

玩儿转数据结构

liuyubobobo

堆和优先队列

优先队列基础

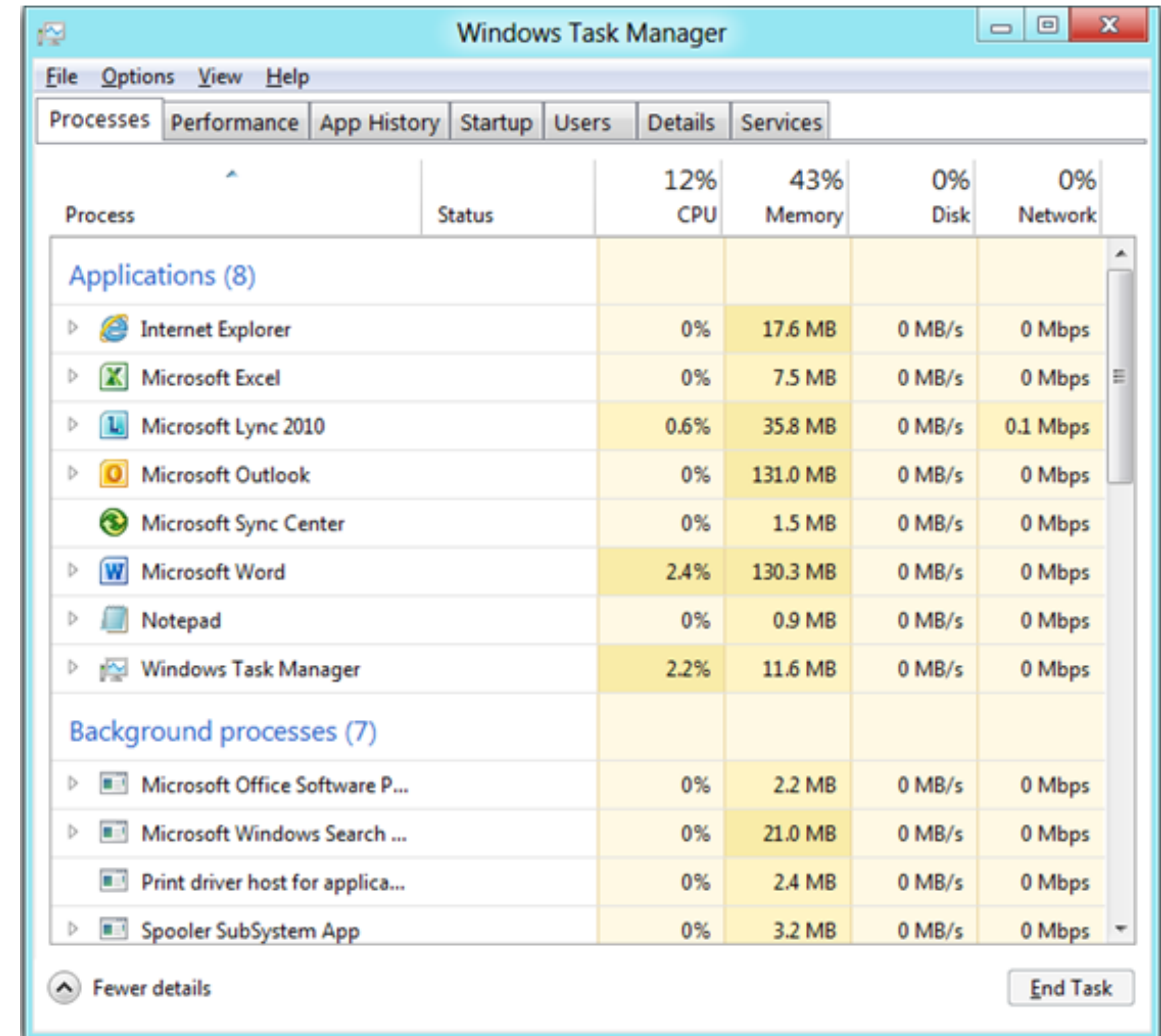
什么是优先队列？

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

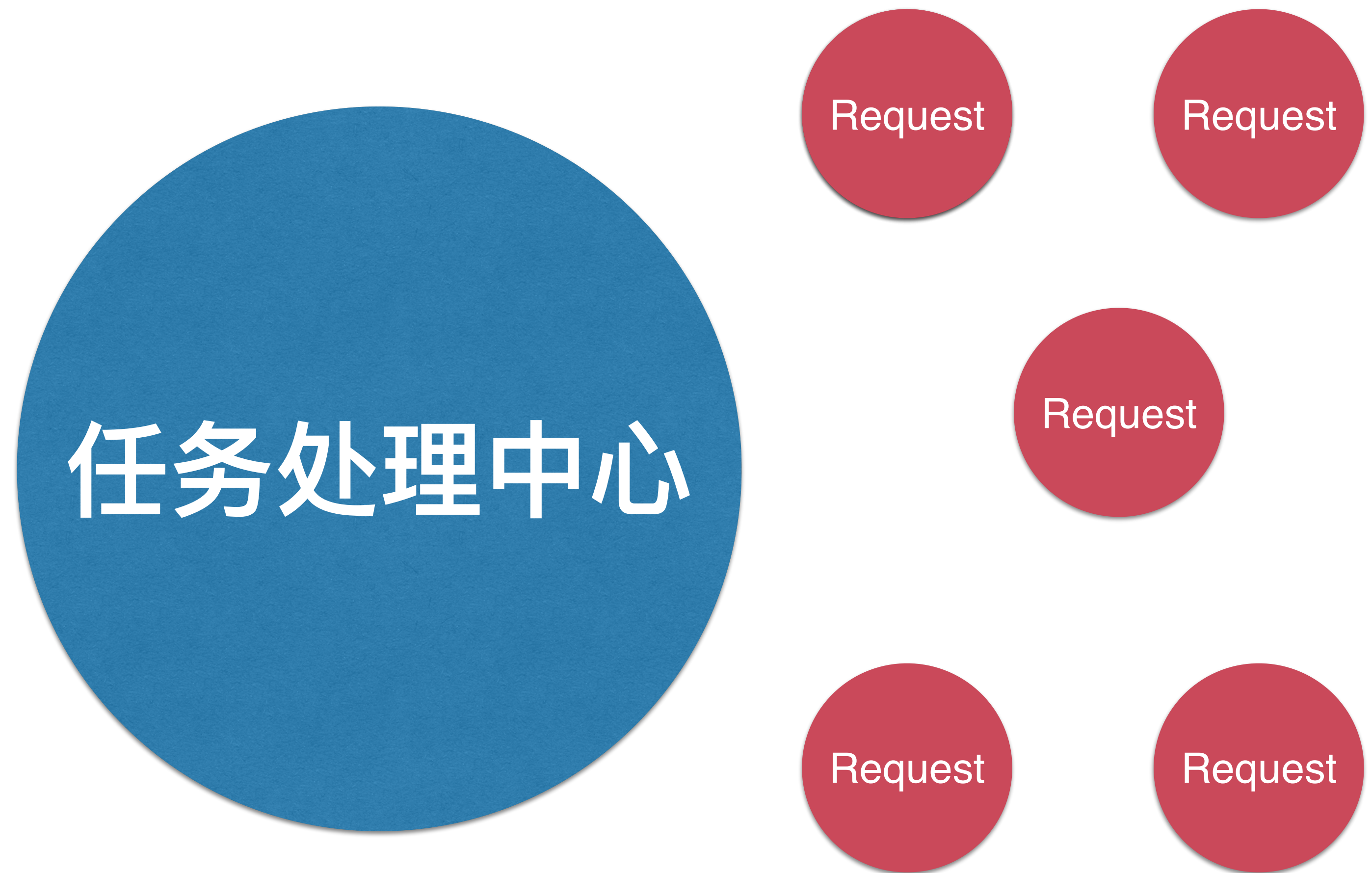


The screenshot shows the Windows Task Manager window with the 'Processes' tab selected. It displays a list of running applications and background processes, along with their CPU, Memory, Disk, and Network usage. The 'Applications (8)' section lists Internet Explorer, Microsoft Excel, Microsoft Lync 2010, Microsoft Outlook, Microsoft Sync Center, Microsoft Word, Notepad, and Windows Task Manager. The 'Background processes (7)' section lists Microsoft Office Software P..., Microsoft Windows Search ..., Print driver host for applica..., and Spooler SubSystem App. The table includes columns for Process, Status, CPU, Memory, Disk, and Network usage.

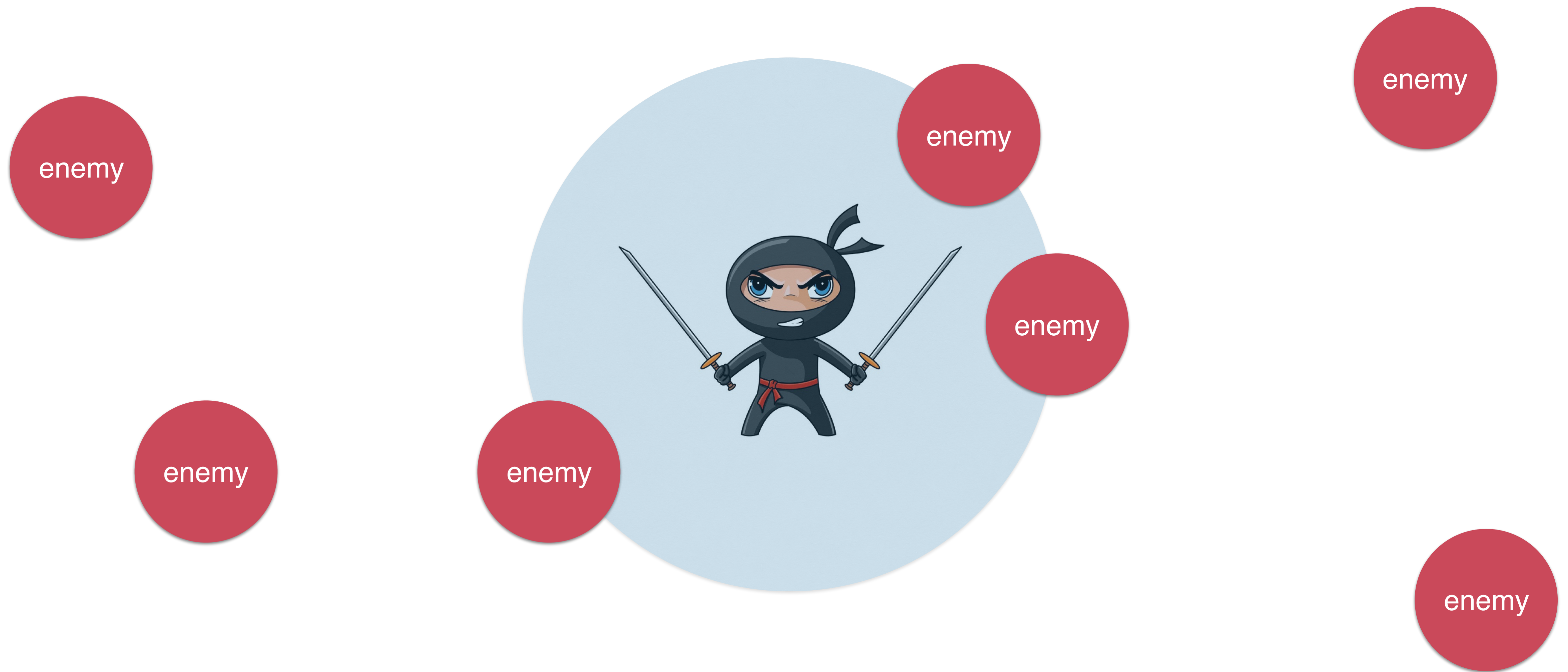
Process	Status	12% CPU	43% Memory	0% Disk	0% Network
Applications (8)					
Internet Explorer		0%	17.6 MB	0 MB/s	0 Mbps
Microsoft Excel		0%	7.5 MB	0 MB/s	0 Mbps
Microsoft Lync 2010		0.6%	35.8 MB	0 MB/s	0.1 Mbps
Microsoft Outlook		0%	131.0 MB	0 MB/s	0 Mbps
Microsoft Sync Center		0%	1.5 MB	0 MB/s	0 Mbps
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



为什么使用优先队列?



优先队列

Interface Queue<E>  PriorityQueue<E>
implement

- void enqueue(E)
- E dequeue()
- E getFront()
- int getSize()
- boolean isEmpty()

可以使用不同的底层实现

优先队列

入队

出队 (拿出最大元素)

