



本节内容

折半查找

知识总览

折半查找

算法思想

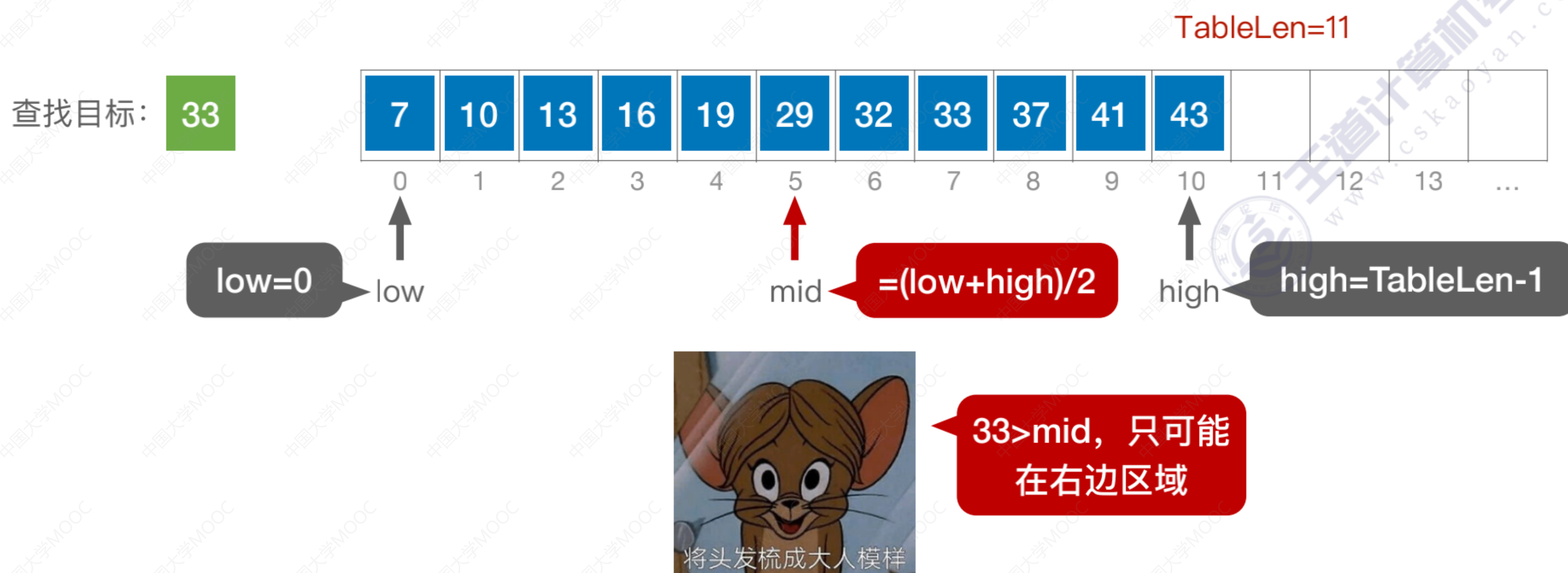
算法实现

查找判定树

折半查找效率

折半查找的算法思想

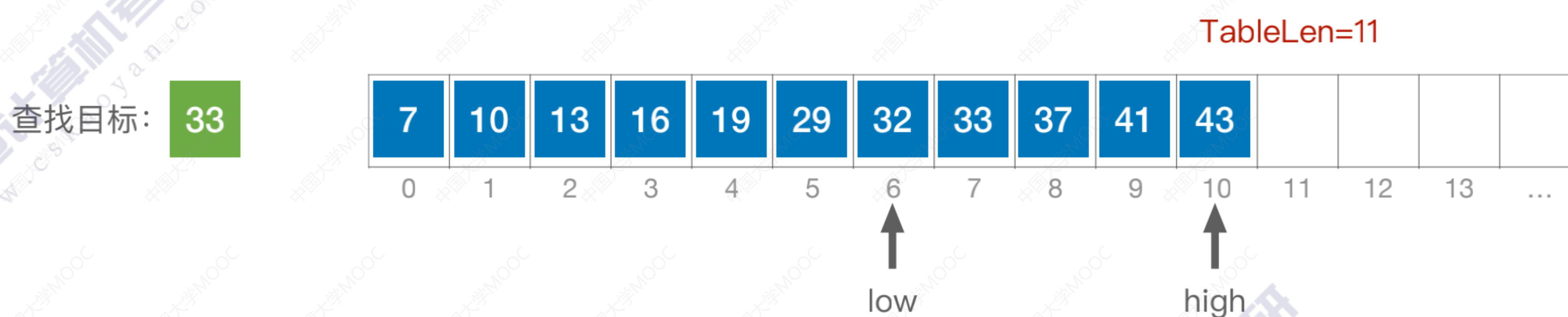
折半查找，又称“二分查找”，仅适用于**有序**的**顺序表**。



王道考研/CSKAOYAN.COM

折半查找的算法思想

折半查找，又称“二分查找”，仅适用于**有序**的**顺序表**。



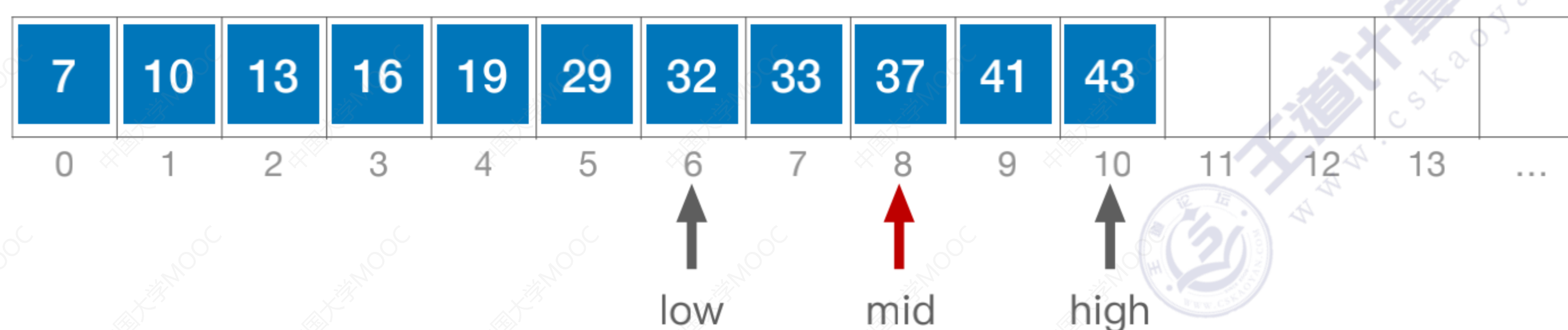
注：只有在 $[low, high]$ 之间才有可能找到目标关键字

王道考研/CSKAOYAN.COM

折半查找的算法思想

折半查找，又称“二分查找”，仅适用于**有序**的**顺序表**。

查找目标: 33



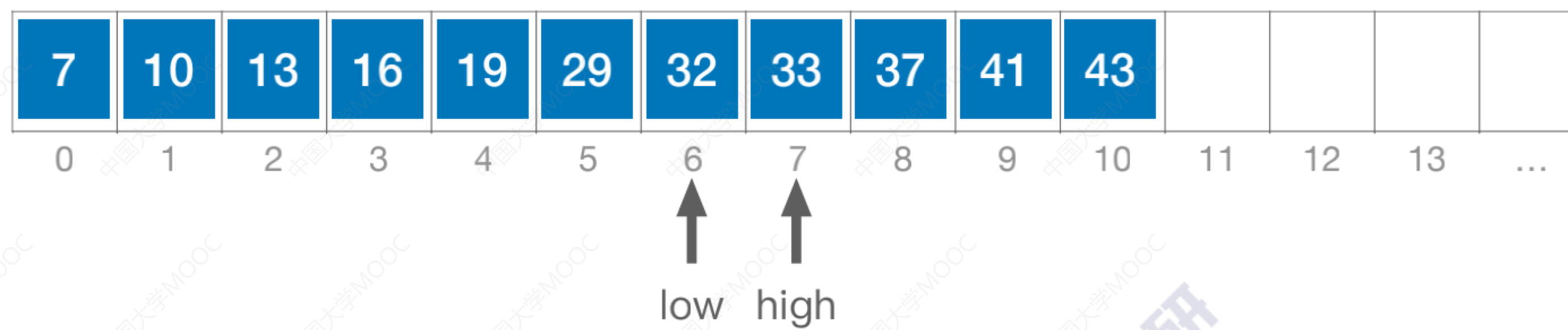
33 < mid, 只可能在左边区域

王道考研/CSKAOYAN.COM

折半查找的算法思想

折半查找，又称“二分查找”，仅适用于**有序**的**顺序表**。

查找目标: 33

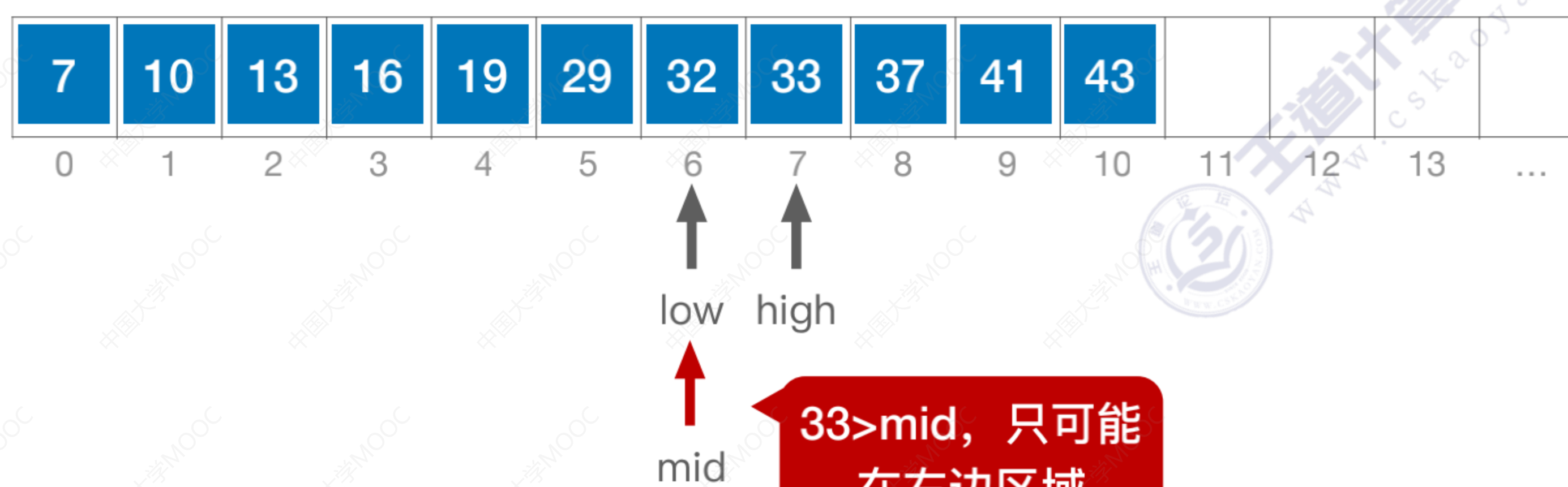


王道考研/CSKAOYAN.COM

折半查找的算法思想

折半查找，又称“二分查找”，仅适用于**有序**的**顺序表**。

查找目标: 33

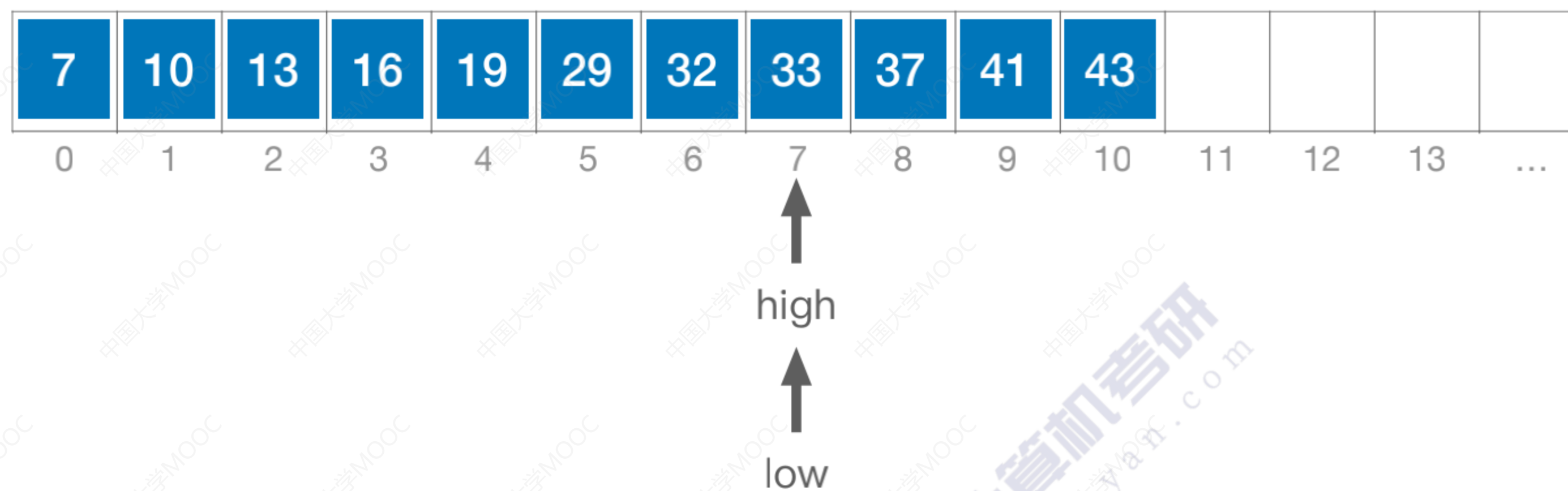


王道考研/CSKAOYAN.COM

折半查找的算法思想

折半查找，又称“二分查找”，仅适用于**有序**的**顺序表**。

查找目标: 33



王道考研/CSKAOYAN.COM

折半查找的算法思想

折半查找，又称“二分查找”，仅适用于**有序**的**顺序表**。

查找目标:

