Брянцев Всеволод Александрович

ПММ, 4 курс, 61 группа

Отчет по лабораторной работе №4

**Задание 1.**

Найти наилучшее глобальное выравнивание между двумя строками нуклеотидных последовательностей при заданной матрицы весов. Файлы для примера брать в соответствии с вариантом из Лабораторной работы 2 из базы данных GenBank.

Входные данные. Строки V и W и матрица весов.

Выходные данные. Глобальное выравнивание между V и W, вес которого (определенный по матрице весов) является максимальным среди всех возможных выравниваний V и W.

**Задание 2.**

Найти наилучшее локальное выравнивание между двумя строками нуклеотидных или аминокислотных последовательностей. Файлы для примера брать в соответствии с вариантом из Лабораторной работы 2 из базы данных GenBank.

Входные данные. Строки V и W и матрица весов.

Выходные данные. Локальное выравнивание, определяемое подстроками строк V и W, глобальное выравнивание которых, определенных матрицей весов, является наилучшим среди всех глобальных выравниваний всех подстрок V и W.

Вариант 4.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | a | t | g | c |
| a | 1 | 0 | 0 | 0 |
| t | 0 | 1 | 0 | 0 |
| g | 0 | 0 | 1 | 0 |
| c | 0 | 0 | 0 | 1 |

Штраф за пробел – 0.

Код программы.

// Вариант 4.

import 'dart:io';

import 'dart:math';

const Map<String, Map<String, int>> scoringMatrix = {

  'a': {'a': 1, 't': 0, 'g': 0, 'c': 0},

  't': {'a': 0, 't': 1, 'g': 0, 'c': 0},

  'g': {'a': 0, 't': 0, 'g': 1, 'c': 0},

  'c': {'a': 0, 't': 0, 'g': 0, 'c': 1},

};

const int gapPenalty = 0;

Future<String> readFile(String path) async {

  return await File(path).readAsString();

}

List<List<int>> initializeMatrix(int rows, int cols, bool isGlobal) {

  List<List<int>> matrix = List.generate(rows, (\_) => List.filled(cols, 0));

  if (isGlobal) {

    for (int i = 1; i < rows; i++) {

      matrix[i][0] = matrix[i - 1][0] + gapPenalty;

    }

    for (int j = 1; j < cols; j++) {

      matrix[0][j] = matrix[0][j - 1] + gapPenalty;

    }

  }

  return matrix;

}

String formatAlignment(String seq1, String seq2) {

  StringBuffer middleLine = StringBuffer();

  for (int i = 0; i < seq1.length; i++) {

    if (seq1[i] == seq2[i]) {

      middleLine.write('|');

    } else {

      middleLine.write(' ');

    }

  }

  StringBuffer formatted = StringBuffer();

  for (int i = 0; i < seq1.length; i += 70) {

    formatted.writeln(seq1.substring(i, min(i + 70, seq1.length)));

    formatted

        .writeln(middleLine.toString().substring(i, min(i + 70, seq1.length)));

    formatted.writeln(seq2.substring(i, min(i + 70, seq2.length)));

    formatted.writeln();

  }

  return formatted.toString();

}

void saveAlignmentToFile(String filename, String alignment) {

  File(filename).writeAsStringSync(alignment);

}

int needlemanWunsch(String seq1, String seq2) {

  int rows = seq1.length + 1;

  int cols = seq2.length + 1;

  List<List<int>> matrix = initializeMatrix(rows, cols, true);

  for (int i = 1; i < rows; i++) {

    for (int j = 1; j < cols; j++) {

      int match =

          matrix[i - 1][j - 1] + scoringMatrix[seq1[i - 1]]![seq2[j - 1]]!;

      int delete = matrix[i - 1][j] + gapPenalty;

      int insert = matrix[i][j - 1] + gapPenalty;

      matrix[i][j] = max(match, max(delete, insert));

    }

  }

  int i = seq1.length, j = seq2.length;

  String alignedSeq1 = '', alignedSeq2 = '';

  while (i > 0 || j > 0) {

    if (i > 0 &&

        j > 0 &&

        matrix[i][j] ==

            matrix[i - 1][j - 1] + scoringMatrix[seq1[i - 1]]![seq2[j - 1]]!) {

      alignedSeq1 = seq1[i - 1] + alignedSeq1;

      alignedSeq2 = seq2[j - 1] + alignedSeq2;

      i--;

      j--;

    } else if (i > 0 && matrix[i][j] == matrix[i - 1][j] + gapPenalty) {

      alignedSeq1 = seq1[i - 1] + alignedSeq1;

      alignedSeq2 = '-$alignedSeq2';

      i--;

    } else {

      alignedSeq1 = '-$alignedSeq1';

      alignedSeq2 = seq2[j - 1] + alignedSeq2;

      j--;

    }

  }

  String alignment = formatAlignment(alignedSeq1, alignedSeq2);

  saveAlignmentToFile("needleman\_wunsch\_output.txt", alignment);

  print("Global Alignment Score: ${matrix[seq1.length][seq2.length]}");

  return matrix[seq1.length][seq2.length];

