



Assignment1

课程： 算法设计与分析

姓名： 雷翔

学号： 2053932

时间： 2023 年 3 月

Q1. Prove (by using the definitions of the notations involved) or disprove (by giving a specific counterexample) the following assertions.

a. If $t(n) \in O(g(n))$, then $g(n) \in \Omega(t(n))$.

b. $\Theta(\alpha g(n)) = O(g(n))$, where $\alpha > 0$.

c. $\Theta(g(n)) = O(g(n)) \cap \Omega(g(n))$.

d. For any two nonnegative functions $t(n)$ and $g(n)$ defined on the set of nonnegative integers, either $t(n) \in O(g(n))$, or $t(n) \in \Omega(g(n))$, or both.

Ans.

a. 正确。因为 $t(n) \in O(g(n))$, 所以存在 $c > 0$, $n_0 > 0$, 使得当 $n \geq n_0$ 时, 满足 $t(n) \leq c \cdot g(n)$ 。不等式两边同时除以常数 c 得 $\frac{1}{c} \cdot t(n) \leq g(n)$, 即 $k \cdot t(n) \leq g(n)$, 其中 $k > 0$, 所以 $g(n) \in \Omega(t(n))$ 。

b. 错误。

令 $\alpha = 1$, 即证 $\Theta(g(n)) \neq O(g(n))$ 。

取 $f(n) = n$, $g(n) = n^2$ 。

$O(g(n)) = \{f(n): \text{存在 } d_1 > 0, n_1 > 0, \text{使得当 } n > n_1 \text{ 时, 满足 } f(n) \leq d_1 g(n)\}$ 。

取 $d_1 = 1$, $n_1 = 1$, 当 $n \geq n_1$ 时, $n^2 \geq n$, 所以 $f(n) \in O(g(n))$ 。

$\Theta(g(n)) = \{f(n): \text{存在 } c_1 > 0, c_2 > 0, n_0 > 0, \text{使得当 } n \geq n_0 \text{ 时, 满足 } c_2 g(n) \leq f(n) \leq c_1 g(n)\}$ 。

不存在 $c_1 > 0, c_2 > 0$, 使得 $c_1 n \leq n^2 \leq c_2 n$ 恒成立, 即 $n \notin \Theta(n^2)$ 。

所以, $\Theta(g(n)) \neq O(g(n))$ 。

c. 正确。

$O(g(n)) = \{f(n): \text{存在 } d_1 > 0, n_1 > 0, \text{使得当 } n > n_1 \text{ 时, 满足 } f(n) \leq d_1 g(n)\}$ 。

$\Omega(g(n)) = \{h(n): \text{存在 } d_2 > 0, n_2 > 0, \text{使得当 } n > n_2 \text{ 时, 满足 } h(n) \geq d_2 g(n)\}$ 。

所以对 $\forall t(n) \in O(g(n)) \cap \Omega(g(n))$ 当 $n \geq \max(n_1, n_2)$ 时, 都有 $t(n) \leq d_1 g(n)$ 和 $t(n) \geq d_2 g(n)$, 即 $d_2 g(n) \leq t(n) \leq d_1 g(n)$, 故 $t(n) \in \Theta(g(n))$ 。

同理, $\Theta(g(n)) = \{f(n): \text{存在 } c_1 > 0, c_2 > 0, n_0 > 0, \text{使得当 } n \geq n_0 \text{ 时, 满足 } c_2 g(n) \leq f(n) \leq c_1 g(n)\}$ 。

所以对 $\forall t(n) \in \Theta(g(n))$, 当 $n \geq n_0$ 时, 都有 $c_2 g(n) \leq t(n) \leq c_1 g(n)$, 即 $t(n) \geq c_2 g(n)$ 和 $t(n) \leq c_1 g(n)$, 故 $t(n) \in O(g(n)) \cap \Omega(g(n))$, 即 $t(n) \in O(g(n)) \cap \Omega(g(n))$ 。

综上, $\Theta(g(n)) = O(g(n)) \cap \Omega(g(n))$ 。

d. 错误。

Q2. Calculate the time complexity of the following algorithms respectively.

a.

```
i = 0;
while ((i + 1) * (i + 1) <= n)
    i = i + 1;
```

b.

```
x = 0;
for (i = 1; i <= n; i++)
    for (j = 1; j <= i; j++)
        for (k = 1; k <= j; k++)
            x++;
```