普通线性结构

$O(1)$

$O(n)$

顺序线性结构

$O(n)$

$O(1)$

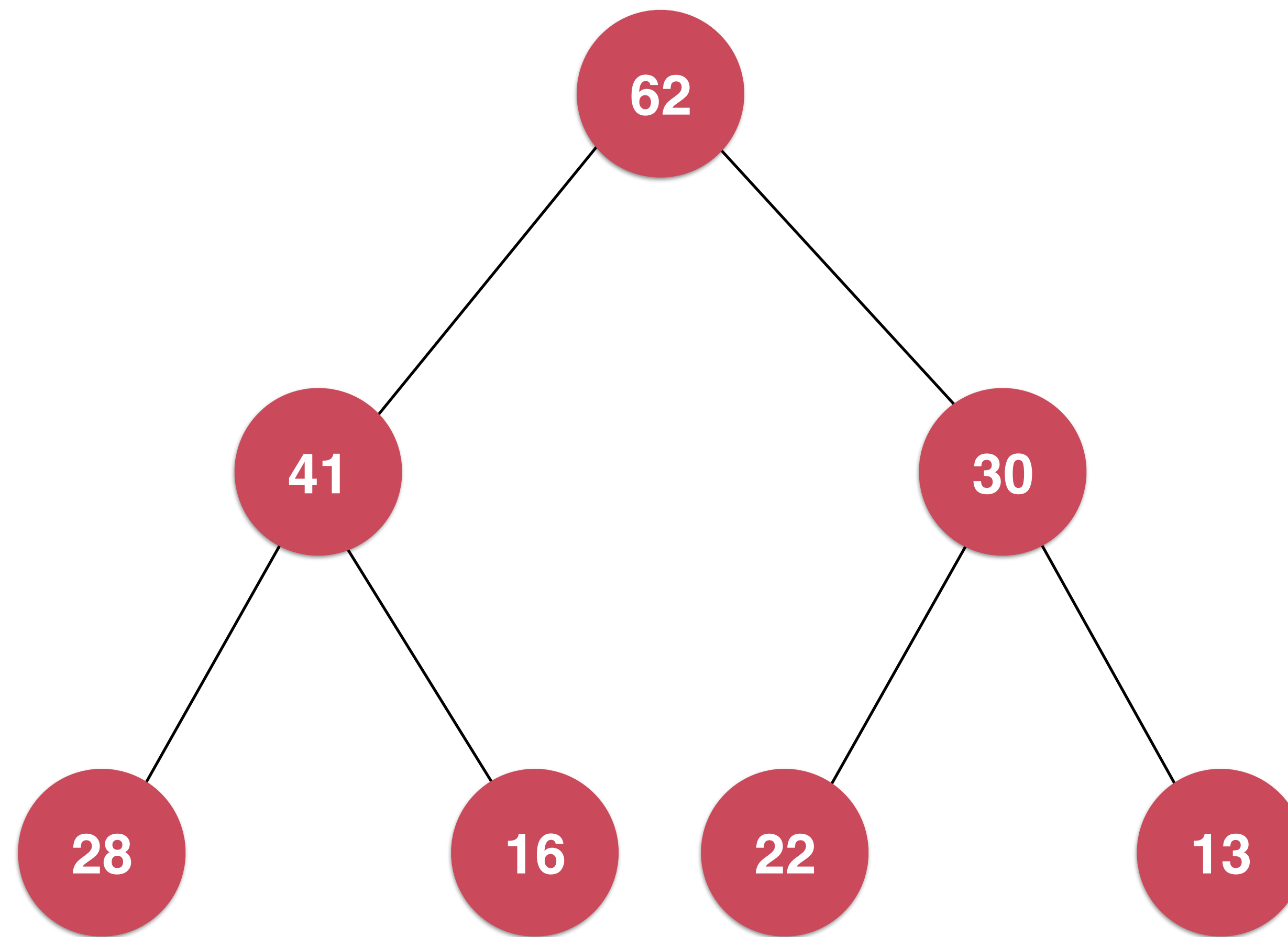
堆

$O(\log n)$

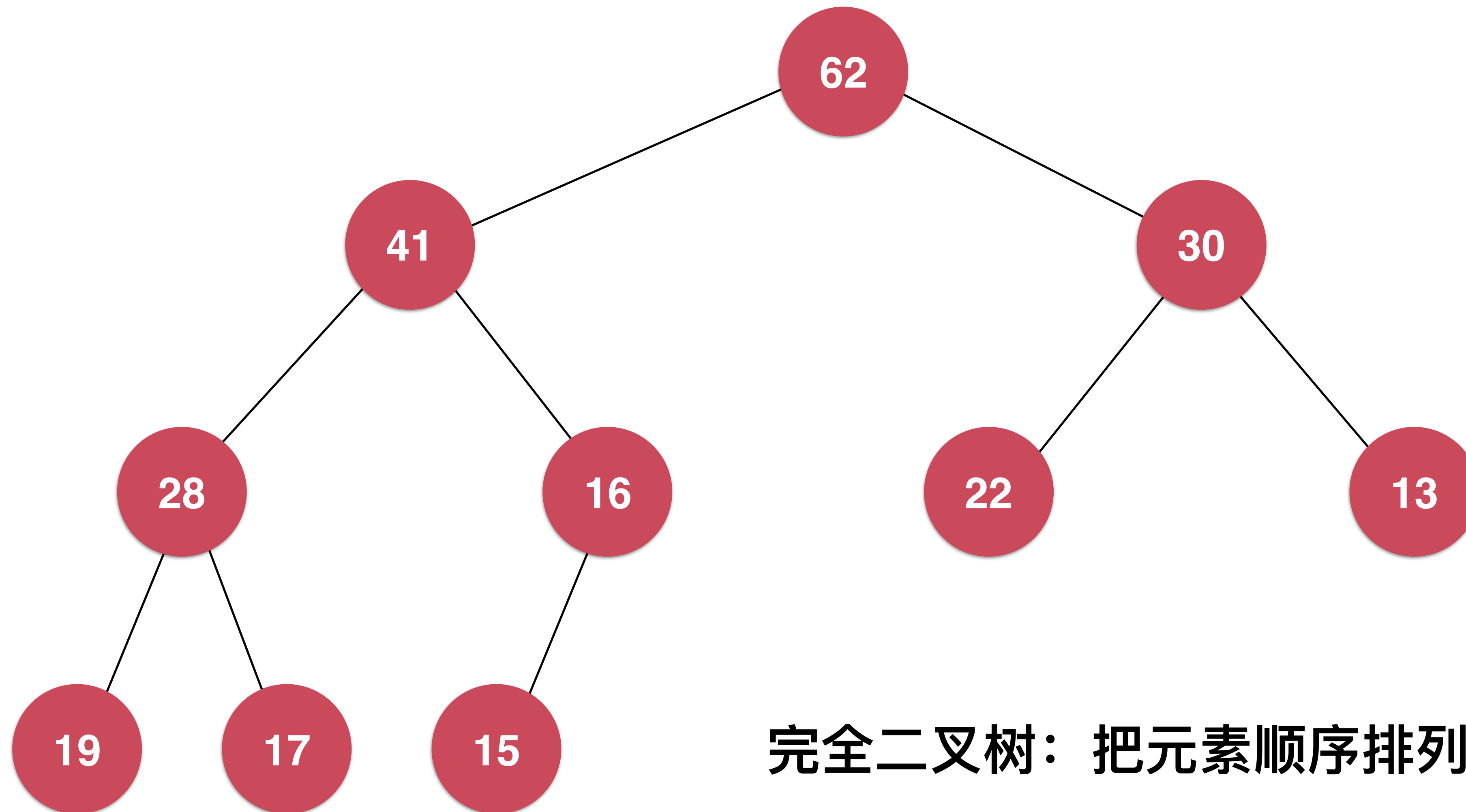
$O(\log n)$

堆的基本结构

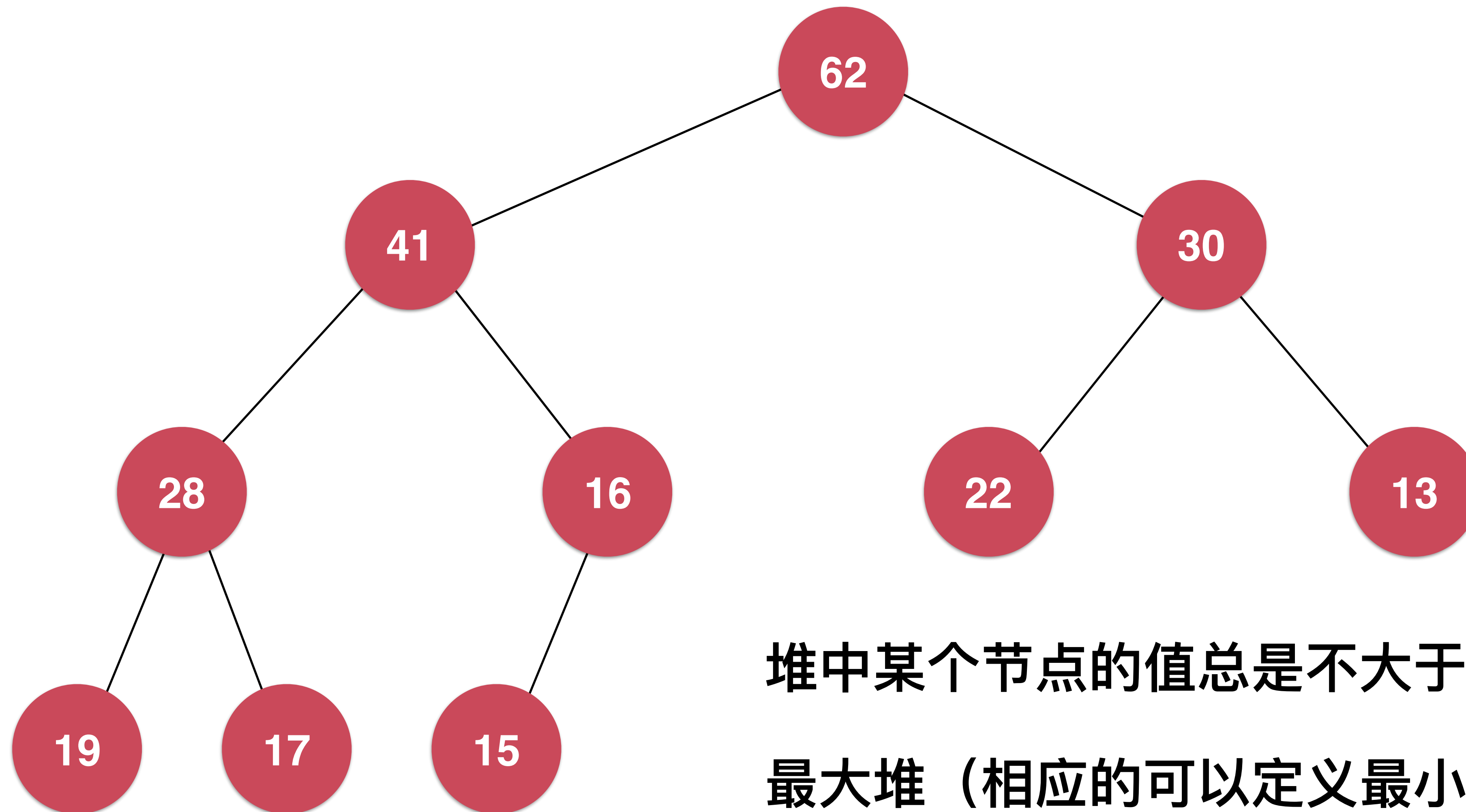
二叉堆 Binary Heap



二叉堆是一棵完全二叉树



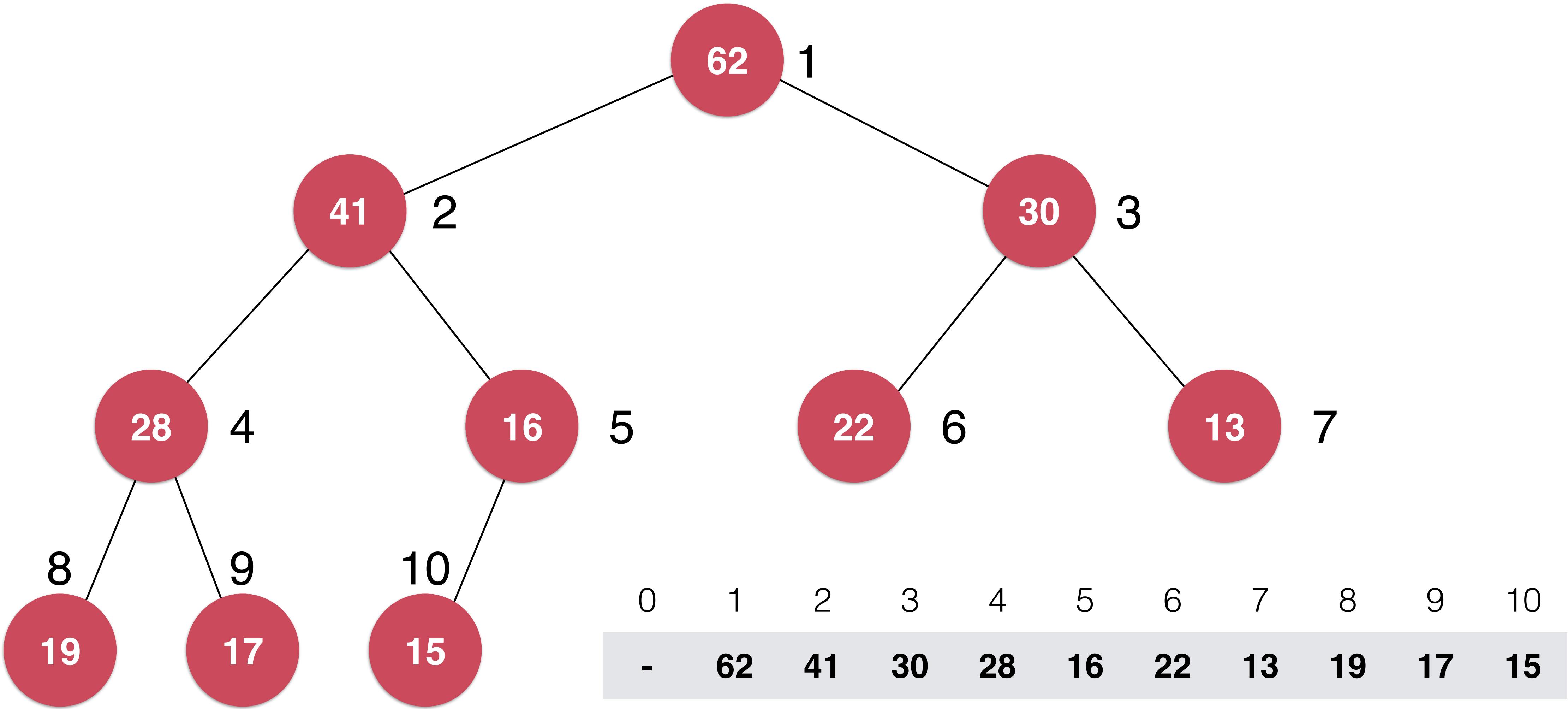
二叉堆的性质



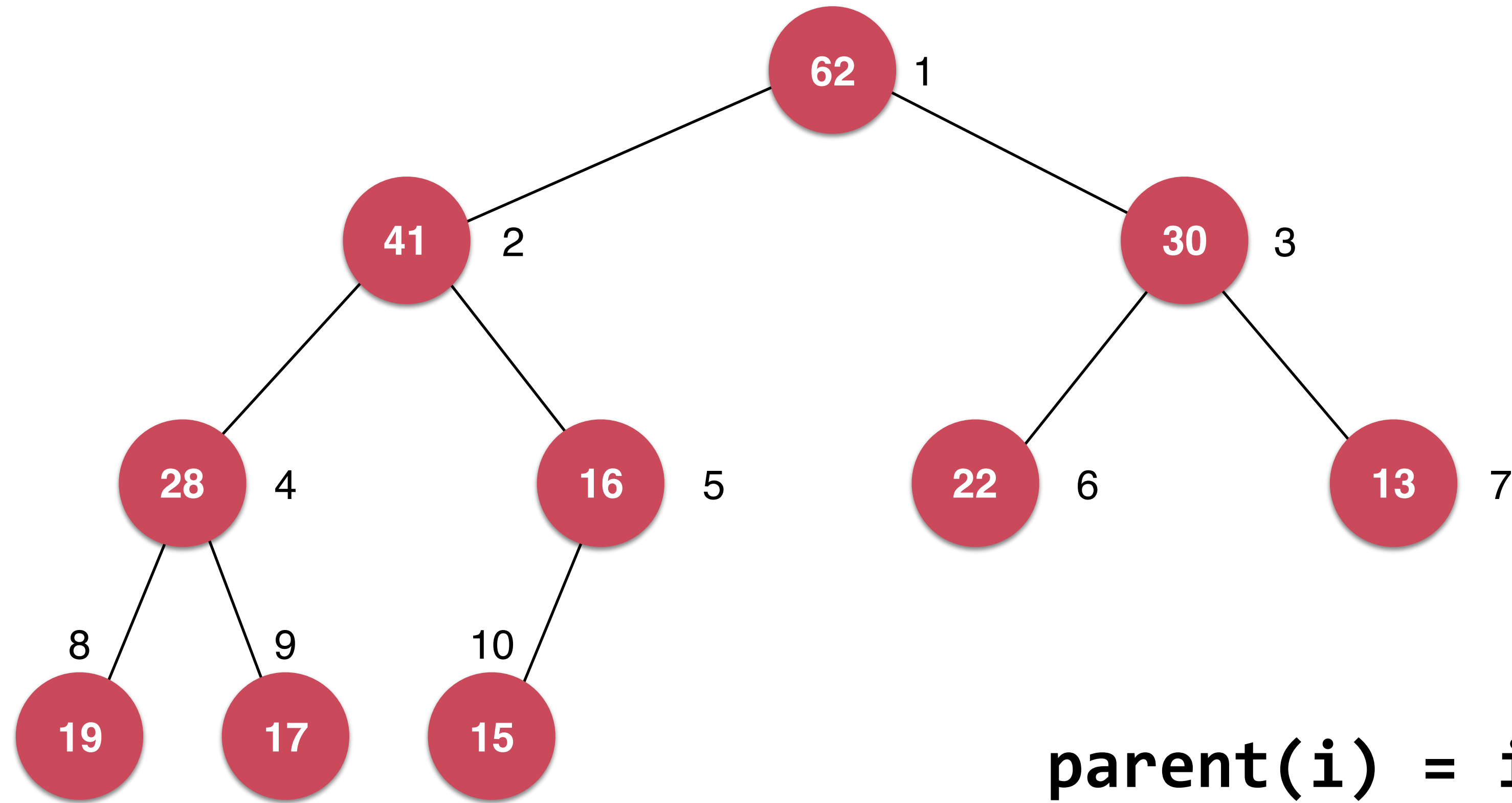
堆中某个节点的值总是不大于其父节点的值

最大堆（相应的可以定义最小堆）

最大堆



用数组存储二叉堆



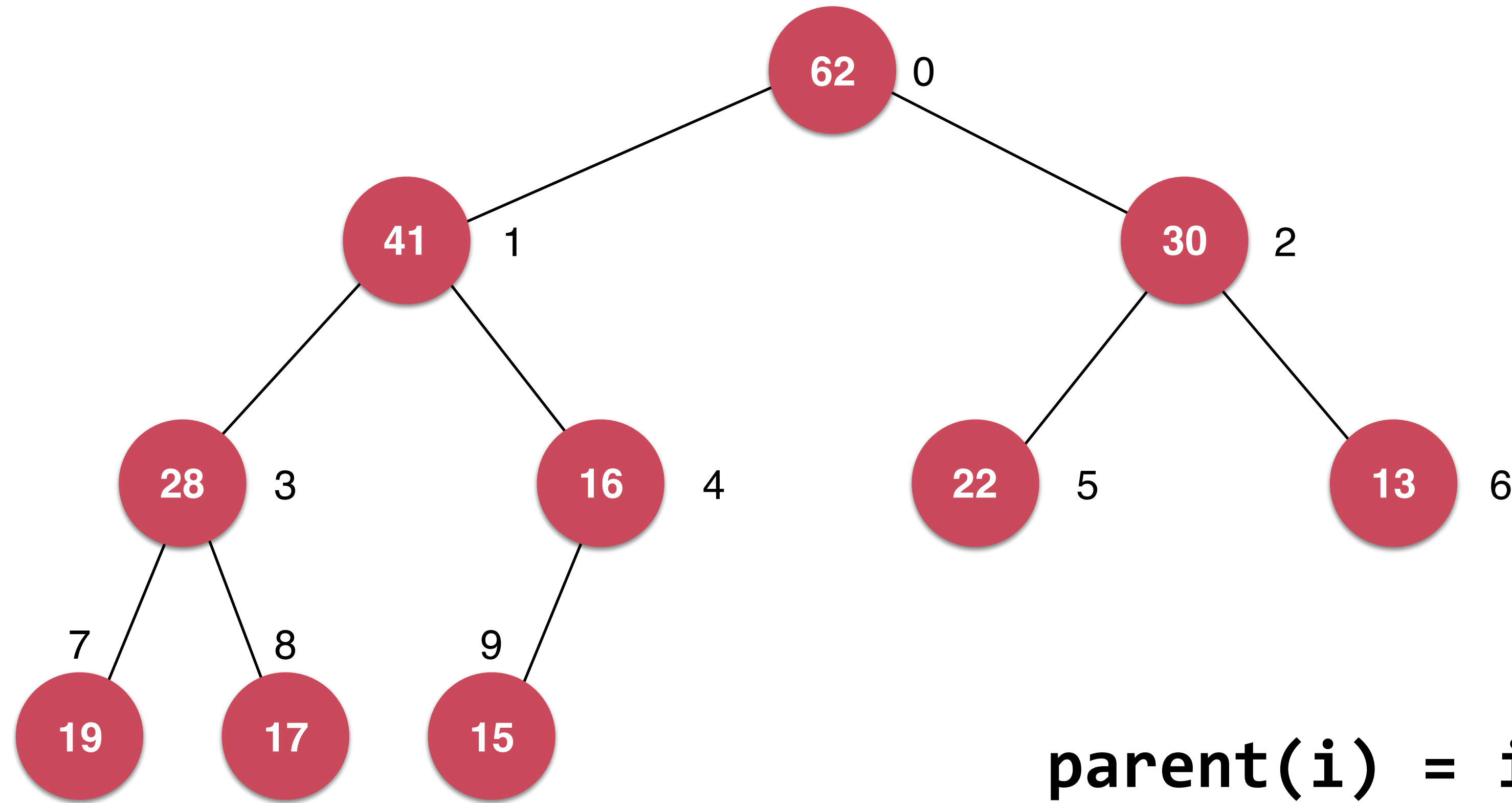
$$\text{parent}(i) = i/2$$

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

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

