

## 本节内容

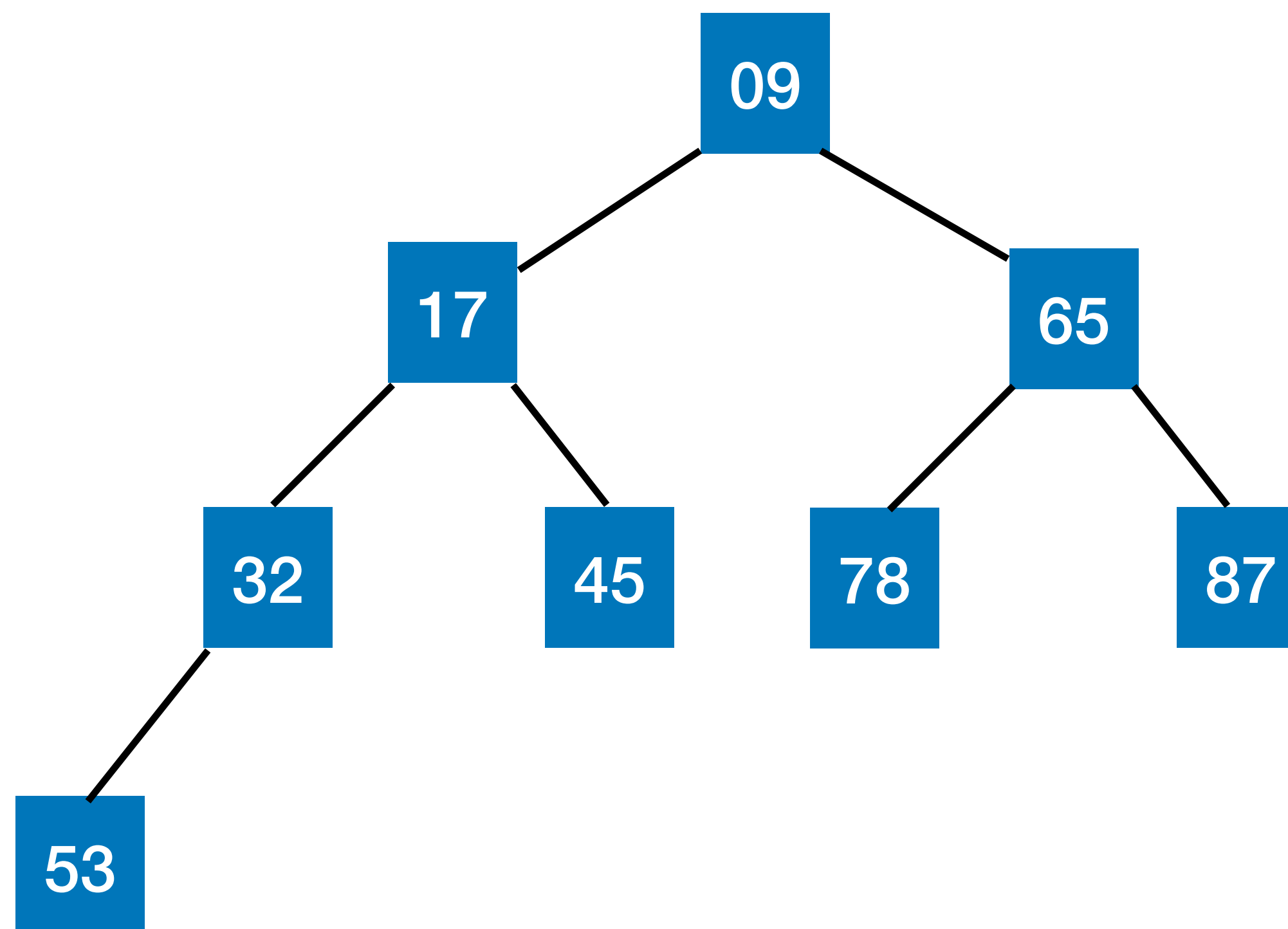
堆  
插入删除

# 在堆中插入新元素



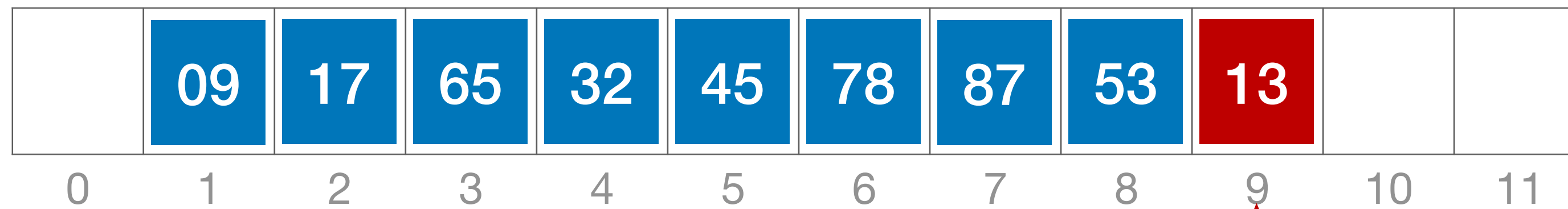
小根堆

	09	17	65	32	45	78	87	53			
0	1	2	3	4	5	6	7	8	9	10	11

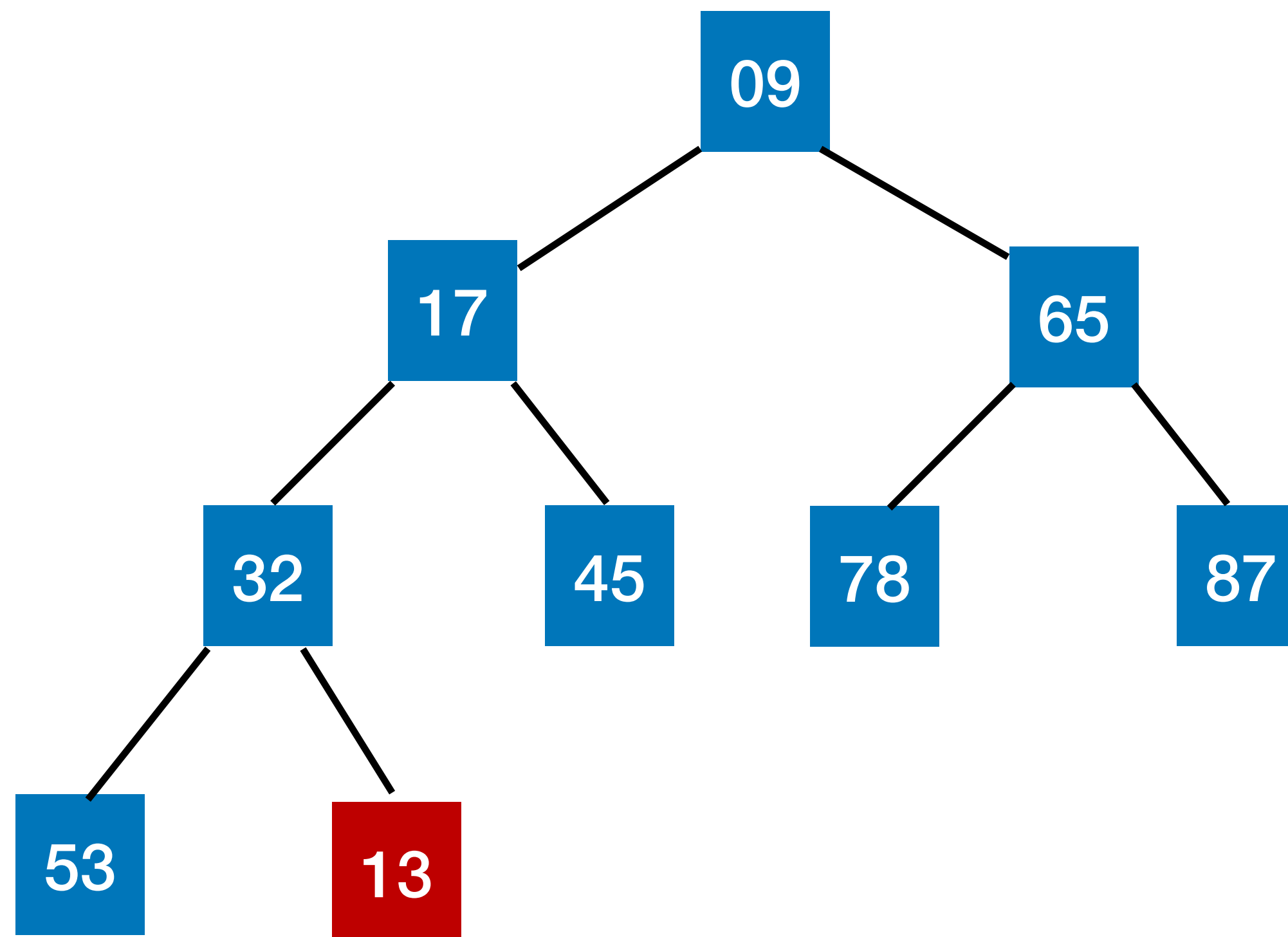


# 在堆中插入新元素

小根堆



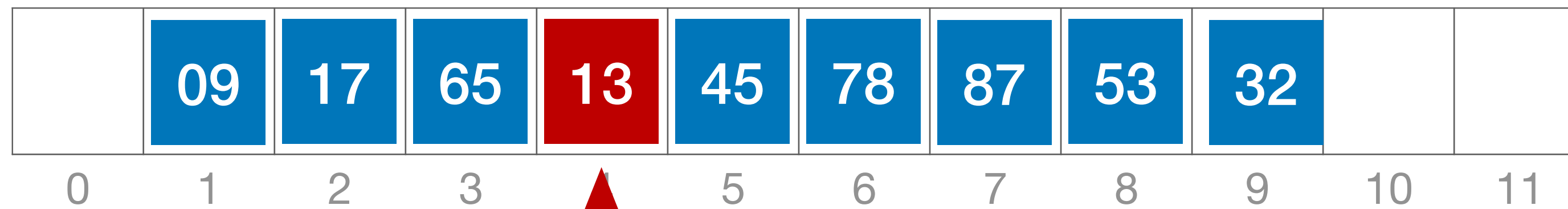
- $i$  的左孩子 ——  $2i$
- $i$  的右孩子 ——  $2i+1$
- $i$  的父节点 ——  $\lfloor i/2 \rfloor$