Ans.

a. 问题规模: n

基本操作: while 语句中的乘法

执行次数: $M(n) = \lfloor \sqrt{n} \rfloor + 1 \quad (n \geq 0)$

时间复杂度: $f(n) \in \Theta(\sqrt{n})$

b. 问题规模: n

基本操作: $x++$

执行次数:

$$\begin{aligned}
 A(n) &= \sum_{i=1}^n \sum_{j=1}^i \sum_{k=1}^j 1 \\
 &= \sum_{i=1}^n \sum_{j=1}^i j \\
 &= \sum_{i=1}^n \frac{1}{2} i(i+1) \\
 &= \frac{1}{6} n(n+1)(n+2)
 \end{aligned}$$

时间复杂度: $f(n) \in \Theta(n^3)$

Q3. Calculate the time complexity of the following recursive algorithms respectively (If it may, the worst, average, and best cases must be investigated separately.)

a.

```
int function(int x, int n)
{
    if (n == 0)
        return 1;
    int t = function(x, n / 2)
    if (n % 2 == 1)
        return t * t * x;
    return t * t;
}
```

b.

```
void Sort(int A[], int low, int high)
{
    if (low < high)
    {
        int pivot = Partition(A, low, high);
        Sort(A, low, pivot - 1);
        Sort(A, pivot + 1, high);
    }
}

int Partition(int A[], int low, int high)
{
    int pivot = A[low];
    while (low < high)
    {
        while (low < high && A[high] >= pivot)
            --high;
        A[low] = A[high];
        while (low < high && A[low] <= pivot)
            ++low;
        A[high] = A[low];
    }
    A[low] = pivot;
    return low;
}
```

Ans.

a. 问题规模: n

基本操作: $n\%2 == 1$ 比较

执行次数: $C(n) = C(n/2) + 1$ ($n \geq 1$) 和 $C(1) = 1$

根据平滑性原则, 令 $n = 2^k$, 则有 $C(2^k) = C(2^{k-1}) + 1$, $C(2^k) = \dots = C(2^{k-k}) + k = C(1) + k = k + 1$, 所以有 $C(2^k) = k + 1$, 故 $C(n) = \log_2(n) + 1$

时间复杂度: $function(x, n) \in \Theta(\log_2(n))$

b. 问题规模: n

基本操作: Partition 函数中三个 while 循环中的比较

执行次数:

最优情况: 每次划分都恰好将数组分成长度相等的子数组

最坏情况: 初始序列有序, 每次分割只产生一个子序列

平均情况:

时间复杂度:

Q4. Solve the following recurrence relations.

a. $T(n) = T(n-1) + n$ for $n > 0, T(0) = 1$

b. $T(n) = 4T(n-1)$ for $n > 0, T(1) = 5$

c. $T(n) = T(n/3) + n$ for $n > 2, T(1) = 1$ (solve for $n = 3^k$)

Ans.

a.

$$\begin{aligned} T(n) &= T(n-1) + n \\ &= T(n-2) + (n-1) + n \\ &= T(n-3) + (n-2) + (n-1) + n \\ &= \dots \\ &= T(0) + 1 + 2 + 3 + \dots + n \\ &= \frac{n^2 + n + 2}{2} \quad (n \geq 0) \end{aligned}$$

$$T(n) \in \Theta(n^2)$$

b.

$$\begin{aligned} T(n) &= 4T(n-1) \\ &= 4 \times 4T(n-2) \\ &= \dots \\ &= 4^{n-1}T(1) \\ &= 5 \times 4^{n-1} \quad (n \geq 1) \end{aligned}$$

$$T(n) \in \Theta(4^n)$$

c. 令 $n = 3^k$, 结合平滑性原则。

$$\begin{aligned} T(3^k) &= T(3^{k-1}) + 3^k \\ &= T(3^{k-2}) + 3^{k-1} + 3^k \\ &= T(3^{k-3}) + 3^{k-2} + 3^{k-1} + 3^k \\ &= \dots \\ &= T(3^0) + 3^1 + 3^2 + \dots + 3^k \\ &= 1 + 3^1 + 3^2 + \dots + 3^k \\ &= \frac{(3^{k+1} - 1)}{2} \end{aligned}$$

所以, $T(n) = \frac{(3n-1)}{2} \ (n \geq 1)$

$T(n) \in \Theta(n)$