

慕课网《算法与数据结构》

算法与数据结构

讲师：liuyubobobo

版权所有 侵权必究

liuyubobobo

慕课网《算法与数据结构》

堆排序 Heap Sort

讲师：liuyubobobo

版权所有，侵权必究

堆和优先队列 Heap and Priority Queue

慕课网《算法与数据结构》

讲师：liuyubobobo

版权所有，侵权必究

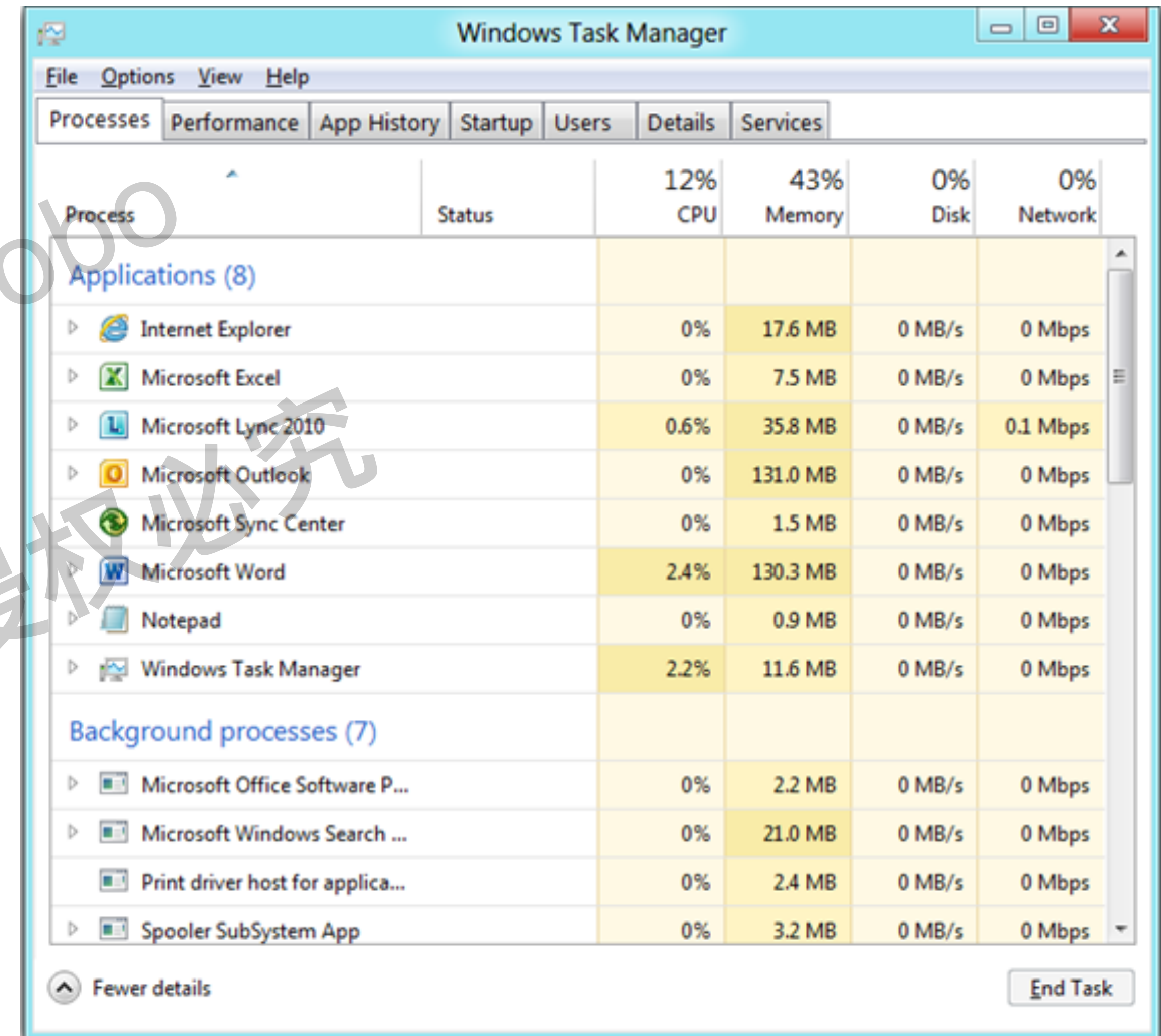
什么是优先队列?

普通队列：先进先出；后进后出

优先队列：出队顺序和入队顺序无关；和优先级相关

为什么使用优先队列?

动态选择优先级最高的任务执行



The screenshot shows the Windows Task Manager window with the 'Processes' tab selected. It displays a list of running applications and background processes, sorted by CPU usage. The table includes columns for Process, Status, CPU, Memory, Disk, and Network usage.

Process	Status	12% CPU	43% Memory	0% Disk	0% Network
Applications (8)					
Internet Explorer		0%	17.6 MB	0 MB/s	0 Mbps
Microsoft Excel		0%	7.5 MB	0 MB/s	0 Mbps
Microsoft Lync 2010		0.6%	35.8 MB	0 MB/s	0.1 Mbps
Microsoft Outlook		0%	131.0 MB	0 MB/s	0 Mbps
Microsoft Sync Center		0%	1.5 MB	0 MB/s	0 Mbps
Microsoft Word		2.4%	130.3 MB	0 MB/s	0 Mbps
Notepad		0%	0.9 MB	0 MB/s	0 Mbps
Windows Task Manager		2.2%	11.6 MB	0 MB/s	0 Mbps
Background processes (7)					
Microsoft Office Software P...		0%	2.2 MB	0 MB/s	0 Mbps
Microsoft Windows Search ...		0%	21.0 MB	0 MB/s	0 Mbps
Print driver host for applica...		0%	2.4 MB	0 MB/s	0 Mbps
Spooler SubSystem App		0%	3.2 MB	0 MB/s	0 Mbps

为什么使用优先队列?

关键词：动态

任务处理中心

Request

Request

Request

Request

Request

为什么使用优先队列?



为什么使用优先队列?

在1,000,000个元素中选出前100名?

