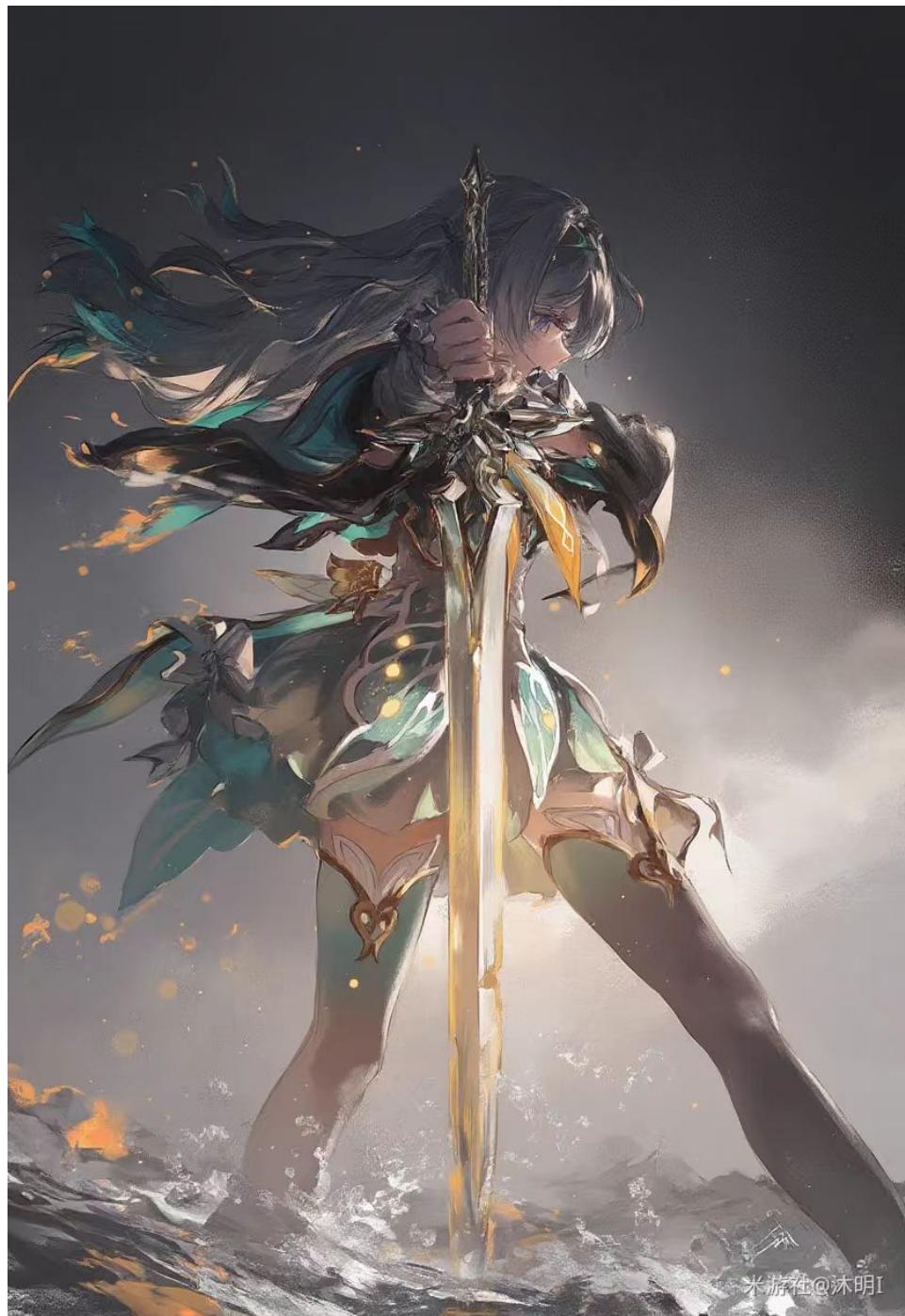


# 绪论

Tan Yiqing

2025 年 10 月 31 日



米游社@沐明I

# 1 问题求解和程序设计

## 2 数据结构的基本概念

### 2.1 数据结构

数据、数据元素、数据项、数据结构、逻辑结构、储存结构

## 3 算法的基本概念

### 3.1 算法的定义

算法是对特定问题求解过程的一种描述，为指令的有限序列。

### 3.2 算法的特性

有穷性，确定性，可行性，输入，输出，基本操作。

好代码的特点：正确性，健壮性，可理解性，抽象分级，高效性。

### 3.3 算法的描述方法

自然语言，伪代码，流程图，程序语言。

### 3.4 算法分析

研究方法：事后统计，事前分析

#### 3.4.1 时间复杂度

认为基本语句的时间一致，关心基本语句执行次数。

定义：对于两个函数  $f(n)$  和  $g(n)$ ，如果存在正常数  $c$  和  $n_0$ ，使得：

$$\forall n > n_0, f(n) \leq cg(n)$$

则称  $f(n)$  的渐近上界为  $O(g(n))$ ，记作  $f(n) = O(g(n))$ 。

#### 1. $O(\log_2 n)$ 示例：二分查找

```
int binarySearch(int arr[], int n, int target) {
    int left = 0, right = n - 1;
    while (left <= right) {
        int mid = left + (right - left) / 2;
        if (arr[mid] == target) return mid;
        else if (arr[mid] < target) left = mid + 1;
        else right = mid - 1;
    }
    return -1;
}
```

