本节内容

算法

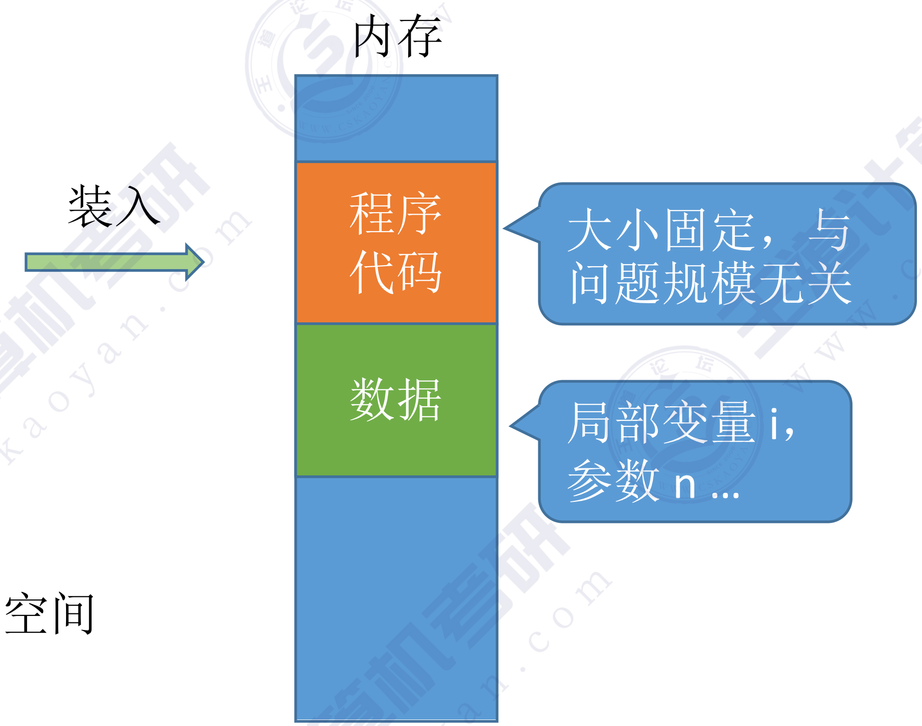
效率的度量

知识总览

- 算法效率的度量
  - 时间复杂度
    - 时间开销与问题规模 n 之间的关系
  - 空间复杂度
    - 空间开销（内存开销）与问题规模 n 之间的关系

程序运行时的内存需求

```c
//算法1— 逐步递增型爱你
void loveYou(int n) {   //n 为问题规模
    int i=1;    //爱你的程度
    while(i<=n){
        i++;    //每次+1
        printf("I Love You %d\n", i);
    }
    printf("I Love You More Than %d\n", n);
}
```



无论问题规模怎么变，算法运行所需的内存空间都是固定的常量，算法空间复杂度为

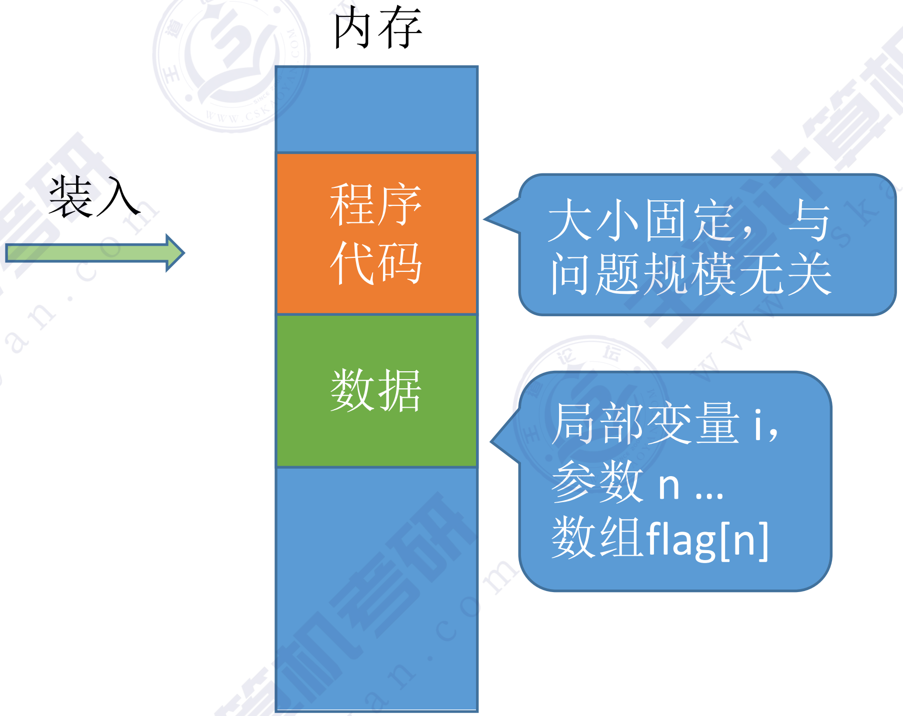
$$S(n) = O(1)$$

注：S 表示 “Space”

