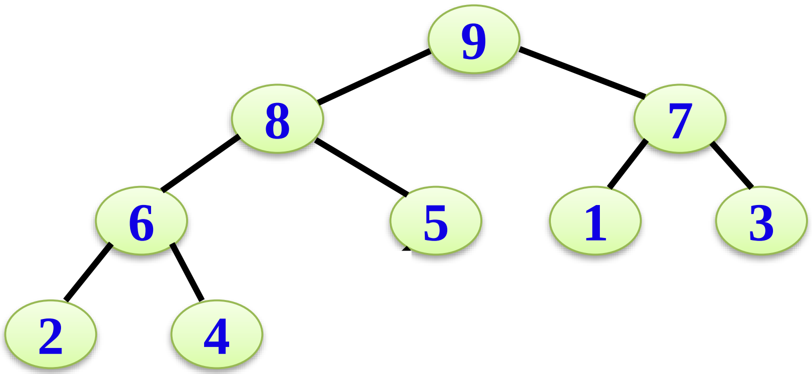
调整成初始大根堆：



调整完毕，成为一个大根堆

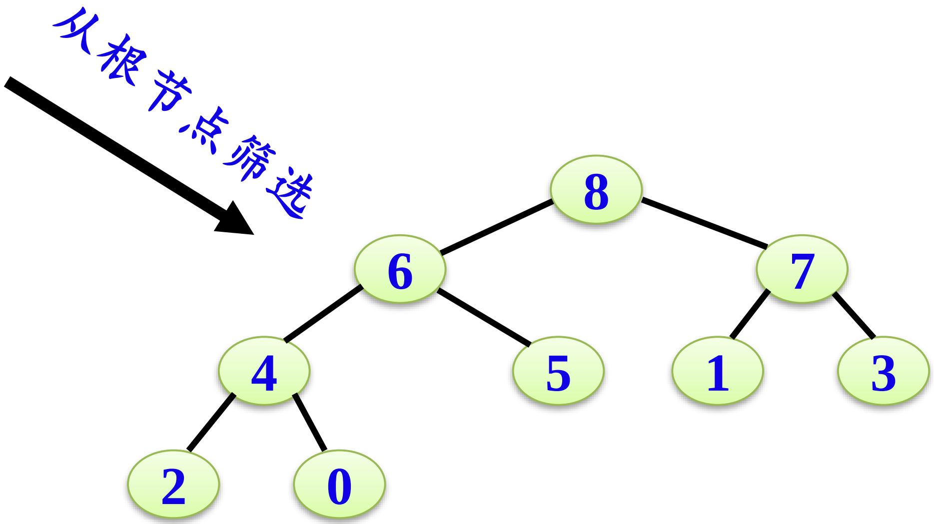
9 8 7 6 5 1 3 2 4 0

第1趟排序

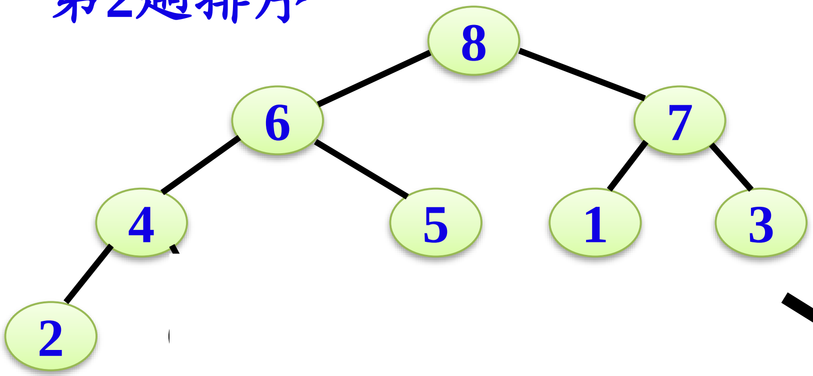


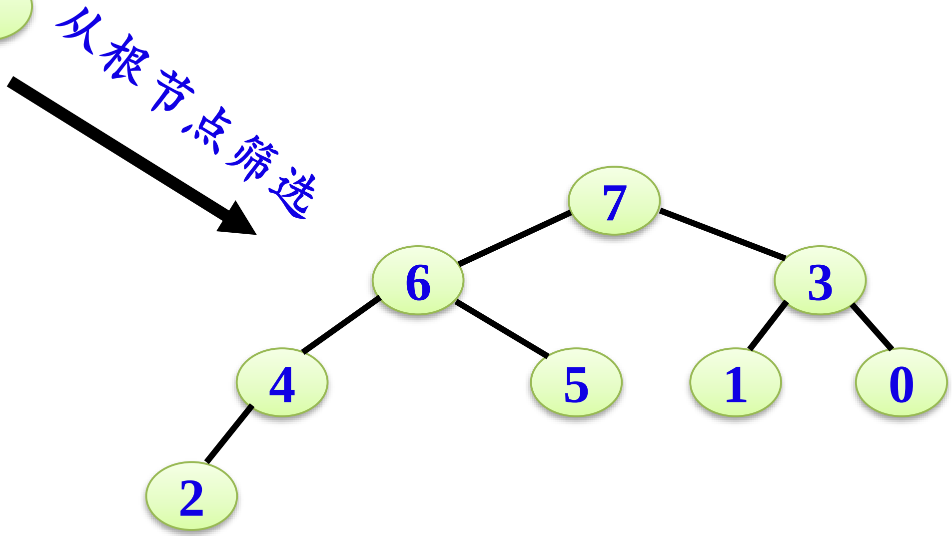
0 8 7 6 5 1 3 2 4 9

输出9（归位）



第2趟排序





0 6 7 4 5 1 3 2 8 9

输出8（归位）

其他各趟排序依此进行

最终结果：

0 1 2 3 4 5 6 7 8 9

3、堆排序算法分析

❶ 对高度为 h 的堆，一次“筛选”所需进行的关键字比较的次数至多为$2(h-1)$。

❷ 对 n 个关键字，建成高度为h（$=\lfloor \log_2 n \rfloor+1$）的堆，所需进行的关键字比较的次数不超过$4n$。

❸ 调整“堆顶”$n-1$ 次，总共进行的关键字比较的次数不超过：

$$2\left(\lfloor \log_2(n-1) \rfloor + \lfloor \log_2(n-2) \rfloor + \cdots + \log_2 2\right) < 2n(\lfloor \log_2 n \rfloor)$$

因此，堆排序的时间复杂度为$O(n\log n)$。

空间复杂度为$O(1)$，不稳定。

数据结构经典算法的启示

简单选择排序算法

利用了连续多次查找最大记录的特性

堆排序算法

在操作系统中，将多个进程放在一个队列中，每个进程有一个优先级，总是出队优先级最高的进程执行。

采用优先队列，用堆来实现！

【例10-6】设有1000个无序的整数，希望用最快的速度挑选出其中前10个最大的元素，最好选用________排序方法。

A.冒泡排序　　B.快速排序

C.堆排序　　D.直接插入排序

n=1000，k=10

- 冒泡排序的大致时间：kn
- 堆排序的大致时间：$4n+k\log_2 n$。

思考题：

选择排序的整体时间性能与初始序列的顺序有关吗？

——本讲完——