在N个元素中选出前M个元素

排序? $N \log N$

使用优先队列? $N \log M$

优先队列主要操作

入队

出队（取出优先级最高的元素）

优先队列的实现

	入队	出队
普通数组	$O(1)$	$O(n)$
顺序数组	$O(n)$	$O(1)$
堆	$O(\lg n)$	$O(\lg n)$

使用堆实现优先队列

对于总共N个请求：

使用普通数组或者顺序数组，最差情况： $O(n^2)$

使用堆： $O(n \lg n)$

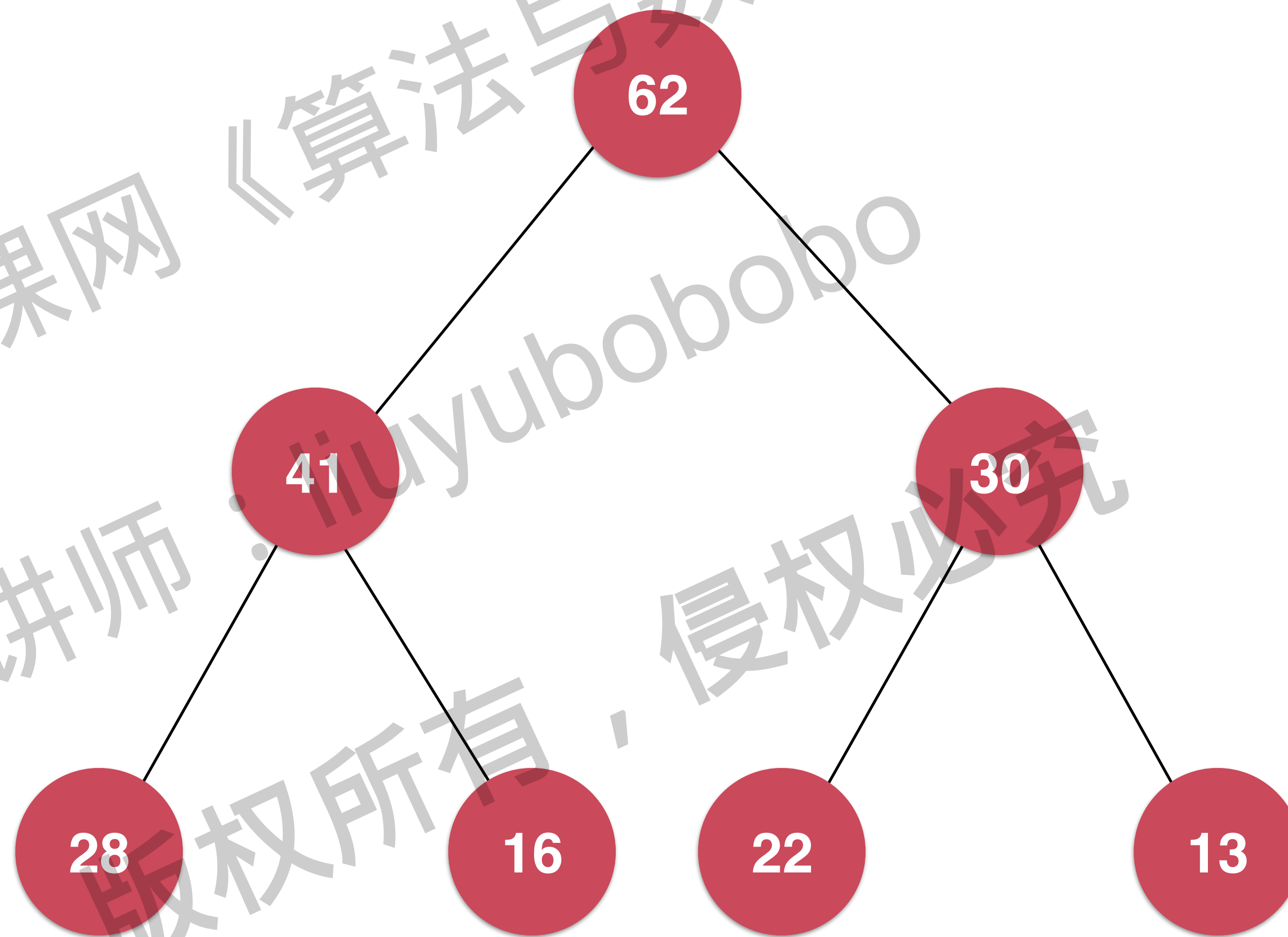
慕课网《算法与数据结构》

堆的基本实现

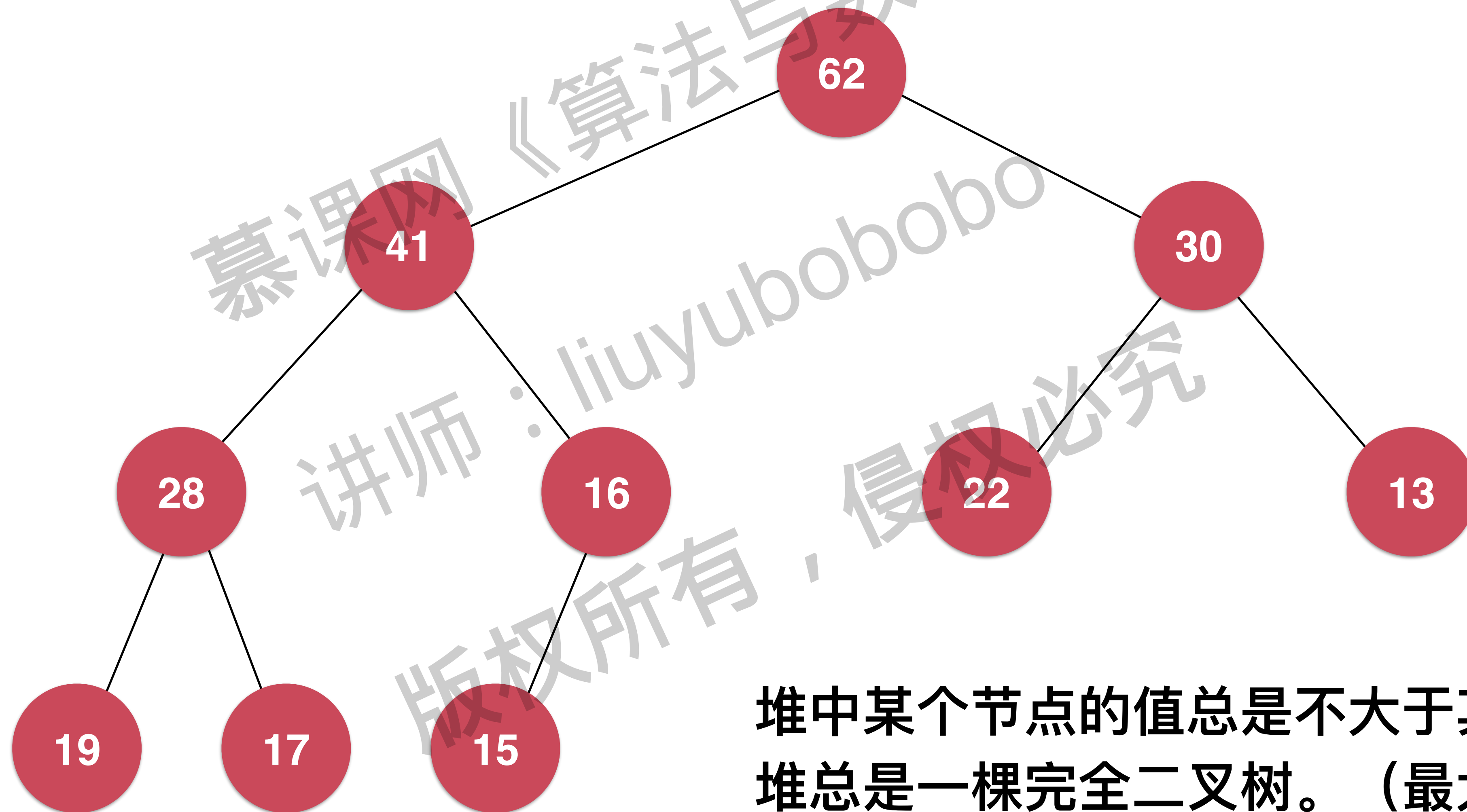
讲师：liuyubobobo

版权所有，侵权必究

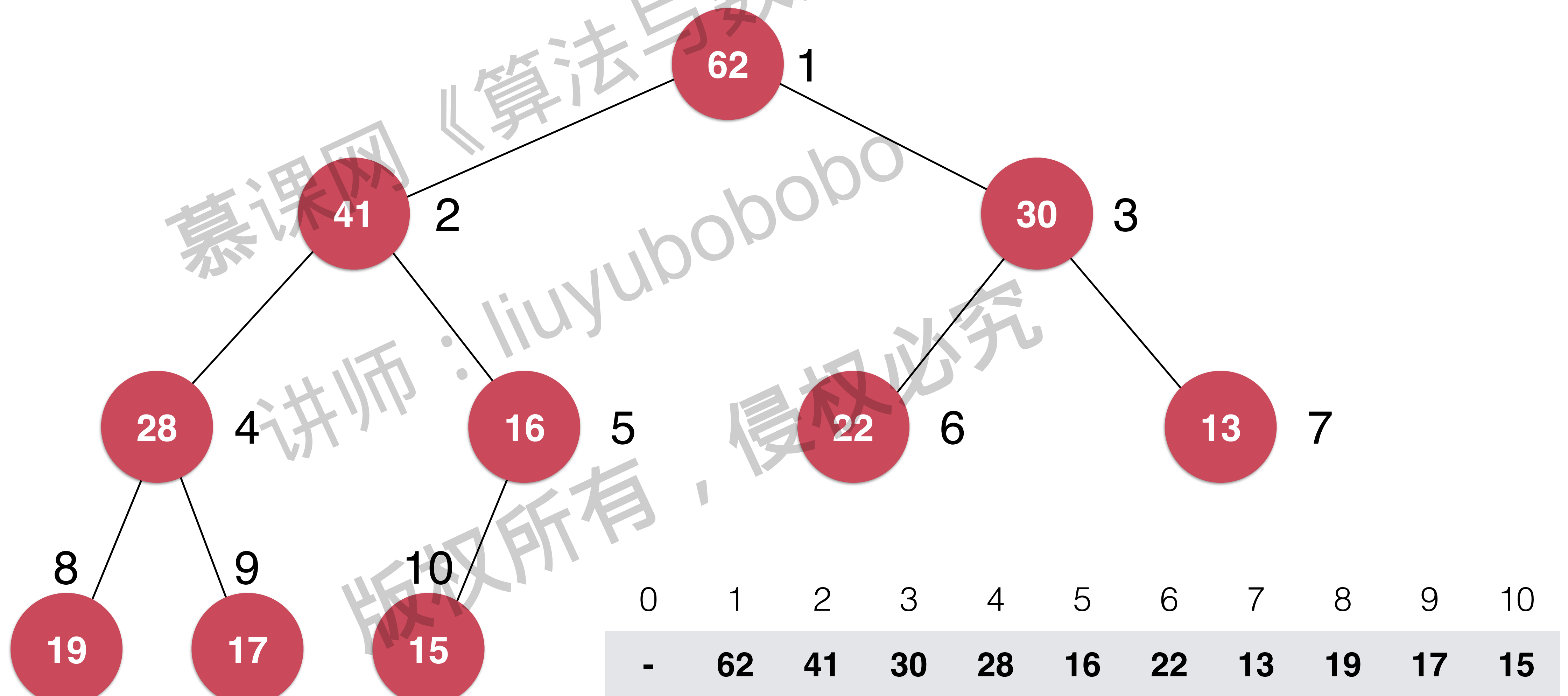
二叉堆 Binary Heap



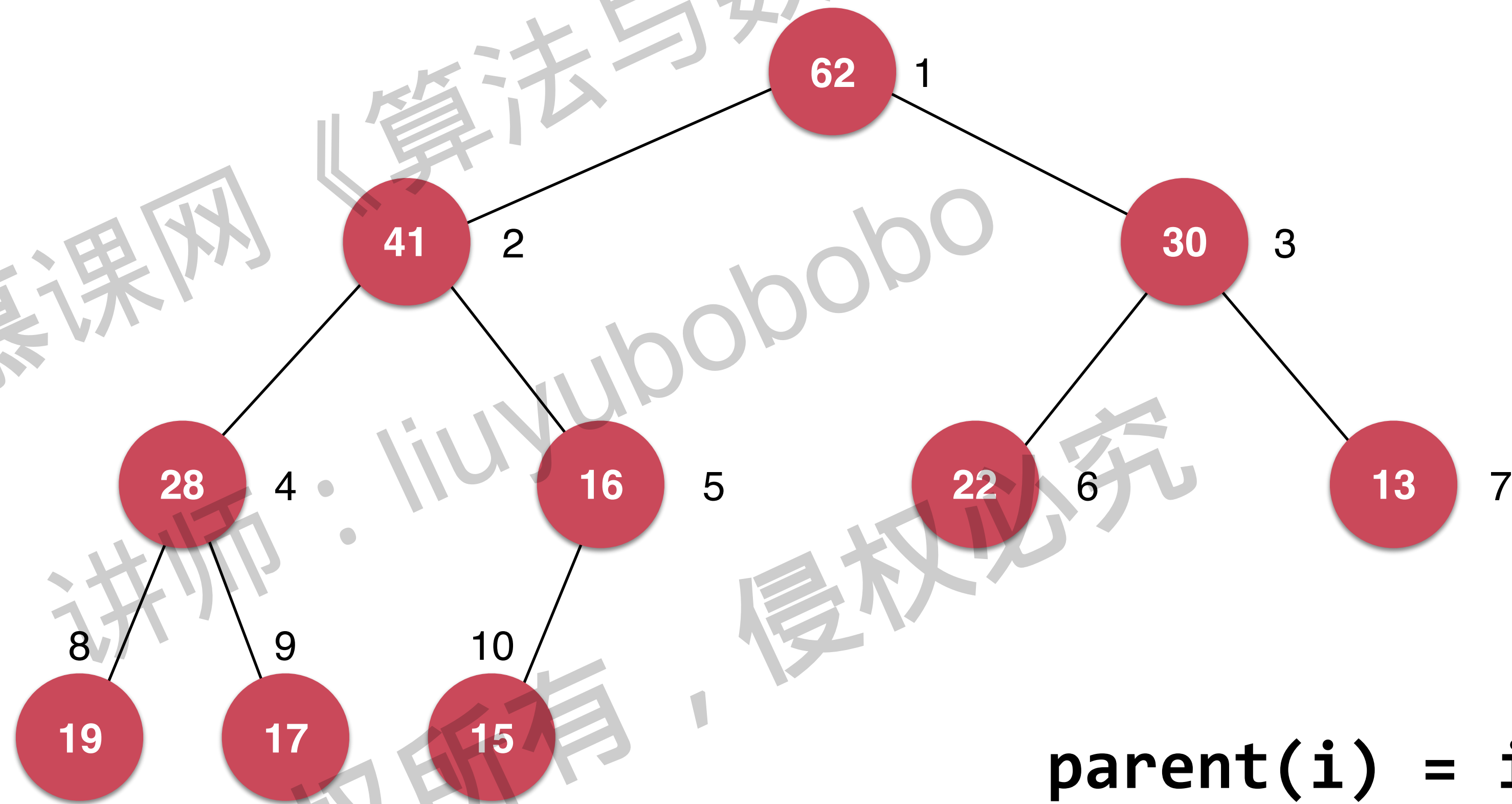
二叉堆 是一棵 完全二叉树



用数组存储二叉堆



用数组存储二叉堆



0	1	2	3	4	5	6	7	8	9	10
-	62	41	30	28	16	22	13	19	17	15

$$\text{parent}(i) = i/2$$

$$\text{left child } (i) = 2*i$$

$$\text{right child } (i) = 2*i + 1$$

慕课网《算法与数据结构》

操作：堆的基本框架

讲师：liuyubobobo

版权所有，侵权必究

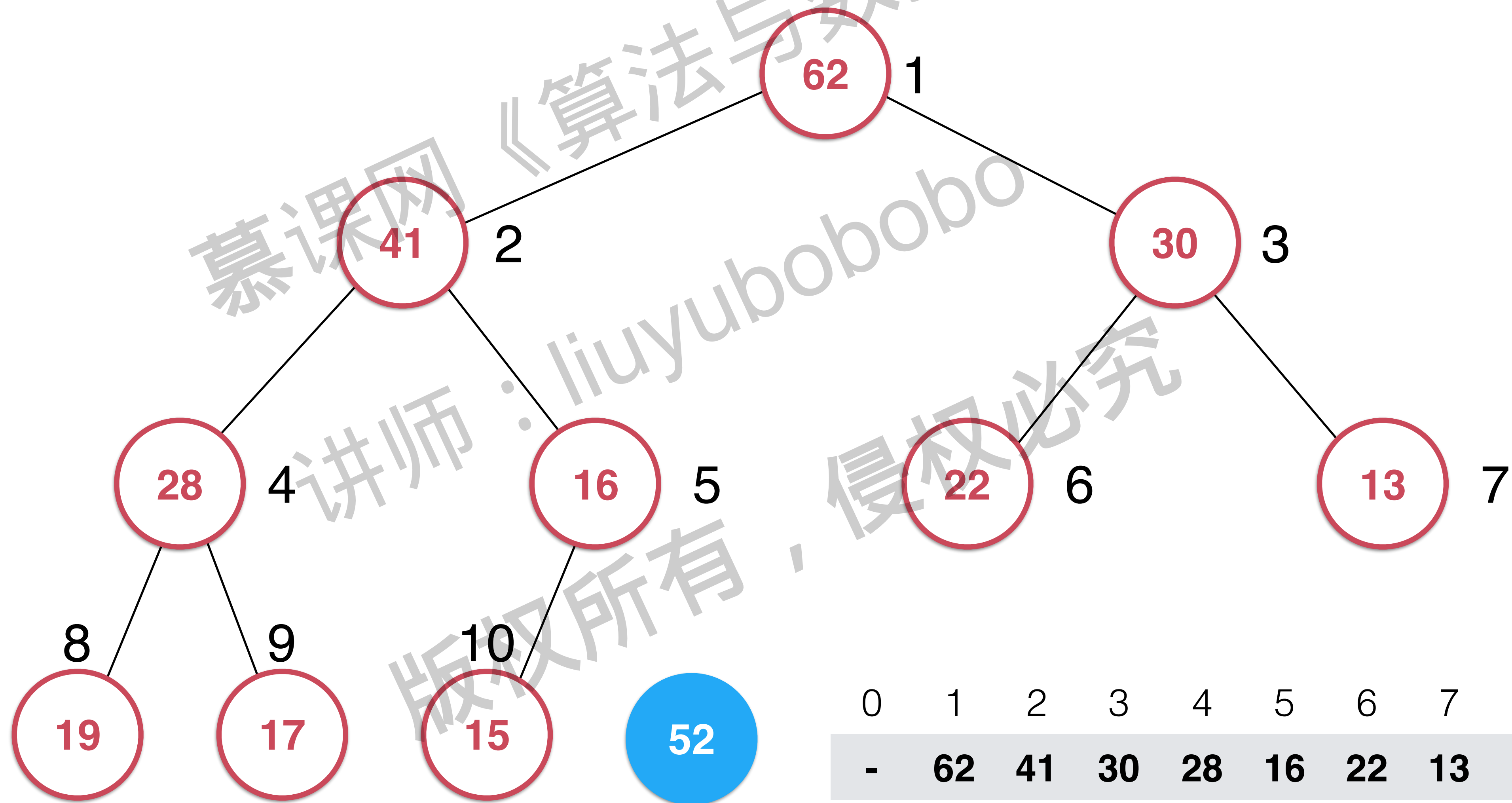
慕课网《算法与数据结构》

Shift Up

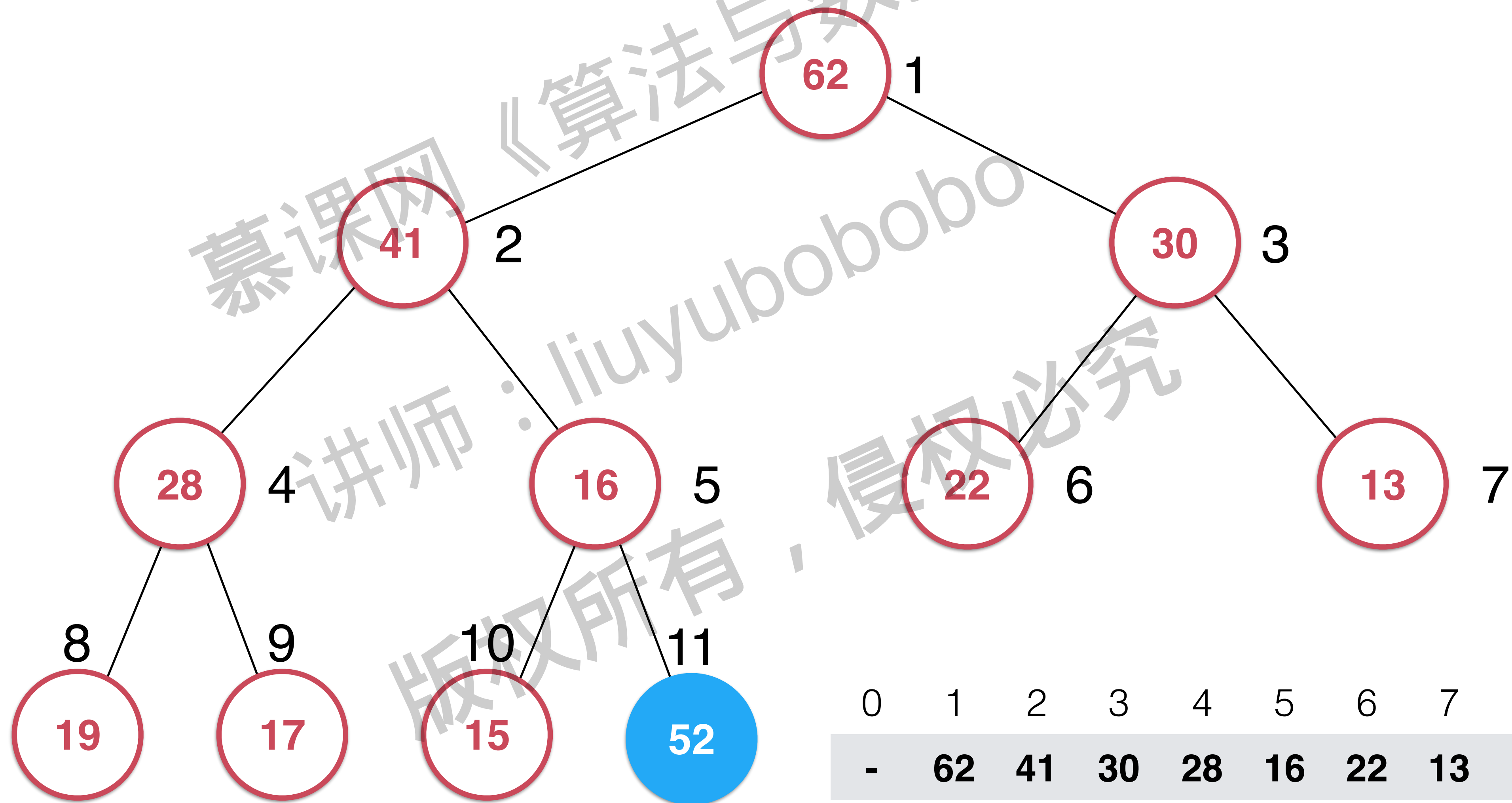
讲师：liuyubobobo

版权所有，侵权必究

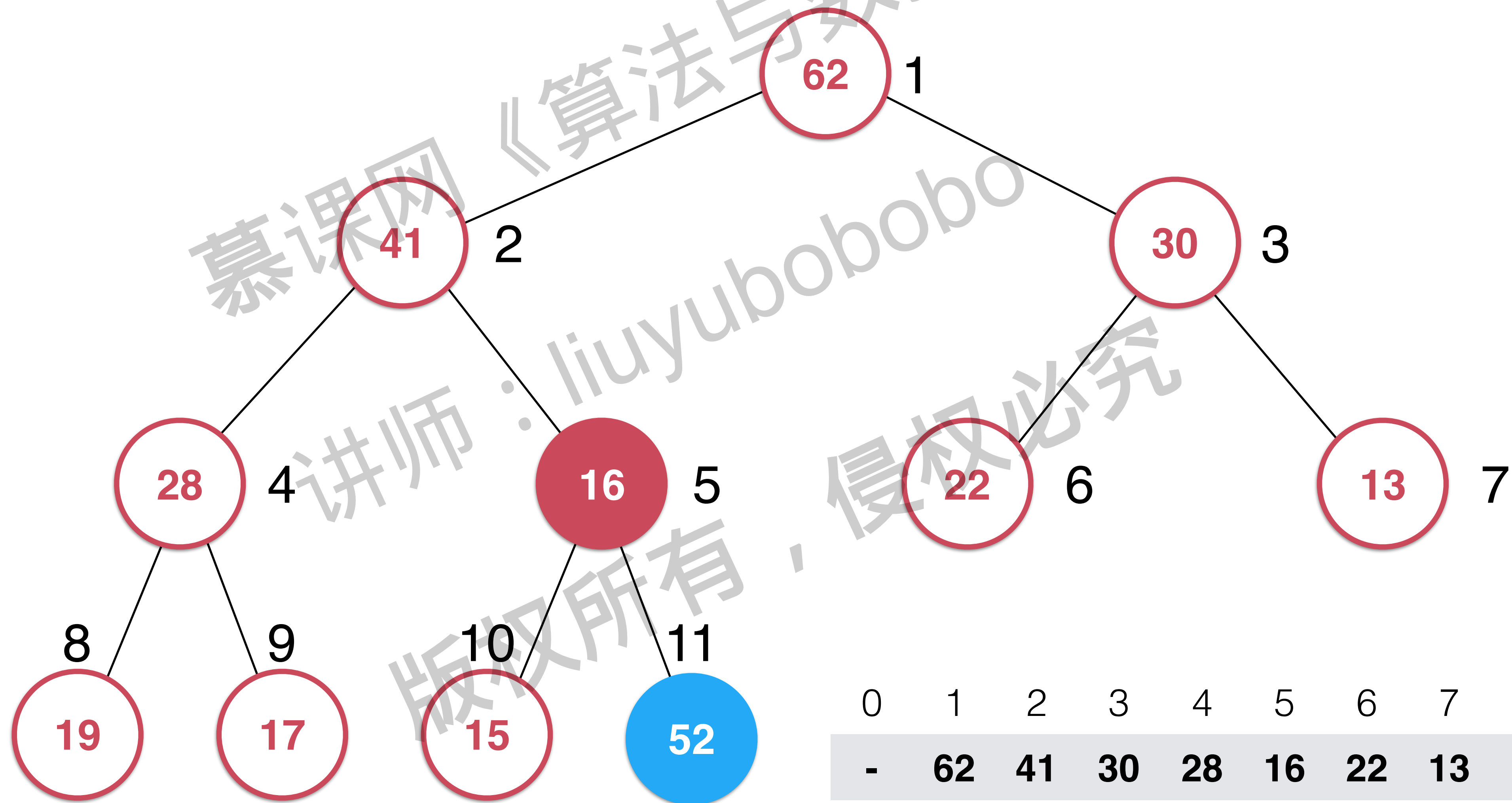
