第四章 串

2022年9月30日 星期五 14:06

串的定义和实现【2023年考点不考这个，了解即可】

定义	• 串：字符串简称串，是由零个或多个字符组成的有限序列 • 子串：串中任意多个连续的字符组成的子序列 • 主串：包含子串的串 • 某个字符在串中的序号称为该字符在串中的位置 • 空格串：由一个或多个空格组成的串

定长顺序存储表示	`#define MAXLEN 255`
	`typedef struct`
	`{`
	`  char ch[MAXLEN];`
	`  int length;`
	`} SString;`

堆分配存储表示	`typedef struct`
	`{`
	`  char *ch;`
	`  int length;`
	`} SString;`

块链存储表示	• 类似于线性表的链式存储结构 • 也可采用链表方式存储串值其中节点称为块 • 最后一个节点占不满时，用#填充
基本操作	• StrAssign(&T,chars):赋值操作 • StrCopy(&T,S):赋值操作 • StrEmpty(S):判空操作 • StrCompare(S,T):比较操作 • StrLength(S):求串长 • StrString(&Sub,S,pos,len):求子串 • Concat(&T,S1,S2):串联接 • Index(S,T):定位操作 • ClearString(&S):清空操作 • DestoyString(&S):销毁串

串的模式匹配

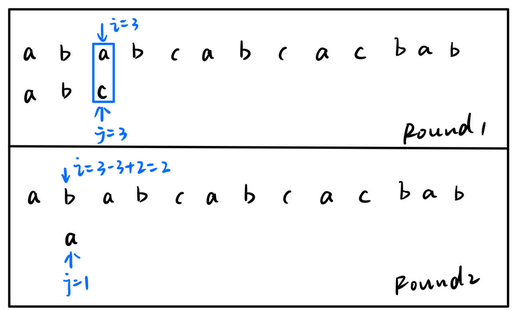
- 模式匹配：子串的定位操作，求的是子串（模式串）在主串中的位置

暴力匹配算法-BF算法

时间复杂度为O(mn)

① 最好情况，平均时间复杂度 O(m+n) [i,j不用回退]

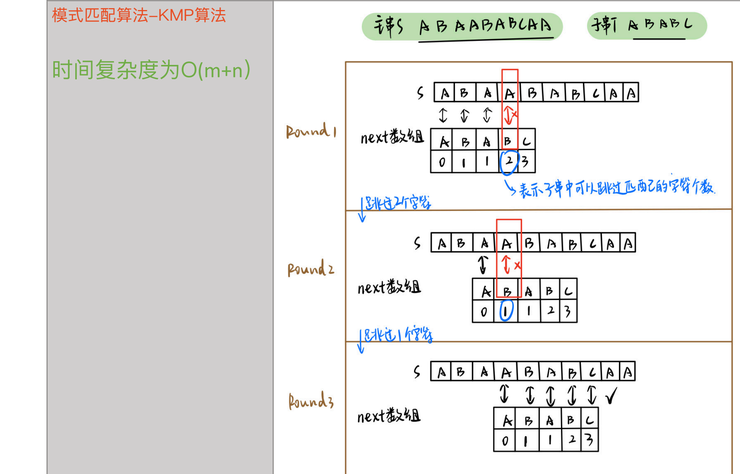
② 最坏情况，平均时间复杂度 O(mn) [i,j一直在回退]



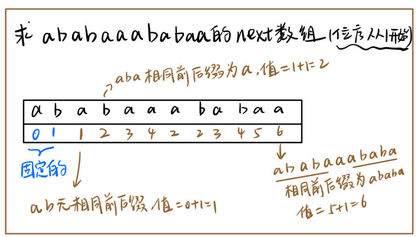

```
int Index_BF(string s, string t)
{
    int i = 1, j = 1;
    int lens = s.length();
    int lent = t.length();
    while (i < lens && j < lent)
    {
        if (s[i] == t[j])
        {
            i++, j++;
            continue;
        }
        else
        {
            i = i - j + 2; // i指示主串S正在比较的字符位置
            j = 1; // j指示子串t正在比较的字符位置
        }
    }
    if (j == lent)
    {
        return i - j;
    }
    return -1;
}
```

模式匹配算法-KMP算法

时间复杂度为O(m+n)



next数组的值 = 寻找子串中最大相同前后缀的长度+1




```
void get_next(String T, int next[]) //计算next数组
{
    int i = 1, j = 0;
    next[1] = 0;
    while (i < T.length)
    {
        if (j == 0 || T.ch[i] == T.ch[j])
        {
            ++i;
            ++j;
            next[i] = j;
        }
        else
            j = next[j];
    }
}
int Index_KMP(String S, String T, int next[])
{
    int i = 1, j = 1;
    while (i <= S.length && j <= T.length) //i永远不递减
    {
        if (j == 0 || S.ch[i] == T.ch[j]) //字符匹配成功,指针后移
        {
            ++i;
            ++j;
        }
        else //字符匹配失败, 根据next跳过前面的一些字符
            j = next[j];
    }
    if (j > T.length) // 匹配成功
        return i - T.length;
    else
        return 0;
}
```