

W14-董天宇-2024244049-qa

Q01 子串搜索

- prompt: 什么是子串搜索 (Substring searching) ?
- model_name: 豆包
- model_version: pro-256k
- response:

1. 定义

- 子串搜索 (Substring Search), 也称为字符串匹配 (String Matching), 是在一个较长的字符串 (通常称为文本串, 用T表示) 中查找一个较短的字符串 (通常称为模式串, 用P表示) 出现的位置的操作。例如, 在文本串“abcdefg”中搜索模式串“cde”, 子串搜索的任务就是确定“cde”在“abcdefg”中首次出现的位置 (在这个例子中是位置3)。

2. 应用场景

- **文本编辑软件**: 如在文档中查找特定的单词或短语, 用于查找和替换功能。当用户想要将文档中所有的“apple”替换为“banana”时, 软件首先需要通过子串搜索找到“apple”在文档中的所有位置。
- **生物信息学**: 在DNA序列分析中, DNA序列可以看作是由A、C、G、T四种碱基组成的长字符串。子串搜索可以用于查找特定的基因序列模式, 这对于研究基因的功能、疾病的遗传因素等方面非常重要。例如, 在一条很长的DNA序列中寻找特定的致病基因片段。
- **网络搜索引擎**: 搜索引擎需要在网页的文本内容中查找用户输入的关键词。当用户在搜索引擎中输入“人工智能发展历程”, 搜索引擎会在其索引的网页文本中进行子串搜索, 找到包含这些关键词的网页, 并根据相关性等因素对搜索结果进行排序。

3. 常见算法

• 朴素算法 (Brute - Force Algorithm)

- 这是最直接的方法。它从文本串T的第一个字符开始, 逐个字符地与模式串P进行比较。如果当前字符匹配, 就继续比较下一个字符; 如果不匹配, 就将文本串的比较位置向后移动一位, 重新开始比较。例如, 对于文本串T = "abcdef"和模式串P = "cde", 首先比较T[0]和P[0] (即'a'和'c'), 不匹配, 然后比较T[1]和P[0] (即'b'和'c'), 也不匹配, 直到比较T[2]和P[0] (即'c'和'c') 匹配, 接着比较T[3]和P[1]、T[4]和P[2], 如果都匹配, 就找到了模式串在文本串中的位置。这种算法简单易懂, 但在最坏情况下, 时间复杂度为 $O(m \times n)$, 其中m是文本串的长度, n是模式串的长度。

• KMP算法 (Knuth - Morris - Pratt Algorithm)

- KMP算法是一种更高效的字符串匹配算法。它利用了模式串本身的特点, 通过一个前缀函数 (也称为失配函数) 来减少不必要的比较。例如, 当在文本串中匹配模式串时, 如果在某个位置失配, KMP算法可以根据前缀函数快速确定模

式串应该向后移动多少位，而不是像朴素算法那样只移动一位。KMP算法的时间复杂度为 $O(m + n)$ ，其中 m 是文本串的长度， n 是模式串的长度，在处理长文本和模式串时效率明显高于朴素算法。

- **Boyer - Moore算法**

- 这种算法从模式串的末尾开始比较，并且根据模式串中字符出现的频率和位置来决定文本串的移动距离。它有两个主要的启发式规则：坏字符规则和好后缀规则。坏字符规则是指当在比较过程中发现文本串中的某个字符（坏字符）与模式串中对应的字符不匹配时，根据这个坏字符在模式串中的位置来决定模式串应该向右移动多远。好后缀规则是当在模式串的后缀部分找到匹配的部分（好后缀），但前面部分不匹配时，根据好后缀的情况来确定模式串的移动距离。Boyer - Moore算法在实际应用中通常有很好的性能，尤其是在处理字母表较大的情况时。其时间复杂度在最好情况下可以达到 $O(n/m)$ ，其中 n 是文本串的长度， m 是模式串的长度，但在最坏情况下时间复杂度为 $O(m \times n)$ 。

Q02 序列比对

- prompt: 序列比对 (Sequence alignment) 在生物信息学中的作用是什么？
- model_name: 豆包
- model_version: pro-256k
- response: 序列比对是生物信息学中的一种基本方法，在多个方面发挥着关键作用：

