5.1　知识点定位

青少年编程能力“Python 四级”核心知识点 4：匹配算法。

5.2　能 力 要 求

能够解释并实现暴力匹配（BF）算法、KMP 算法、BM 算法，掌握三种字符串匹配算法的时间复杂度和空间复杂度计算方法。

5.3　建议教学时长

本单元建议 6 课时。

5.4　教 学 目 标

1.　知识目标

本单元主要学习三种常见字符串匹配算法的运行原理和代码实现，帮助学习者理解暴力匹配、KMP、BM 算法的算法思想，掌握各匹配算法的时间复杂

度和空间复杂度。

2. 能力目标

学习者能够根据应用场景和数据特征选择合适的算法完成字符串匹配操作。

3. 素养目标

通过对各字符串匹配算法的学习，理解算法中蕴含的工程学思想和求解策略，感受算法在社会生活和工作中发挥的重要作用，通过对算法从感性到理性的认知提升过程，获得对计算机科学的探知欲望。

5.5 知识结构

本单元知识结构如图 5-1 所示。

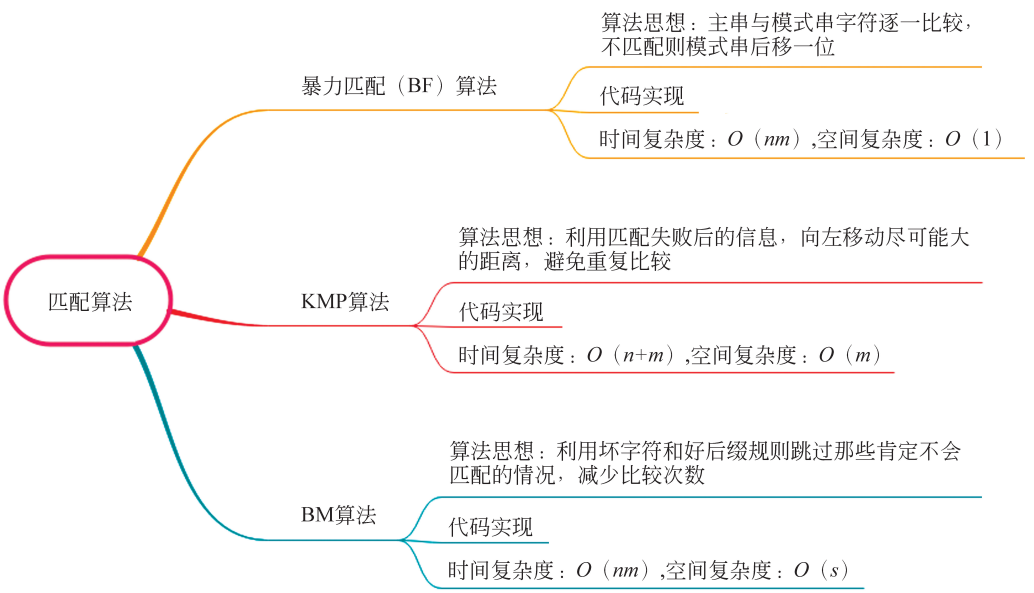

图 5-1　匹配算法知识结构

5.6 补充知识

1. 常见字符串匹配算法

除教材中提及的暴力匹配（BF）算法、KMP 算法、BM 算法外，常见的匹配算法还有 Rabin-Karp（哈希检索）算法、有限自动机算法（Finite Automation）、Simon 算法、Colussi 算法、Galil-Giancarlo 算法、Apostolico-Crochemore 算法、Horspool 算法和 Sunday 算法等。

2. Rabin-Karp（哈希检索）算法

RK 算法的全称为 Rabin-Karp 算法，用两位发明者 Rabin 和 Karp 的名字来命名。RK 算法是在 BF 算法的基础上作了一些改进：通过字符串哈希值比较代替字符串的遍历比较。该算法的核心思想就是通过比较两个字符串的哈希值来判断是否包含对方。RK 算法也可以进行多模式匹配，在论文查重等实际应用中一般都是使用此算法。

