

慕课网《玩转算法面试》

玩儿转算法面试

讲师：liuyubobobo

版权所有，侵权必究

liuyubobobo

面试中的时间复杂度分析

慕课网《玩转算法面试》

讲师：liuyubobobo

版权所有，侵权必究

慕课网《玩转算法面试》

时间复杂度

讲师：liuyubobobo

版权所有，侵权必究

到底什么是大O

n表示数据规模

$O(f(n))$ 表示运行算法所需要执行的指令数，和 $f(n)$ 成正比。

到底什么是Big O

二分查找法 $O(\log n)$

所需执行指令数: $a \cdot \log n$

寻找数组中的最大/最小值 $O(n)$

所需执行指令数: $b \cdot n$

归并排序算法 $O(n \log n)$

所需指令数: $c \cdot n \log n$

选择排序法 $O(n^2)$

所需指令数: $d \cdot n^2$

到底什么是Big O

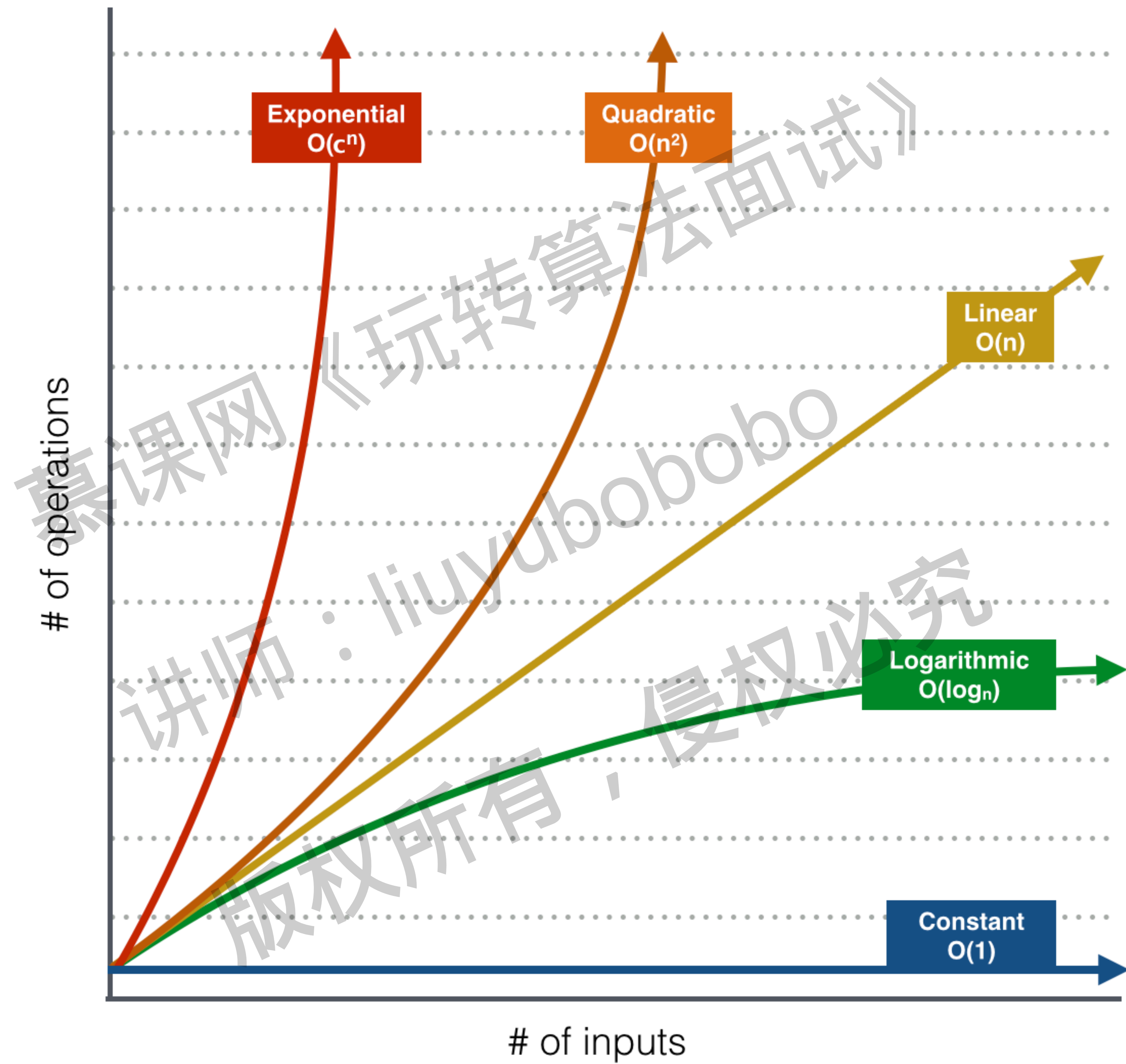
算法A: $O(n)$

所需执行指令数: $10000 \cdot n$

算法B: $O(n^2)$

所需执行指令数: $10 \cdot n^2$

n	A指令数 $10000n$	B指令数 $10n^2$	倍数
10	10^5	10^3	100
100	10^6	10^5	10
1000	10^7	10^7	1
10000	10^8	10^9	0.1
10^5	10^9	10^{11}	0.01
10^6	10^{10}	10^{13}	0.001



到底什么是大O

在学术界，严格地讲， $O(f(n))$ 表示算法执行的上界

归并排序算法的时间复杂度是 $O(n \log n)$ 的，同时也是 $O(n^2)$

在业界，我们就使用O来表示算法执行的最低上界

我们一般不会说归并排序是 $O(n^2)$ 的

到底什么是大O

$$O(n \log n + n) = O(n \log n)$$

$$O(n \log n + n^2) = O(n^2)$$

到底什么是大O

$$O(A \log A + B)$$

$$O(A \log A + B^2)$$

对邻接表实现的图进行遍历：

$$\text{时间复杂度： } O(V + E)$$

一个时间复杂度的问题

有一个字符串数组，将数组中的每一个字符串按照字母序排序；之后再将整个字符串数组按照字典序排序。整个操作的时间复杂度？

$$O(n * n \log n + n \log n) = O(n^2 \log n)$$

一个时间复杂度的问题

有一个字符串数组，将数组中的每一个字符串按照字母序排序；之后再将整个字符串数组按照字典序排序。整个操作的时间复杂度？

假设最长的字符串长度为 s ；数组中有 n 个字符串