33

TableLen=11

7	10	13	16	19	29	32	33	37	41	43				
0	1	2	3	4	5	6	7	8	9	10	11	12	13	...

high

low

mid

33==mid
查找成功

王道考研/CSKAOYAN.COM

折半查找的算法思想

折半查找，又称“二分查找”，仅适用于**有序**的**顺序表**。

查找目标:

12

TableLen=11

7	10	13	16	19	29	32	33	37	41	43				
0	1	2	3	4	5	6	7	8	9	10	11	12	13	...

low

12 < mid, 只可能在左边区域

high

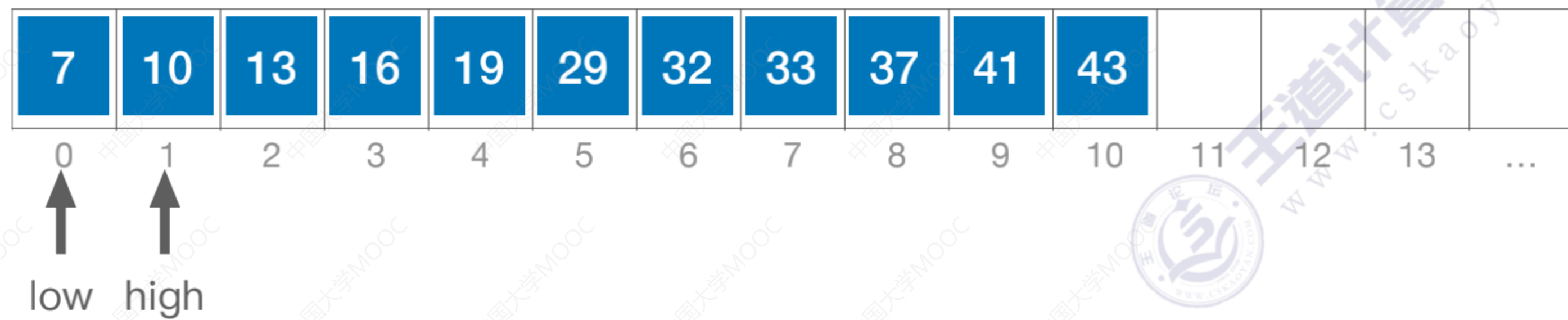
王道考研/CSKAOYAN.COM

折半查找的算法思想

折半查找，又称“二分查找”，仅适用于**有序**的**顺序表**。

查找目标:

12



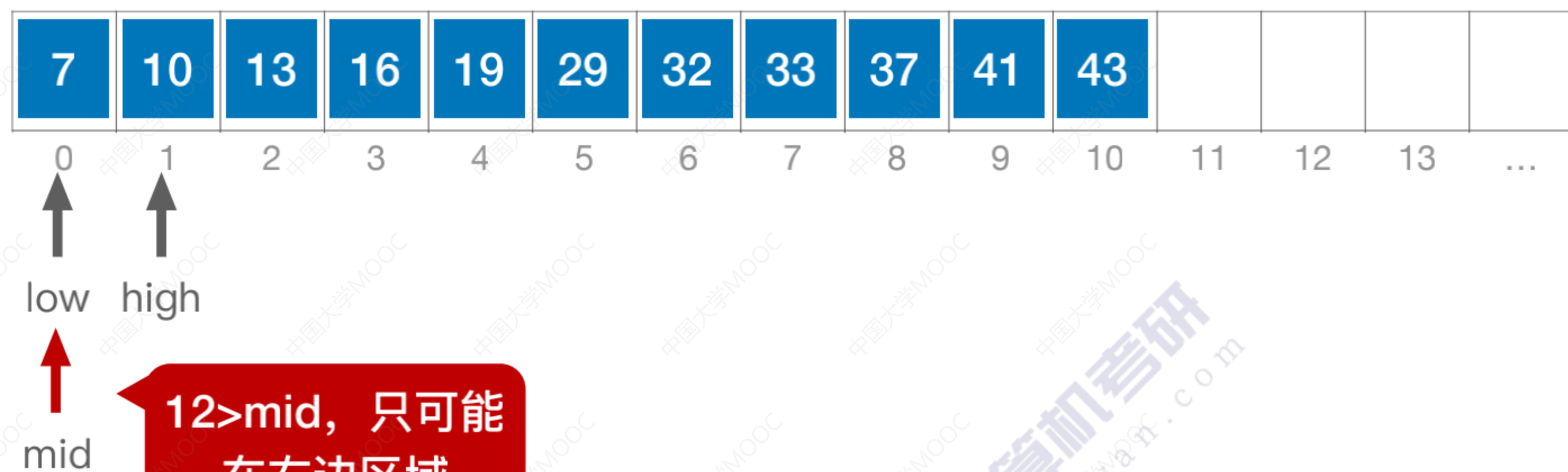
王道考研/CSKAOYAN.COM

折半查找的算法思想

折半查找，又称“二分查找”，仅适用于**有序**的**顺序表**。

查找目标:

12



王道考研/CSKAOYAN.COM

折半查找的算法思想

折半查找，又称“二分查找”，仅适用于**有序**的**顺序表**。

查找目标:

12

TableLen=11

7	10	13	16	19	29	32	33	37	41	43				
0	1	2	3	4	5	6	7	8	9	10	11	12	13	...

↑
high

↑
low

王道考研/CSKAOYAN.COM

折半查找的算法思想

折半查找，又称“二分查找”，仅适用于**有序**的**顺序表**。

查找目标:

12

TableLen=11

7	10	13	16	19	29	32	33	37	41	43				
0	1	2	3	4	5	6	7	8	9	10	11	12	13	...

↑
high

↑
low

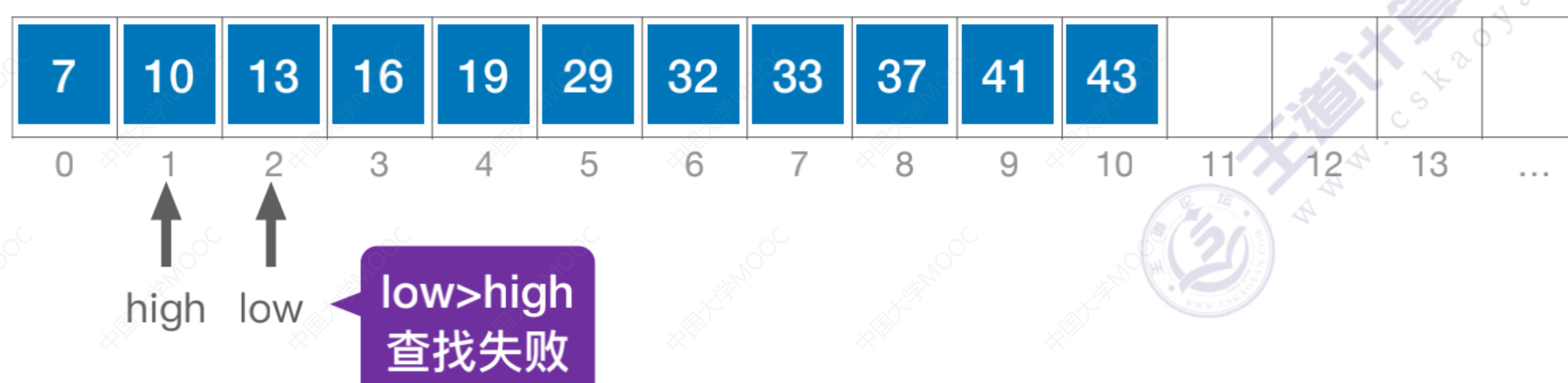
↑
mid

王道考研/CSKAOYAN.COM

折半查找的算法思想

折半查找，又称“二分查找”，仅适用于**有序**的**顺序表**。

查找目标: 12



王道考研/CSKAOYAN.COM

折半查找的实现

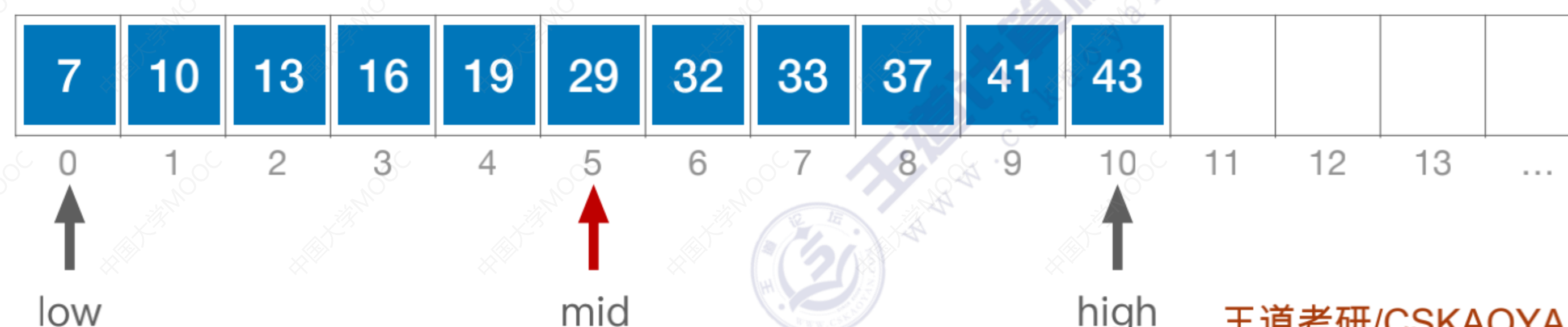
```
typedef struct{
    ElemType *elem; //查找表的数据结构 (顺序表)
    int TableLen; //动态数组基址
}SSTable; //表的长度
```

折半查找，又称“二分查找”，仅适用于**有序**的**顺序表**。

顺序表拥有随机访问的特性，链表没有

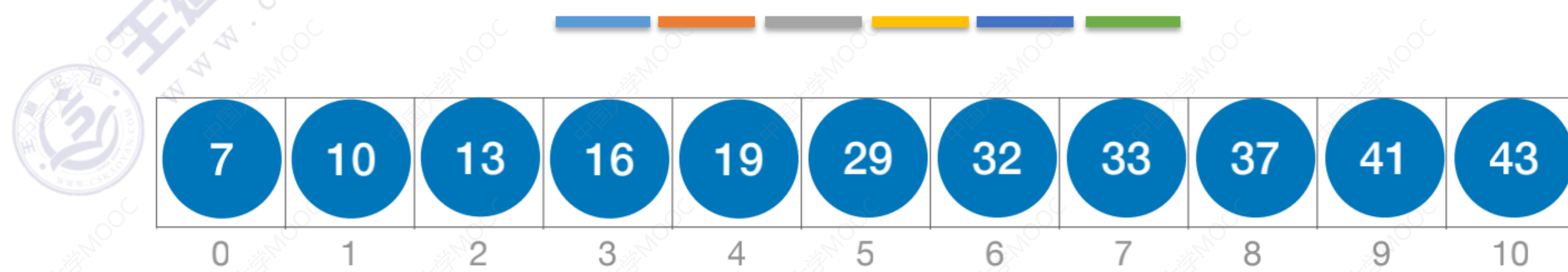
```
//折半查找
int Binary_Search(SSTable L, ElemType key){
    int low=0, high=L.TableLen-1, mid;
    while(low<=high){
        mid=(low+high)/2; //取中间位置
        if(L.elem[mid]==key) //查找成功则返回所在位置
            return mid;
        else if(L.elem[mid]>key) //从前半部分继续查找
            high=mid-1;
        else //从后半部分继续查找
            low=mid+1;
    }
    return -1; //查找失败，返回-1
}
```

