

04-B3

栈与队列

调用栈：消除递归

邓俊辉

deng@tsinghua.edu.cn

动机 + 方法

❖ 递归函数的空间复杂度

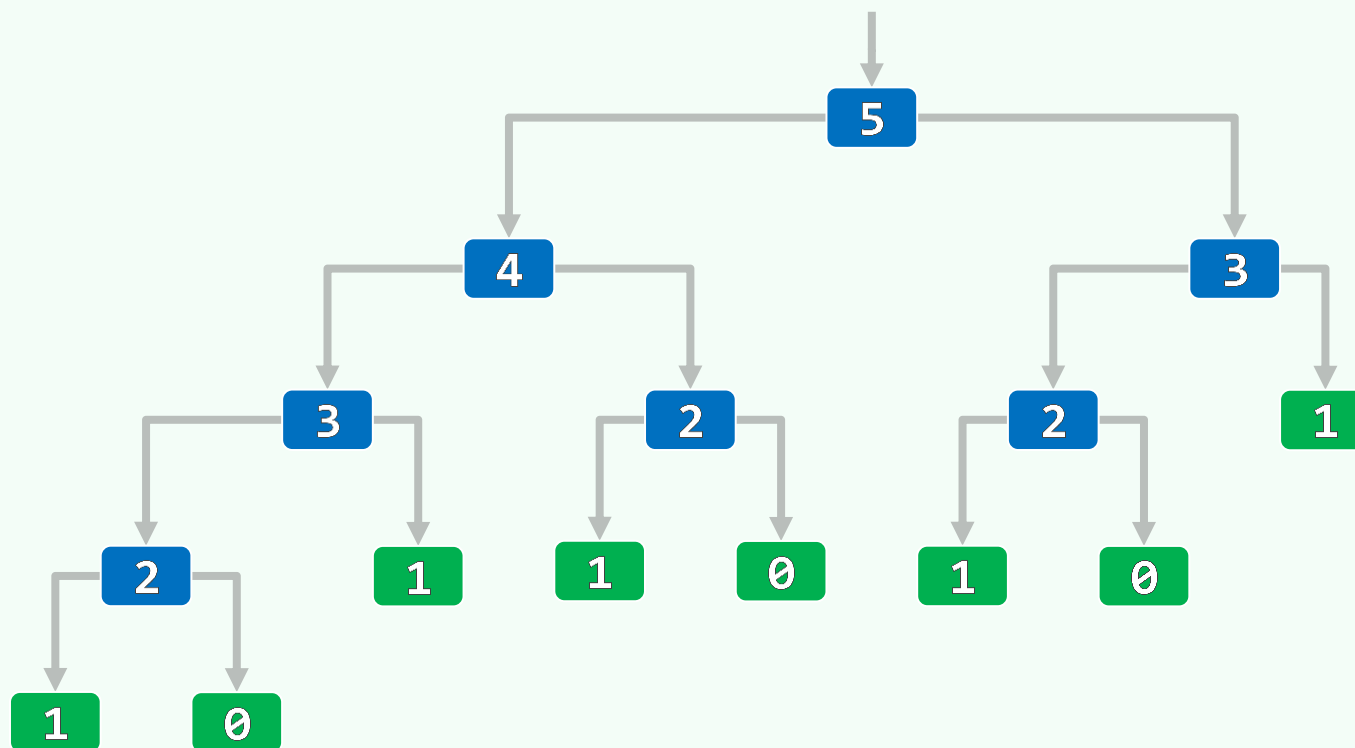
- 主要取决于**最大递归深度**
- 而非**递归实例总数**

❖ 为**隐式地**维护调用栈

需花费额外的时间、空间

❖ 为节省空间，可

- **显式地**维护调用栈
- 将递归算法改写为迭代版本...



实例

❖ 通常，消除递归只是在**常数**意义上优化空间

❖ 但也可能有实质的改进

```
❖ int fac( int n ) {  
    int f = 1; //O(1)空间  
    while ( n > 1 )  
        f *= n--;  
    return f;  
}
```

```
❖ void hailstone( int n ) { //O(1)空间  
    while ( 1 < n )  
        n = n % 2 ? 3*n + 1 : n/2;  
}
```

```
❖ int fib( int n ) { //O(1)空间  
    int f = 0, g = 1;  
    while ( 0 < n-- )  
        { g += f; f = g - f; }  
    return f;  
}
```