对每个字符串排序： $O(s \log s)$

将数组中的每一个字符串按照字母序排序： $O(n * s \log(s))$

将整个字符串数组按照字典序排序： $O(s * n \log(n))$

一个时间复杂度的问题

有一个字符串数组，将数组中的每一个字符串按照字母序排序；之后再将整个字符串数组按照字典序排序。整个操作的时间复杂度？

$$\begin{aligned} O(n * s \log(s)) + O(s * n \log(n)) &= O(n * s * \log s + s * n * \log n) \\ &= O(n * s * (\log s + \log n)) \end{aligned}$$

算法复杂度在有些情况是用例相关的

插入排序算法 $O(n^2)$

最差情况: $O(n^2)$

最好情况: $O(n)$

平均情况: $O(n^2)$

快速排序算法 $O(n \log n)$

最差情况: $O(n^2)$

最好情况: $O(n \log n)$

平均情况: $O(n \log n)$

慕课网《玩转算法面试》

对数据规模有一个概念

讲师：lilysoobobo

版权所有，侵权必究

数据规模的概念

对 10^5 的数据进行选择排序，结果计算机假死？

数据规模的概念

如果要想在1s之内解决问题：

$O(n^2)$ 的算法可以处理大约 10^4 级别的数据；

$O(n)$ 的算法可以处理大约 10^8 级别的数据；

$O(n\log n)$ 的算法可以处理大约 10^7 级别的数据

空间复杂度

多开一个辅助的数组： $O(n)$

多开一个辅助的二维数组： $O(n^2)$

多开常数空间： $O(1)$

空间复杂度

递归调用是有空间代价的

空间复杂度 $O(1)$

```
int sum1( int n ){  
    assert( n >= 0 );  
    int ret = 0;  
    for( int i = 0 ; i <= n ; i ++ )  
        ret += i;  
    return ret;  
}
```

空间复杂度 $O(n)$

```
int sum2( int n ){  
    assert( n >= 0 );  
    if( n == 0 )  
        return 0;  
    return n + sum2(n-1);  
}
```

慕课网《玩转算法面试》

常见的复杂度分析

讲师：lucybobobo

版权所有，侵权必究

$O(1)$

```
void swapTwoInts( int &a, int &b ){  
    int temp = a;  
    a = b;  
    b = temp;  
}
```