查找目标: 33



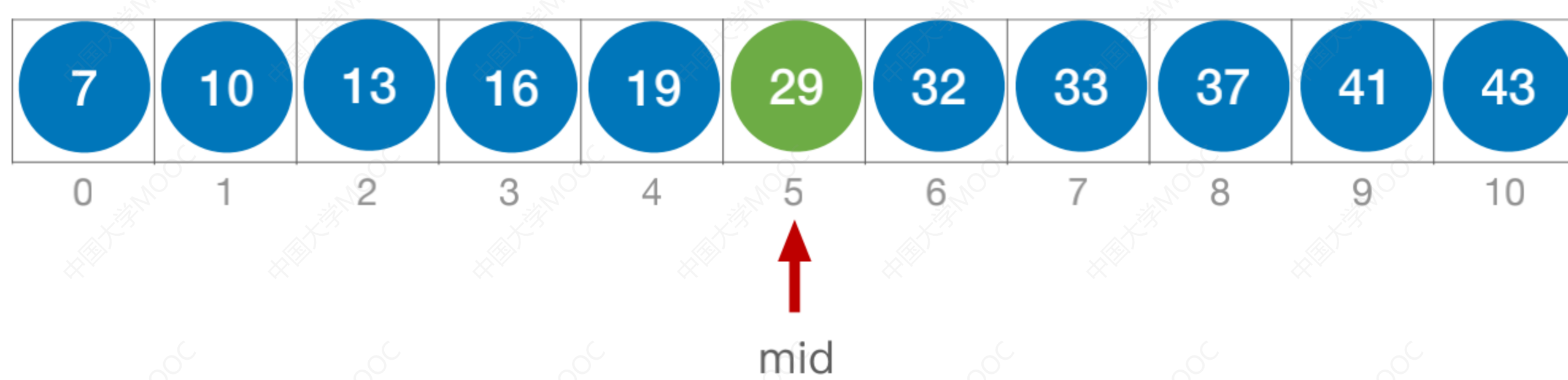
王道考研/CSKAOYAN.COM

查找效率分析



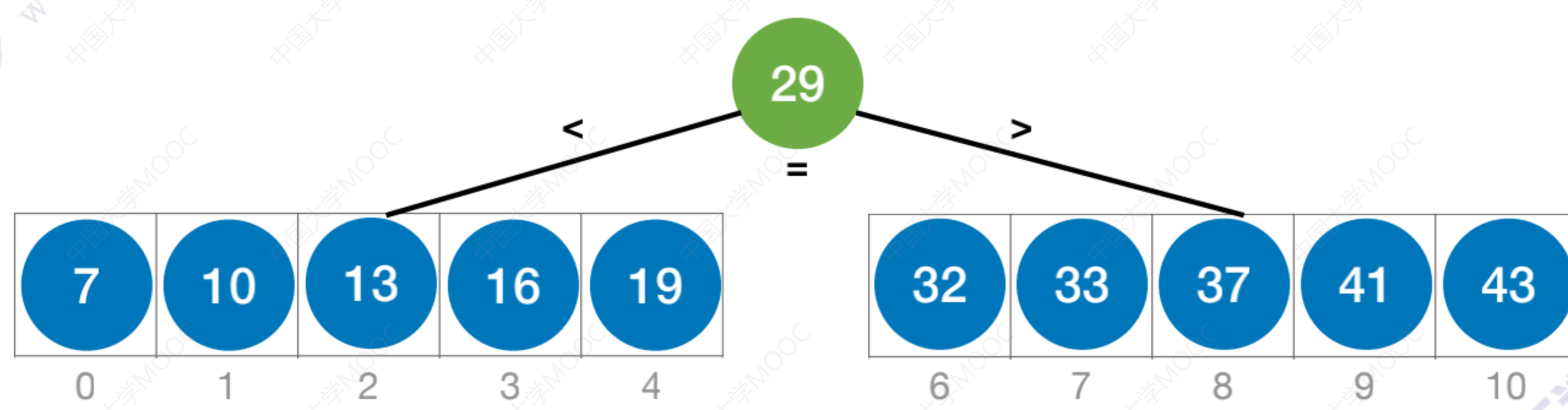
王道考研/CSKAOYAN.COM

查找效率分析



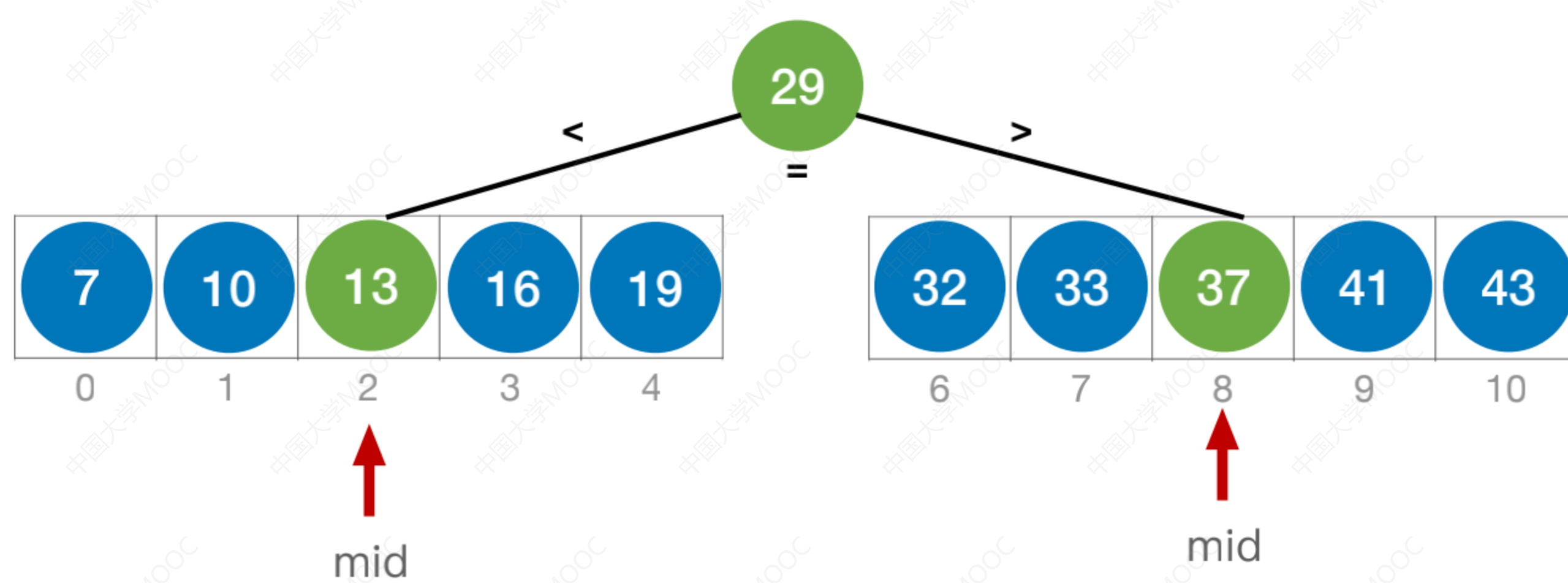
王道考研/CSKAOYAN.COM

查找效率分析



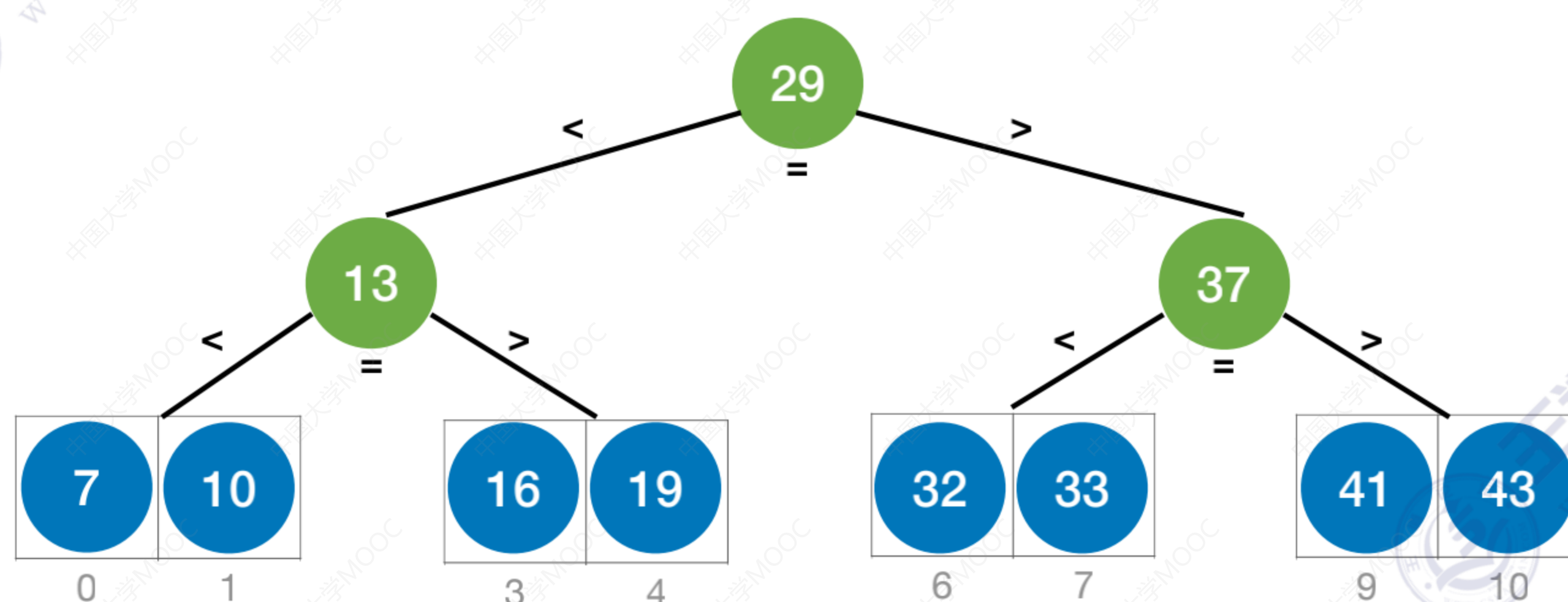
王道考研/CSKAOYAN.COM

查找效率分析



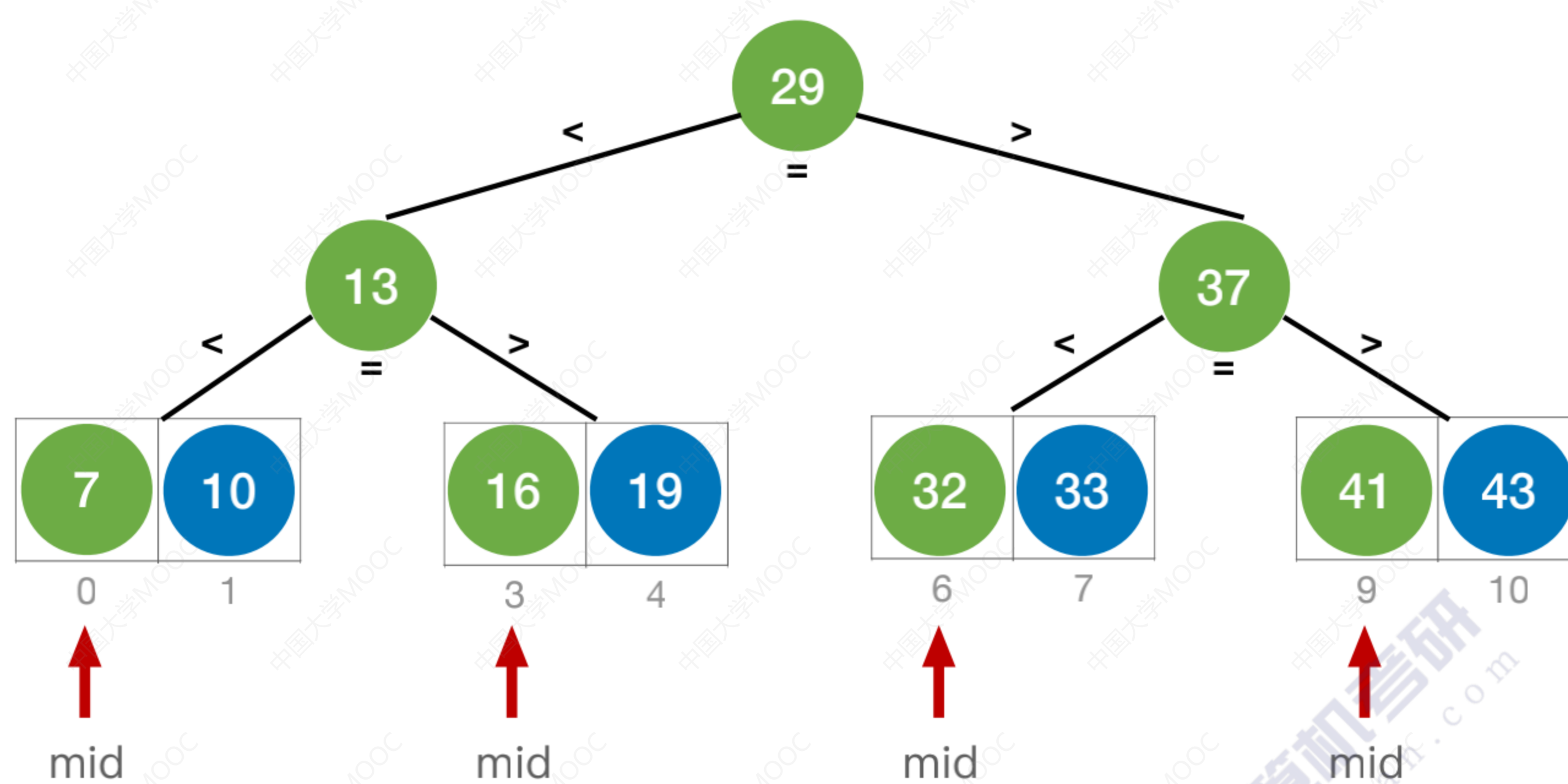
王道考研/CSKAOYAN.COM

查找效率分析



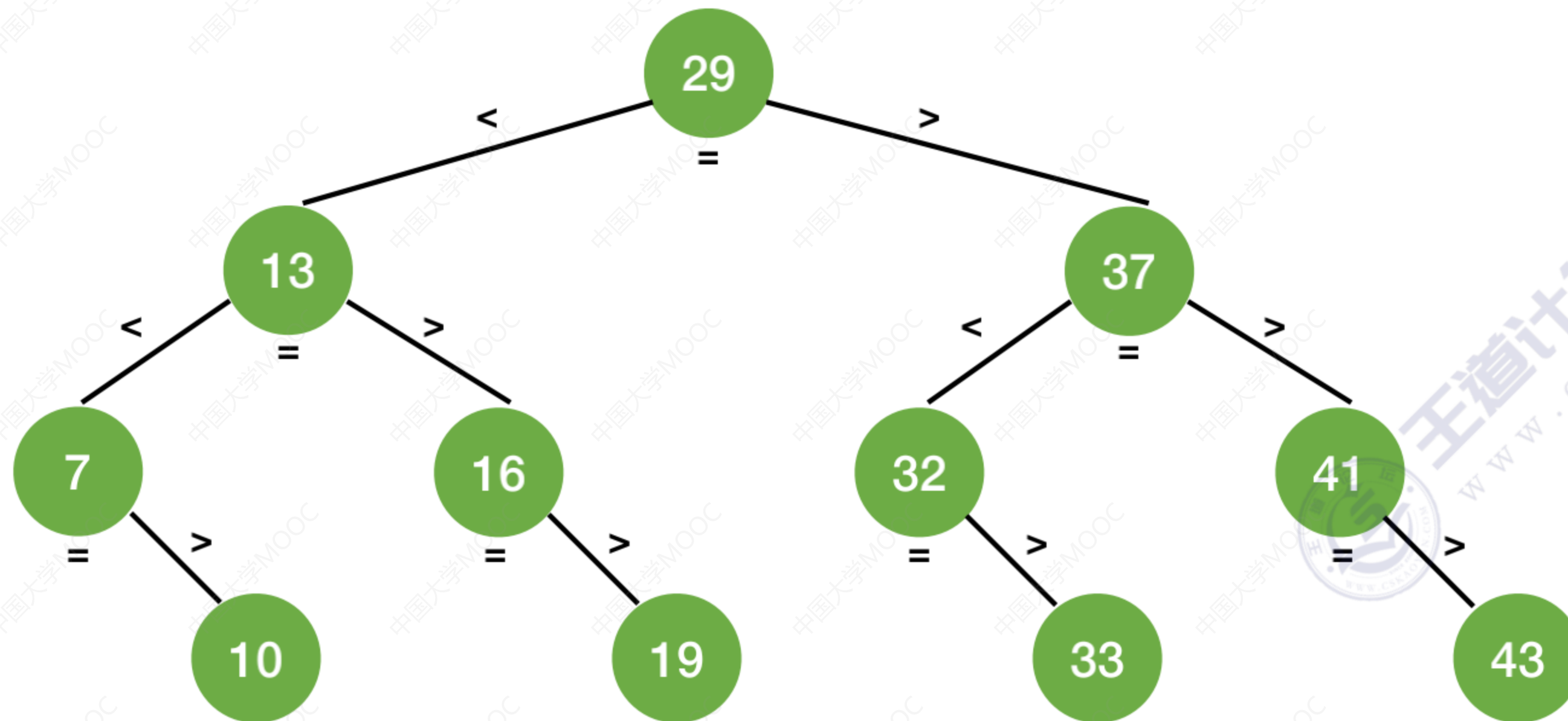
王道考研/CSKAOYAN.COM

查找效率分析



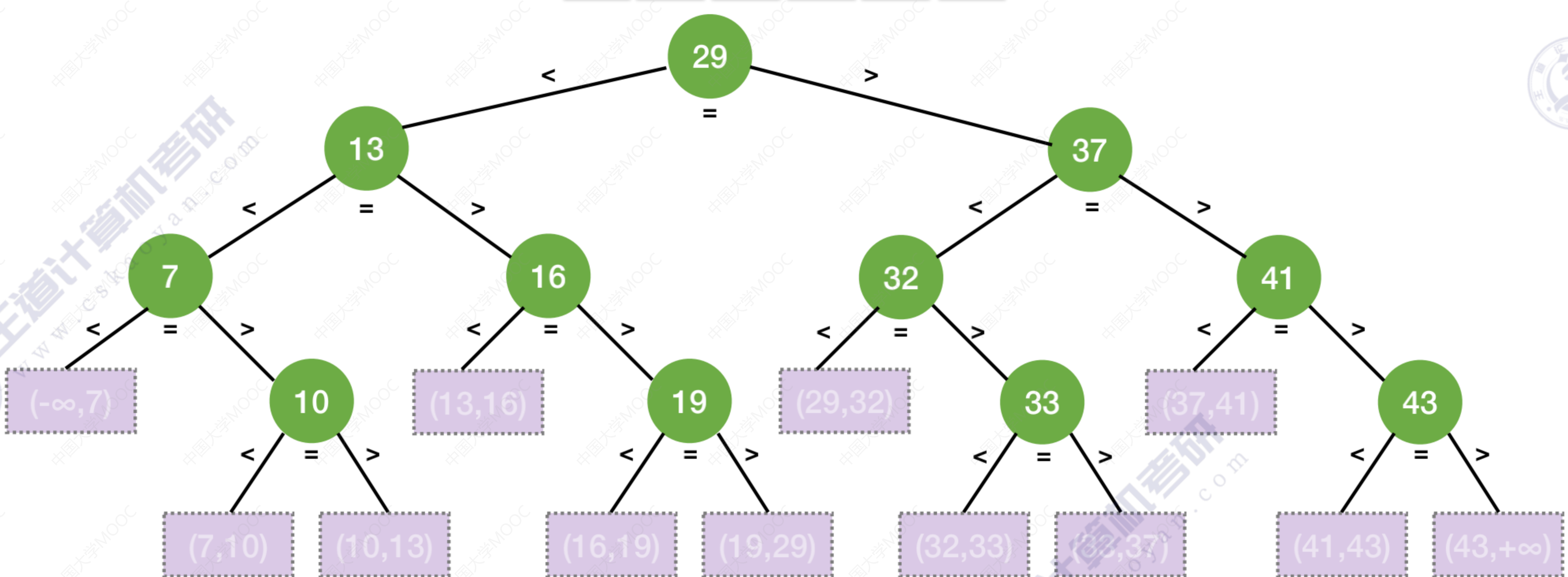
王道考研/CSKAOYAN.COM

查找效率分析



王道考研/CSKAOYAN.COM

查找效率分析