算法原地工作——算法所需内存空间为常量

空间复杂度

```c
void test(int n) {
    int flag[n];    //声明一个长度为n的数组
    int i;
    //......此处省略很多代码
}
```



假设一个 int 变量占 4B...
则所需内存空间 = 4 + 4n + 4 = 4n + 8

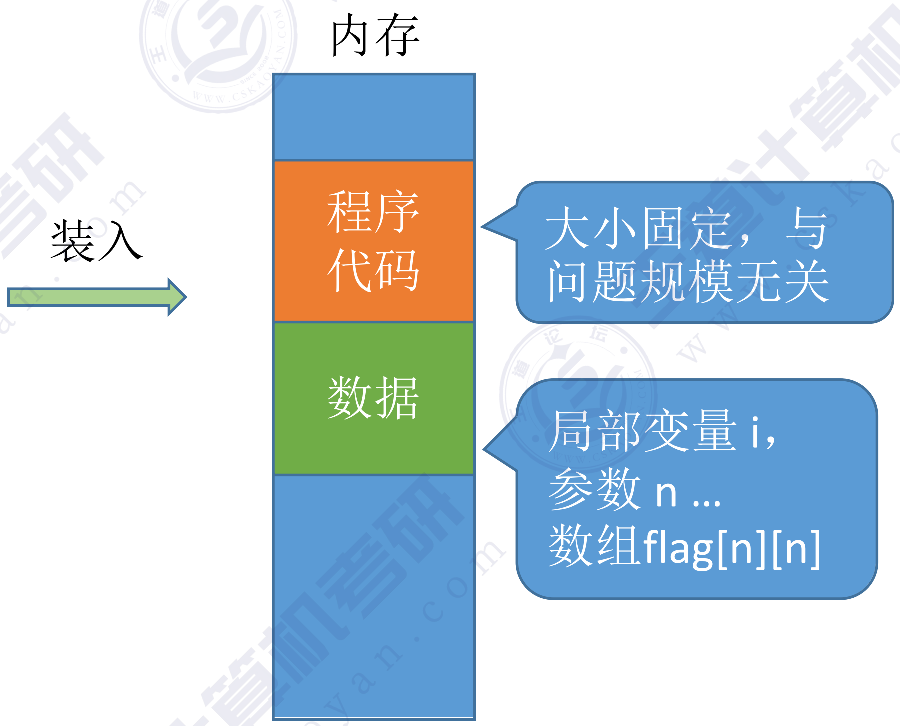
只需关注存储空间大小与问题规模相关的变量

$S(n) = O(n)$

空间复杂度

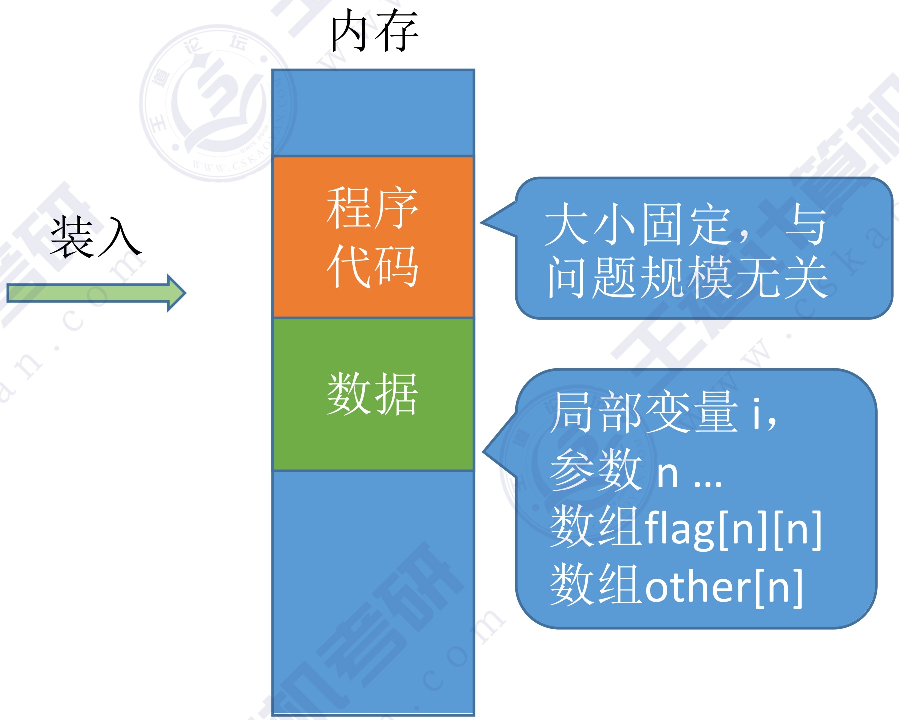
```
void test(int n) {
    int flag[n][n]; //声明 n*n 的二维数组
    int i;
    //......此处省略很多代码
}
```