$O(n)$

```
int sum( int n ){  
    int ret = 0;  
    for( int i = 0 ; i <= n ; i++ )  
        ret += i;  
    return ret;  
}
```

$O(n)$

```
void reverse( string &s ){  
    int n = s.size();  
    for( int i = 0 ; i < n/2 ; i ++ )  
        swap( s[i] , s[n-1-i] );  
}
```

1/2*n次swap操作: $O(n)$

$O(n^2)$

```
void selectionSort(int arr[], int n){  
    for(int i = 0 ; i < n ; i ++){  
        int minIndex = i;  
        for( int j = i + 1 ; j < n ; j ++ )  
            if( arr[j] < arr[minIndex] )  
                minIndex = j;  
        swap( arr[i] , arr[minIndex] );  
    }  
}
```

$(n-1) + (n-2) + (n-3) + \dots + 0$
 $= (0+n-1)*n/2$
 $= (1/2)n*(n-1)$
 $= 1/2*n^2 - 1/2*n$
 $= O(n^2)$

$O(n^2)$?

```
void printInformation(int n){  
    for( int i = 1 ; i <= n ; i ++ )  
        for( int j = 1 ; j <= 30 ; j ++ )  
            cout<<"Class " <<i<<" - " <<"No. " <<j<<endl;  
    return;  
}
```

30n次基本操作: $O(n)$

$O(\log n)$

```
int binarySearch(int arr[], int n, int target){  
    int l = 0, r = n-1;  
    while( l <= r ){  
        int mid = l + (r-l)/2;  
        if( arr[mid] == target ) return mid;  
        if( arr[mid] > target ) r = mid - 1;  
        else l = mid + 1;  
    }  
    return -1;  
}
```


$O(\log n)$

二分查找法的时间复杂度是 $O(\log n)$ 的

在 n 个元素中寻找

在 $n/2$ 个元素中寻找

在 $n/4$ 个元素中寻找

.....

在 1 个元素中寻找

n 经过几次“除以2”操作后，等于1？

$\log_2 n = O(\log n)$

$O(\log n)$

```
string intToString( int num ){  
    string s = "";  
    while( num ){  
        s += '0' + num%10;  
        num /= 10;  
    }  
  
    reverse(s);  
    return s;  
}
```

n经过几次“除以10”操作
后，等于0？

$\log_{10} n = O(\log n)$

$O(\log n)$

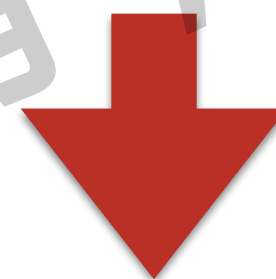
$\log_a N$

$\log_b N$



$$\log_a N = \log_a b * \log_b N$$

(常数)



$O(\log n)$

整形转成字符串

```
string intToString( int num ){  
    string s = "";  
    while( num ){  
        s += '0' + num%10;  
        num /= 10;  
    }  
    reverse(s);  
    return s;  
}
```

$O(n^2)$?

```
void hello(int n){  
    for( int sz = 1 ; sz < n ; sz += sz )  
        for( int i = 1 ; i < n ; i ++ )  
            cout<<"Hello, Algorithm!"<<endl;  
}
```

$O(\sqrt{n})$

```
bool isPrime( int n ){  
    for( int x = 2 ; x*x <= n ; x ++ )  
        if( n%x == 0 )  
            return false;  
    return true;  
}
```


慕课网《玩转算法面试》

复杂度实验

讲师：huayupobobo

版权所有，侵权必究

复杂度试验

我们自以为写出了一个 $O(n \log n)$ 的算法，但实际是 $O(n^2)$ 的算法？

数据规模的概念

如果要想在1s之内解决问题：

$O(n^2)$ 的算法可以处理大约 10^4 级别的数据；

$O(n)$ 的算法可以处理大约 10^8 级别的数据；

$O(n\log n)$ 的算法可以处理大约 10^7 级别的数据

复杂度试验

实验，观察趋势

每次将数据规模提高两倍，看时间的变化

慕课网《玩转算法面试》
讲师：lily0909bobo
版权所有，侵权必究

实践：复杂度实验 $O(n)$, $O(n^2)$

复杂度试验

$$\log_2 N / \log N$$

$$= (\log_2 + \log N) / \log N$$

$$= 1 + \log_2 / \log N$$

实践：复杂度实验 $O(\log n)$, $O(n \log n)$

慕课网《玩转算法面试》

递归算法的复杂度分析

讲师：liuyubobobo

版权所有，侵权必究

递归算法的复杂度分析

不是有递归的函数就一定是 $O(n\log n)$!