}

int smithWaterman(String seq1, String seq2) {

  int rows = seq1.length + 1;

  int cols = seq2.length + 1;

  List<List<int>> matrix = initializeMatrix(rows, cols, false);

  int maxScore = 0;

  int maxI = 0, maxJ = 0;

  for (int i = 1; i < rows; i++) {

    for (int j = 1; j < cols; j++) {

      int match =

          matrix[i - 1][j - 1] + scoringMatrix[seq1[i - 1]]![seq2[j - 1]]!;

      int delete = matrix[i - 1][j] + gapPenalty;

      int insert = matrix[i][j - 1] + gapPenalty;

      matrix[i][j] = max(0, max(match, max(delete, insert)));

      if (matrix[i][j] > maxScore) {

        maxScore = matrix[i][j];

        maxI = i;

        maxJ = j;

      }

    }

  }

  String alignedSeq1 = '', alignedSeq2 = '';

  int i = maxI, j = maxJ;

  while (i > 0 && j > 0 && matrix[i][j] > 0) {

    if (matrix[i][j] ==

        matrix[i - 1][j - 1] + scoringMatrix[seq1[i - 1]]![seq2[j - 1]]!) {

      alignedSeq1 = seq1[i - 1] + alignedSeq1;

      alignedSeq2 = seq2[j - 1] + alignedSeq2;

      i--;

      j--;

    } else if (matrix[i][j] == matrix[i - 1][j] + gapPenalty) {

      alignedSeq1 = seq1[i - 1] + alignedSeq1;

      alignedSeq2 = '-$alignedSeq2';

      i--;

    } else {

      alignedSeq1 = '-$alignedSeq1';

      alignedSeq2 = seq2[j - 1] + alignedSeq2;

      j--;

    }

  }

  String alignment = formatAlignment(alignedSeq1, alignedSeq2);

  saveAlignmentToFile("smith\_waterman\_output.txt", alignment);

  print("Local Alignment Score: $maxScore");

  return maxScore;

}

Future<void> main() async {

  String seq1 = (await readFile('bin\\input1.txt')).trim();

  String seq2 = (await readFile('bin\\input2.txt')).trim();

  needlemanWunsch(seq1, seq2);

  smithWaterman(seq1, seq2);

}

Файлы input1 и input2 содержат последовательности V и W:

input1:

ttcgtaagtgccctcaccatcctatgccatgccgcgcggctgcctggtagccctcggccctgcgcaatcatgaacacgcaactaacaagctcgcgttgtaggacaaggatggcgatggtcagtactcccccttcgaactcacttccgcatactctgccctcaaagcagcgccgcatctccatcccacgcaatcggtaaaggggccggagcgaggcttgctggctaggggtccaaaacaccgctcagagctacaagaccacgacatccgcccgactcagtaacaagcacaactgacgatgatgcgccacaggtcaaatcaccaccaaggagctaggcaccgtcatgcgctcgctcggccaaaaccccagcgagtctgagctccaggacatgatcaacgaggtcgatgccgacaacaacggcaccattgacttcccaggtacatcctctcgtacgagtccaacgtgccacagctaacttctccagaattccttaccatgatggcccgcaagatgaaggacaccgactccgaggaggagatccgggaagccttcaaggtcttcgaccgcgataacaacggcttcatctccgccgccgaactgcgtcacgtcatgacttctattggcgagaaattgaccgatgacgaggtcgacgagatgatccgggaggctgaccaggacggtgacggccgtatcgactgtaggtcacaact

input2:

ccgagttcaaggaggccttctccctcttcgtaagtagctccctgtctgcttgacgcgcggctgcctagagccttgtcgtcaacaccatgaccatgttgttgctaacacatgttctctctataggacaaggatggcgatggttagtagtcccccttcgaactcacgcccgcgcactctacccgaagccagctcgcagtcctcggtctgcacgcgtctcacaccagacaatcgctgaccatgttgcgccgcaggtcaaatcaccaccaaggagctcggaaccgtcatgcgctcgctcggccaaaatcccagcgagtctgagctacaggacatgatcaacgaggtcgacgccgacaacaacggcaccattgacttcccaggtacctcaattctccgactaccaactcaccaaaaccctcactaaccacaacccagagttcctcaccatgatggcccgcaagatgaaggacaccgactctgaggaggagattcgggaagccttcaaggtctttgaccgtgacaacaacggcttcatctccgccgctgaactgcgtcacgtcatgacttctattggcgaaaaattgaccgatgacgaggttgacgagatgatccgggaggctgaccaggacggcgacggccgcatcgactgtaggcgctaaagagttgctcggttccttcaaacactaaa

Результат для заданных последовательностей:



Выравнивания выводятся в текстовые файлы needleman\_wunsch\_output и smith\_waterman\_output блоками по 100 символов в строке для удобства отображения:



