

Lecture 2: 算法和算法评价

2.1 算法和算法评价

2.1.1 什么是算法

概念：什么是算法

- 程序 = 数据结构 + 算法
- 数据结构是要处理的信息
- 算法是处理信息的步骤

概念：算法的特性

1. **有穷性：** 执行有穷步结束 (程序可以无穷)
2. **确定性：** 算法中每条指令必须有确切的含义，对于相同的输入只能得出相同的输出
3. **可行性：** 算法中描述的操作都可以通过已经实现的基本运算执行有限次来实现
4. **输入：** 一个算法有零个或多个输入，这些输入取自某个特定的对象的集合
5. **输出：** 一个算法有一个或多个输出，这些输出是输入有着某种特定关系的量

概念：好算法的性质

1. **正确性：** 算法应能够正确地解决求解问题
2. **可读性：** 算法应有良好的可读性，以帮助人们理解
3. **健壮性：** 输入非法数据时，算法能适当地做出反应或进行处理，而不会产生莫名奇妙的输出结果
4. **高效率** 与 **低存储量需求** (时间、空间复杂度低)

2.1.2 算法的时间复杂度

概念：算法时间复杂度

- **概念：**
 - 事前预估算法时间开销 $T(n)$ 与问题规模 n 的关系 (T 表示 **time**)
- **注意**
 - 可以只考虑阶数高的部分
- **公式：**

- $$T(n) = O(f(n))$$

- $$T(n) = O(f(n)) \iff \lim_{n \rightarrow \infty} \frac{T(n)}{f(n)} = k$$

- O 表示**同阶**，等同数量级
- 即：当 $n \rightarrow \infty$ 时二者之比为常数

概念：加法规则 - 多项相加，只保留最高阶的项，且系数变为 1

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

概念：乘法规则 - 多项相乘，都保留

$$T(n) = T_1(n) \times T_2(n) = O(f(n)) \times O(g(n)) = O(f(n) \times g(n))$$

总结：数量级

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

2.1.3 时间复杂度

分析：时间复杂度分析

1. 顺序执行的代码只会影响常数项，可以忽略
2. 只需挑循环中的一个基本操作分析它的执行次数与 n 的关系即可
3. 如果有多层嵌套循环，只需关注最深层循环循环了几次

举例

- 举例 1
 - 出现二层嵌套循环： $T(n) = O(n) + O(n^2) = O(n^2)$
 - 其中： $O(n) + O(n^2)$ 中的 $O(n)$ 为外层循环， $O(n^2)$ 为内层循环；
- 举例 2
 - 出现一层循环，但循环的终止条件和循环次数有关：

代码示例

```
void function(int n)
{
    int i = 0;
    while(i<=n)
    {
        i*=2;
        //操作
    }
}
```

- 分析：此时函数操作次数假设为 x ，而当 $2^x > n$ 时，函数终止；
 - 所以由 $2^x > n$ 可知其 $x = \log_2 n$ ，时间复杂度为 $\log$ 数量级；
 - 一般情况下我们只会计算最坏时间复杂度和平均时间复杂度（取 n 趋于无穷），最好时间复杂度一般不计算；

2.1.4 空间复杂度

概念：空间复杂度与递归程序

- 函数递归调用带来的内存开销

- **空间复杂度 = 递归调用的深度；**
- $S(n) = O(n)$: 递归程序的空间复杂度为 $O(n)$;

总结：计算方法

- 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 $x=f(x)$
 2. x 的数量级 $O(x)$ 就是算法空间复杂度 $S(n)$
 3. **注意：** 有的算法各层所需存储空间不同，分析方法略有区别

2.1.4.1 递归程序的空间复杂度计算

代码示例

```
void function(int n,int breakNumber = 0)
{
    int numbers[n];
    if(n > breakNumber)
    {
        function(n-1);
    }
}
```

- 分析：
 - 基本情况
 - 每次循环时的代价 = 数组长度 + n (+ 函数调用栈、函数元信息等) ;
 - 且数组长度和 n 有关
 - 分析方法：从有限到无穷 -> 数学归纳法的思想
 - $n=1, S = 1 + 1$
 - $n=2, S = 2 + 1$
 - $n=3, S = 3 + 1$
 - ...
 - $n=n-1, S = n-1 + 1$
 - $n=n, S = n+1$
 - 求和： $1+2+3+...+n + (1+1+1...) = \frac{1}{2}n^2 + \frac{1}{2}n$