

# 位置相关服务（LBS）： 基于网格索引的查询

2017年

# 第三章 基于网格索引的查询



## 最近邻查询

---

# 堆

## ❖ 堆的定义

- 堆是满足下列性质的数据序列 $\{r_1, r_2, \dots, r_n\}$ ：

完全二叉树的性质？

$$\begin{cases} r_{2i} \geq r_i \\ r_{2i+1} \geq r_i \end{cases} \text{ (小顶堆)} \quad \text{或} \quad \begin{cases} r_{2i} \leq r_i \\ r_{2i+1} \leq r_i \end{cases} \text{ (大顶堆)}$$

$\{12, 36, 27, 65, 40, 34, 98, 81, 73, 55, 49\}$

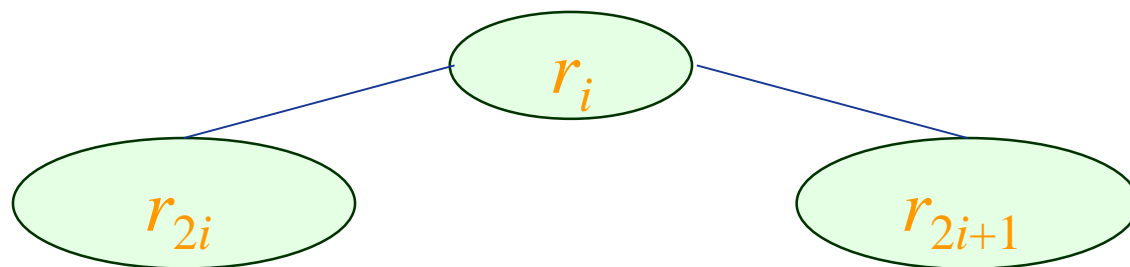
是小顶堆

$\{12, 36, 27, 65, 40, 14, 98, 81, 73, 55, 49\}$

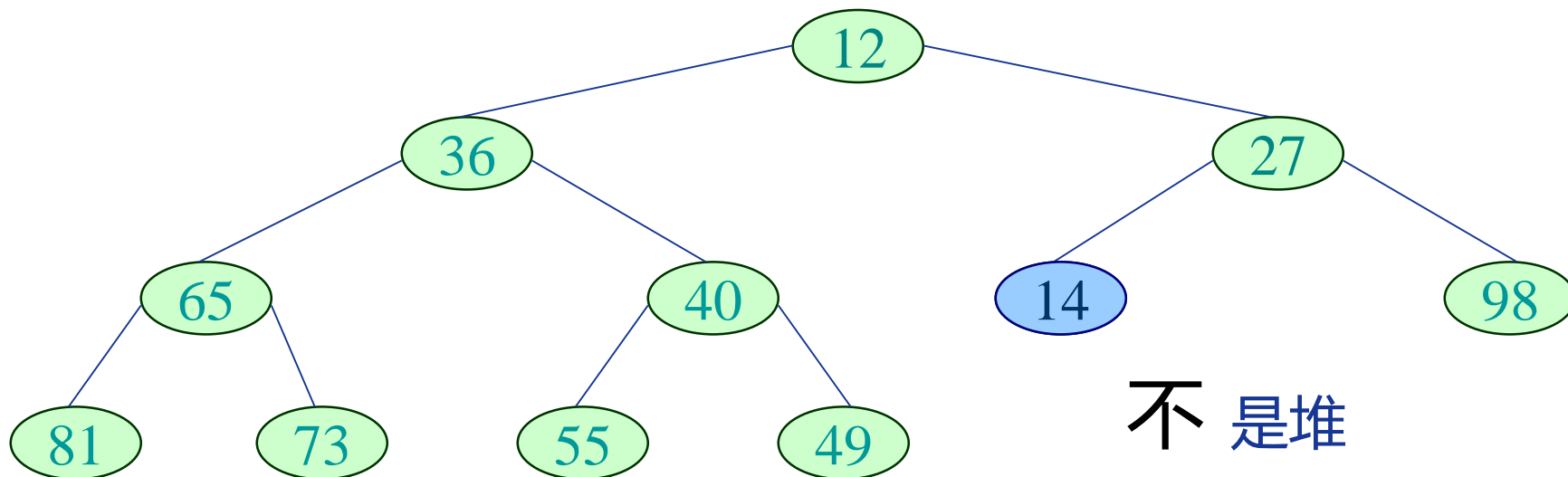
不是堆

# 堆

若将该数列视作完全二叉树，则  $r_{2i}$  是  $r_i$  的左孩子； $r_{2i+1}$  是  $r_i$  的右孩子。



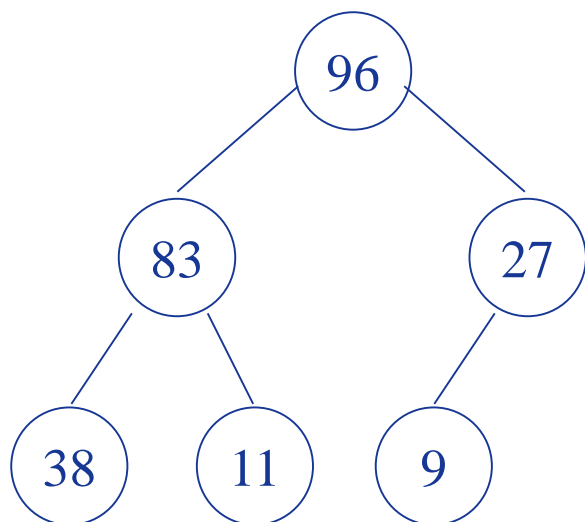
例如: {12, 36, 27, 65, 40, 34, 98, 81, 73, 55, 49}



