题目：序列聚类

Contents

SE@SJTU 2022

1 背景 **1**

2 题目任务 **1**

2.1 数据读取 1

2.2 编辑距离 2

2.3 计算相似 DNA 序列集合 2

2.4 DBSCAN 3

2.5 最小半径 4

3 输入输出样例 **4**

4 提交要求和考核标准 **5**

4.1 编码要求 5

4.2 评分标准 5

4.3 提交要求 5

1 背景

DNA 是由 A、T、C、G 四种碱基组成的双螺旋结构。双螺旋中的每一条 DNA 单链，都是由 A、T、C、G 排列组成。DNA 测序可以将 DNA 链中碱基的排列顺序测量出来，在生活和生产中发挥着非常重要的作用。在对同一种 DNA 单链进行测序时，测序过程中的错误会导致测序结果中存在多种不同的序列。因此，当多种 DNA 单链混合在一起进行测序之后，往往需要使用聚类算法才能得知不同 DNA 序列的数量和具体内容。

在本题目中，你需要实现一个 DBSCAN 聚类算法，并使用该算法，得出给定的测序结果中 DNA 序列的数量和具体内容。

2 题目任务

在本题目中，你需要按照指示完成以下步骤。

2.1 数据读取

函数名：本步骤需要实现为一个名为 `LoadData` 的函数，下面是该 C++ 中该函数的一个参考签名，你可以根据语言和自己的需求对函数签名进行更改，但不应改变函数名。后续步骤的要求与此相同。

```
1  void LoadData();
```

首先从从标准输入中读取一个字符串，为输入文件的路径。此后读取该文件中的信息。文件第一行为三个数字 N、E 和 P。其中 N 表示在该次测序中共测得 N 条 DNA 序列；E 表示此后聚类中使用的 `eps`；P 表示此后聚类中使用的 `MinPts`。

此后 N 行，每行为一个由 ATCG 四个大写字母组成的字符串，表示一个 DNA 序列。

为了简化题目，输入文件中 N 个 DNA 序列 **各不相同**。

标准输出：

在本步骤中，你需要的按上述方式读取数据，并在标准输出中打印 N 条序列中的最短序列的长度和最长序列的长度。两个数字使用一个空格隔开，共占一行。

2.2 编辑距离

函数名：你需要实现为一个名为 `LevDist` 的函数，计算两个 DNA 序列的编辑距离。

```
int LevDist(const string &a, const string &b);
```

在 DNA 序列的聚类算法中，会使用编辑距离（又称莱文斯坦距离，Levenshtein）计算两条 DNA 序列的距离。对于两个字符串 A、B，它们的编辑距离为，在允许插入、删除、替换操作的情况下，字符串 A 变换为 B 所需要的最少操作次数。编辑距离的定义如下：

$$Lev_{A,B}(i,j) = \begin{cases} max(i,j) & ,if min(i,j) = 0 \\ min = \begin{cases} Lev_{A,B}(i-1,j)+1 \\ Lev_{A,B}(i,j-1)+1 \\ Lev_{A,B}(i-1,j-1) & ,if A_i = B_j \\ Lev_{A,B}(i-1,j-1)+1 & ,if A_i \neq B_j \end{cases} & ,otherwise \end{cases} \tag{1}$$

编辑距离可根据上述定义使用动态规划求得。

标准输出：

在本步骤中，你需要实现 `LevDist` 函数，并在程序的主函数中（在 C/C++ 中为 `main` 函数）计算前两个 DNA 序列的编辑距离，并输出结果，一个数字共占一行。

2.3 计算相似 DNA 序列集合

函数名：你需要实现一个名为 `RangeQuery` 的函数。

```
vector<string> RangeQuery(const vector<string> &dna_set,
                          const string &target_dna,
                          int eps);
```

该函数接收一个 DNA 序列集合（`dna_set`）和一个指定 DNA 序列（`target_dna`），返回在 `dna_set` 中与 `target_dna` 距离不超过 `eps` 的所有 DNA 序列的集合。