#### 2. $O(n)$ 示例：线性查找

```

int linearSearch(int arr[], int n, int target) {
    for (int i = 0; i < n; ++i) {
        if (arr[i] == target) return i;
    }
    return -1;
}

```

### 3. $O(n^2)$ 示例：冒泡排序

```

void bubbleSort(int arr[], int n) {
    for (int i = 0; i < n - 1; ++i) {
        for (int j = 0; j < n - i - 1; ++j) {
            if (arr[j] > arr[j + 1]) {
                swap(arr[j], arr[j + 1]);
            }
        }
    }
}

```

### 4. $O(2^n)$ 示例：斐波那契数列

```

int fib(int n) {
    if (n <= 1) return n;
    return fib(n - 1) + fib(n - 2);
}

```

### 5. $O(n!)$ 示例：全排列

```

void permute(vector<int>& nums, int l, int r) {
    if (l == r) {
        // print permutation
        return;
    }
    for (int i = l; i <= r; ++i) {
        swap(nums[1], nums[i]);
        permute(nums, l + 1, r);
        swap(nums[1], nums[i]);
    }
}

```

## 1. 辗转相除法时间复杂度证明：

设欧几里得算法为：对  $a > b > 0$ ，不断执行  $(a, b) \leftarrow (b, a \bmod b)$  直到  $b = 0$ 。记余数序列

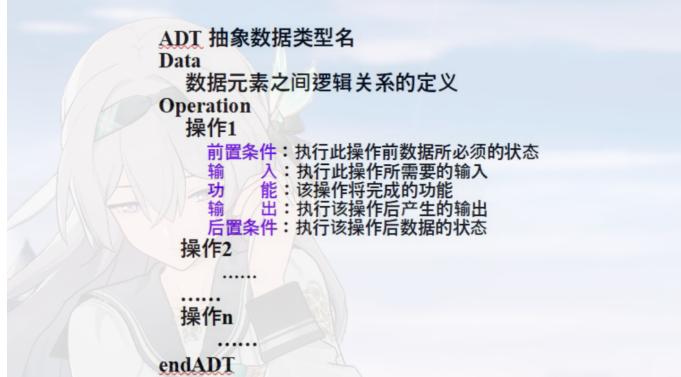
$$r_{-1} = a, \quad r_0 = b, \quad r_{i-1} = q_i r_i + r_{i+1} \quad (q_i \in \mathbb{Z}_{\geq 1}), \quad 0 \leq r_{i+1} < r_i.$$

算法在  $r_t = 0$  时结束，步数为  $t$ 。

**关键不等式（两步减半）：**由  $r_i = q_{i+1}r_{i+1} + r_{i+2}$  且  $q_{i+1} \geq 1$ ，得

$$r_i \geq r_{i+1} + r_{i+2} \geq 2r_{i+2} \Rightarrow r_{i+2} \leq \frac{r_i}{2} \quad (\forall i \geq 0).$$

## 抽象数据类型的书写方法)



因此对任意  $k \geq 0$ ,

$$r_{2k} \leq \frac{r_0}{2^k} = \frac{b}{2^k}, \quad r_{2k+1} \leq \frac{r_1}{2^k} \leq \frac{a}{2^k}.$$

**步数上界:** 当  $k > \log_2 b$  时,  $r_{2k} < 1$ , 而  $r_{2k}$  为非负整数, 只能为 0, 算法已在(或早于)第  $2k$  步结束。故

$$t \leq 2\lceil \log_2 b \rceil = O(\log b) \subseteq O(\log n), \quad (n = \min\{a, b\}).$$

结论: 欧几里得算法时间复杂度为  $O(\log n)$ 。

## 4 数据结构

**定义:** 数据结构由数据与结构组成。数据之间的关系即为其结构。

结构的视点分为逻辑结构与存储结构, 前者为数据之间的逻辑关系(如前面有看过的例子); 后者又称物理结构, 为数据及逻辑结构在计算机中的表示。

逻辑结构是用户视角, 面向问题的。存储结构是实现视角, 面向计算机的。

**数据类型:** 一组值的集合以及定义于这个集上的一组操作的总称。(例: 整数类型)

**抽象:** 抽象出本质的特征而略过细节。

**抽象数据类型 (Abstract DataType, ADT):** 一个数据结构及对该结构上的一组操作的总称。