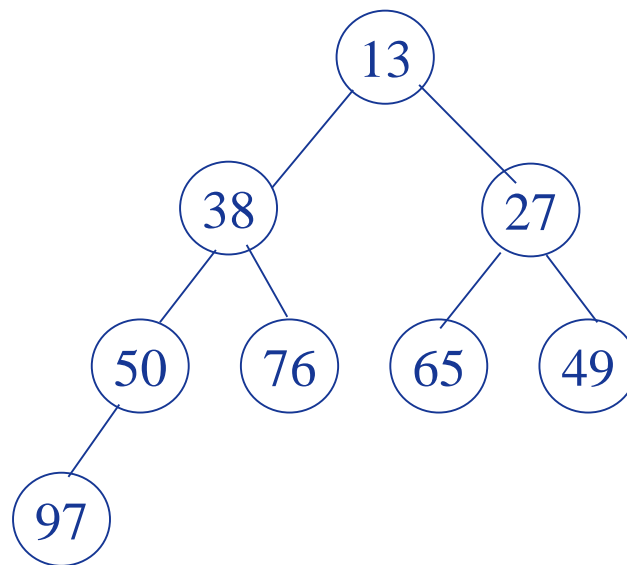
不是堆

# 大顶堆和小顶堆

例 ( 96 , 83 , 27 , 38 , 11 , 9 )



例 ( 13 , 38 , 27 , 50 , 76 , 65 , 49 , 97 )



堆序列是完全二叉树，则堆顶元素（完全二叉树的根）必为序列中n个元素的最小值或最大值

# 堆排序

## ❖ 堆排序

- 堆排序即是利用堆的特性对记录序列进行排序的一种排序方法

## ❖ 堆排序的过程

- 对 $n$ 个数据的原始无序序列建堆，输出堆顶元素
- 将剩下的 $n-1$ 个元素调整为堆，输出堆顶元素

堆排序需解决的两个问题：

- 如何由一个无序序列建成一个堆？
- 如何在输出堆顶元素之后，调整剩余元素，使之成为一个新的堆？  
→ 筛选

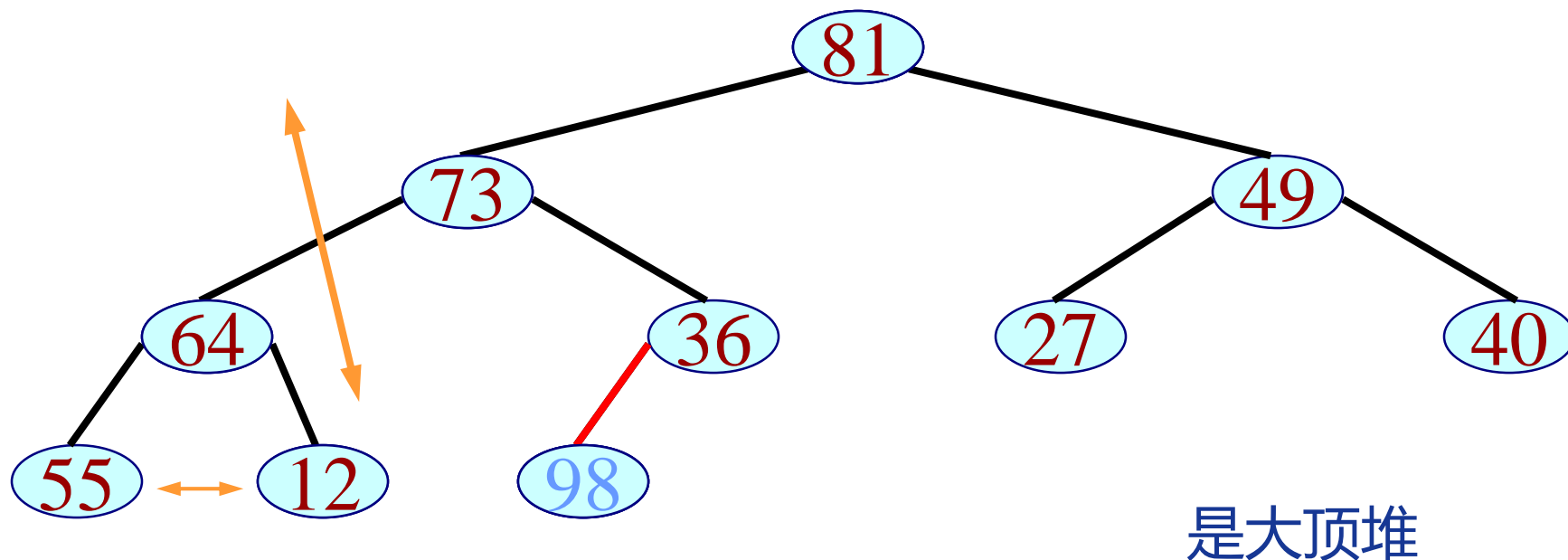
# 筛选

## ❖ 第二个问题解决方法——筛选

- 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”根结点使整个二叉树也成为堆
- 方法（小顶堆）：输出堆顶元素之后，以堆中最后一个元素替代之；然后将根结点值与左、右子树的根结点值进行比较，并与其中小者进行交换；重复上述操作，直至叶子结点，将得到新的堆，称这个从堆顶至叶子的调整过程为“筛选”