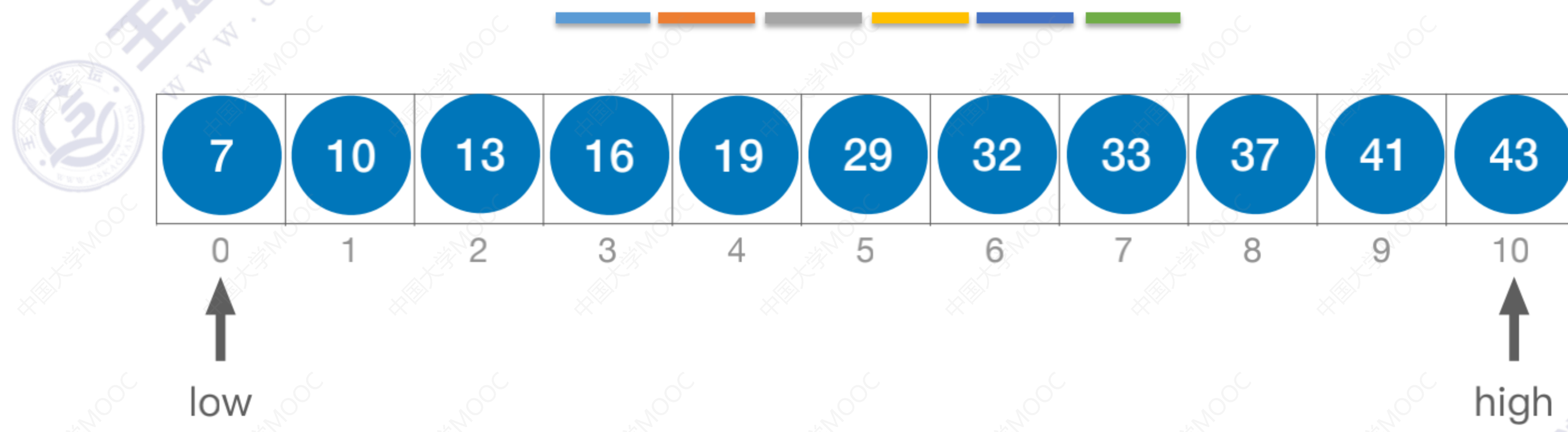


$$ASL_{成功} = (1 \cdot 1 + 2 \cdot 2 + 3 \cdot 4 + 4 \cdot 4) / 11 = 3$$

$$ASL_{失败} = (3 \cdot 4 + 4 \cdot 8) / 12 = 11/3$$

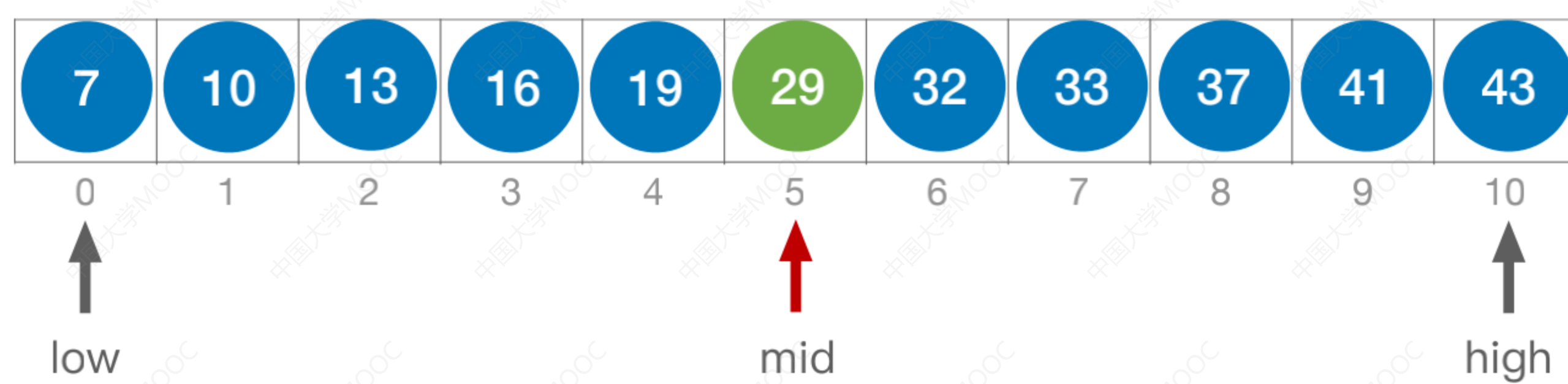
王道考研/CSKAOYAN.COM

折半查找判定树的构造



王道考研/CSKAOYAN.COM

折半查找判定树的构造

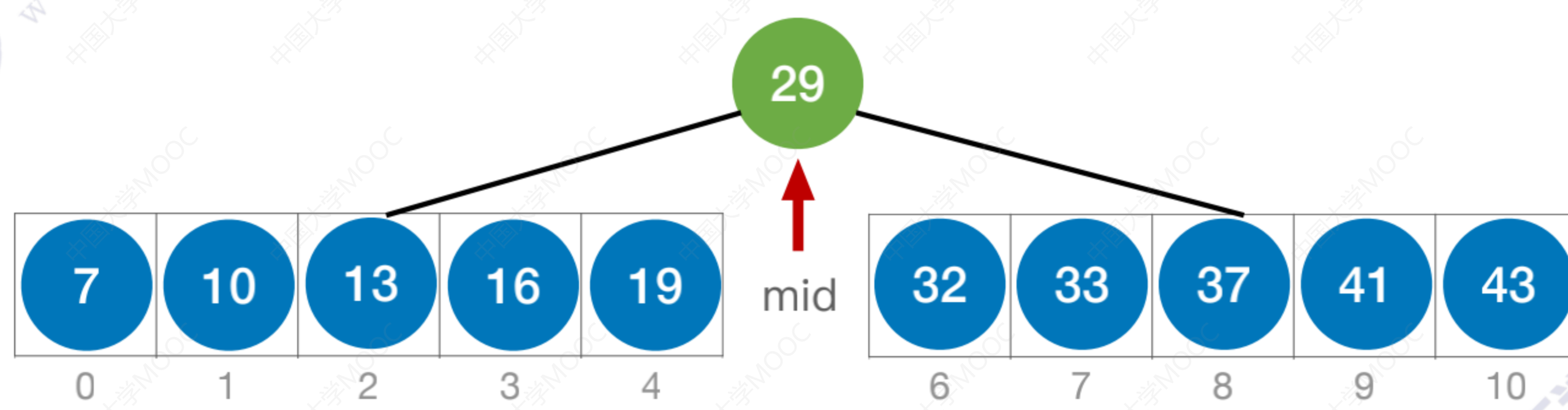


$$mid = \lfloor (low + high) / 2 \rfloor$$

如果当前low和high之间有奇数个元素，则 mid 分隔后，左右两部分元素个数相等

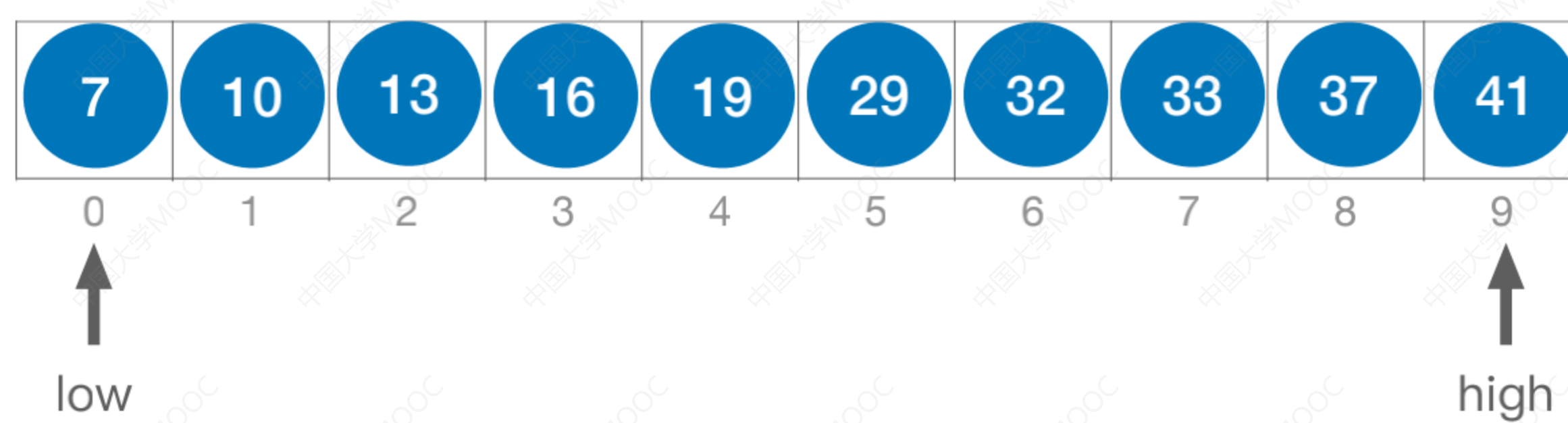
王道考研/CSKAOYAN.COM

折半查找判定树的构造

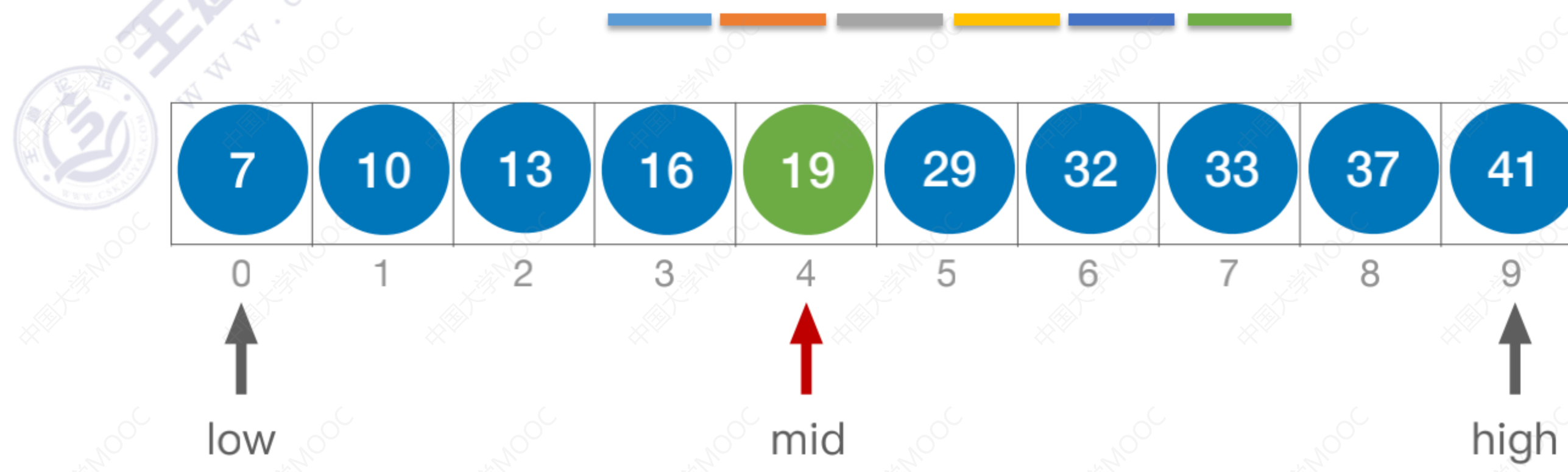