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

用数组存储二叉堆



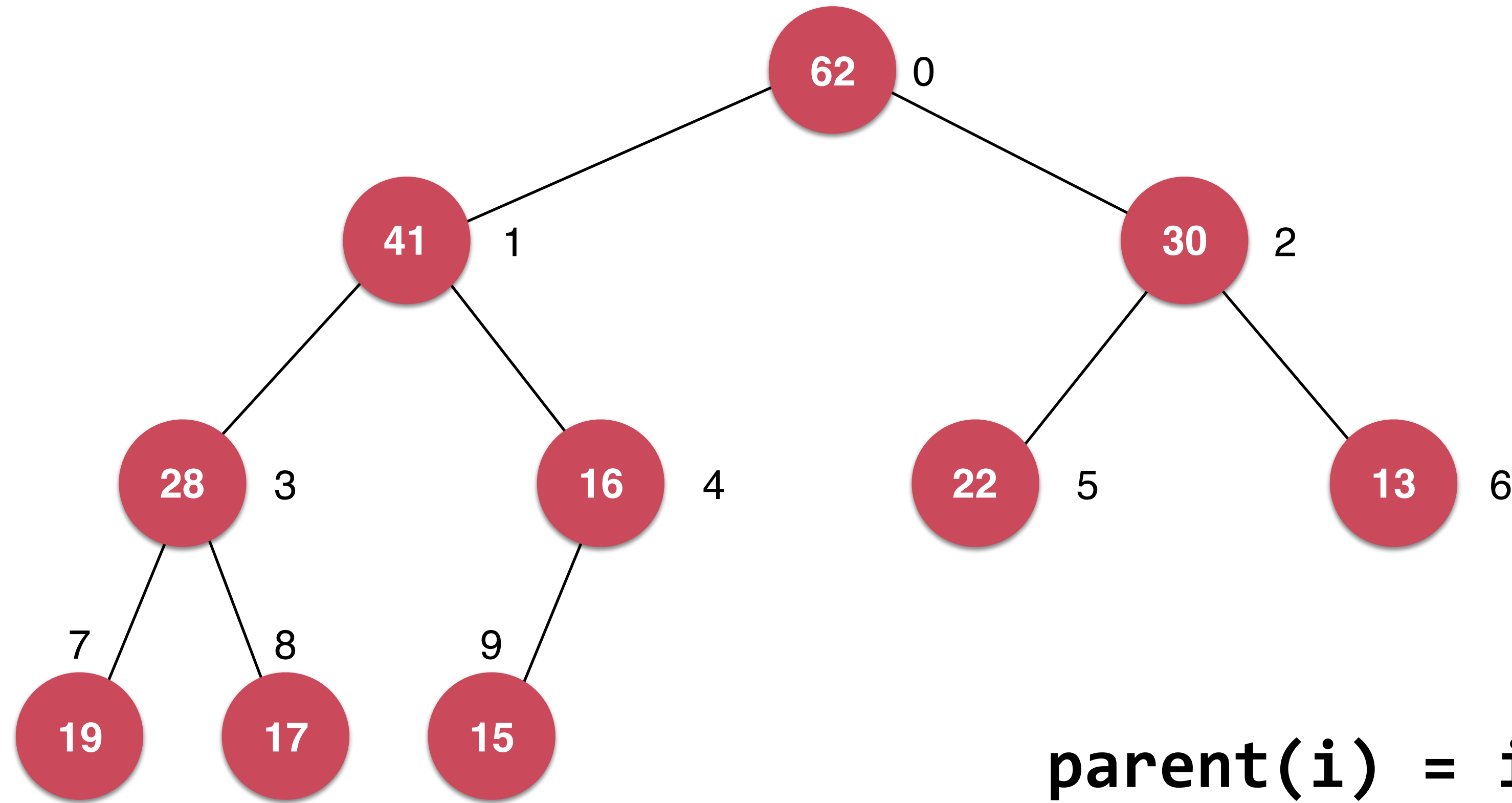
$$\text{parent}(i) = i/2$$

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

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

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

用数组存储二叉堆



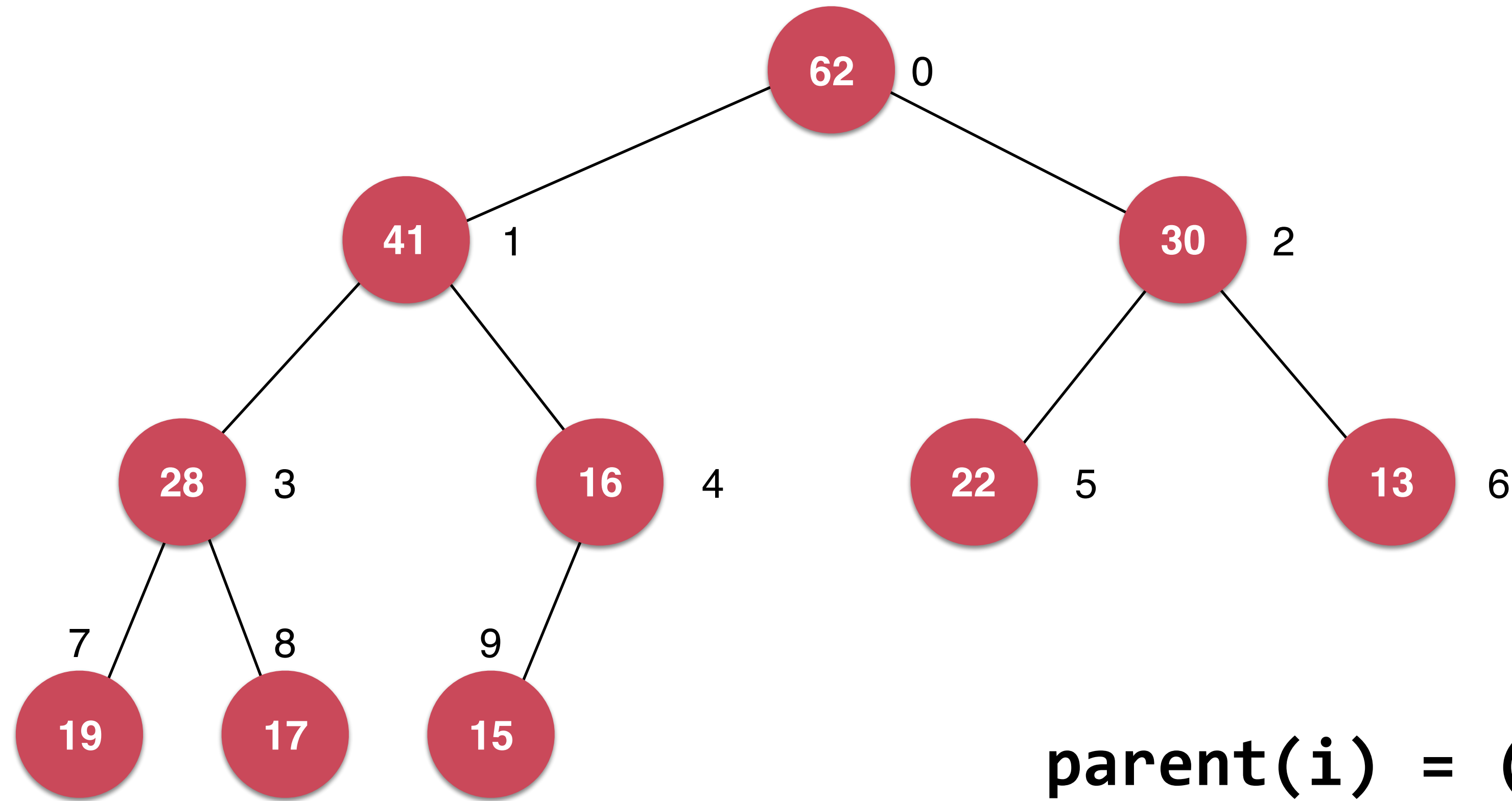
$$\text{parent}(i) = i/2$$

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

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

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