标准输出：

在本步骤中，你需要实现 `RangeQuery` 函数，并调用该函数，得到在输入文件中给定的 DNA 序列集合中，与其中第一个 DNA 序列相似的 DNA 序列集合，eps 值为输入文件中的 E。在得到集合后，输出该集合中 DNA 序列的数量，一个数字共占一行。

2.4 DBSCAN

函数名：你需要将 DBSCAN 实现为一个名为 `DBSCAN` 的函数。

```
vector<vector<string>> DBSCAN(const vector<string> &dna_set,
                              int eps,
                              int minpts,
                              int &num_cores,
                              int &num_borders,
                              int &num_outliers,
                              int &num_clusters);
```

我们首先通过一个二维平面上点集的 DBSCAN 进行介绍。

DBSCAN 有三个参数，分别为：

- 最小点数量 `MinPts`
- 距离计算函数 `DistFunc`
- 最小距离 `eps`

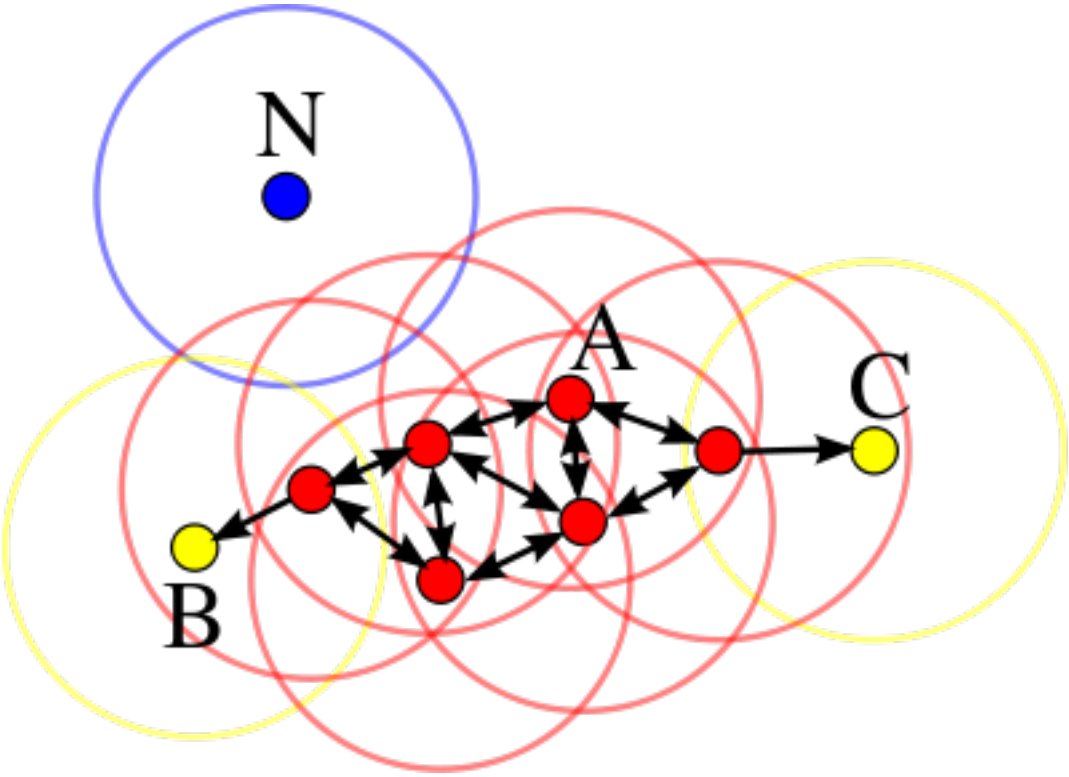
根据这三个参数，我们可以将点集合中的点分为三种：

- 核心点：对于点集中的任何一个点，如果在其半径为 eps 的圆形范围内（包括落在圆边界上的点，下同），有不少于 MinPts 个点（包括该点自己），则该点为核心点；
- 边界点：如果一个点，其半径为 eps 的圆形范围内的点数量小于 MinPts，但是该点在某个核心点的 eps 半径范围内，则该点为边界点；
- 噪声点：如果一个点，其半径为 eps 的圆形范围内的点数量小于 MinPts，且该点不在任何核心点的 eps 半径范围内，则该点为噪声点。

