以下是基于文档内容提出的一些问题：

### 关于子字符串搜索

1. Knuth-Morris-Pratt (KMP) 算法如何避免字符串的回溯？其时间复杂度为什么是 O(n)？
2. 在使用有限状态自动机（FSA）进行字符串匹配时，如何高效地构建转换函数 δ(q, a)？
3. 在构建后缀树时，如何保证树的结构满足后缀树的定义（例如 SF1-SF5）？
4. 后缀树有哪些实际应用场景，例如在字符串重复子串查找或子串匹配问题中？

### 关于序列对齐（Pairwise Sequence Alignment）

1. 对比 Needleman-Wunsch 算法和 Smith-Waterman 算法，它们在全局和局部序列对齐中的适用场景是什么？
2. 在动态规划方法中，如何根据相似性和插入得分设计最优的评分函数？
3. 为什么重复序列（尤其是长度不同的重复）会对配对序列对齐造成困难？

### 关于多序列对齐（MSA）

1. 动态规划在多序列对齐中的时间复杂度高达 O(n^k)，有哪些实际的启发式算法用于应对这种复杂性？
2. Progressive alignment 和 Iterative refinement 在多序列对齐中的优势和局限性是什么？
3. 使用 Clustal 等工具时，“一旦产生缺口，始终保留缺口” 的规则是如何影响最终对齐结果的？
4. MSA 中的评分函数（例如 Sum of Pairs 和 Entropy-based Score）如何选择和应用？

### 关于生物信息学中的应用

1. 在基因组组装中，如何利用后缀树和多序列对齐算法构建“contig”图？
2. MSA 的结果如何用于蛋白质家族的结构预测？

### 其他算法与扩展

1. Boyer-Moore 和 Karp-Rabin 等其他字符串匹配算法与 KMP 的性能如何比较？它们的适用场景有哪些差异？
2. 为什么局部对齐算法（如 Smith-Waterman）在处理含有类似基序的长序列时更有效？
3. 文中提到的 MAFFT 和 MUSCLE 等多序列对齐工具相比，它们的优劣点是什么？

### 关于递归公式与实现

1. 在动态规划公式中，递归方程如何被修正以考虑相似性评分或缺口扩展？
2. 在 Needleman-Wunsch 算法中，如何在大规模序列中高效处理得分矩阵的构建和回溯过程？

这些问题能够帮助深入理解文档的核心内容，并为实际应用或算法改进提供参考方向。