

# 枚举（“10”）

T E A C H I N G   C O U R S E W A R E   P O W P O I N T

授课时间：2025.07.26

# 目录

- PART-01 算法引入 TEACHING COURSEWARE
- PART-02 概念分析 TEACHING COURSEWARE
- PART-03 经典例题 TEACHING COURSEWARE
- PART-04 总结思考 TEACHING COURSEWARE

01

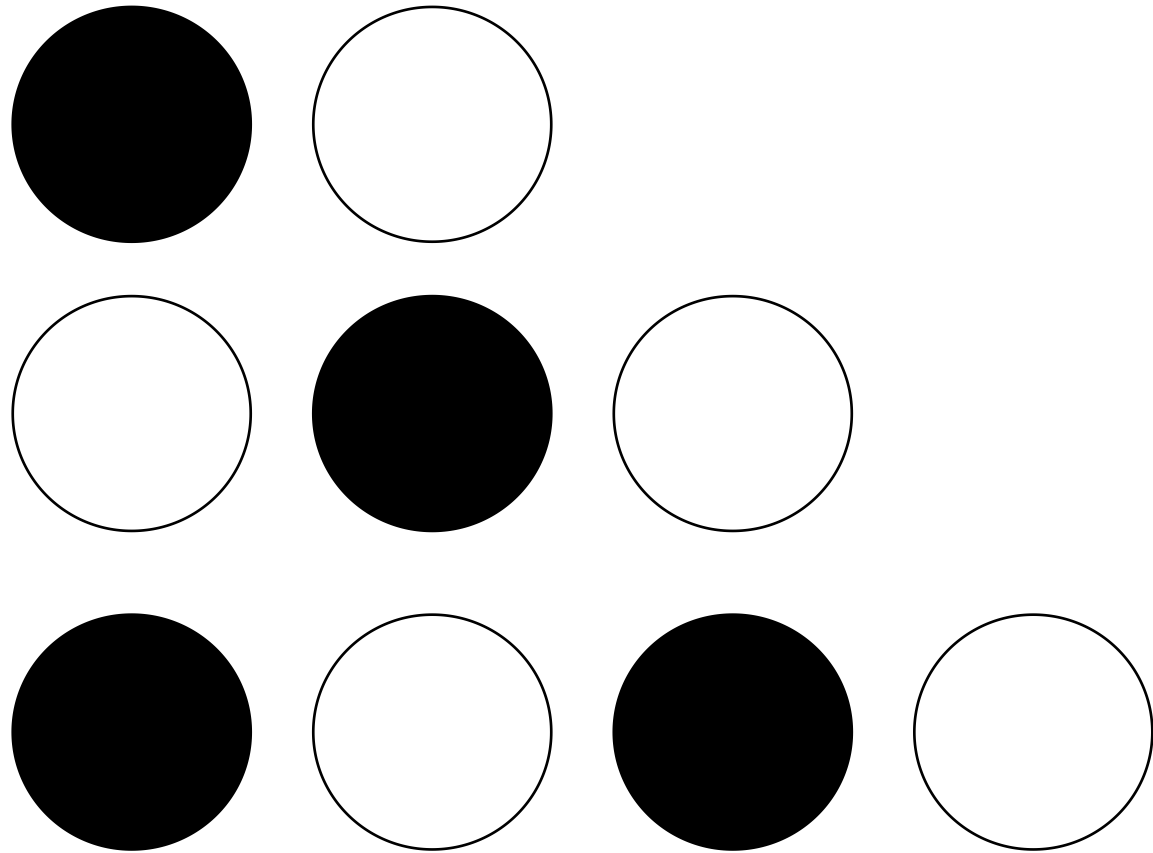
# 算法引入

TEACHING  
COURSEWARE

TEACH



# 算法引入



02

# 概念分析

TEACHING  
COURSEWARE

TEACH

# 概念分析

## 1. 什么是二进制？

二进制是仅用0和1表示数值的计数法（基数为2），与我们熟悉的十进制（基数为10）对应：

十进制：每满10进1（如1, 2, ..., 9, 10, 11...）；

二进制：每满2进1（如1, 10, 11, 100, 101...）。

## 2. 二进制与十进制的转换

二进制转十进制：每一位的“数值 $\times 2^{\text{位数}}$ ”之和（从右往左数，第0位开始）。

例：二进制101  $\rightarrow 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0 = 4 + 0 + 1 = 5$ （十进制）。

十进制转二进制：除以2取余数，倒序排列。

例：十进制6  $\rightarrow 6 \div 2 = 3$ 余0,  $3 \div 2 = 1$ 余1,  $1 \div 2 = 0$ 余1  $\rightarrow$  二进制110。

