Выравнивание двух строк по алгоритму Нидлмана-Вунша. Отчет по задаче.

Владимирова Элина

23 декабря 2020 г.

1 Резюме

Для решения задачи об оптимальном выравнивании двух последовательностей был реализован классический алгоритм Нидлмана-Вунша.

Входные данные - длины двух строк. Преобразованиям подвергаются построенные по ним путем случайной генерации символов из заданного заранее алфавита буквенные последовательности. Результат работы программы - две строки, полученные из исходных вставками пропусков для достижения наибольшей посимвольной схожести последовательностей при их наименьшей длине.

2 Постановка задачи

Необходимо найти оптимальное выравнивание 2 заданных последовательностей, состоящих из символов алфавита $\{A, G, C, T, \}$ т.е. выявить наименее трудозатратную комбинацию элементарных преобразований строк, позволяющую получить последовательности с максимальным совпадением символов на одинаковых местах, имеющие при этом минимальную длину.

3 Допущения

Строки генерируются внутри программы, их длины задаются вручную. Алгоритм, однако, не привязан к содержимому строк, что позволяет регулировать как алфавит, так и непосредственно строки.

4 Описание решения

Был реализован классический алгоритм Нидлмана-Вунша, предусматривающий работу с матрицей размера $(n + 2) * (m + 2)$, где n, m - длины строк.

5 Результаты

Правильность решения была проверена на строках малой длины (4, 15 и 40 символов), в том числе на строках с сравнительно большим относительно длин модулем разности длин (строки

длин 2 и 4, 3 и 15, 5 и 40 символов). Были проведены тесты как с генерируемыми случайно и потому значительно отличающимися друг от друга, так и со схожими, отличающимися 1-3 символами строками, а также со строкой и некоторым ее фрагментом:

```
                                    40                                      10
                                    20                                      3
                AGA   ACGT    abracaabra    GCTGCCGGGATGCCACAGAGGCCCATAAGCTGATGCCTGG      CAATCATCTA
                AGA   TGCA    abracadabra   ATGGCTATGCAAAGTAAGTA                          CAT

                AGA   ACG_T   abraca_abra   GCTGCCGGGATGCCACAGAGGCCCATAAGCTGATGCCTGG      CAATCATCTA
                AGA   _TGCA   abracadabra   _ATG__GCTATGC_A_A_AG_____TAAG_T_A_______      CA_T______
120
120
CGGTGCTAGACTGTATGCGTCACTTCCAACTCGAGTCTAATATGTACCGAGTTGACGTAGATCGTGATAGAGTACCGCACACGTAGGTGATCGATAAATTGGAATGCAGTTTGTCAATAG
CATGCCCTGGCGTCGGCCGTCCTTGTCAAAACTTCCAATTGTAAAGCATCGCTTCGAATTACTGATGGTCGCTCAGAGGCTCCAGAGCCTGGACCTTTGTTCAACACGGACCGGACGCAG

CGGTGC__TAGACTGTATG_C_GTCACTT__CCAA__CT_CGAGTCT__AATA___T_G_TACCGAGTTGACGTAGATCG_T_GAT_AGAG__TACCGCACA_C__GTAGGT__GAT_CGA_TAAATTGGAATGCA_GTTTGTCAATAG
C_ATGCCCT_GGC_GT_CGGCCGTC_CTTGTC_AAAACTTCCAAT_TGTAA_AGCATCGCTTC_GAATT_AC_T_GAT_GGTCGCTCAGAGGCT_CC__AGAGCCTGGACCTTTG_TTC_AACA___CGGACCGGACG_____CA___G
```

Программа успешно обрабатывает последовательности длиной до 10 тыс. знаков - по преодолении этого порога возникают проблемы с памятью, причиной которым - невозможность хранения выстраиваемой в ходе решения матрицы и произведения внутри нее предусмотренных алгоритмом операций.