The background is a traditional Chinese ink wash painting. It depicts a vast landscape with layered, misty mountains in shades of green and blue. A calm river flows through the center, reflecting the sky and mountains. In the lower-left foreground, a small red boat with a person is on the water. Several birds are scattered across the scene: two large white cranes with black wings and red heads are flying in the upper right, and several smaller birds are in flight throughout the sky. A large, bright red sun is positioned in the upper left corner. The overall style is serene and atmospheric.

# 数据结构与算法设计第四 章串



A traditional Chinese ink wash painting of a landscape. The scene features misty, layered mountains in shades of green and blue, a calm lake in the foreground, and a large, bright red sun in the upper left corner. Several birds are depicted in flight across the sky. The overall style is soft and atmospheric, typical of classical Chinese art.

# 目录

- 串的基本概念
- 串的模式匹配算法
- 串的排序与索引
- 串的应用



01

# 串的基本概念







# 串的定义



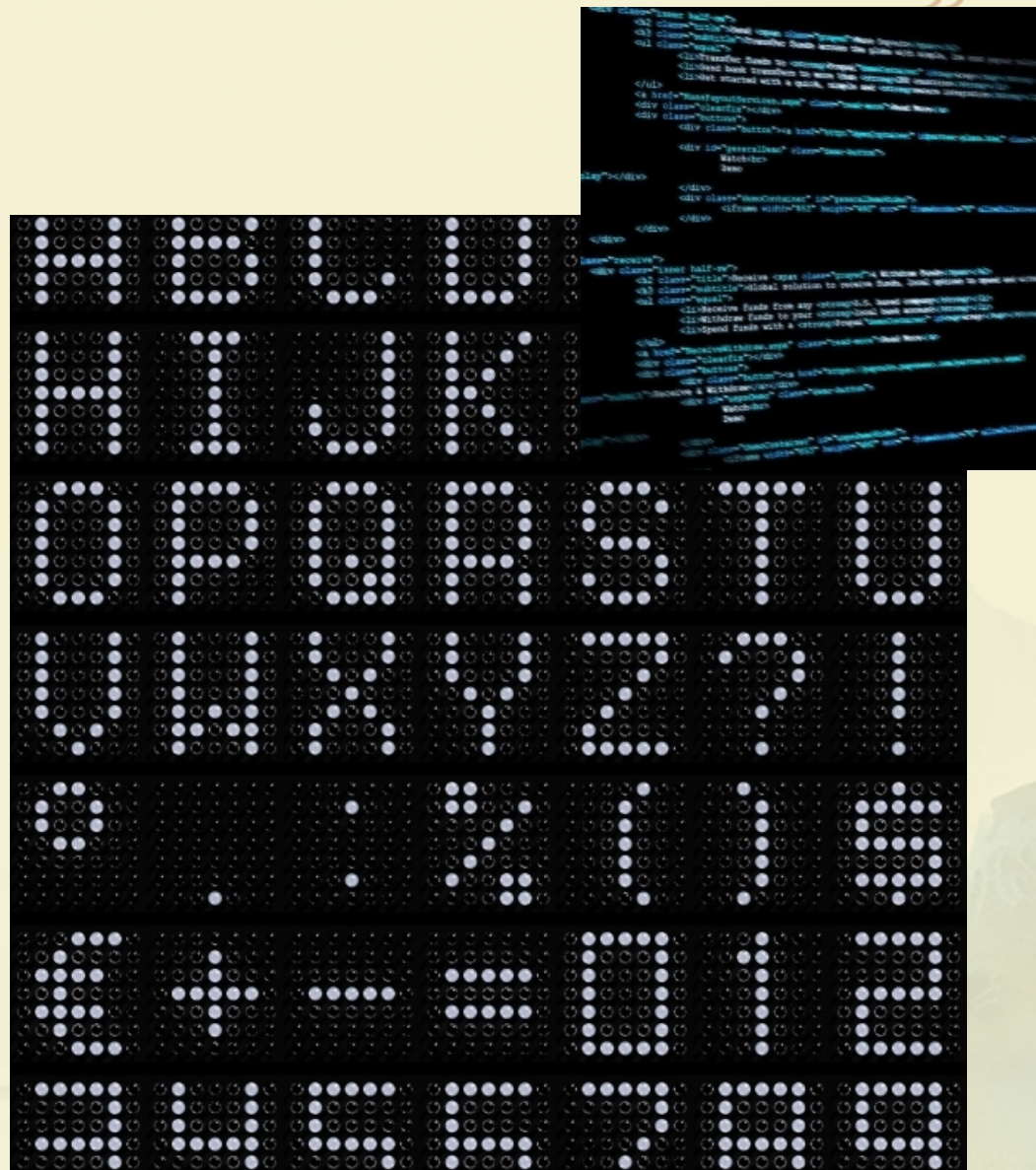
串是由零个或多个字符组成的有限序列。



串的长度是指串中字符的个数。



空串是指长度为零的串。





# 串 的 表 示 和 存 储



```
while (p < sizeof(unsigned int) && count > 0) {
    if (put_user*((tchar *H)sample_step+p), buf)
        return -EFAULT;
    buf++; p++; count--; read++;
}
pnt = (char *)jorof - buffer + p - sizeof(atomic_t);
if (copy_to_user(buf, (void *)pnt, count))
    return -EFAULT;
read = count;
*ppos = read;
return read;
}

/*
 * Writing to /proc/profile resets the counters
 * Writing a profiling_multiplier value into it also sets the profiling
 * interrupt frequency, on architectures that support this.
 */
static ssize_t write_profile(struct file *file, const char __user *buf,
                             size_t count, loff_t *ppos)
{
    if (CONFIG_SMP
        extern int setup_profiling_timer(unsigned int multiplier);

    if (count == sizeof(int)) {
        unsigned int multiplier;
        if (copy_from_user(&multiplier, buf, sizeof(int)))
            return -EFAULT;
        if (setup_profiling_timer(multiplier))
            return -EINVAL;
    }
    return count;
}

profile_discard_fifo_buffers();
memset(profile_buffer, 0, profilen * sizeof(atomic_t));
return count;
}

10 EPS VECTOR BACKGROUND
```