递归中进行一次递归调用的复杂度分析

```
int binarySearch(int arr[], int l, int r, int target){  
    if( l > r )  
        return -1;  
  
    int mid = l + (r-l)/2;  
    if( arr[mid] == target )  
        return mid;  
    else if( arr[mid] > target )  
        return binarySearch(arr, l, mid-1, target);  
    else  
        return binarySearch(arr, mid+1, r, target);  
}
```

时间复杂度： $O(\log n)$

递归中进行一次递归调用的复杂度分析

如果递归函数中，只进行一次递归调用，

递归深度为depth;

在每个递归函数中，时间复杂度为T;

则总体的时间复杂度为 $O(T * \text{depth})$

递归中进行一次递归调用的复杂度分析

```
int sum( int n ){  
    assert( n >= 0 );  
  
    if( n == 0 )  
        return 0;  
    return n + sum(n-1);  
}
```

递归深度: n

时间复杂度: $O(n)$

递归中进行一次递归调用的复杂度分析

```
double pow( double x, int n ){  
    assert( n >= 0 );  
  
    if( n == 0 )  
        return 1.0;  
  
    double t = pow(x, n/2);  
    if( n%2 )  
        return x*t*t;  
  
    return t*t;  
}
```

递归深度: $\log n$

时间复杂度: $O(\log n)$

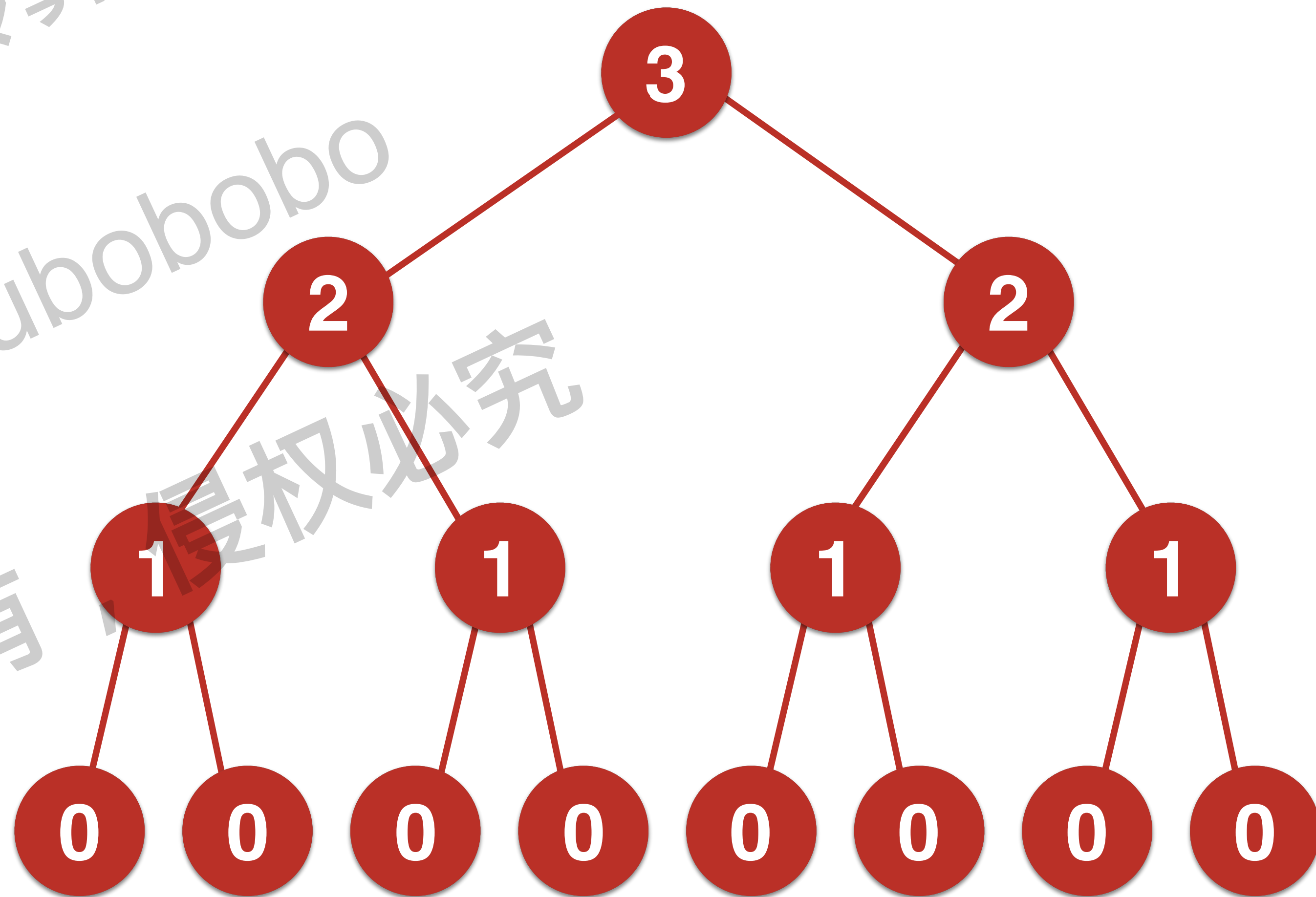
递归中进行多次递归调用

```
int f(int n){  
    assert( n >= 0 );  
  
    if( n == 0 )  
        return 1;  
  
    return f(n-1) + f(n-1);  
}
```

