Lab2 复杂 DNA 序列的比对

2025-04-28

1 背景：常见 DNA 变异

在 DNA 序列分析中，各种变异非常常见。以下介绍常见的变异类型

单核苷酸突变

这是最常见的一类 DNA 突变，主要表现形式为一个核苷酸发生了改变。例如

```
ref:   ATCGA
       | |||
query: AACGA
```

在这个例子中，序列第 2 位的 T 发生变异，变成了 A。

插入突变

插入突变指的是，相比于 ref 序列，query 中出现了额外的核苷酸或 DNA 片段。例如

```
ref:   AT----CGA
       ||    |||
query: ATACTTCGA
```

在这个例子中，相比于 ref 序列，query 中出现了额外的插入片段 ACTT。

删除突变

删除突变指的是，相比于 ref 序列，query 中缺失了核苷酸或 DNA 片段。例如

```
ref:   ATGGTACGA
       ||    |||
query: AT----CGA
```

在这个例子中，相比于 ref 序列，query 中缺失了片段 GGTA。

重复突变

上一 Lab 已经涉及，重复突变的主要表现为 query 出现了额外重复的片段。例如

```
ref:   ATCG---A
       ||||   |
query: ATCGTCGA
```

在这个例子中，相比于 ref 序列，query 中发生了片段 TCG 的重复，重复次数为 1。

倒位（逆转）突变

上一 Lab 已经涉及，其主要表现为 query 中的序列是 ref 的反向互补序列。例如

```
ref:    ATCG
query: CGAT
```

在这个例子中，ref 序列为 5'-ATCG-3'，其反向互补序列为 3'-TAGC-5'，或者写作 5'-CGAT-3'（一般情况下默认 5' 写在左侧，3' 写在右侧）。

片段移位

片段移位可以理解为同时发生了插入突变和删除突变，具体表现为相比于 ref 序列，query 中的一个片段不再位于原来的位置，而是出现在另一个位置。例如

```
ref:    ATGGTACGA---TTC
        |||    |||    |||
query: ATG---CGAGTATTC
```

在这个例子中，query 序列中的 GTA 离开了原来所处的位置（第 4-6 个核苷酸处），移动到了序列的其他地方（第 7-9 个核苷酸处）。

2 项目目标

本次 Lab 的目标是实现复杂 DNA 序列的比对，需要同时兼顾算法的时空复杂度和最终结果准确性。要求分别在 PPT 给出的两组 query 和 reference 中，实现 query 和 reference 的匹配，并输出具体的匹配情况。其中，匹配过程中可能会遇到以上提及的变异类型，也可能出现其他类型的变异。

3 具体细节

3.1 输入与输出

代码输入：query 和 reference 两个序列，均为由 ATCG 构成的字符串。详见实验二 PPT 第 4-7 页。

代码输出：若干元组，其中每个元组按照格式 (query_start, query_end, ref_start, ref_end) 书写，表示 query 中从 query_start 到 query_end 的序列与 reference 中 ref_start 到 ref_end 的序列匹配。若两者为反向互补匹配（即 query_start 与 ref_end 配对），无需特别修改 start 和 end 的顺序，提交时会自动判断。例如上一个 Lab 的结果按照该格式书写应为：

```
[(   0, 400,  0,400), ( 400, 450,350,400), (450,500,350,400),
 ( 500, 550,350,400), ( 550, 600,350,400), (600,670,330,400),
 ( 670, 740,330,400), ( 740, 810,330,400), (810,910,300,400),
 ( 910,1010,300,400), (1010,1410,400,800)]
```

这里的区间是左闭右开区间，因此请特别注意右端点是否需要额外 +1

3.2 实现思路提示

本次 Lab 可以利用此前 Lab 的代码框架，但需要思考以下问题：

1. 上一 Lab 中我们明确定义了什么样的变异是合法的。但在本次 Lab 中，由于采用的是较为真实的数据，变异的具体情况是比较模糊的，没有所谓“合法的变异”一说。也正因为如此，本次 Lab 不再要求计算出从 ref 到 query 所发生的具体变换方式，仅要求计算两者的匹配关系，即 query 中的某一片段与 ref 中的哪一片段相匹配。因此，请在编写代码时考虑更加通用的情况，避免用固定化的表达式来描述每一类变异。

2. 上一 Lab 中 ref 和 query 的碱基是逐个进行匹配的。但实际上，我们是否可以一次匹配多个碱基，从而大幅减小复杂度中的系数？hash 在这里可能会很有用：可以通过设计特定的 hash 函数使得两个相似的序列发生碰撞，从而认为这两个相似序列几乎是一致的，以减小微小变异（例如单核苷酸突变）对整体匹配的干扰

3. 常规的动态规划方法要求填完整个打分矩阵，因此复杂度必然是 $\Omega(mn)$ 级别（其中 m, n 分别为 ref 和 query 的长度）。如果想要降低复杂度，一个思路是考虑在打分矩阵中有哪些位置是一定不用计算的，或者在已知某些位置的分数下有哪些位置可以不用计算。

4. 本次 Lab 需要使用图相关算法实现。图的结点表示 ref 和 query 中核苷酸的匹配情况，图的边表示上一个匹配到下一个匹配的转移过程，算法目标是找到图中的最长路径。提示关键词：“anchor”（匹配的片段作为锚点），“chain”（延伸并连接锚点）

4 相关要求

需要完成的任务：编写代码，根据输入的 query 和 reference 计算两者的匹配关系，输出格式见3.1节。编程语言不限。**需使用图相关算法**。

扩展性要求：本次 Lab 只需要处理 PPT 所给的两组 ref+query 序列即可
正确性要求：需要将算法的输出提交到评分网站，分数不得低于基线分数。
复杂度要求：时间复杂度不得高于平方量级 $O(mn)$（不含平方量级）。其中，复杂度几乎为线性量级可得满分，平方量级可得大部分分数。
提交要求：在 Elearning 上提交实验文档，包括算法伪代码、时空复杂度、运行结果。在 GitHub 上更新代码。在评分网站上实名提交结果。
评分网站
第一组输入: http://10.20.26.11:8550
第二组输入: http://10.20.26.11:8551
需实名提交，多次提交将保留最高分。如需测试，请使用群内提供的测试脚本。原测试用户名 test 已不再支持

截止时间：2025.5.31 23:59