# 筛选

大顶堆调节过程：



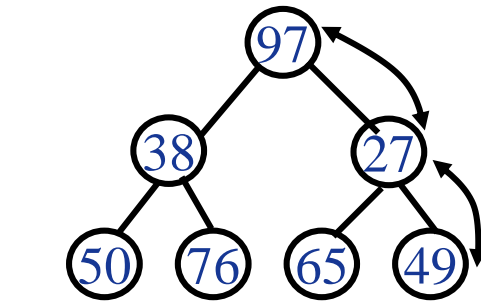
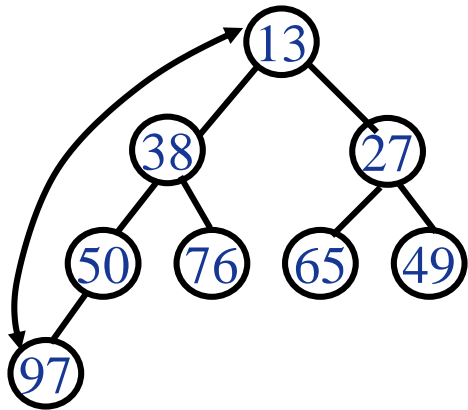
- 输出98 ( 98 和 12 进行互换 ) 之后 , 它就不是堆了
- 需要对它进行 “筛选”



# 堆排序

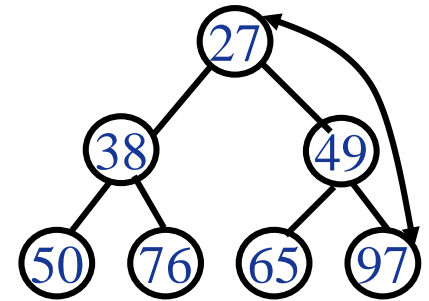
例：小顶堆

( 13 , 38 , 27 , 50 , 76 , 65 , 49 , 97 )