用数组存储二叉堆



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

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

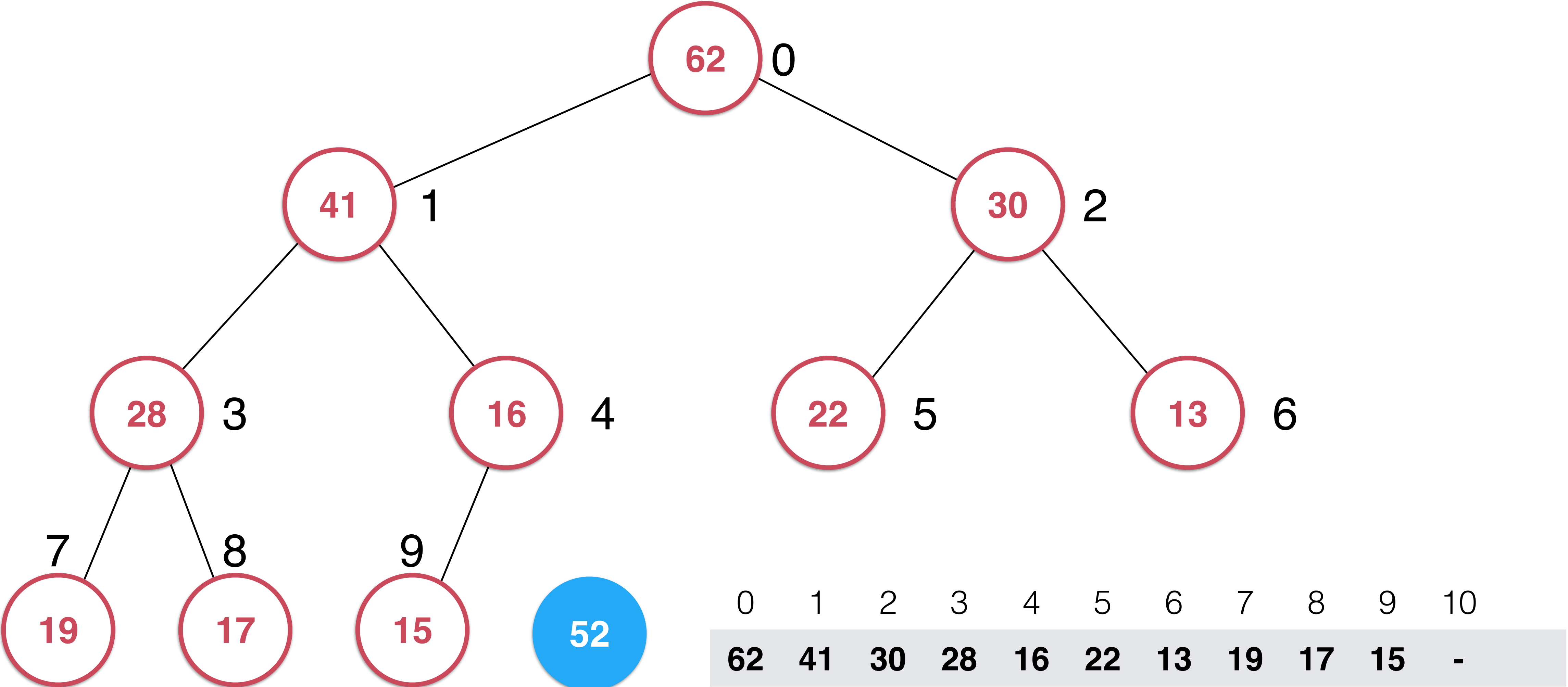
$$\text{right child } (i) = 2 * i + 2$$

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

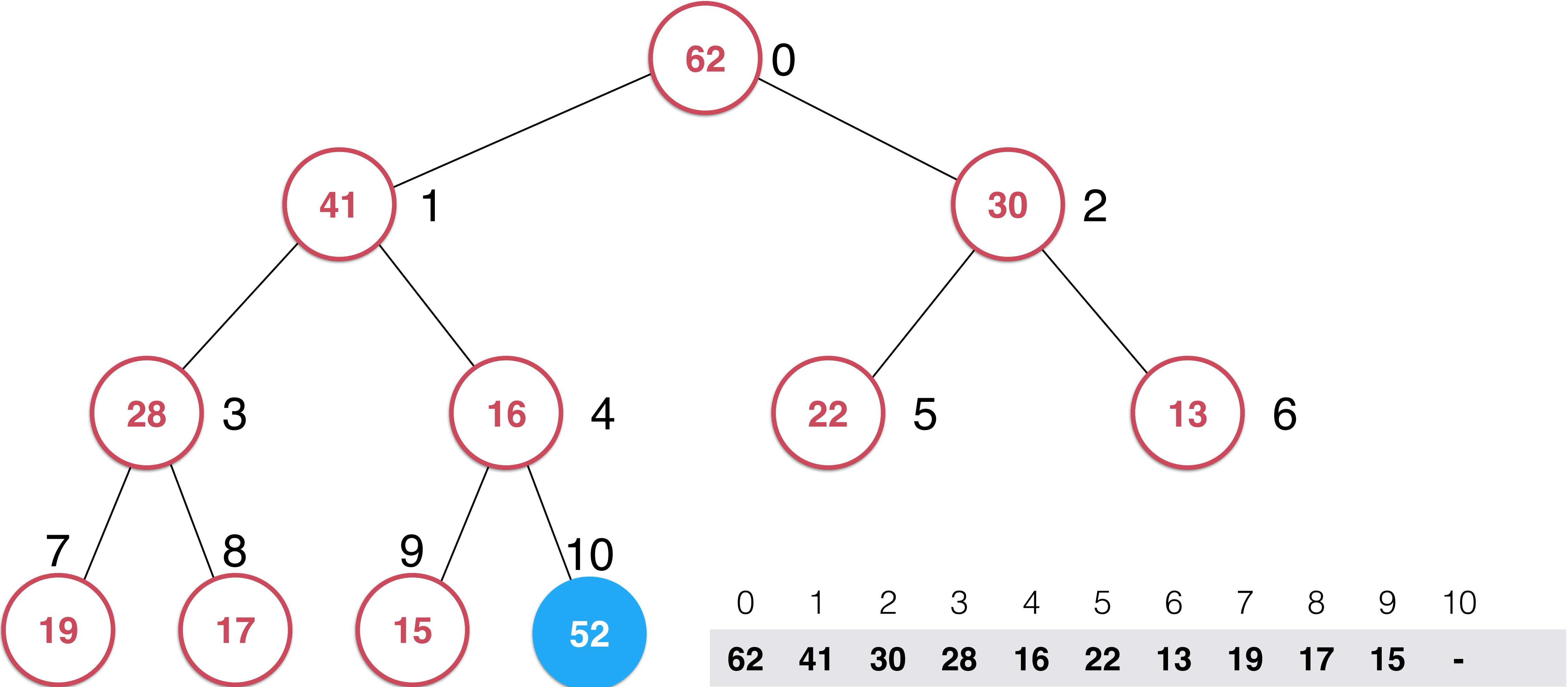
实践：堆的基本框架

向堆中添加元素和Sift Up

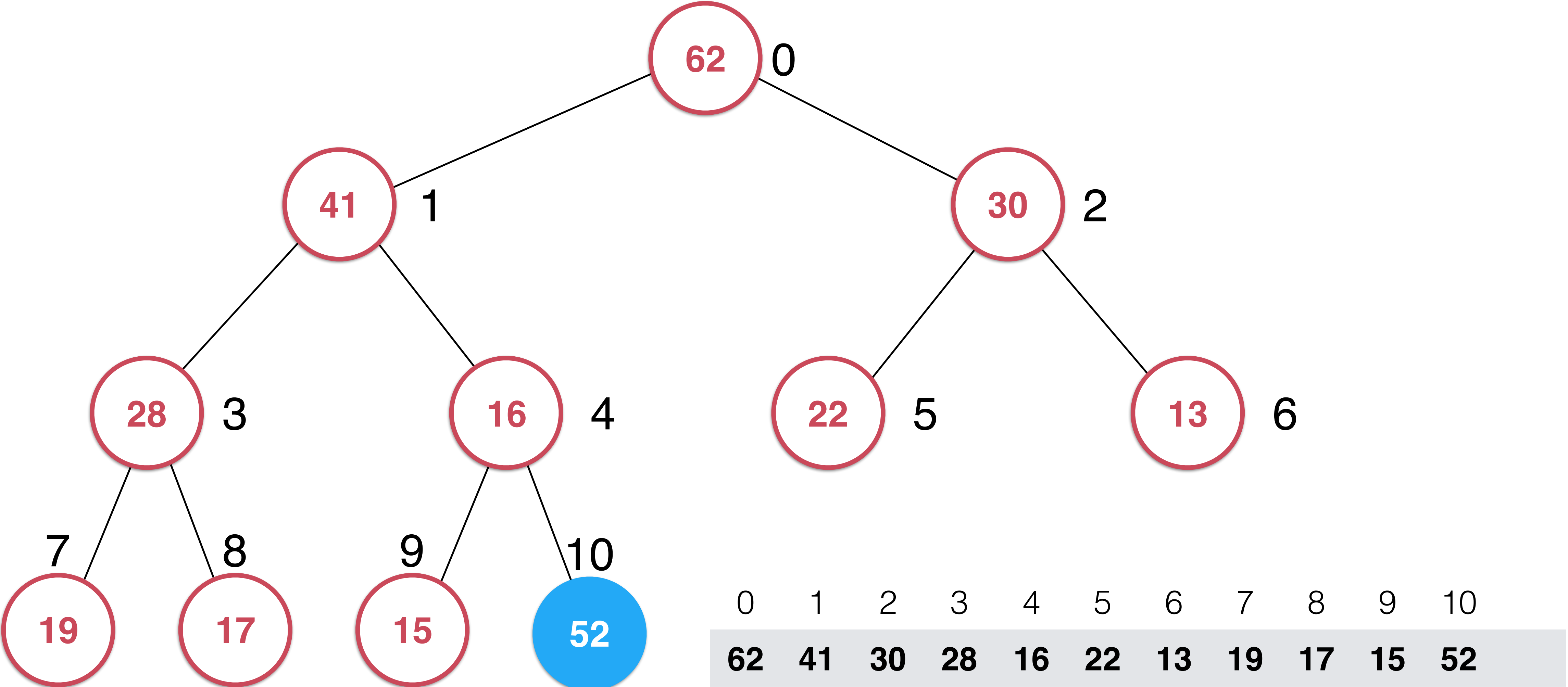
Sift Up



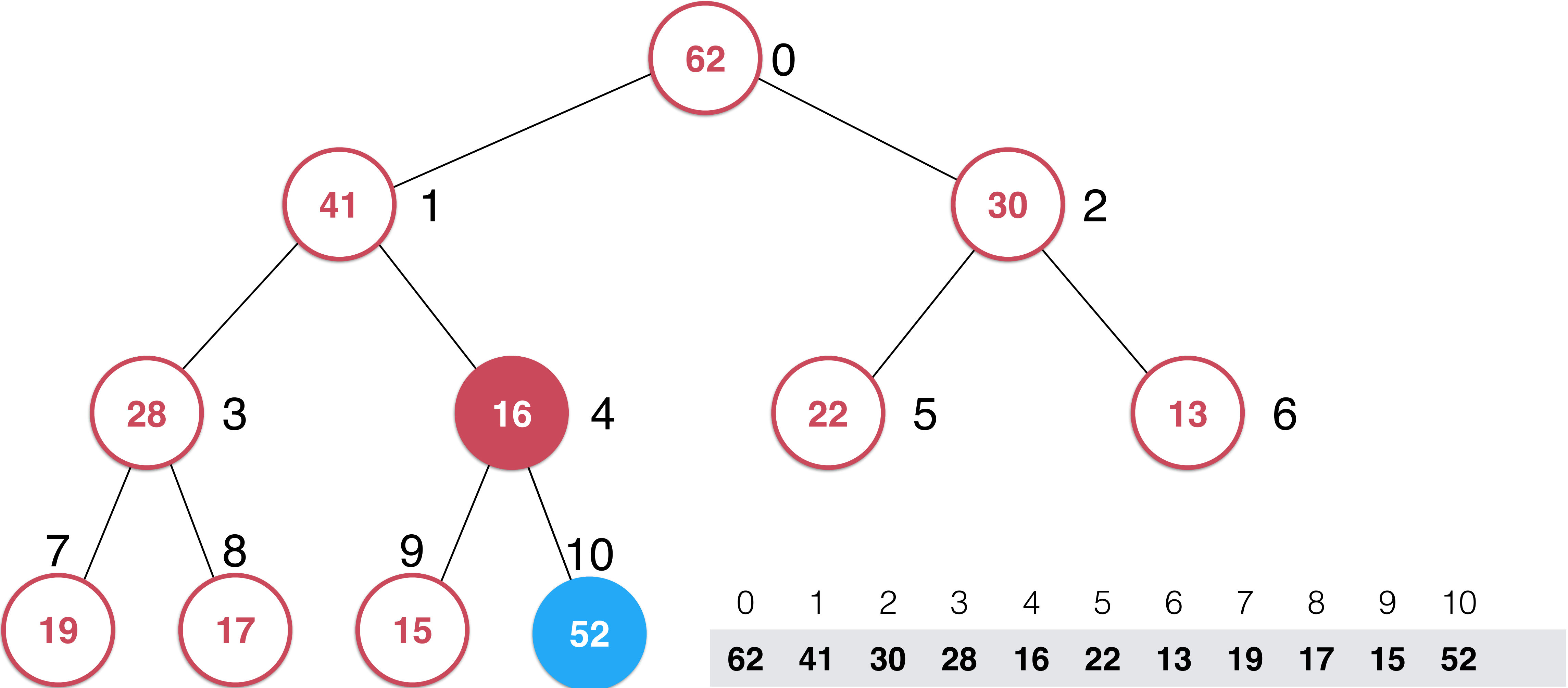
Sift Up



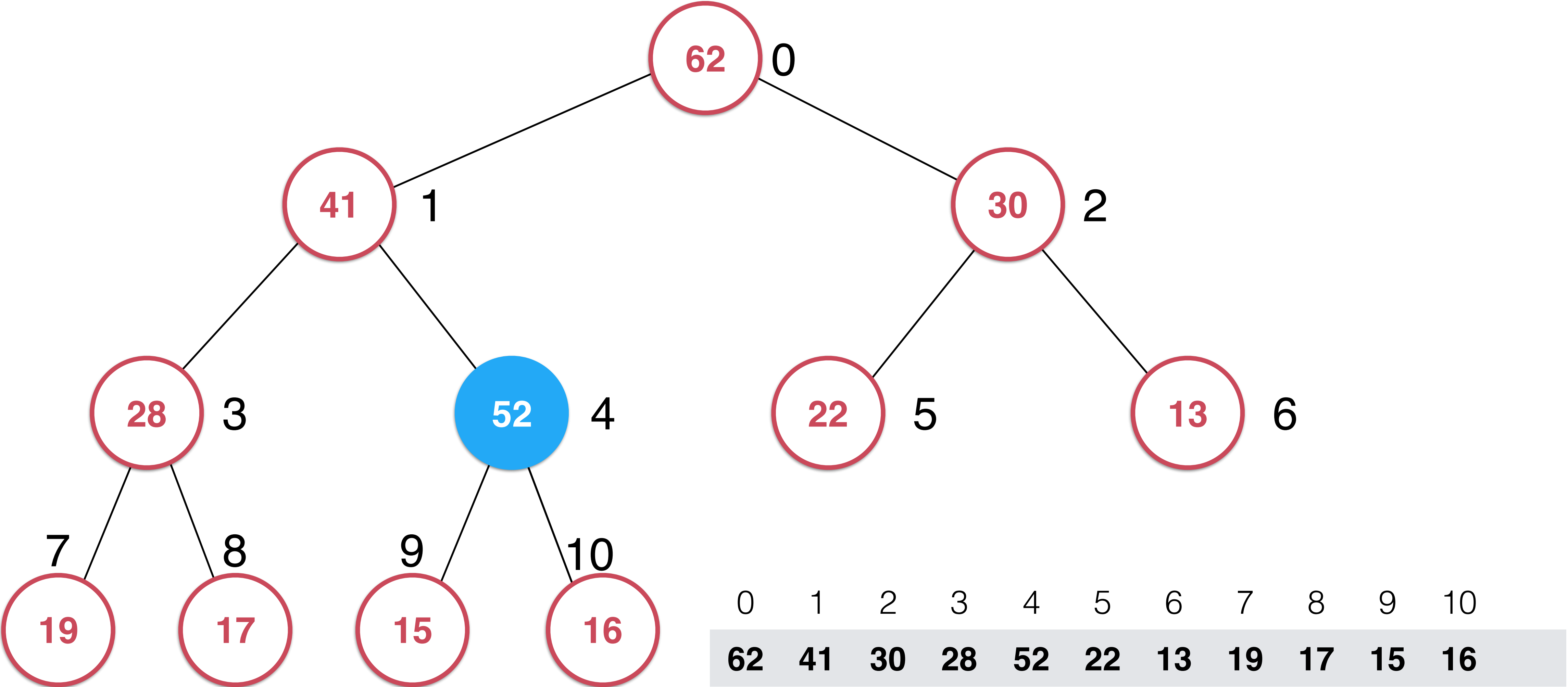
Sift Up



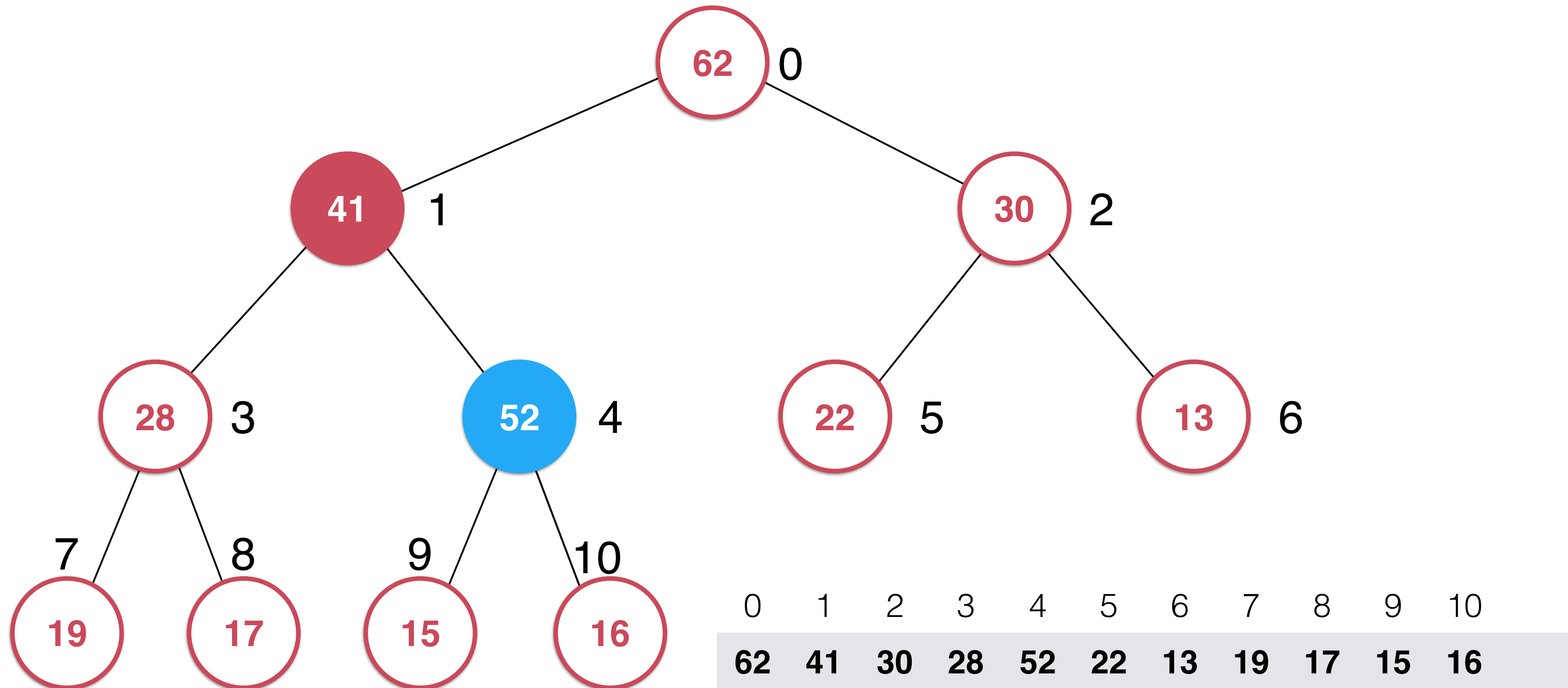