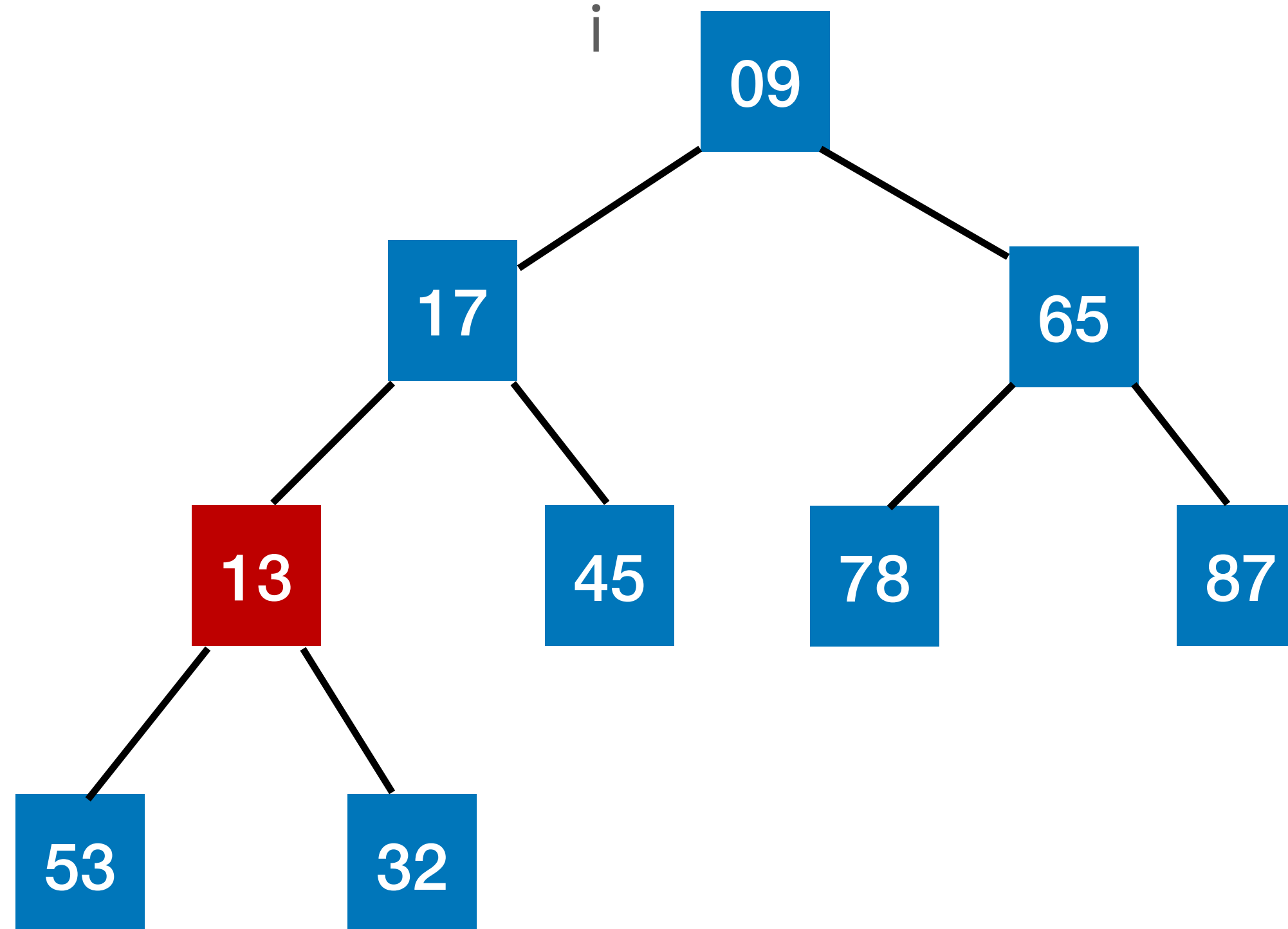
对于小根堆，新元素放到表尾，与父节点对比，若新元素比父节点更小，则将二者互换。新元素就这样一路“上升”，直到无法继续上升为止

# 在堆中插入新元素

小根堆



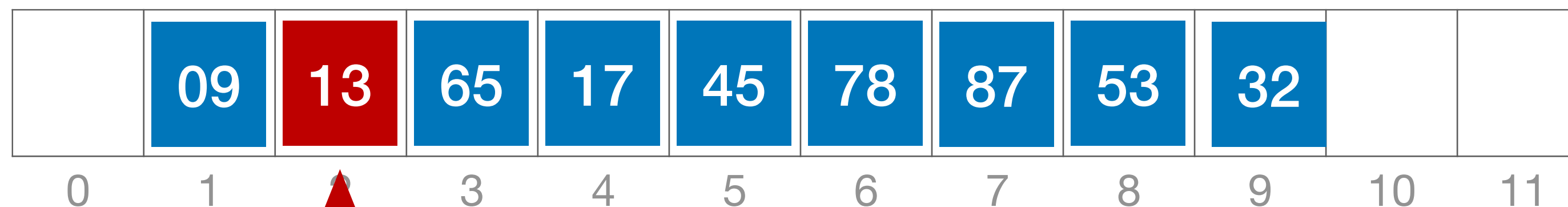
- $i$  的左孩子 ——  $2i$
- $i$  的右孩子 ——  $2i+1$
- $i$  的父节点 ——  $\lfloor i/2 \rfloor$



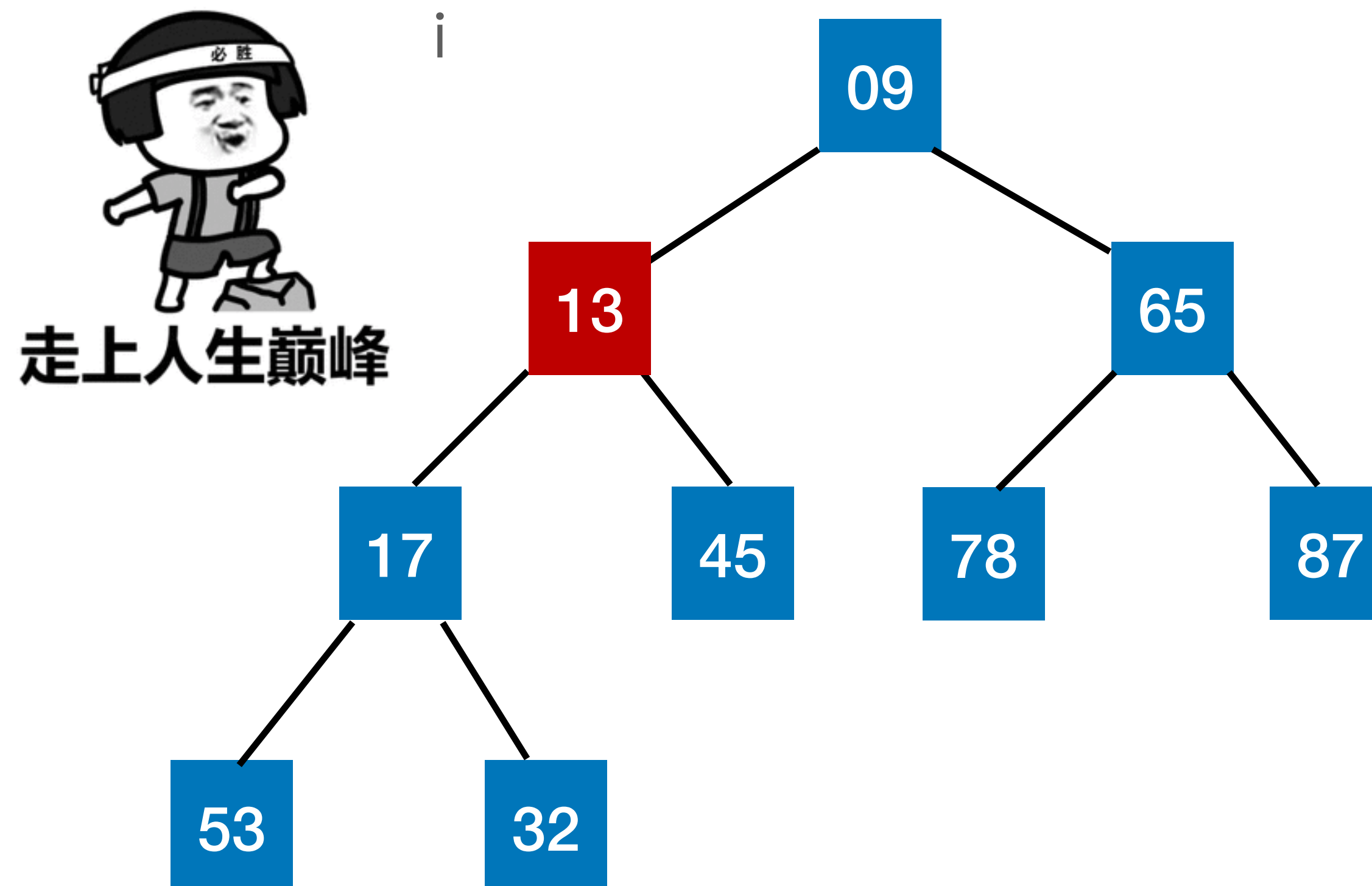
对于小根堆，新元素放到表尾，与父节点对比，若新元素比父节点更小，则将二者互换。新元素就这样一路“上升”，直到无法继续上升为止