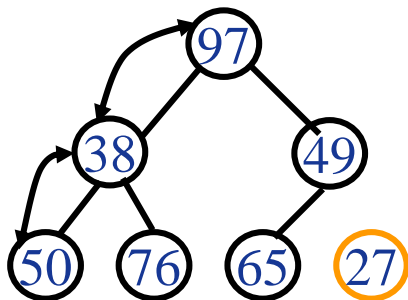
13

输出：13



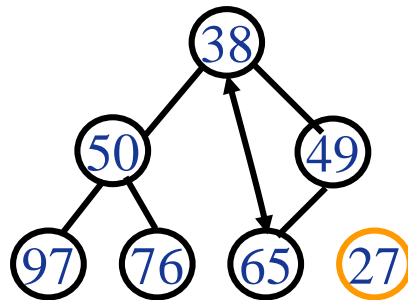
13

输出：13



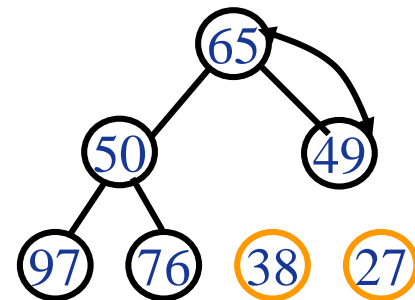
13

输出：13 27



13

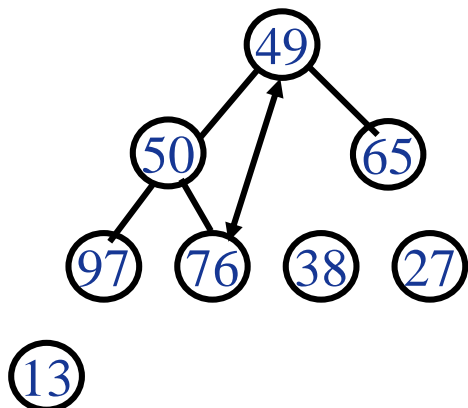
输出：13 27



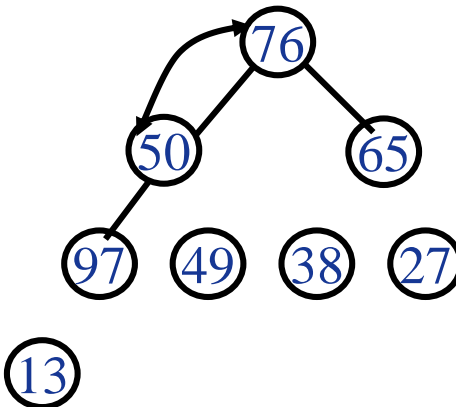
13

输出：13 27 38

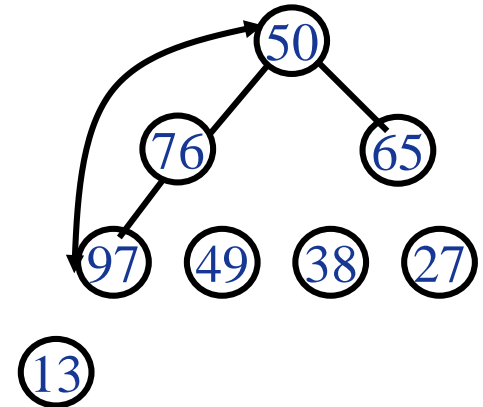
# 堆排序



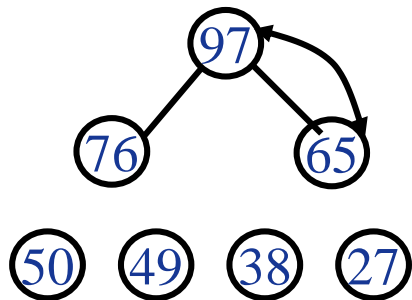
输出 : 13 27 38



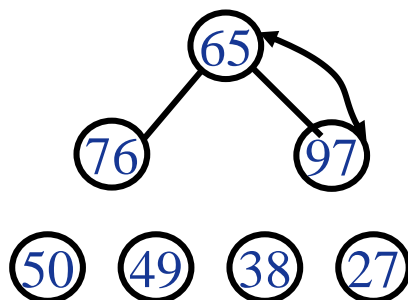
输出 : 13 27 38 49



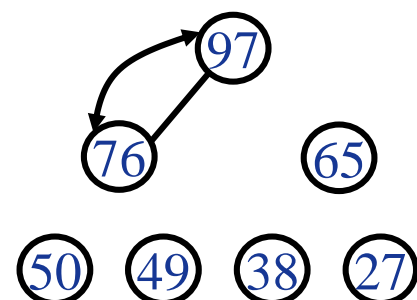
输出 : 13 27 38 49



输出 : 13 27 38 49 50

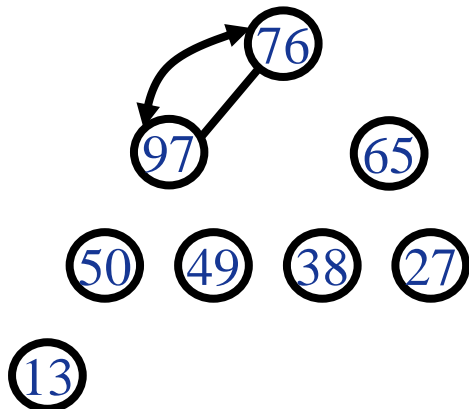


输出 : 13 27 38 49 50

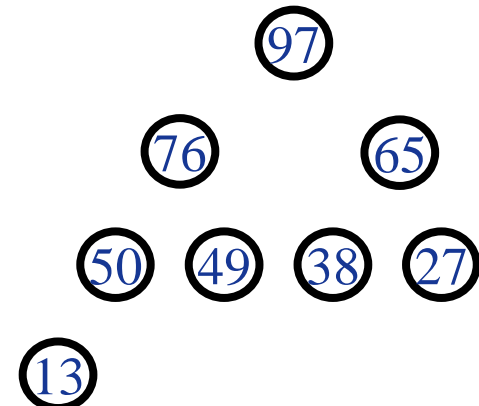


输出 : 13 27 38 49 50 65

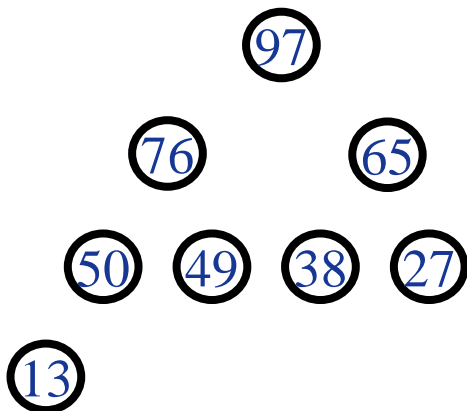
# 堆排序



输出 : 13 27 38 49 50 65



输出 : 13 27 38 49 50 65 76



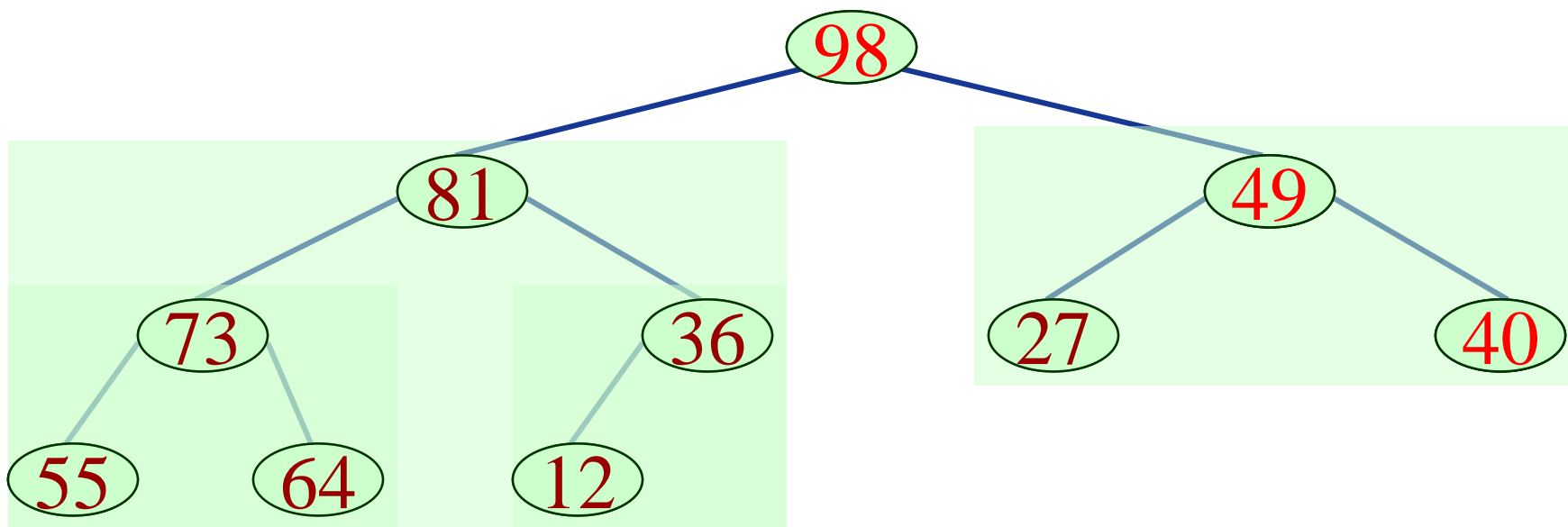
输出 : 13 27 38 49 50 65 76 97

# 创建堆

建堆是一个从下往上进行“筛选”的过程。

例如: 原始序列为 ( 40,55,49,73,12,27,78,81,64,36 )