Sift Up



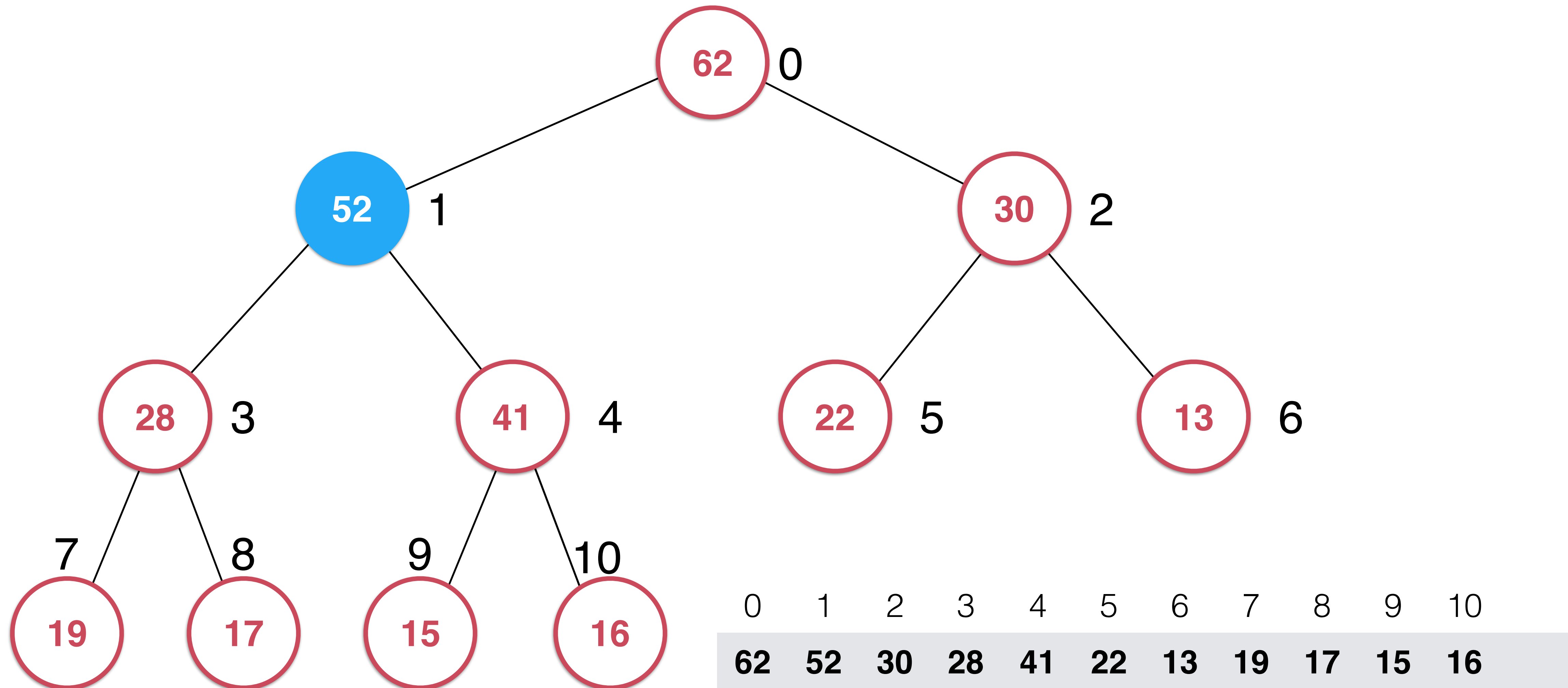
Sift Up



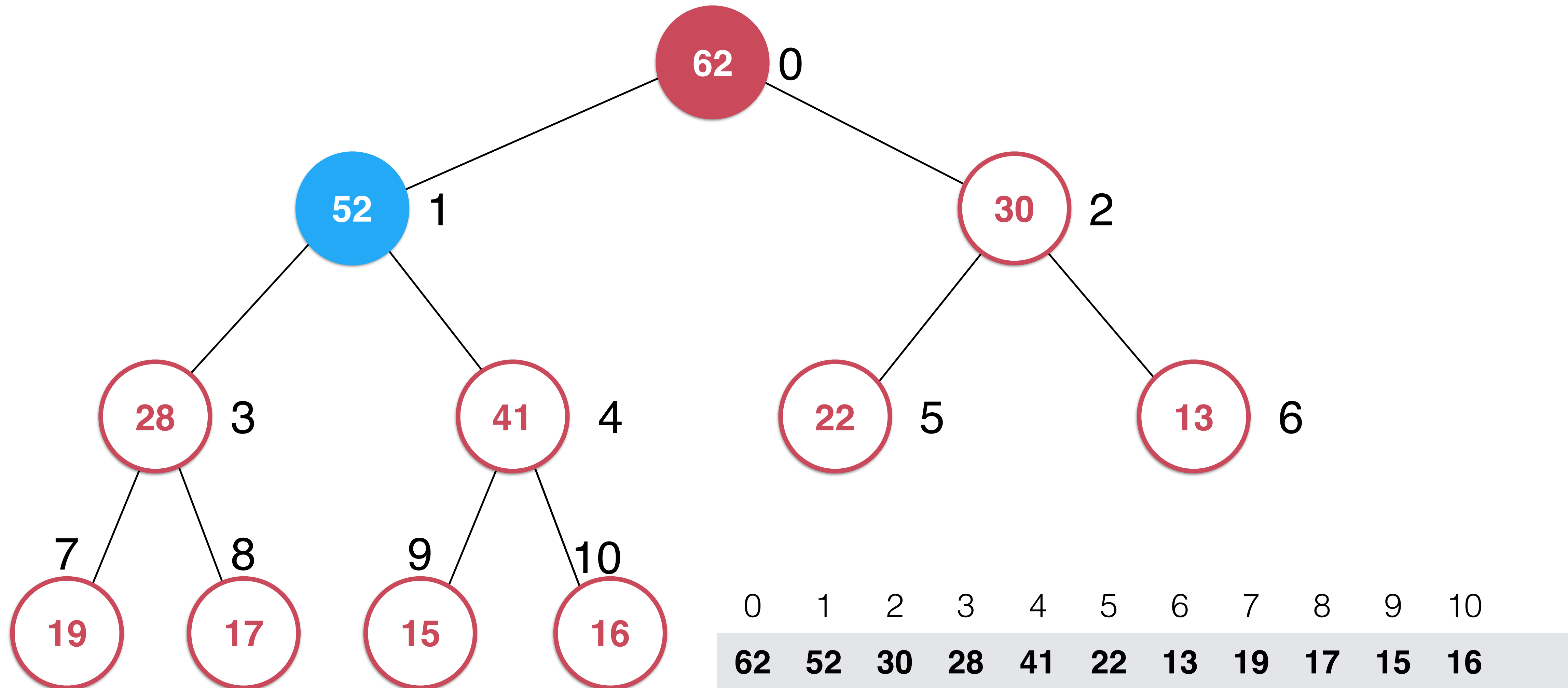
Sift Up



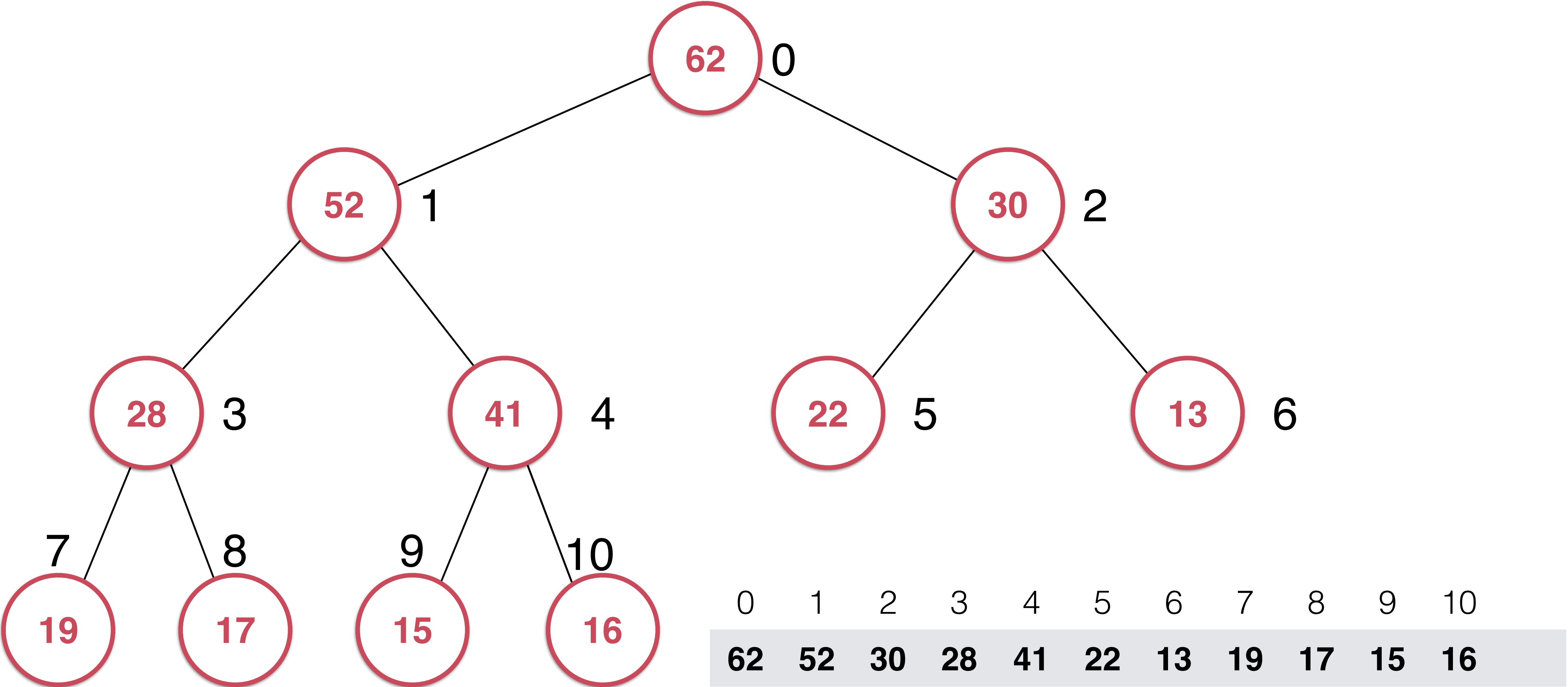
Sift Up



Sift Up



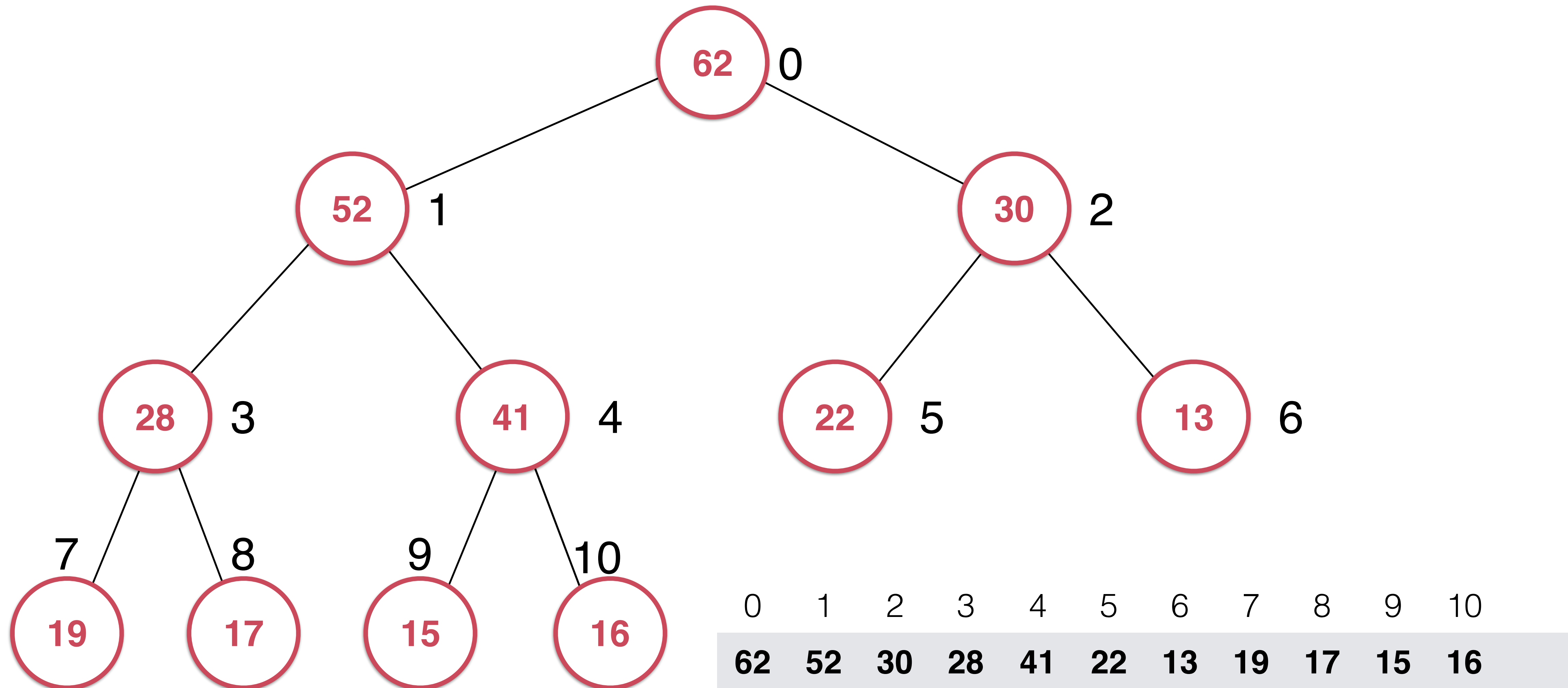
Sift Up



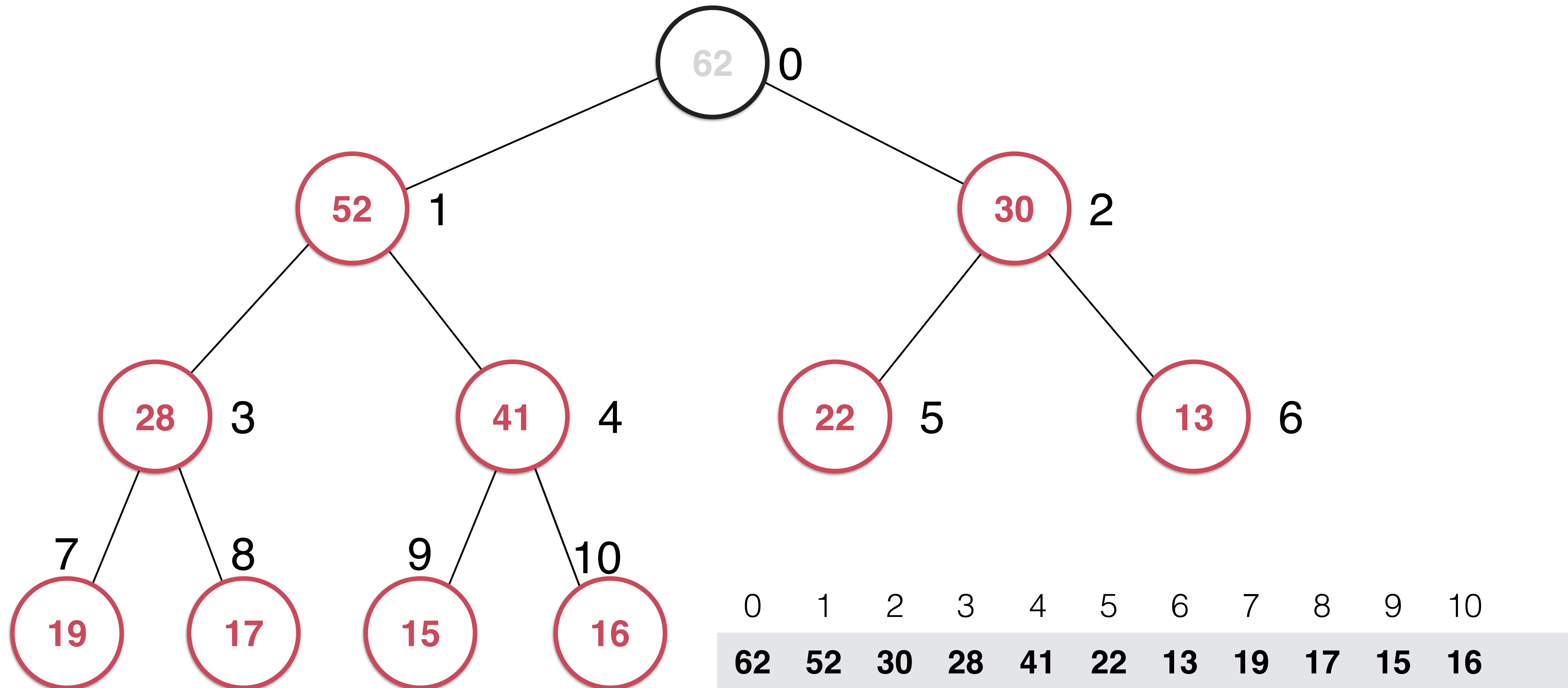
实践： Sift Up 和 add

取出堆中的最大元素和Sift Down

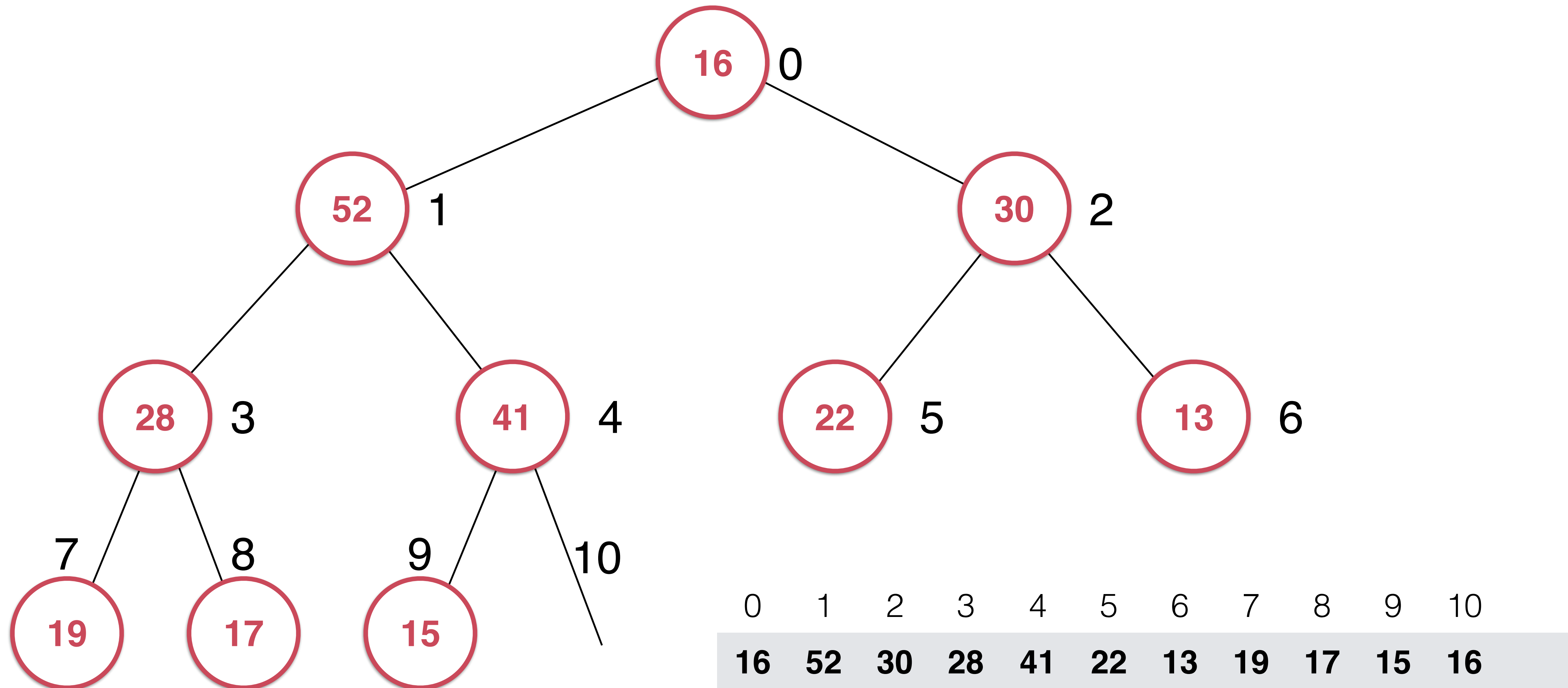
Sift Down



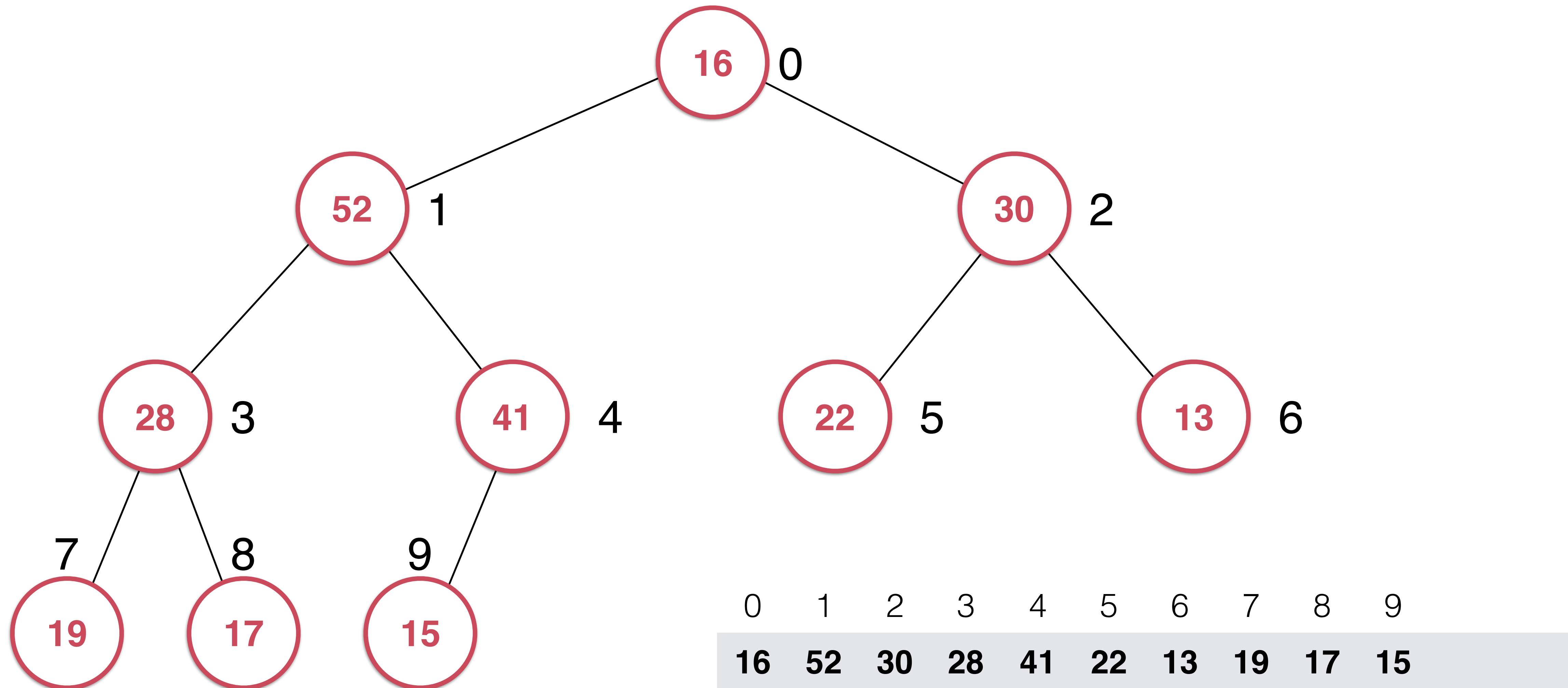
Sift Down



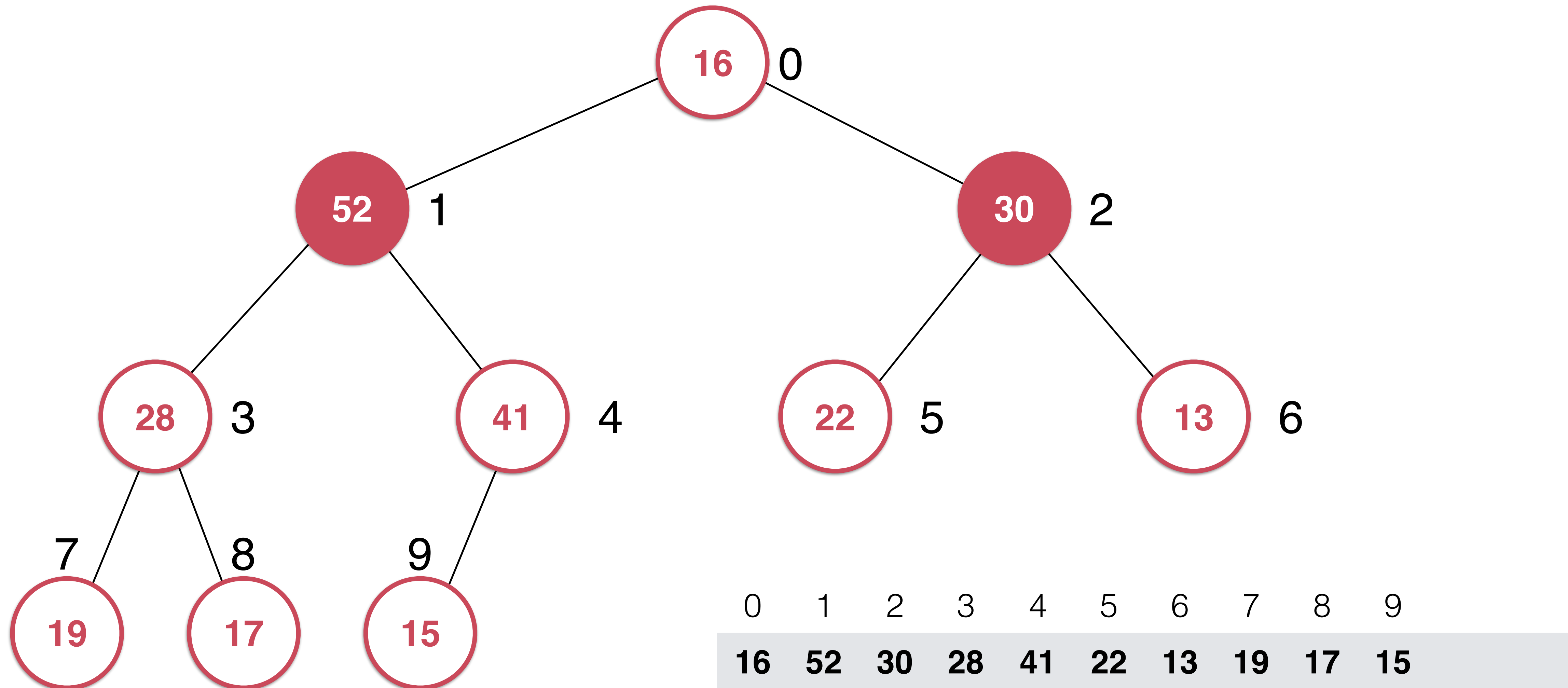
Sift Down



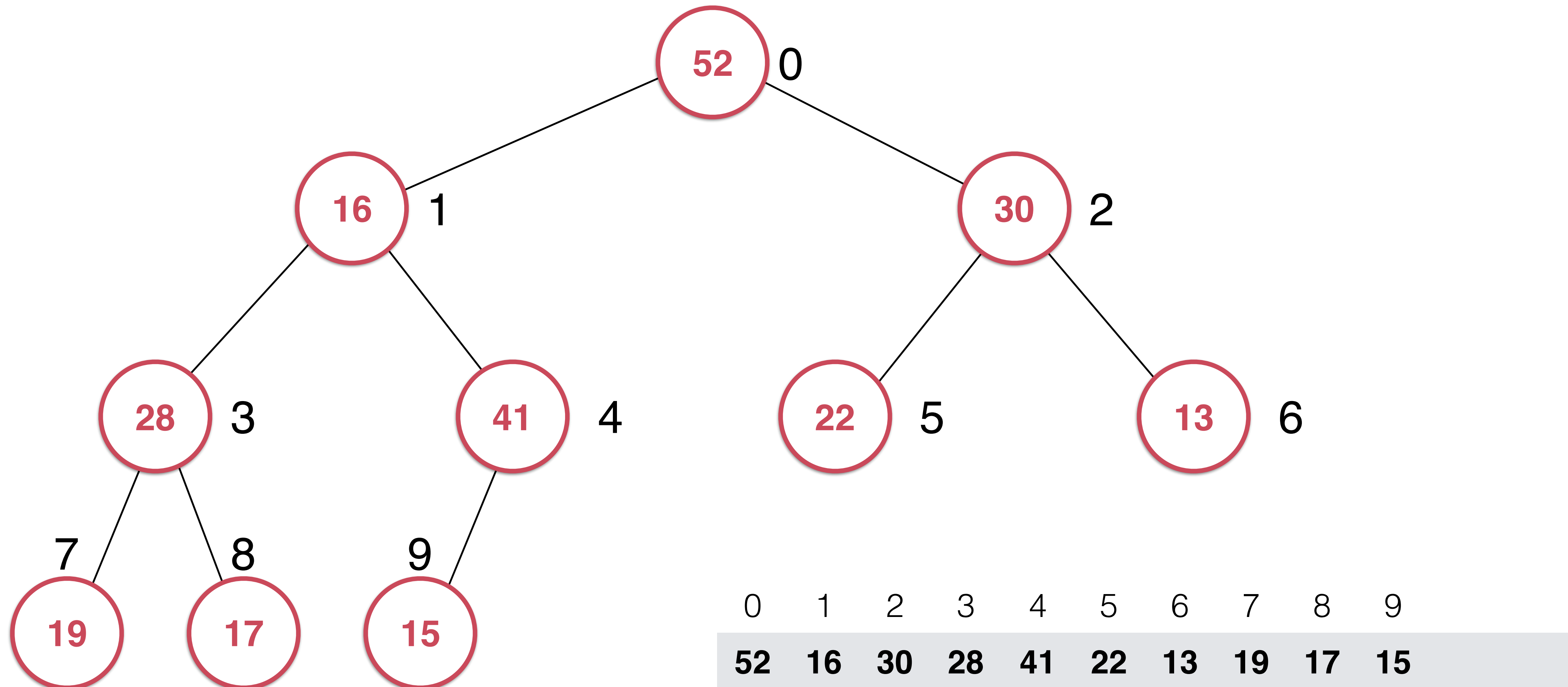
Sift Down