一、相似性搜索与功能预测

1. 基因和蛋白质功能推断

- 许多基因和蛋白质在进化过程中是保守的。通过序列比对，可以发现与已知功能基因或蛋白质序列高度相似的未知序列。例如，如果一个新发现的基因序列与一个已知的编码酶的基因序列有很高的相似性（比如通过核苷酸序列比对，相似性达到80%以上），那么可以推测这个新基因可能也具有类似的酶功能。
- 对于蛋白质序列，结构和功能通常是紧密相关的。当一个蛋白质序列与另一个功能已知的蛋白质序列比对后发现高度匹配，特别是在一些关键的功能结构域上匹配，就可以为其功能注释提供有力的线索。比如，在细胞信号转导通路中，一些受体蛋白的功能结构域是高度保守的。通过序列比对找到这些结构域的相似性，能够推断新发现蛋白质是否也属于受体蛋白家族以及可能参与的信号通路。

2. 寻找同源序列

- 序列比对有助于识别同源序列，包括直系同源和旁系同源。直系同源序列是指不同物种中起源于共同祖先基因的序列，它们通常具有相似的功能。例如，人类的胰岛素基因和小鼠的胰岛素基因是直系同源的。通过比对它们的序列，可以了解在进化过程中基因序列的变化情况，并且可以利用小鼠模型来研究人类胰岛素相关疾病。
- 旁系同源序列是在同一物种基因组内由于基因复制事件产生的相似序列。通过序列比对可以发现这些旁系同源基因，研究它们在功能分化上的情况。例如，在植物基因组中，往往存在多个基因家族成员，这些旁系同源基因可能在不同的组织、发育阶段或者环境胁迫响应中发挥不同的功能。通过序列比对来分析它们的差异，可以更好地理解基因家族的进化和功能多样性。

二、进化分析

1. 构建系统发育树

- 序列比对是构建系统发育树的基础步骤。通过对多个物种的基因或蛋白质序列进行比对，计算序列之间的差异程度（如通过计算核苷酸或氨基酸的替换数等），可以推断物种之间的进化关系。例如，在研究哺乳动物的进化历程时，对线粒体DNA序列进行比对。因为线粒体DNA是母系遗传的，其序列变化可以反映物种的进化分支情况。通过比对不同哺乳动物的线粒体DNA序列，确定序列差异，再利用合适的算法（如邻接法、最大似然法等）构建系统发育树，从而展示物种之间的亲缘关系和进化分支顺序。
- 系统发育树能够帮助我们了解生物进化的模式和过程。例如，通过比较不同植物物种叶绿体基因序列的比对结果构建系统发育树，可以揭示植物在地质历史时期的演化顺序，包括从藻类到苔藓植物、蕨类植物、裸子植物和被子植物的进化过程，以及各个类群之间的分支关系。

2. 研究进化速率

- 基于序列比对的结果，可以分析基因或蛋白质在进化过程中的变化速率。对于一些高度保守的基因（如编码核糖体RNA的基因），它们在不同物种中的序列比对显示出较低的变化率，这表明这些基因在维持生命基本过程中起着至关重要的作用，受到较强的选择压力。
- 相反，一些与环境适应相关的基因（如在细菌中编码抗生素抗性蛋白的基因）可能会在较短时间内出现较多的序列变化，通过序列比对可以量化这些变化，研究其进化速率，从而了解这些基因是如何快速适应环境变化（如抗生素的使用）的。

三、基因组学研究

1. 基因组注释

- 在基因组测序完成后，需要对基因组中的基因进行注释，确定基因的位置、结构和功能。序列比对是基因组注释的重要工具之一。通过将基因组序列与已知的基因序列数据库进行比对，可以识别出编码基因、非编码RNA等基因组元件。例如，在人类基因组注释中，通过将新测序的基因组片段与已经注释好的基因数据库（如RefSeq）进行比对，可以确定新基因组片段中是否包含基因，以及基因的外显子-内含子结构等信息。
- 对于重复序列的注释也依赖于序列比对。基因组中存在大量的重复序列，包括串联重复序列和散在重复序列。通过序列比对可以确定这些重复序列的类型、分布和拷贝数。例如，在分析植物基因组时，通过比对可以发现转座子等重复序列的位置和数量变化，这对于理解基因组的大小变化、基因进化和表观遗传调控等方面都有重要意义。

2. 比较基因组学

- 比较不同物种的基因组是理解物种进化和生物学特征差异的关键。序列比对在比较基因组学中用于比较基因的顺序、结构和序列相似性。例如，在比较人类和黑猩猩的基因组时，通过全基因组序列比对，可以发现大量的相似区域，包括基因编码区和非编码区。同时，也可以找到一些差异区域，这些差异可能与人类和黑猩猩在智力、行为等方面的差异有关。