$S(n) = O(n^2)$



空间复杂度

```
void test(int n) {
    int flag[n][n]; //声明 n*n 的二维数组
    int other[n];   //声明一个长度为n的数组
    int i;
    //......此处省略很多代码
}
```



$$S(n) = O(n^2)+O(n)+O(1) = O(n^2)$$

a）加法规则

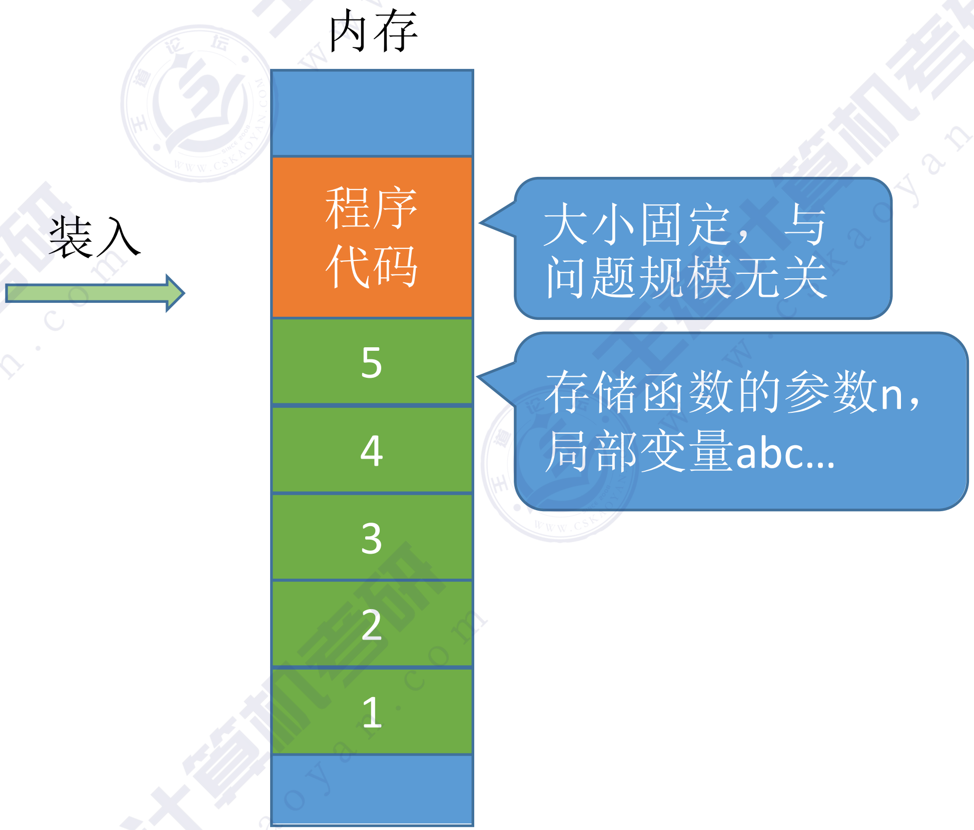
$T(n) = T_1(n) + T_2(n) = O(f(n)) + O(g(n)) = O(\max(f(n), g(n)))$

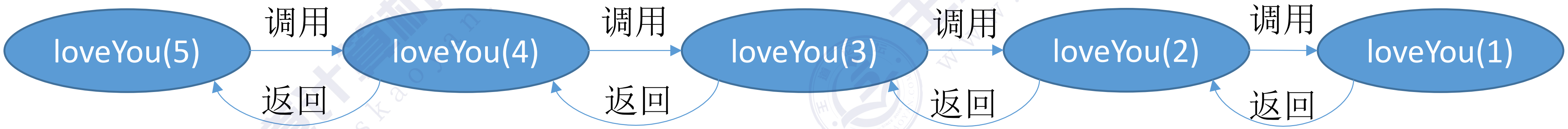
$O(1) < O(\log_2 n) < O(n) < O(n\log_2 n) < O(n^2) < O(n^3) < O(2^n) < O(n!) < O(n^n)$