# 在堆中插入新元素

小根堆



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

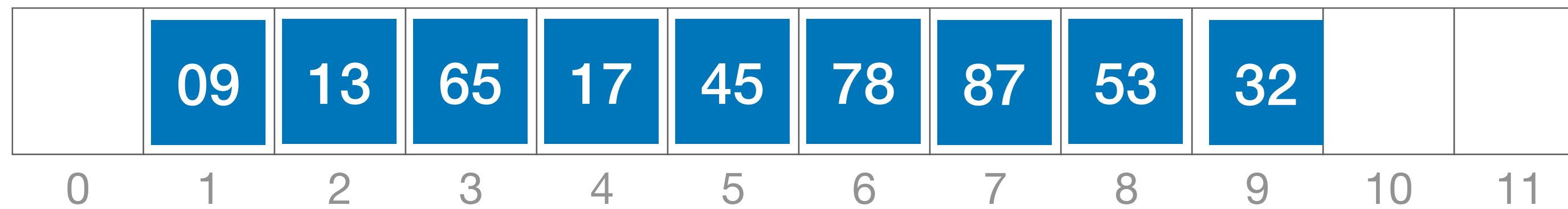


对于小根堆，新元素放到表尾，与父节点对比，若新元素比父节点更小，则将二者互换。新元素就这样一路“上升”，直到无法继续上升为止

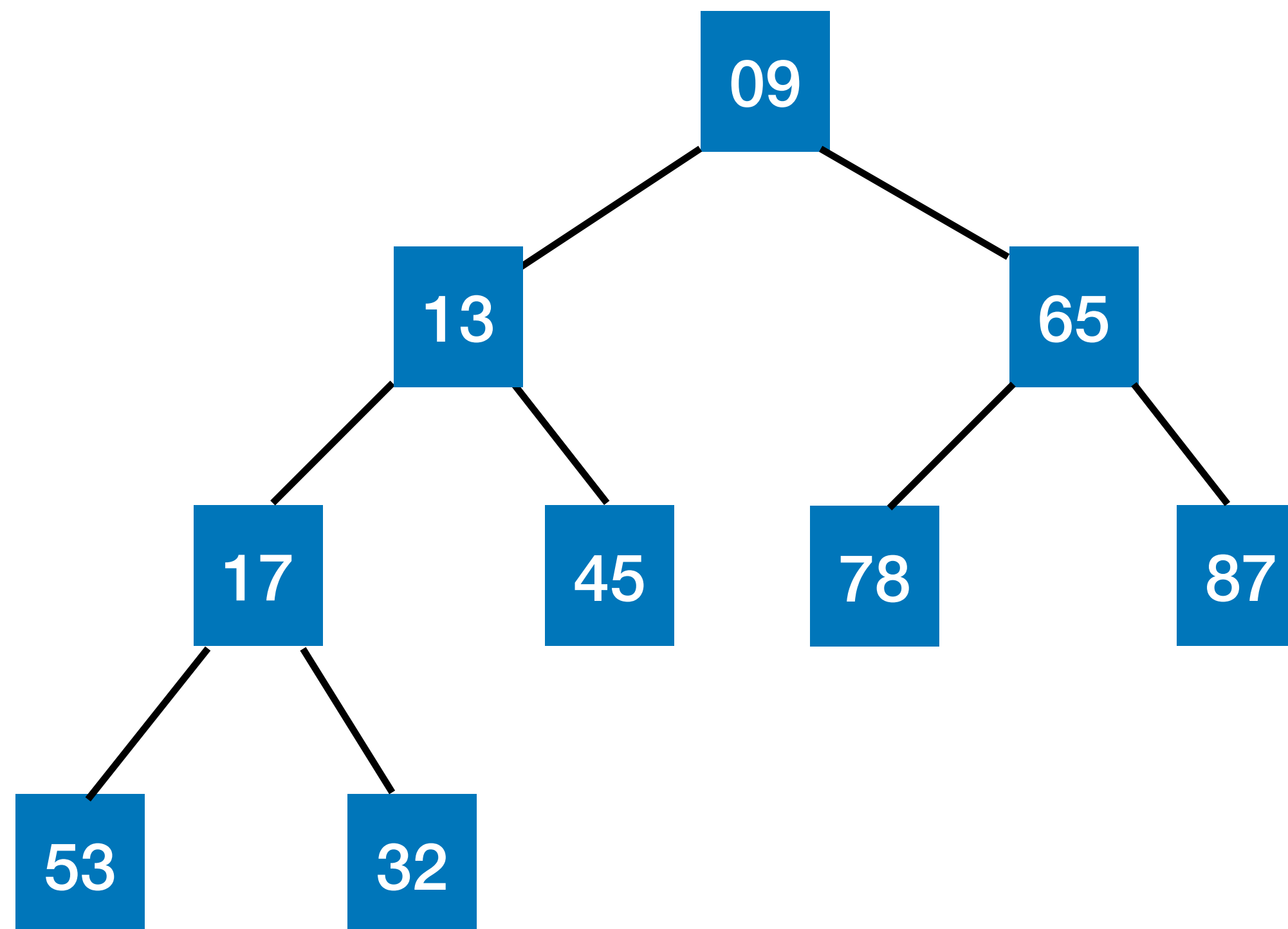
对比关键字的次数 = 3次

# 在堆中插入新元素

小根堆



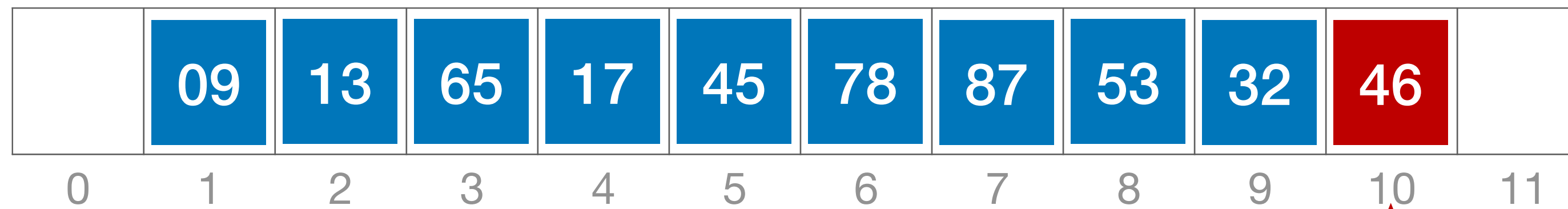
- $i$  的左孩子 ——  $2i$
- $i$  的右孩子 ——  $2i+1$
- $i$  的父节点 ——  $\lfloor i/2 \rfloor$



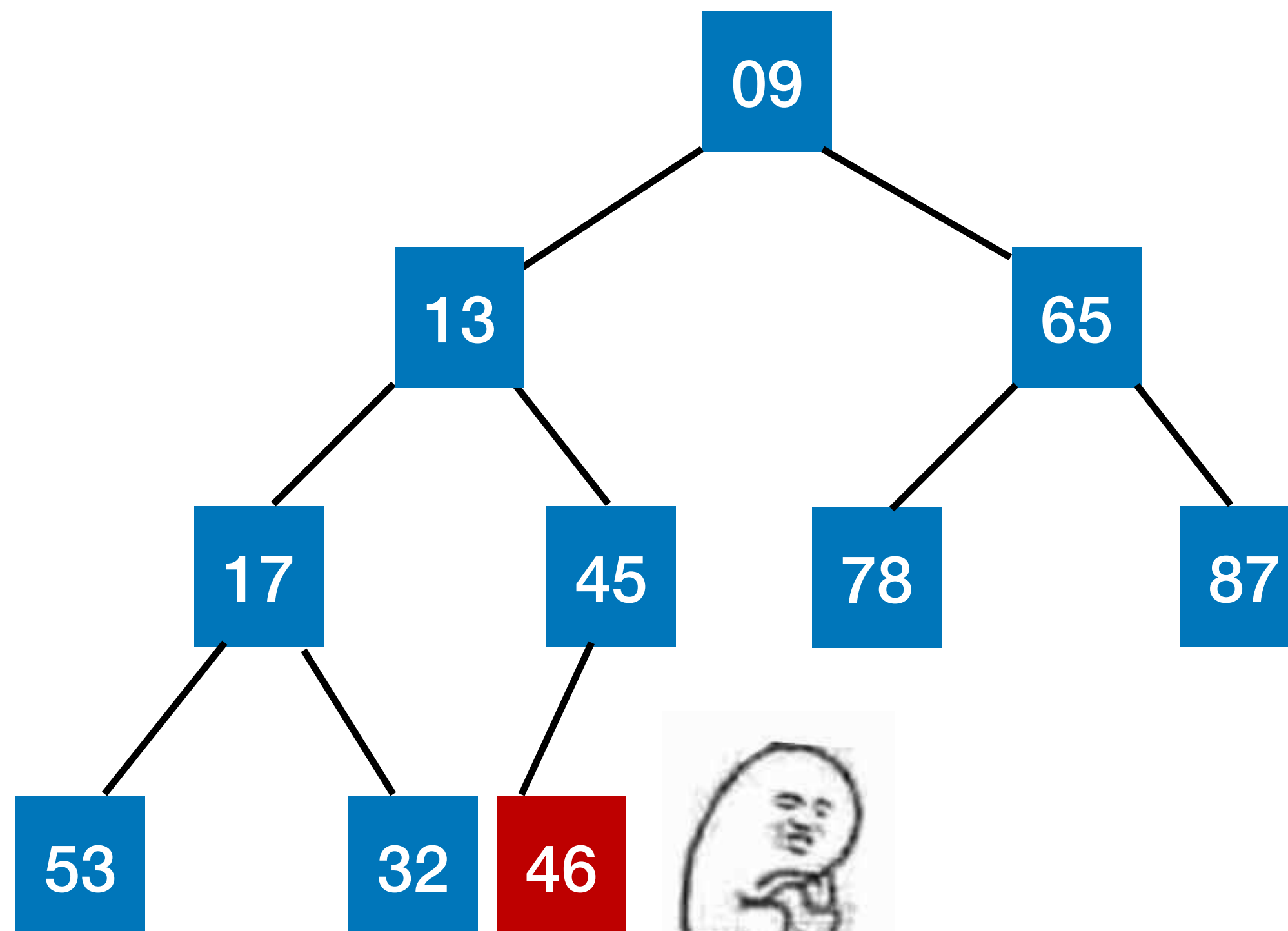
对于小根堆，新元素放到表尾，与父节点对比，若新元素比父节点更小，则将二者互换。新元素就这样一路“上升”，直到无法继续上升为止

