

Lecture 32: 选择排序

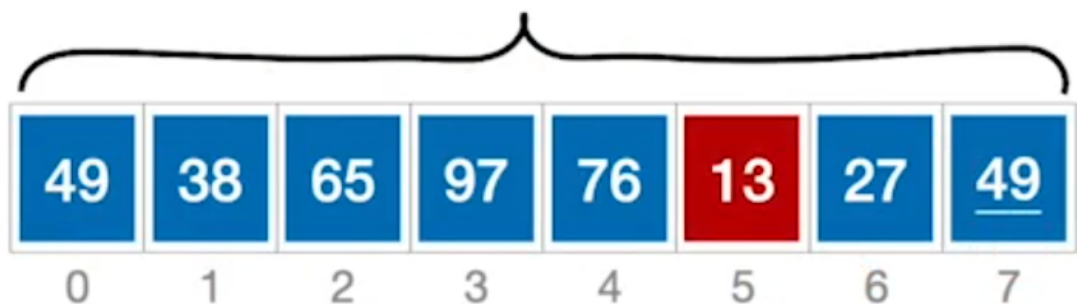
32.1 简单选择排序

32.1.1 基础概念

概念

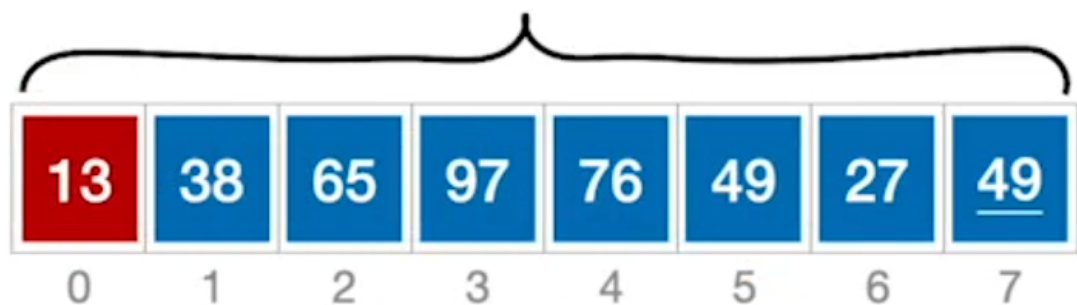
- 思想：
 - 每一趟在待排序元素中选取关键字最小 (或最大) 的元素加入有序子序列;
- 性能：
 - 空间复杂度: $O(1)$
 - 时间复杂度: $O(n^2)$
 - 总共需要对比关键字 $(n-1) + (n-2) + (n-3) \dots + 1 = \frac{n(n-1)}{2}$
 - 稳定度: 不稳定;
- 适用性：
 - 既可以用于顺序表, 也可用链表;
- 图示：
 - 待排序序列:
 - 从左往右扫描, 找到最小的;

待排序元素



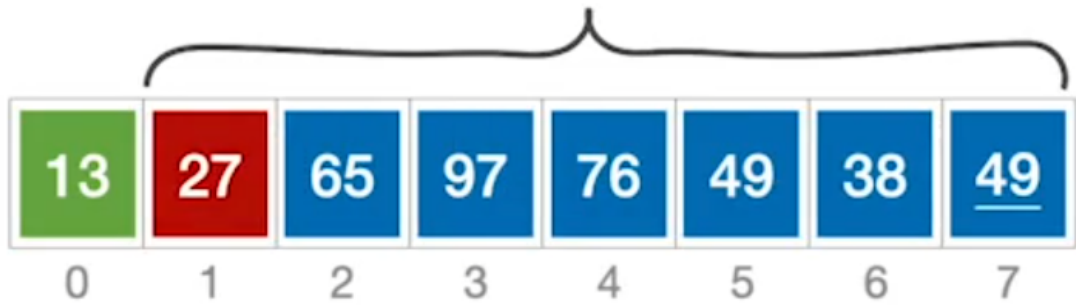
- 移动位置:

待排序元素



- 第二趟:

待排序元素



- 第三趟、第四趟也一样;
- 最后一趟不需要处理:



32.1.2 代码实现

代码实现

```
//简单选择排序
void SelectSort(int A[],int n){
    for(int i=0;i<n-1;i++){          //一共进行n-1趟
        int min=i;                  //记录最小元素位置
        for(int j=i+1;j<n;j++)      //在A[i...n-1]中选择最小的元素
            if(A[j]<A[min])
                min=j;              //更新最小元素位置
        if(min!=i)
            swap(A[i],A[min]);      //封装的swap()函数共移动元素3次
    }
}
```