在将所有的点分成上述三种之后，通过以下的规则可将所有的点进行聚类，划分为多个点集（称为簇）：

- 对于任何一个核心点，其半径为 eps 的圆形范围内的所有点，应与该点属于同一簇。
- 按前一条规则，一个边界点有可能同时属于两个簇，此时将其归为首先发现该边界点的簇即可。

下图为 DBSCAN 算法的一个经典示例，其中 MinPts 为 4，eps 为图中圆的半径。图中红色的点均为核心点，黄色的点为边界点，蓝色的点为噪声点。在进行聚类时，蓝色的点 N 之外，其余点属于同一簇。



我们可以认为每个点对应一个 DNA 序列，点与点之间的距离为两个 DNA 序列之间的编辑距离，因而可以使用 DBSCAN 进行 DNA 序列的聚类。

在实现 DBSCAN 算法时，你需要使用到 `LevDist` 和 `RangQuery` 两个函数。你可以使用广度优先搜索等方法完成对点的聚类，具体实现方式不做要求。

标准输出：

在本步骤中，你需要根据输入文件中提供的 DNA 序列集合、E 和 P 的值运行 DBSCAN 算法。输出一行信息，包括四个数字，分别为 核心点数、边界点数、噪声点数、聚类后的簇数（边界点不属于任何簇），数字之间使用一个空格隔开，共占一行。

2.5 最小半径

函数名：你需要实现一个名为 `MinEPS` 的函数，求得能够消除噪声点的最小 eps 值。

```
1  int MinEPS(const vector<string> &dna_set, int minpts);
```

在 DBSCAN 算法中，当半径 eps 越大，噪声点的个数越少。在输入文件中给定的 DNA 序列集合和 P 值的要求下，请计算能够消除噪声点的最小 eps 值。

标准输出：

在本步骤中，你需要输出上述要求的最小 eps 值，一个数字，共占一行。

3 输入输出样例

为了方便大家进行调试，我们在 data 目录中给出了样例（sample）、小规模（small）和大规模（large）数据。

给出的每个测试包括四个文本文件，以 sample 为例：

- sample.stdin 为标准输入的内容（仅作参考，程序的标准输入为 sample.txt 文件的路径）
- sample.txt 为输入文件的内容（对应标准输入中的 `./data/sample.txt`）
- sample.stdout 为正确的标准输出的内容

大规模测试的 stdout 文件不予给出。

4 提交要求和考核标准

4.1 编码要求

语言不限，但请根据题目选择合适的语言并按照要求进行设计和编码。同时，不可直接调用现有类库中 DBSCAN 等具有聚类功能的库函数。

4.2 评分标准

设计和实现程序，完成前述功能。如果不能完成全部程序功能，也请不要担心，我们会根据各个方面独立评分。具体标准如下：

1. 数据读取：15 分；
2. 编辑距离：20 分；
3. 计算相似 DNA 序列集合：15 分；
4. DBSCAN：20 分；
5. 参数优化：10 分；
6. 通过所有测试数据（每个测试数据运行限时 15 秒）：10 分；
7. 代码规范、命名标准、代码风格、注释和说明等：10 分；

4.3 提交要求

请将程序源代码和需要提交的文件使用 7z 格式压缩后命名为“姓名.7z”并执行以下两个操作：

- 将其 **放置在考试环境桌面的显著位置**；
- 将其上传到：https://jbox.sjtu.edu.cn/l/01wL6z 中的“题目提交”目录中；
  - 如果不小心传错文件，可以重新以“姓名-v2.7z”上传，最终以版本号最大的版本为准。

压缩包中应仅包含源代码和指定提交的文件，不包含测试用的输入输出、中间文件或可执行文件。压缩包大小原则上不超过 5MB。

考试完毕后请记得结束录屏。关闭远程连接的时候，请直接断开连接，不要关机。