Shift Up



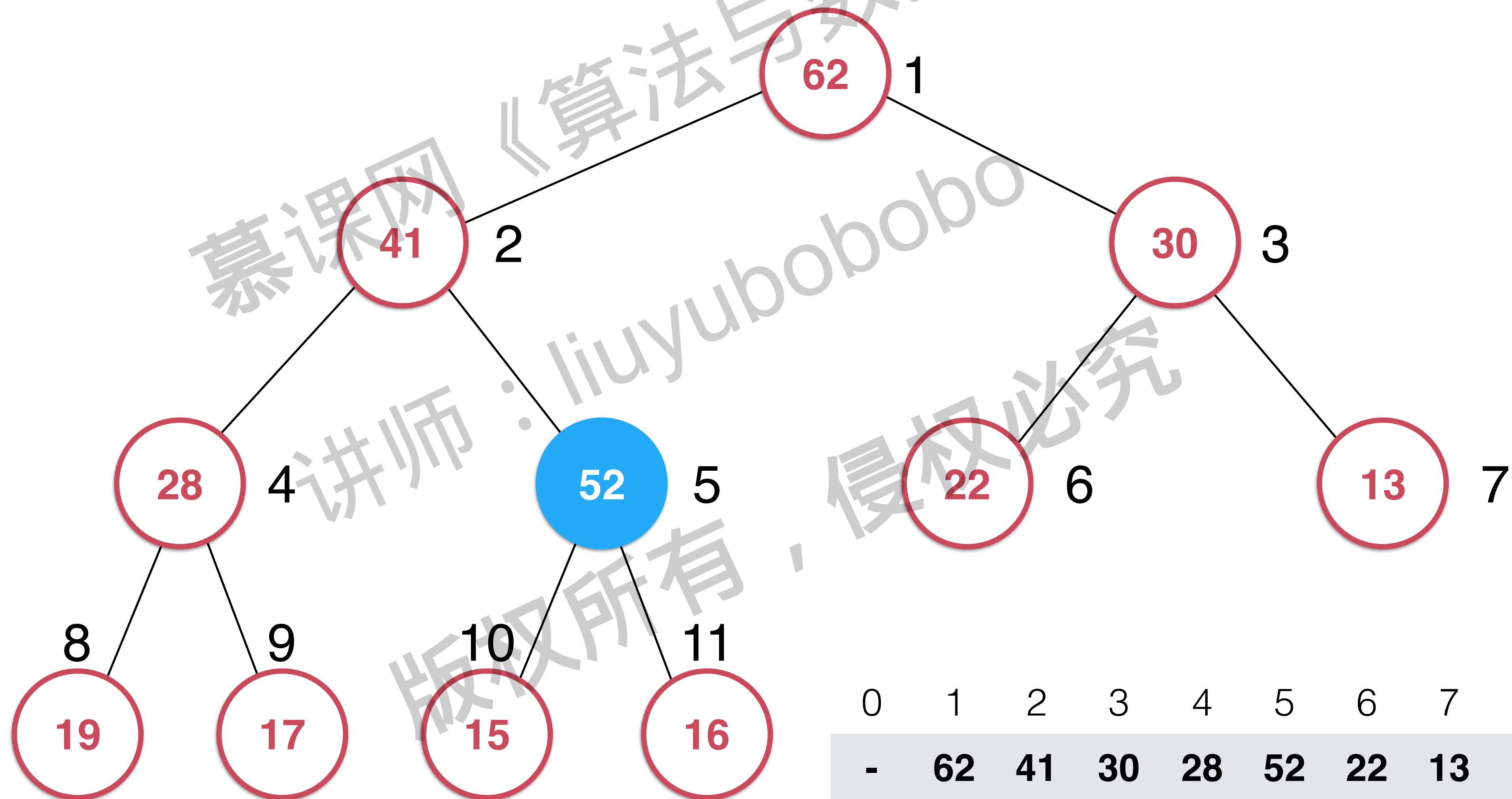
Shift Up



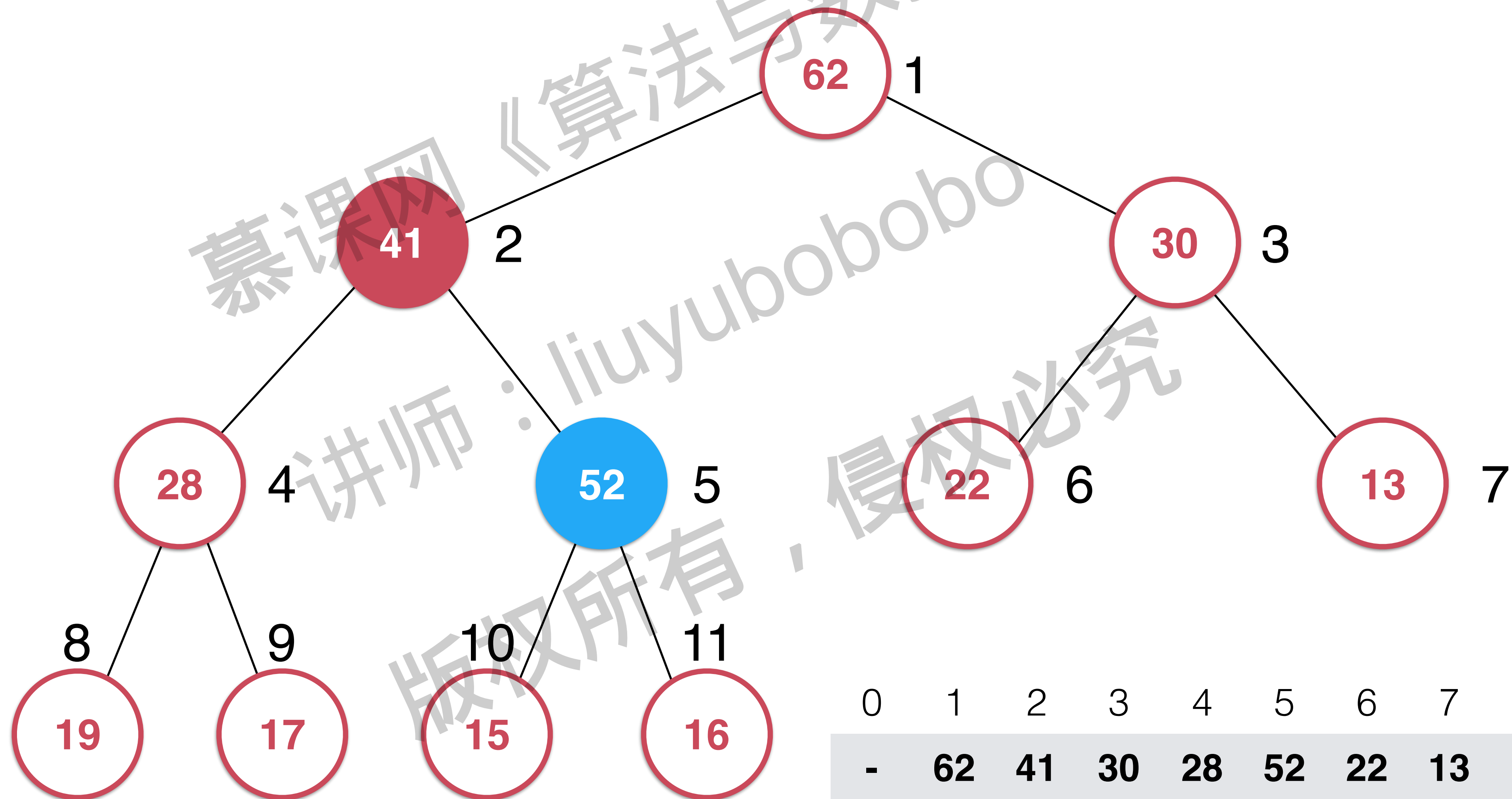
Shift Up



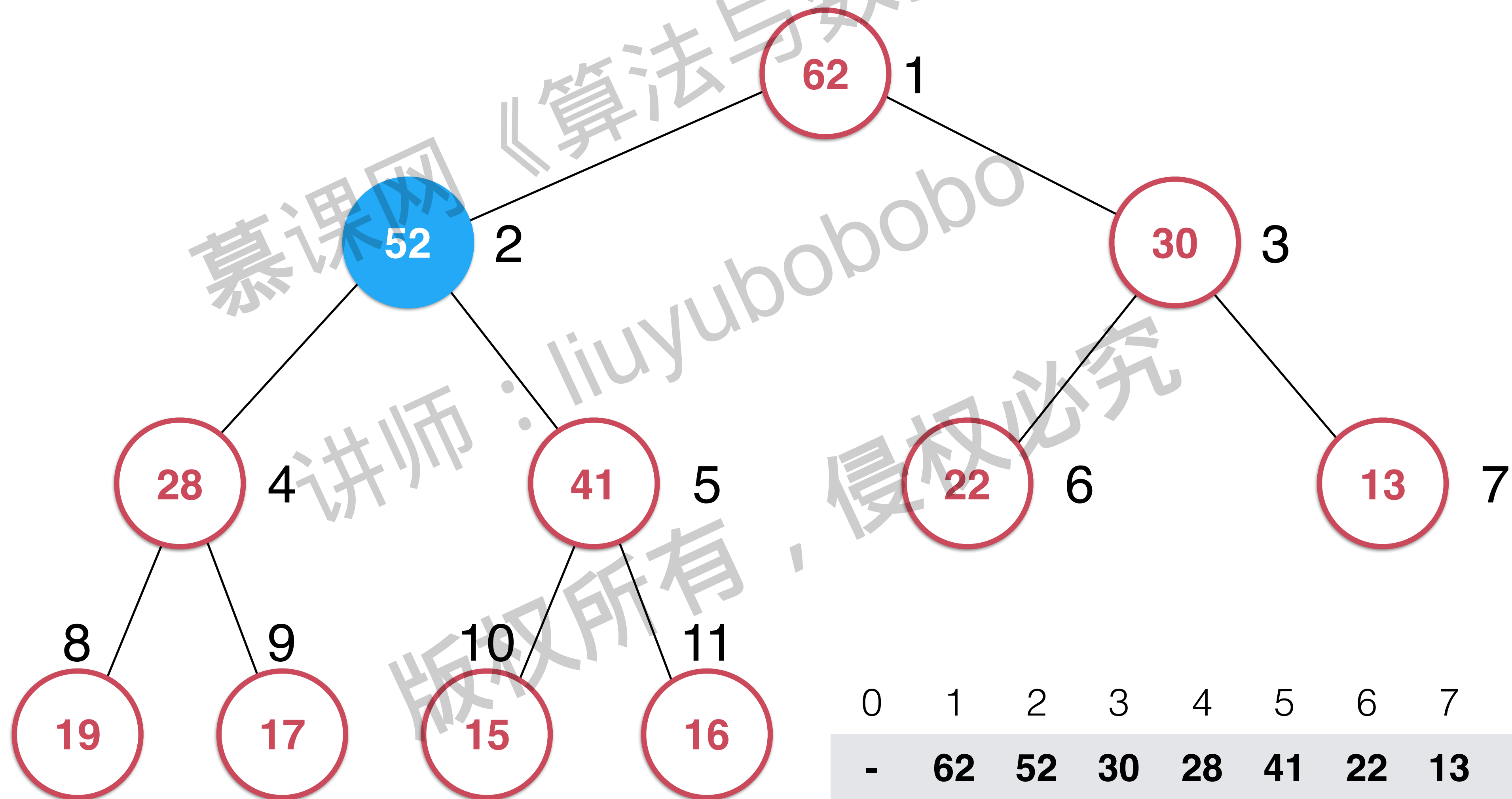
Shift Up



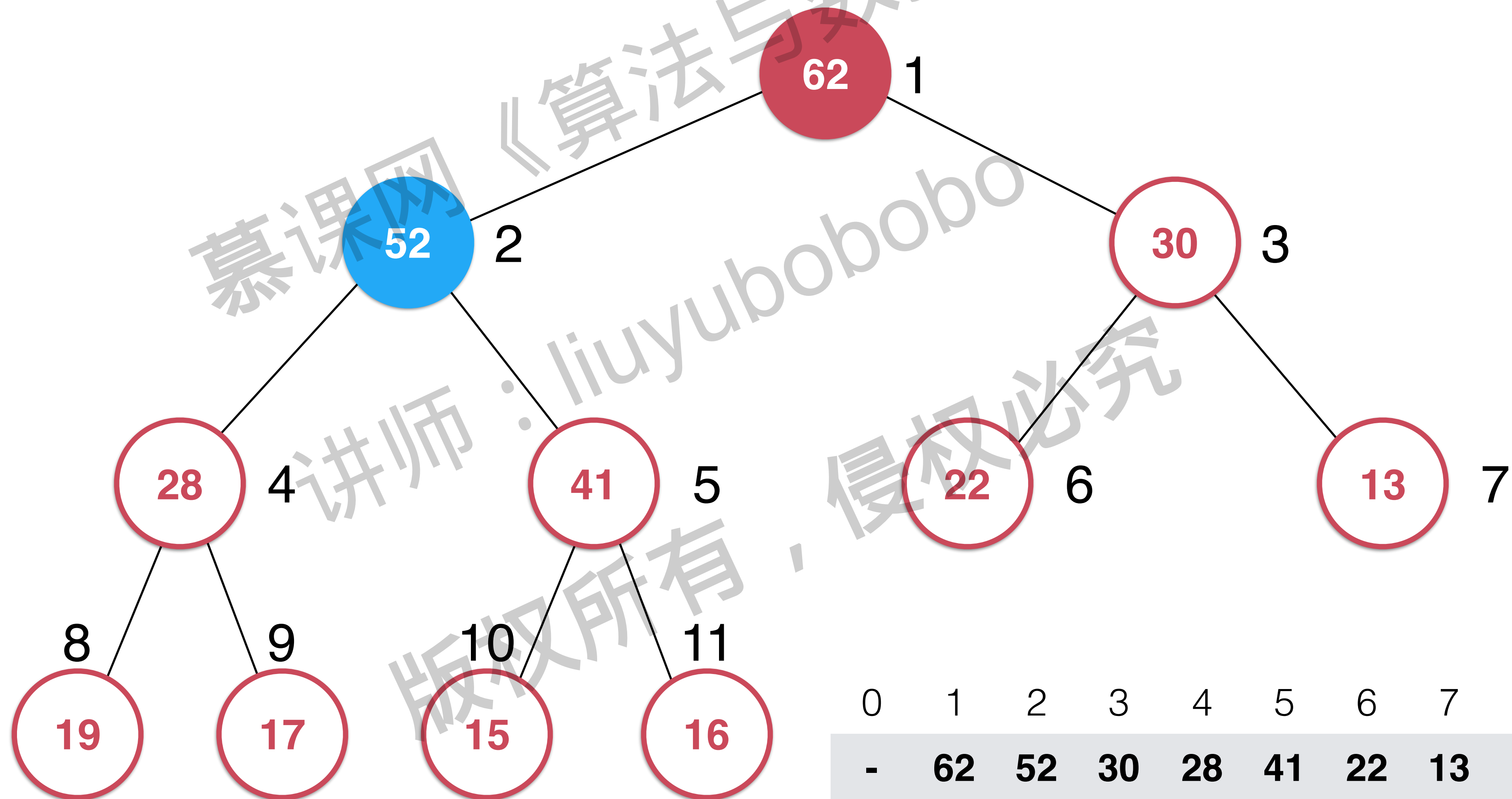
Shift Up



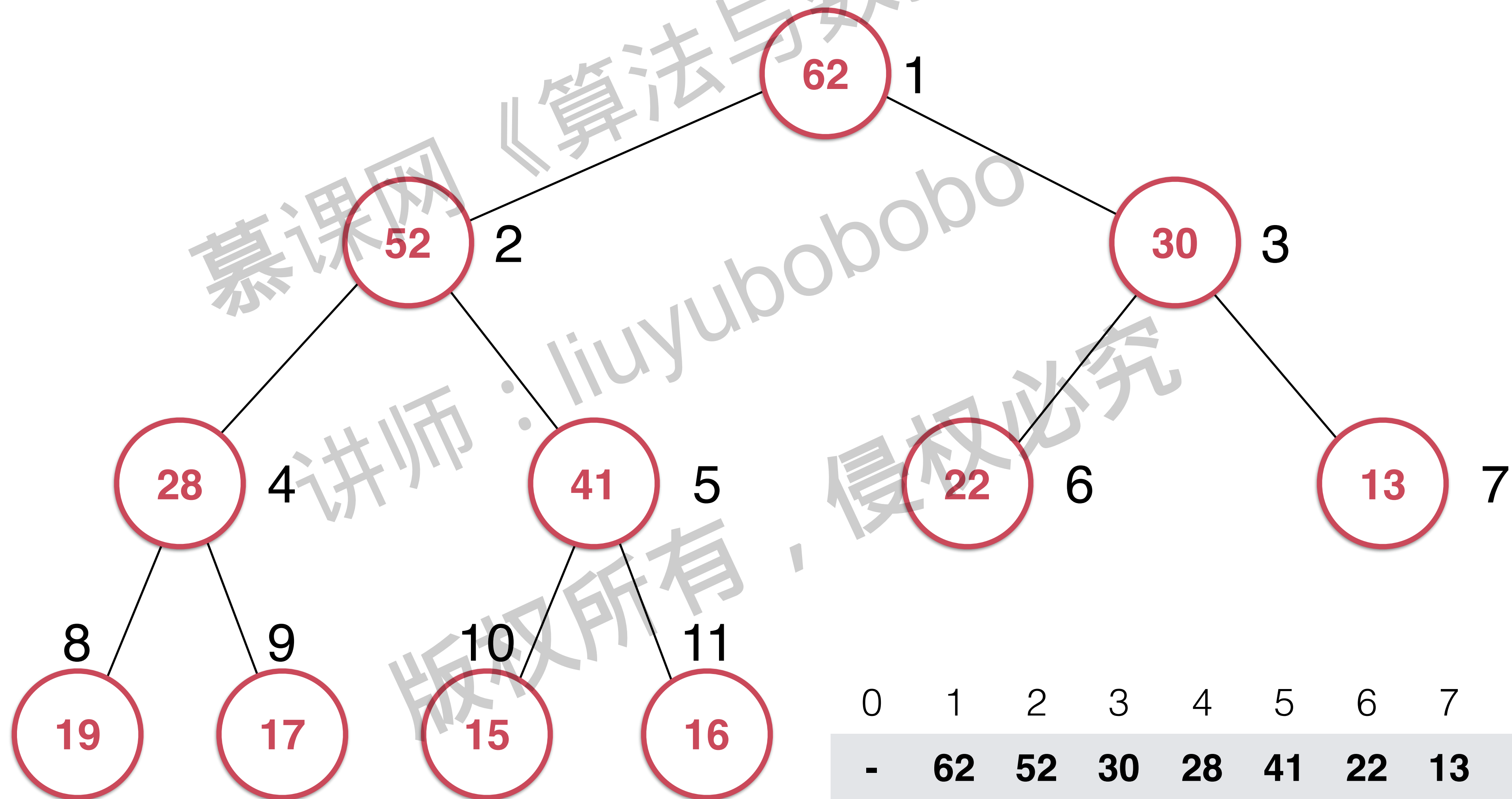
Shift Up



Shift Up



Shift Up



操作：Shift Up 和 insert

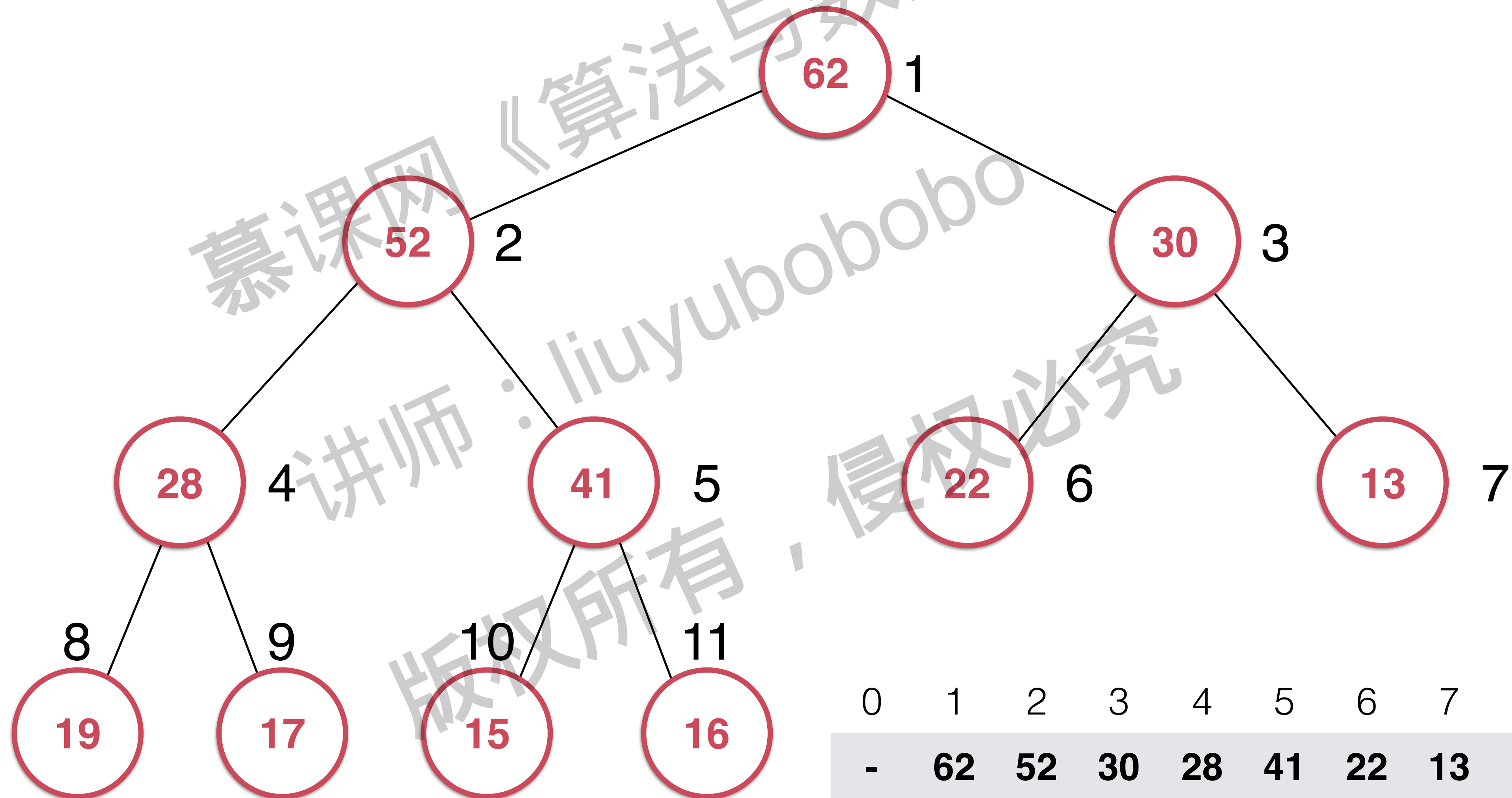
慕课网《算法与数据结构》

Shift Down

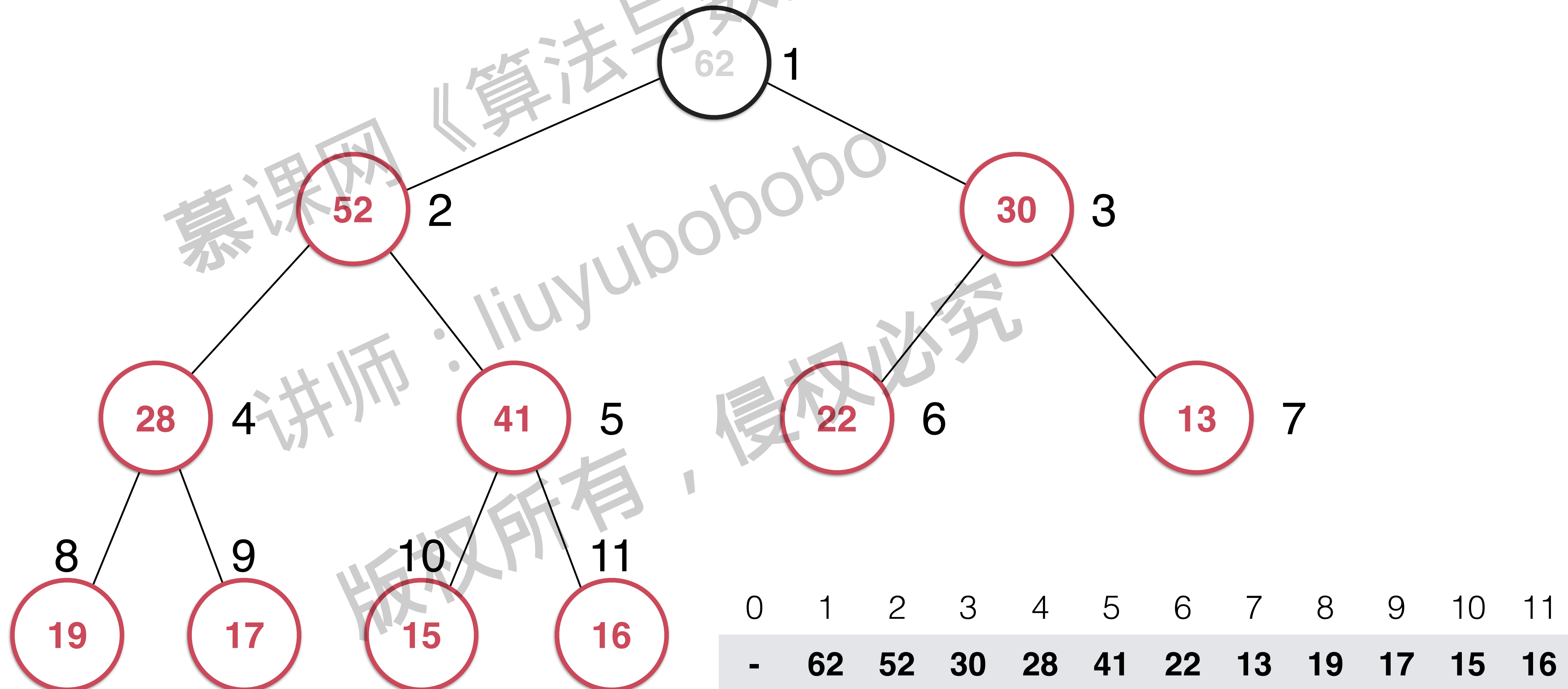
讲师：liuyubobobo

版权所有，侵权必究

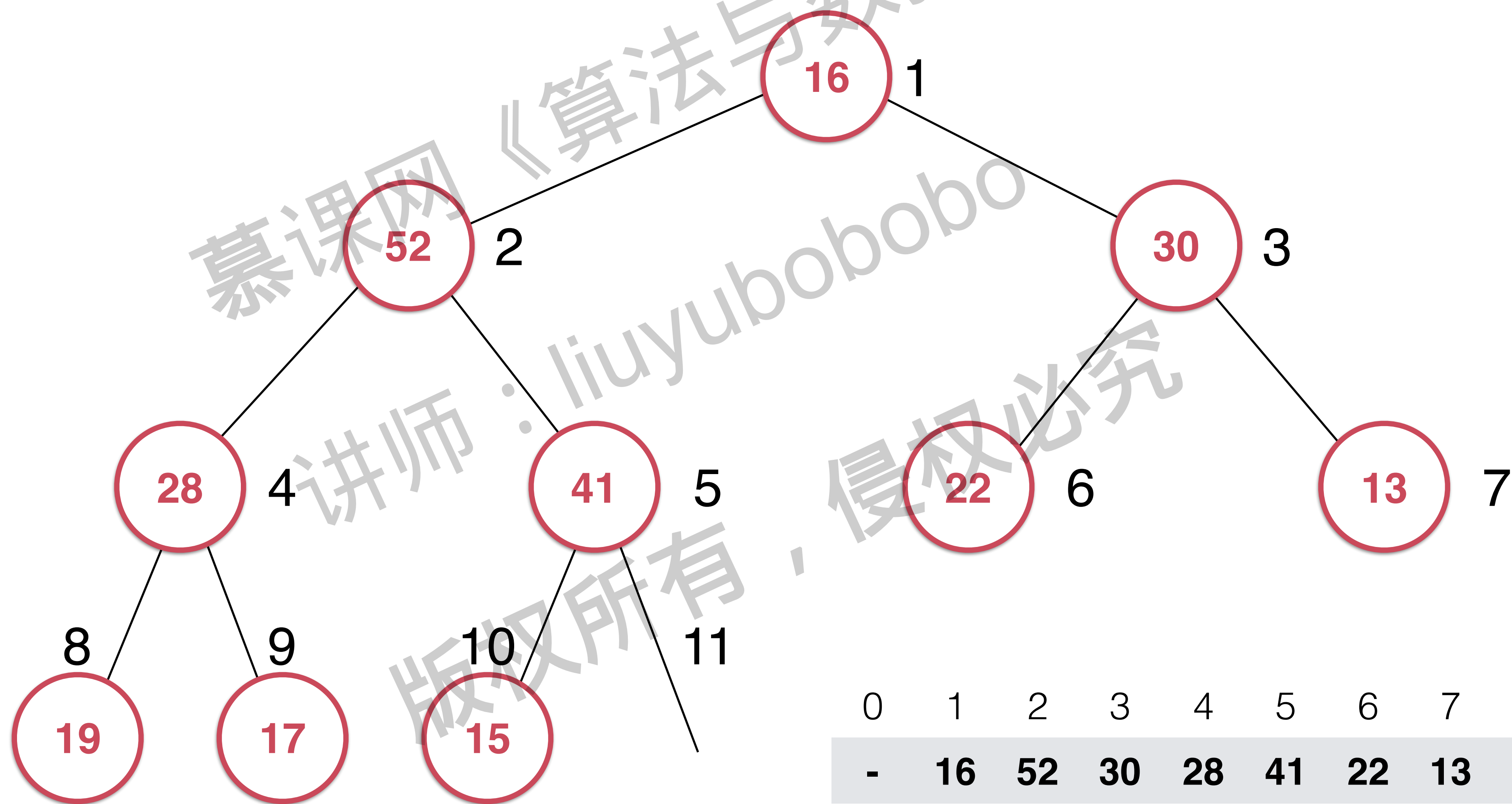
Shift Down



Shift Down



Shift Down

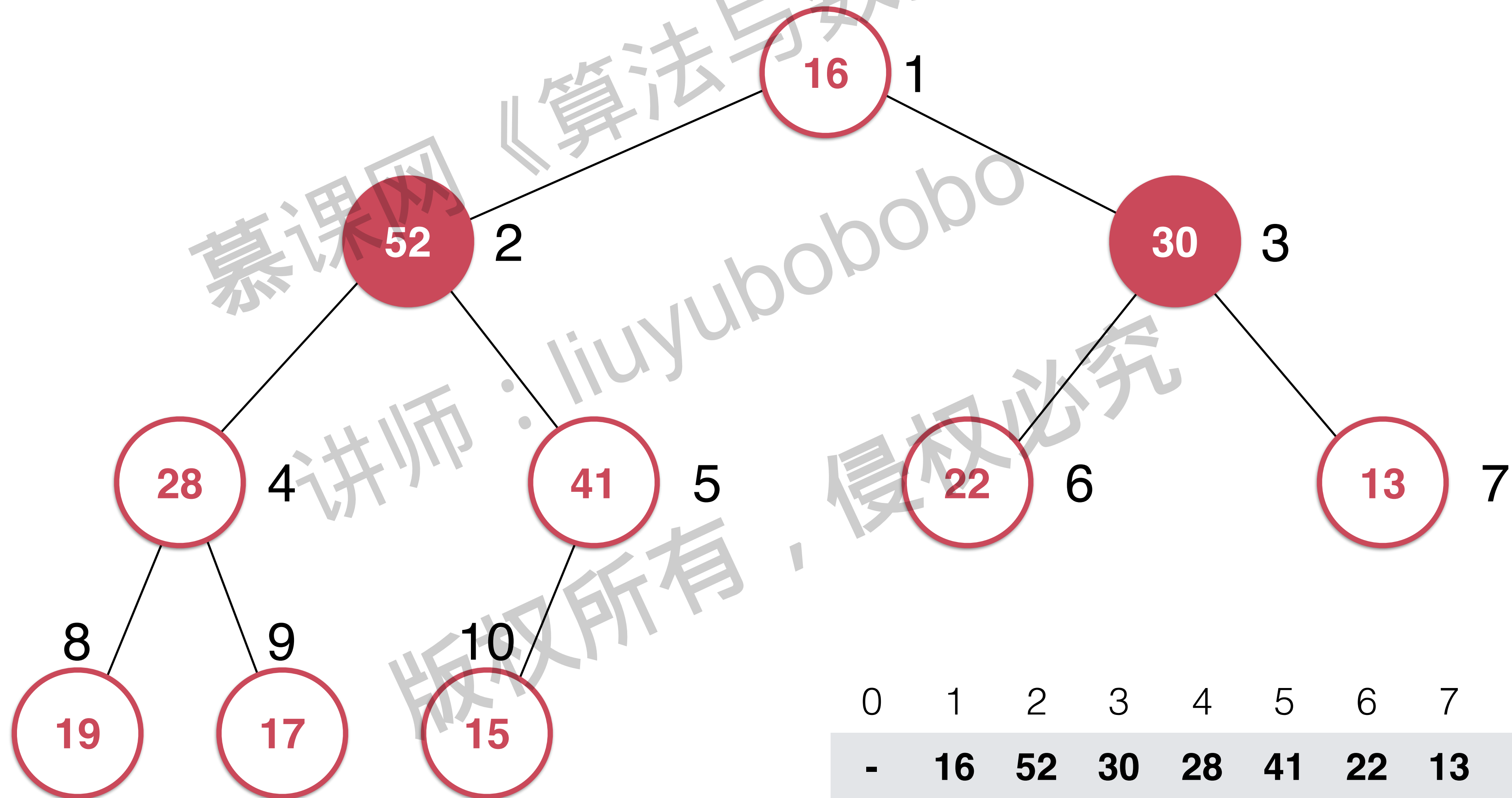


count

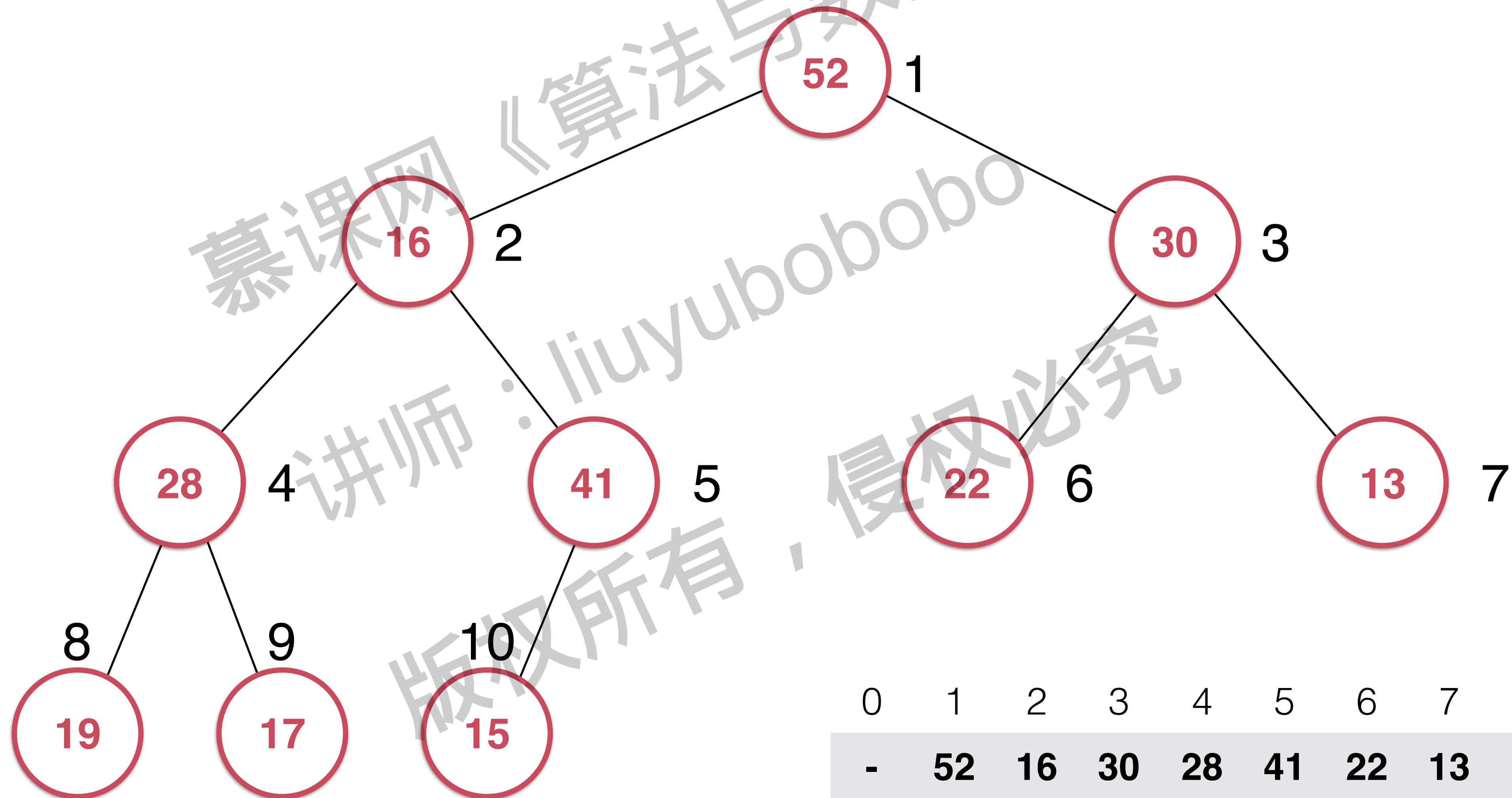


0	1	2	3	4	5	6	7	8	9	10	11
-	16	52	30	28	41	22	13	19	17	15	16

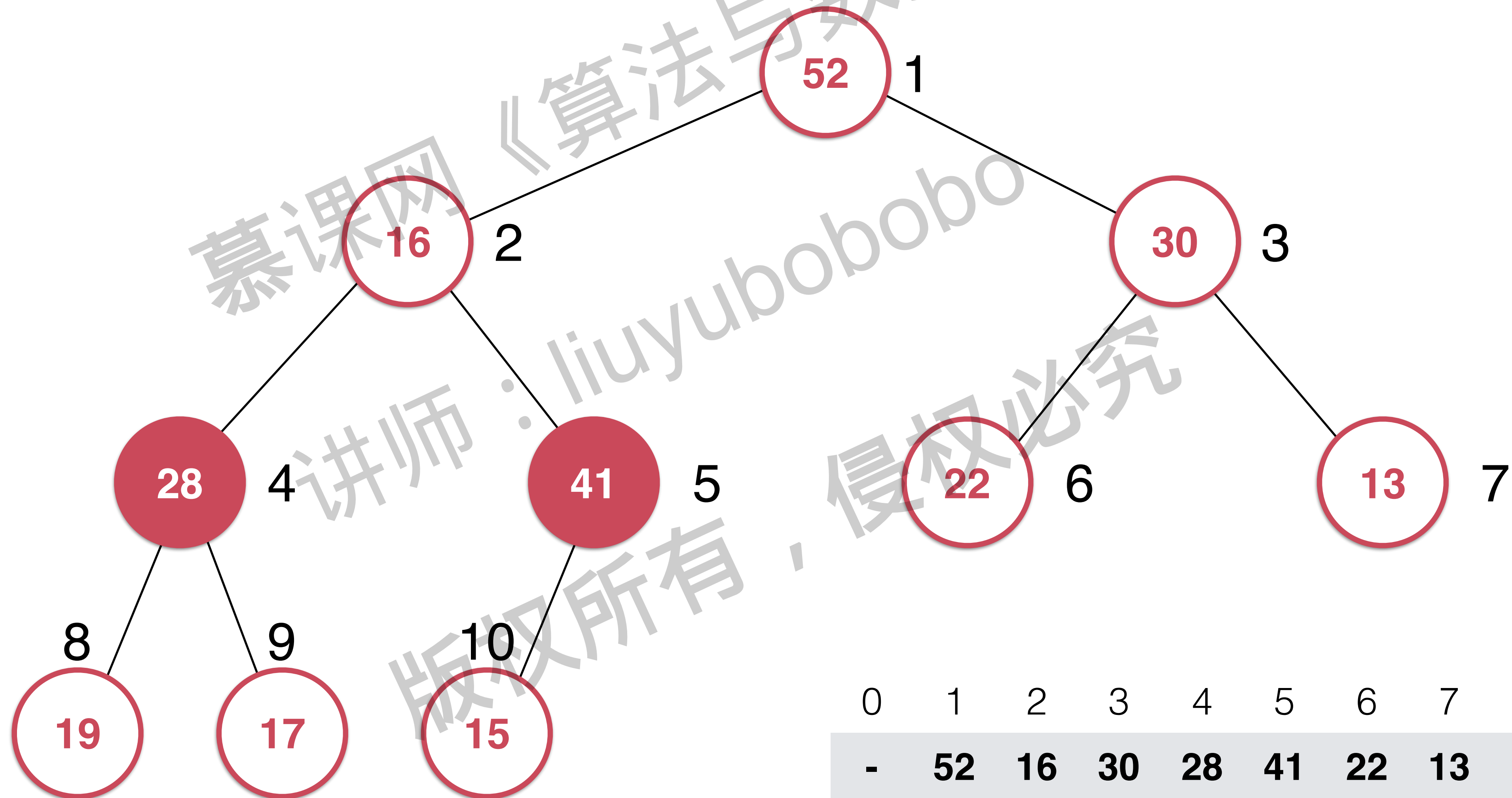
Shift Down



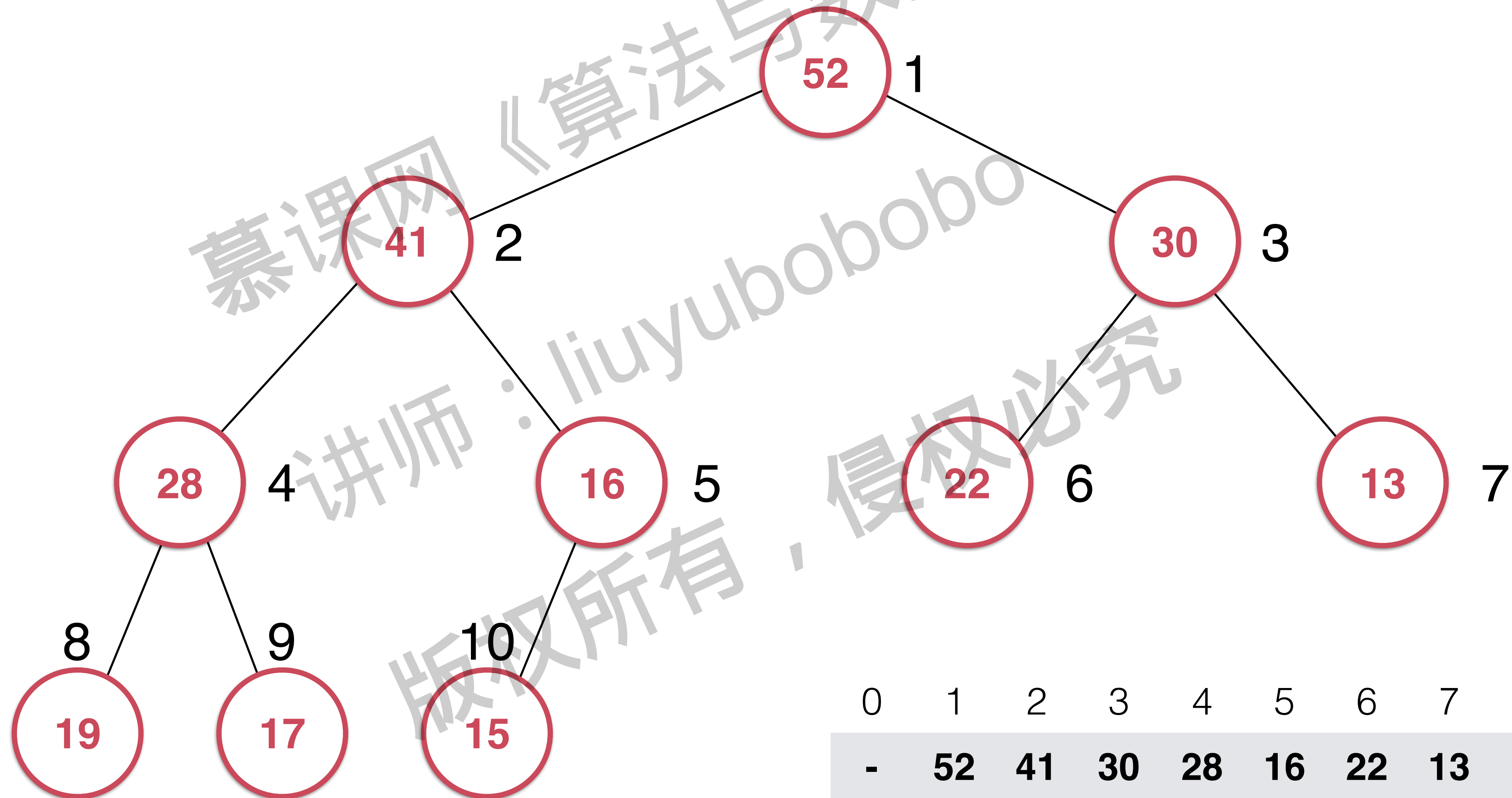
Shift Down



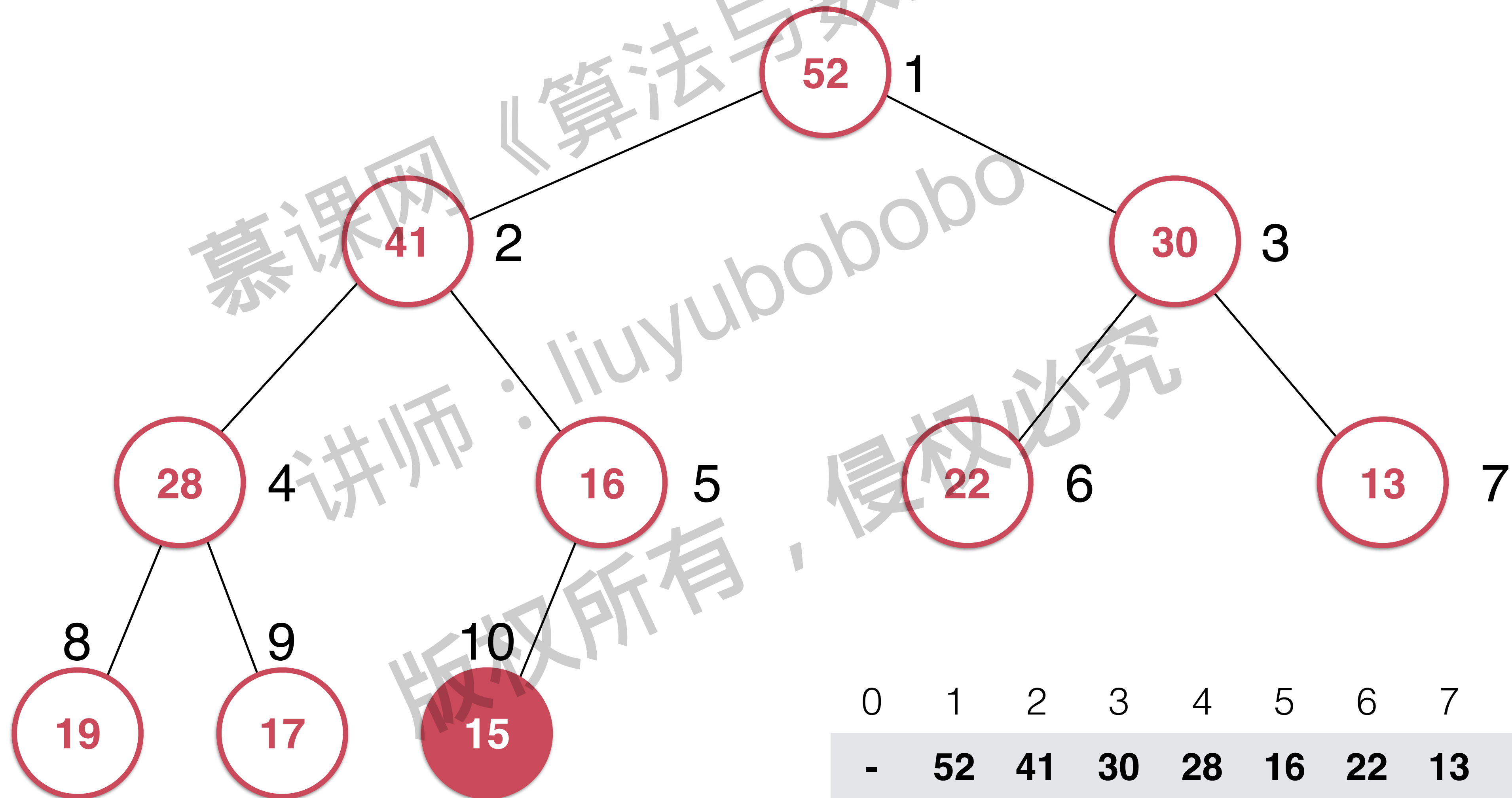
Shift Down