# 在堆中插入新元素

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

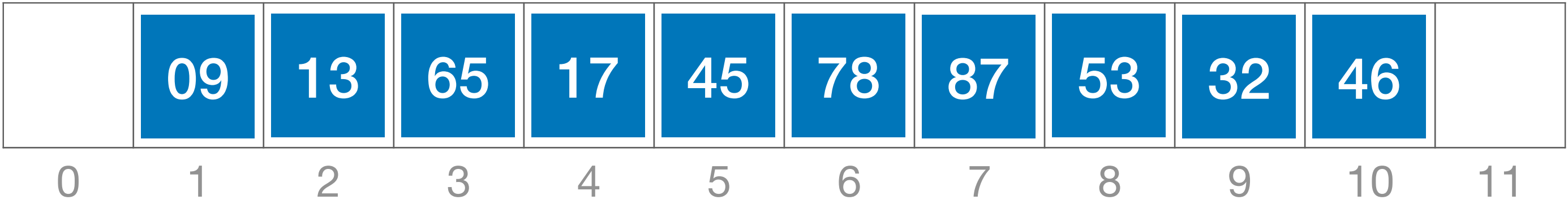


对比关键字的次数 = 1次

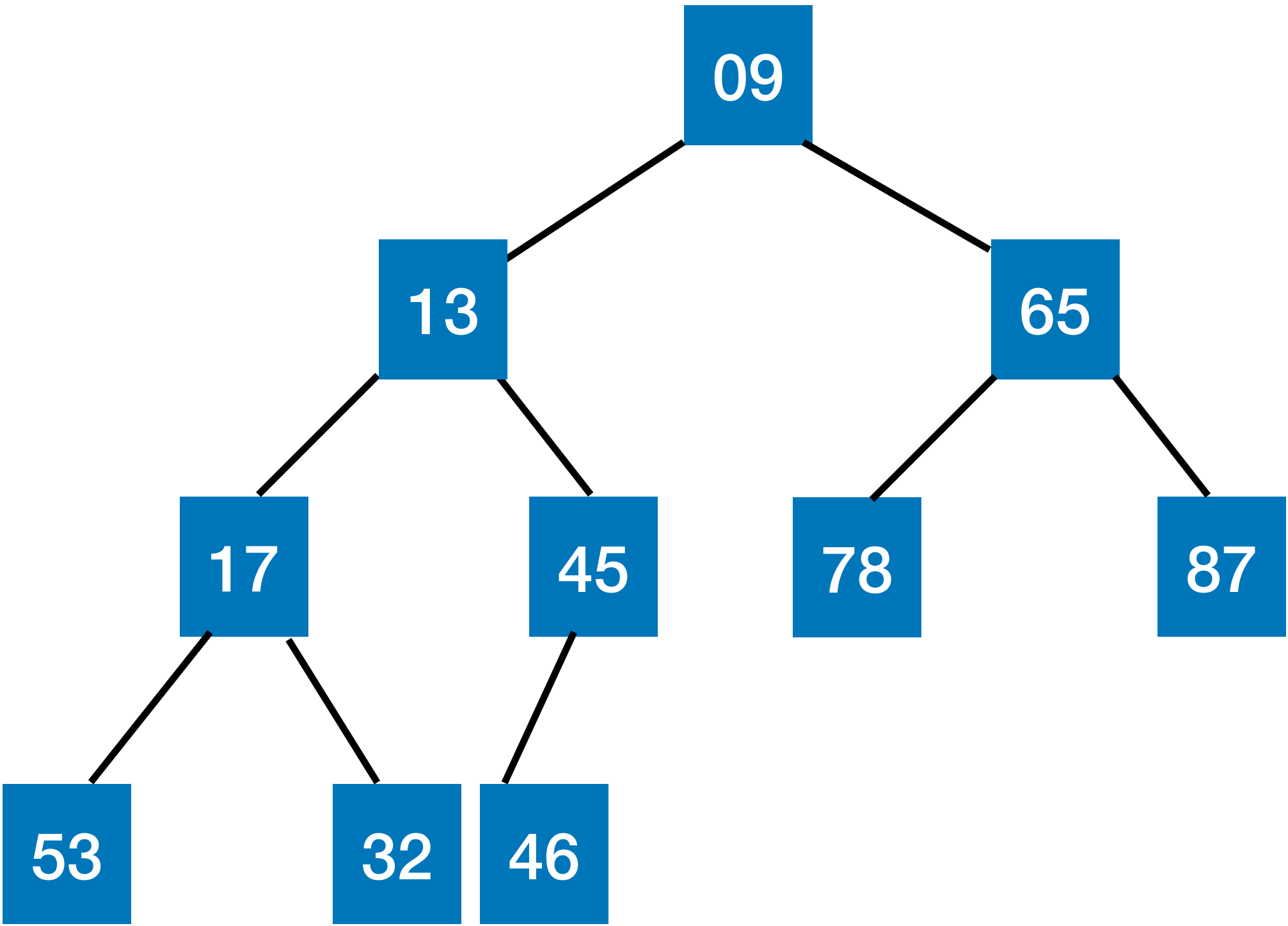
# 在堆中删除元素



小根堆



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

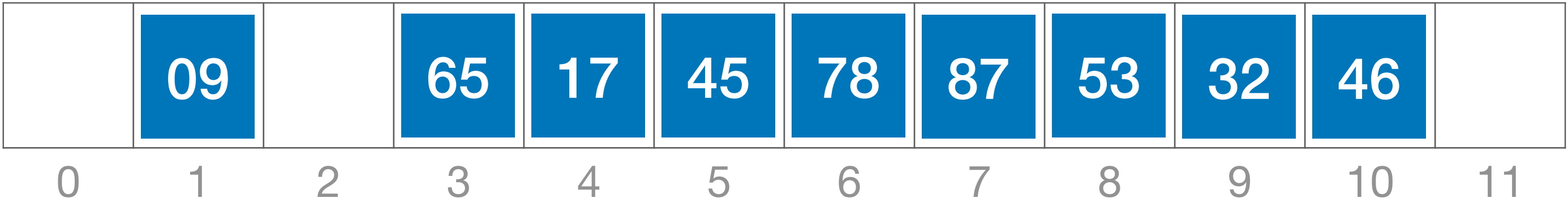




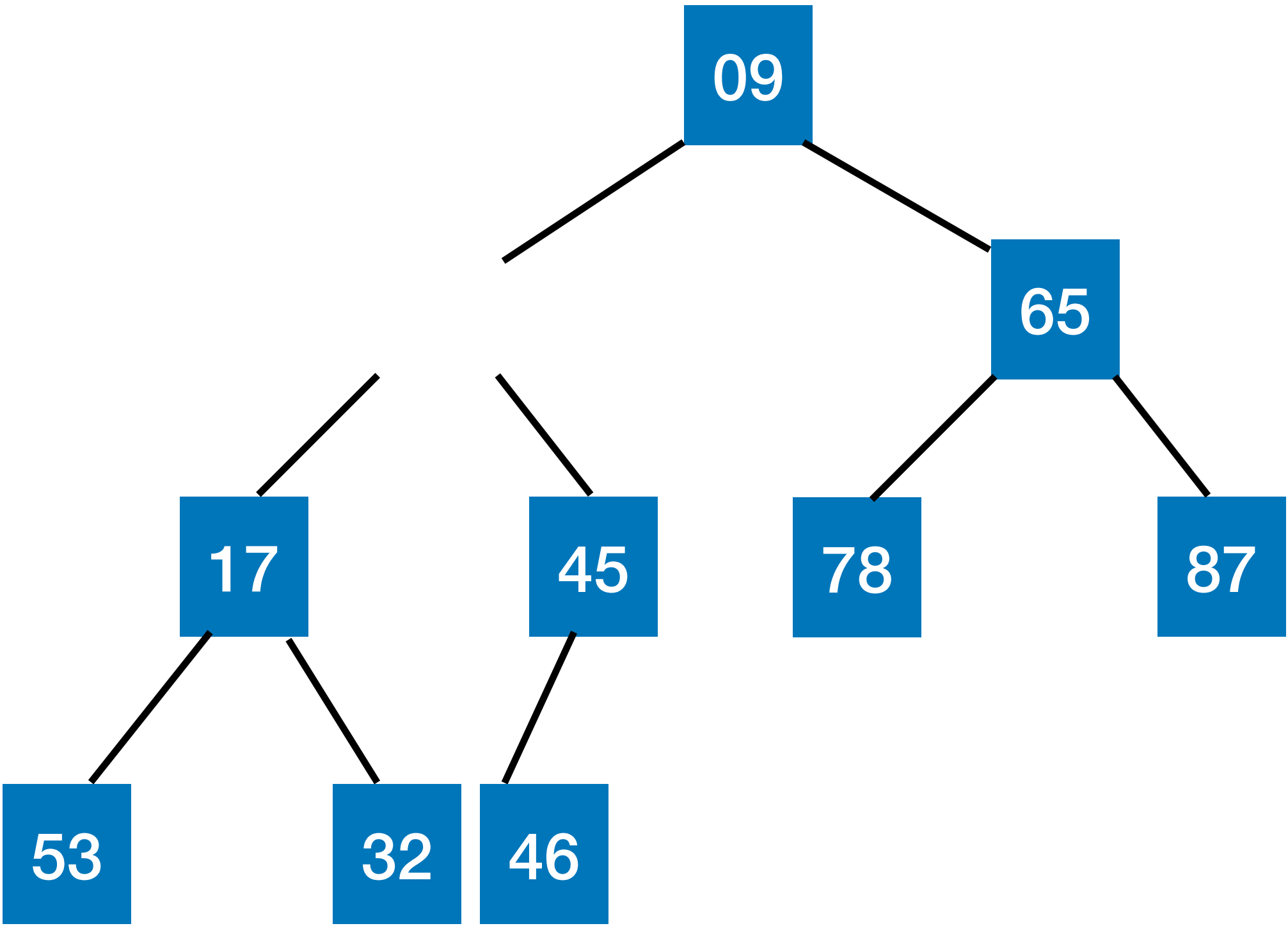
# 在堆中删除元素



小根堆



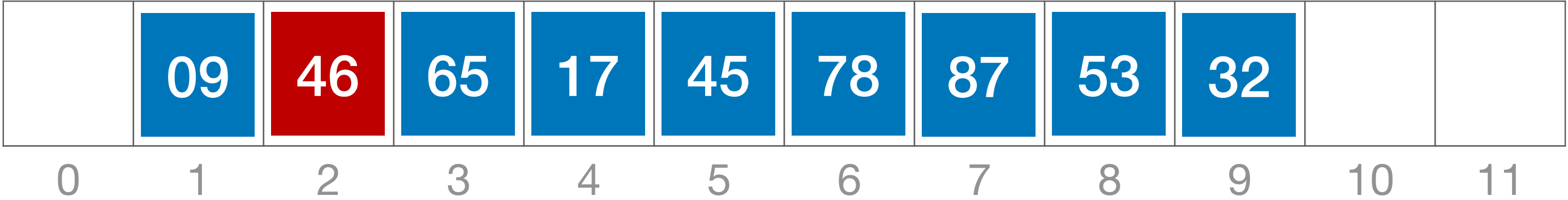
- $i$  的左孩子       $2i$
- $i$  的右孩子       $2i+1$
- $i$  的父节点       $\lfloor i/2 \rfloor$



