# 算法入门-贪心算法

Junlist

2025 年 11 月 12 日

## 1 贪心算法概述

贪心算法（又称贪婪算法）是指，在对问题求解时，总是做出在当前看来是最好的选择。也就是说，不从整体最优上加以考虑，算法得到的是在某种意义上的局部最优解。

贪心算法不是对所有问题都能得到整体最优解，关键是贪心策略的选择。

### 1.1 贪心算法的基本步骤

贪心算法一般按如下步骤进行：

1. 建立数学模型来描述问题
2. 把求解的问题分成若干个子问题
3. 对每个子问题求解，得到子问题的局部最优解
4. 把子问题的解局部最优解合成原来问题的一个解

### 1.2 贪心算法的正确性证明

在使用贪心算法解决问题时，**必须证明贪心策略的正确性**。贪心算法的难点在于如何证明其选择策略能得到全局最优解。

#### 1.2.1 数学归纳法

- 先验证最小规模（如 $n = 1$）时成立
- 再证明 $n$ 成立时 $n + 1$ 也成立

### 1.2.2 交换论证法

- 假设存在更优解
- 通过交换局部选择，如果不会得到更优结果
- 则当前贪心解为最优

实际做题时，通常不要求严格证明。判断是否可用贪心算法，可以通过：

- **直觉尝试**：先用贪心思路做一遍，看看局部最优能否推出全局最优
- **举反例**：尝试构造反例，如果找不到局部最优导致全局最优失败的例子，基本可以用贪心法

贪心算法是一种极其灵活的算法，但也意味着对其的应用存在一定风险。例如在做简单的壹角、伍分、壹分的找零问题时候我们可以使用每次选择面值最大的硬币这个贪心策略。但面对同样的找零问题如果硬币换为壹角壹分、伍分、壹分便不可使用。

# 2 Havel-Hakimi定理

## 2.1 定理描述

设 $d = (d_1, d_2, \ldots, d_n)$ 是一个非增的非负整数序列，其中 $d_1 \geq d_2 \geq \cdots \geq d_n$。该序列可图化当且仅当：

1. $d_1 \leq n - 1$
2. 序列 $d' = (d_2 - 1, d_3 - 1, \ldots, d_{d_1+1} - 1, d_{d_1+2}, \ldots, d_n)$ 可图化

### 2.2 Havel-Hakimi算法

**Algorithm 1** Havel-Hakimi算法：判断度序列是否可图化

**procedure** HavelHakimi($d$)
  将序列 $d$ 按非递增顺序排序
  **while** 序列不为全零序列 **do**
    **if** 序列中存在负数 **then**
      **return** 不可图化
    **end if**
    **if** 第一个元素 $d_1$ 大于剩余元素个数 **then**
      **return** 不可图化
    **end if**
    **for** $i \leftarrow 2$ to $d_1 + 1$ **do**
      $d_i \leftarrow d_i - 1$
    **end for**
    移除第一个元素 $d_1$
    重新排序序列为非递增顺序
  **end while**
  **return** 可图化
**end procedure**

## 3 C++中的sort()函数

### 3.1 sort()函数概述

C++标准库中的`sort()`函数是`<algorithm>`头文件中提供的快速排序实现，具有$O(n \log n)$的时间复杂度。 `sort()`函数主要有两种重载形式：

```
template <class RandomAccessIterator>
void sort(RandomAccessIterator first, RandomAccessIterator
    last);

template <class RandomAccessIterator, class Compare>
void sort(RandomAccessIterator first, RandomAccessIterator
    last, Compare comp);
```

Listing 1: sort函数的两种形式

**参数说明：**

- **first**：随机访问迭代器，指向要排序序列的**第一个元素**
- **last**：随机访问迭代器，指向要排序序列的**最后一个元素的下一个位置**
- **comp**：二元谓词（比较函数），定义排序规则（可选参数）

**first 和 last 参数**

- **类型**：必须满足随机访问迭代器（RandomAccessIterator）
- **支持的容器**：vector, deque, array, 普通数组等
- **范围**：排序区间为 $[first, last)$，即包含first，不包含last

## 3.2 comp()函数定义

comp函数（比较函数）是一个二元谓词（Binary Predicate），接收两个参数并返回一个布尔值。其是C++标准库排序算法中的核心组件，其有三种形式：普通函数、函数对象（仿函数）、Lambda表达式，它定义了元素之间的排序规则。正确理解和定义comp函数对于使用`sort()`等算法至关重要。

```
bool comp(const Type& a, const Type& b) {
    // return true if a should come before b
    // return false otherwise
}
```

### 3.2.1 严格弱序要求

comp函数必须满足**严格弱序**（Strict Weak Ordering）的数学要求：

- **非自反性**：对于任何元素a，`comp(a, a)` 必须返回 `false`

- **反对称性**：如果 comp(a, b) 为 true，则 comp(b, a) 必须为 false
- **传递性**：如果 comp(a, b) 为 true 且 comp(b, c) 为 true，则 comp(a, c) 必须为 true
- **等价传递性**：如果 !comp(a, b) && !comp(b, a) 且 !comp(b, c) && !comp(c, b)，则 !comp(a, c) && !comp(c, a)