如果当前low和high之间有奇数个元素，则 mid 分隔后，左右两部分元素个数相等

折半查找判定树的构造



折半查找判定树的构造

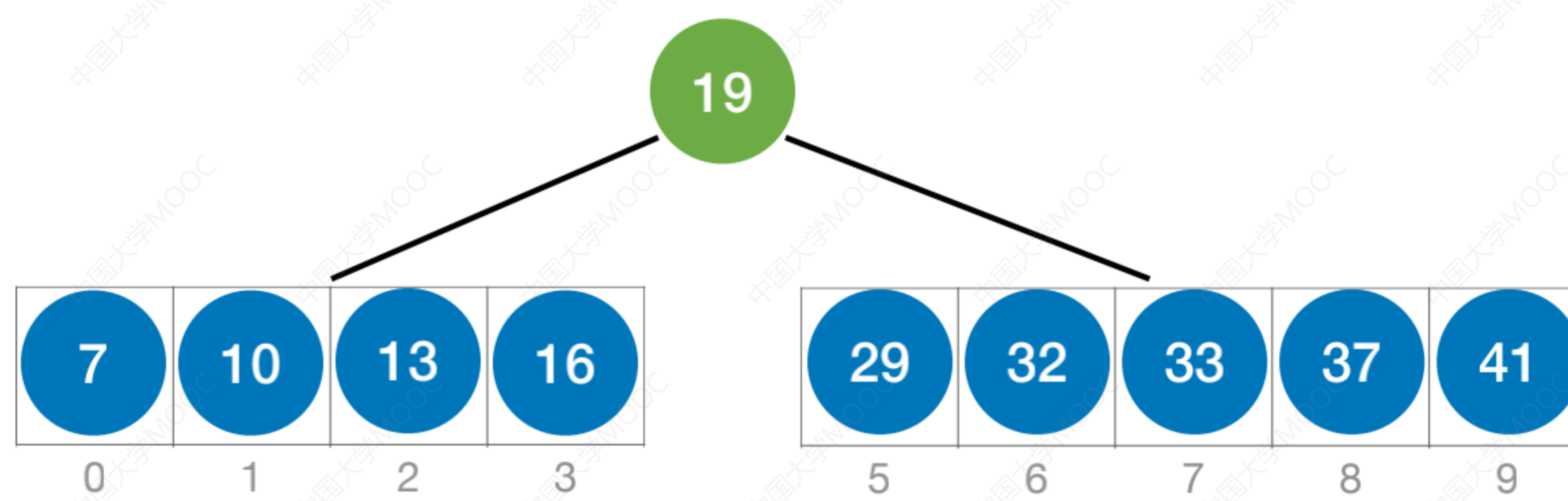


$$mid = \lfloor (low + high) / 2 \rfloor$$

如果当前low和high之间有偶数个元素，则mid分隔后，左半部分比右半部分少一个元素

王道考研/CSKAOYAN.COM

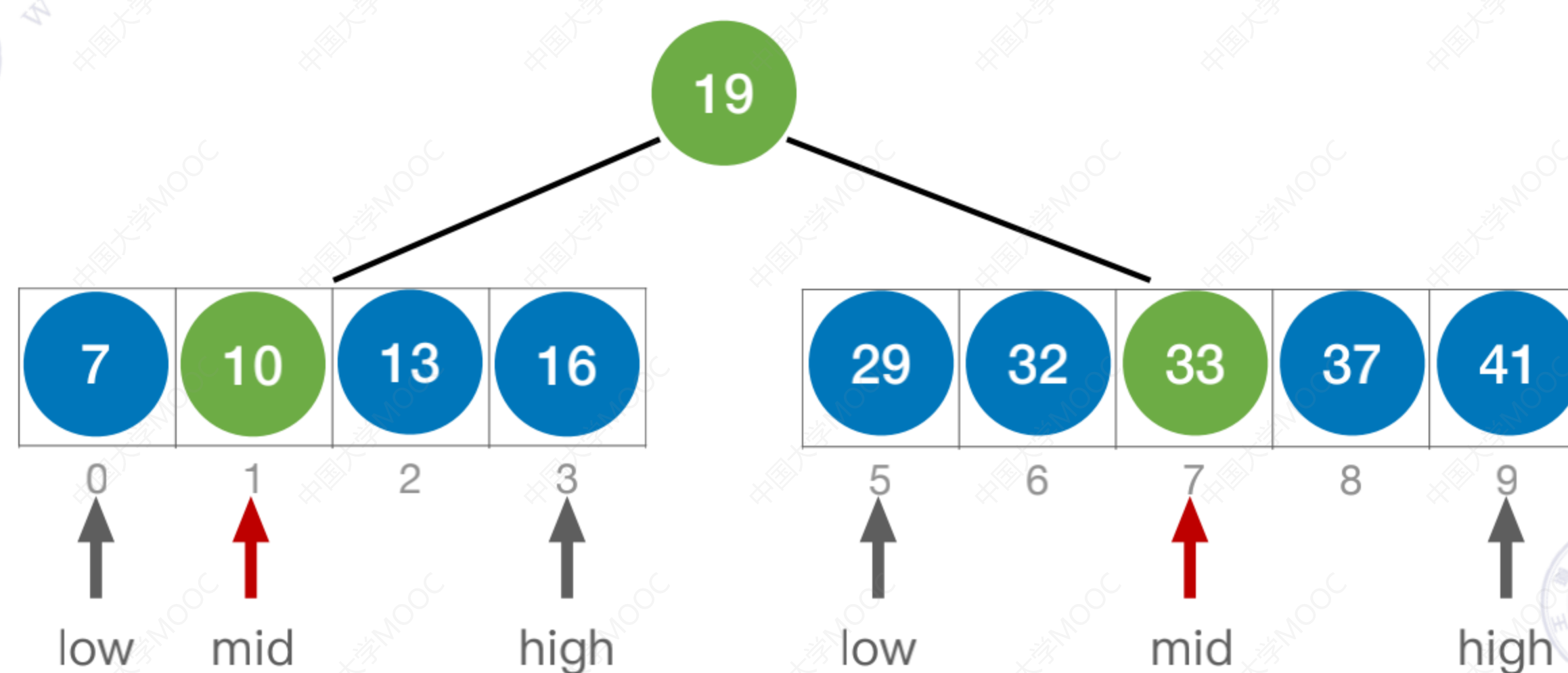
折半查找判定树的构造



如果当前low和high之间有偶数个元素，则mid分隔后，左半部分比右半部分少一个元素

王道考研/CSKAOYAN.COM

折半查找判定树的构造

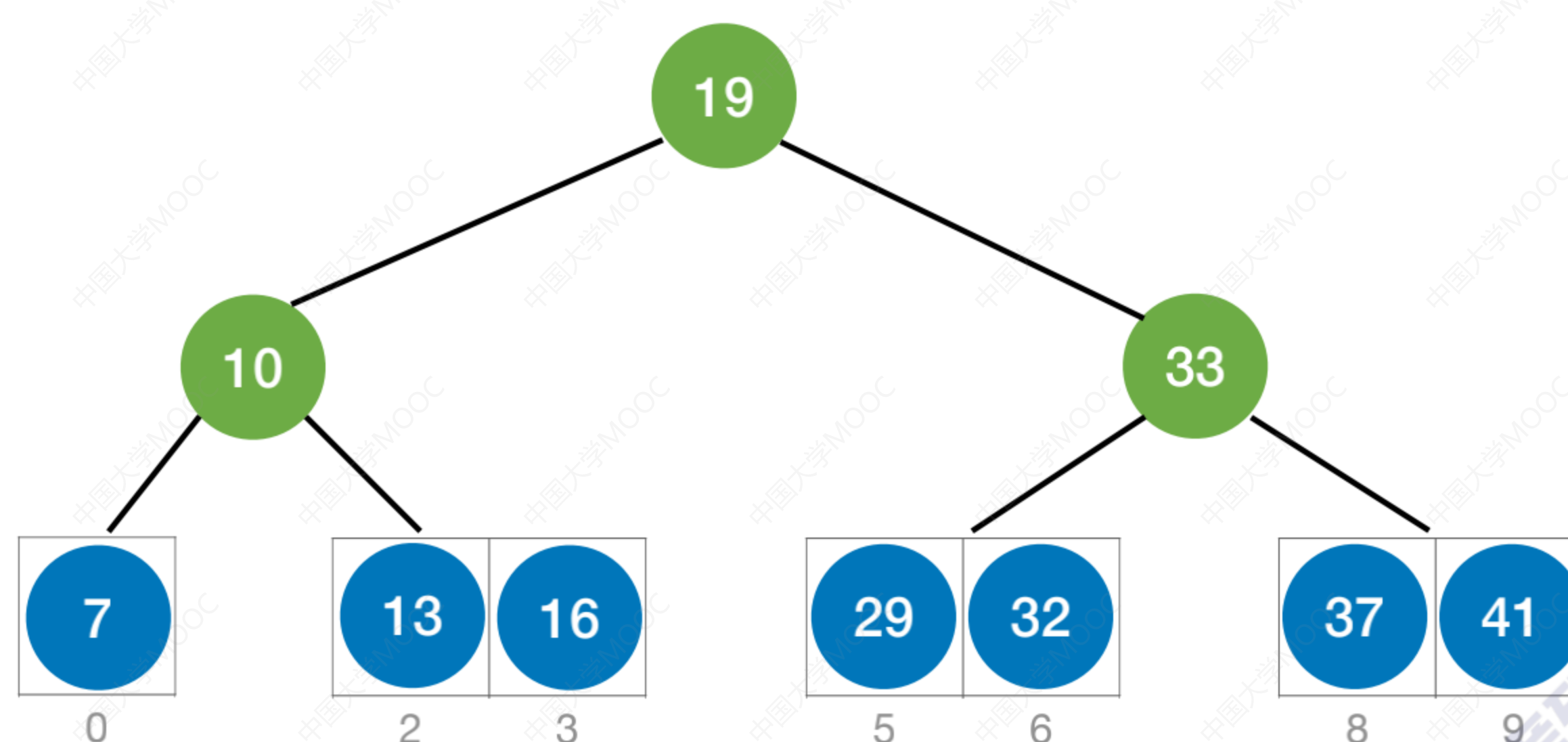


$$mid = \lfloor (low + high) / 2 \rfloor$$

如果当前low和high之间有奇数个元素，则 mid 分隔后，左右两部分元素个数相等
如果当前low和high之间有偶数个元素，则 mid 分隔后，左半部分比右半部分少一个元素