计算调用的次数

递归中进行多次递归调用

```
int f(int n){  
    assert( n >= 0 );  
  
    if( n == 0 )  
        return 1;  
  
    return f(n-1) + f(n-1);  
}
```



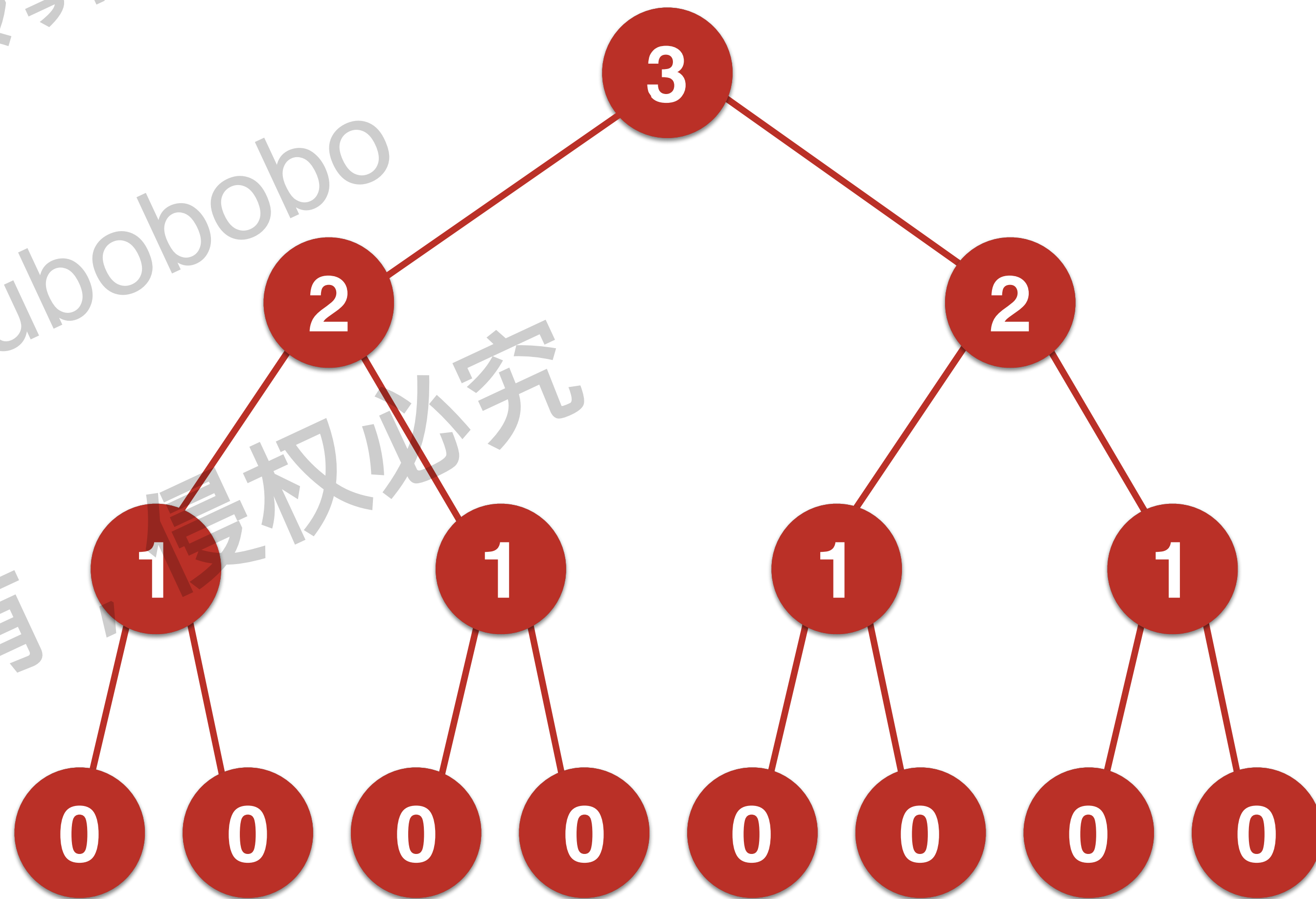
递归中进行多次递归调用