算法描述：

步骤 1：首先计算模式串的哈希值。

步骤 2：分别从主串中取与模式串长度相同的多个子串，计算各子串的哈希值。

步骤 3：比较模式串哈希值与子串哈希值两者是否相等，如果哈希值不同，则两者必定不匹配。如果相同，由于哈希冲突存在，也需要按照 BF 算法再次判定。

举例说明：

主串 S 为“ABCDEFG”，模式串 P 为“DEF”，如图 5-2 所示，使用 RK 算法进行匹配的过程如下。

（1）首先计算子串“DEF”哈希值为 Hd，之后从原字符串中依次取长度为 3 的字

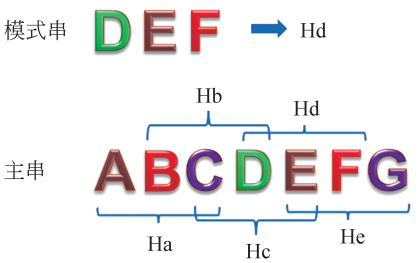

图 5-2　计算模式串和各子串哈希值

符串“ABC”“BCD”“CDE”“DEF”计算哈希值，分别为 Ha、Hb、Hc、Hd。

（2）依次将各子串哈希值与 Hd 进行比较，当哈希值相等时，仍然要比较一次子串“DEF”和原字符串“DEF”是否一致。

算法复杂度：

RK 算法相较 BF 提高了字符串的比较效率，但其算法的时间复杂度仍与 BF 算法相同，时间复杂度为 $O(mn)$（实际应用中往往较快，期望时间为 $O(m+n)$）。这是因为计算每一个连续子串的哈希值也有时间消耗，且跟遍历比较的时间复杂度一样都是 $O(m)$，所有子串的哈希值的计算时间复杂度为 $O(mn)$，因此，RK 算法的整体时间复杂度还是 $O(mn)$，进一步的优化点可以放在哈希值的计算上。

算法补充说明如下：

（1）RK 算法可以通过优化字符串累加方法来提升运行效率，即从第二个子串开始，每个子串的哈希值都可以通过计算上一个哈希值的增量得到。

（2）通过使用优化方法，最终 RK 算法的时间复杂度可以优化为 $O(n)$。

（3）与 BF 算法相比，免去了很多无谓的字符比较，时间复杂度上有很大提高。

（4）RK 算法的缺点在于哈希冲突，每一次哈希冲突时都要进行子串和模式串的逐个比较，如果冲突过多，RK 算法就会退化成 BF 算法。

3. Sunday 算法

Sunday 算法由 Daniel M.Sunday 在 1990 年提出，其思想是从前往后匹配，匹配失败时关注的不是失配位，而是主串中参加匹配的最末位字符的下一位字符。

Sunday 算法的匹配策略可以归结为以下两点。

（1）如果该字符没有在模式串中出现，则在下一轮比较时直接跳过，即：移动位数 = 模式串长度 + 1。

（2）如果该字符在模式串中出现过，则移动位数 = 模式串长度 – 该字符最右出现的位置（以 0 开始）= 模式串中该字符最右出现的位置到尾部的距离 + 1。

举例说明：

主串 S 为“abdeababca”，模式串 P 为“abc”，如图 5-3 所示，使用 Sunday 算法进行匹配的过程如下。

图 5-3　第一轮比较

第一轮比较：S[2]!=P[2], 发生失配，那

么就关注主串中参与匹配的最末位字符的下一位，也就是 S[3]，因为 'e' 不存在于模式串 P 中，那么就将模式串移动 4 位 (模式串长度 +1)。

第二轮比较：S[6]!=P[2] 发生失配，关注的字符为 S[7]('b'), 它存在于模式串 P 中，移动模式串使得 S[7] 与该字符在模式串中最右的位置匹配上，即移动 2 位，如图 5-4 所示。

第三轮比较：匹配成功，如图 5-5 所示。

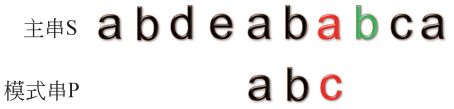

图 5-4　第二轮比较

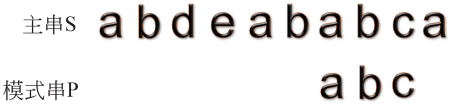

图 5-5　第三轮比较

算法复杂度：

Sunday 算法最优情况的时间复杂度为 $O(n/m)$，最差情况的时间复杂度为 $O(mn)$。假设主串是“aaabaaabaaabaaab”，模式串是“aaaa”。使用 Sunday 算法进行匹配时，在大部分情况下，关注字符都是在模式串中存在的 (模式串只有 'a' 一个字符), 换言之，在大部分情况下模式串都只能移动 1 位，时间复杂度变为 $O(mn)$。在实际使用中，Sunday 算法比 BM 算法略优。

