

1. 绪论

迭代与递归
减而治之

迭代乃人工，递归方神通

To iterate is human,
to recurse, divine.

邓俊辉

deng@tsinghua.edu.cn

Sum

❖ 问题：计算任意n个整数之和

❖ 实现：逐一取出每个元素，累加之

```
int SumI( int A[], int n ) {  
    int sum = 0; //O(1)  
    for ( int i = 0; i < n; i++ ) //O(n)  
        sum += A[i]; //O(1)  
    return sum; //O(1)  
}
```

❖ 无论A[]内容如何，都有：

$$T(n) = 1 + n*1 + 1 = n + 2 = O(n) = \Omega(n) = \Theta(n)$$

❖ 空间呢？

Decrease-and-conquer

❖ 为求解一个大规模的问题，可以

将其划分为两个子问题：其一**平凡**，另一规模**缩减**

//单调性

分别求解子问题

由子问题的解，得到原问题的解



Linear Recursion: Trace

```
❖ sum( int A[], int n ) {  
    return  
        n < 1 ?  
        0 : sum(A, n - 1) + A[n - 1];  
}
```

❖ 递归跟踪分析

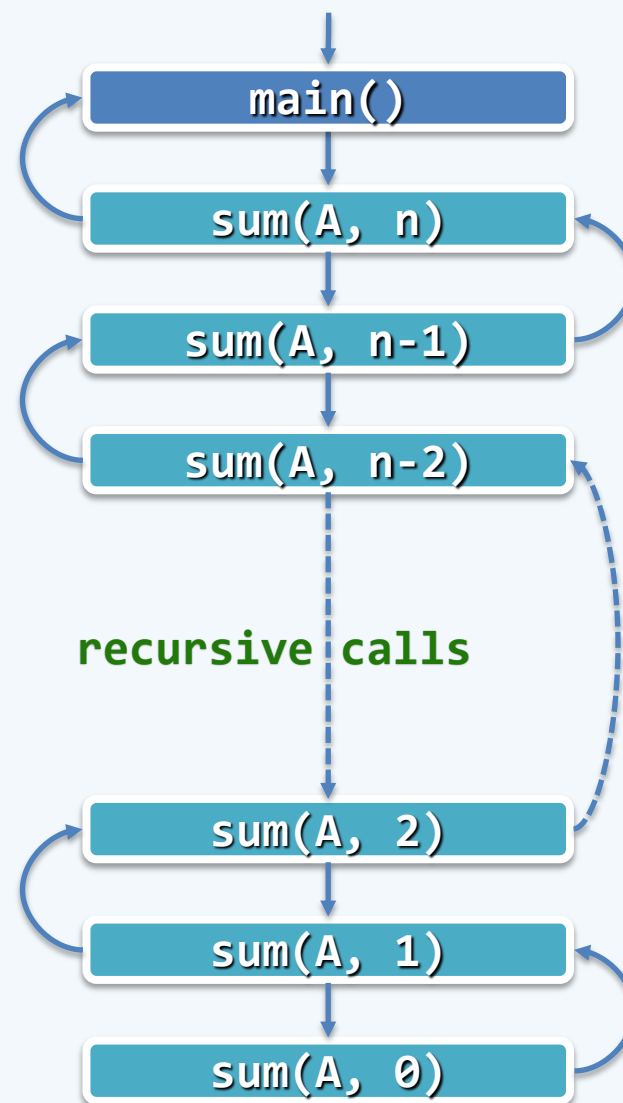
检查每个 **递归实例**

累计所需时间（调用语句本身，计入对应的子实例）

其总和即算法执行时间

❖ 本例中，单个递归实例自身只需 $O(1)$ 时间

$$T(n) = O(1) * (n + 1) = O(n)$$



Linear Recursion: Recurrence

❖ 从递推的角度看，为求解 $\text{sum}(A, n)$ ，需

- 递归求解规模为 $n-1$ 的问题 $\text{sum}(A, n - 1)$ ，再 $//T(n-1)$
- 再累加上 $A[n - 1]$ $//O(1)$

❖ 递推方程 $T(n) = T(n - 1) + O(1)$ $//\text{recurrence}$

$T(0) = O(1)$ $//\text{base: } \text{sum}(A, 0)$

❖ 求解

$$\begin{aligned}T(n) - n &= T(n - 1) - (n - 1) = \dots \\&= T(2) - 2 \\&= T(1) - 1 \\&= T(0) \\T(n) &= O(1) + n = O(n)\end{aligned}$$

Reverse

❖ 任给数组 $A[0, n)$, 将其中的子区间 $A[lo, hi]$ 前后颠倒

统一接口 : `void reverse(int * A, int lo, int hi);`

❖ `if (lo < hi)` // 问题规模的奇偶性不变, 需要两个递归基

// 递归版

```
{ swap( A[lo], A[hi] ); reverse( A, lo + 1, hi - 1 ); }
```

❖ next:

```
if (lo < hi)
```

```
{ swap( A[lo], A[hi] ); lo++; hi--; goto next; }
```

❖ `while (lo < hi) swap(A[lo++], A[hi--]);`

// 迭代原始版

// 迭代精简版