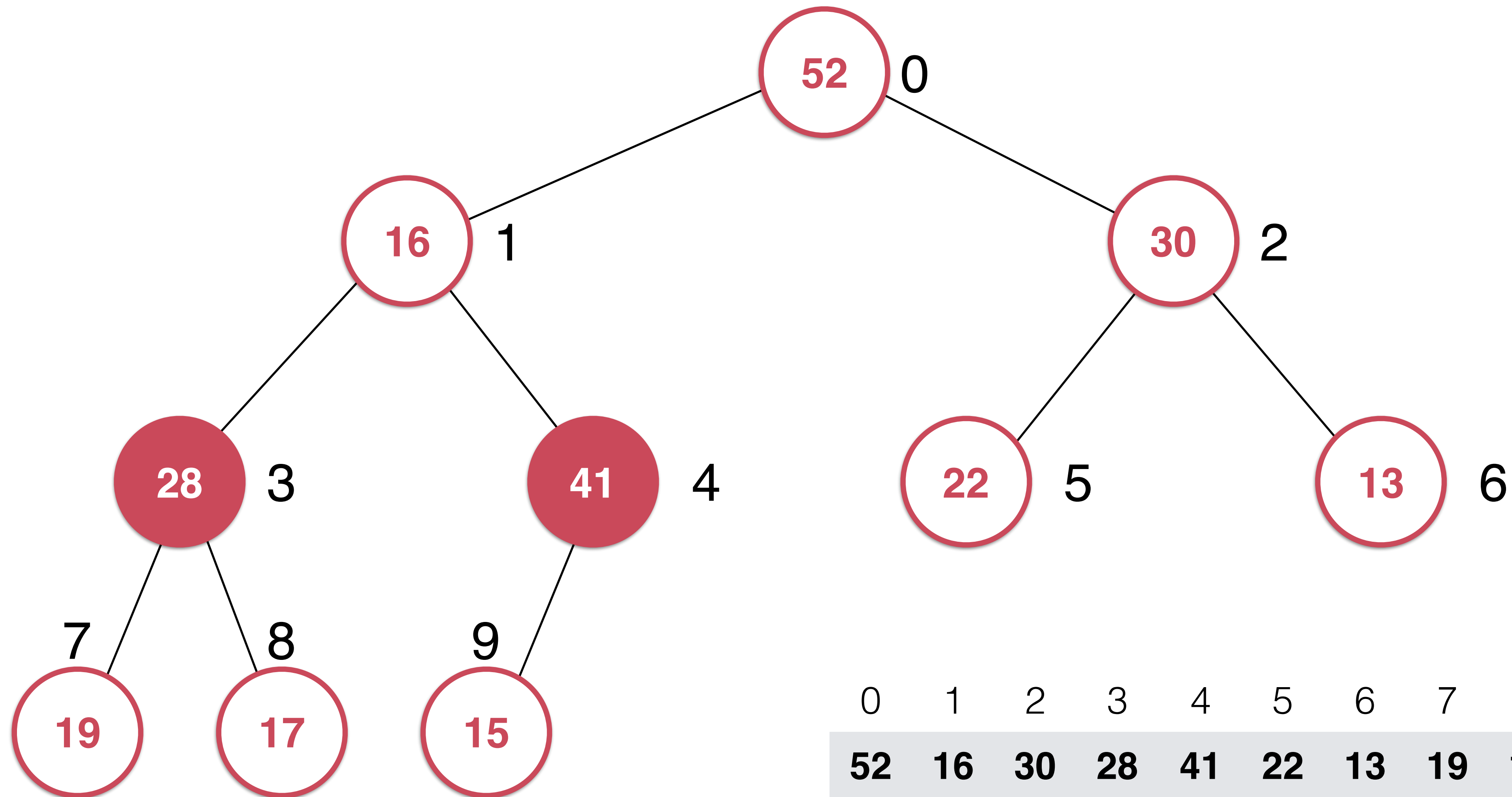
Sift Down



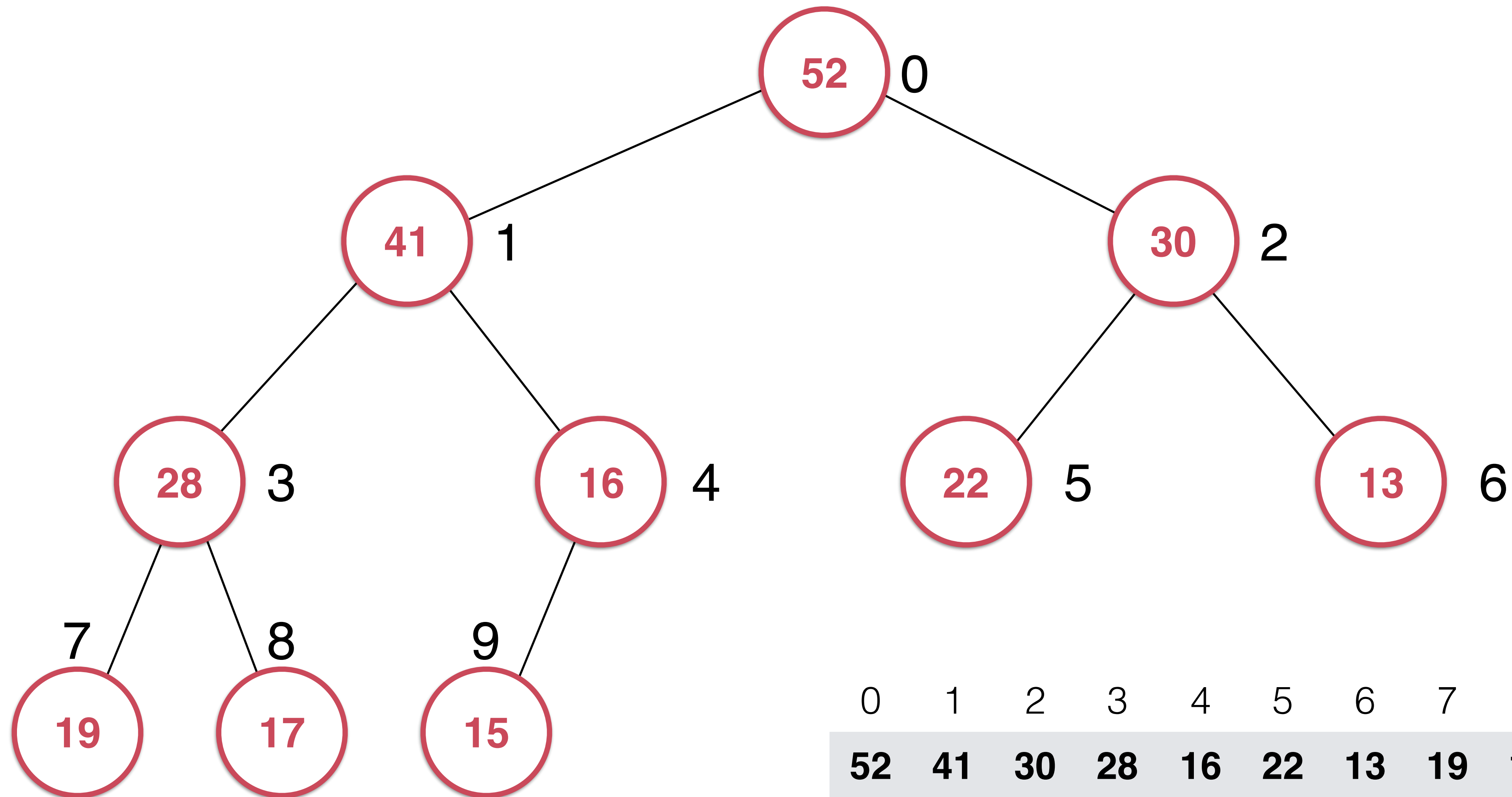
Sift Down



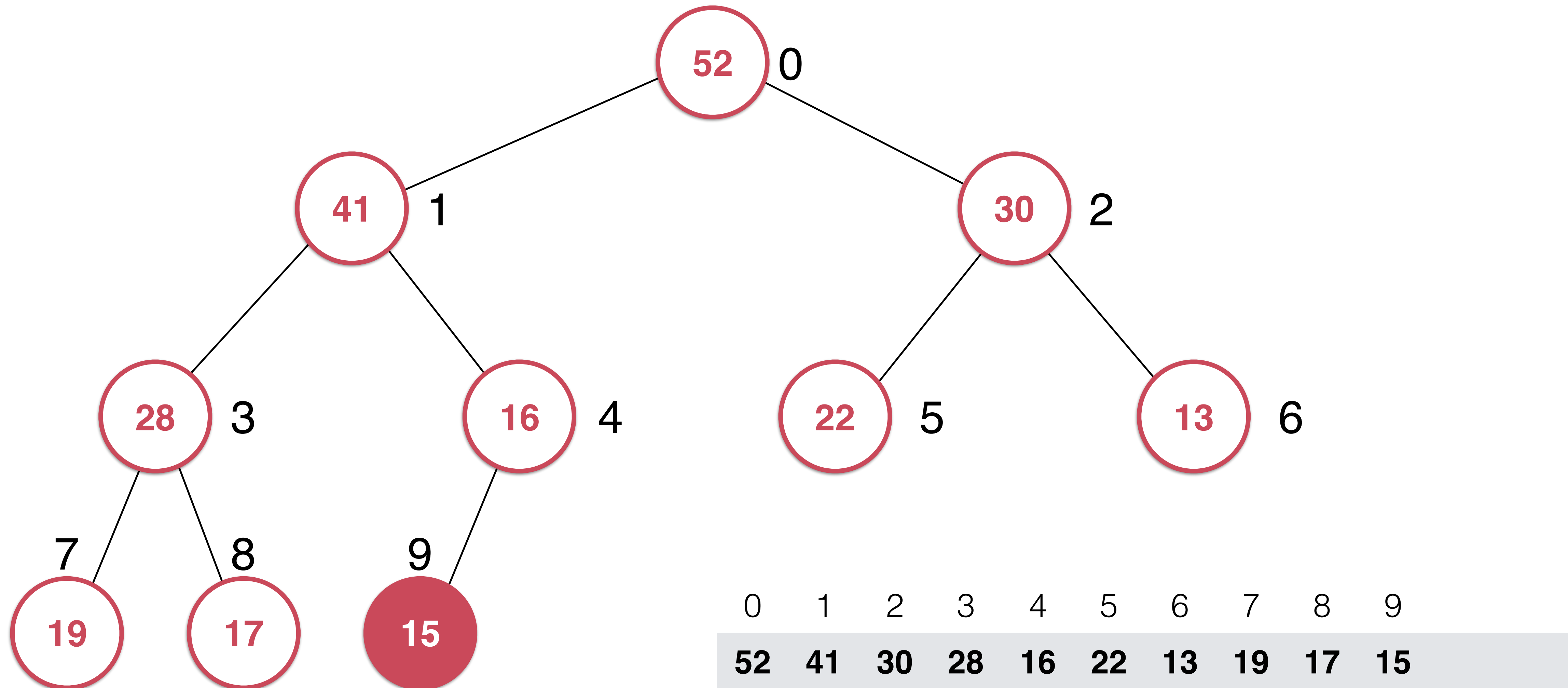
Sift Down



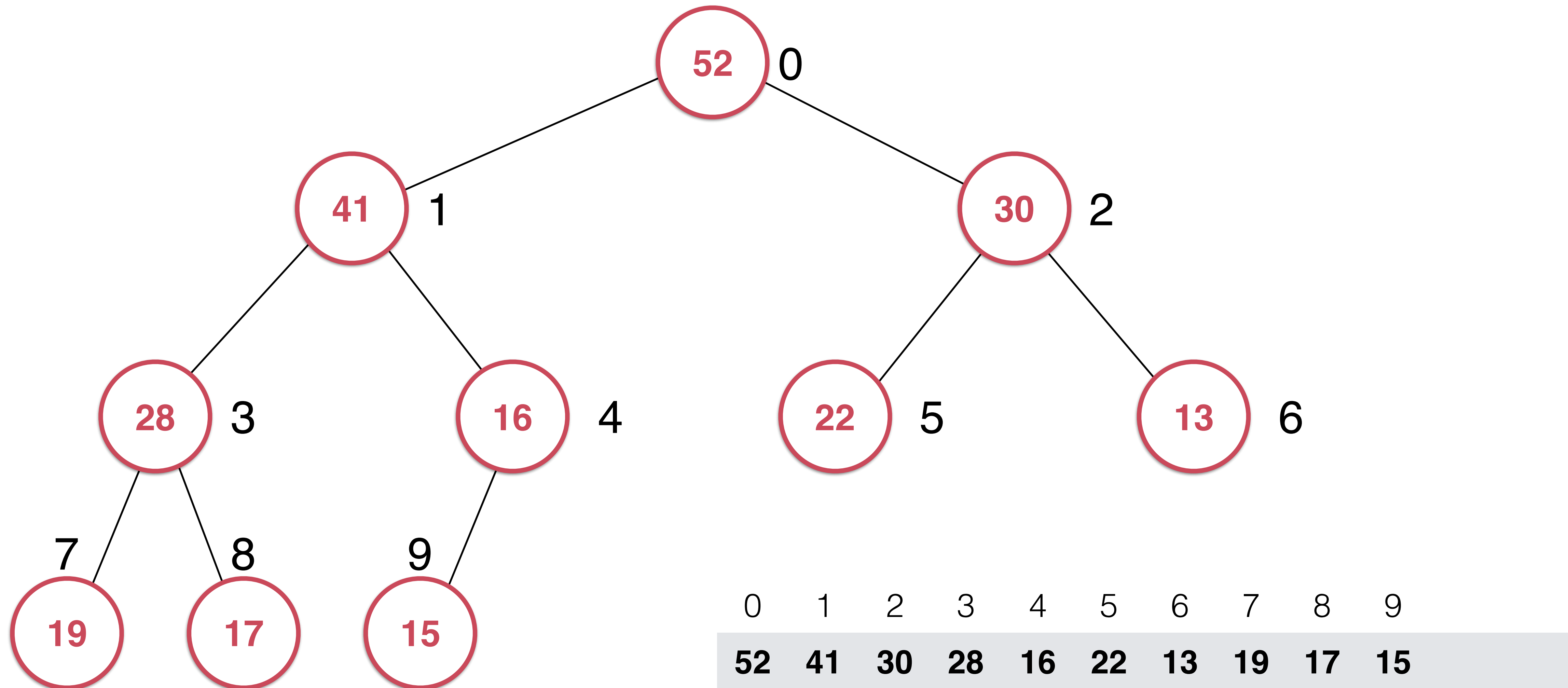
Sift Down



Sift Down



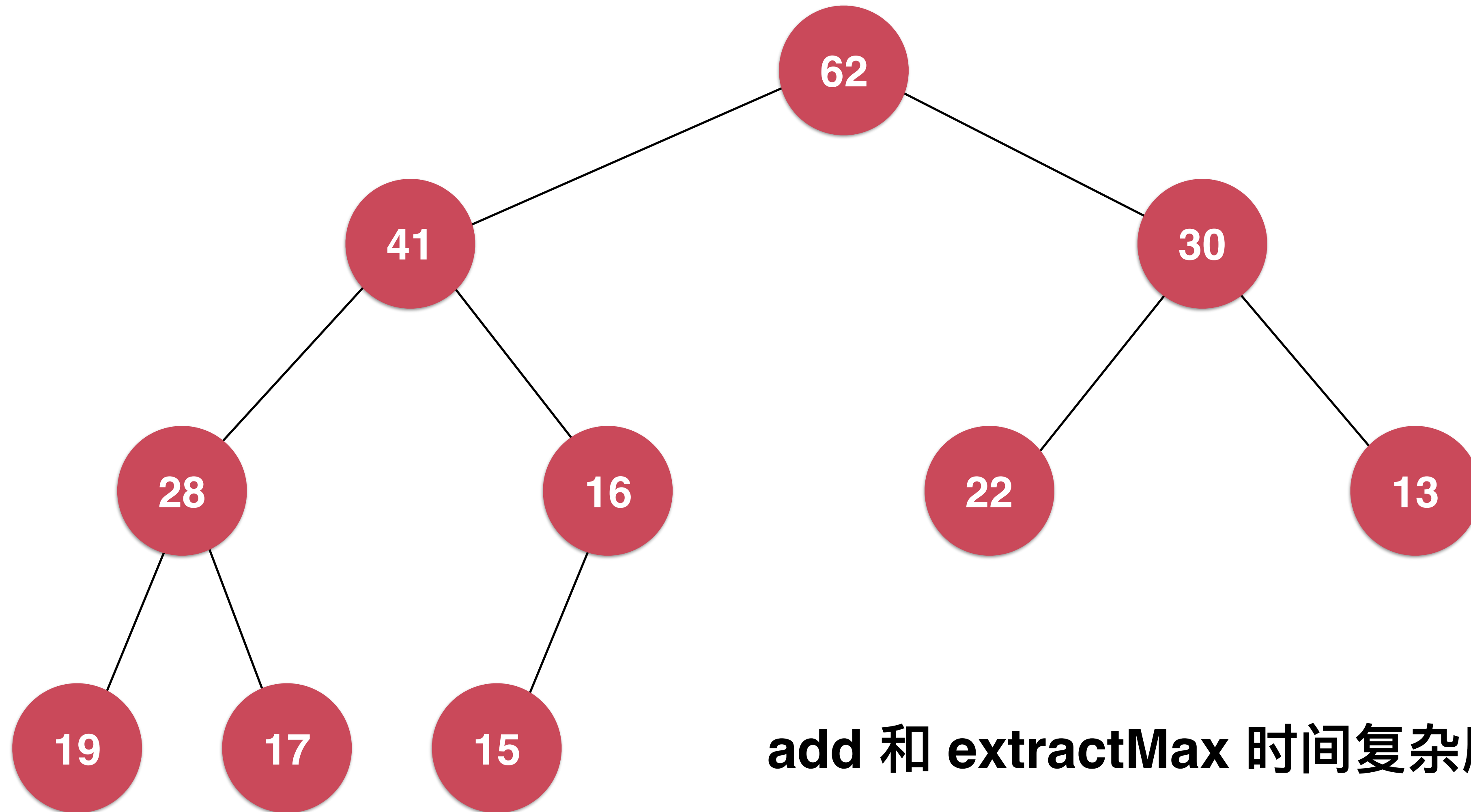
Sift Down



实践： Sift Down 和 extractMax

实践：测试堆

堆的时间复杂度分析



add 和 extractMax 时间复杂度都是 $O(\log n)$

Heapify 和 replace

replace

replace: 取出最大元素后, 放入一个新元素

实现: 可以先extractMax, 再add, 两次 $O(\log n)$ 的操作

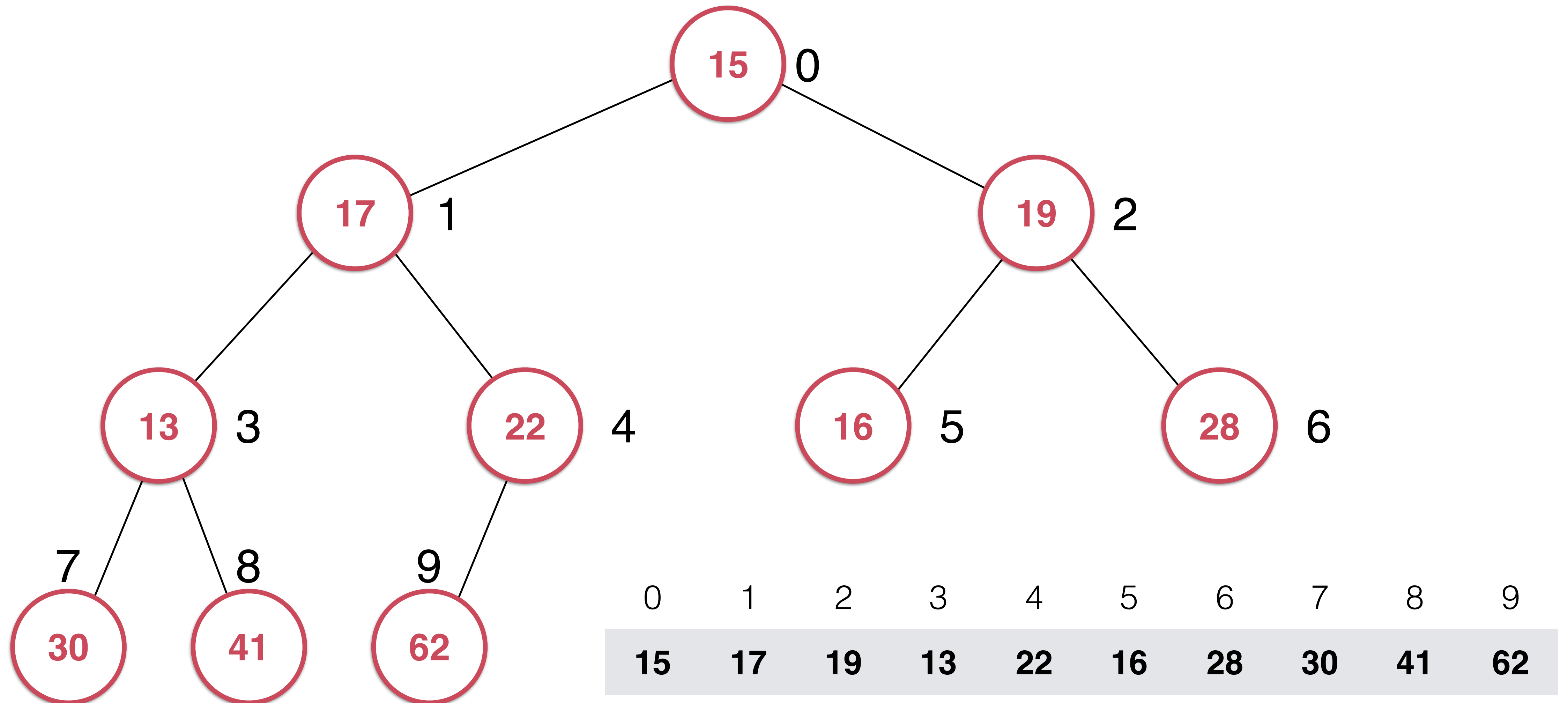
实现: 可以直接将堆顶元素替换以后Sift Down, 一次 $O(\log n)$ 的操作