## 3. 二进制的核心意义

二进制的每一位（bit）只有0或1两种状态，恰好可以表示“是/否”“有/无”“选/不选”等对立状态——这是二进制枚举的基础。



# 位运算

| 运算符号 | 名称   | 作用（对二进制每一位）                          | 例子（二进制）               | 十进制结果       |
|------|------|--------------------------------------|-----------------------|-------------|
| &    | 按位与  | 两位均为 1 则为 1，否则为 0                    | 101(5) & 110(6) = 100 | 5 & 6 = 4   |
|      | 按位或  | 两位至少有 1 则为 1，否则为 0                   | 101(5)   110(6) = 111 | 5   6 = 7   |
| ^    | 按位异或 | 两位不同则为 1，相同则为 0                      | 101(5) ^ 110(6) = 011 | 5 ^ 6 = 3   |
| <<   | 左移   | 所有位向左移 k 位，右边补 0（相当于 $\times 2^k$ ）  | 101(5) << 1 = 1010    | 5 << 1 = 10 |
| >>   | 右移   | 所有位向右移 k 位，左边补符号位（相当于 $\div 2^k$ 取整） | 1010(10) >> 1 = 101   | 10 >> 1 = 5 |

# 二进制枚举

## 1. 核心思想

当需要枚举“ $n$ 个元素中**所有可能**的选择组合”（如选0个、1个、...、 $n$ 个元素）时，可用 **$n$ 位二进制数**表示一种组合：

二进制数的第 $i$ 位（从右数，0开始）为1  $\rightarrow$  选中第 $i$ 个元素； 为0  $\rightarrow$  不选第 $i$ 个元素。

## 2. 步骤（以 $n=3$ 个元素为例）

总共有 $2^3=8$ 种组合（二进制数从000到111）；

遍历所有数（0~7），对每个数检查每一位的状态：

000 (0)  $\rightarrow$  全不选；

001 (1)  $\rightarrow$  选第3个； **第0位**

010 (2)  $\rightarrow$  选第2个； **第1位**

011 (3)  $\rightarrow$  选第2、3个； **第0, 1位**

...

111 (7)  $\rightarrow$  选第1、2、3个。 **第0, 1, 2位**





## 二进制枚举

3. 关键操作：如何判断“第*i*位是否为1”？

用按位与（&）运算：对于数*x*，若  $x \& (1 \ll i) \neq 0$ ，则第*i*位为1（选中）。

例：  $x=5$ （二进制101）：

判断第0位：  $1 \ll 0 = 1$ （001）

$101 \& 001 = 001 \neq 0$

→ 第0位为1

判断第1位：  $1 \ll 1 = 2$ （010）

$101 \& 010 = 000$

→ 第1位为0。

03

# 经典例题

TEACHING  
COURSEWARE

TEACH

## 经典例题

### 子集和问题

现给定一个正整数集合  $S$  和正整数  $k$ ，编程求解有多少个  $S$  的子集的和等于  $k$ 。

#### 输入格式：

第一行有2个正整数， $n$ 和 $k$  ( $1 \leq n \leq 20$ ， $1 \leq k \leq 10^7$ )；

$n$  表示  $S$  的元素个数， $k$ 是子集和的目标值；

第二行有  $n$  个正整数，表示集合  $S$  中的元素，保证所有元素的和在 `int` 范围内。

#### 输出格式：

子集和为 $k$ 的子集的数量

样例输入：

5 10

2 2 6 5 4

样例输出：

2



$\{2, 2, 6, 5, 4\}$

Question1:

我们能否用普通枚举解决这个问题？

Question2:

手动模拟子集和的过程，  
选哪些是一种解。

# 代码实现

```
#include<bits/stdc++.h>
using namespace std;
int a[25];
int main()
{
    int n, k; // n: 元素个数; k: 目标和
    cin >> n >> k; // 输入元素个数和目标和
    int ans = 0; // 用于统计和为k的子集数量
    for(int i = 0 ; i < n ; i++)
    {
        cin >> a[i];
    }
}
```

```
int nums = 1 << n; // 1 << n 等价于 2^n
// 枚举所有可能的子集（用二进制数num表示一个子集）
// num的二进制位从低到高（第0位到第n-1位）对应数组a[0]到a[n-1]
for(int num = 0 ; num < nums ; num++)
{
    int sum = 0; // 存储当前子集的元素和
    // 遍历num的每一位，判断是否选中对应元素
    for(int i = 0; i < n ; i++)
    {
        // 核心位运算：判断num的第i位是否为1
        // 1 << i 表示“只有第i位为1的二进制数”，与num按位与后非0，说明第i位为1
        if(num & (1 << i))
        {
            sum += a[i]; // 若第i位为1，累加对应元素a[i]
        }
    }
}
```

```
// 若当前子集和等于目标值k
if(sum == k)
{
    ans++; // 计数+1
    // 输出当前子集对应的二进制表示 (0表示未选中, 1表示选中)
    // 从第0位到第n-1位输出, 与数组a[0]到a[n-1]顺序对应
    for(int i = 0; i < n ; i++)
    {
        if(num & (1 << i)) // 第i位为1, 输出1
            cout << 1;
        else // 第i位为0, 输出0
            cout << 0;
    }
    cout << endl;
}
cout << ans << endl; // 输出和为k的子集总数量
return 0;
```

04

Q&A

TEACHING  
COURSEWARE

TEACH



1. 普通枚举和二进制枚举的区别？
2. 二进制枚举适用于什么情况？（从二进制特性出发）
3. 二进制枚举的随着规模变大，运行时间增长的原理是什么（从解空间的角度出发）