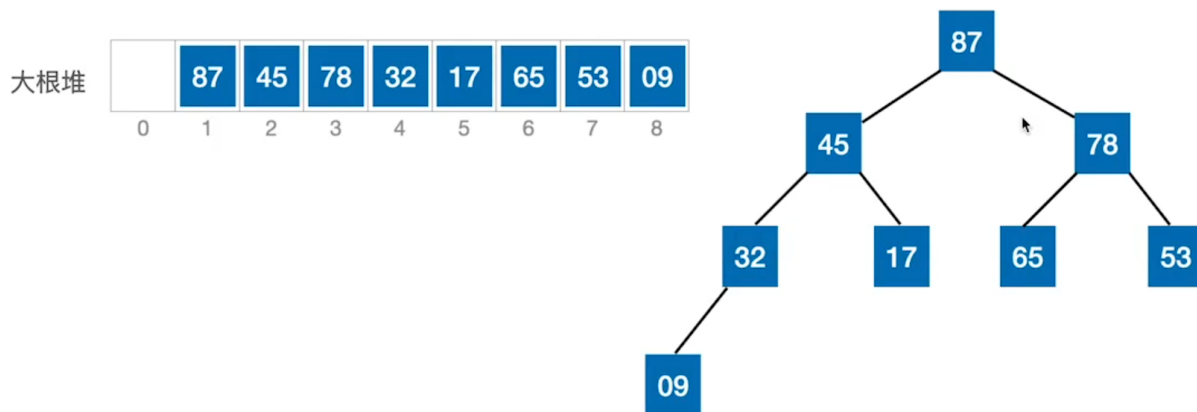
```
//交换
void swap(int &a,int &b){
    int temp = a;
    a = b;
    b = temp;
}
```

32.2 堆排序

32.2.1 什么是堆

堆的概念

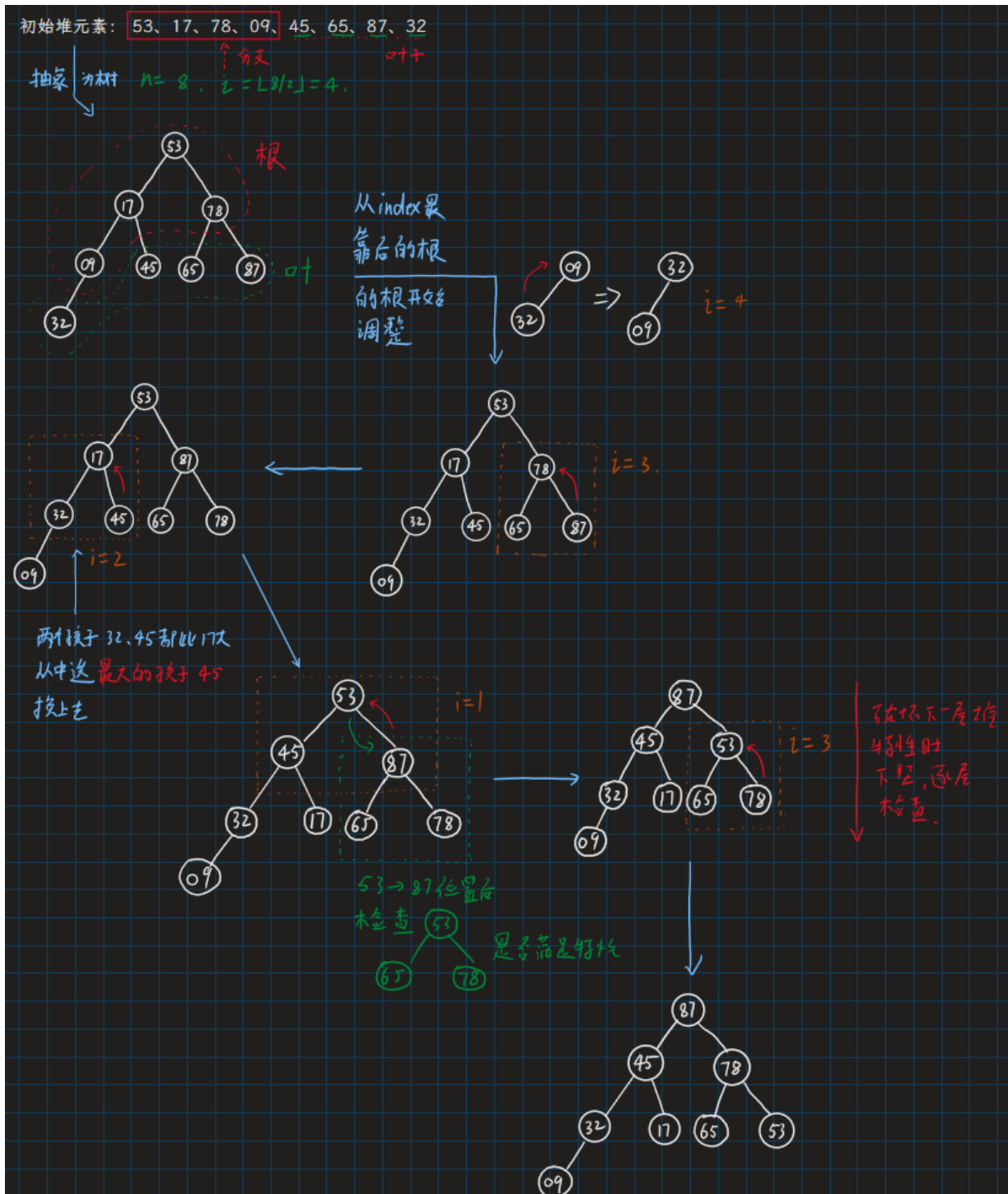
- 定义：
 - 堆：
 - 若 n 个关键字序列 $L[1..N]$ 满足下面某一条性质，则称为堆 (Heap)：
 - 大根堆：
 - 定义 $\rightarrow$ 若满足： $L(i) \geq L(2i)$ 且 $L(i) \geq L(2i+1)$ ($1 \leq i \leq n/2$)
 - 理解 $\rightarrow$ 完全二叉树中，根 $\geq$ 左、右
 - 小根堆：
 - 若满足： $L(i) \leq L(2i)$ 且 $L(i) \leq L(2i+1)$ ($1 \leq i \leq n/2$)
 - 理解 $\rightarrow$ 完全二叉树中，根 $\leq$ 左、右
- 堆的树状形式：