从第一个非叶节点开始调节



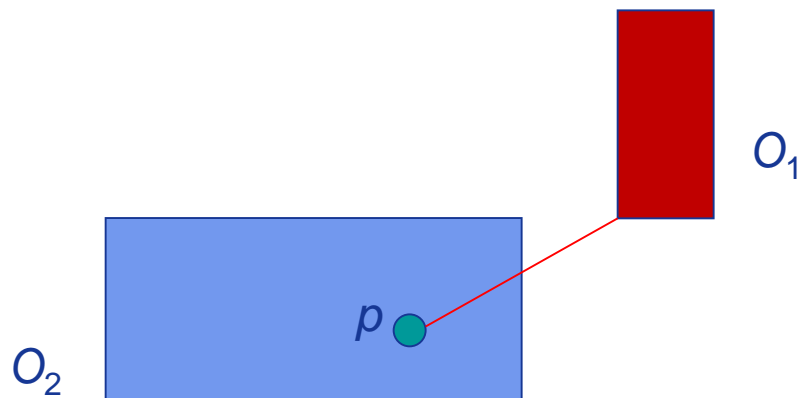
- 左/右子树都已经调整为堆
- 最后只要调整根结点，使整个二叉树是个“堆”即可。

# 预备知识1:MinDist

$P$ : 给定的查询点       $O$ :  $R$ 树结点中的MBR

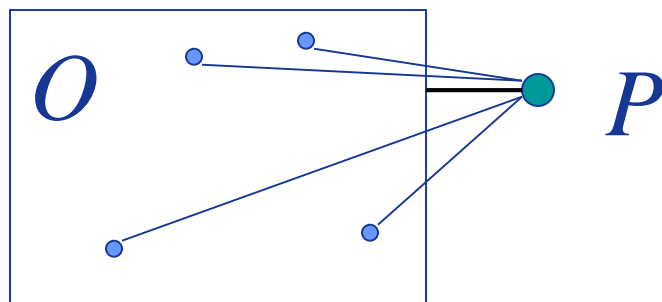
## ❖ 定义1：最小距离MinDist

- 如果点 $P$ 在 $O$ 内部，最小距离为0
- 如果点 $P$ 在 $O$ 外部，最小距离为 $P$ 同 $O$ 最近的一条边之间的距离



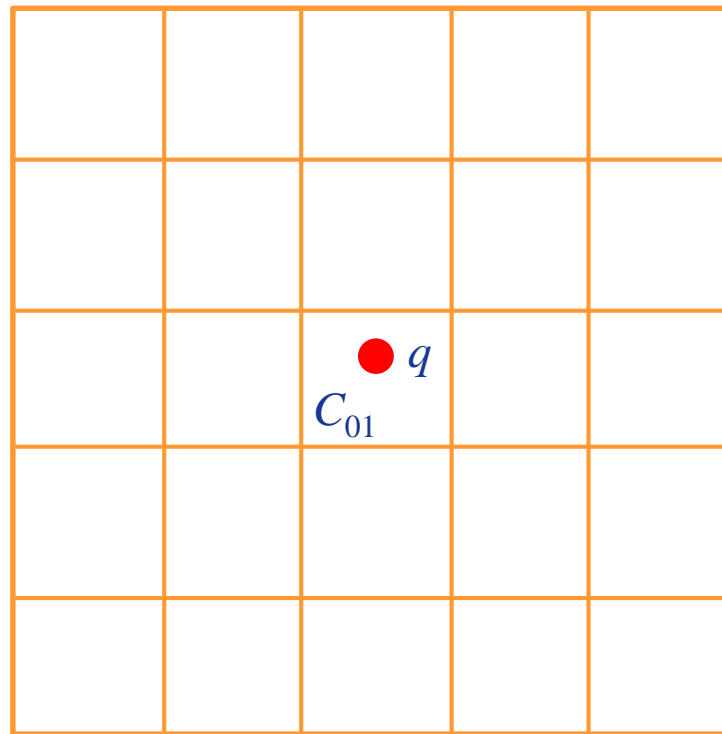
# 预备知识1:MinDist

- ❖ 推论1 : MinDist等于点 $P$ 与 $O$ 的边界上任意点之间的最小距离
- ❖ 定理1 :  $P$ 与 $O$ 之间的MinDist小于或等于 $P$ 同 $O$ 中可能包含的所有数据之间的距离