5.7　教学组织安排

教 学 环 节	教 学 过 程	建议课时
知识导入	以虚拟人物间的对话为引，抛出有关“字符串匹配算法”的相关话题，引导学生自主思考字符串匹配实现方法，激发学习兴趣，为后续知识点引入作铺垫	2 课时
暴力匹配（BF）（原理讲授）	对算法的实现原理进行简要介绍，结合案例对算法的操作过程进行实例讲解	
暴力匹配（BF）（代码实现）	(1) 对暴力匹配算法的代码实现进行讲解。 (2) 布置课堂练习，对学生进行辅导，使其当堂完成代码编写	
暴力匹配（BF）（算法复杂度分析）	对暴力匹配算法的时间复杂度和空间复杂度分析方法进行讲解	

续表

教 学 环 节	教 学 过 程	建议课时
KMP 算法 （原理讲授）	（1）对算法的实现原理进行简要介绍，讲解的要点如下。 ① 最长公共前后缀的确定方法。 ② Next 表的计算方法。 （2）结合案例对算法的操作过程进行实例讲解，帮助学生完成算法思想的内化吸收	2 课时
KMP 算法 （代码实现）	（1）对 KMP 算法的实现方法进行讲解。 （2）布置课堂练习，对学生进行辅导，使其当堂完成代码编写	
KMP 算法 （算法复杂度分析）	对 KMP 算法的时间复杂度和空间复杂度分析方法进行讲解	
BM 算法 （原理讲授）	（1）对 BM 算法的实现原理进行简要介绍，BM 算法原理讲解的要点在于：与 KMP 算法相比，BM 算法借助“好后缀规则”和“坏字符规则”能够更高效地移动模式串。 （2）结合案例对算法的实现原理和操作过程进行实例讲解，引导学生自主完成算法操作步骤的总结、凝练，让学生体会到 BM 算法的优势所在	2 课时
BM 算法 （代码实现）	（1）对 BM 算法的代码实现进行讲解。 （2）布置课堂练习，对学生进行辅导，使其当堂完成代码编写	
BM 算法 （算法复杂度分析）	对 BM 算法的时间复杂度和空间复杂度分析方法进行讲解	
单元总结	对本讲知识进行总结归纳，布置课后作业和拓展阅读内容	

5.8 教学实施参考

1. 讨论式知识导入

以老师与小萌、小帅的对话为引，引导学生自主思考字符串匹配的实现方法，暴力匹配法最符合人类的思维习惯，根据学生回答自然过渡到暴力匹配算法的讲解。

2. 知识点一：暴力匹配的算法思想

（1）结合示例对暴力匹配的算法思想进行讲解，其核心思想是“主串与模式串字符逐一比较，不匹配则模式串后移一位，开始新一轮的比较”。

（2）暴力匹配算法原理简单粗暴，但因其与人类思维习惯一般无二，所以具有易于理解和实现的优点。

3. 知识点二：暴力匹配的代码实现

对暴力匹配算法的核心关键代码进行讲解，配合练习题帮助学生完成知识内化吸收。

4. 知识点三：暴力匹配的算法复杂度计算

（1）假设主串长度为 n，模式串长度为 m，主串中的每个字符都可能与模式串中的字符进行一一比较，所以其时间复杂度为 $O(nm)$。

（2）当 n 和 m 值较小时，算法效率尚可，但当 m 和 n 值较大时，算法效率下降明显。

（3）暴力匹配算法的空间复杂度为 $O(1)$。

5. 知识点四：KMP 的算法思想

（1）KMP 算法的核心思想是“利用匹配失败后的信息，向右移动尽可能大的距离，避免重复比较”。

（2）KMP 算法能在发生不匹配时打破后移一位的限制，利用最长公共前后缀计算得到 Next 表，根据 Next 表动态确定模式串后移的距离，移动到下一个最可能发生匹配的位置，从而大大提升算法的匹配效率。