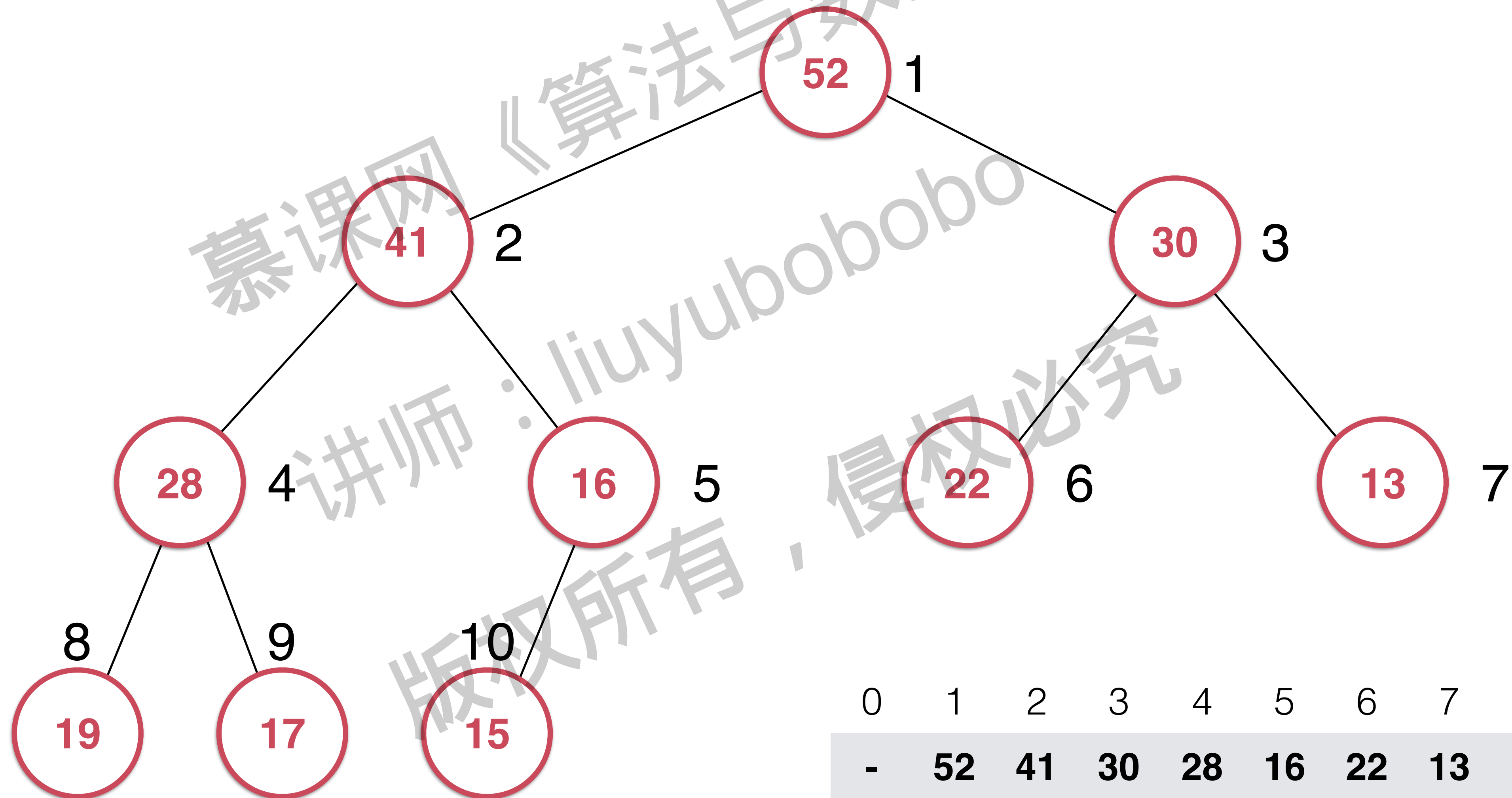
Shift Down



Shift Down



Shift Down



操作：Shift Down 和 extract Max

慕课网《算法与数据结构》

操作：Basic Heap Sort

讲师：liuyubobobo

版权所有，侵权必究

操作：Basic Heap Sort 和 Merge Sort, Quick Sort 的比较

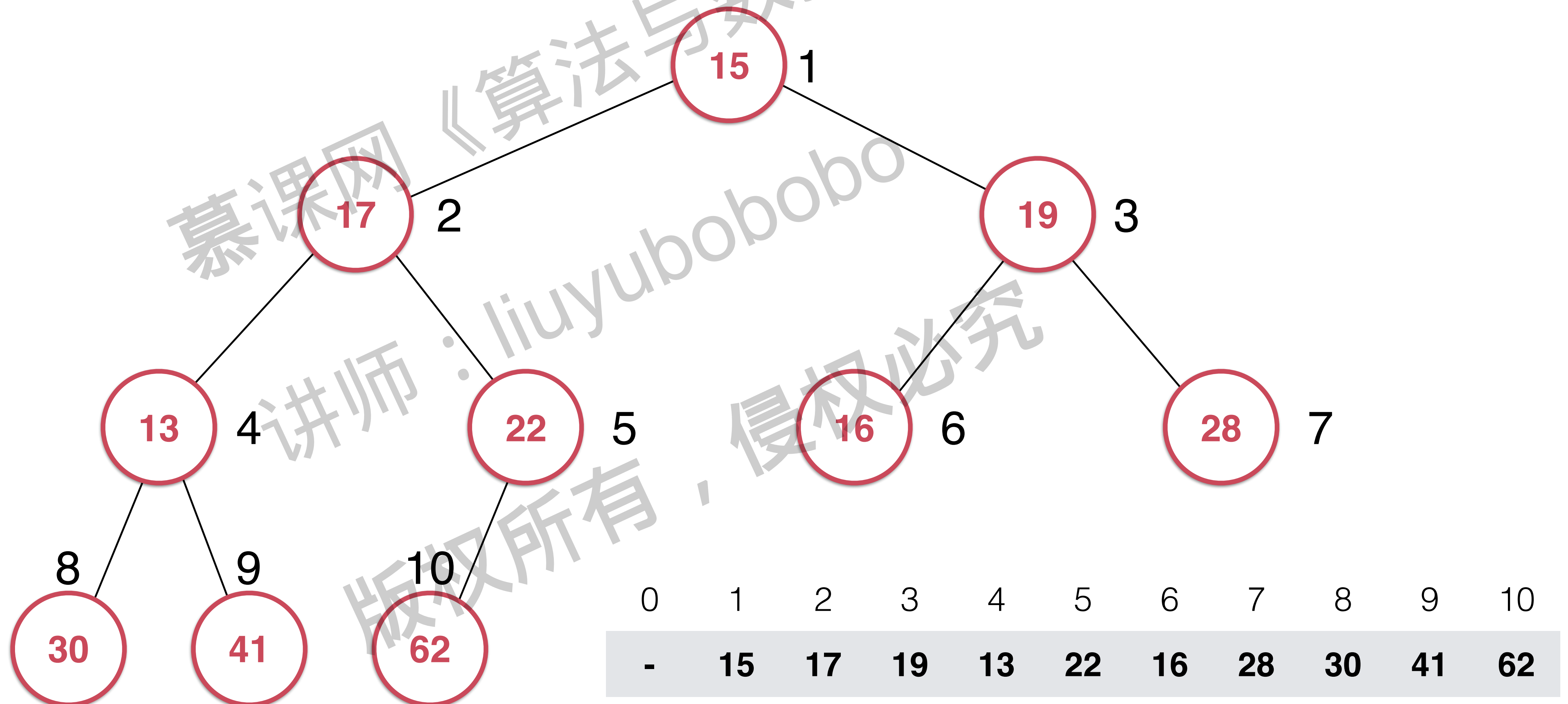
慕课网《算法与数据结构》

Heapify

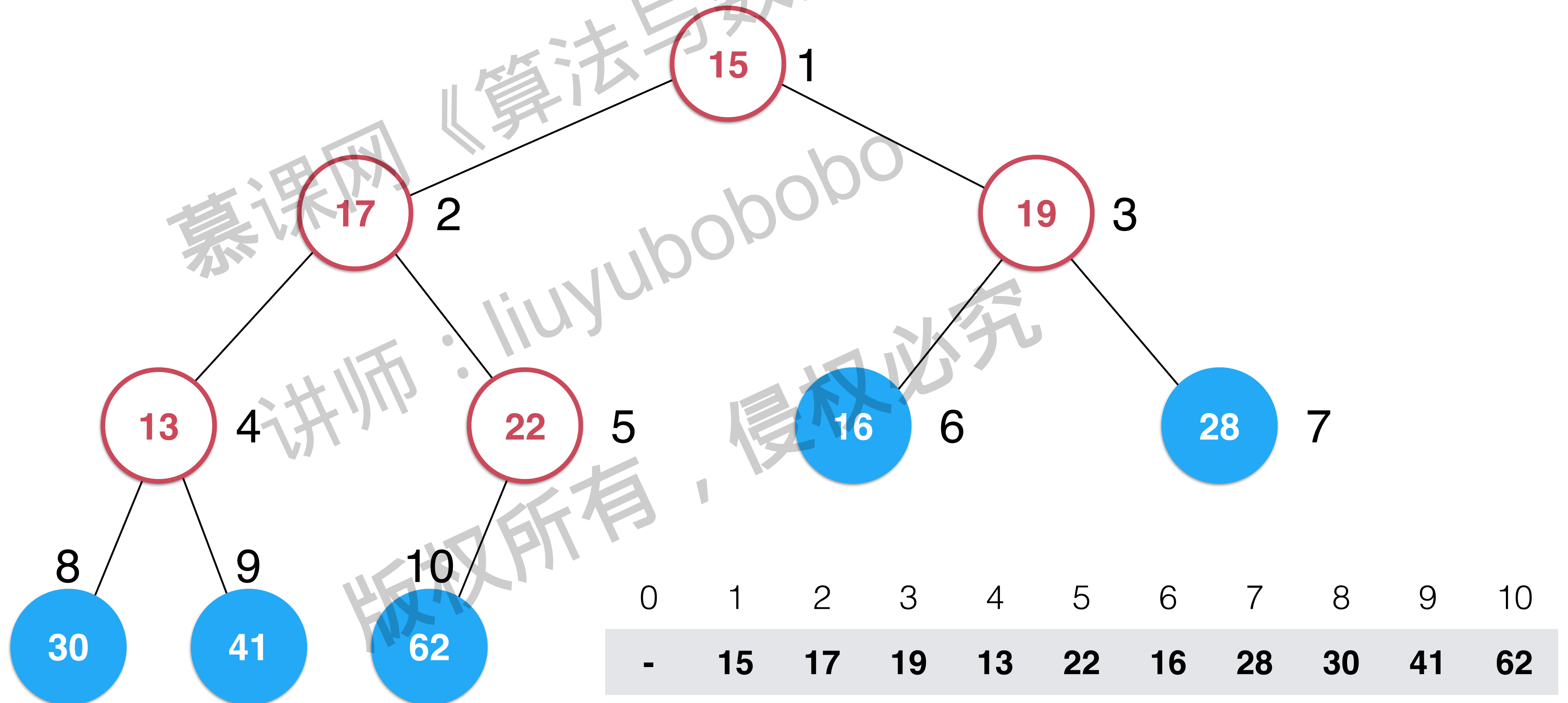
讲师：liuyubobobo

版权所有，侵权必究

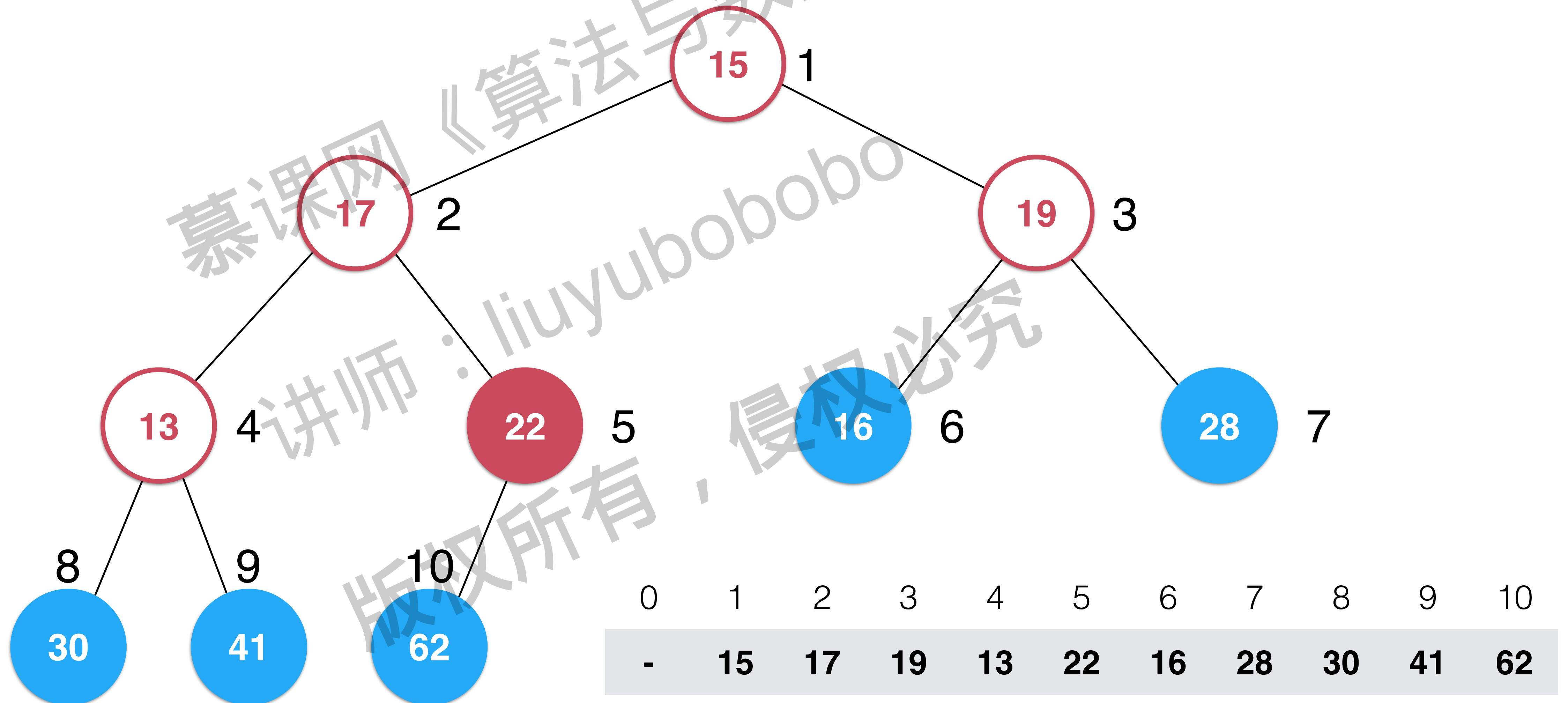
Heapify



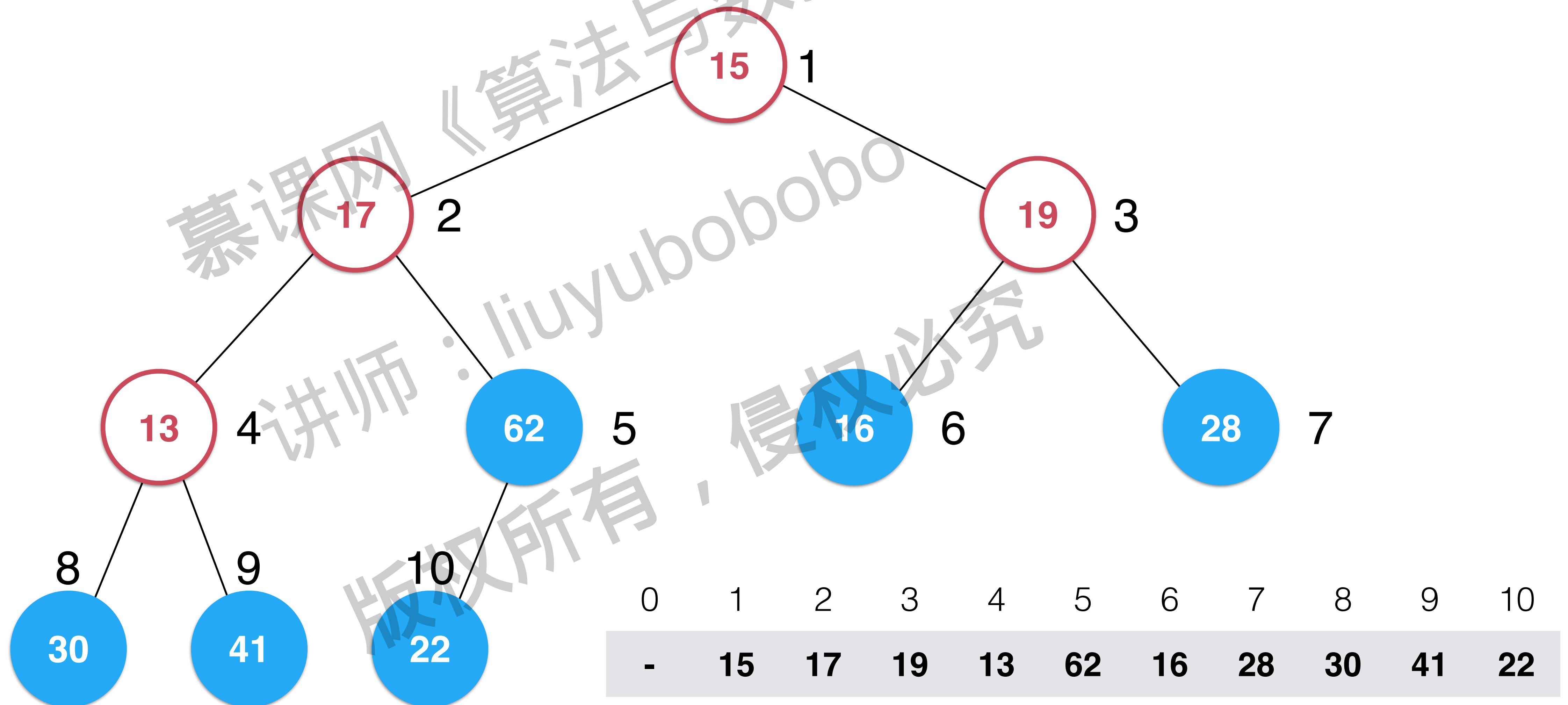
Heapify



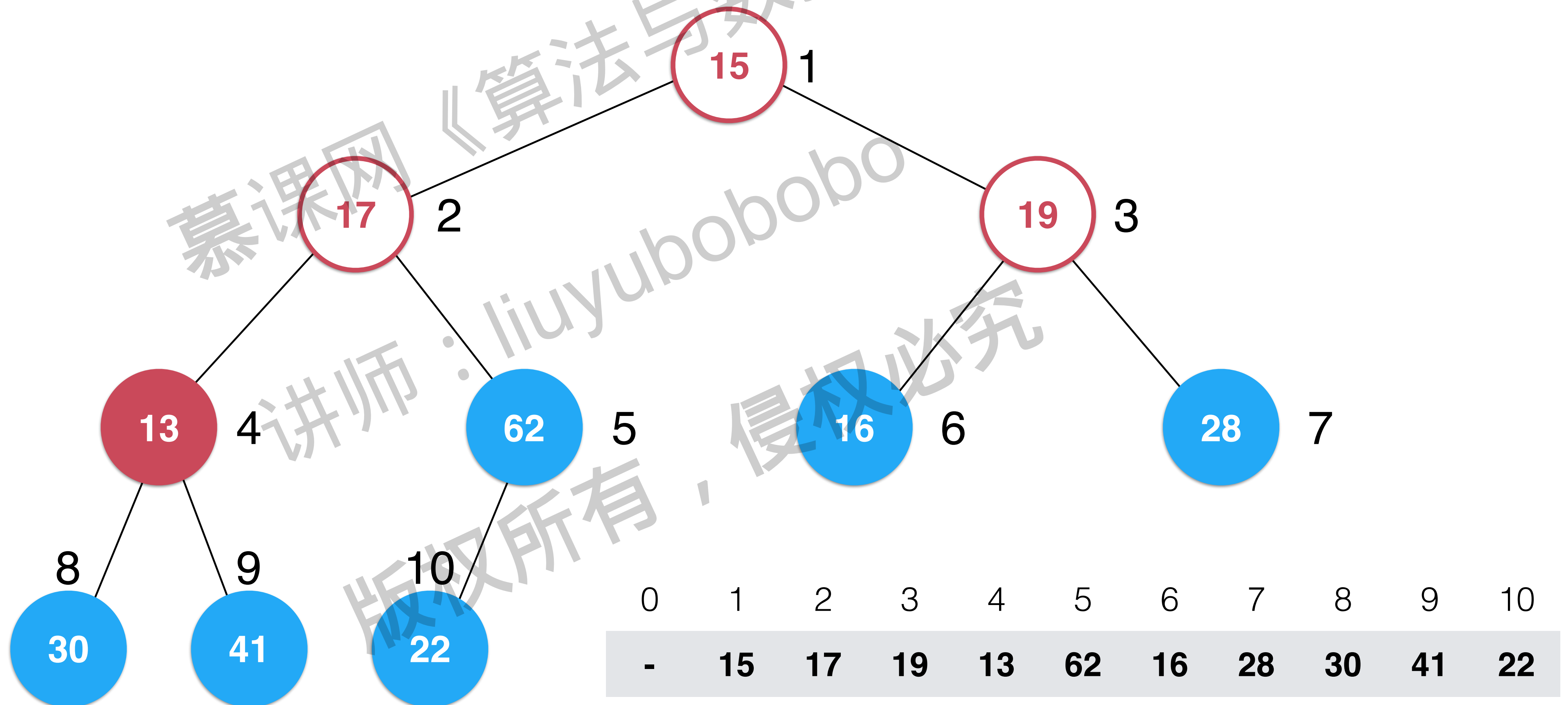
Heapify



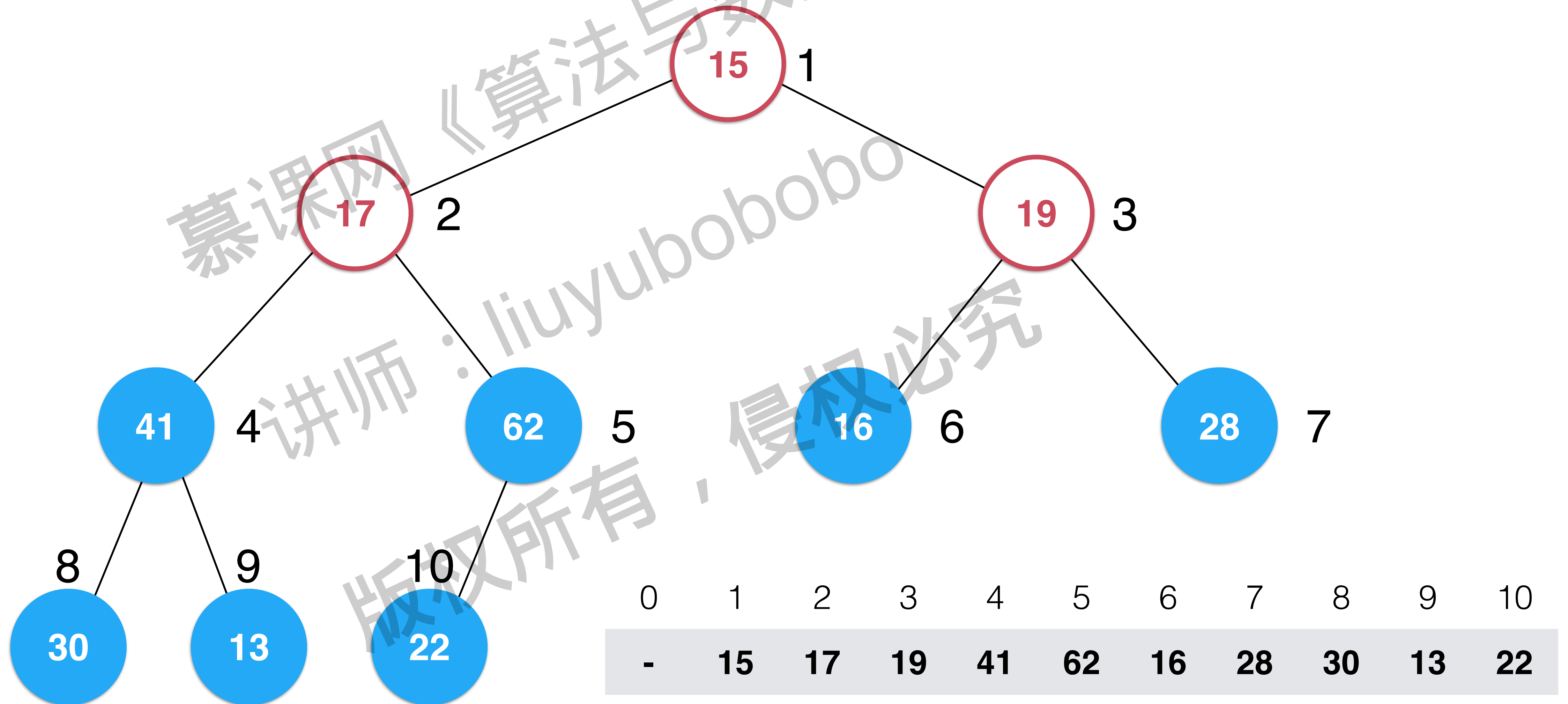
Heapify



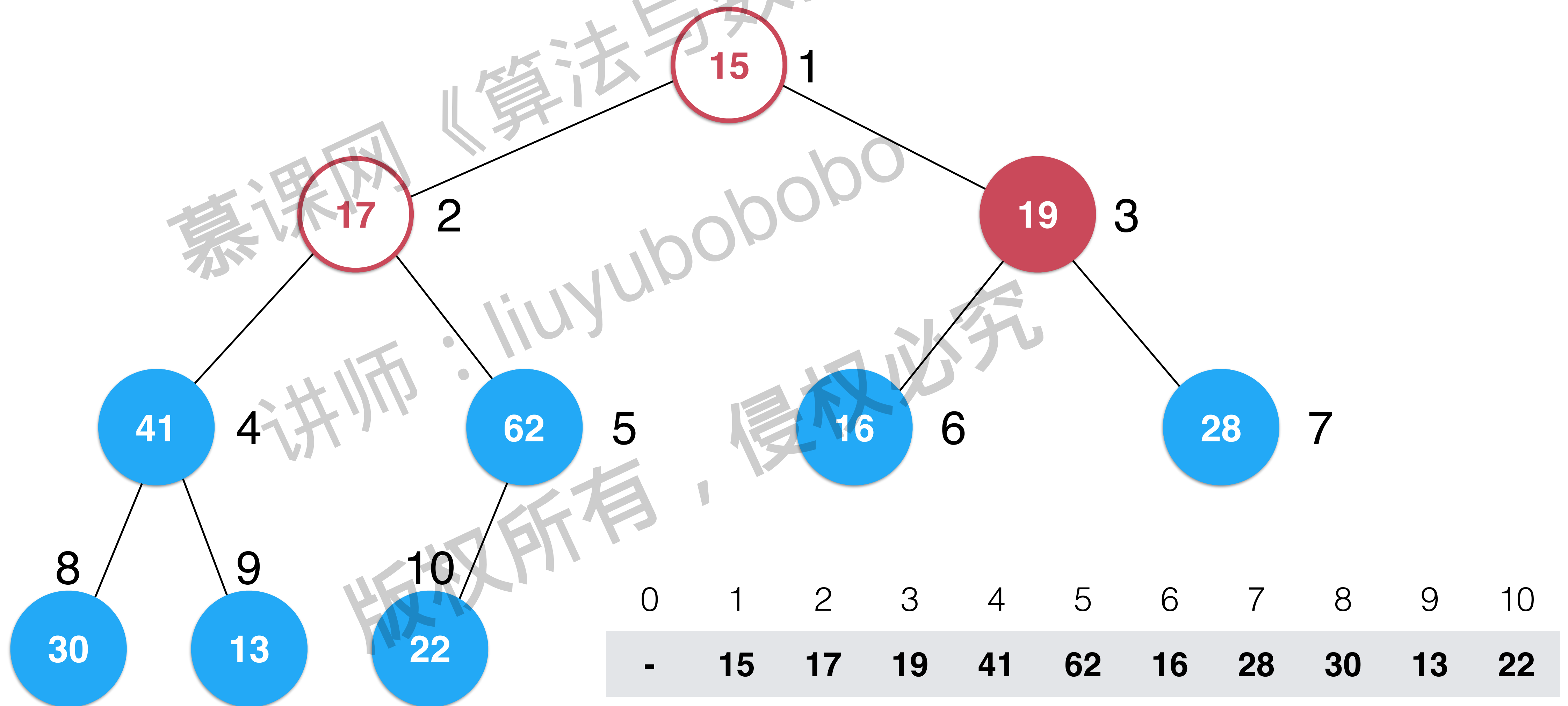
Heapify



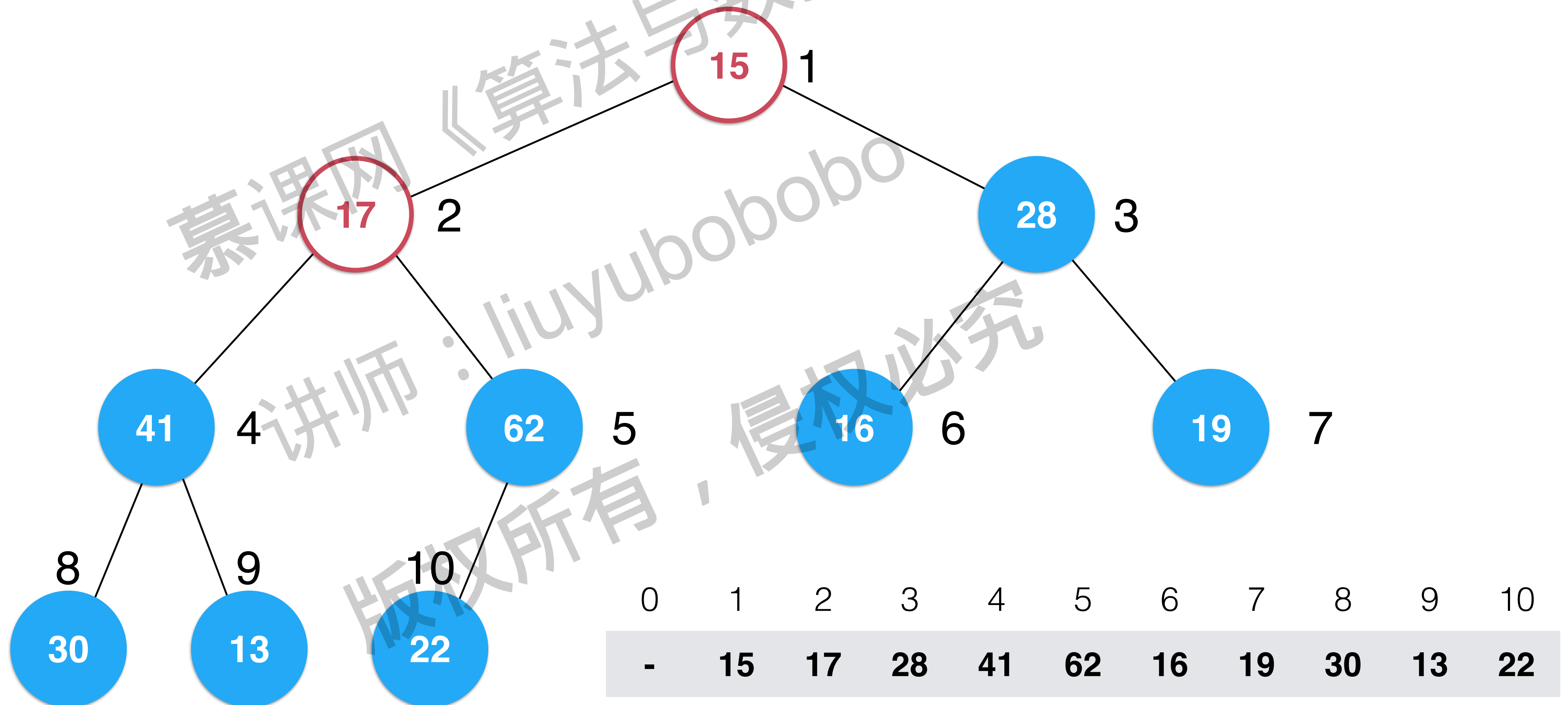
Heapify



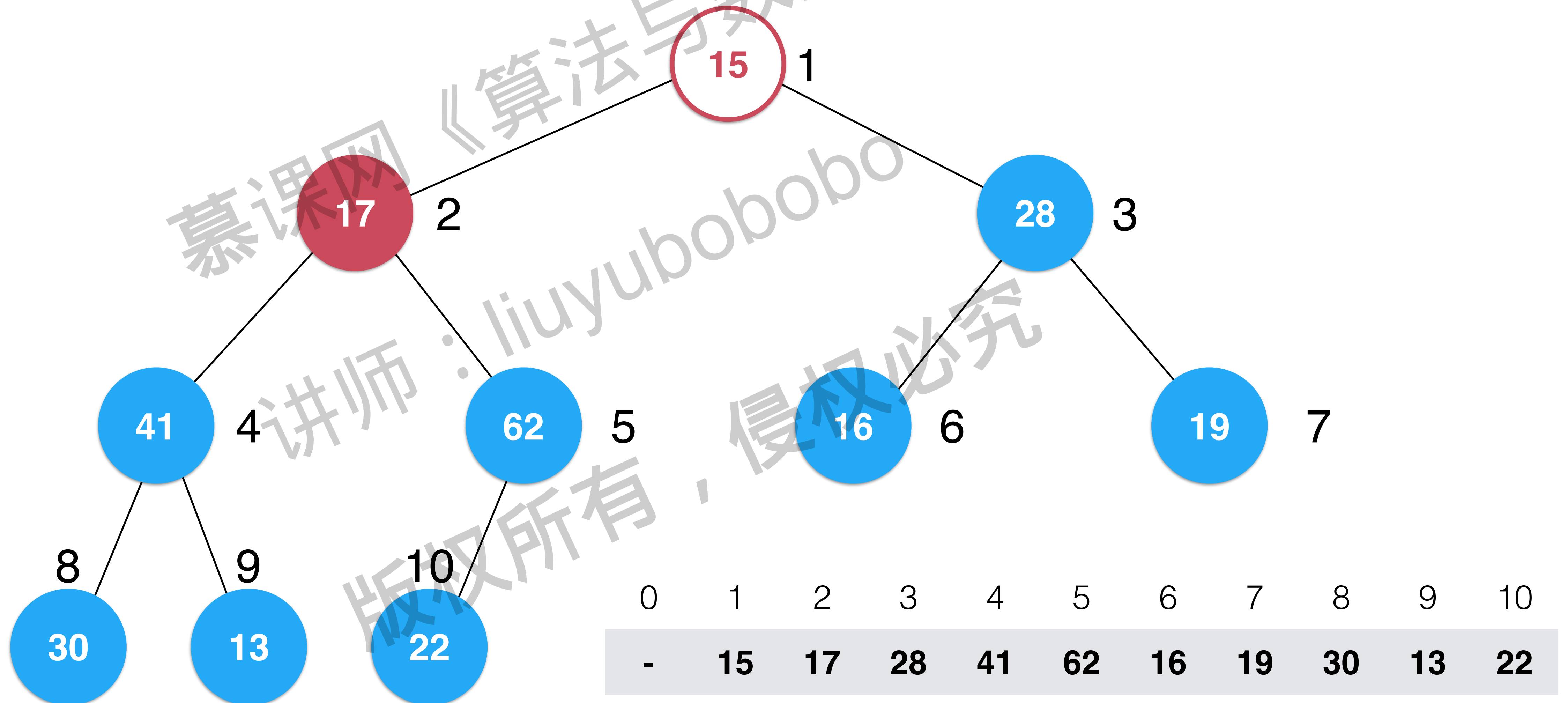
Heapify



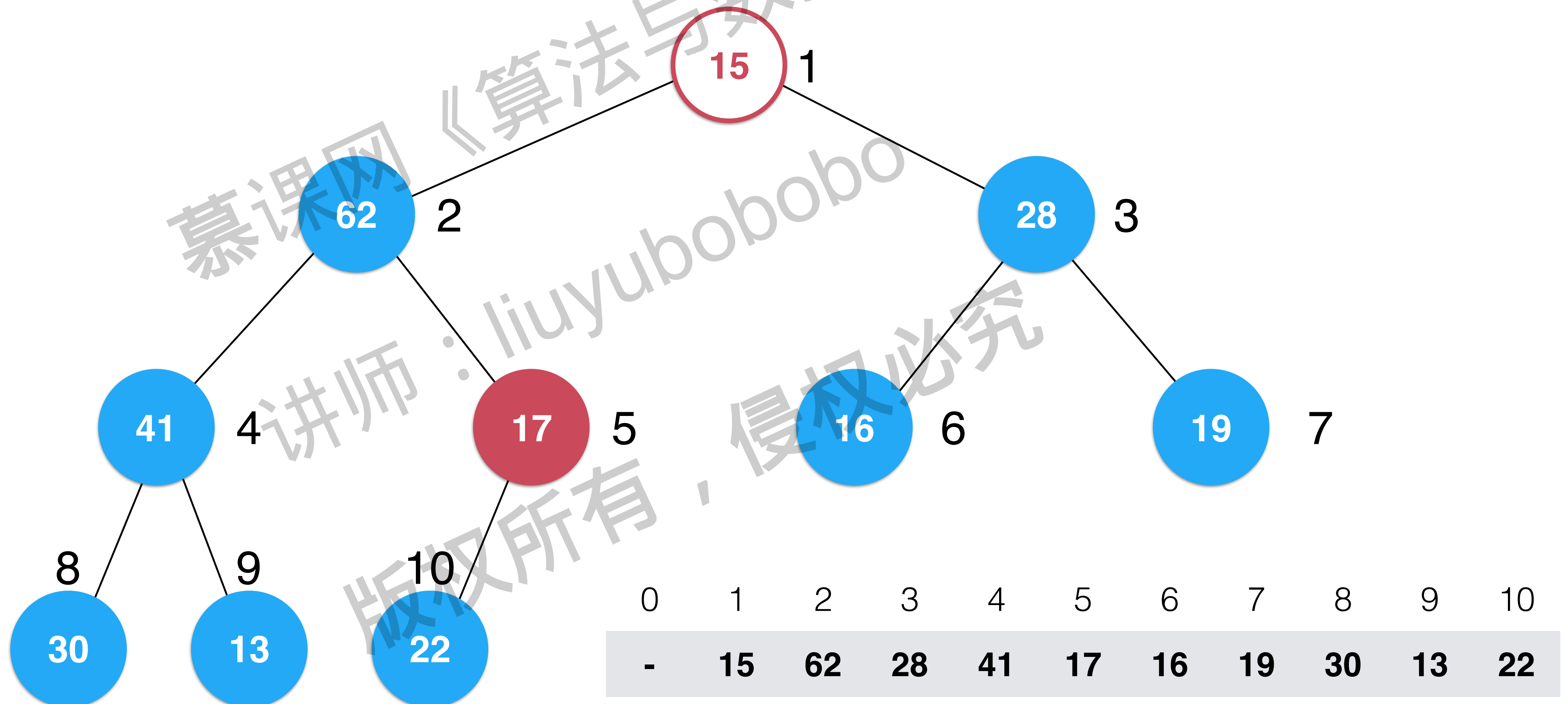
Heapify



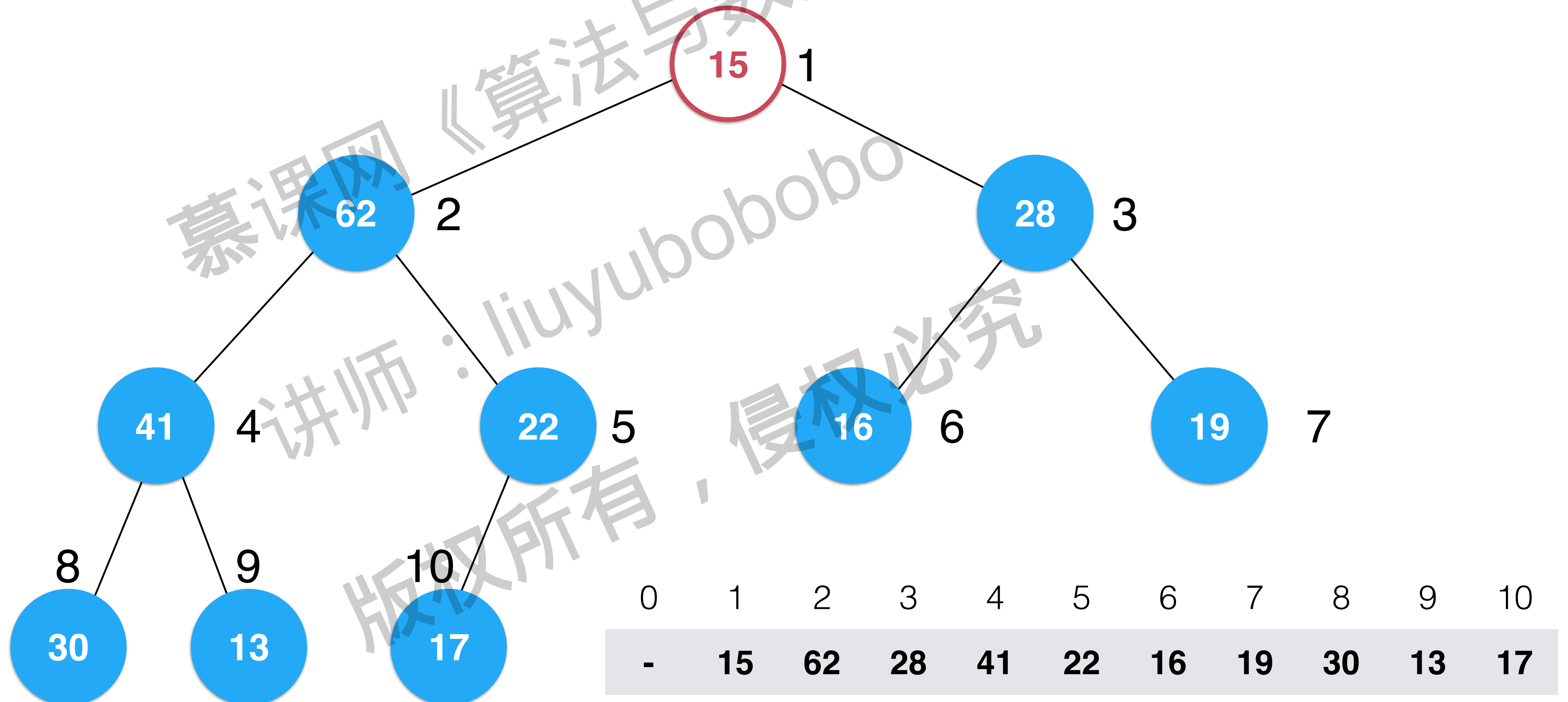
Heapify



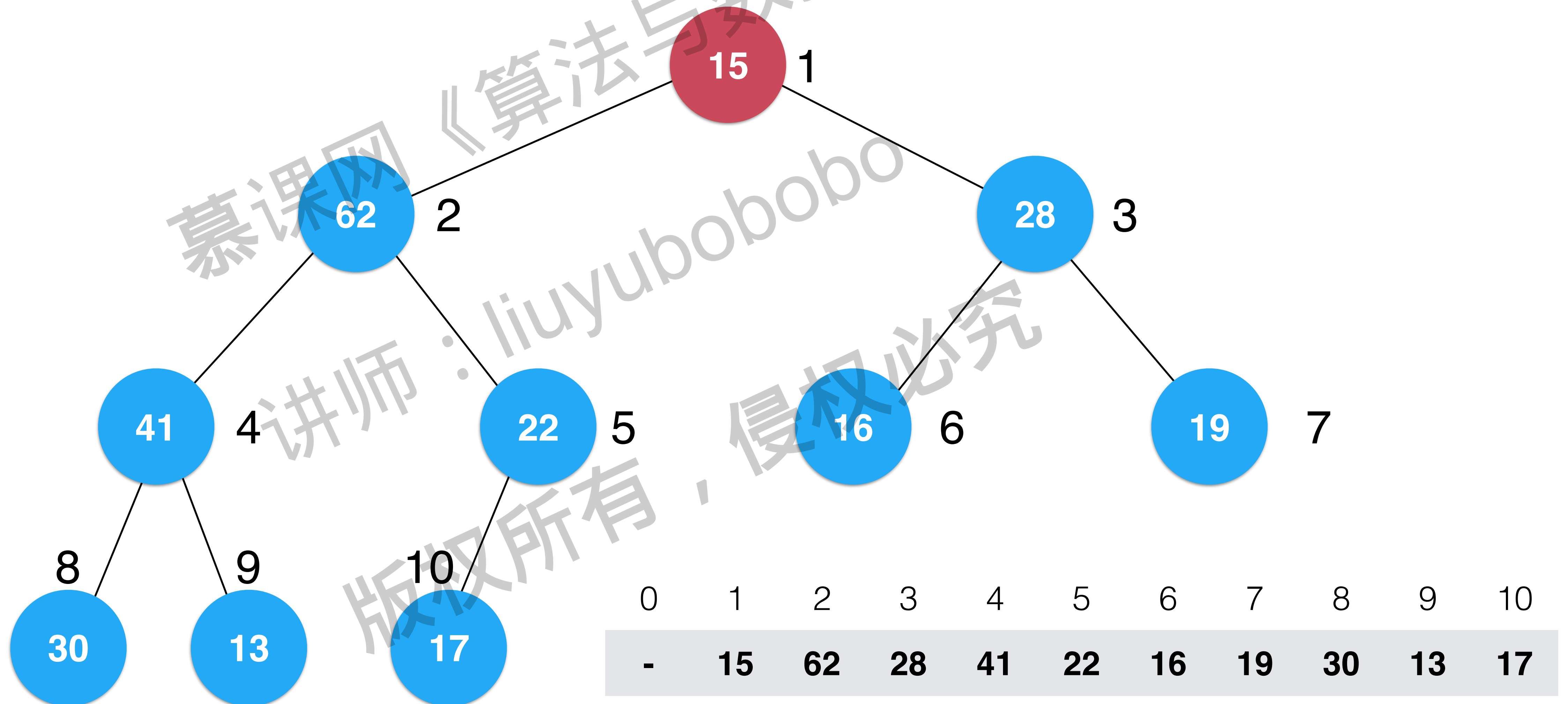
Heapify



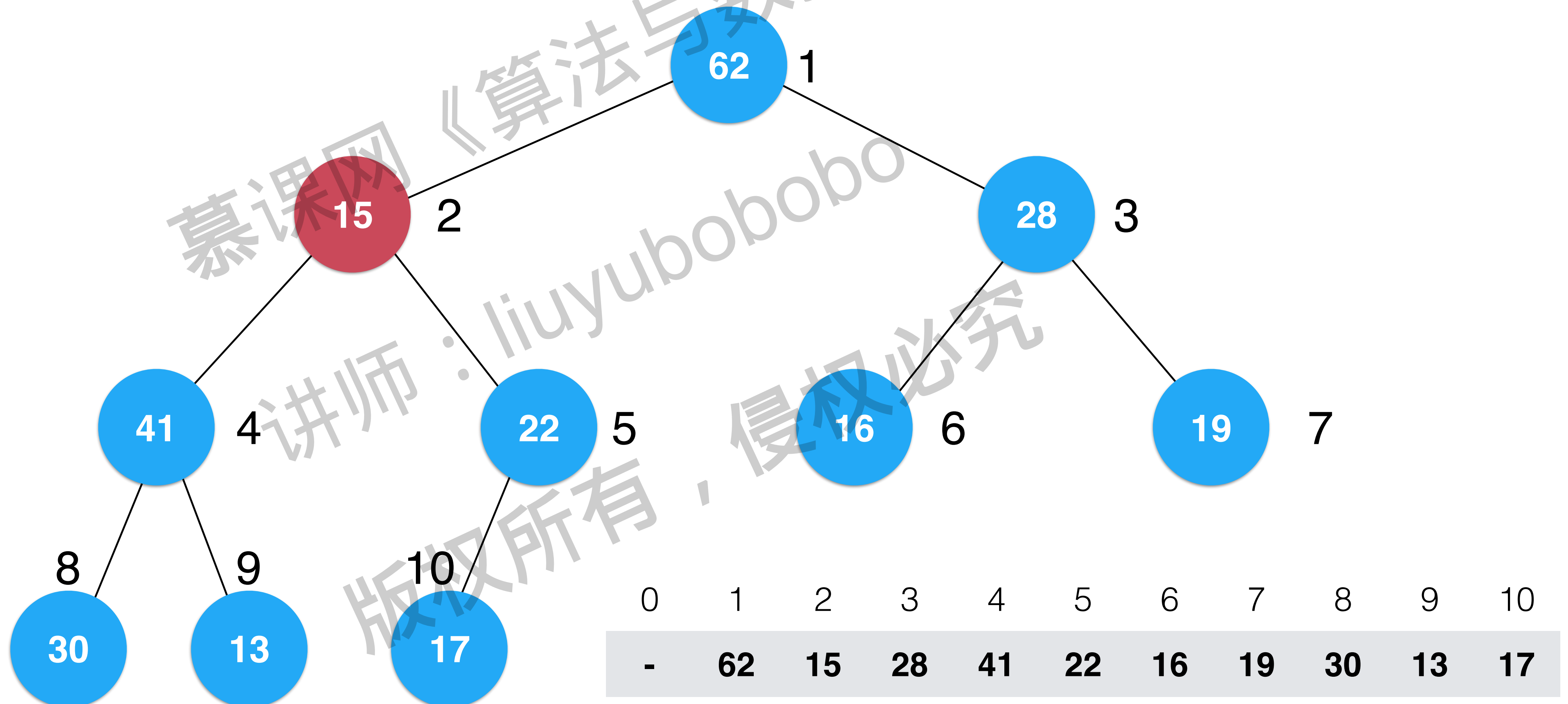