王道考研/CSKAOYAN.COM

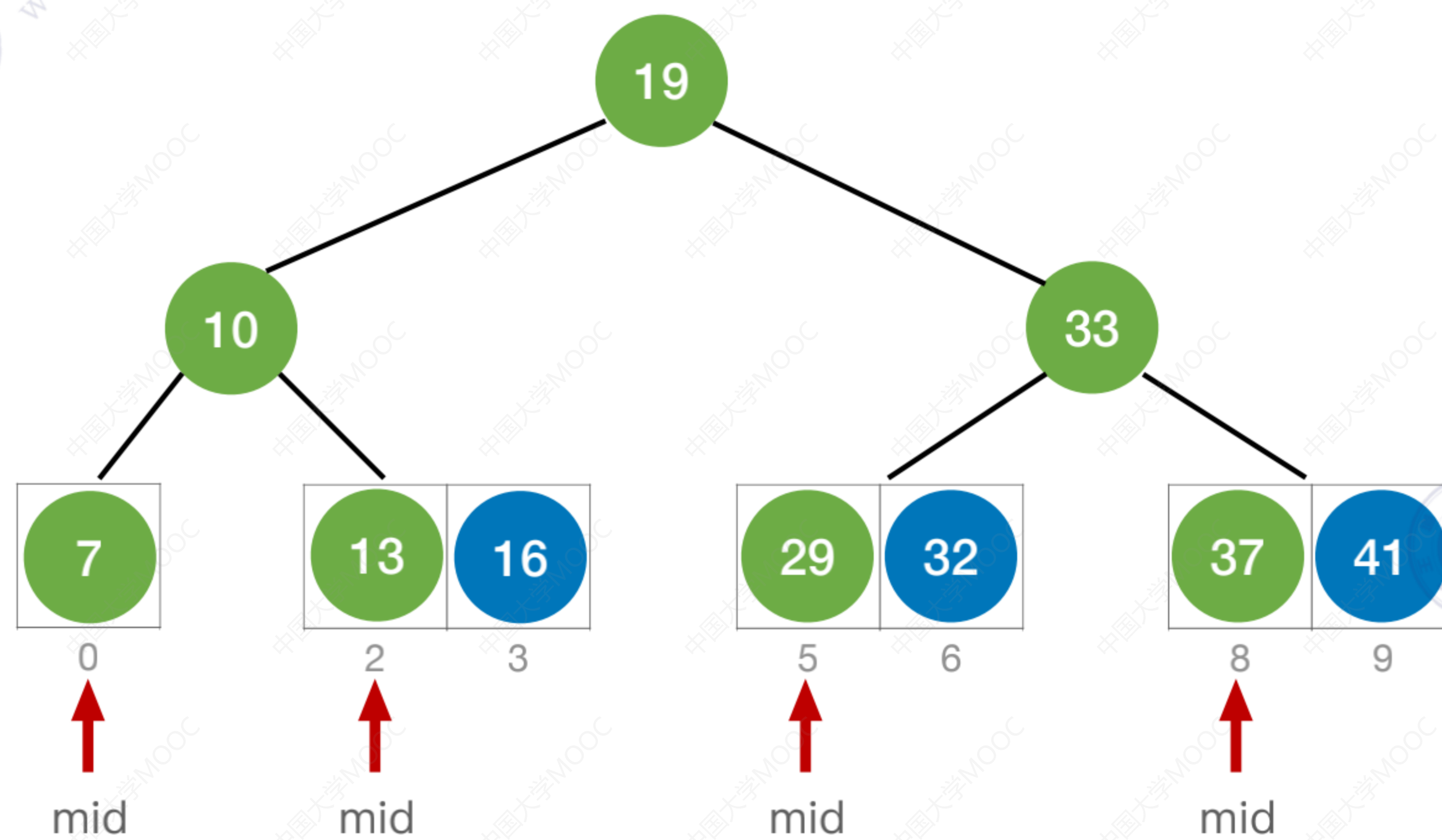
折半查找判定树的构造



如果当前low和high之间有奇数个元素，则 mid 分隔后，左右两部分元素个数相等
如果当前low和high之间有偶数个元素，则 mid 分隔后，左半部分比右半部分少一个元素

王道考研/CSKAOYAN.COM

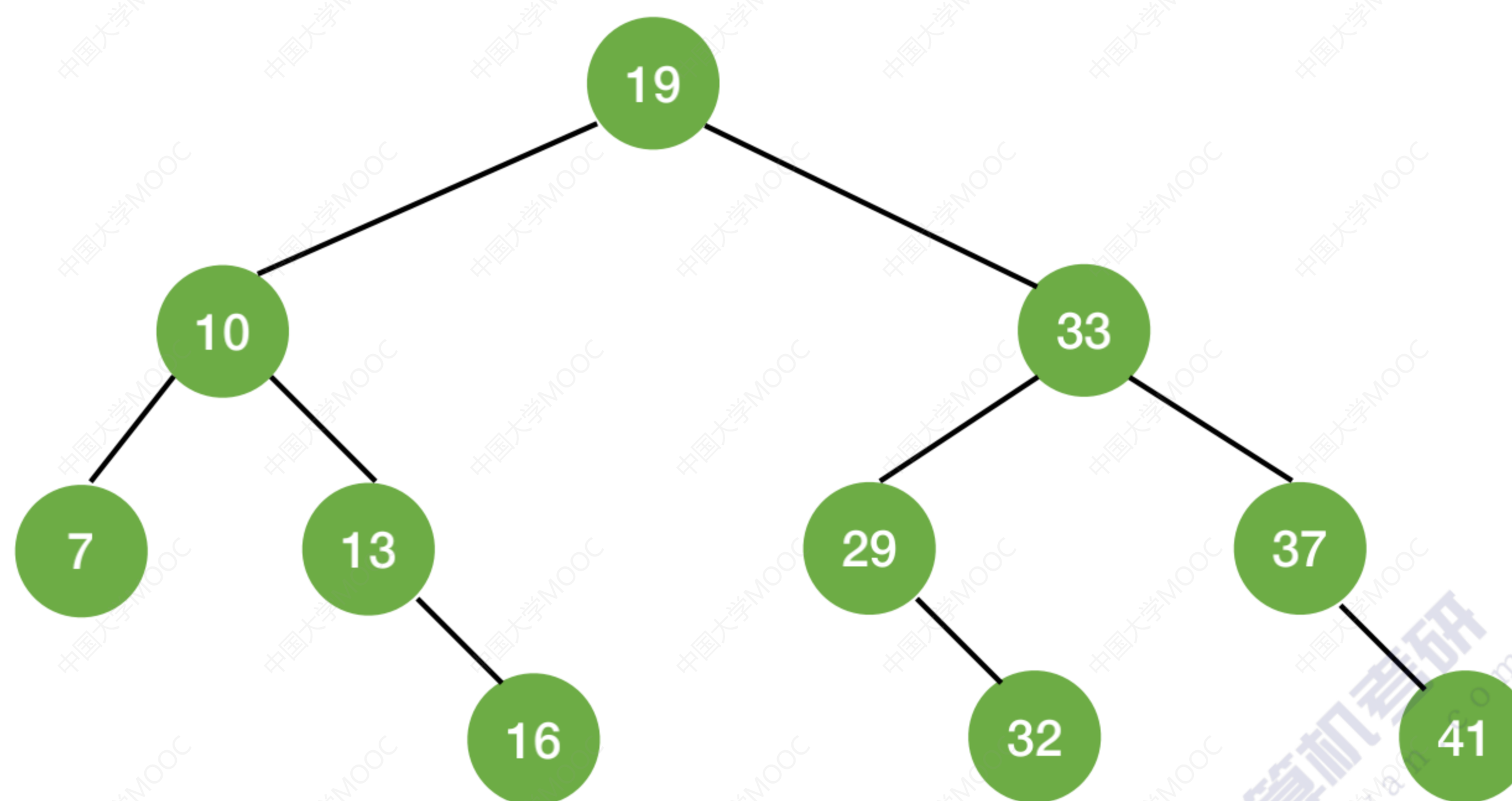
折半查找判定树的构造



如果当前low和high之间有奇数个元素，则 mid 分隔后，左右两部分元素个数相等
如果当前low和high之间有偶数个元素，则 mid 分隔后，左半部分比右半部分少一个元素

王道考研/CSKAOYAN.COM

折半查找判定树的构造



如果当前low和high之间有奇数个元素，则 mid 分隔后，左右两部分元素个数相等
如果当前low和high之间有偶数个元素，则 mid 分隔后，左半部分比右半部分少一个元素

王道考研/CSKAOYAN.COM

折半查找判定树的构造

如果当前low和high之间有奇数个元素，则mid分隔后，左右两部分元素个数相等
如果当前low和high之间有偶数个元素，则mid分隔后，左半部分比右半部分少一个元素

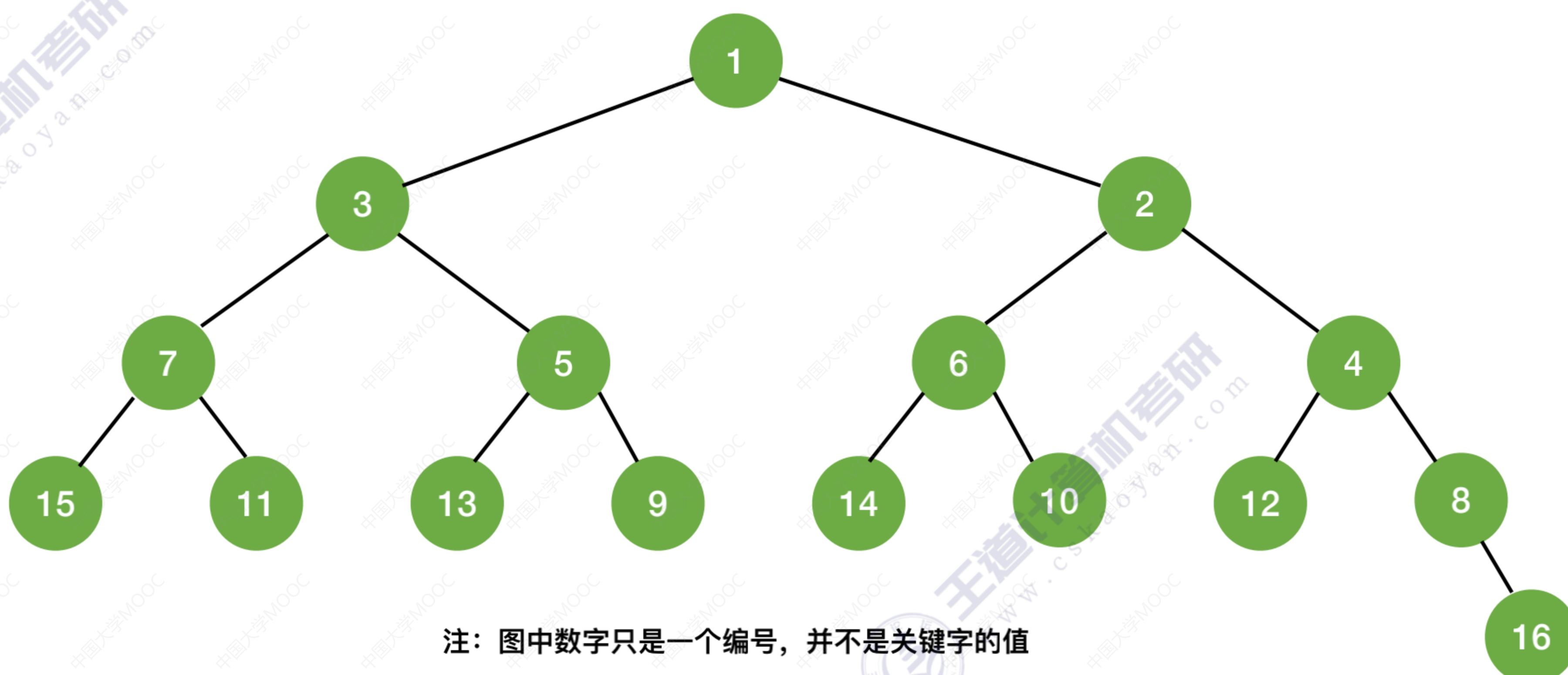


折半查找的判定树中，若 $mid = \lfloor (low + high) / 2 \rfloor$ ，则对于任何一个结点，必有：
右子树结点数-左子树结点数=0或1

王道考研/CSKAOYAN.COM

折半查找判定树的构造

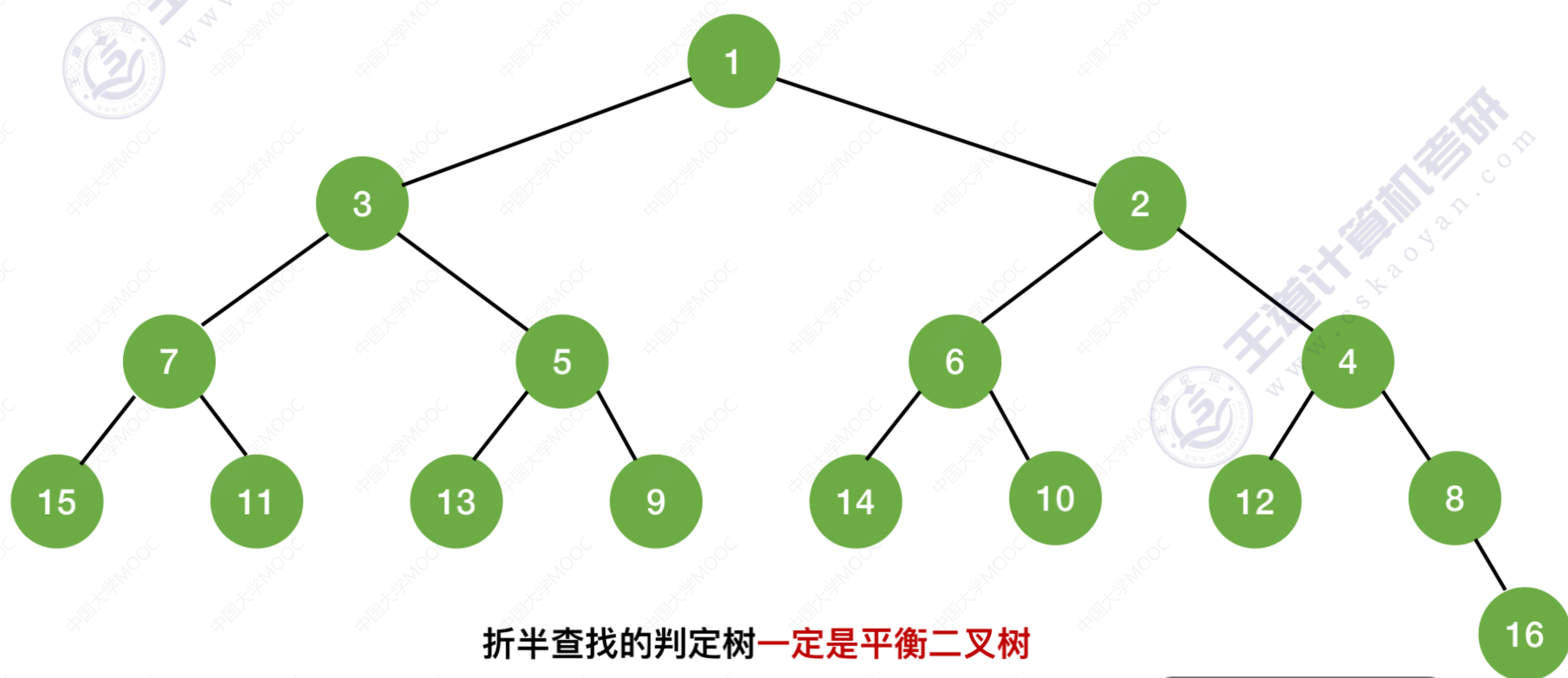
练习：若 $mid = \lfloor (low + high) / 2 \rfloor$ ，画出含1个元素、2个元素、3个元素...16个元素的查找表对应的折半查找判定树，注：暂不考虑失败结点（Key：右子树结点数-左子树结点数=0或1）