函数递归调用带来的内存开销

```c
//算法5— 递归型爱你
void loveYou(int n) {    //n 为问题规模
    int a,b,c;   //声明一系列局部变量
    //...省略代码
    if (n > 1) {
        loveYou(n-1);
    }
    printf("I Love You %d\n", n);
}

int main(){
    loveYou(5);
}
```

```
I Love You 1
I Love You 2
I Love You 3
I Love You 4
I Love You 5
```

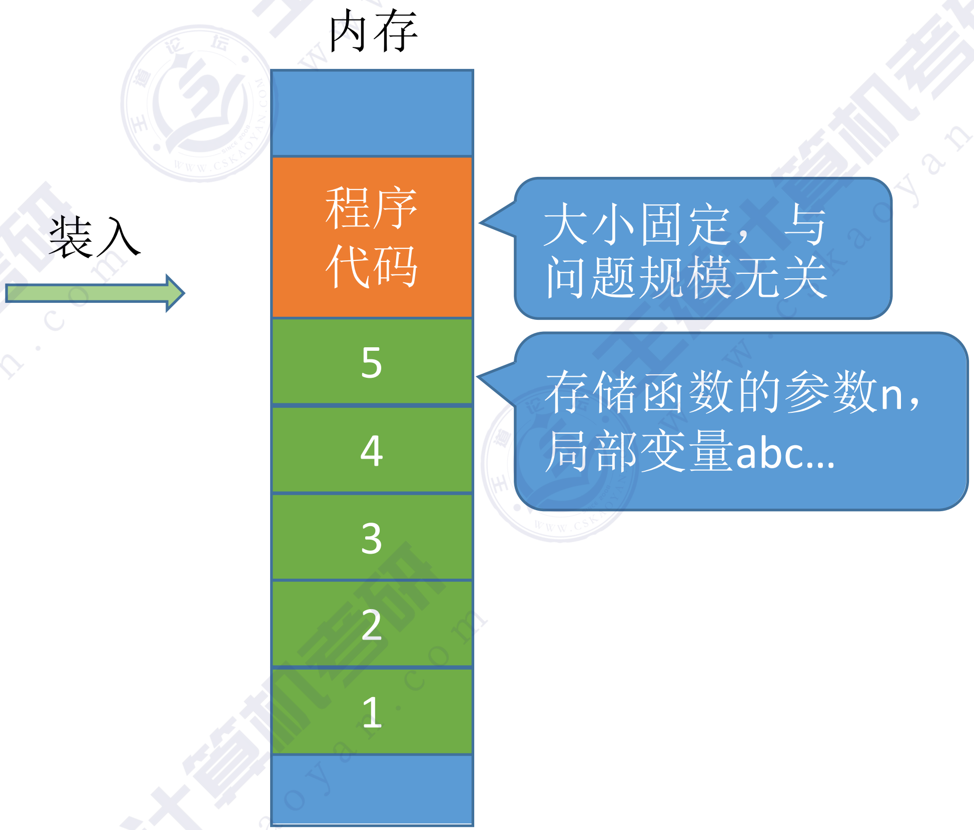




函数递归调用带来的内存开销

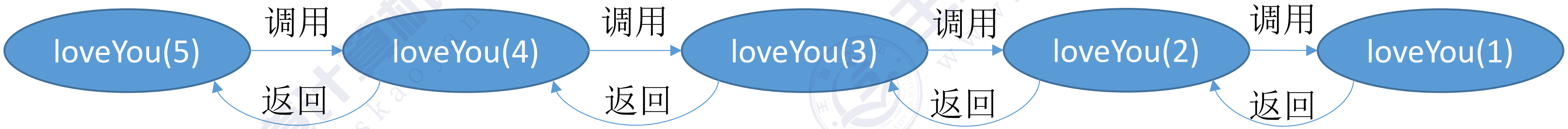
```
//算法5— 递归型爱你
void loveYou(int n) {    //n 为问题规模
    int a,b,c;   //声明一系列局部变量
    //...省略代码
    if (n > 1) {
        loveYou(n-1);
    }
    printf("I Love You %d\n", n);
}

int main(){
    loveYou(5);
}
```

```
I Love You 1
I Love You 2
I Love You 3
I Love You 4
I Love You 5
```