$$1 + 2 + 4 + 8 = 15$$

$$2^0 + 2^1 + 2^2 + 2^3 + \dots + 2^n$$

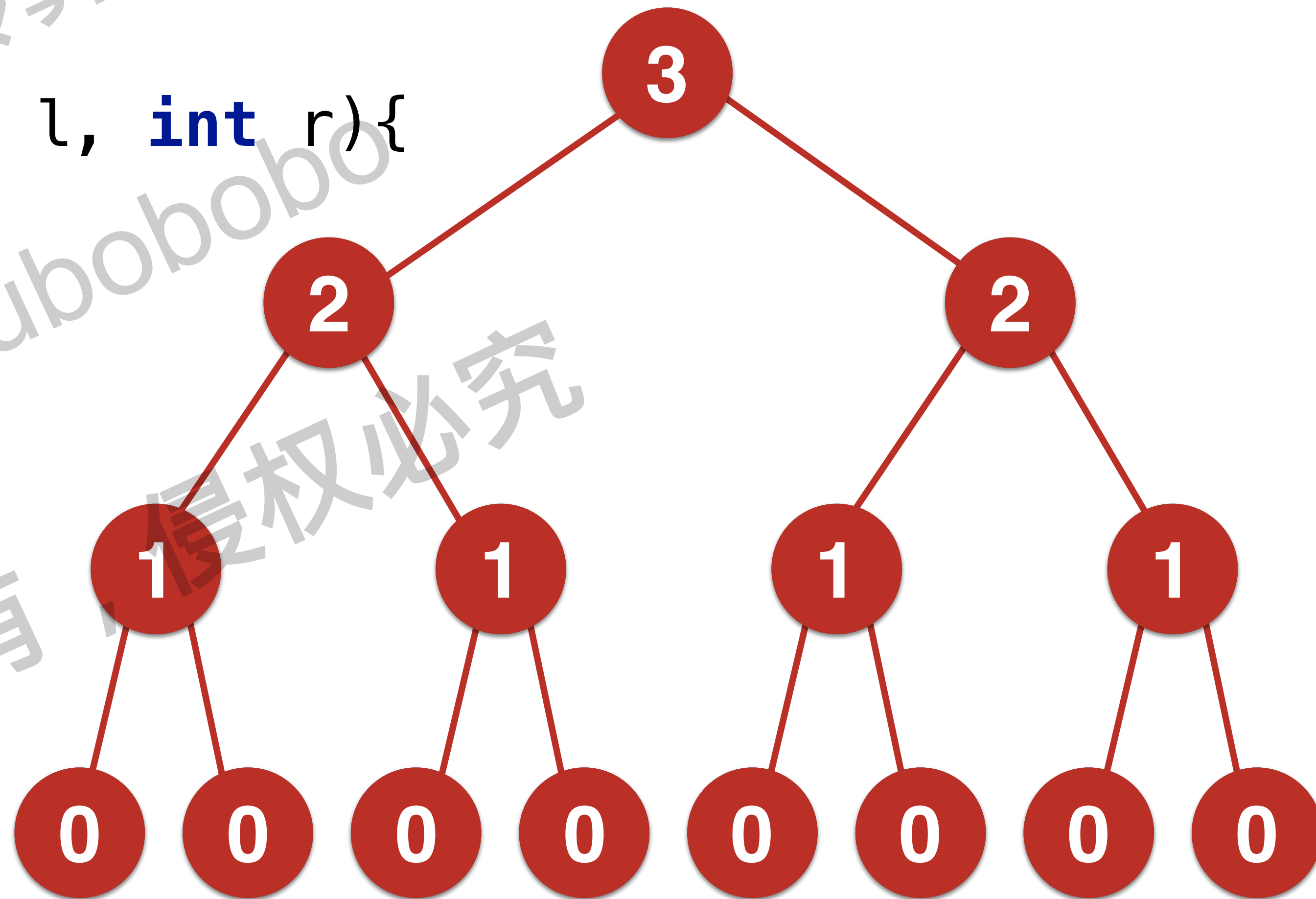
$$= 2^{n+1} - 1$$

$$= O(2^n)$$



递归中进行多次递归调用

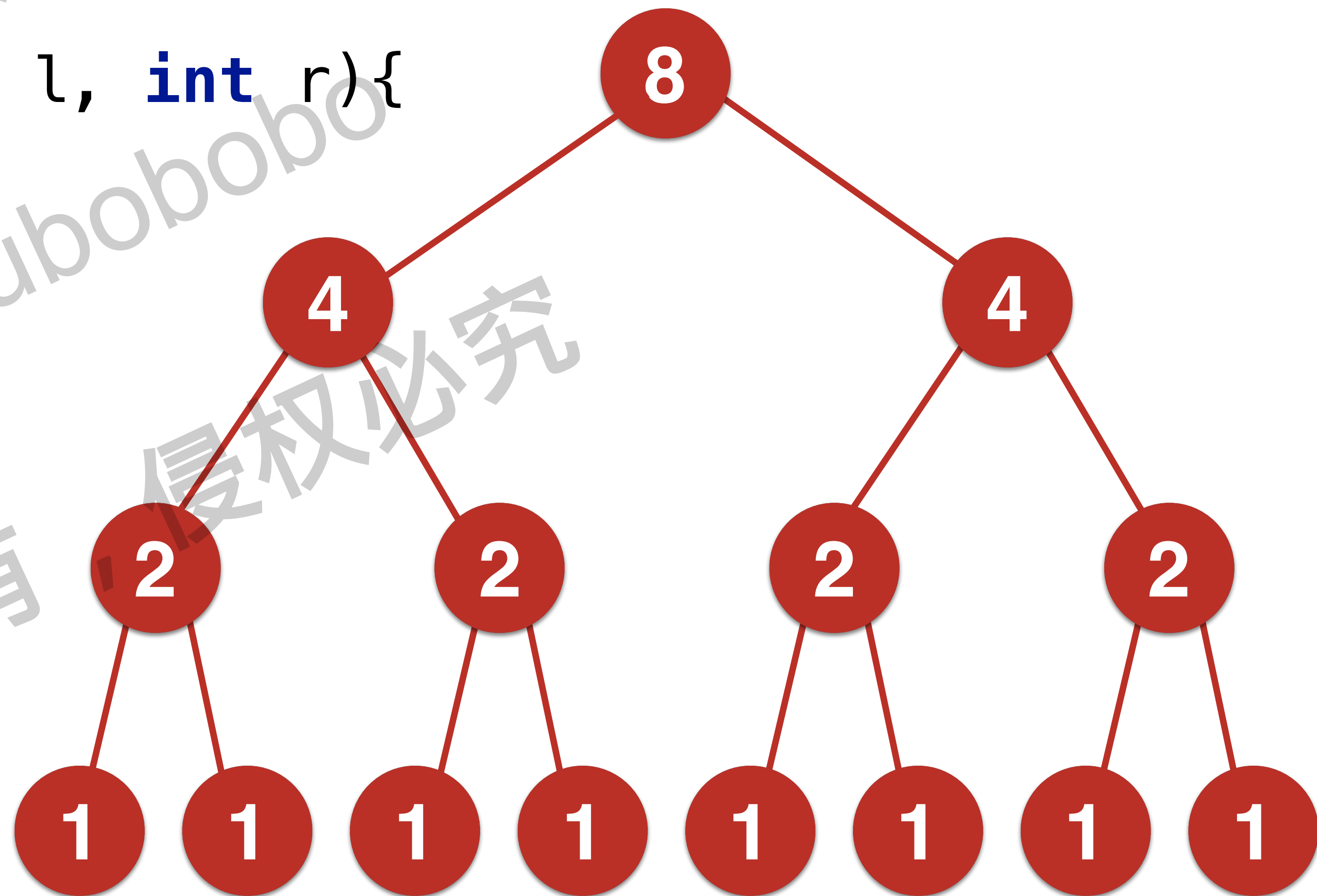
```
void mergeSort(int arr[], int l, int r){  
    if( l >= r )  
        return;  
  
    int mid = (l+r)/2;  
    mergeSort(arr, l, mid);  
    mergeSort(arr, mid+1, r);  
    merge(arr, l, mid, r);  
}
```