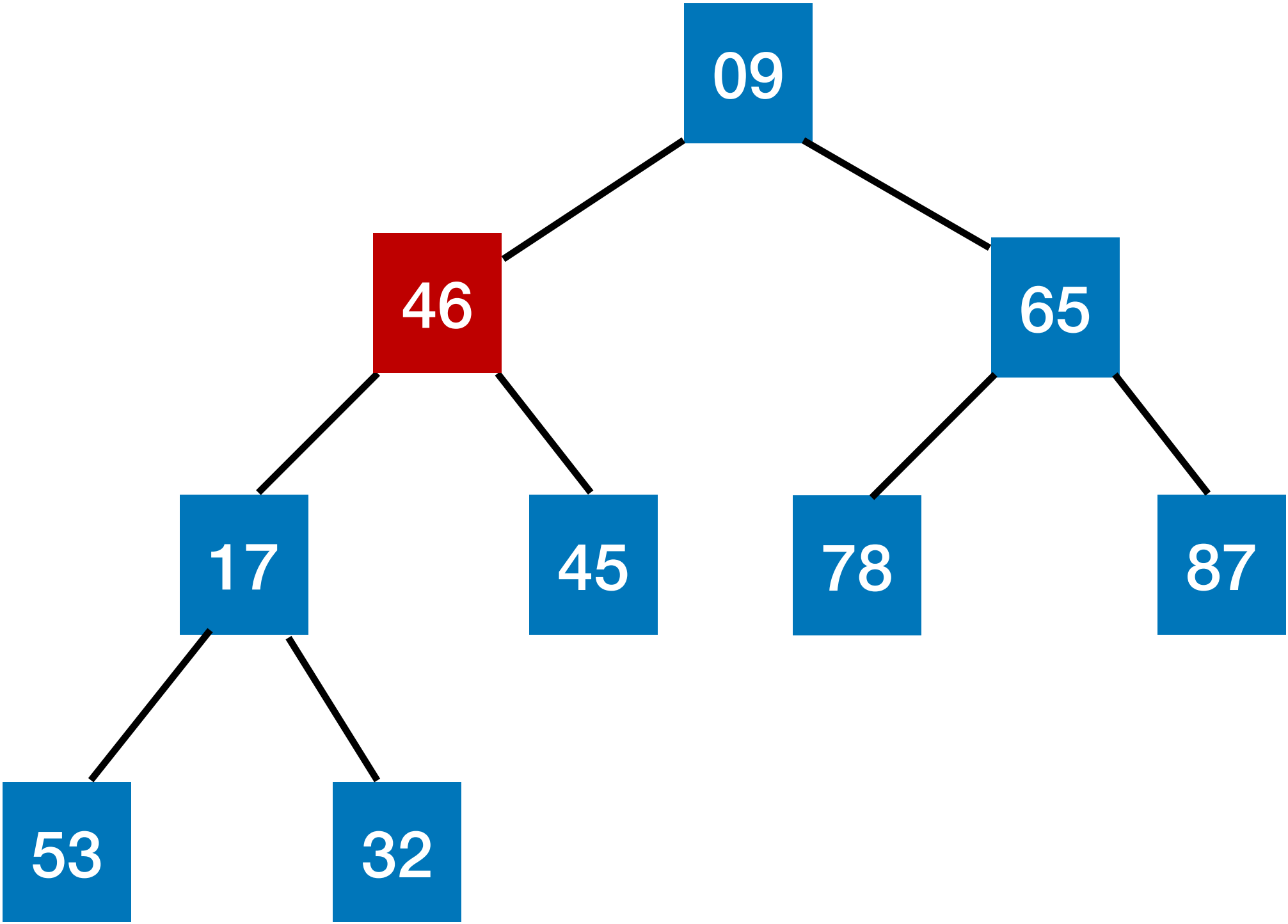
# 在堆中删除元素



小根堆



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

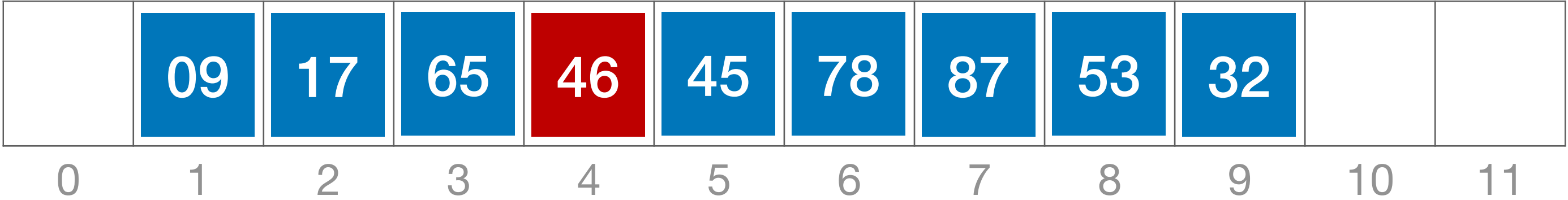


被删除的元素用堆底元素替代，然后让该元素不断“下坠”，直到无法下坠为止

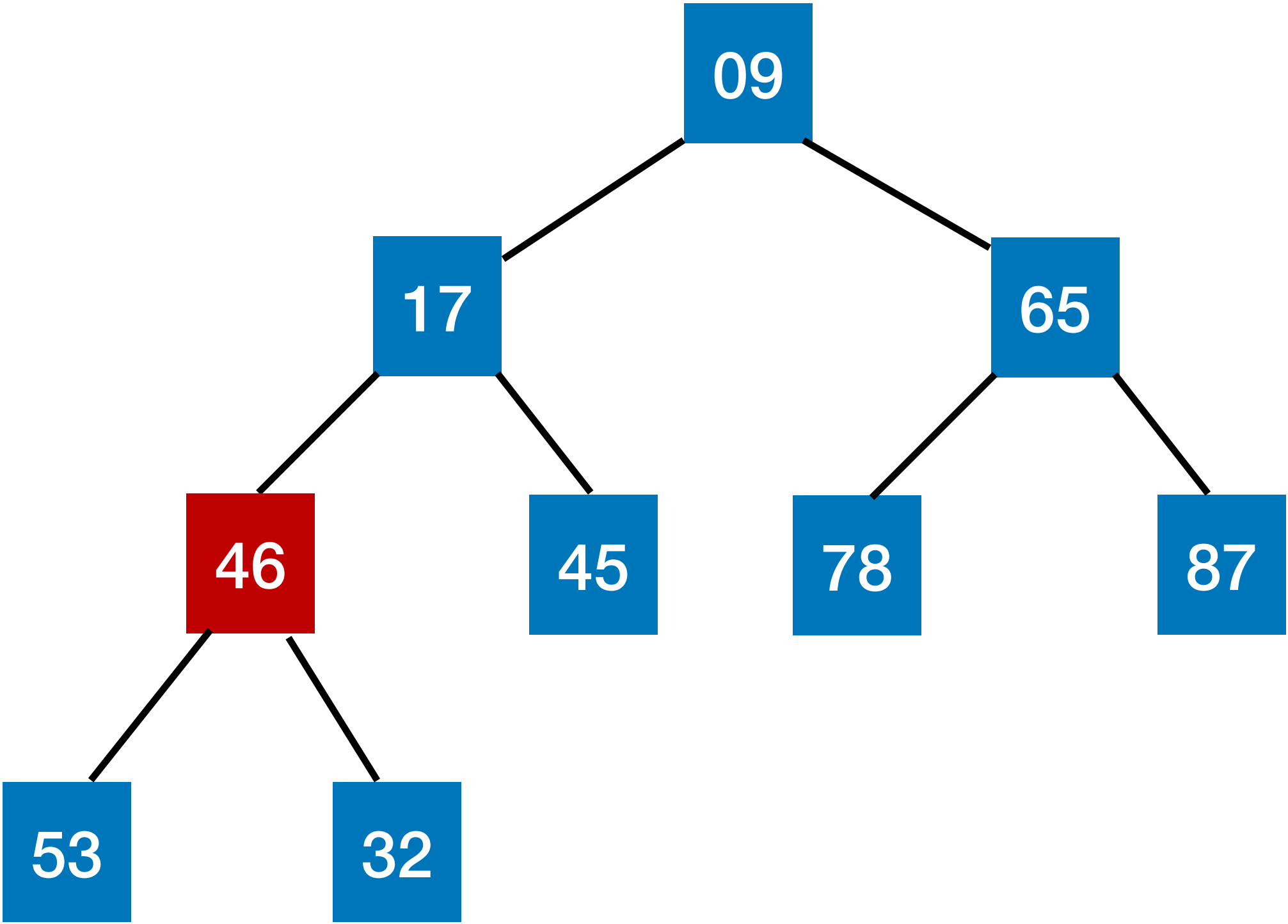
# 在堆中删除元素



小根堆



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

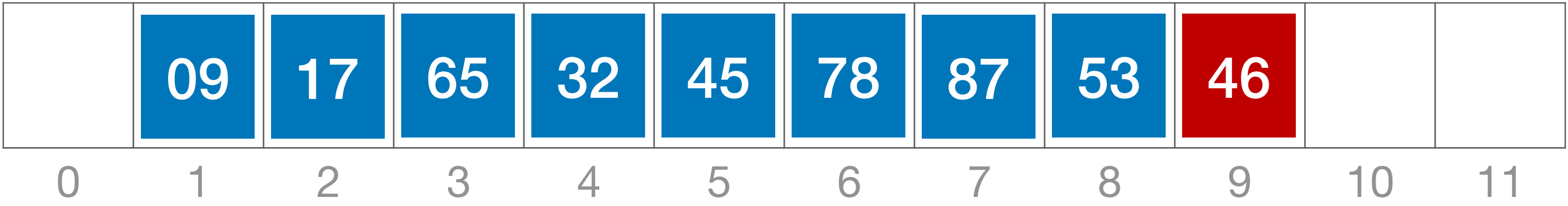


被删除的元素用堆底元素替代，然后让该元素不断“下坠”，直到无法下坠为止

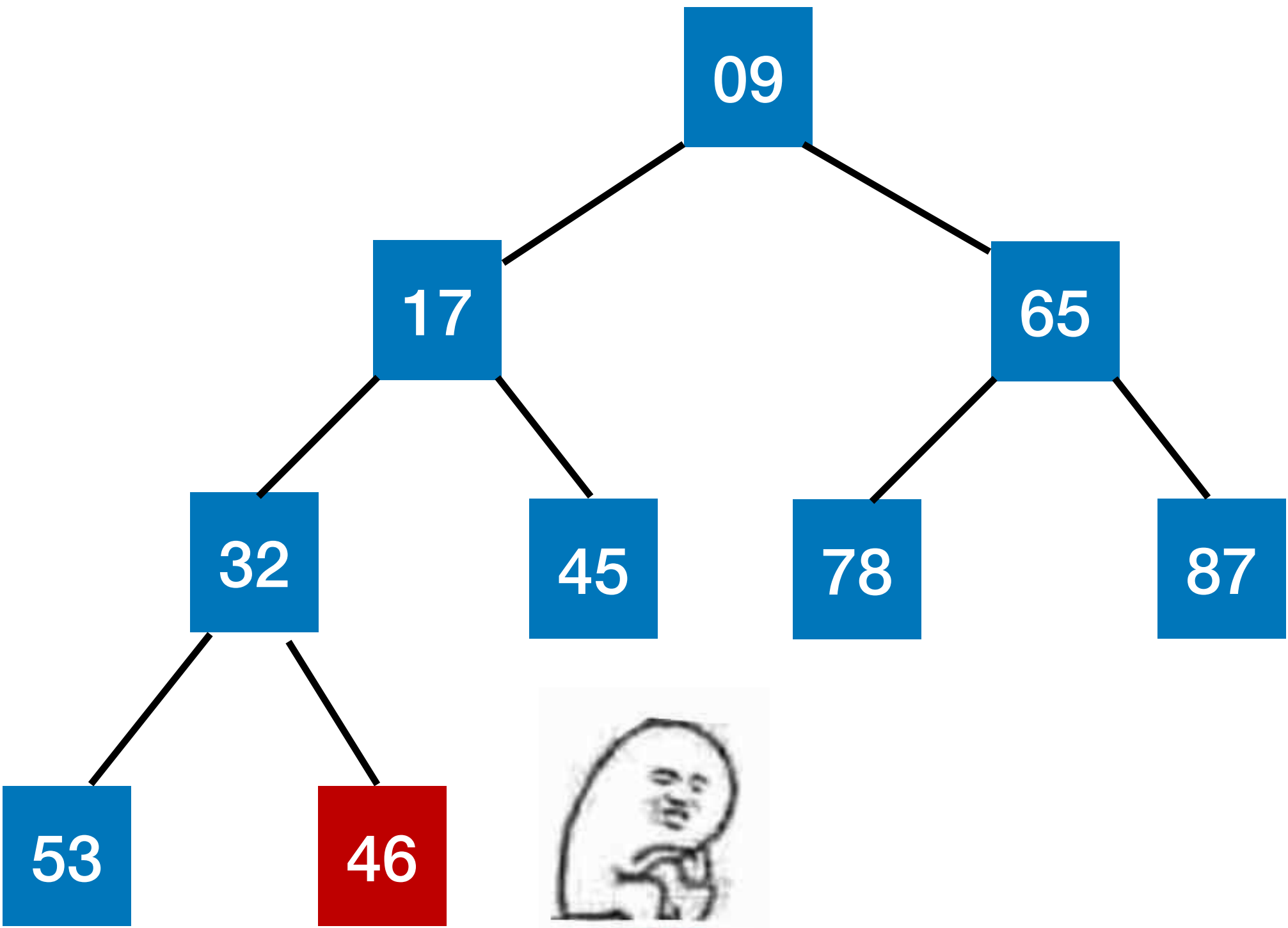
# 在堆中删除元素



小根堆



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



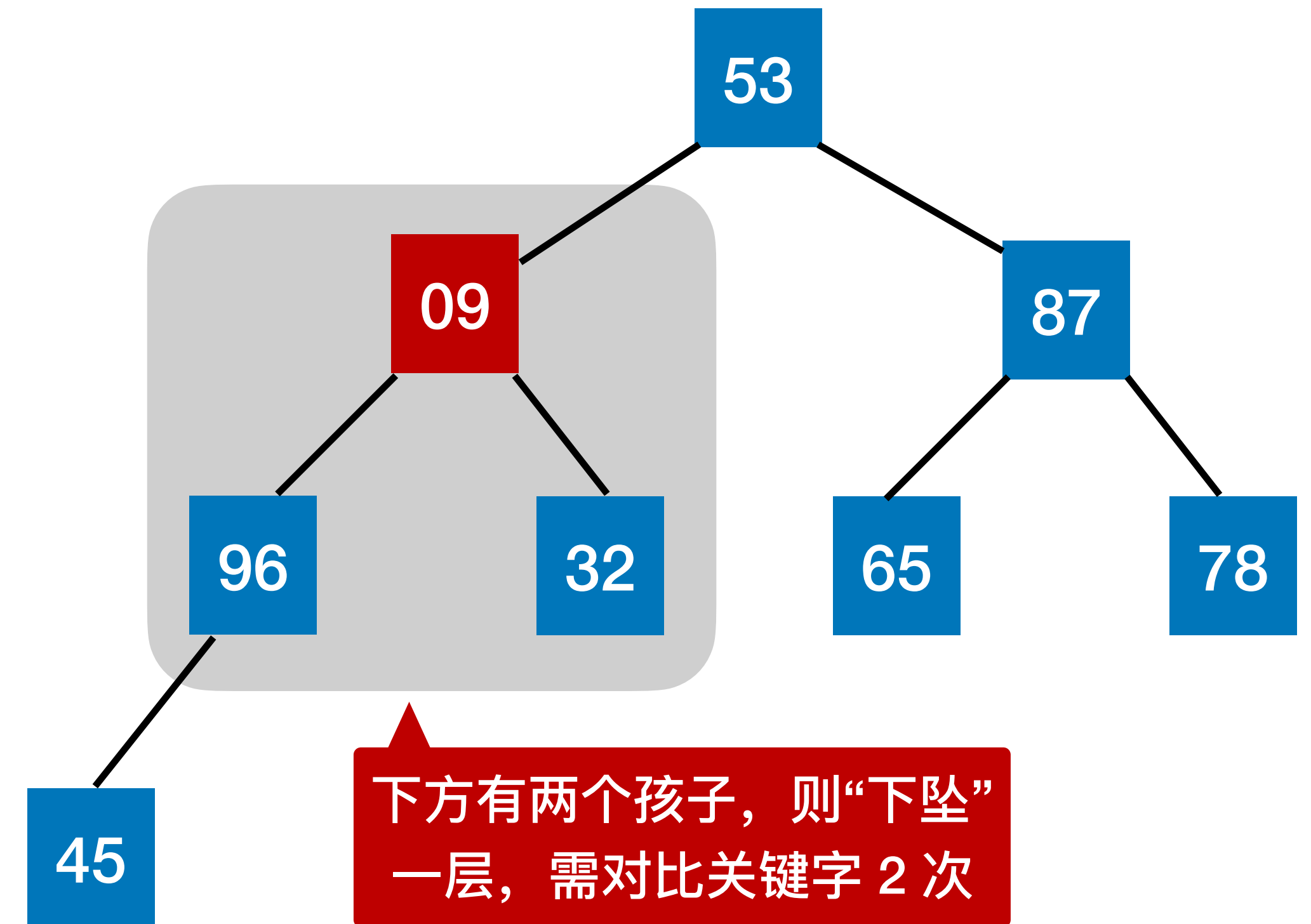
被删除的元素用堆底元素替代，然后让该元素不断“下坠”，直到无法下坠为止

对比关键字的次数 = 4次

# 上节PPT乱入

//将以  $k$  为根的子树调整为大根堆

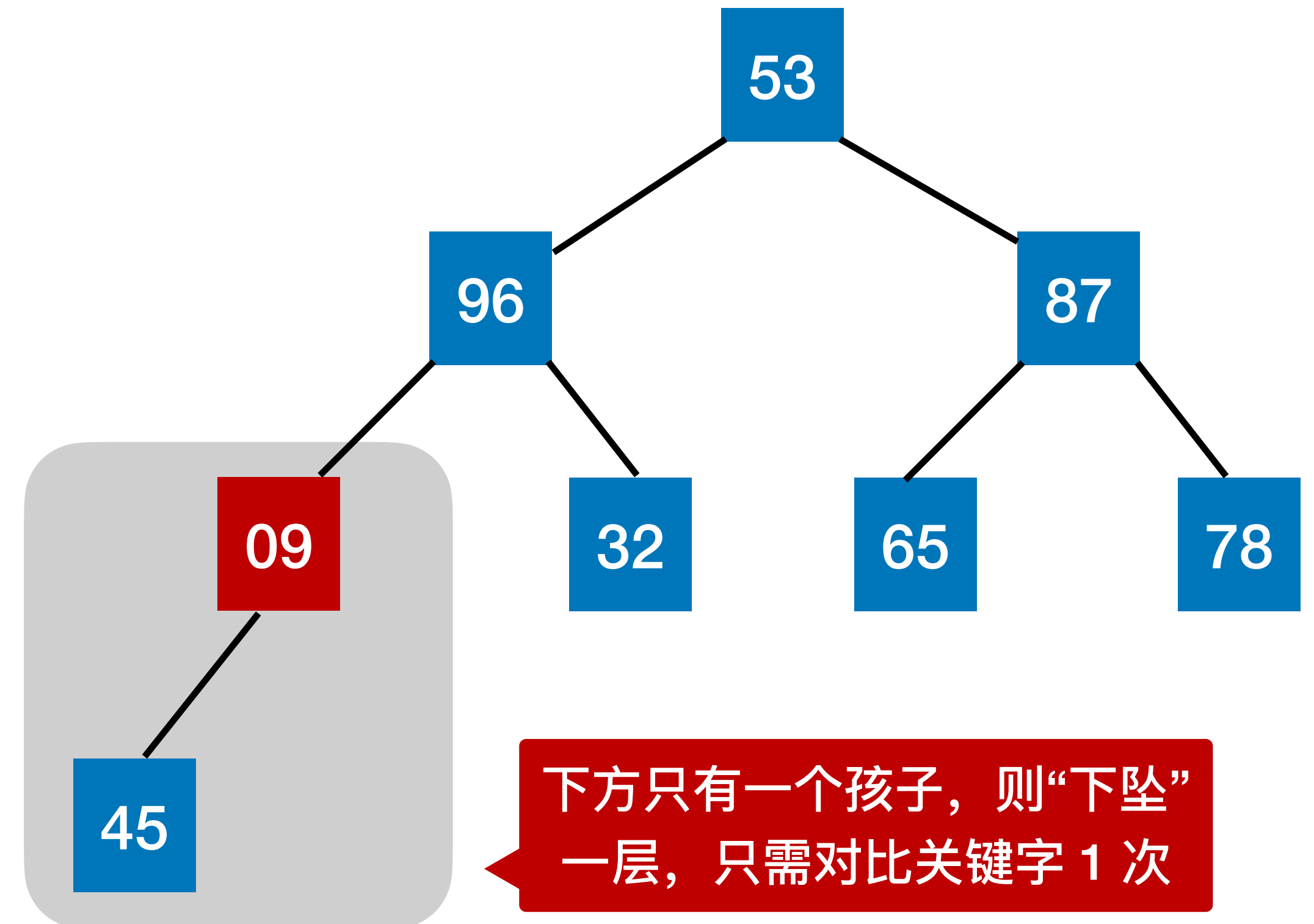
```
void HeadAdjust(int A[],int k,int len){
    A[0]=A[k];           //A[0]暂存子树的根结点