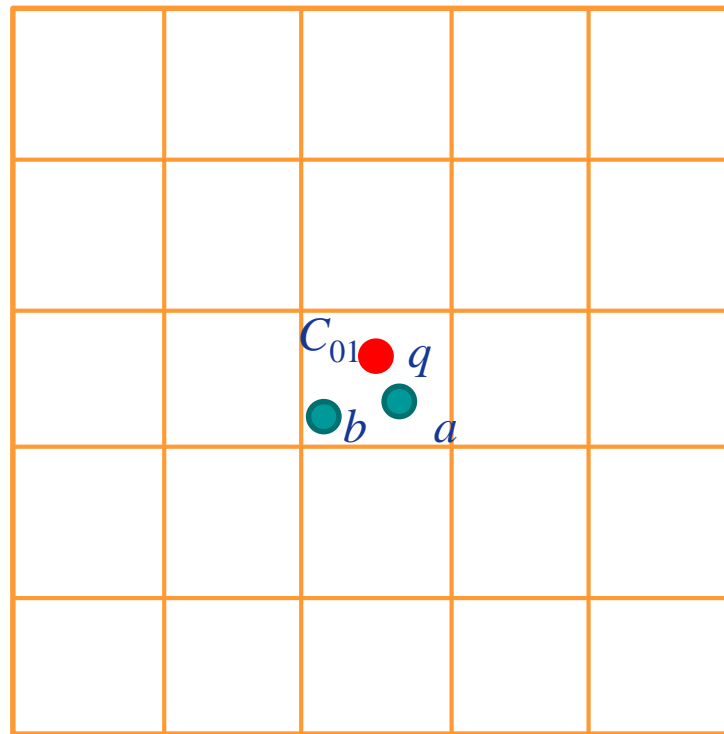
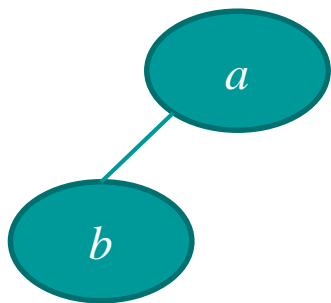
# 最近邻查询

- ❖ 计算查询点 $q$ 所在的网格  $C_{01}$
- ❖ 计算 $q$ 与 $C_{01}$ 的MinDist, 为0
- ❖ 依据 $q$ 与 $C_{01}$ 的距离, 创建小顶堆



# 最近邻查询

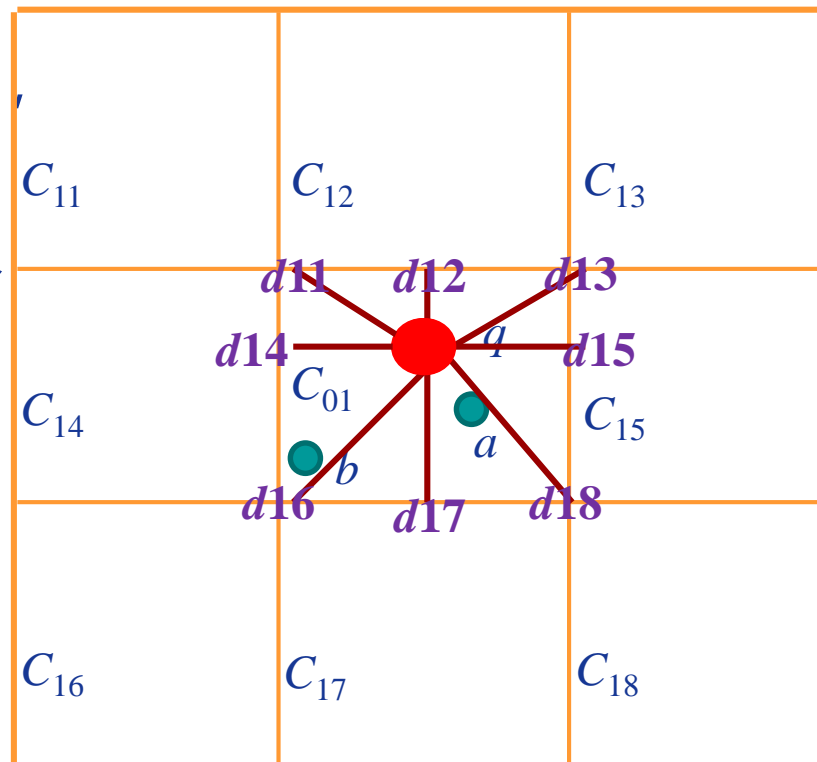
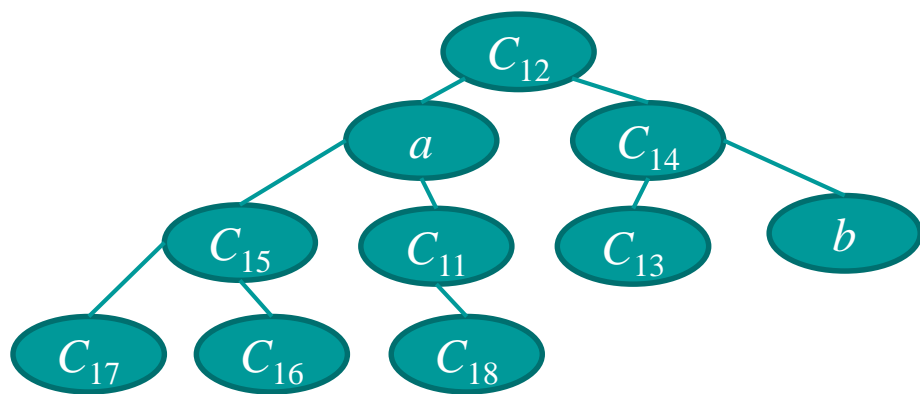
- ❖ 将堆顶元素弹出( $C_{01}$ )，若为网格
- ❖ 则找出网格对应的空间点
- ❖ 依据 $q$ 与空间点的距离，将空间点插入小顶堆中，并调整小顶堆