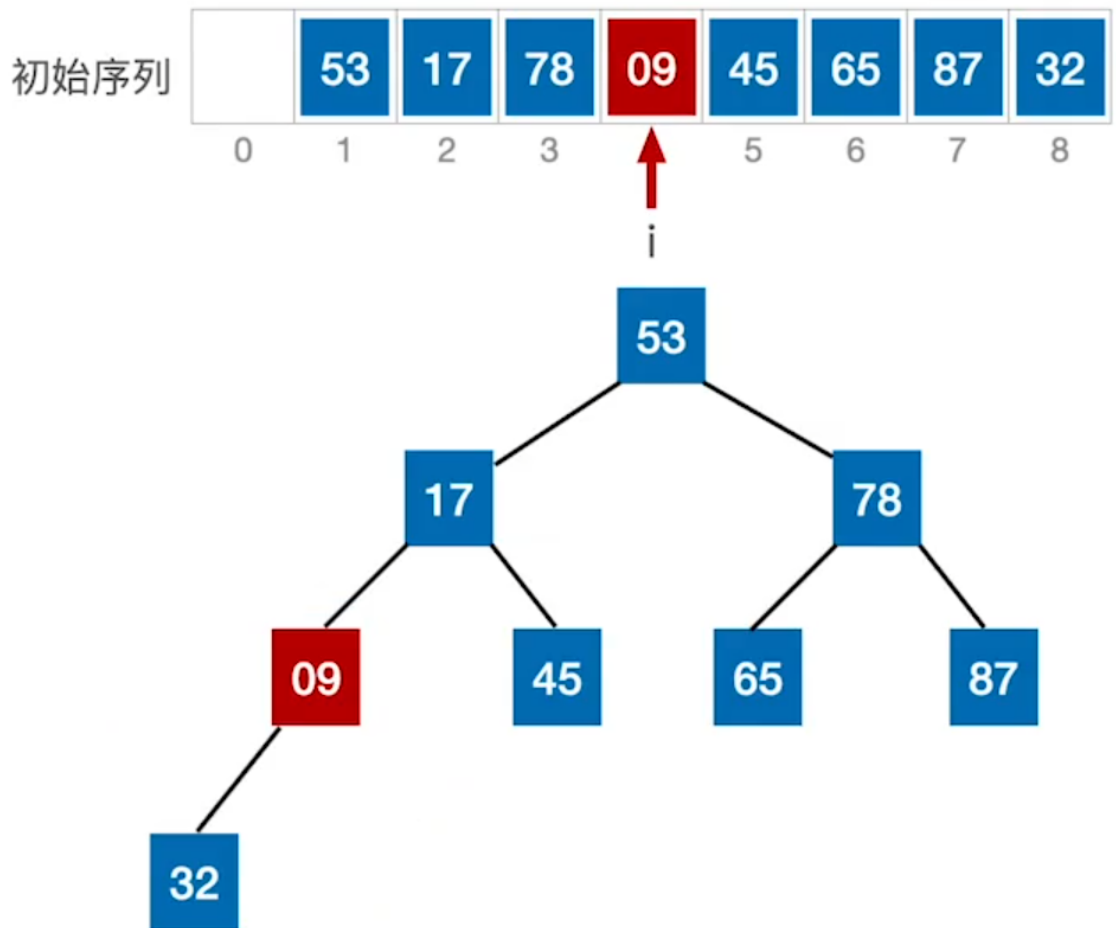
32.2.2 大根堆的建立

实现

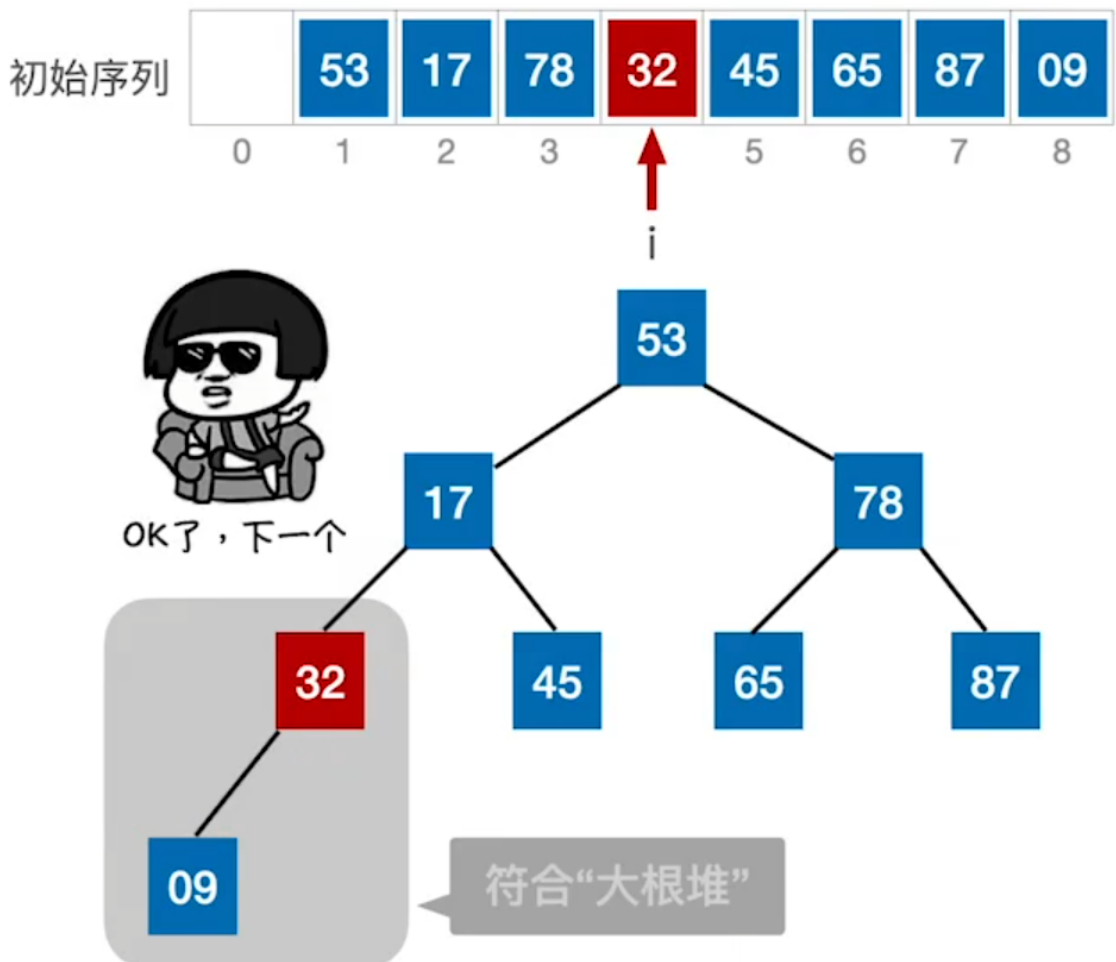
- 大根堆：
 - 根 $\geq$ 左、右
- 思路：
 - 1. 把每一个**非终端结点**（也就是分支节点）都检查一遍，是否满足根 $\geq$ 左、右；
 - 2. 如果满足则不调整，如果不满足，则进行调整，将当前结点与更大的一个孩子互换；
 - 3. 若互换后，元素的互换破坏了下一级的堆，则采用相同的方法继续往下调整（小元素不断“下坠”）；
- 位置查询：
 - 在顺序存储的完全二叉树中，非终端节点的编号为 $i \leq \lfloor n/2 \rfloor$
 - i 的左孩子 —— $2i$
 - i 的右孩子 —— $2i+1$
 - i 的父节点 —— $\lfloor i/2 \rfloor$
- 图示：