    for(int i=2*k;i<=len;i*=2){ //沿key较大的子结点向下筛选
        if(i<len&&A[i]<A[i+1])
            i++;           //取key较大的子结点的下标
        if(A[0]>=A[i]) break; //筛选结束
        else{
            A[k]=A[i];      //将A[i]调整到双亲结点上
            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
}
```



# 上节PPT乱入

//将以  $k$  为根的子树调整为大根堆

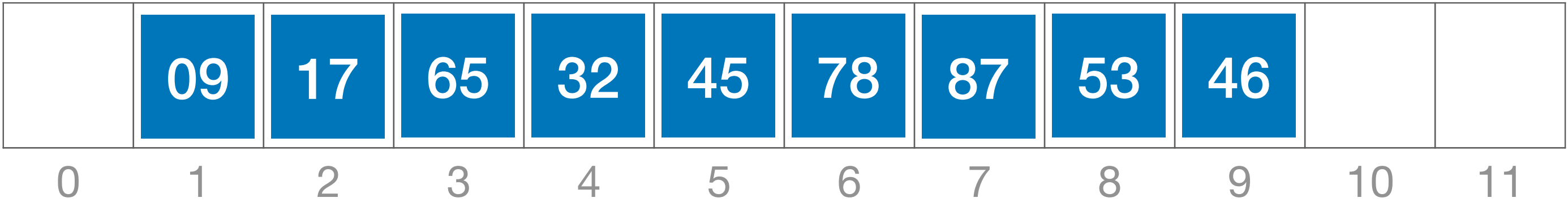
```
void HeadAdjust(int A[],int k,int len){  
    A[0]=A[k];           //A[0]暂存子树的根结点  
    for(int i=2*k;i<=len;i*=2){ //沿key较大的子结点向下筛选  
        if(i<len&&A[i]<A[i+1])  
            i++;           //取key较大的子结点的下标  
        if(A[0]>=A[i]) break; //筛选结束  
        else{  
            A[k]=A[i];       //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  
}
```