Heapify



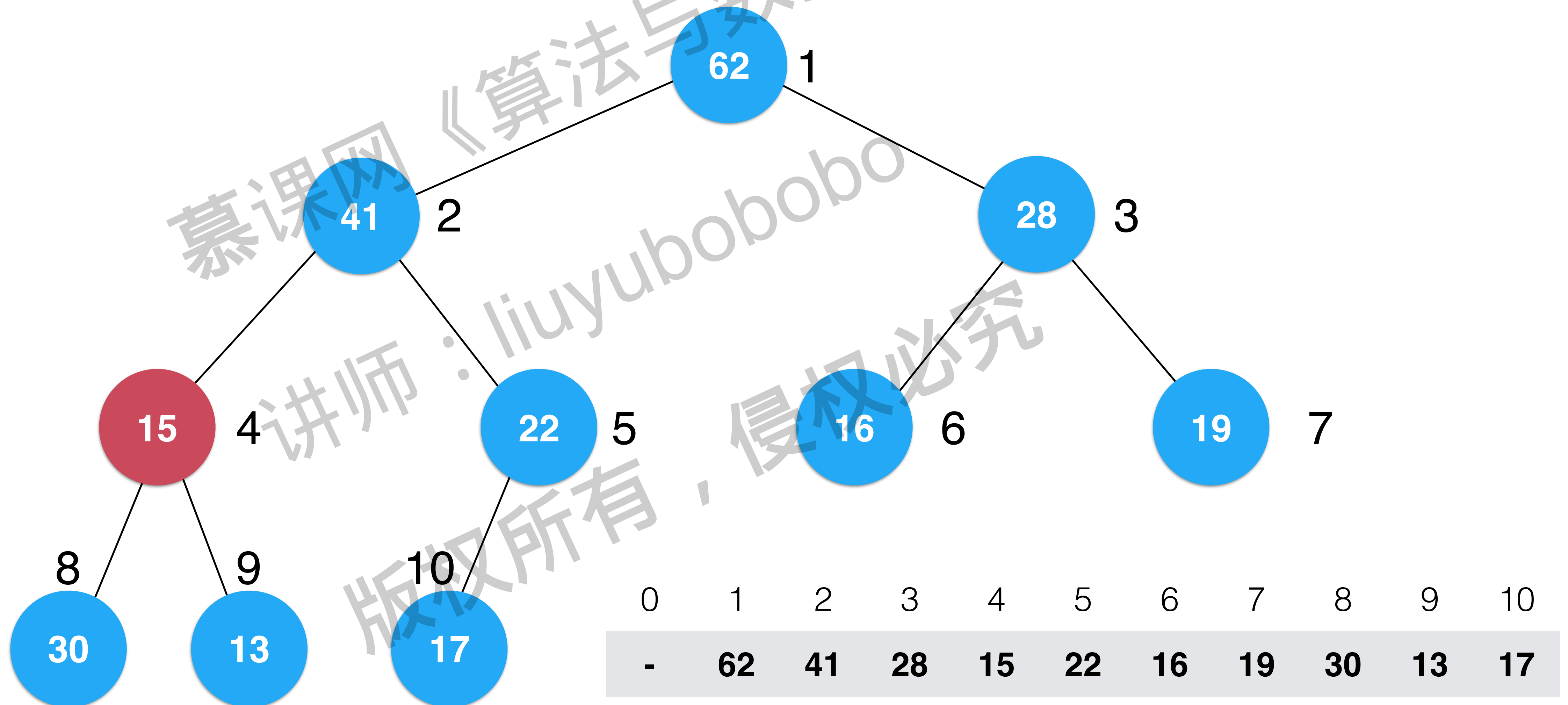
Heapify



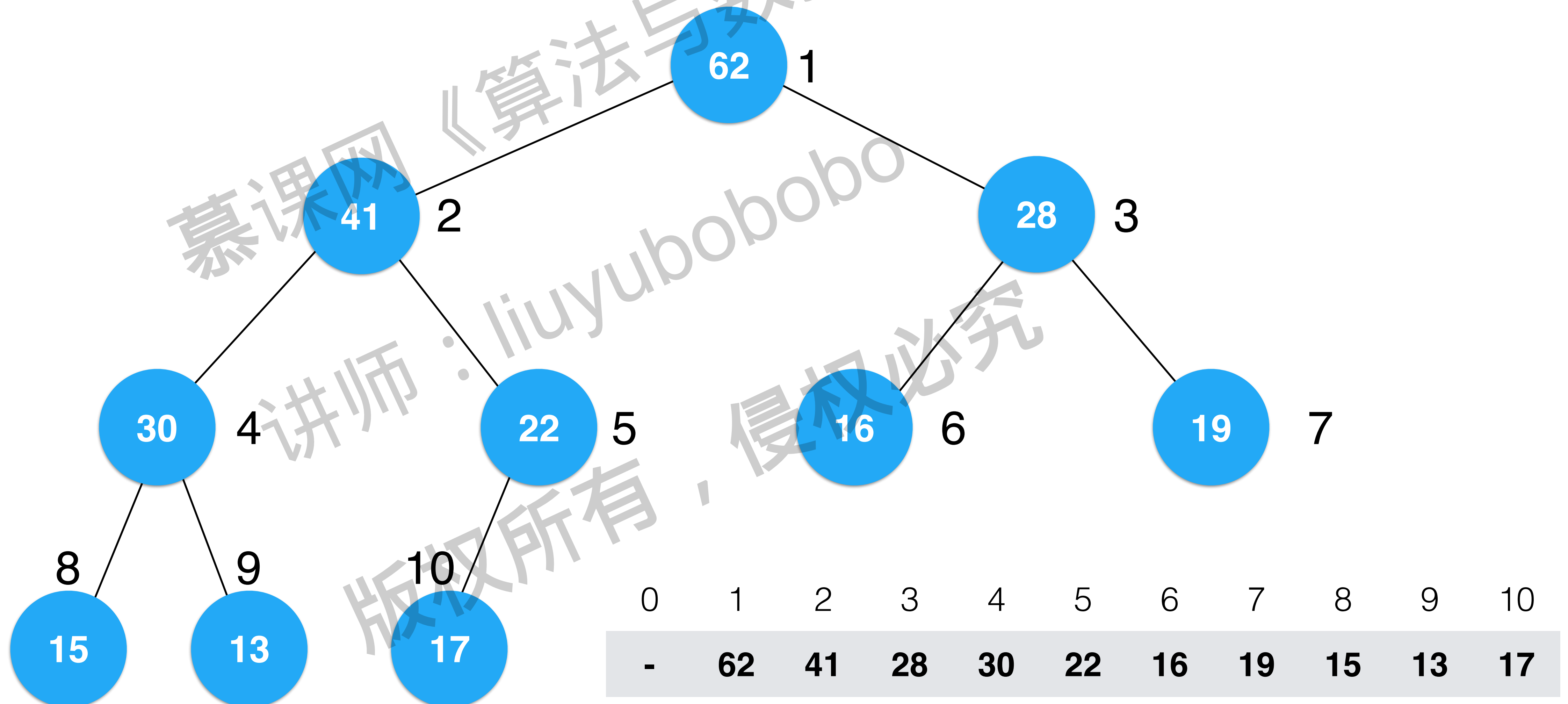
Heapify



Heapify



Heapify



慕课网《算法与数据结构》

操作：Heapify

讲师：liuyubobobo

版权所有，侵权必究

慕课网《算法与数据结构》

操作：使用Heapify的Heap Sort 和 Basic Heap Sort 的比较

讲师：liuyubobobo

版权所有，侵权必究

Heapify 的算法复杂度

将 n 个元素逐个插入到一个空堆中，算法复杂度是 $O(n\log n)$

heapify的过程，算法复杂度为 $O(n)$

慕课网《算法与数据结构》

原地堆排序

讲师：liuyubobobo

版权所有，侵权必究

原地堆排序

max



v

Max Heap

原地堆排序

max



v

Max Heap

w

原地堆排序

max



原地堆排序

max



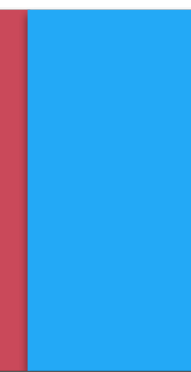
原地堆排序

max



v

Max Heap



原地堆排序

max



原地堆排序

max



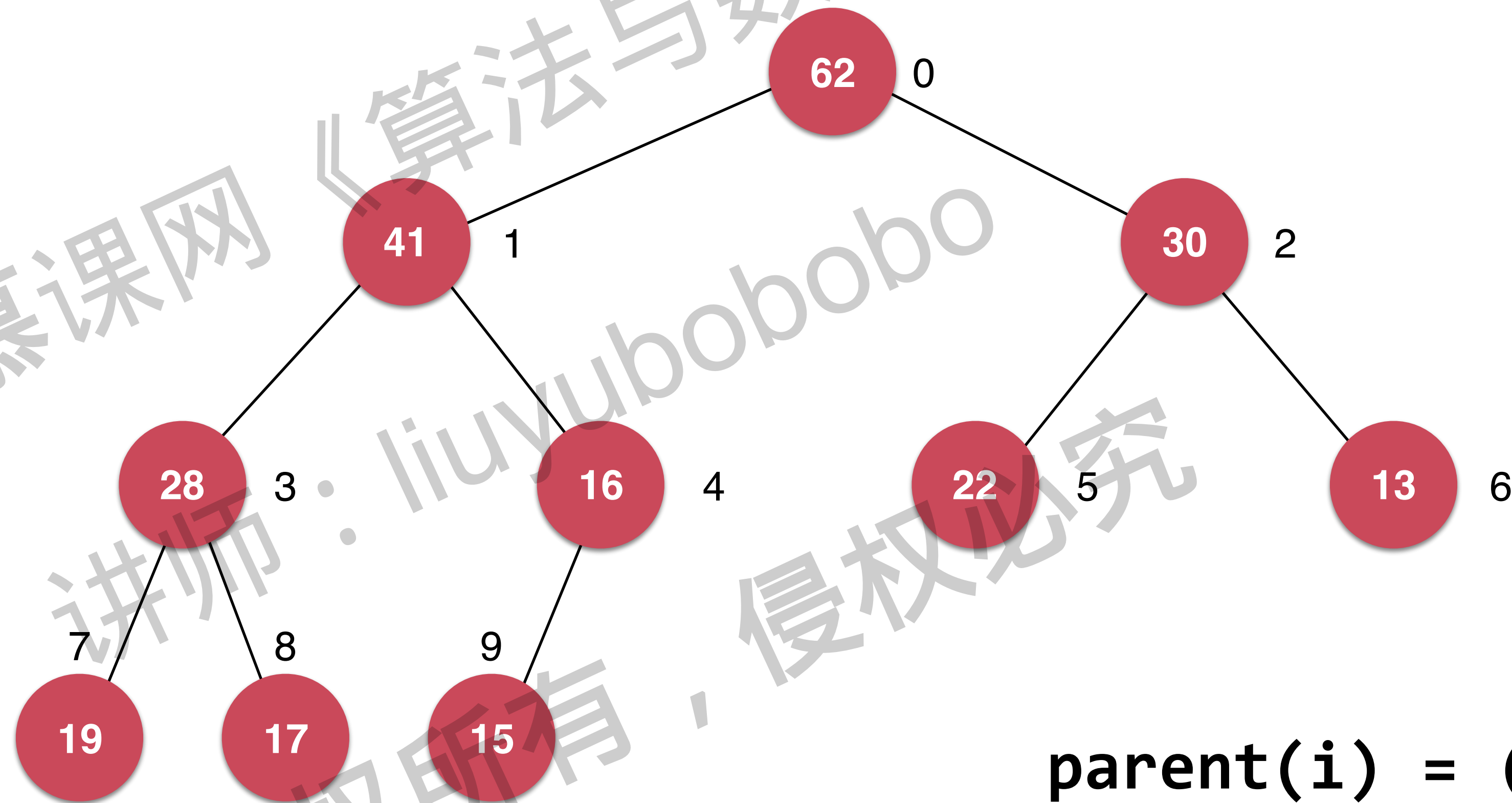
原地堆排序



原地堆排序

Max Heap

用数组存储二叉堆



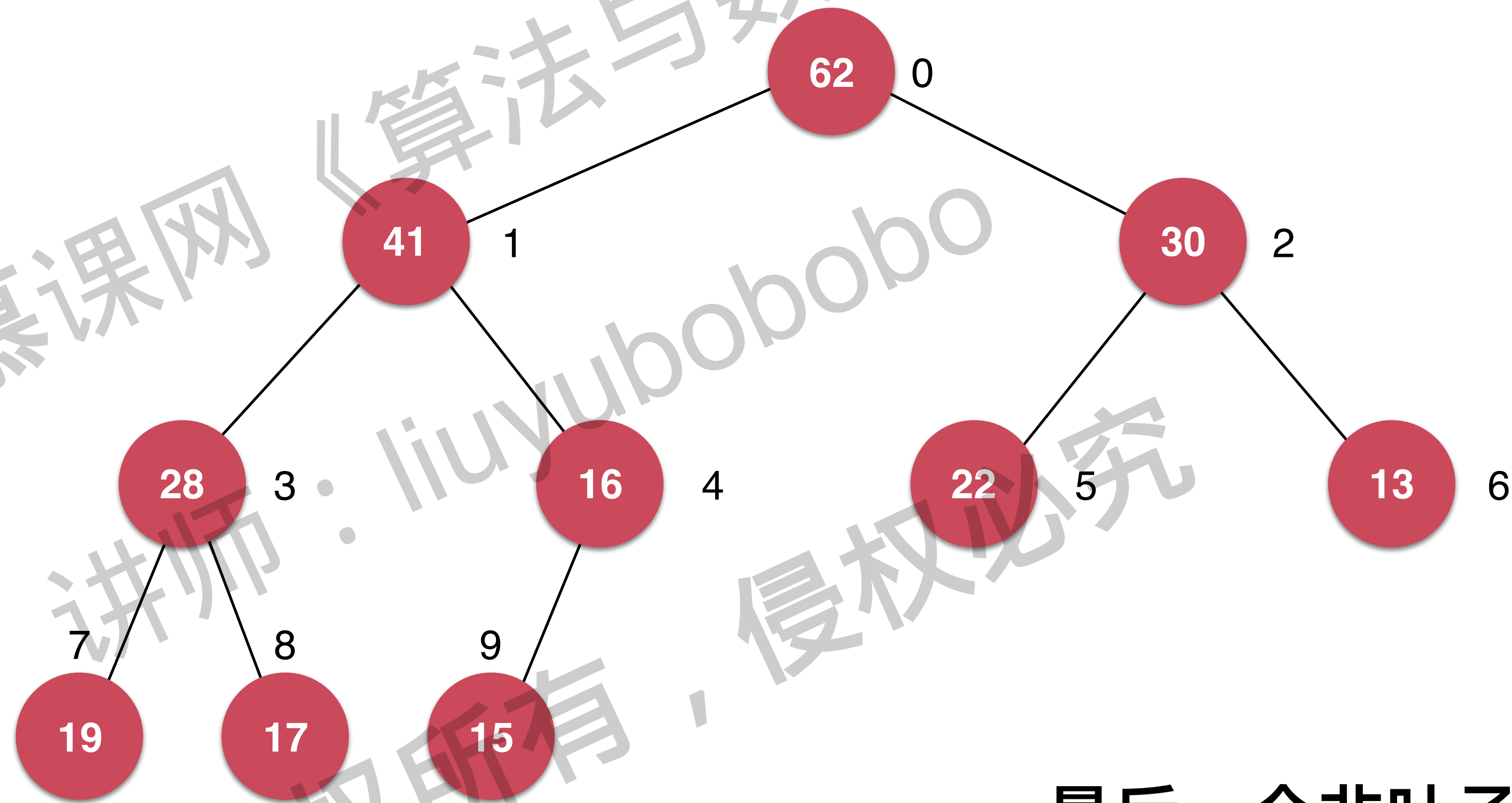
0	1	2	3	4	5	6	7	8	9
62	41	30	28	16	22	13	19	17	15

$$\text{parent}(i) = (i-1)/2$$

$$\text{left child } (i) = 2*i + 1$$

$$\text{right child } (i) = 2*i + 2$$