注：图中数字只是一个编号，并不是关键字的值

王道考研/CSKAOYAN.COM

折半查找判定树的构造



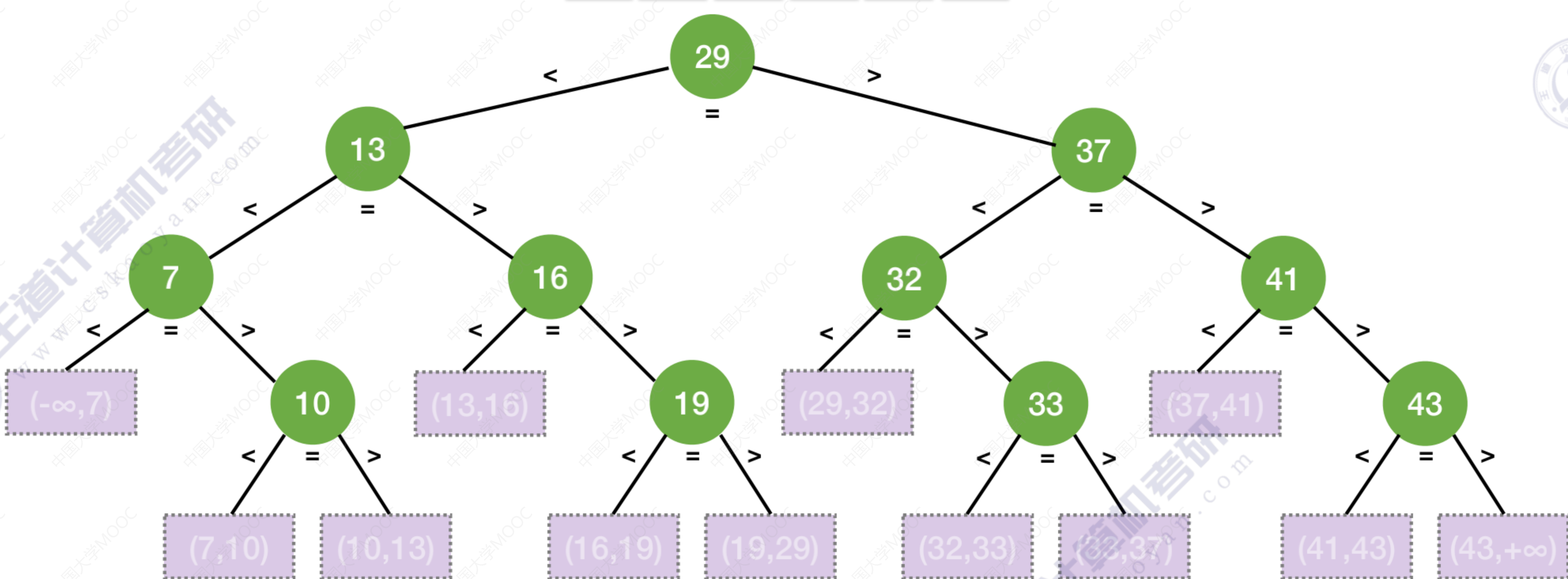
折半查找的判定树一定是平衡二叉树

折半查找的判定树中，只有最下面一层是不满的
因此，元素个数为 n 时树高 $h = \lceil \log_2(n + 1) \rceil$

注：计算方法同“完全二叉树”