6. 知识点五：KMP 算法最长公共前后缀的计算方法

（1）对前缀、后缀、公共前后缀的概念进行讲解，结合示例，对不同字符组合情况下公共前后缀计算过程进行演示，帮助学生理解。

（2）注意强调：在出现多个公共前后缀时，选择最长的那一对。

7. 知识点六：KMP 中 Next 表的计算方法

（1）Next 表的计算与主串无关，仅根据“不匹配字符自身”和“最长公共前后缀”就可以单方面完成计算。

（2）在匹配算法运行前，就可以完成 Next 表的计算，该阶段又称为预处理阶段。

（3）为了便于学生理解，讲解应从“发生不匹配时的移动方案”入手，举例说明如何得到每个字符失配时的移动方案，最终得出规律：如果最长公共前后缀的长度为 n，那么移动方案就是“n+1 号位与主串当前位比较”。

（4）在此基础上，利用每个字符的移动方案平滑引出 Next 表的计算方法。

（5）对 Next 表的作用进行着重强调，即 Next 表指明了当特定字符发生不匹配时，模式串要移动到的位置。

8. 知识点七：KMP 的代码实现

对 KMP 算法的核心关键代码进行讲解，讲解主要分为两部分：Next 数组的构建和 KMP 字符串匹配主函数。因代码相对比较复杂，可要求学生参照教材案例完成练习题，在模仿的过程中逐渐完成知识的内化吸收。

9. 知识点八：KMP 的算法复杂度计算

（1）设主串长度为 n, 模式串长度为 m。预处理阶段的时间复杂度为 $O(m)$，空间复杂度为 $O(m)$；匹配过程中主串不回溯，比较次数可记为 n，因此 KMP 算法的总时间复杂度为 $O(m+n)$，空间复杂度记为 $O(m)$。

（2）相较暴力匹配算法，KMP 算法通过一点点的空间消耗换得了极高的时间提速，这是空间换时间思想的集中体现。

10. 知识点九：BM 的算法思想

（1）BM 算法的核心思想是“利用坏字符和好后缀规则跳过那些肯定不会匹配的情况，减少比较次数”。

（2）BM 算法在移动模式串的时候是从左到右，而进行比较的时候是从右到左，即从模式串的尾部开始匹配。

（3）实现跨越式移动的关键在于“坏字符”和“好后缀”规则，两个规则都会对模式串的移动距离造成影响，BM 算法向右移动模式串的距离就是取坏字符和好后缀算法得到的最大值。

知识点十：BM 算法的“坏字符规则”

（1）通过举例对坏字符规则的两种应用场景进行讲解。

（2）帮助学生总结出坏字符规则下的模式串移动距离计算公式。

知识点十一：BM 算法的“好后缀规则”

（1）通过举例对好后缀规则的三种应用场景进行讲解。

（2）帮助学生总结出好后缀规则下的模式串移动距离计算公式。

知识点十二：BM 算法的代码实现

对 BM 算法的核心关键代码进行讲解，讲解主要分为三部分：好后缀表的构建、坏字符表的构建和 BM 字符串匹配主函数。因代码相对比较复杂，可要求学生参照教材案例完成练习题，在模仿的过程中逐渐完成知识的内化吸收。

知识点十三：BM 的算法复杂度计算

（1）设主串长度为 n, 模式串长度为 m。其预处理阶段的时间复杂度为 $O(m+s)$，空间复杂度为 $O(s)$，s 是与模式串和子串相关的有限字符集长度。

（2）搜索阶段，在最好的情况下，总的比较次数为 $(n–m)/m$, 此时的时间复杂度为 $O(n/m)$；在最坏的情况下，比较次数为 $(n–m)/2(m–2)$，也就是 $O(mn)$，因此，BM 算法的时间复杂度是 $O(mn)$。

（3）从表面上看 KMP 算法拥有 $O(m+n)$ 的时间复杂度，BM 的时间复杂度为 $O(mn)$，但多篇学术文献的研究结果表明，在实际运行效率上反而是 BM 算法更高，这是由于 BM 算法在实际运行过程中经常能达到算法的平均效率，甚至最好情况下达到 $O(n/m)$ 的时间复杂度。