用数组存储二叉堆



最后一个非叶子节点的索引

$(\text{count}-2)/2$

0	1	2	3	4	5	6	7	8	9
62	41	30	28	16	22	13	19	17	15

慕课网《算法与数据结构》

操作：原地堆排序

讲师：liuyubobobo

版权所有，侵权必究

慕课网《算法与数据结构》

排序算法总结

讲师：liuyubobobo

版权所有，侵权必究

排序算法总结

平均时间复杂度

插入排序
Insertion Sort

$O(n^2)$

归并排序
Merge Sort

$O(n \log n)$

快速排序
Quick Sort

$O(n \log n)$

堆排序
Heap Sort

$O(n \log n)$

排序算法总结

	平均时间复杂度	原地排序
插入排序 Insertion Sort	$O(n^2)$	✓
归并排序 Merge Sort	$O(n \log n)$	×
快速排序 Quick Sort	$O(n \log n)$	✓
堆排序 Heap Sort	$O(n \log n)$	✓

排序算法总结

	平均时间复杂度	原地排序	额外空间
插入排序 Insertion Sort	$O(n^2)$	✓	$O(1)$
归并排序 Merge Sort	$O(n \log n)$	×	$O(n)$
快速排序 Quick Sort	$O(n \log n)$	✓	$O(\log n)$
堆排序 Heap Sort	$O(n \log n)$	✓	$O(1)$

排序算法总结

	平均时间复杂度	原地排序	额外空间	稳定排序
插入排序 Insertion Sort	$O(n^2)$	✓	$O(1)$	✓
归并排序 Merge Sort	$O(n \log n)$	×	$O(n)$	✓
快速排序 Quick Sort	$O(n \log n)$	✓	$O(\log n)$	×
堆排序 Heap Sort	$O(n \log n)$	✓	$O(1)$	×