实践： replace

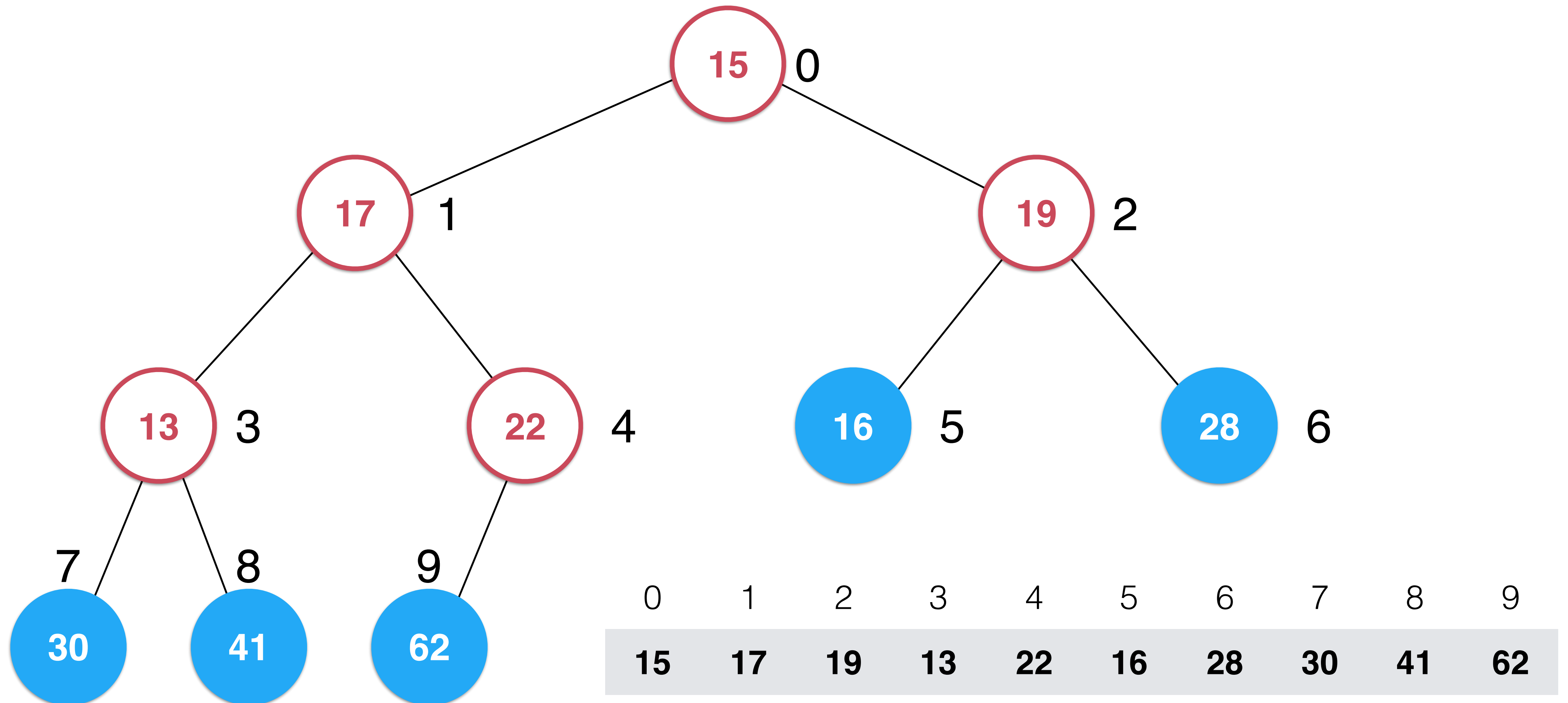
heapify

heapify: 将任意数组整理成堆的形状

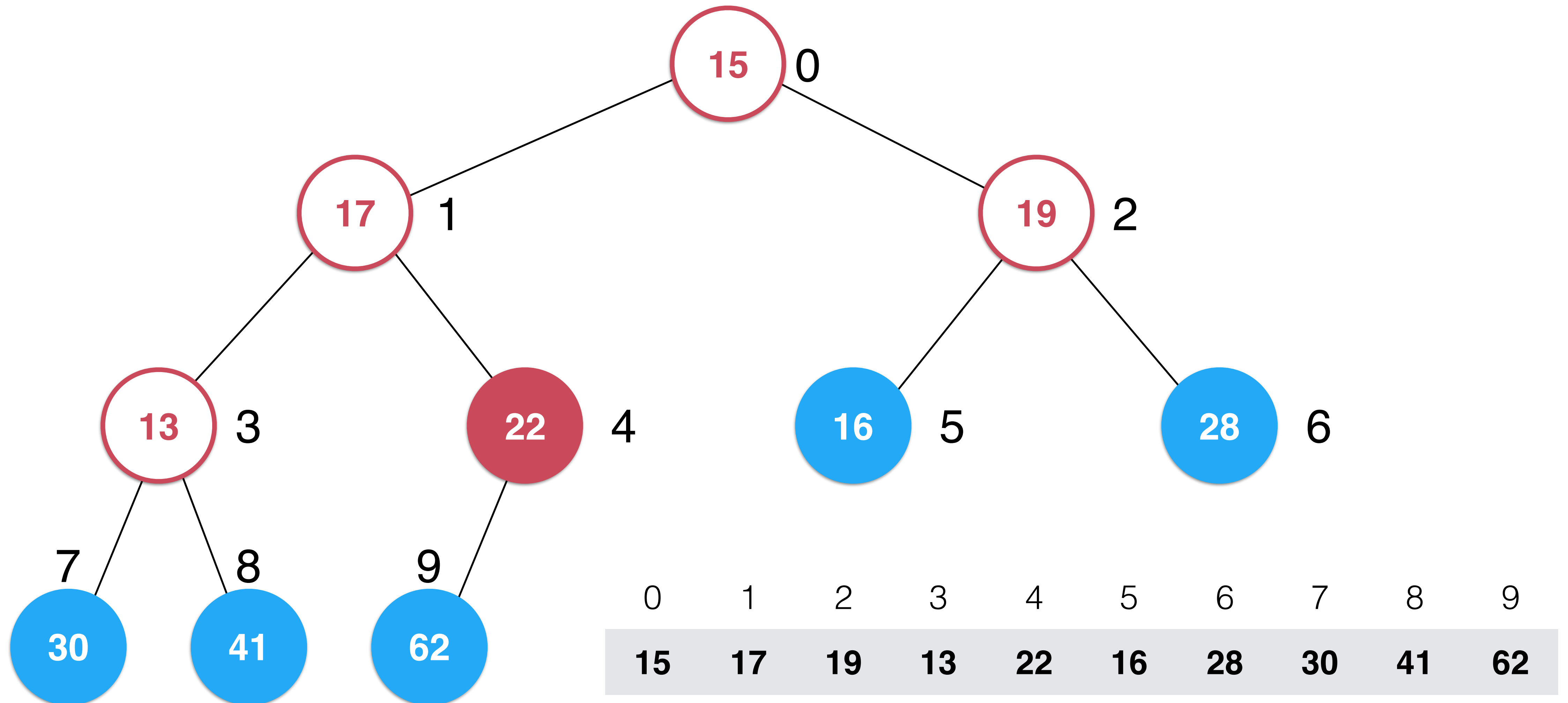
Heapify



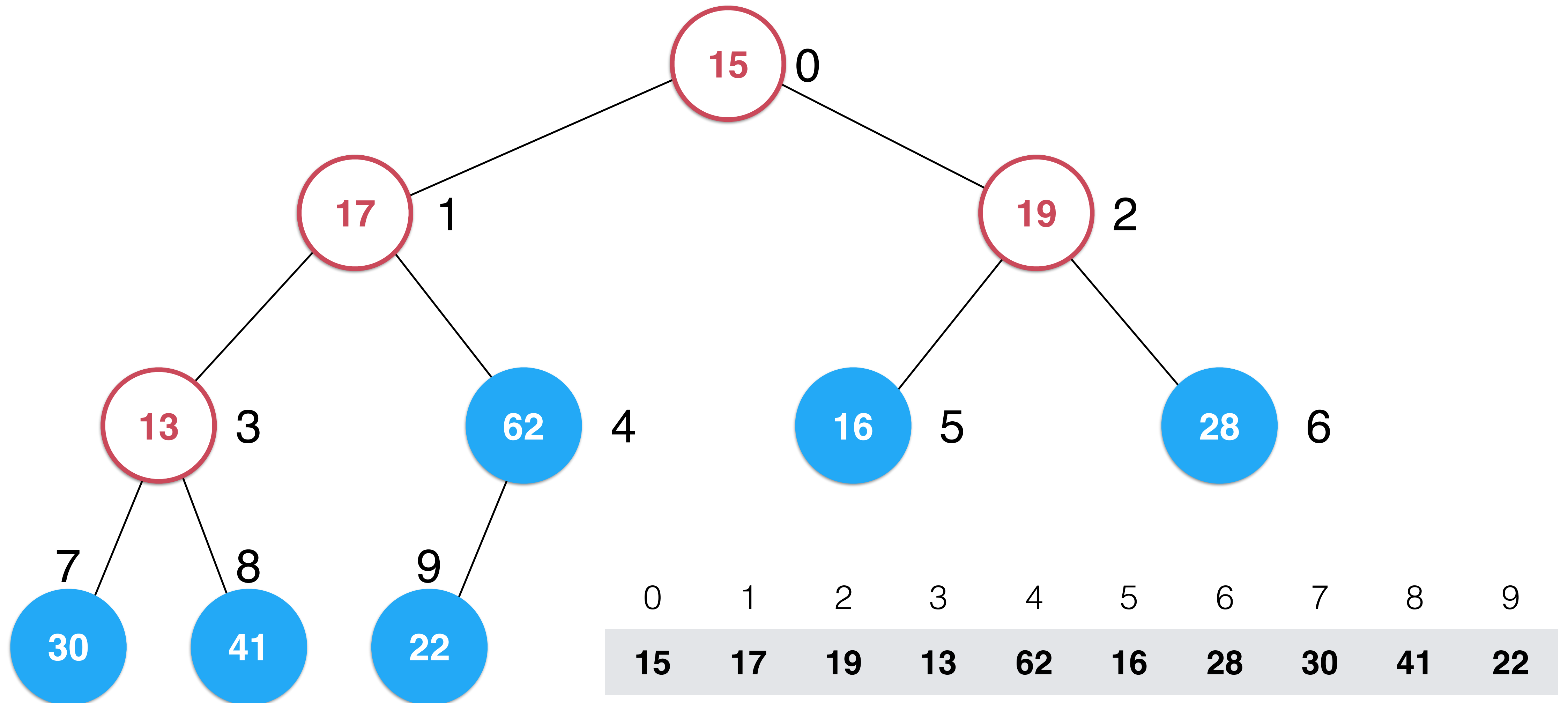
Heapify



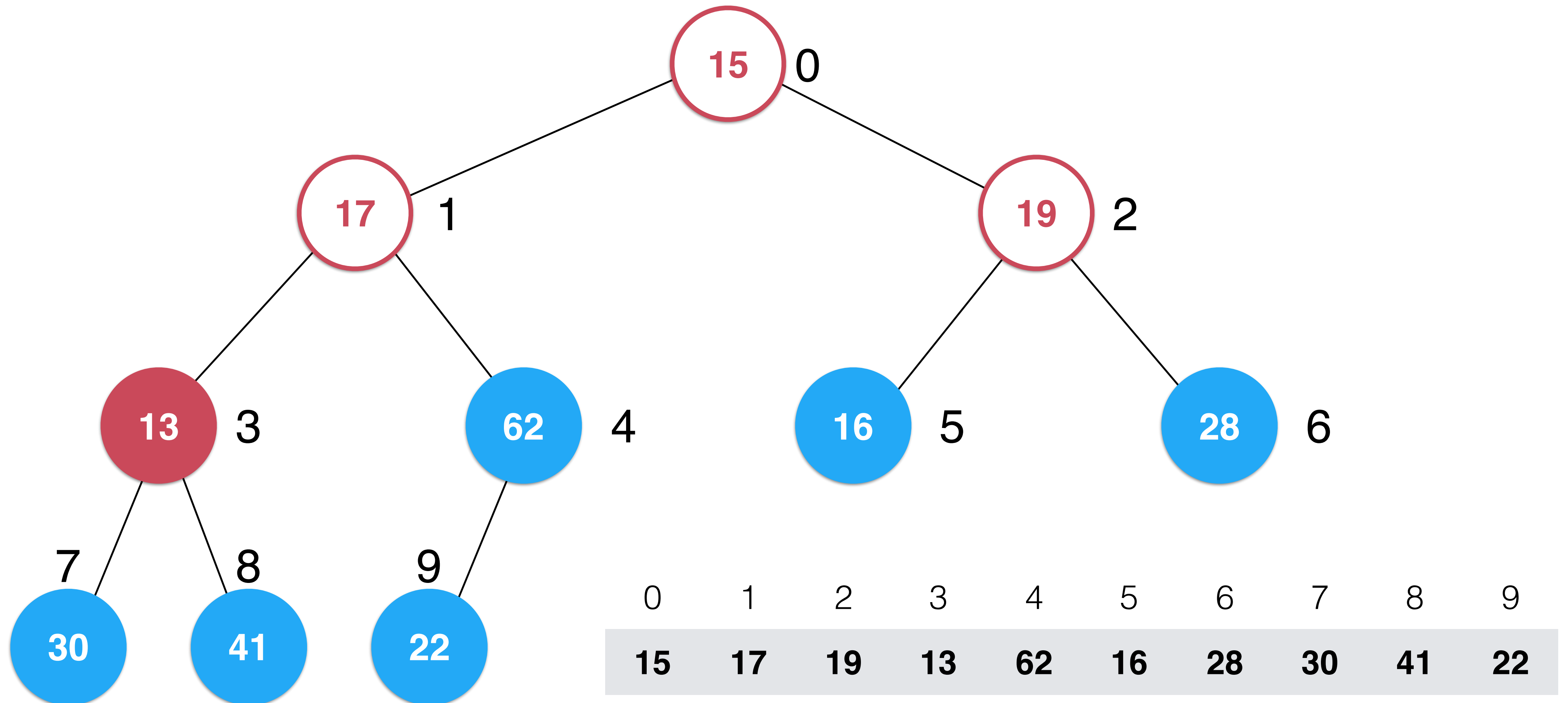
Heapify



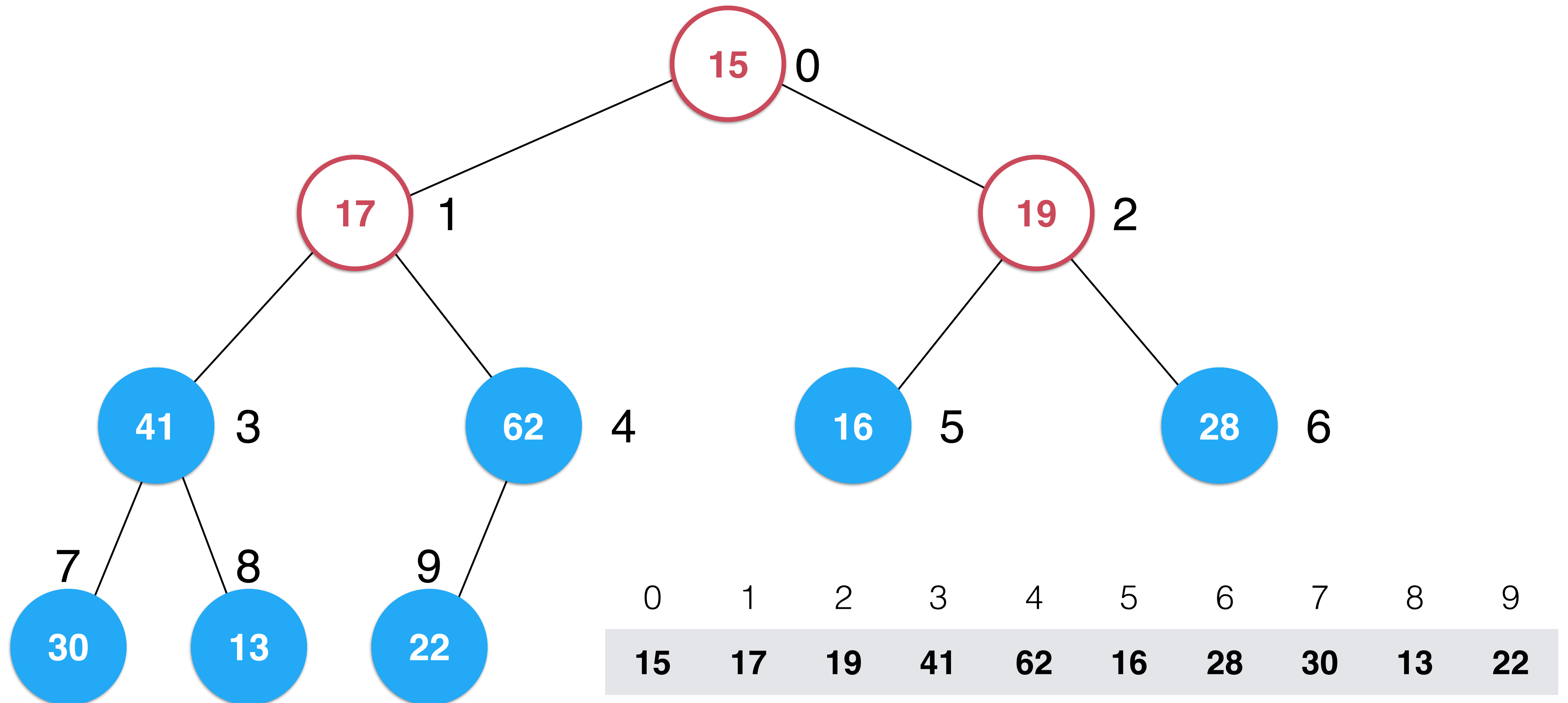
Heapify



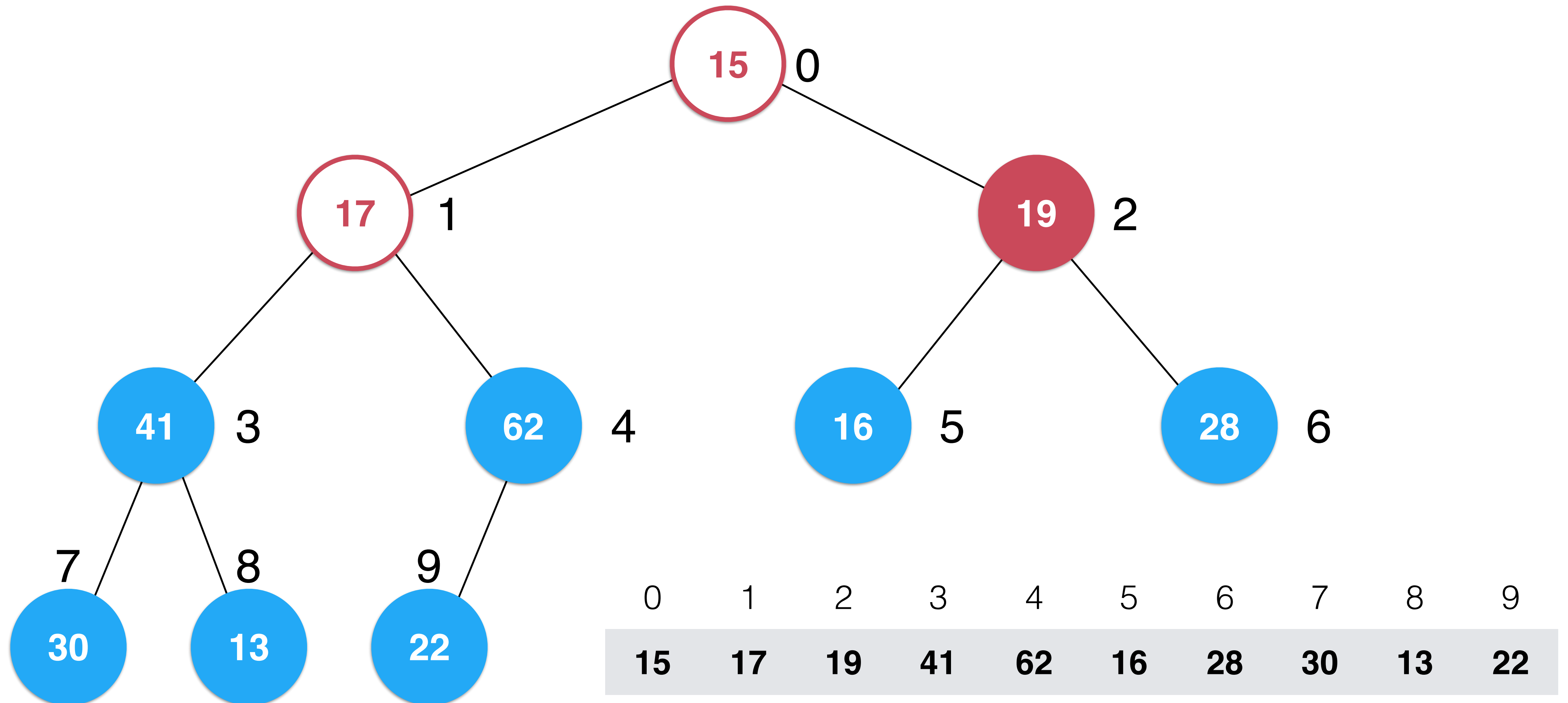
Heapify



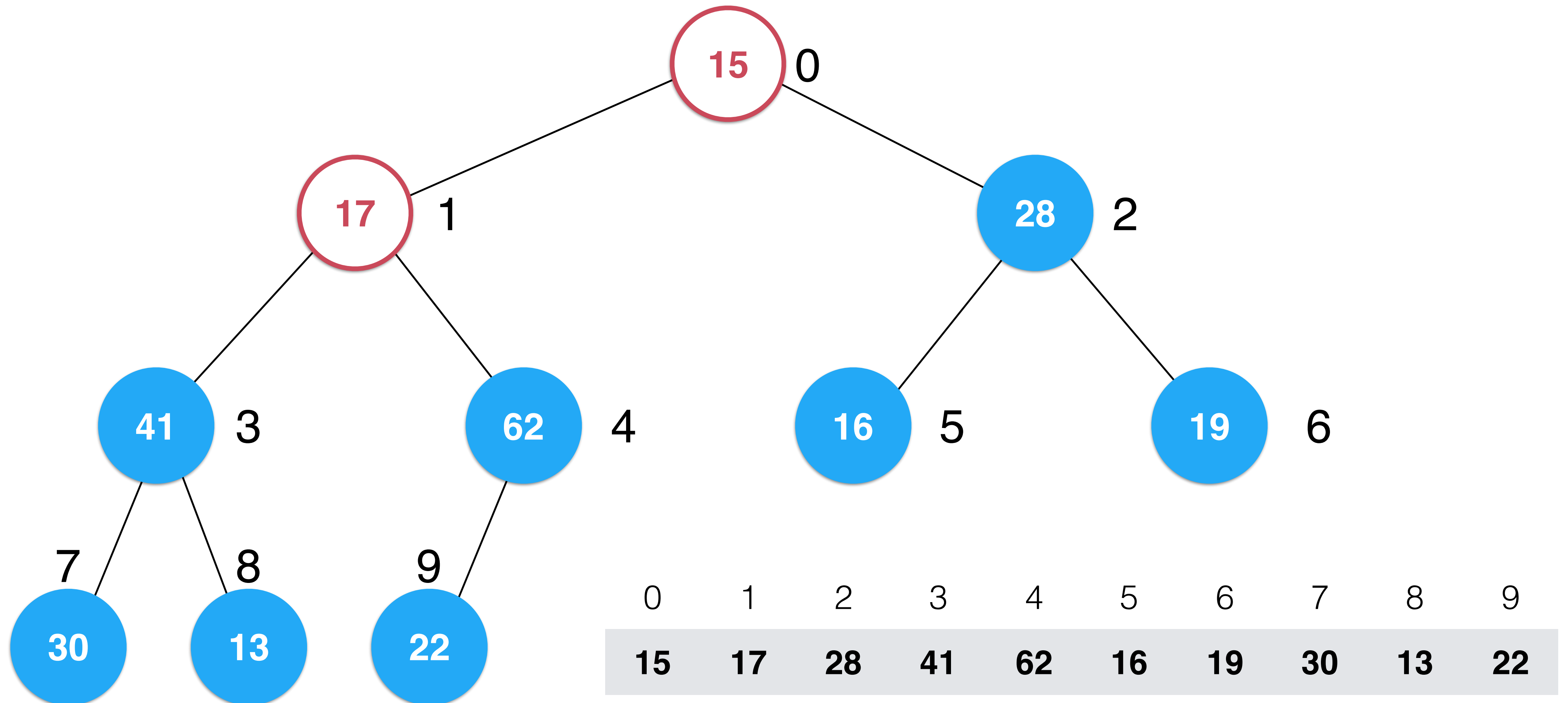
Heapify



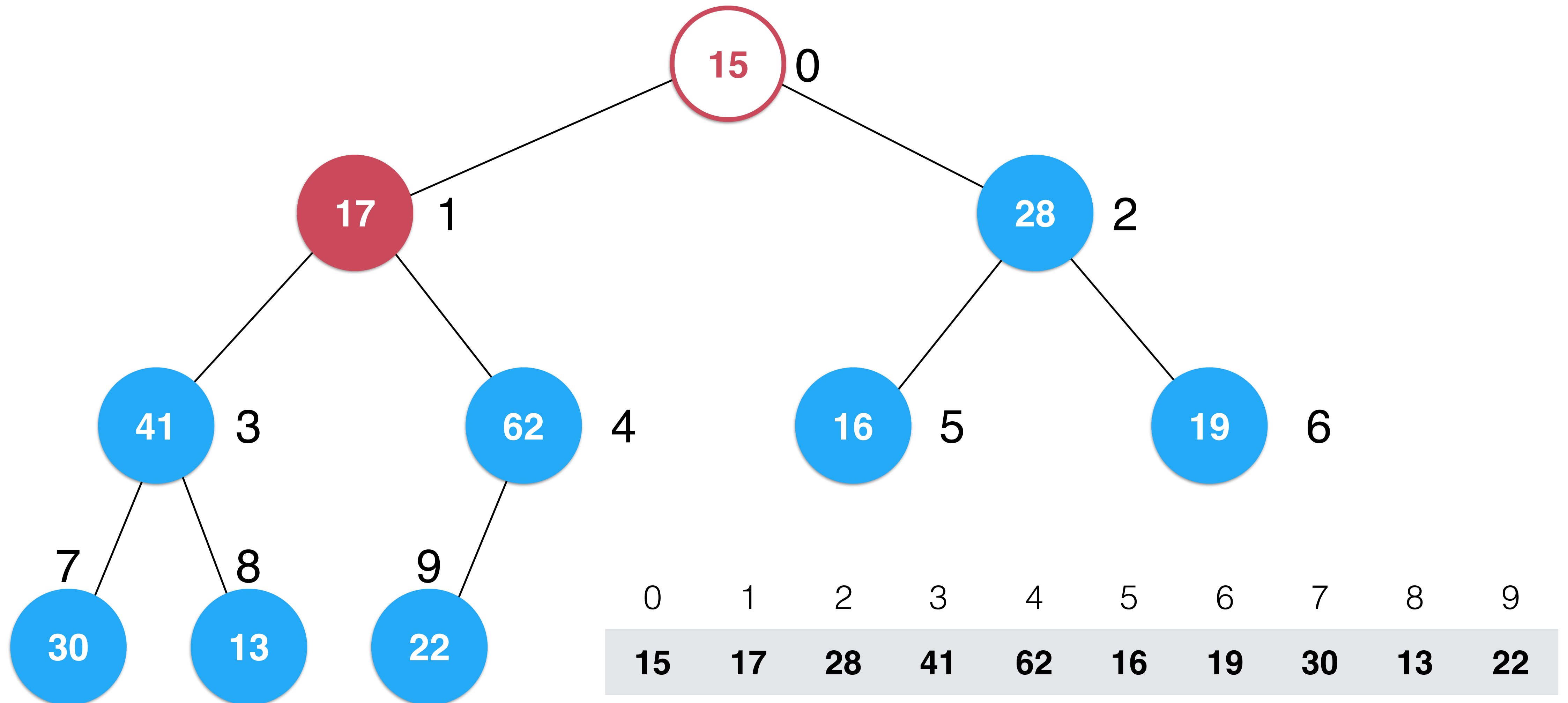
Heapify



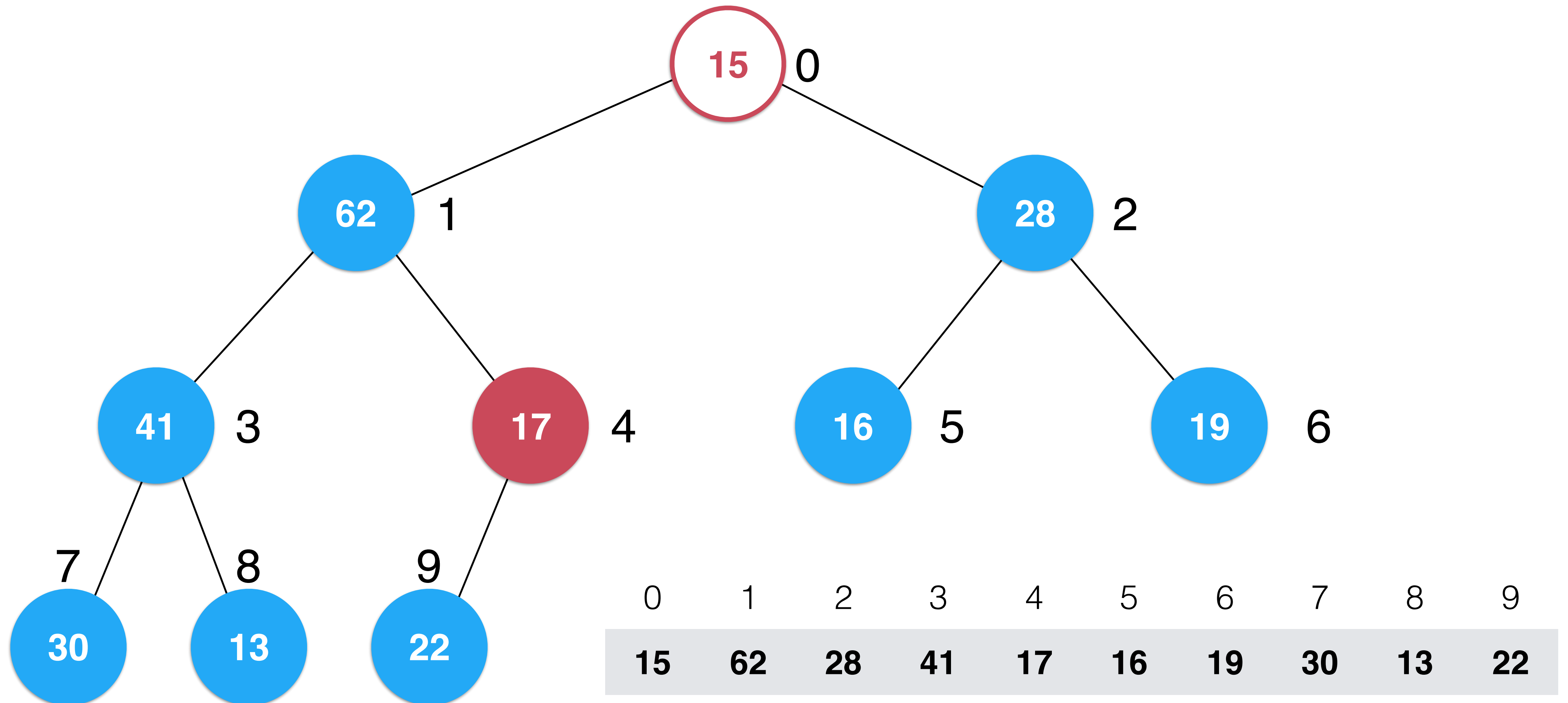
Heapify



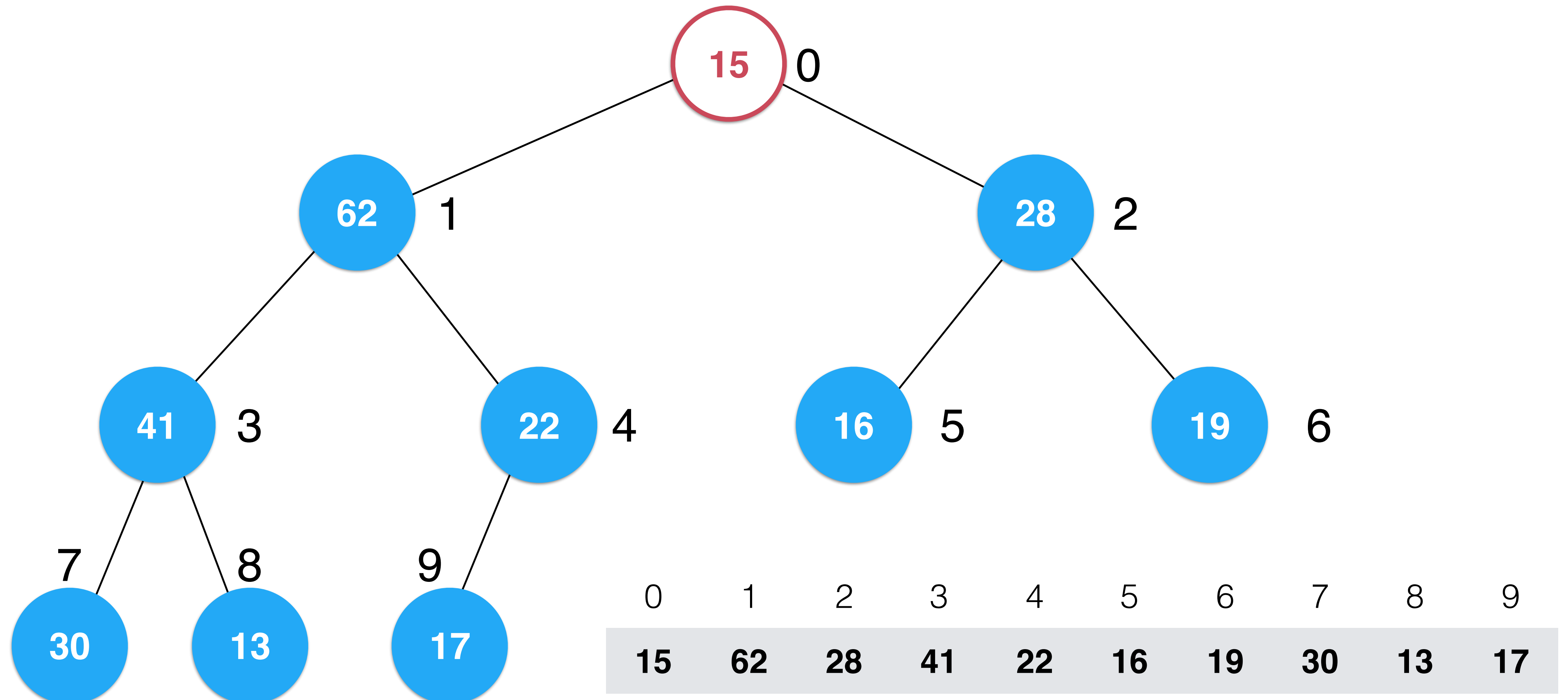
Heapify



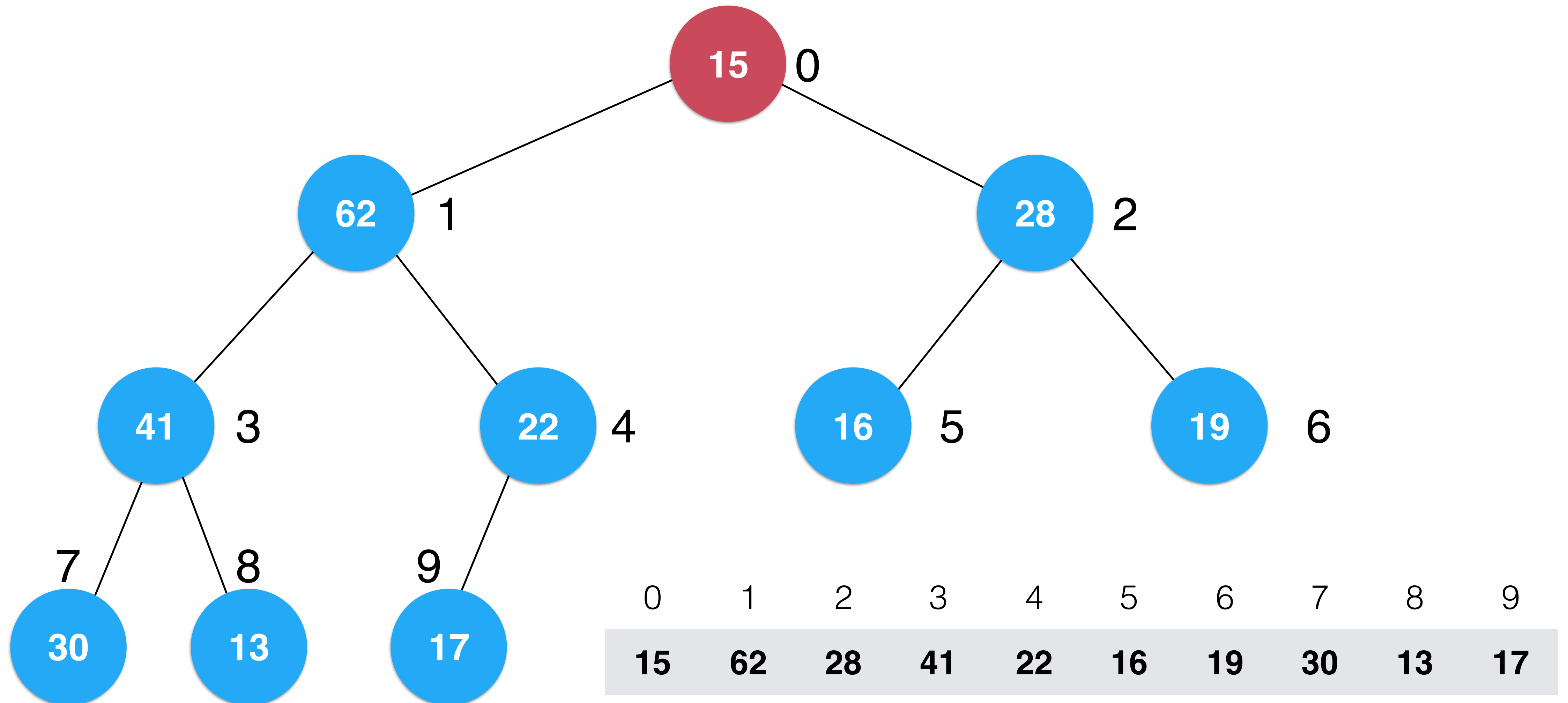
Heapify



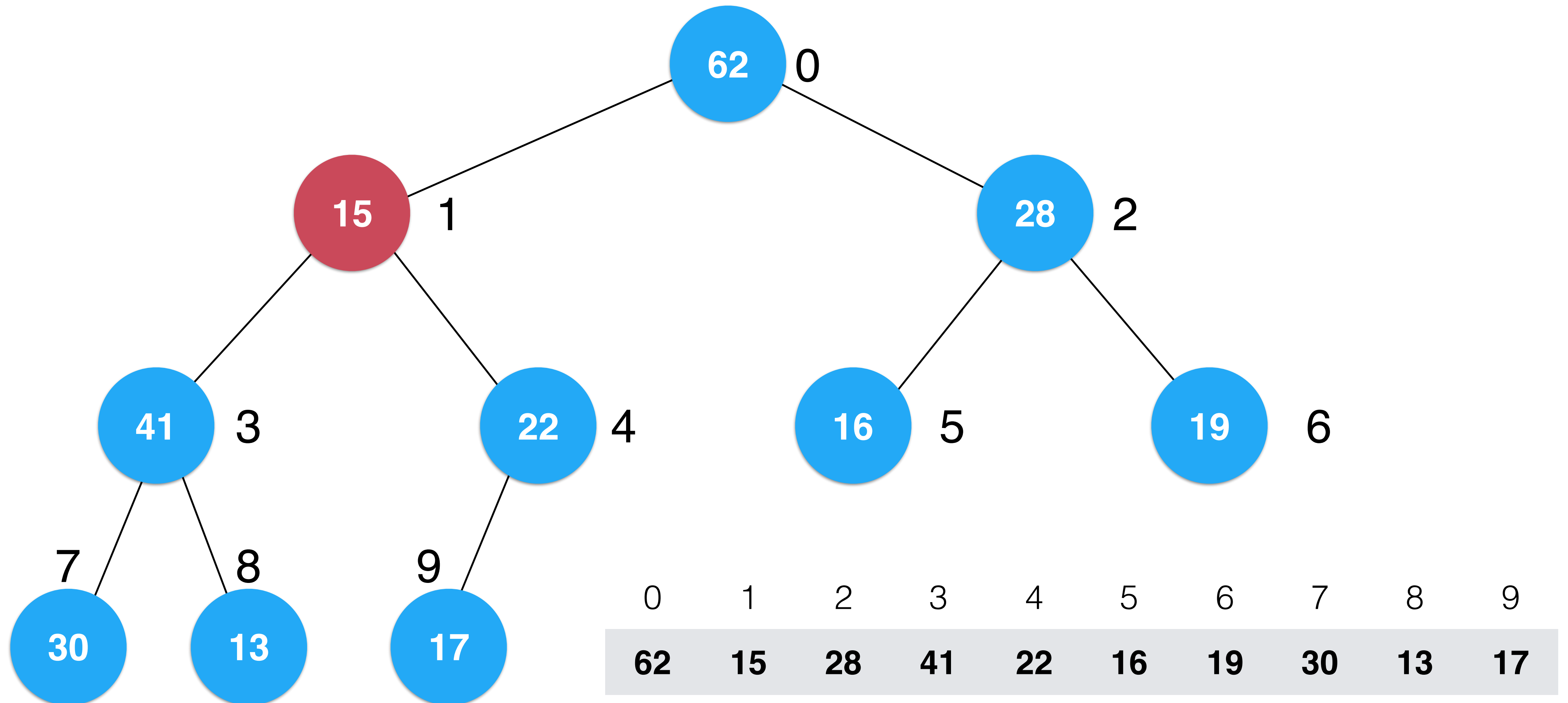
Heapify



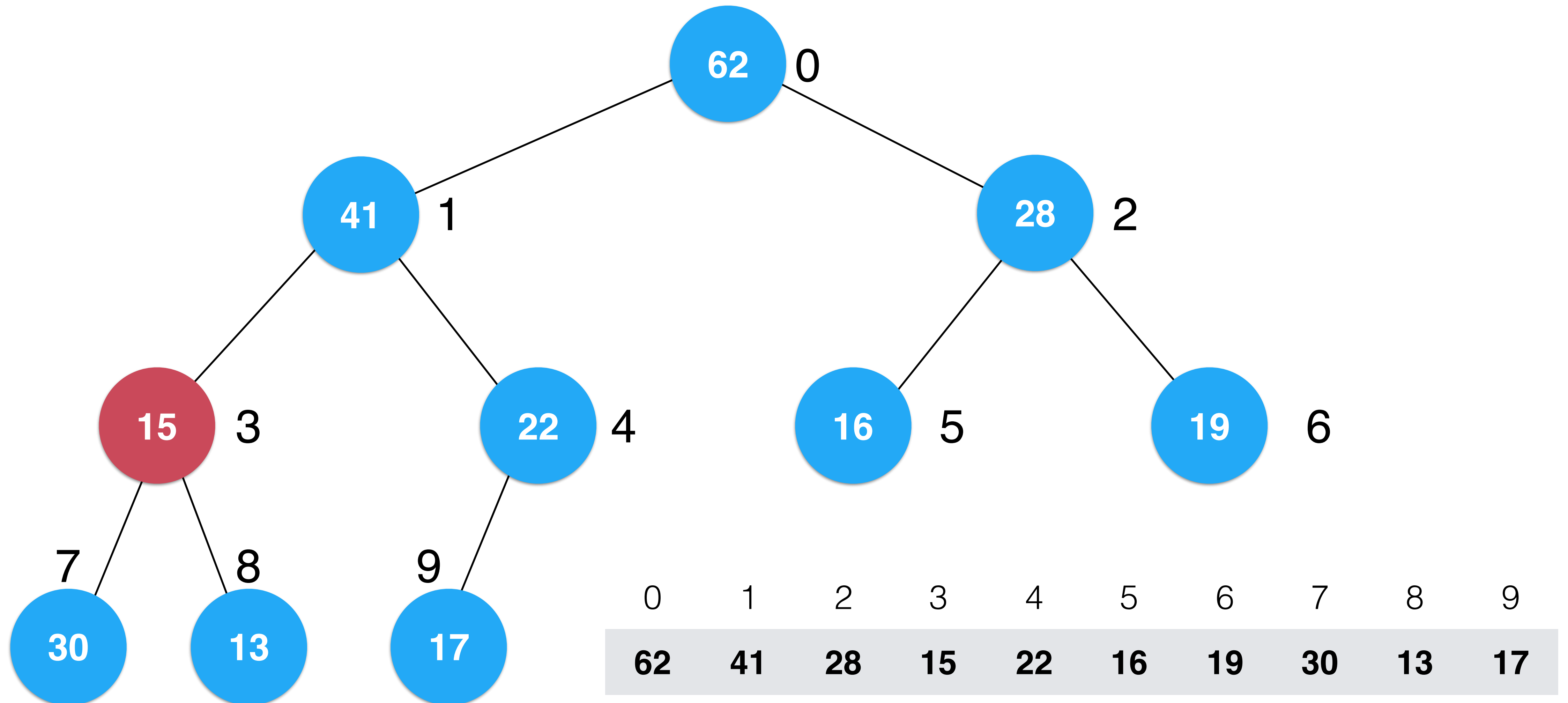
Heapify



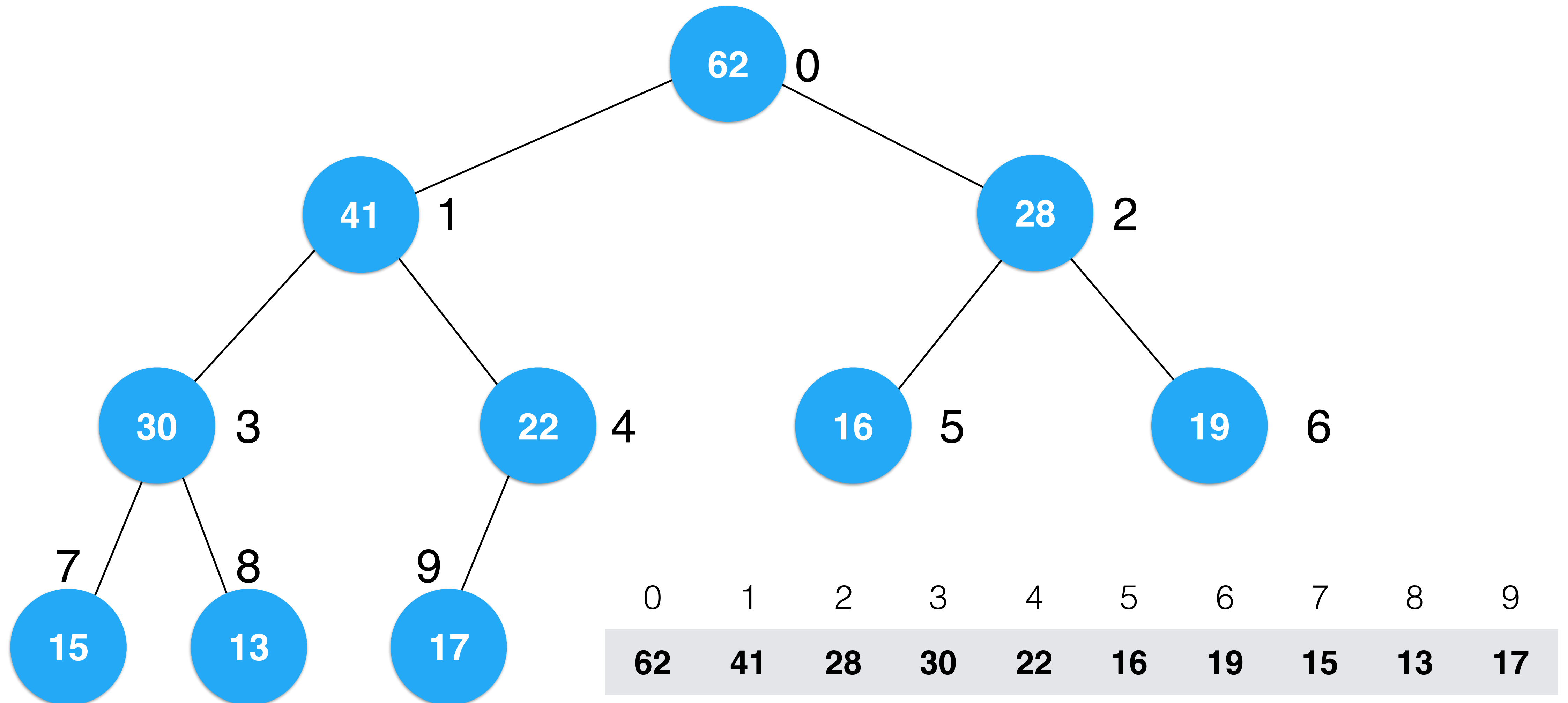
Heapify



Heapify



Heapify



操作：Heapify