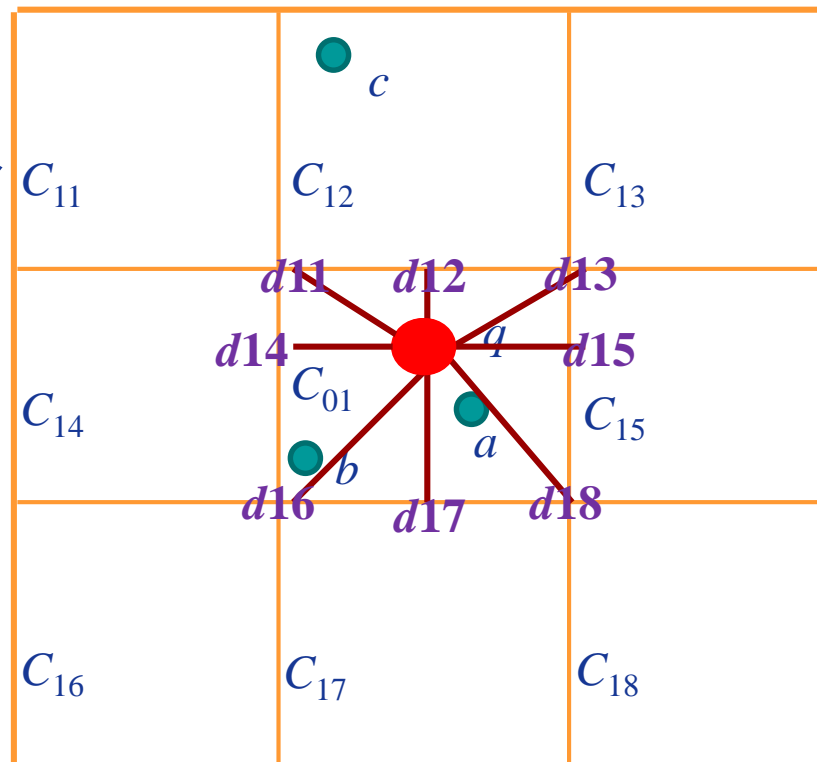
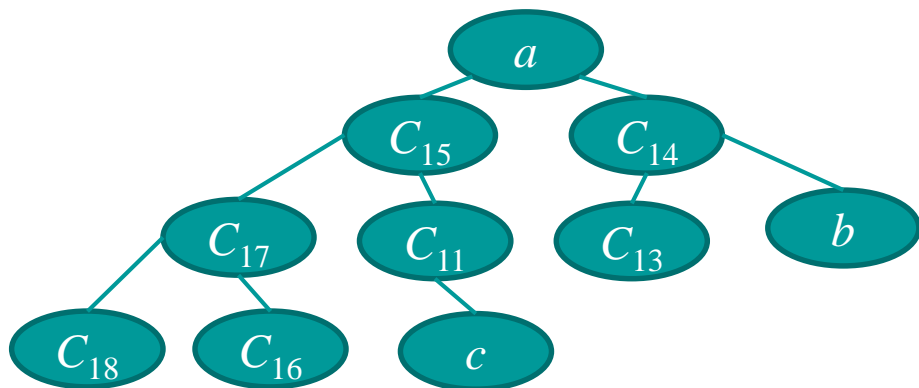
# 最近邻查询

- ❖ 计算以被删网格( $C_{01}$ )为中心的周围8个网格 ( $C_{11}, C_{12}, \dots, C_{18}$ )
- ❖ 计算 $q$ 与网格 $C_{11}, C_{12}, \dots, C_{18}$ 的MinDist, 分别为 $d_{11}, d_{12}, \dots, d_{18}$
- ❖ 依据最小距离, 将 $C_{11}, C_{12}, \dots, C_{18}$ 插入小顶堆中, 并调整小顶堆



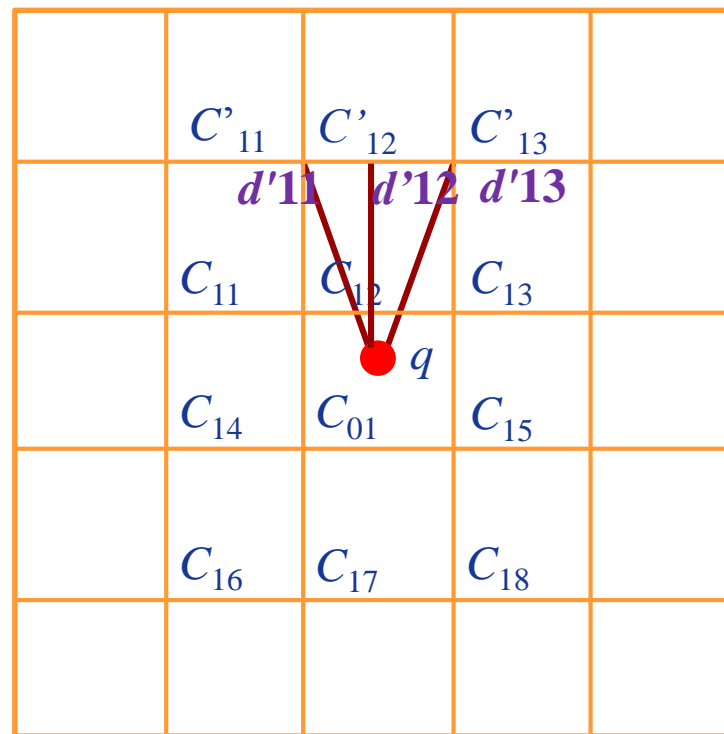
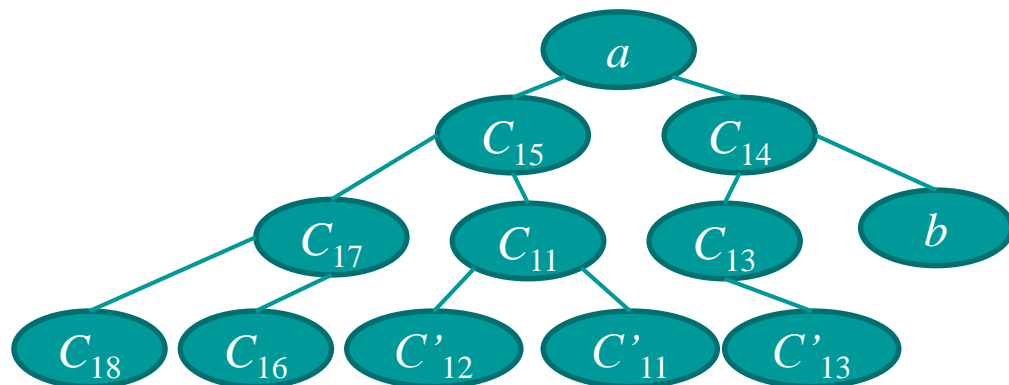
# 最近邻查询