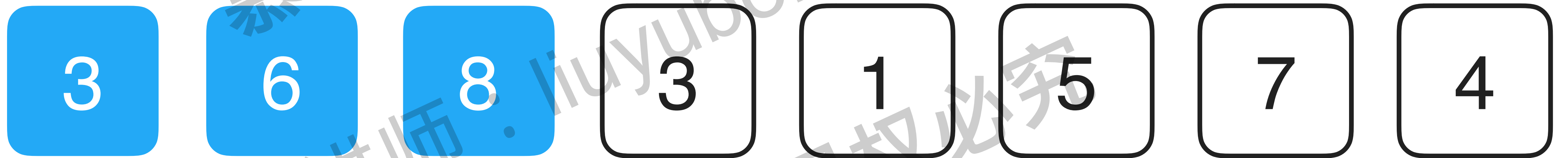
排序算法的稳定性 Stable

稳定排序：对于相等的元素，在排序后，原来靠前的元素依然靠前。

相等元素的相对位置没有发生改变。



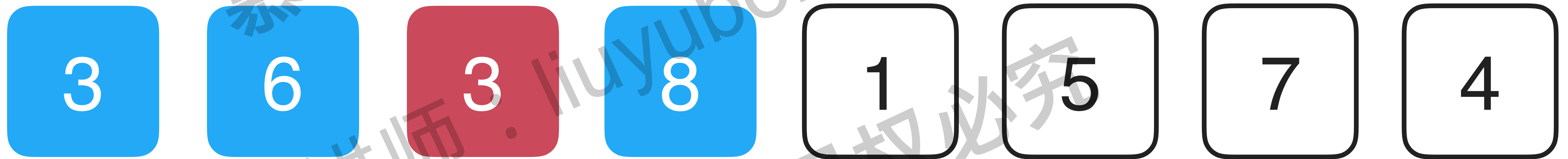
插入排序 Insertion Sort



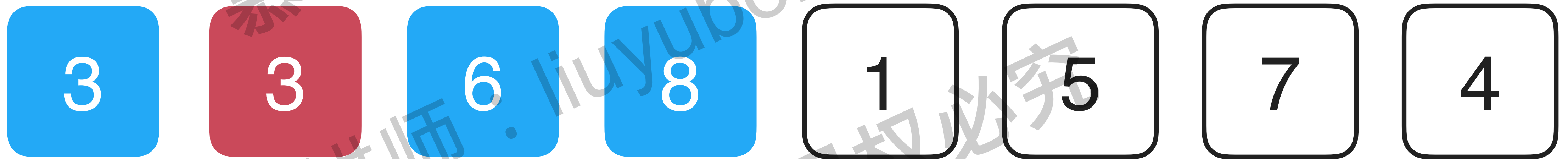
插入排序 Insertion Sort



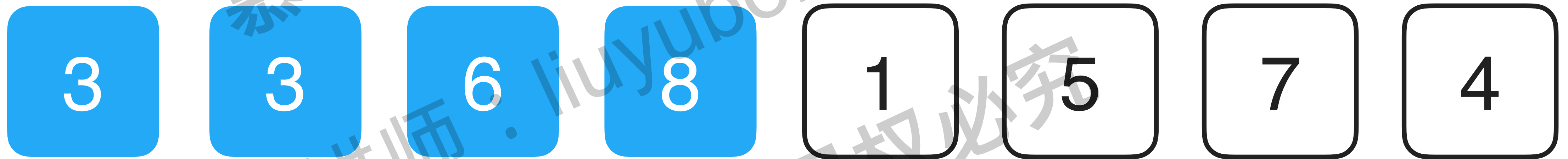
插入排序 Insertion Sort



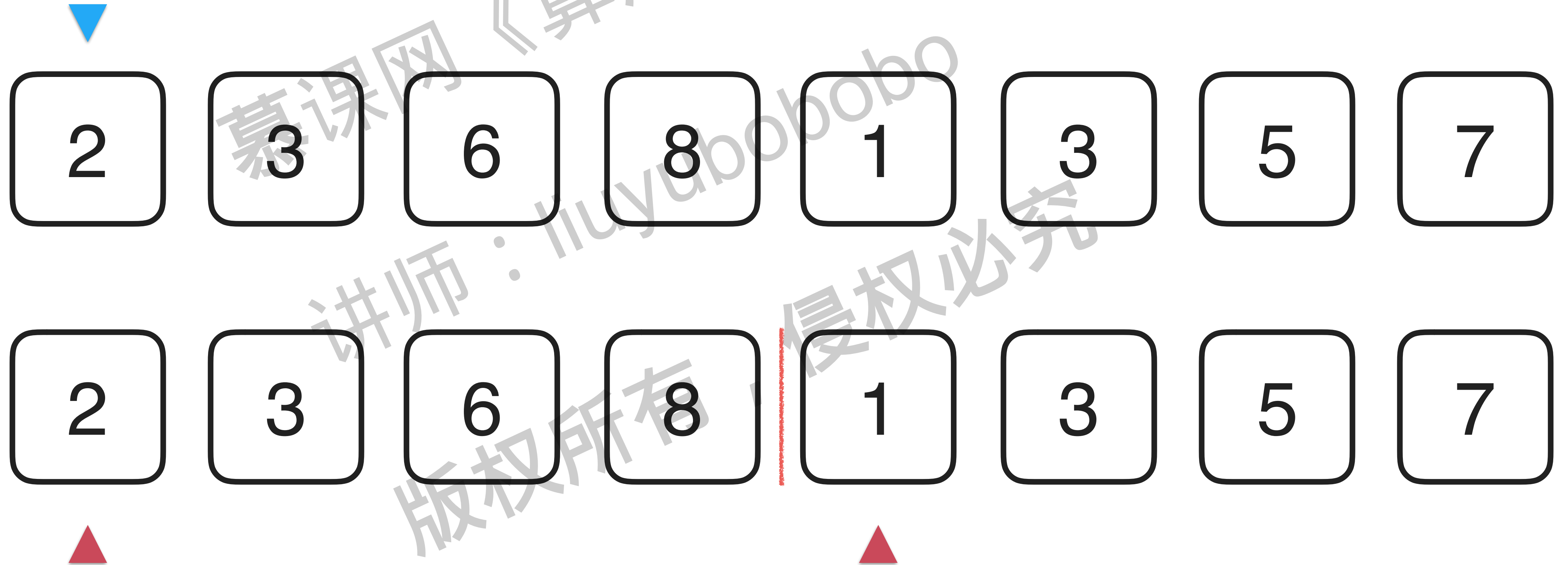
插入排序 Insertion Sort



插入排序 Insertion Sort



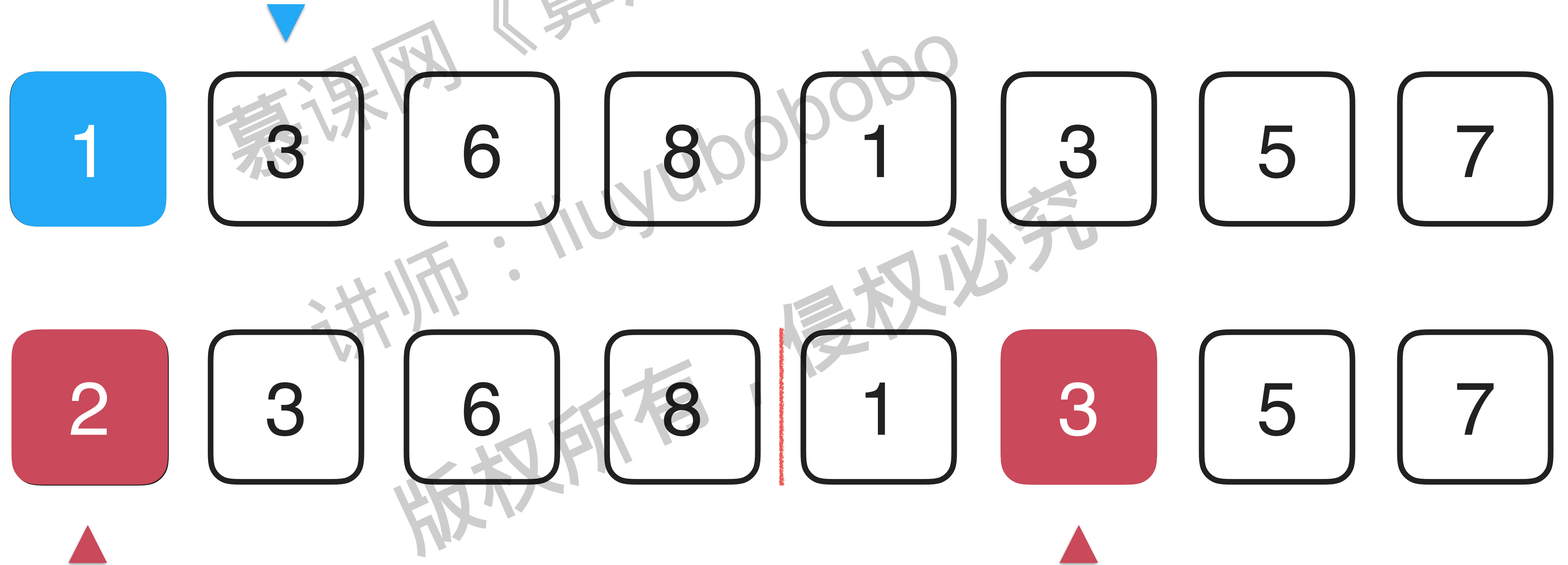
归并过程 Merge



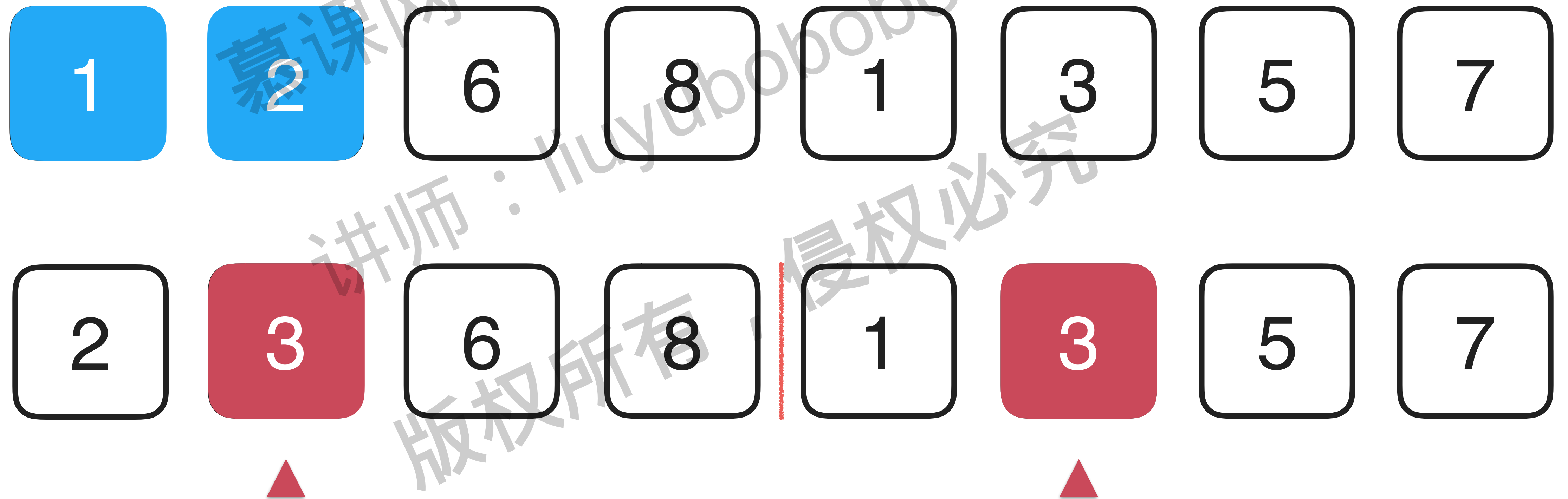
归并过程 Merge



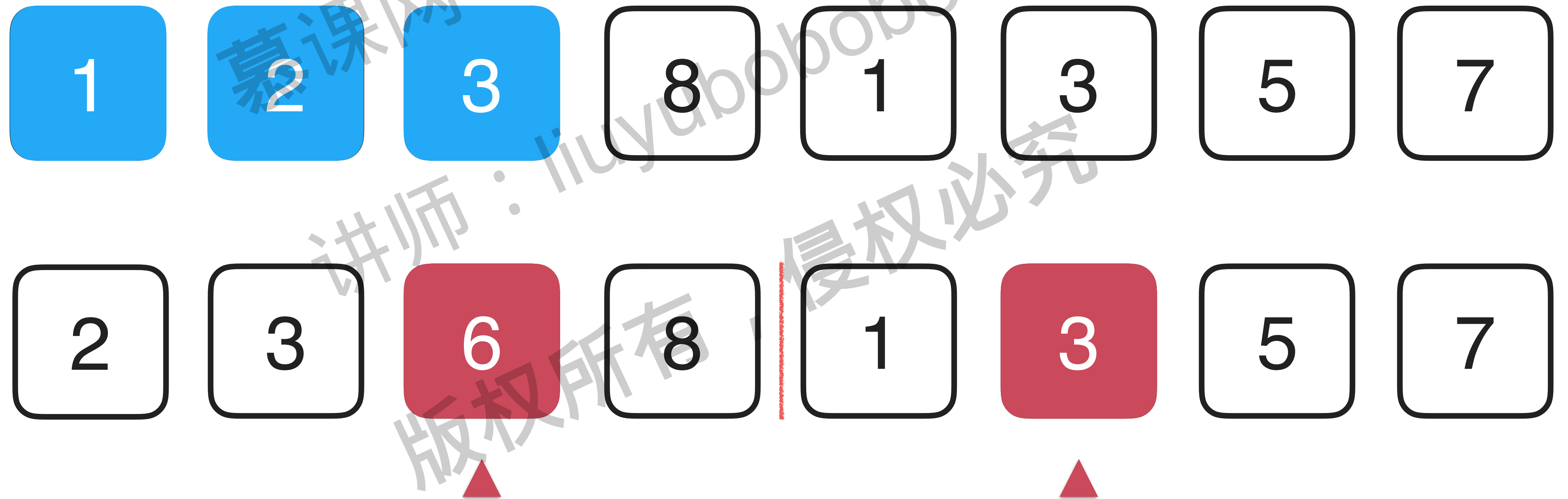
归并过程 Merge



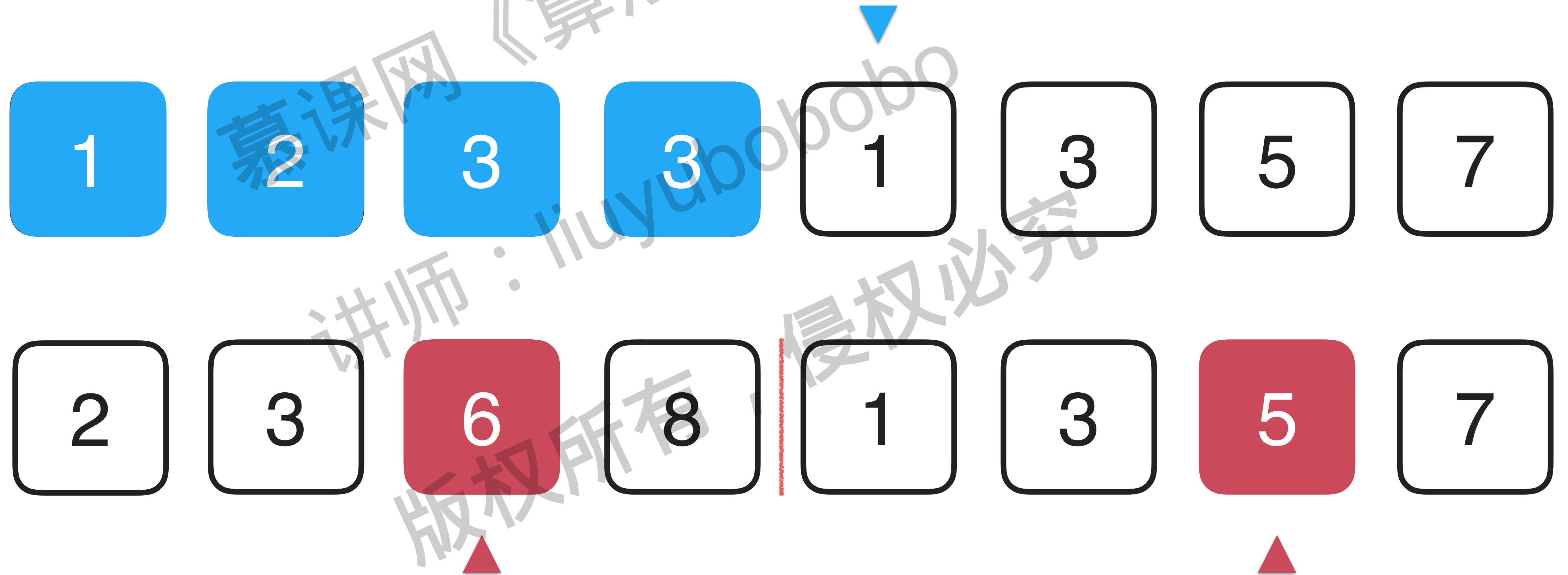
归并过程 Merge



归并过程 Merge



归并过程 Merge



排序算法的稳定性 Stable

可以通过自定义比较函数，让排序算法不存在稳定性的问题。

```
bool operator<(const Student& otherStudent){  
    return score != otherStudent.score ?  
        score > otherStudent.score :  
        name < otherStudent.name;  
}
```

	平均时间复杂度	原地排序	额外空间	稳定排序
插入排序 Insertion Sort	$O(n^2)$	✓	$O(1)$	✓
归并排序 Merge Sort	$O(n \log n)$	✗	$O(n)$	✓
快速排序 Quick Sort	$O(n \log n)$	✓	$O(\log n)$	✗
堆排序 Heap Sort	$O(n \log n)$	✓	$O(1)$	✗
神秘的排序算法?	$O(n \log n)$	✓	$O(1)$	✓