- 比较基因组学还可以用于研究基因家族的扩张和收缩。通过对多个相关物种基因组序列的比对和分析，确定基因家族成员的增减情况。例如，在研究植物对环境适应的过程中，通过比较沙漠植物和水生植物的基因组，发现一些与水分吸收和运输相关的基因家族在沙漠植物基因组中可能发生了扩张，通过序列比对可以具体分析这些基因家族成员的序列变化和功能分化。

Q03 Knuth - Morris - Pratt

- prompt: Knuth - Morris - Pratt 算法中的前缀函数 (prefix function) 是如何计算的?
- model_name: 豆包
- model_version: pro-256k
- response:

1. 前缀函数的定义

- 在Knuth - Morris - Pratt (KMP) 算法中，前缀函数 (prefix function) 也叫失配函数 (failure function)。对于长度为 n 的字符串 $S = s_0s_1 \cdots s_{n-1}$ ，其前缀函数 $\pi[i]$ (其中 $0 \leq i < n$) 定义为： $\pi[i]$ 是 $S[0 \cdots i]$ (即字符串 S 从开始到第 i 个字符组成的子串) 的最长真前缀 (不包括自身) 的长度，这个真前缀同时也是 $S[0 \cdots i]$ 的后缀。
- 例如，对于字符串 $S = \text{"ababaca"}$ ，当 $i = 3$ 时，子串 $S[0 \cdots 3] = \text{"abab"}$ ，它的最长真前缀同时也是后缀的是 "aba" ，长度为3，所以 $\pi[3] = 2$ 。

2. 计算前缀函数的方法 (递推计算)

- 基础情况： $\pi[0] = 0$ ，因为空字符串 (长度为0) 没有符合要求的真前缀。
- 对于 $i > 0$ ，假设已经计算出了 $\pi[0], \pi[1], \cdots, \pi[i-1]$ 的值，要计算 $\pi[i]$ 。
- 设 $k = \pi[i-1]$ ，考虑子串 $S[0 \cdots i-1]$ 和 $S[0 \cdots k-1]$ (其中 k 是 $\pi[i-1]$ 的值)。
- 比较 $S[i]$ 和 $S[k]$ ：
 - 如果 $S[i] = S[k]$ ，那么 $\pi[i] = k + 1$ 。这是因为如果 $S[i]$ 和 $S[k]$ 相等，那么 $S[0 \cdots k]$ 就是 $S[0 \cdots i]$ 的最长真前缀同时也是后缀，长度为 $k + 1$ 。
 - 如果 $S[i] \neq S[k]$ ，则需要回溯。令 $k = \pi[k-1]$ ，然后再次比较 $S[i]$ 和 $S[k]$ ，重复这个过程直到找到一个 k 使得 $S[i] = S[k]$ 或者 $k = 0$ 。如果 $k = 0$ 且 $S[i] \neq S[0]$ ，那么 $\pi[i] = 0$ 。

3. 示例计算

- 以字符串 $S = \text{"abcabacd"}$ 为例来计算前缀函数。
- 首先， $\pi[0] = 0$ 。
- 对于 $i = 1$ ，比较 $S[1]$ 和 $S[0]$ ， $S[1] = 'b'$ ， $S[0] = 'a'$ ，不相等，所以 $\pi[1] = 0$ 。
- 对于 $i = 2$ ，比较 $S[2]$ 和 $S[0]$ ， $S[2] = 'c'$ ， $S[0] = 'a'$ ，不相等，所以 $\pi[2] = 0$ 。
- 对于 $i = 3$ ，比较 $S[3]$ 和 $S[0]$ ， $S[3] = 'a'$ ， $S[0] = 'a'$ ，相等，所以 $\pi[3] = 1$ 。
- 对于 $i = 4$ ，比较 $S[4]$ 和 $S[1]$ (因为 $k = \pi[3] = 1$)， $S[4] = 'b'$ ， $S[1] = 'b'$ ，相等，所以 $\pi[4] = 2$ 。
- 对于 $i = 5$ ，比较 $S[5]$ 和 $S[2]$ (因为 $k = \pi[4] = 2$)， $S[5] = 'c'$ ， $S[2] = 'c'$ ，相等，所以 $\pi[5] = 3$ 。
- 对于 $i = 6$ ，比较 $S[6]$ 和 $S[3]$ (因为 $k = \pi[5] = 3$)， $S[6] = 'd'$ ， $S[3] = 'a'$ ，不相等。令 $k = \pi[2] = 0$ ，比较 $S[6]$ 和 $S[0]$ ，不相等，所以 $\pi[6] = 0$ 。