递归中进行多次递归调用

```
void mergeSort(int arr[], int l, int r){  
    if( l >= r )  
        return;  
  
    int mid = (l+r)/2;  
    mergeSort(arr, l, mid);  
    mergeSort(arr, mid+1, r);  
    merge(arr, l, mid, r);  
}
```

时间复杂度: $O(n\log n)$



递归函数的时间复杂度

慕课网

《玩转算法面试》

讲师：liuyubobobo

主定理

版权所有，侵权必究

慕课网《玩转算法面试》

均摊复杂度分析

Amortized Time

讲师：huoyubao

版权所有，侵权必究

慕课网《玩转算法面试》

动态数组 (Vector)

讲师：lilypobobo

版权所有，侵权必究

慕课网《玩转算法面试》

实践：编写动态数组

讲师：liuchubobobo

版权所有，侵权必究

均摊复杂度分析

假设当前数组容量为n



1 1 1

每一次添加耗费为 $O(1)$

1 n

n

2

平均来看： $O(1)$

慕课网《玩转算法面试》

实践：验证均摊复杂度

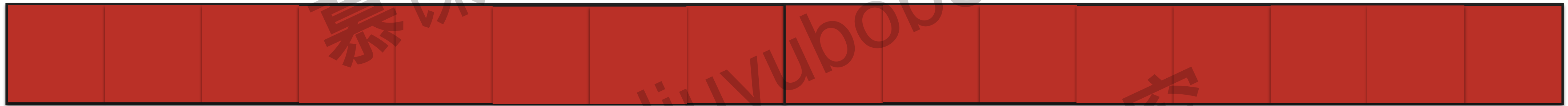
讲师：wuyubobobo

版权所有，侵权必究

实践：删除元素运用resize

均摊复杂度分析

假设当前数组容量为 $2 \times n$

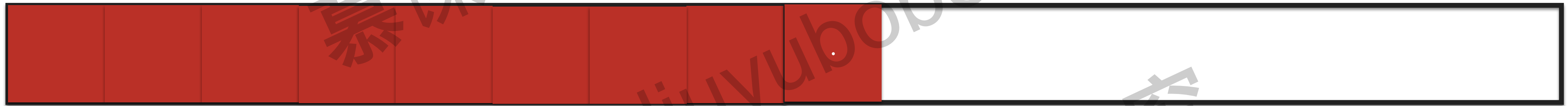


$n+1$ 每一次删除耗费为 $O(1)$ 1 1

2 平均来看： $O(1)$

防止复杂度的震荡

假设当前数组容量为 $2 \cdot n$



添加元素： n

删除元素： n

重复这个过程，无法均摊，复杂度为 $O(n)$

复杂度震荡的解决方案

假设当前数组容量为 $2*n$



当元素个数为数组容量的 $1/4$ 时, `resize`

为再添加元素留出余地

慕课网《玩转算法面试》
讲师：labobobo
版权所有，侵权必究

实践：修改删除元素的resize

慕课网《玩转算法面试》

实践：验证均摊复杂度

讲师：wuyubobobo

版权所有，侵权必究

均摊复杂度

动态数组

动态栈

动态队列

慕课网《玩转算法面试》

玩儿转算法面试

讲师：liuyubobobo

版权所有，侵权必究

liuyubobobo