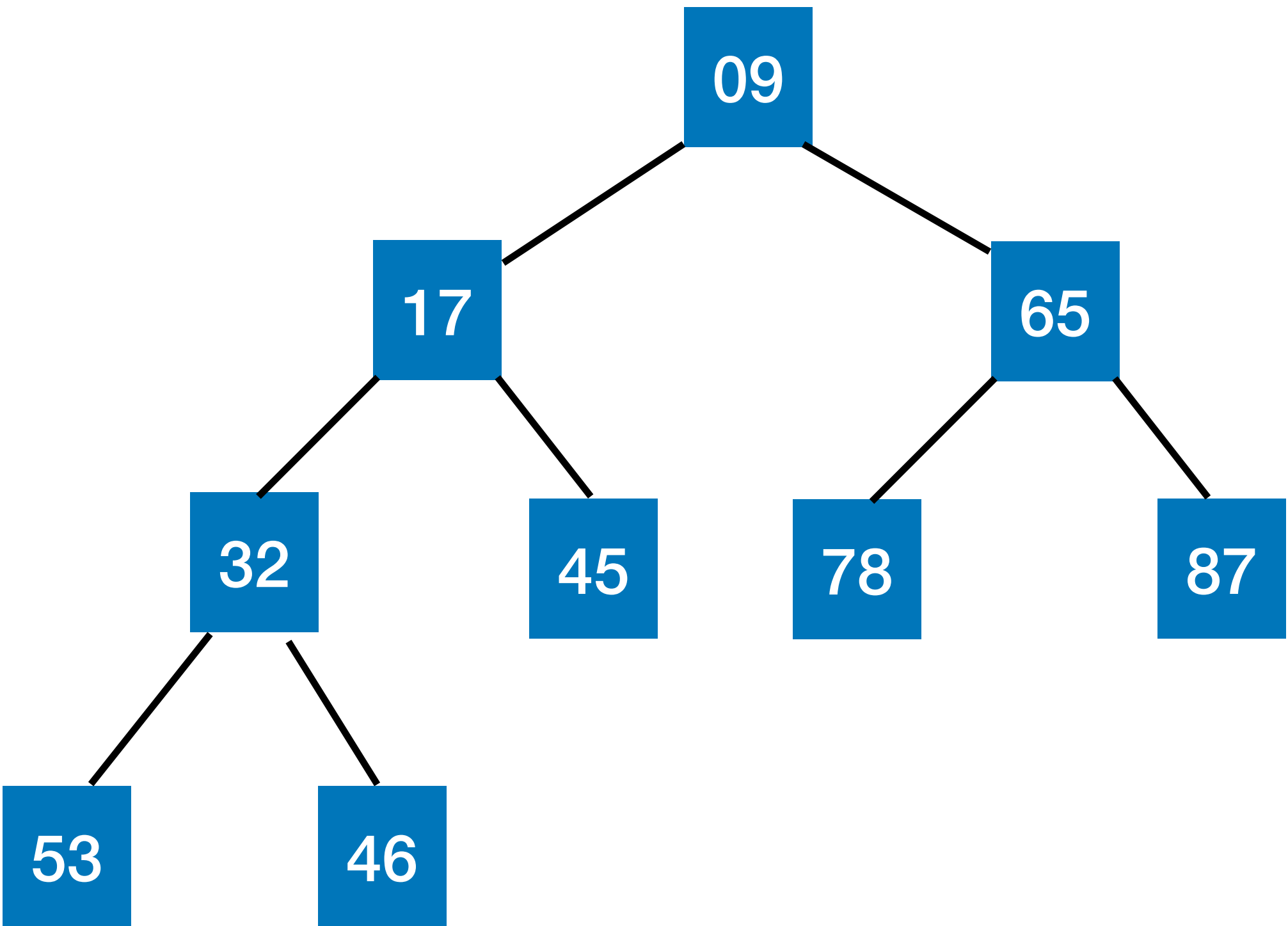
# 在堆中删除元素



小根堆



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



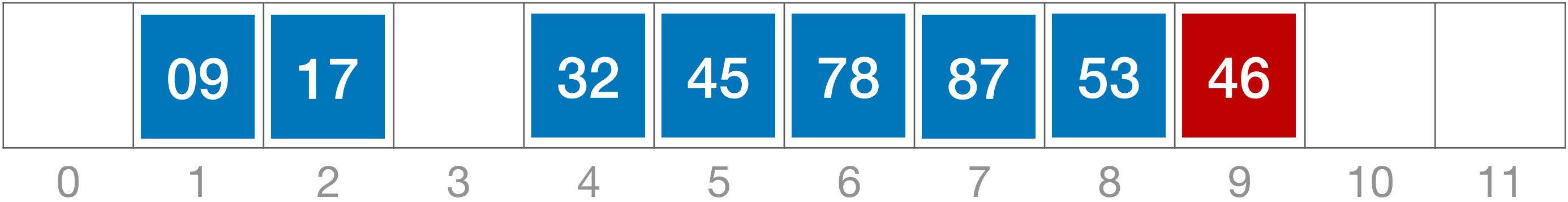
被删除的元素用堆底元素替代，然后让该元素不断“下坠”，直到无法下坠为止

对比关键字的次数 = 4次

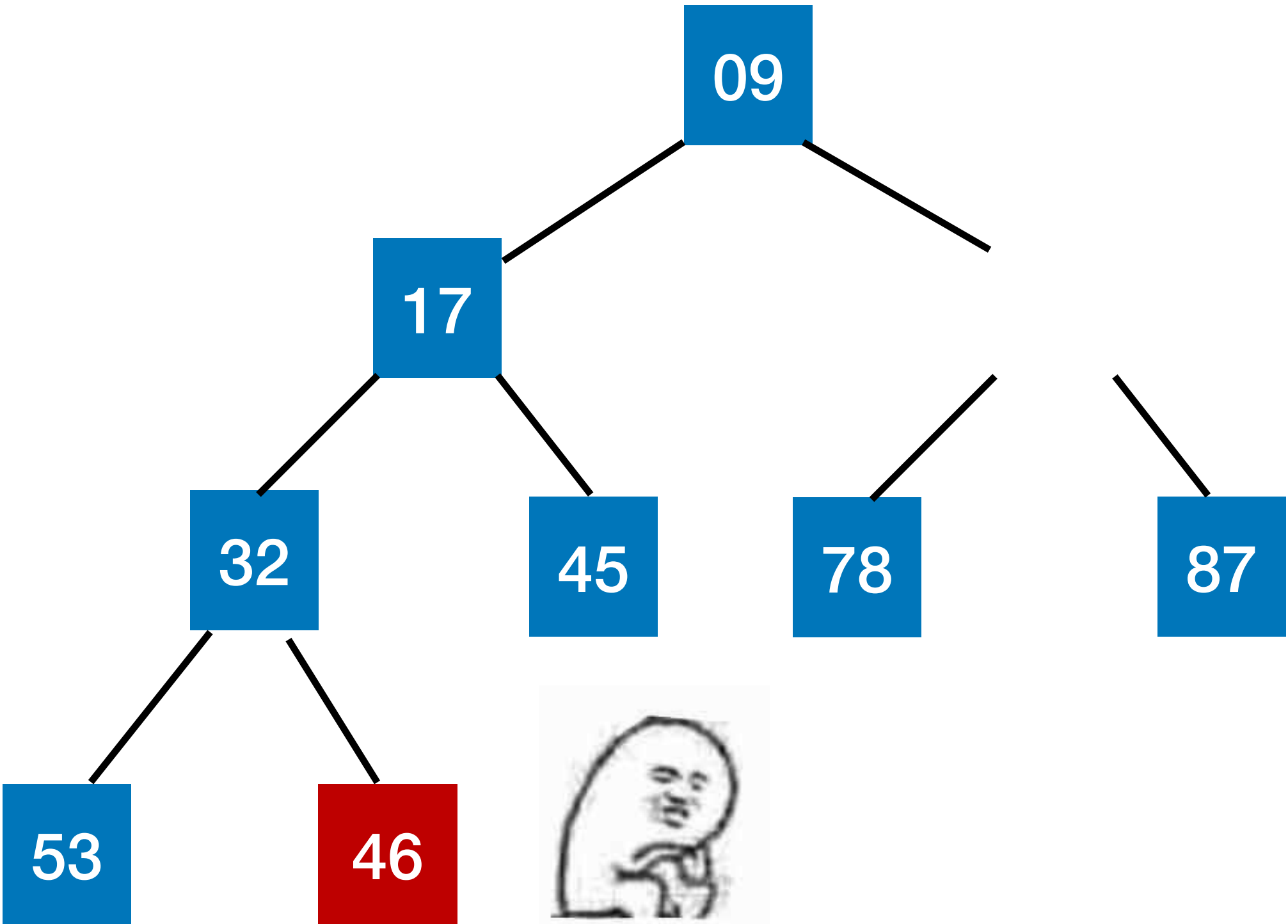
# 在堆中删除元素



小根堆



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



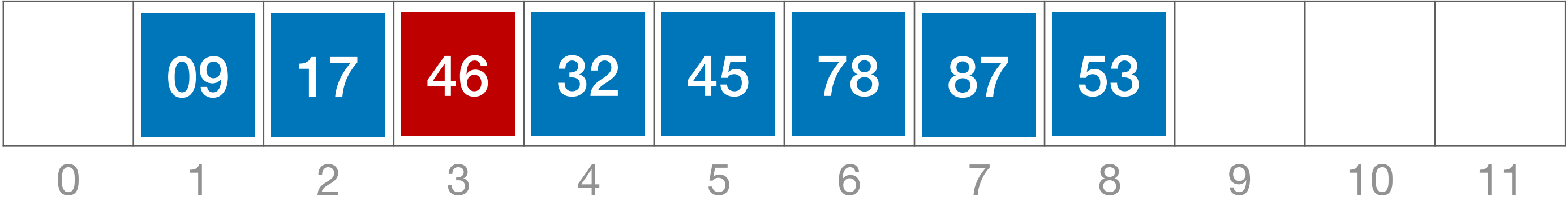
被删除的元素用堆底元素替代，然后让该元素不断“下坠”，直到无法下坠为止



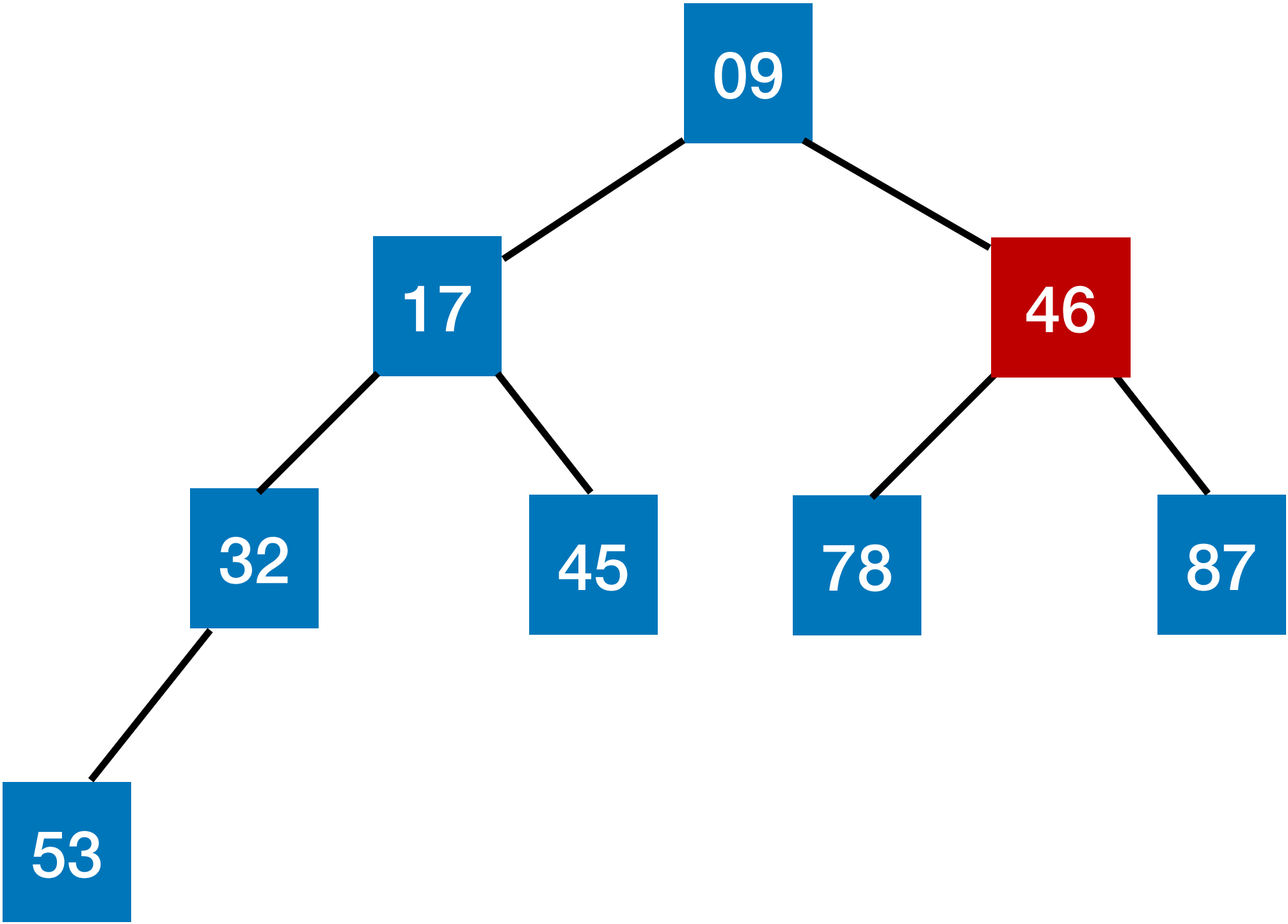
# 在堆中删除元素



小根堆



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