前缀函数的计算是KMP算法中的关键步骤，它能够帮助在匹配过程中有效地跳过已经匹配过的部分，从而提高字符串匹配的效率。

Q04 LCS递归方程

- prompt: 在动态规划方法用于最长公共子序列（LCS）计算时，递归方程是如何推导的？
- model_name: 豆包
- model_version: pro-256k
- response:

1. 定义和基础概念

- 设两个序列为 $X = [x_1, x_2, \dots, x_m]$ 和 $Y = [y_1, y_2, \dots, y_n]$ ， $LCS(X, Y)$ 表示序列 X 和 Y 的最长公共子序列。
- 当 $i = 0$ 或 $j = 0$ 时，即序列 X 或 Y 为空序列，此时 $LCS(X, Y) = 0$ 。

2. 推导递归方程

- 对于 $i > 0$ 和 $j > 0$ ，考虑序列 X 的第 i 个元素 x_i 和序列 Y 的第 j 个元素 y_j 。
- 如果 $x_i = y_j$ ，那么这个元素一定是最长公共子序列的一部分。此时 $LCS(X[1 \dots i], Y[1 \dots j]) = LCS(X[1 \dots i - 1], Y[1 \dots j - 1]) + 1$ 。
- 例如，设 $X = [1, 3, 4, 5, 6, 7, 7, 8]$ 和 $Y = [3, 5, 7, 4, 8, 6, 7, 8]$ ，当比较到 $x_4 = 5$ 和 $y_3 = 7$ 时不相等，继续比较；当比较到 $x_5 = 6$ 和 $y_5 = 8$ 时不相等，继续比较；当比较到 $x_6 = 7$ 和 $y_3 = 7$ 时相等，此时 $LCS(X[1 \dots 6], Y[1 \dots 3]) = LCS(X[1 \dots 5], Y[1 \dots 2]) + 1$ 。
- 如果 $x_i \neq y_j$ ，那么 $LCS(X[1 \dots i], Y[1 \dots j]) = \max\{LCS(X[1 \dots i - 1], Y[1 \dots j]), LCS(X[1 \dots i], Y[1 \dots j - 1])\}$ 。
- 这是因为最长公共子序列要么不包含 x_i （即 $LCS(X[1 \dots i - 1], Y[1 \dots j])$ ），要么不包含 y_j （即 $LCS(X[1 \dots i], Y[1 \dots j - 1])$ ），我们取这两种情况中的最大值。
- 例如，还是上述的 X 和 Y 序列，当比较到 $x_4 = 5$ 和 $y_3 = 7$ 时不相等，此时 $LCS(X[1 \dots 4], Y[1 \dots 3]) = \max\{LCS(X[1 \dots 3], Y[1 \dots 3]), LCS(X[1 \dots 4], Y[1 \dots 2])\}$ 。

所以，最长公共子序列问题的递归方程为：

$$LCS(X[1 \dots i], Y[1 \dots j]) = \begin{cases} 0 & \text{if } i = 0 \text{ or } j = 0 \\ LCS(X[1 \dots i - 1], Y[1 \dots j - 1]) + 1 & \text{if } x_i = y_j \\ \max\{LCS(X[1 \dots i - 1], Y[1 \dots j]), LCS(X[1 \dots i], Y[1 \dots j - 1])\} & \text{if } x_i \neq y_j \end{cases}$$

Q05 多序列比对研究方向

- prompt: 目前在多序列比对（Multiple sequence alignment）领域，有哪些新的算法或技术改进方向？
- model_name: 豆包
- model_version: pro-256k

- response:

以下是目前多序列比对领域一些新的算法或技术改进方向：

深度学习与强化学习的融合

- **深度强化学习模型**：利用深度神经网络强大的特征学习能力和强化学习的决策能力，来处理多序列比对问题。例如，通过设计新的路由编码器提取多目标和单目标序列信息，再使用深度强化学习模型并行生成近似 Pareto 最优解，从而提高比对精度和速度，有效解决大规模序列比对问题。
- **卷积神经网络 (CNN) 和循环神经网络 (RNN) 及其变体**：CNN 可以自动提取序列中的局部特征，RNN 及其变体如长短期记忆网络 (LSTM) 和门控循环单元 (GRU) 能够处理序列中的长期依赖关系。将它们结合或单独应用于多序列比对，可更准确地捕捉序列特征和相似性，进而提高比对质量。