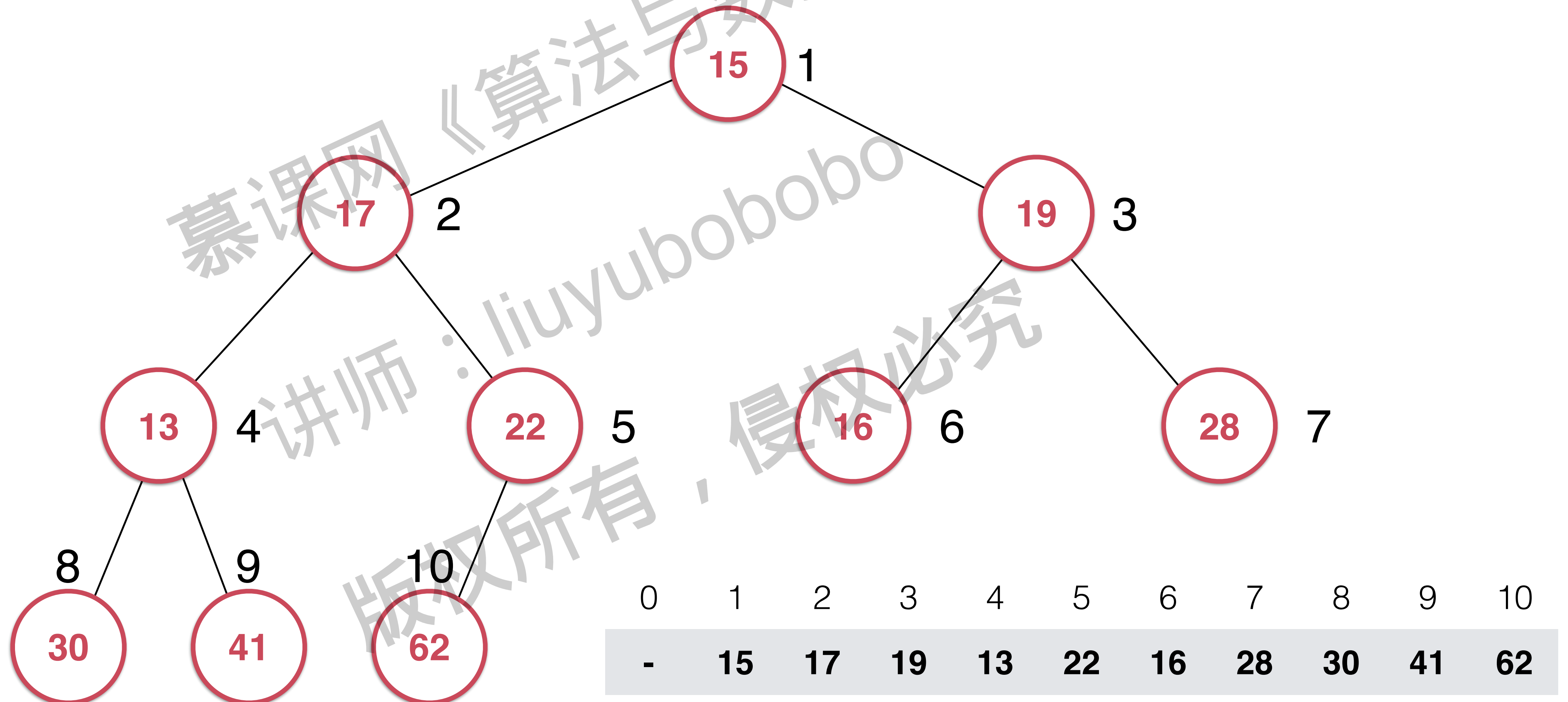
慕课网《算法与数据结构》

索引堆 Index Heap

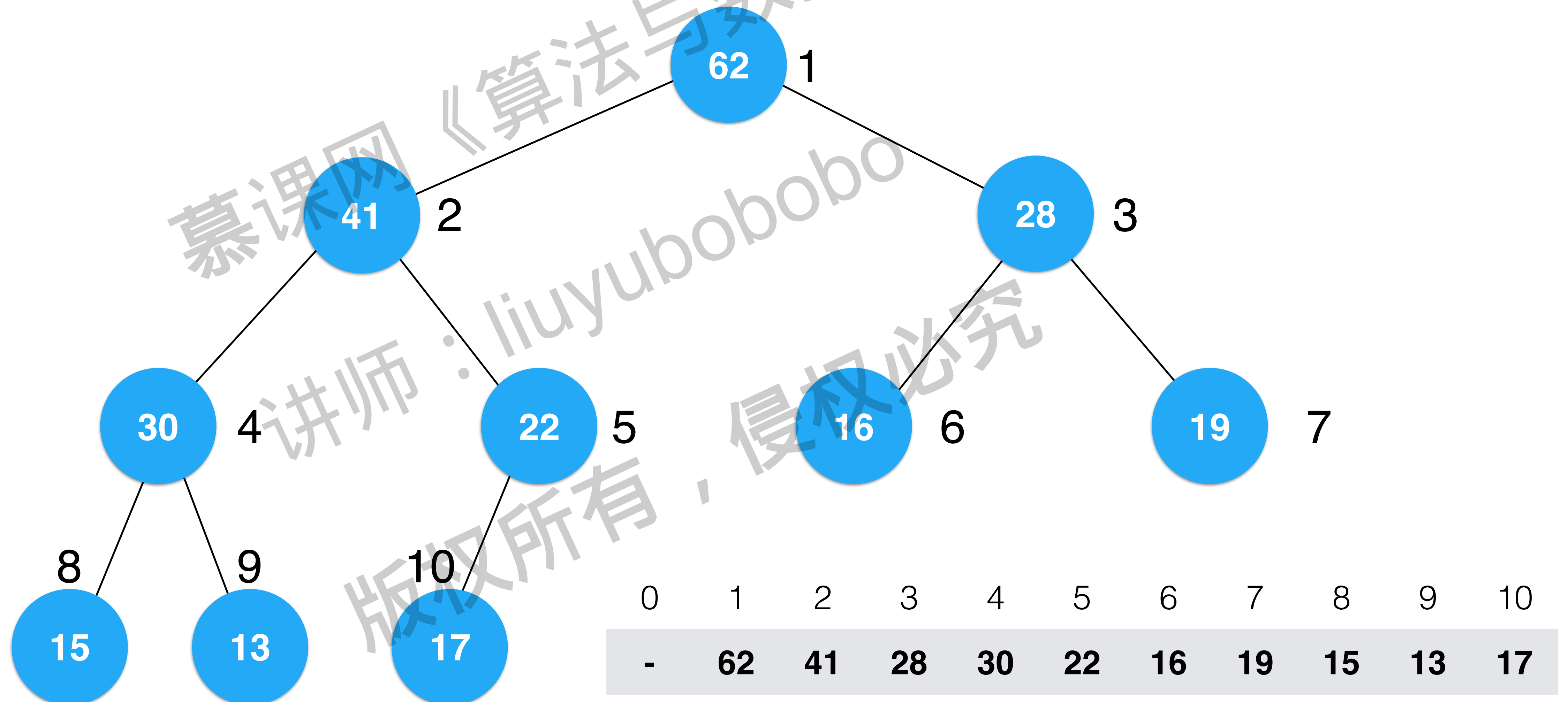
讲师：liuyubobobo

版权所有，侵权必究

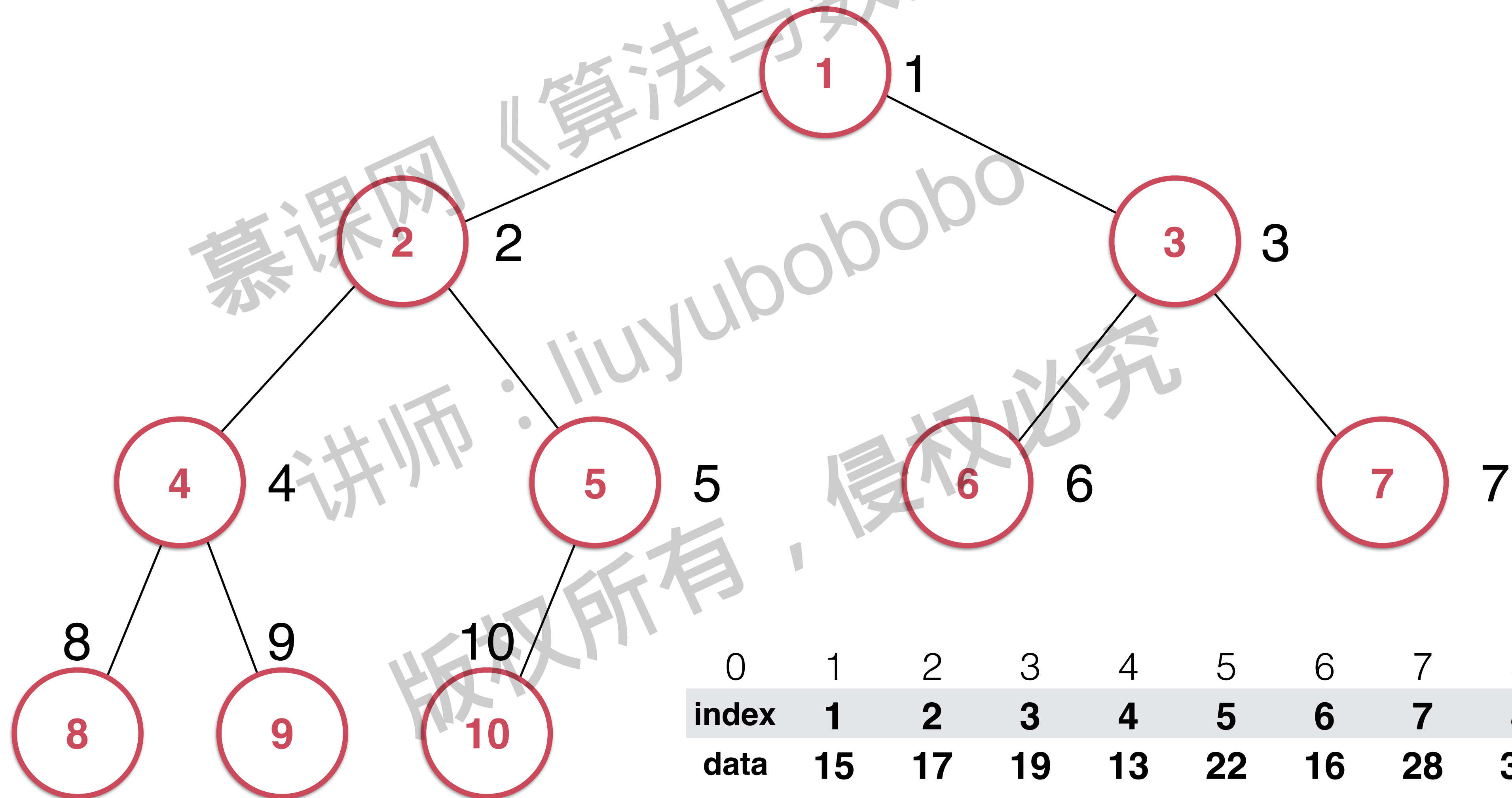
Heap



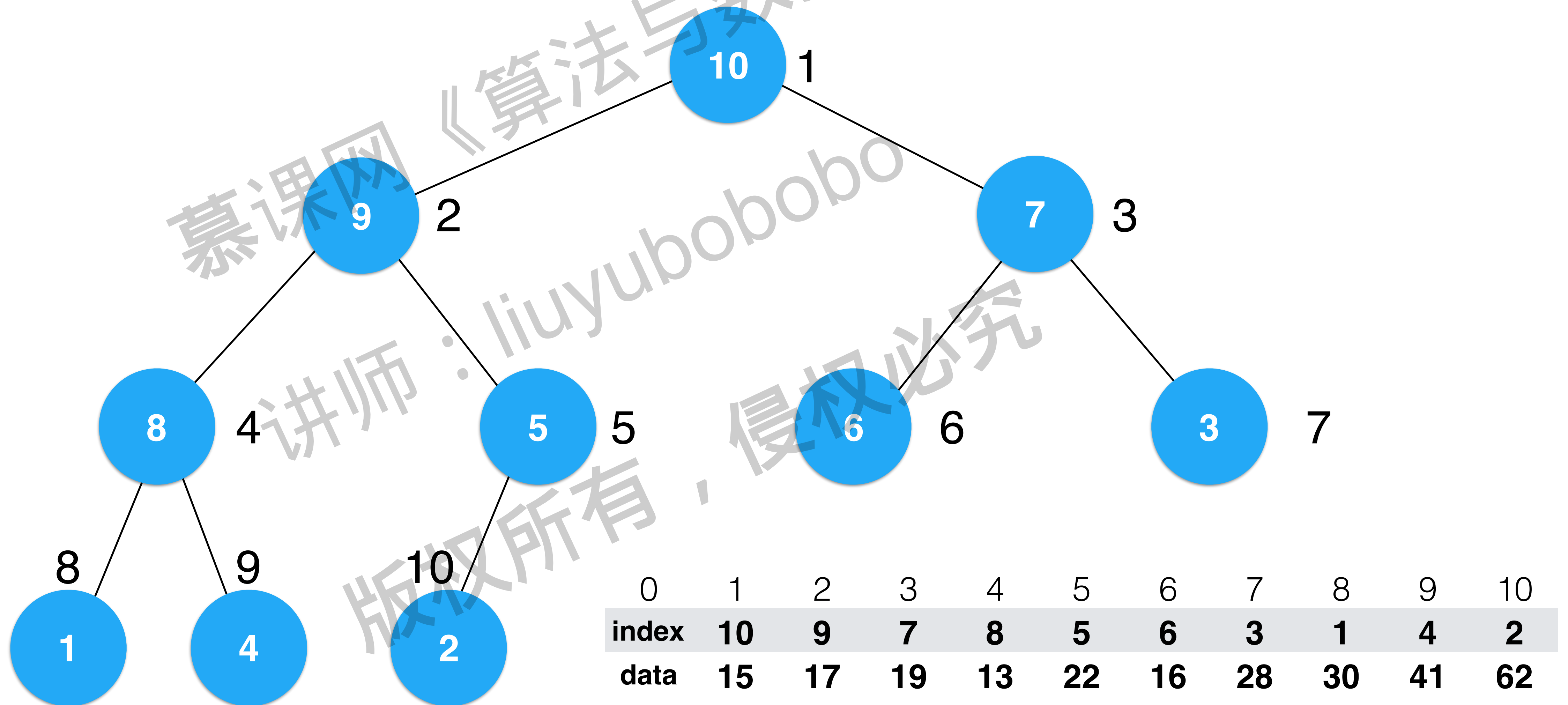
Heap



Index Max Heap



Index Max Heap



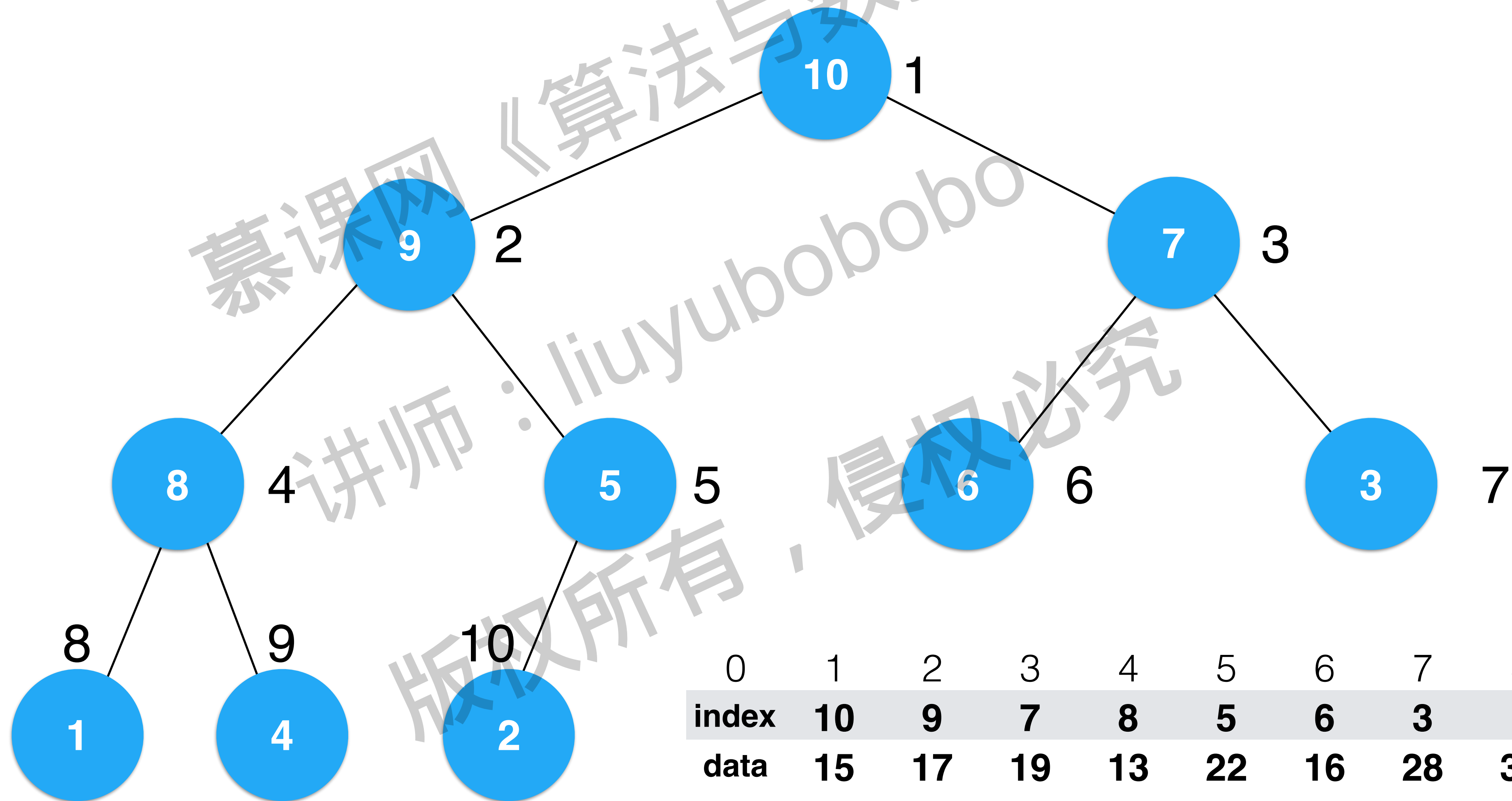
操作：编写基础 Index Max Heap

使用reverse数组反向查找

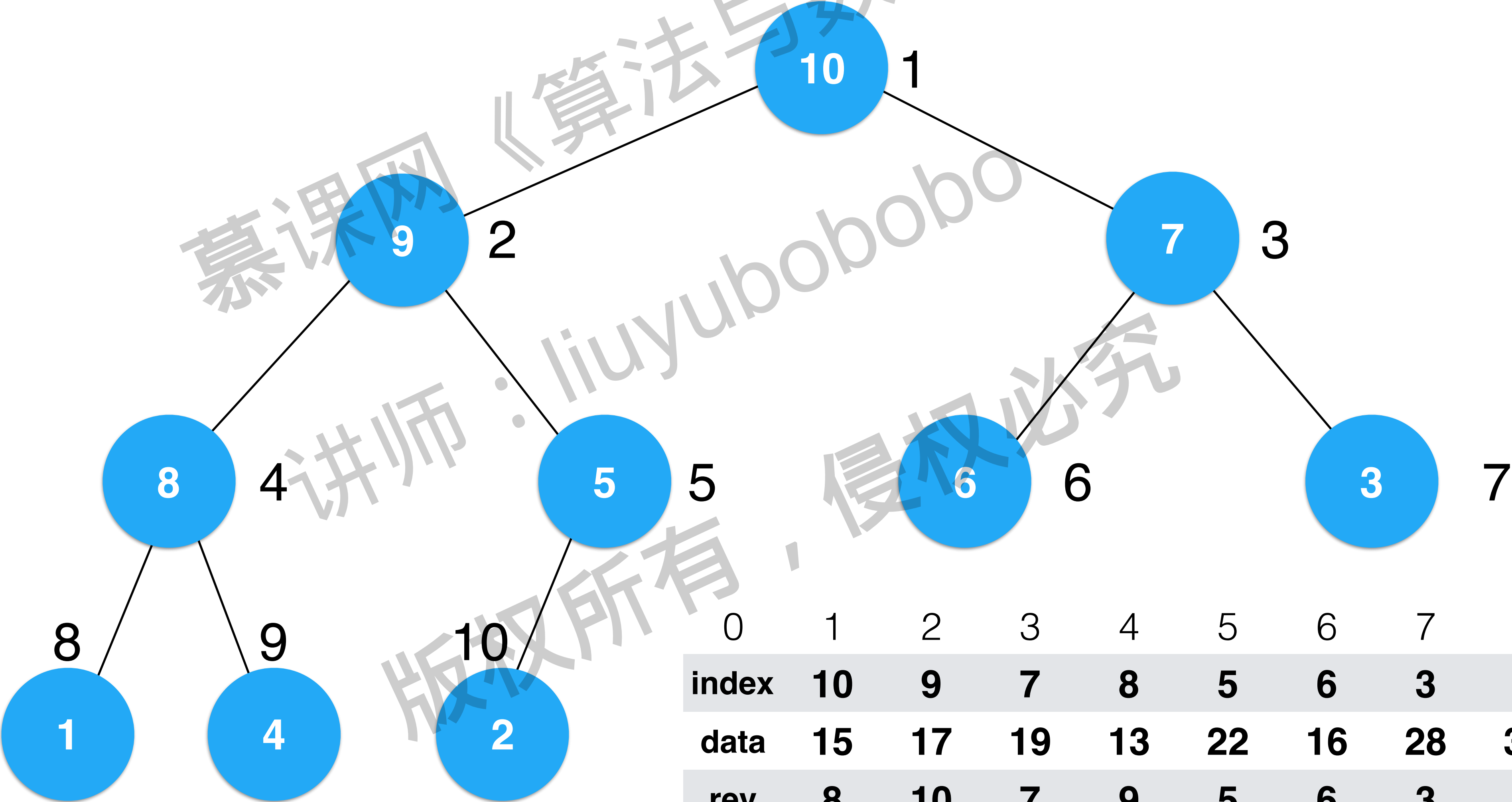
讲师：liuyubobobo

版权所有，侵权必究

Index Max Heap



Index Max Heap



	0	1	2	3	4	5	6	7	8	9	10
index	10	9	7	8	5	6	3	1	4	2	
data	15	17	19	13	22	16	28	30	41	62	
rev	8	10	7	9	5	6	3	4	2	1	

Index Max Heap

	0	1	2	3	4	5	6	7	8	9	10
index		10	9	7	8	5	6	3	1	4	2
data		15	17	19	13	22	16	28	30	41	62
rev		8	10	7	9	5	6	3	4	2	1

`reverse[i]` 表示索引*i*在indexes(堆)中的位置

`indexes[i] = j`

`reverse[j] = i`

`indexes[reverse[i]] = i`

`reverse[indexes[i]] = i`

操作：编写拥有reverse反向查找的Index Max Heap

慕课网《算法与数据结构》

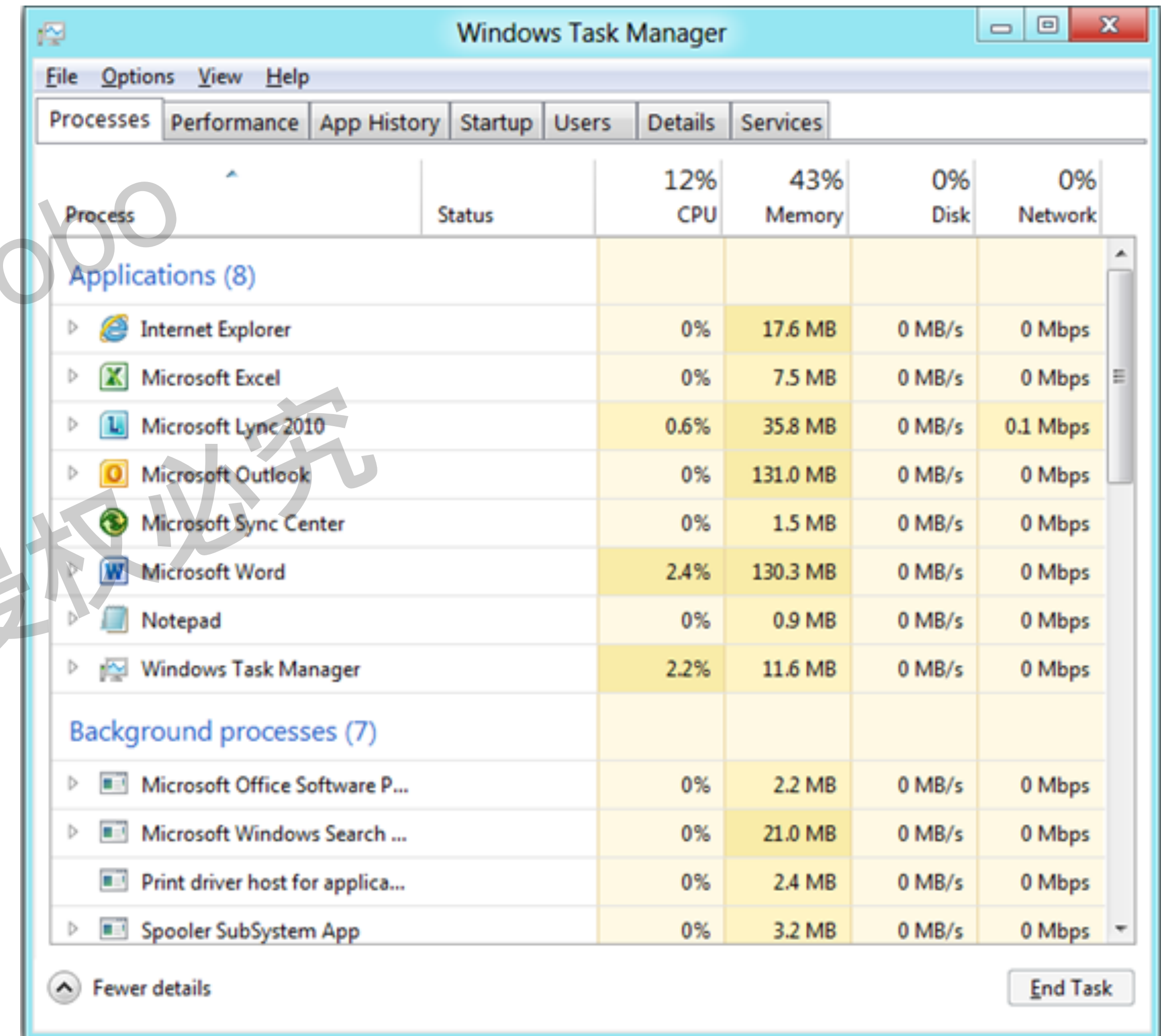
和堆相关的问题

讲师：liuyubobobo

版权所有，侵权必究

使用堆实现优先队列

动态选择优先级最高的任务执行



The screenshot shows the Windows Task Manager window with the 'Processes' tab selected. It displays a list of running applications and background processes, sorted by CPU usage. The table includes columns for Process, Status, CPU, Memory, Disk, and Network usage.

Process	Status	12% CPU	43% Memory	0% Disk	0% Network
Applications (8)					
Internet Explorer		0%	17.6 MB	0 MB/s	0 Mbps
Microsoft Excel		0%	7.5 MB	0 MB/s	0 Mbps
Microsoft Lync 2010		0.6%	35.8 MB	0 MB/s	0.1 Mbps
Microsoft Outlook		0%	131.0 MB	0 MB/s	0 Mbps
Microsoft Sync Center		0%	1.5 MB	0 MB/s	0 Mbps
Microsoft Word		2.4%	130.3 MB	0 MB/s	0 Mbps
Notepad		0%	0.9 MB	0 MB/s	0 Mbps
Windows Task Manager		2.2%	11.6 MB	0 MB/s	0 Mbps
Background processes (7)					
Microsoft Office Software P...		0%	2.2 MB	0 MB/s	0 Mbps
Microsoft Windows Search ...		0%	21.0 MB	0 MB/s	0 Mbps
Print driver host for applica...		0%	2.4 MB	0 MB/s	0 Mbps
Spooler SubSystem App		0%	3.2 MB	0 MB/s	0 Mbps

使用堆实现优先队列



使用堆实现优先队列

在1,000,000个元素中选出前100名?

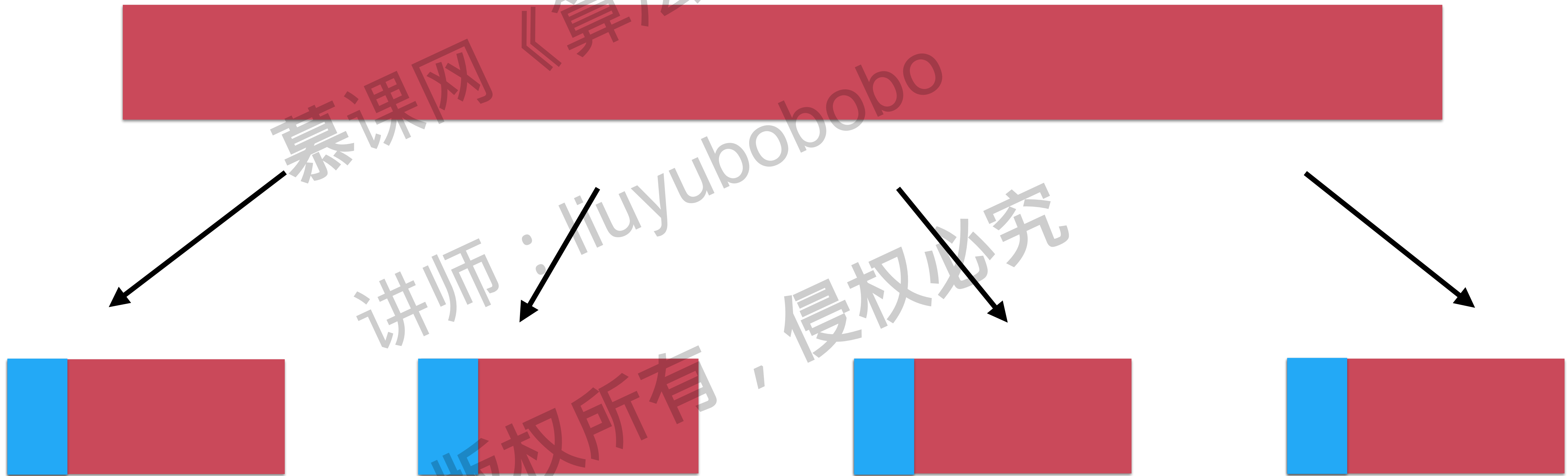
在N个元素中选出前M个元素

使用优先队列? $N \log M$

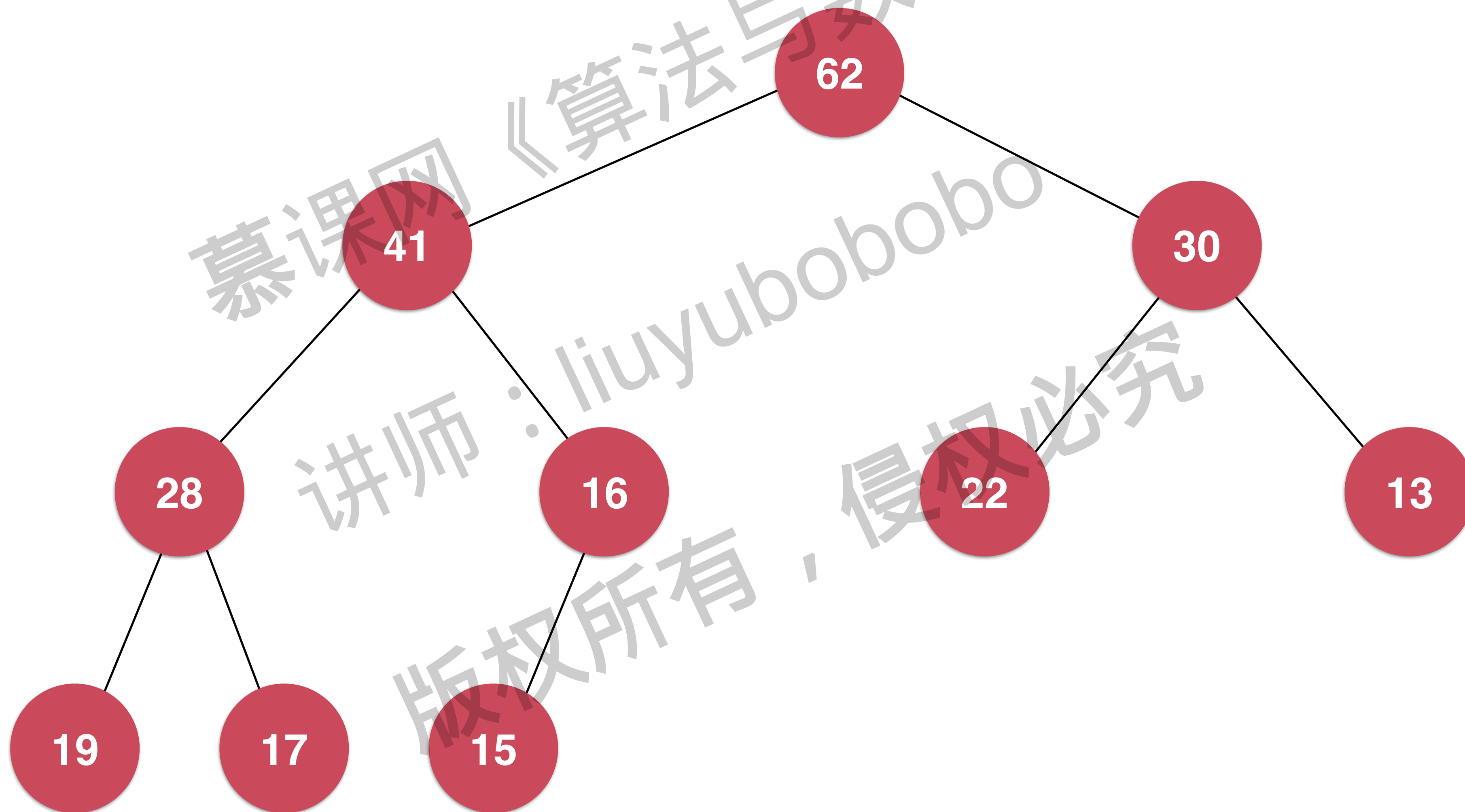
多路归并排序



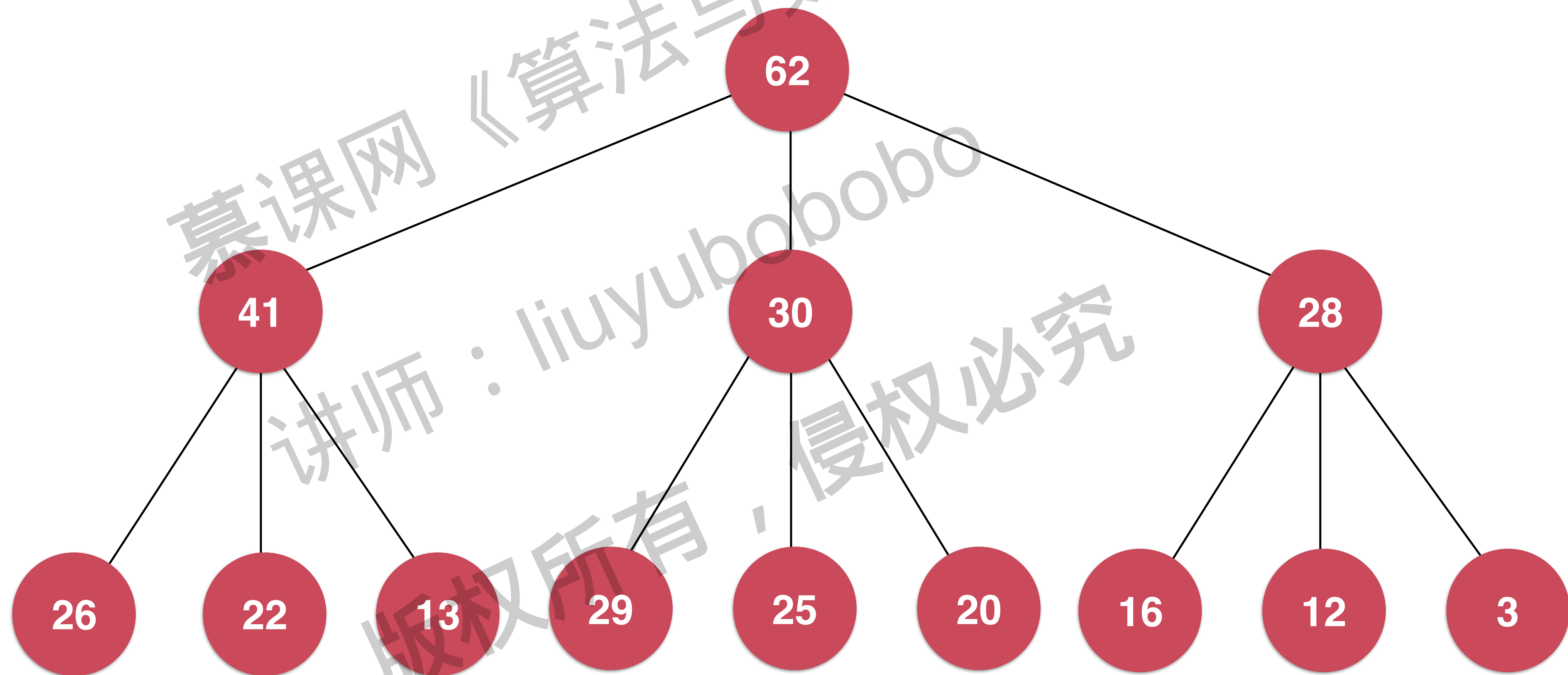
多路归并排序



二叉堆 Binary Heap



d 叉堆 d-ary heap



最大堆 最大索引堆

最小堆 最小索引堆

堆的实现细节优化

ShiftUp 和 ShiftDown 中使用赋值操作替换swap操作

表示堆的数组从0开始索引

没有capacity的限制，动态的调整堆中数组的大小

慕课网《算法与数据结构》

最大最小队列

讲师：liuyubobobo

版权所有，侵权必究

慕课网《算法与数据结构》

二项堆

讲师：liuyubobobo

斐波那契堆

版权所有，侵权必究

其他

欢迎大家关注我的个人公众号：是不是很酷



慕课网《算法与数据结构》

算法与数据结构

讲师：liuyubobobo

版权所有 侵权必究

liuyubobobo