- 图示:
 - 对于非终端节点 09, 其左孩子为 32:



- 因为不满足根 $\geq$ 左、右，将当前结点与更大的左孩子 32 互换；
- 调整位置：



代码实现：建立大根堆

```
//建立大根堆
void BuildMaxHeap(int A[],int len){
    for(int i=len/2;i>0;i--)    //从后往前调整所有非终端结点
        HeadAdjust(A,i,len);
}

//将以k为根的子树调整为大根堆
void HeadAdjust(int A[],int k,int len){
    A[0]=A[k];    //A[0]暂存子树的根结点
    for(int i=2*k;i<=len;i*=2){//沿key较大的子结点向下筛选
        if(i<len&&A[i]<A[i+1])
            i++;    //取key较大的子结点的下标
        if(A[0]>=A[i])    //筛选结束
            break;
        else{
            A[k]=A[i];    //将A[i]调整到双亲结点上
            k=i;    //修改k值，以便继续向下筛选
        }
    }
    A[k]=A[0];    //被筛选结点的值放入最终位置
}
```

32.2.3 大根堆的排序

基于大根堆进行排序

- 选择排序：
 - 每一趟在待排序元素中选取**关键字最大的元素**加入有序子序列；
- 堆排序：
 - 每一趟将堆顶元素加入有序子序列，与待排序序列中的最后一个元素交换；
 - 为什么交换 -> 因为堆顶元素是最大的一个元素；
 - 核心：就是将最大的元素逐个拿出来，放到序列的最后被，完成排序 -> 堆的 Max 特性，堆顶总是 MAX 值；
 - 并将待排序元素序列再次调整为大根堆（小元素不断“下坠”）；
- 效果：
 - 对于 **N** 个元素的大根堆，经过 **N-1** 趟处理就可以得到**递增序列**；

代码实现：堆排序

```
//堆排序的完整逻辑
void HeapSort(int A[],int len){
    BuildMaxHeap(A,len);    //初始建堆
    for(int i=len;i>1;i--){    //n-1趟的交换和建堆过程
        swap(A[i],A[1]);    //堆顶元素和堆底元素交换
        HeadAdjust(A,1,i-1);    //把剩余的待排序元素整理成堆
    }
}
```

}

算法效率分析

- 一次下坠：
 - 若树高为 h ，某结点在第 i 层，则将这个结点向下调整最多只需要“下坠” $N-1$ 层，关键字对比次数不超过 $2(h-i)$ ；
- 每一趟的复杂度：
 - 不超过 $O(h) = O(\log_2 n)$
- 树高：
 - n 个结点的完全二叉树的树高 $h = \lfloor \log_2 n \rfloor + 1$
 - 第 i 层最多有 2^{i-1} 个结点，而只有第 $1 \sim (h-1)$ 层的结点才有可能需要“下坠”调整
- 调整次数：

$$\sum_{i=h-1}^1 2^{i-1} 2(h-i) = \sum_{i=h-1}^1 2^i (h-i) = \sum_{j=1}^{h-1} 2^{h-j} j \leq 2n \sum_{j=1}^{h-1} \frac{j}{2^j} \leq 4n$$

