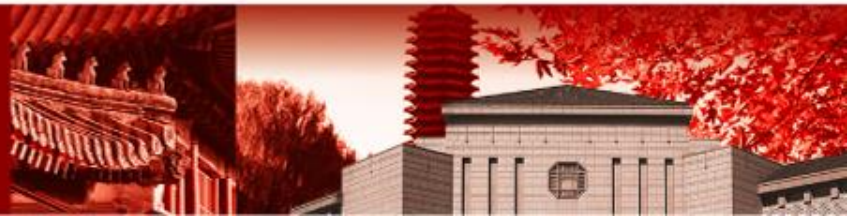
数据结构与算法B

03 – 字符串

目录

- 3.1 字符串基本概念
- 3.2 Python中的字符串
- 3.3 模式匹配算法
  - 朴素模式匹配算法
  - 无回溯模式匹配(KMP)算法

3.1 字符串的基本概念



字符串示例

- s1=“123”
- s2=“Data Structure and Algorithm”
- s3=“Hello World”
- s4=“BB ”
- s5=“”

字符串：简称为串，是零个或多个字符组成的有限序列。一般记为：s=“$s_0s_1\ldots s_{n-1}$”（$n\geq1$），s为串名，每个字符s_i ($0 <= i <= n-1$)可以是字母、数字或其它字符

术语表

- **串的长度**：字符串中字符个数
- **空串**：长度为零的字符串，记为s = “”
  - 注意与空格串 “ ” 的区别
- **子序列**：字符串s1中，任意个字符按顺序组成的序列s2称为s1的子序列
  - s1=“Hello world”, s2=“Helo word”
- **子串与主串**：字符串s1中任意个连续字符组成的序列s2称为s1的子串，称s1为s2的主串。
  - s1=“Hello world”, s2=“Hello”
  - 空串是任意串的子串; 除s本身之外，s的其他子串称为s的真子串。

术语表

- **字符在串中的位置**：该字符在串中第一次出现时的位置。
- **子串在主串中的位置**：该子串在串中第一次出现时，第一个字符的位置。
- **两个字符串相等的充分必要条件**：长度相等，且对应位置上字符相同。

字符串是一种特殊的线性结构

- 字符串与一般线性表的区别与联系
  - 串的数据对象约束为字符集，其每个结点包含一个字符；
  - 线性表的基本操作大多以“单个元素”为操作对象，而串的基本操作通常以“串的整体”作为操作对象；
  - 线性表的存储方法同样适用于字符串，在选择存储结构时，应根据不同情况选择合适的存储表示。

3.2 Python中的字符串

Python中的字符串

- Python中，str类型是不可变类型
  - 字符串可以用单引号 '、双引号 " 或三引号 ''' /""" 创建。

    例如：s1 = 'Hello‘
    s2 = "World"
    s3 = """This is a multi-line
    string"""
  - 不可变类型的变量，一经创建，不可修改其变量
  - 回忆：同样作为容器的类型，list是可变类型，tuple则是不可变类型
- 那么，Python如何实现各种字符串操作？
  - 一些操作不需要修改字符串：切片（如s[1:4]）、查找子串（如s.find("is")
  - 对于修改操作，通过创建新的字符串来实现对字符串的“修改”
  - 最常见的拼接操作，即创建一个新的字符串
    - 例如，S = “Python ” + “is ” + “awesome”
    - 拼接大量字符串时，造成性能下降
    - 使用join运算：S = “ ”.join([“Python”, “is”, “awesome”])

Python中的字符串

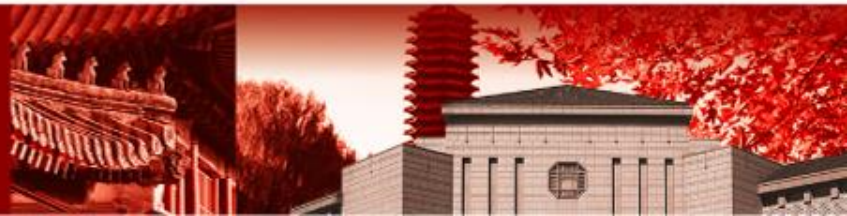
- Python中str类型的常用方法
  - __len__ 返回串的长度
  - __add__ 两个字符串的拼接
  - __eq__, __gt__, __lt__ 字符串的相等/大于/小于运算（字典序）
  - split 以指定分隔符（默认空格）分割字符串
  - join 将序列中的元素以指定字符串连接
  - strip 移除字符串两侧的空格
  - find 在字符串中检索子串

示例1

```
>>> s2 = "Alice, Bob, Carol, Dave, Eve"
>>> s2.split(",")
['Alice', ' Bob', ' Carol', ' Dave', ' Eve’]

>>> [name.strip() for name in s2.split(",")]
['Alice', 'Bob', 'Carol', 'Dave', 'Eve’]

>>> ", ".join(["Alice", "Bob", "Carol", "Dave", "Eve"])
'Alice, Bob, Carol, Dave, Eve'
```

示例2

```
>>> s1 = "Find a substring in string s1"
>>> s1.find("substring")
7
>>> s1.find("string s1")
20
>>> s1.find("string s2")
-1
```

- 如何高效实现find操作？
- 本节课来重点探讨