- ❖ 将堆顶元素弹出( $C_{12}$ )，若为网格
- ❖ 则找出网格对应的空间点
- ❖ 依据 $q$ 与空间点的距离，将空间点插入小顶堆中，并调整小顶堆



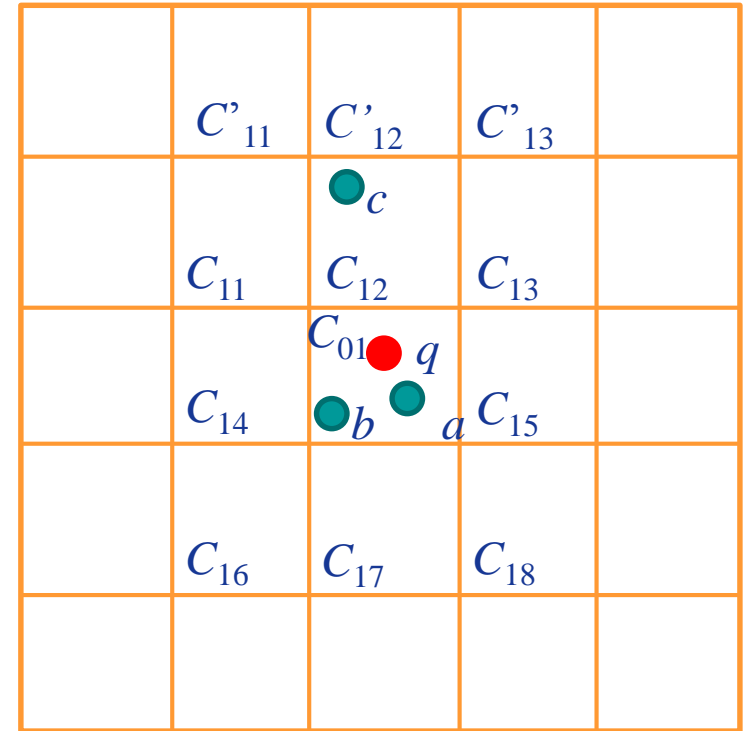
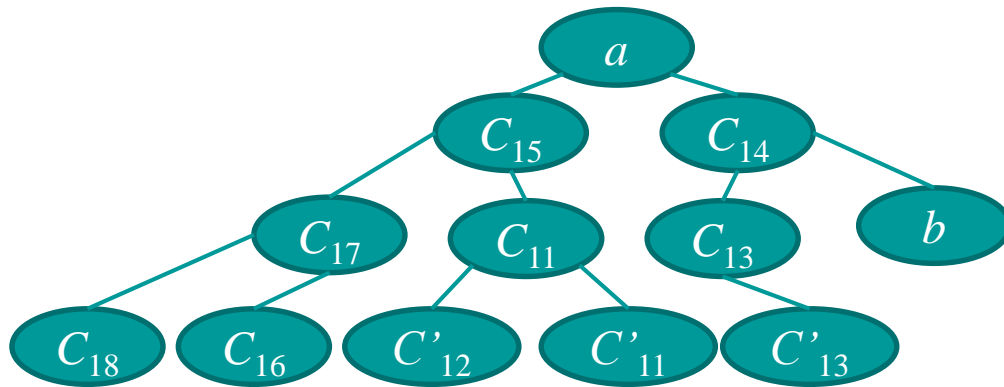
# 最近邻查询

- ❖ 计算以被删网格( $C_{12}$ )为中心的周围8个网格 ( $C'_{11}, C'_{12}, C'_{13}$ )
- ❖ 计算 $q$ 与网格 $C'_{11}, C'_{12}, C'_{13}$ 的MinDist, 分别为 $d'_{11}, d'_{12}, d'_{13}$
- ❖ 依据最小距离, 将 $C'_{11}, C'_{12}, C'_{13}$ 插入小顶堆中, 并调整小顶堆



# 最近邻查询

- ❖ 将堆顶元素弹出( $a$ )，若为空间点
- ❖ 则找出最近邻的空间点
- ❖ 查找结束



**谢谢！**