- 效率：
 - 1. 关键字对比次数不超过 $4n$
 - 2. 建堆时间复杂度 $O(n)$
- 排序的效率：
 - 排序 = 执行的趟次 $\times$ 一次下坠 = $O(n-1) * O(\log_2 n) = O(n \log_2 n)$
 - 堆 = $O(n)$
 - 堆排序：
 - 堆排序 = $O(n) + O(n \log_2 n) = O(n \log_2 n)$
 - 空间复杂度： $O(1)$
- 稳定性：
 - 不稳定；

32.3 堆的插入与删除

32.3.1 插入

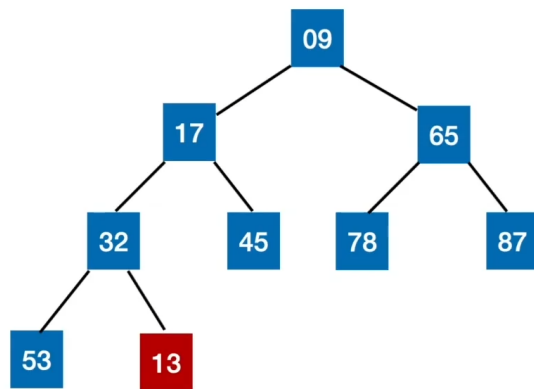
实现

- 概念：
 - 核心：新元素放到表尾，根据大小根堆要求，新元素不断上升；
 - 对于小根堆，新元素放到表尾，与父节点对比，若新元素比父节点更小，则将二者互换；
 - 新元素就这样一路“上升”，直到无法继续上升为止；
- 图示：
 - 插入元素后：

小根堆



- i 的左孩子 —— $2i$
- i 的右孩子 —— $2i+1$
- i 的父节点 —— $\lfloor i/2 \rfloor$



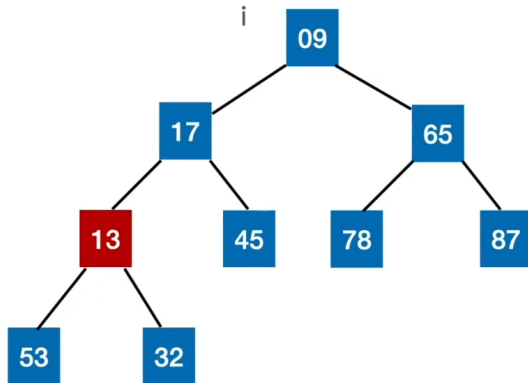
对于小根堆，新元素放到表尾，与父节点对比，若新元素比父节点更小，则将二者互换。新元素就这样一路“上升”，直到无法继续上升为止

- 找到其父节点：使用 $\text{index}/2$ 然后向下取整，得到 index 4 位置：

小根堆



- i 的左孩子 —— $2i$
- i 的右孩子 —— $2i+1$
- i 的父节点 —— $\lfloor i/2 \rfloor$



对于小根堆，新元素放到表尾，与父节点对比，若新元素比父节点更小，则将二者互换。新元素就这样一路“上升”，直到无法继续上升为止

- 一直和父节点对比，直到无法上升；

32.3.2 删除

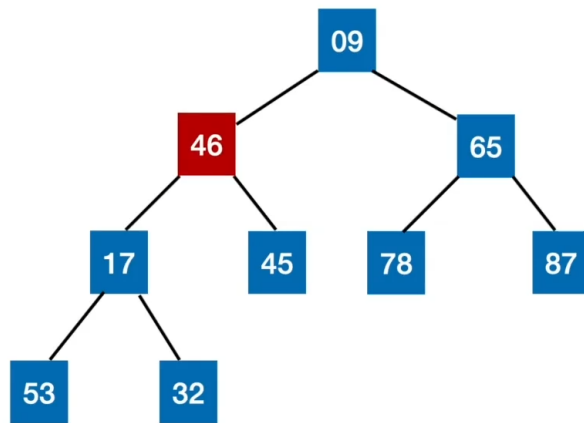
概念

- 实现：
 - 被删除的元素用堆底元素替代，然后让该元素不断“下坠”，直到无法下坠为止
- 关键字对比次数：
 - 每次“上升”调整只需对比关键字 1 次；
 - 每次“下坠”调整可能需要对比关键字 2 次，也可能只需对比 1 次；
- 图示：

小根堆



- i 的左孩子 —— $2i$
- i 的右孩子 —— $2i+1$
- i 的父节点 —— $\lfloor i/2 \rfloor$



被删除的元素用堆底元素替代，然后让该元素不断“下坠”，直到无法下坠为止

获取笔记持续更新可加微信: WandD222