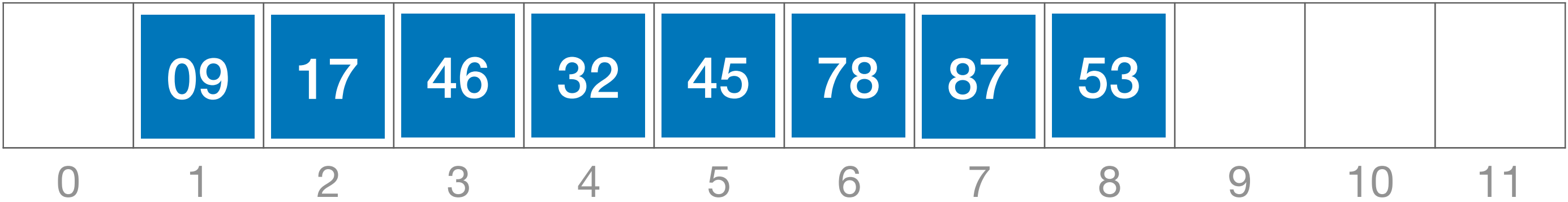


被删除的元素用堆底元素替代，然后让该元素不断“下坠”，直到无法下坠为止

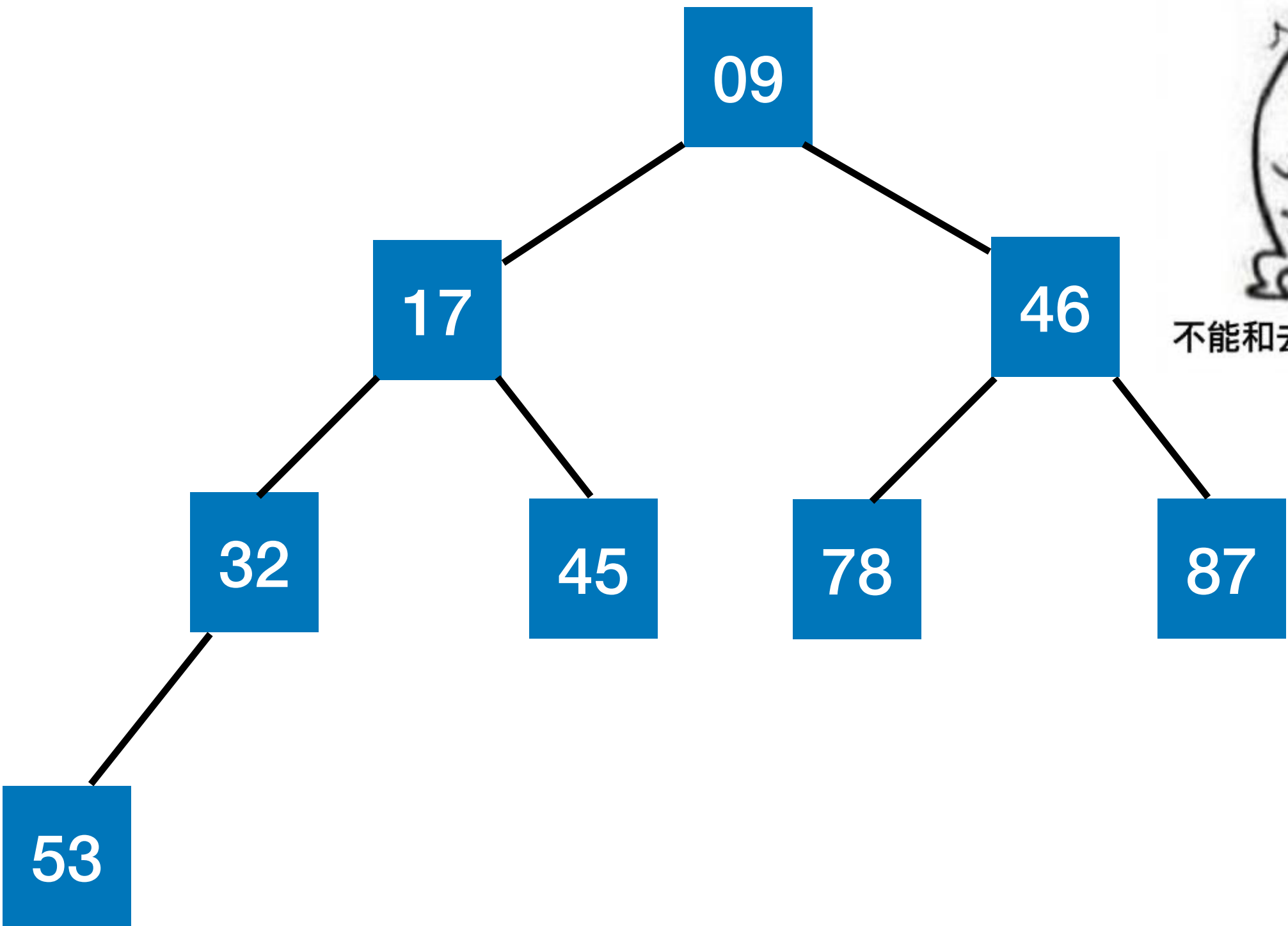
# 在堆中删除元素



小根堆



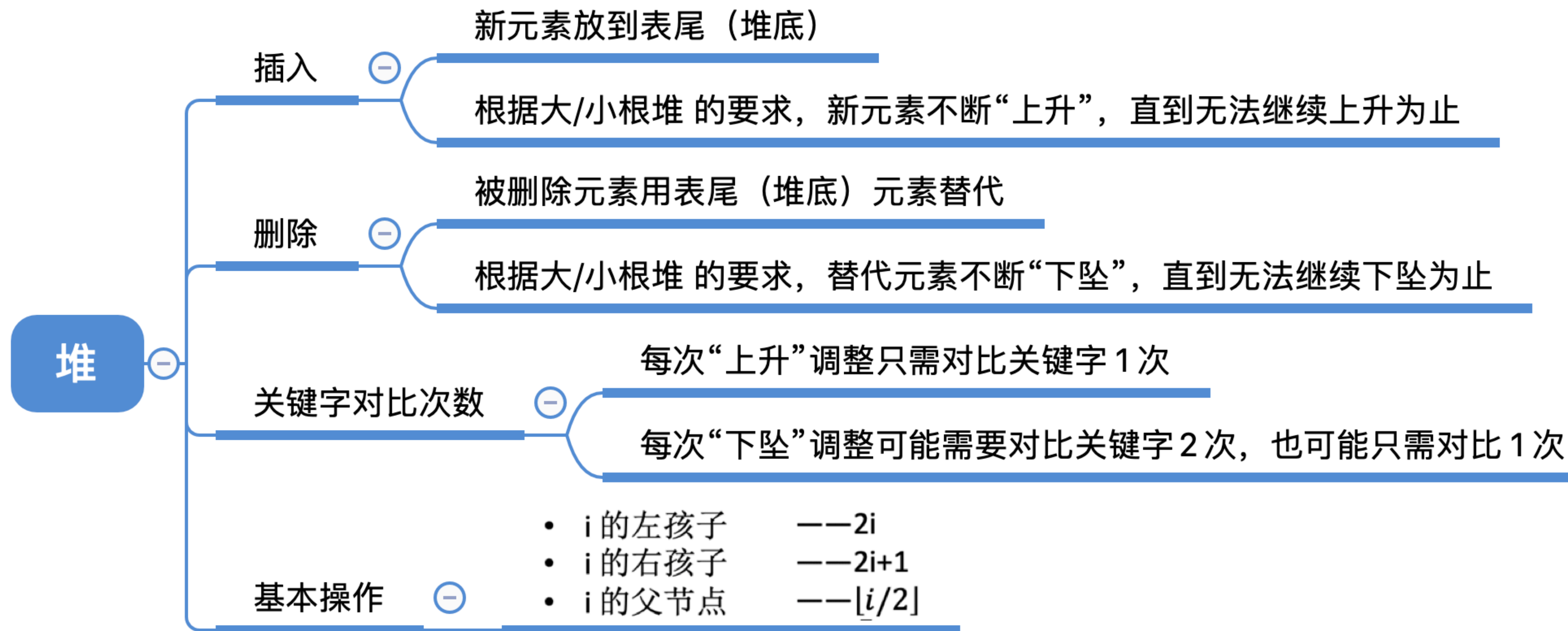
- $i$  的左孩子       $2i$
- $i$  的右孩子       $2i+1$
- $i$  的父节点       $\lfloor i/2 \rfloor$



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

对比关键字的次数 = 2次

# 知识回顾与重要考点

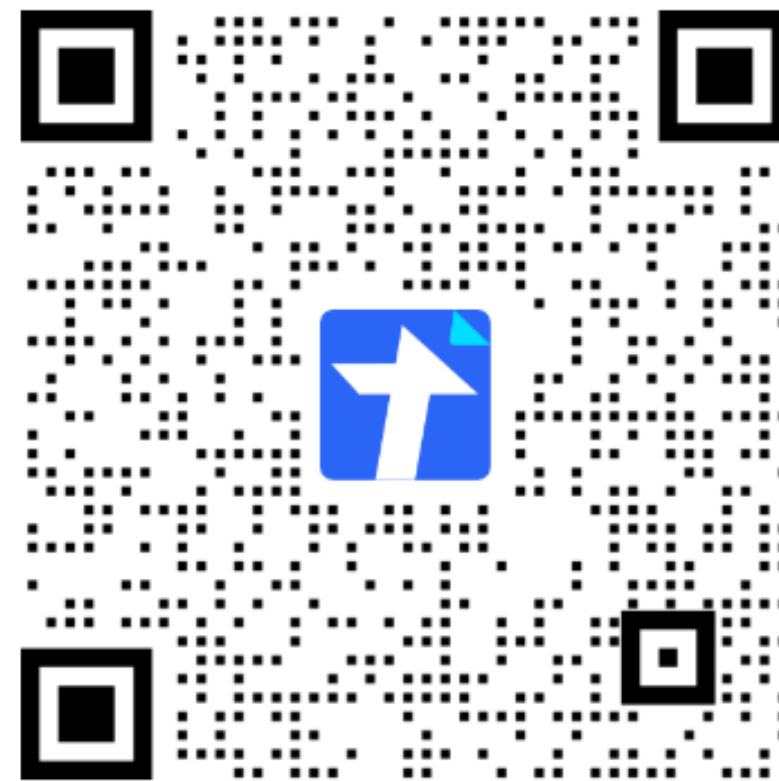


# 欢迎大家对本节视频进行评价~



学员评分：8.4.2\_2 堆...

扫一扫二维码打开或分享给好友



— 腾讯文档 —

可多人实时在线编辑，权限安全可控



公众号：王道在线



b站：王道计算机教育



抖音：王道计算机考研