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 $O(n)$

更多Heap相关的操作和Priority Queue

优先队列

Interface Queue<E>  PriorityQueue<E>
 implement

- void enqueue(E)
- E dequeue()
- E getFront()
- int getSize()
- boolean isEmpty()

可以使用不同的底层实现

实践：基于堆的优先队列实现

Leetcode 上优先队列相关的问题

优先队列的经典问题

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? $N \log M$

优先队列的经典问题

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

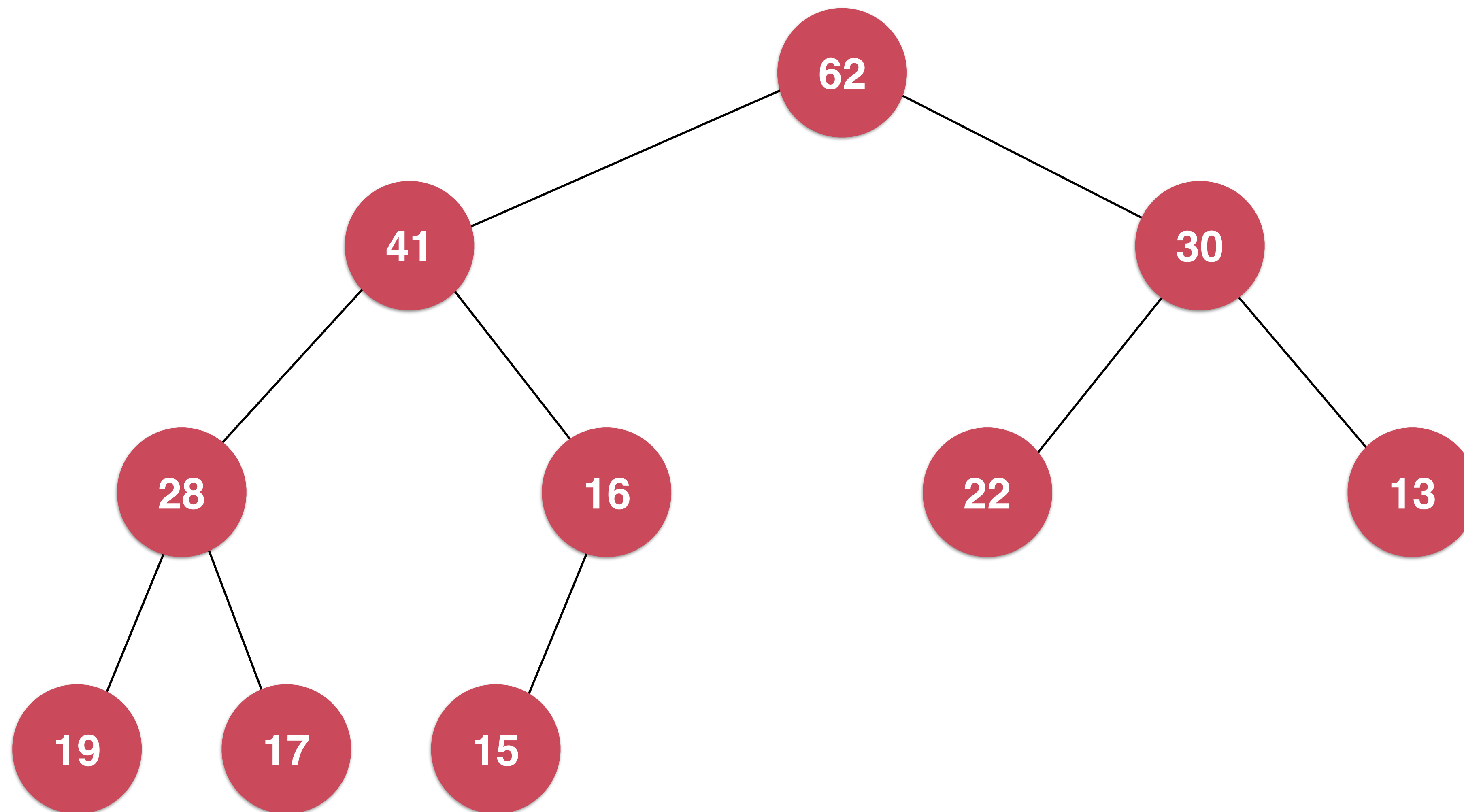
使用优先队列，维护当前看到的前M个元素

需要使用最小堆

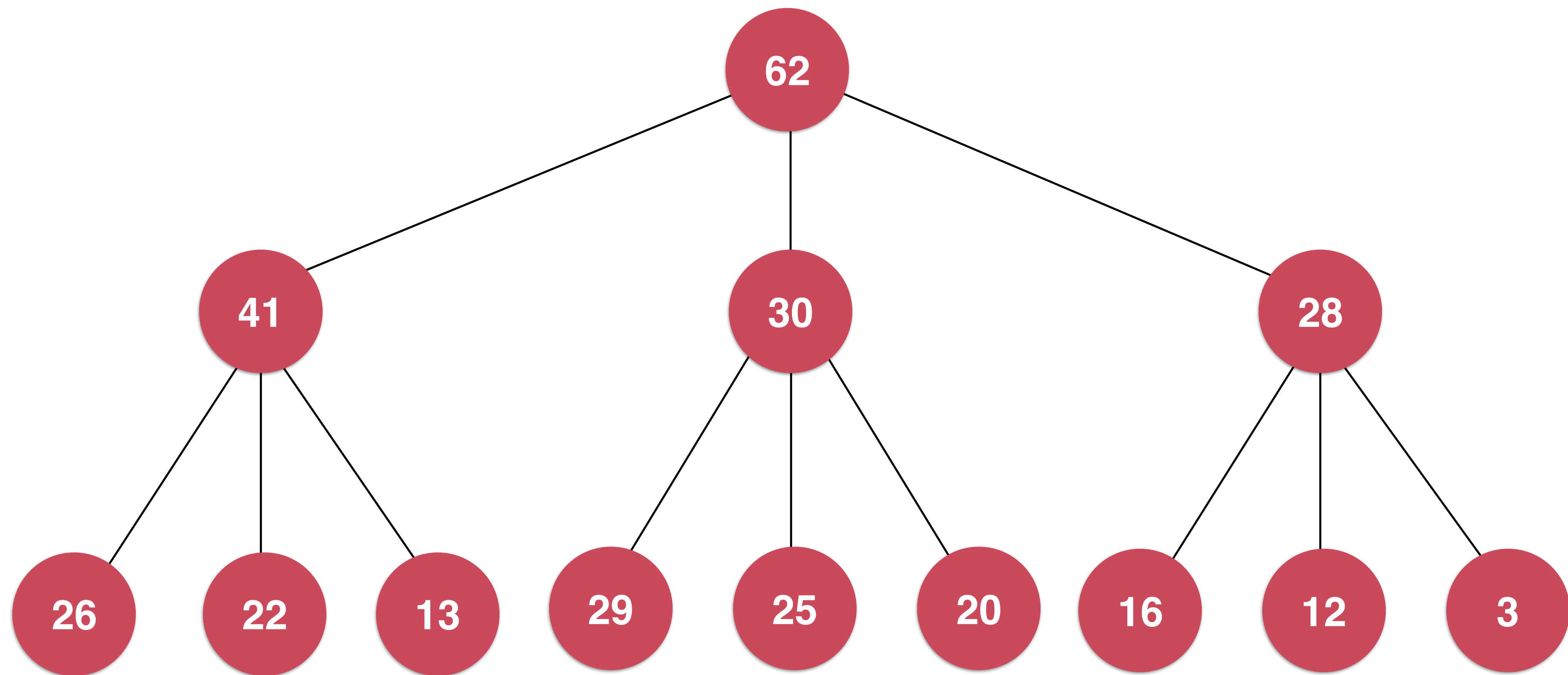
Java标准库中的优先队列

更多和堆，队列相关的话题

二叉堆 Binary Heap



d 叉堆 d-ary heap



索引|堆

二项堆

斐波那契堆

广义队列

广义队列

Interface Queue<E>

- void enqueue(E)
- E dequeue()
- E getFront()
- int getSize()
- boolean isEmpty()

广义队列

普通队列，优先队列

栈，也可以理解成是一个队列

随机队列

其他

欢迎大家关注我的个人公众号：是不是很酷



玩儿转数据结构

liuyubobobo