多目标优化算法的应用

- **考虑多个比对目标**：不再仅仅依赖单一的比对得分函数，而是综合考虑多个目标，如比对准确性、空位罚分、序列相似性等，以更全面地评估比对结果。通过多目标优化算法，如 NSGA-II、MOEA/D 等，在多个目标之间进行权衡和优化，找到一组近似 Pareto 最优解，为用户提供更多符合不同需求的比对结果选择。
- **动态调整目标权重**：根据序列的特点和用户的需求，动态地调整各个比对目标的权重。在比对过程中，随着比对的进行和序列特征的逐渐显现，自动地优化目标权重，以更好地适应不同阶段和不同类型序列的比对要求，提高比对的灵活性和准确性。

遗传算法及相关改进

- **改进的自适应遗传算法**：采用适应度值标定、实现群体多样性，并利用如 sigmoid 函数对交叉率和突变率进行非线性自适应调整，提高算法的寻优能力和稳定性，避免早熟收敛，从而改善多序列比对的质量。
- **遗传退火算法**：将遗传算法与模拟退火算法相结合，在种群更新阶段通过模拟退火算法的 Metropolis 准则调整遗传算法的进化过程，克服遗传算法在迭代后期搜索效率低、易过早收敛于局部最优解的问题，在保证种群多样性的前提下，加快收敛速度，提高全局和局部搜索能力及效率。

基于图论和网络分析的方法

- **构建序列图模型**：将序列表示为图的节点，序列之间的相似性或比对关系表示为图的边，通过图论中的算法和技术，如最短路径算法、图聚类算法等，来寻找最优的比对路径和比对结果。这种方法能够更直观地反映序列之间的复杂关系，有助于发现隐藏在序列数据中的结构信息和相似模式。
- **网络比对算法的应用**：借鉴网络比对领域的算法和思想，将多序列比对问题转化为网络比对问题。通过比较不同序列之间的网络结构相似性，来确定它们的比对关系，对于处理具有复杂结构和相互作用的生物序列数据，如蛋白质相互作用网络、基因调控网络等，具有潜在的优势。

并行计算与分布式计算技术

- **多线程并行计算**：利用现代计算机的多核处理器架构，设计多线程并行的比对算法，将比对任务分配到多个线程中同时进行，充分发挥硬件的并行计算能力，大大提高比对速度。例如，在渐进式比对算法的各个阶段，如两两比对、构建进化树、多序列比对等，都可以采用多线程并行计算，显著缩短比对时间。
- **分布式计算框架**：借助分布式计算框架，如 Hadoop、Spark 等，将大规模的序列比对任务分布到多个计算节点上进行处理。这种方式可以处理超大规模的序列数据，适用于基因组学等领域中对海量序列进行比对的需求，通过并行和分布式计算，实现高效的比对和分析。

基于统计模型和机器学习的方法

- **隐马尔可夫模型 (HMM)**：HMM 可以对序列中的隐藏状态进行建模，通过学习序列的统计特征来预测比对结果。例如，通过构建多序列的 HMM 模型，利用 Baum-Welch 算法等进行参数估计和模型训练，然后根据训练好的模型进行序列比对和相似性搜索，对于处理具有复杂结构和噪声的序列数据具有较好的效果。
- **支持向量机 (SVM) 和其他机器学习分类器**：将序列比对问题转化为分类问题，利用 SVM 等机器学习分类器来学习和预测序列之间的比对关系。通过提取序列的特征向量，如 k-mer 频率、序列长度、碱基组成等，作为分类器的输入，训练得到一个能够准确预测序列比对结果的模型，为多序列比对提供一种新的思路和方法。

近似算法和启发式策略的改进

- **改进的渐进式比对算法**：对传统的渐进式比对算法进行改进，如在构建进化树时采用更准确的距离度量方法，或者在比对过程中动态调整序列的比对顺序，以提高比对的准确性。同时，结合局部比对信息和全局比对信息，对渐进式比对的结果进行进一步的优化和修正。
- **迭代比对算法的优化**：优化迭代比对算法的收敛条件和迭代策略，减少不必要的迭代次数，提高比对效率。例如，通过设置合理的停止准则，当比对结果的改进小于一定阈值时停止迭代，或者采用自适应的迭代步长，根据比对的进展情况动态调整迭代的幅度，加快比对速度并保证比对质量。