## 字符数组

使用字符数组来存储串，每个字符占用一个数组元素的空间

。

## 动态分配

使用动态内存分配函数（如malloc、calloc等）来为串分配内存空间。



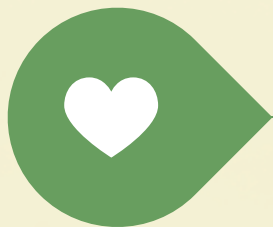


# 串的基本操作



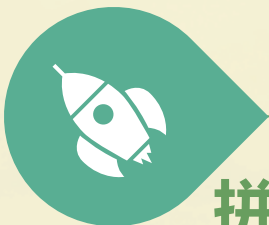
## 初始化

为串分配内存空间并赋初值。



## 赋值

将一个串的值赋给另一个串。



## 拼接

将两个串连接起来形成一个新的串。



## 比较

比较两个串是否相等或按照某种顺序进行比较。





02

# 串的模式匹配算法





# 朴素模式匹配算法



01

时间复杂度： $O(n*m)$ ，其中 $n$ 为主串长度， $m$ 为模式串长度。

02

空间复杂度： $O(1)$ 。

03

适用场景：适用于模式串较短的情况。





# KMP算法



1

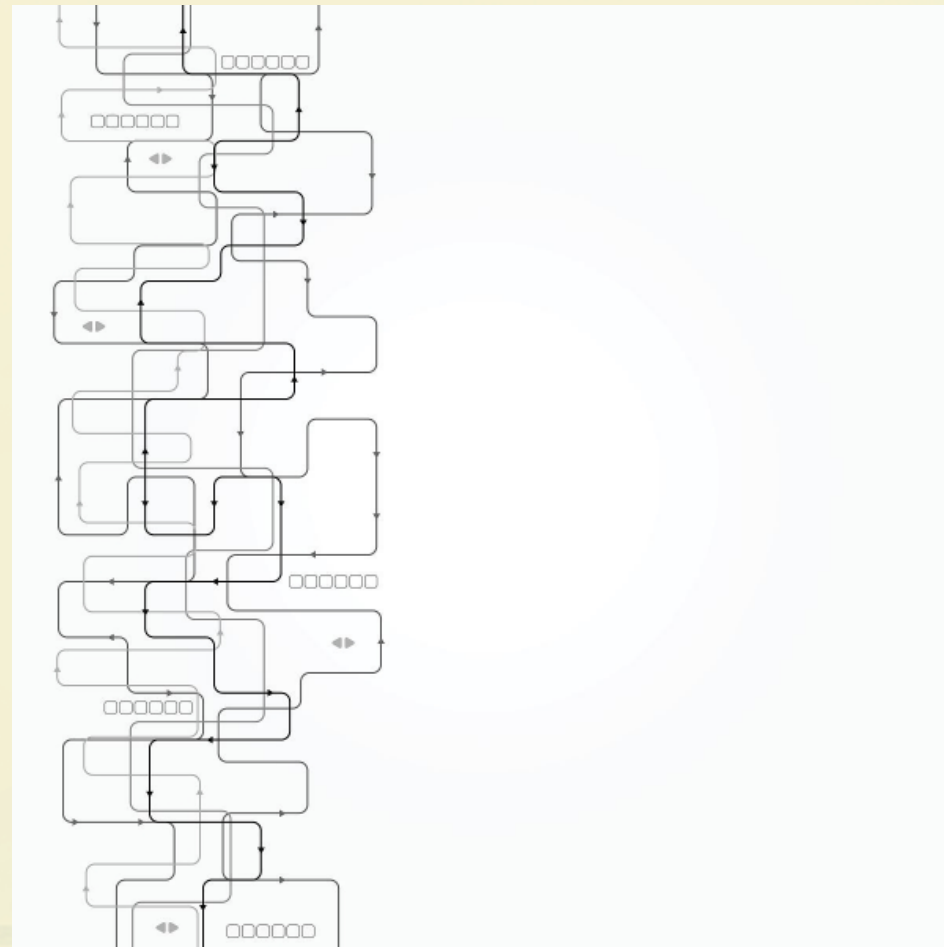
时间复杂度： $O(n+m)$ ，其中 $n$ 为主串长度， $m$ 为模式串长度。

2

空间复杂度： $O(m)$ 。

3

适用场景：适用于模式串较长的情况。





# BM算法



01

时间复杂度： $O(n/m)$ ，其中 $n$ 为主串长度， $m$ 为模式串长度。

02

空间复杂度： $O(m)$ 。

03

适用场景：适用于模式串较长且主串中存在大量重复字符的情况。



以上内容仅为本文档的试下载部分，为可阅读页数的一半内容。如要下载或阅读全文，请访问：  
<https://d.book118.com/667134006001006060>