王道考研/CSKAOYAN.COM

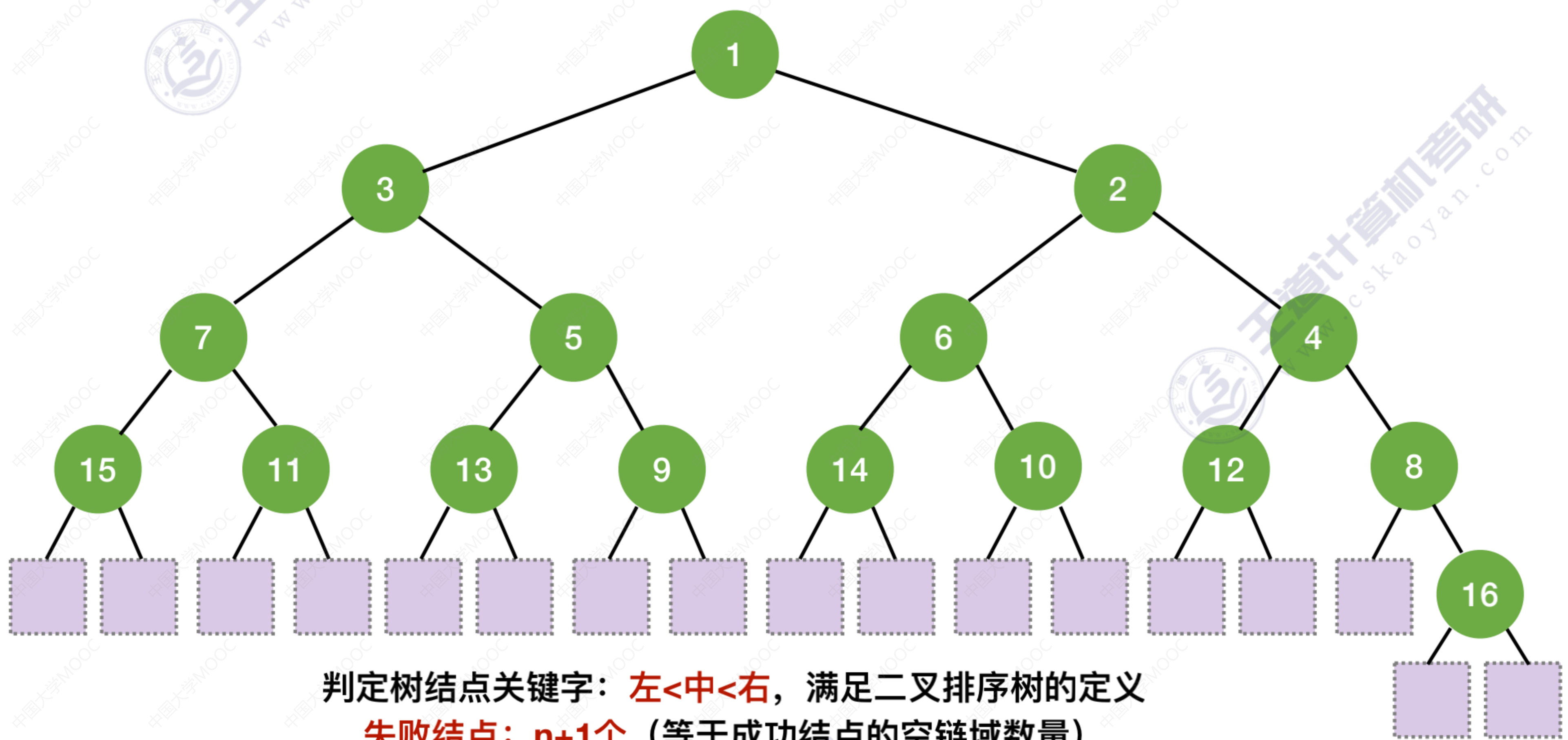
折半查找判定树的构造



判定树结点关键字：左<中<右，满足二叉排序树的定义
失败结点： $n+1$ 个（等于成功结点的空链域数量）

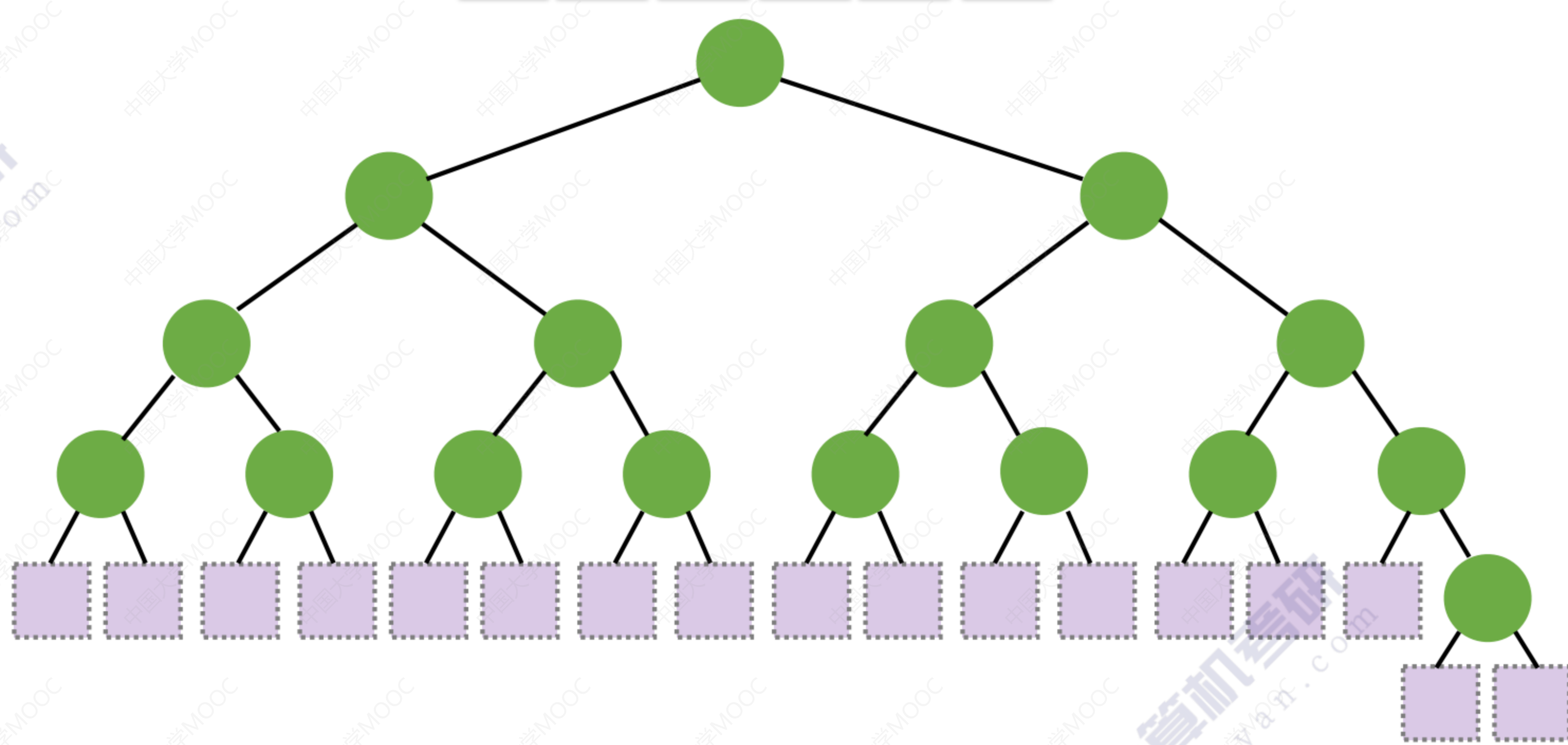
王道考研/CSKAOYAN.COM

折半查找判定树的构造



王道考研/CSKAOYAN.COM

折半查找的查找效率



树高 $h = \lceil \log_2(n + 1) \rceil$

查找成功的ASL $\leq h$

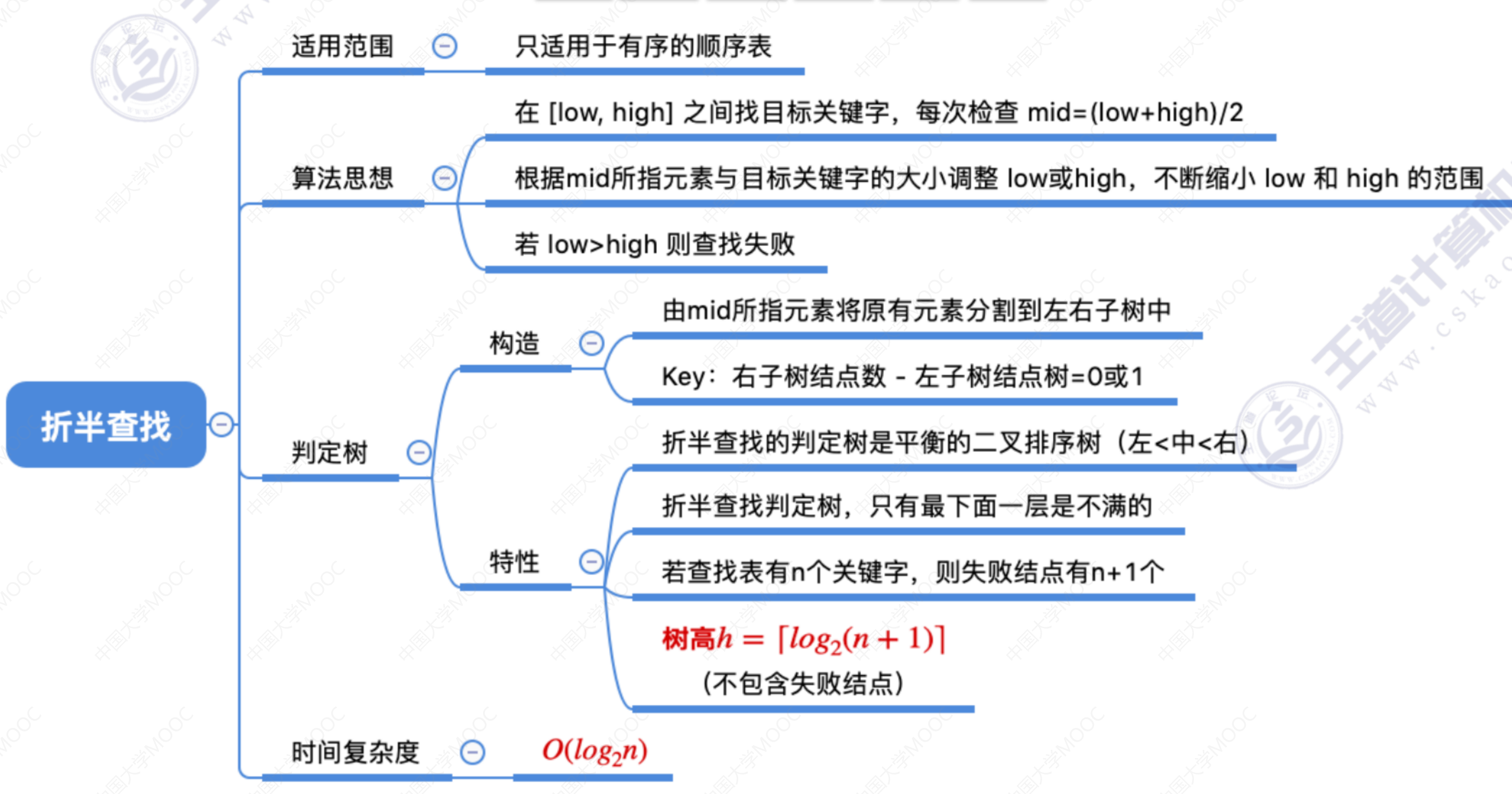
折半查找的时间复杂度 = $O(\log_2 n)$

注：该树高不包含失败结点

查找失败的ASL $\leq h$

王道考研/CSKAOYAN.COM

知识回顾与重要考点



王道考研/CSKAOYAN.COM

拓展思考



折半查找时间复杂度 = $O(\log_2 n)$
 顺序查找的时间复杂度 = $O(n)$

辣么，折半查找的速度一定比顺序查找更快？

TableLen=11

查找目标: 7

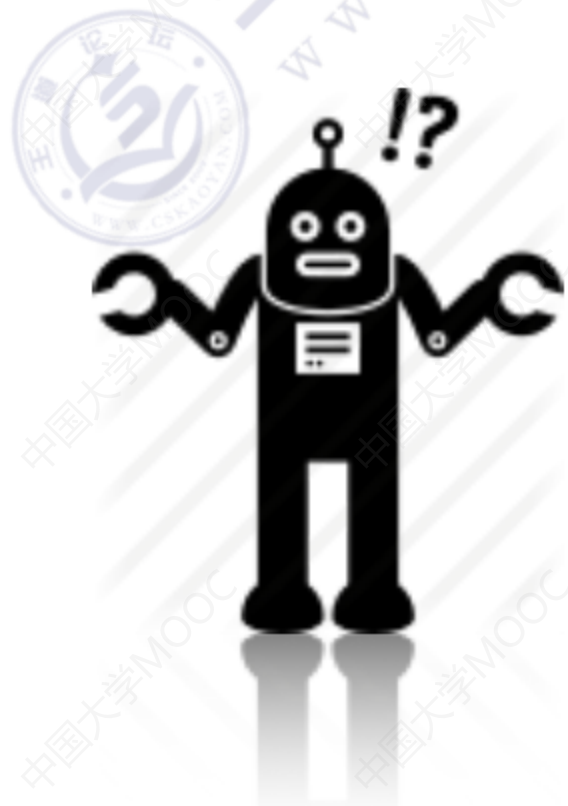
7	10	13	16	19	29	32	33	37	41	43				
0	1	2	3	4	5	6	7	8	9	10	11	12	13	...

否认三连

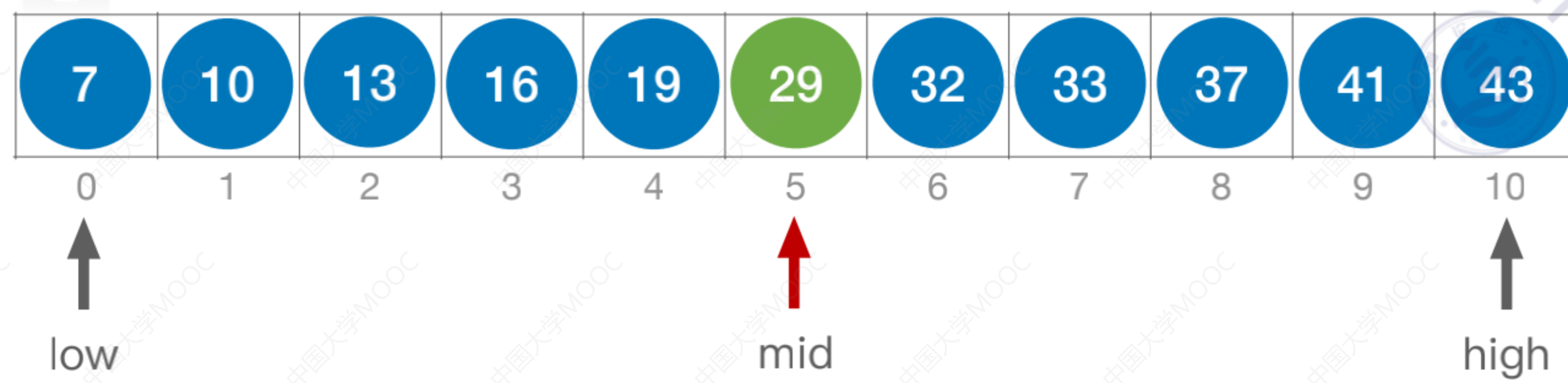


王道考研/CSKAOYAN.COM

拓展思考



如果 $mid = \lfloor (low + high) / 2 \rfloor$
辣么，判定树是什么亚子？

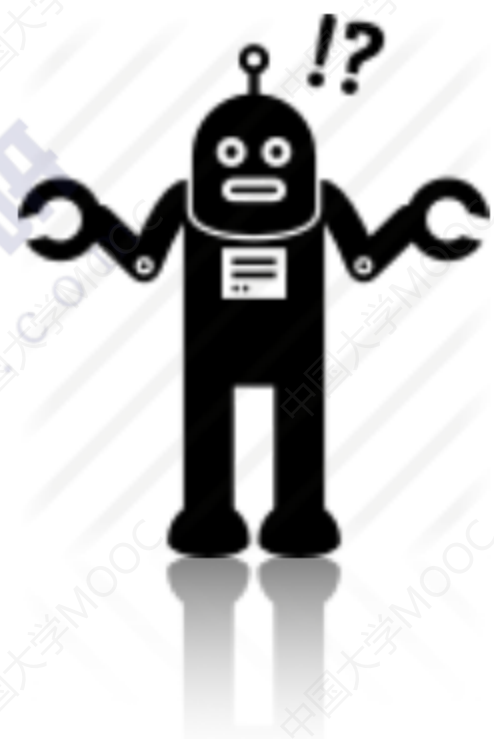


$$mid = \lfloor (low + high) / 2 \rfloor$$

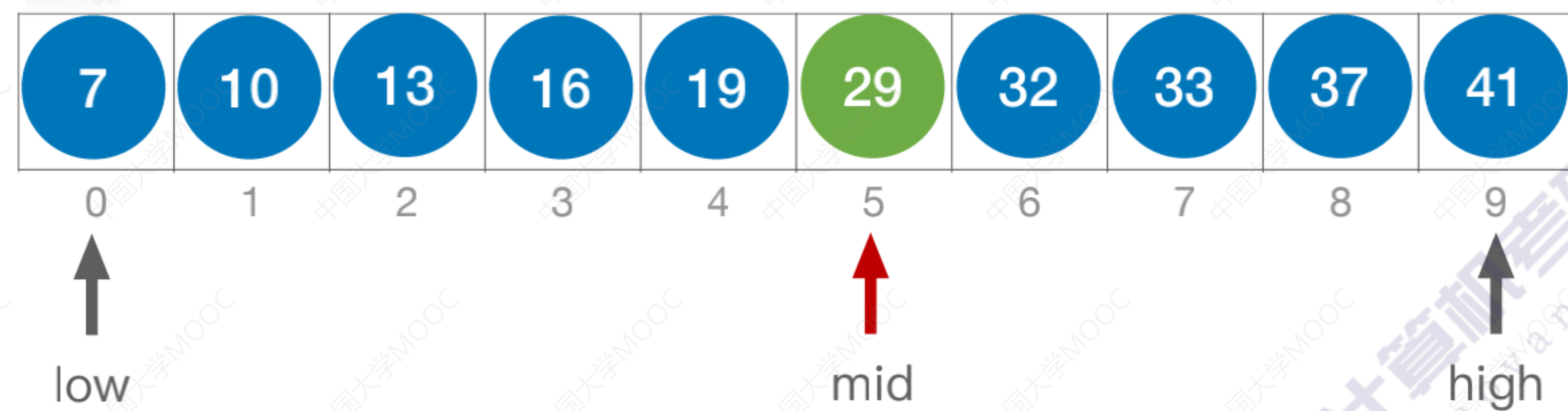
如果当前low和high之间有奇数个元素，则 mid 分隔后，左右两部分元素个数相等

王道考研/CSKAOYAN.COM

拓展思考



如果 $mid = \lfloor (low + high) / 2 \rfloor$
辣么，判定树是什么亚子？



$$mid = \lfloor (low + high) / 2 \rfloor$$

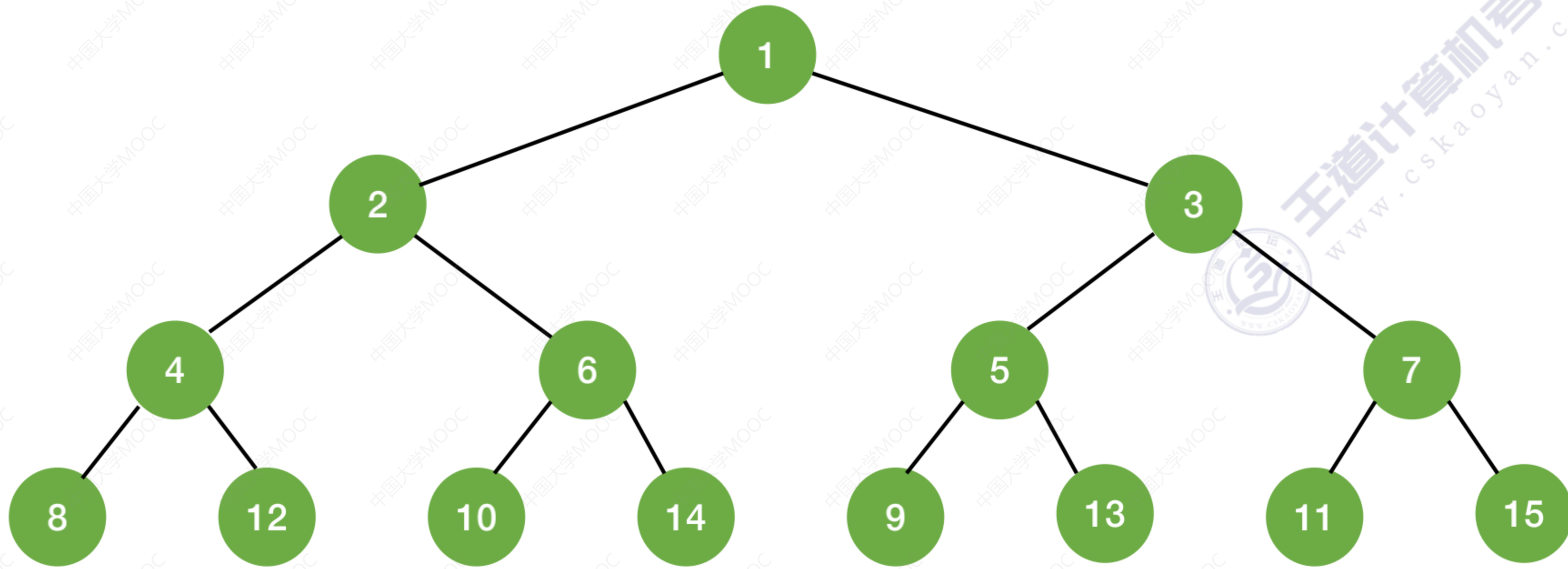
如果当前low和high之间有偶数个元素，则 mid 分隔后，左半部分比右半部分多一个元素

王道考研/CSKAOYAN.COM

拓展思考



折半查找的判定树中，若 $mid = \lceil (low + high)/2 \rceil$ ，则对于任何一个结点，必有：
左子树结点数-右子树结点数=0或1



注：图中数字只是一个编号，并不是关键字的值

王道考研/CSKAOYAN.COM



@王道论坛



@王道计算机考研备考



等撩

@王道咸鱼老师-计算机考研

@王道楼楼老师-计算机考研



等撩

@王道计算机考研



知乎

@王道计算机考研

微信视频号

@王道计算机考研

微信公众平台

@王道在线