$S(n) = O(n)$

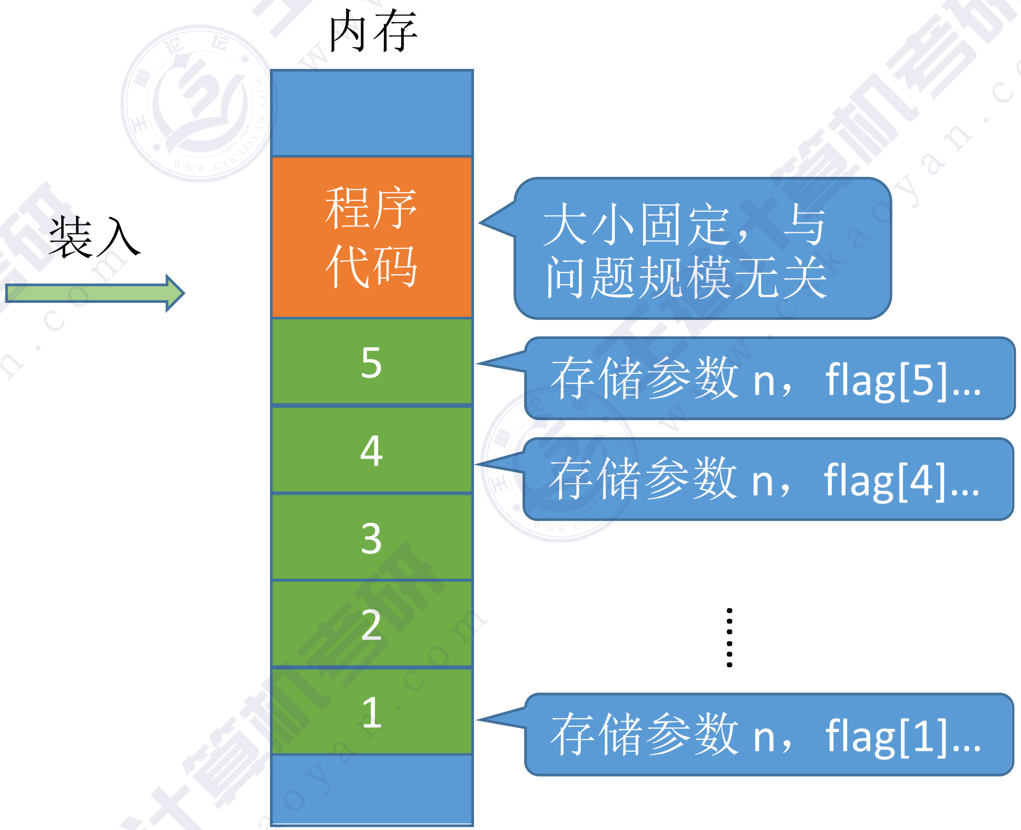
空间复杂度 = 递归调用的深度



函数递归调用带来的内存开销

```c
//算法5— 递归型爱你
void loveYou(int n) {    //n 为问题规模
    int flag[n];     //声明一个数组
    //...省略数组初始化代码
    if (n > 1) {
        loveYou(n-1);
    }
    printf("I Love You %d\n", n);
}

int main(){
    loveYou(5);
}
```

```
I Love You 1
I Love You 2
I Love You 3
I Love You 4
I Love You 5
```



$$1+2+3+...+n = [n(1+n)]/2 = \frac{1}{2}n^2 + \frac{1}{2}n$$

$$S(n) = O(n^2)$$

知识回顾与重要考点

- 空间复杂度
  - 如何计算
    - 普通程序
      - ①找到所占空间大小与问题规模相关的变量
      - ②分析所占空间 x 与问题规模 n 的关系 x=f(n)
      - ③ x 的数量级 O(x) 就是算法空间复杂度 S(n)
    - 递归程序
      - ①找到递归调用的深度 x 与问题规模 n 的关系 x=f(n)
      - ② x 的数量级 O(x) 就是算法空间复杂度 S(n)
      - 注：有的算法各层函数所需存储空间不同，分析方法略有区别
  - 常用技巧
    - 加法规则：$O(f(n)) + O(g(n)) = O(\max(f(n), g(n)))$
    - 乘法规则：$O(f(n))\times O(g(n)) = O(f(n)\times g(n))$
    - “常对幂指阶”
      $O(1) < O(\log_2 n) < O(n) < O(n\log_2 n) < O(n^2) < O(n^3) < O(2^n) < O(n!) < O(n^n)$