УДК 681.3

Медведев М. Г.

ВІДСТАНЬ ЛЕВЕНШТЕЙНА ТА ПОВ'ЯЗАНІ З НЕЮ ЗАДАЧІ

У статті наводиться розв'язок задачі пошуку відстані Левенштейна між рядками. Досліджується її зв'язок із задачами знаходження найбільшої спільної підпослідовності та оптимальним зіставленням рядків.

Задача пошуку та порівняння підрядків використовується в різних галузях науки. Наприклад, наступні задачі часто виникають у молекулярній біології:

- зберігання, оновлення та порівняння послідовностей нуклеотидів;
- пошук рядків, що містять задані підрядки;
- порівняння двох та більше рядків на подібність;
- пошук підрядків, що найчастіше зустрічаються у заданій послідовності;
- пошук інформативних елементів у протеїнах та амінокислотних послідовностях.

Розв'язок наведених задач грунтується на вивченні структури та функціональності протеїну навіть без проведення конкретних експериментів та фізичної його побудови. Основна ідея полягає в тому, що подібні послідовності породжують подібні протеїни. Основними змінами, які відбувалися у послідовності нуклеотидів, є:

- вставка літери (літер) у послідовність;
- видалення літери (літер) із послідовності;
- заміна літери іншою у послідовності.

Вставка та видалення літер є взаємнооберненими операціями: якщо послідовність X можна отримати із послідовності Y в результаті вставки літери, то послідовність Y можна отримати із X видаленням цієї літери.

Якщо кожному перетворенню присвоїти вагу, то можна ввести поняття *відстані* між послідовностями як мінімальну суму ваг перетворень, що трансформують одну послідовність нуклеотидів в іншу.

Наступні задачі часто виникають у молекулярній біології.

Задача 1. *Глобальне порівняння.* Дано два рядки A та B приблизно однакової довжини. Яка між ними відстань?

Задача 2. *Локальне порівняння.* Дано два рядки A та B приблизно однакової довжини. Чому дорівнює найменша відстань між підрядками A та B заданої довжини?

Алгоритми пошуку підрядків

Текстом T будемо називати масив символів $T[1 \dots n]$ довжини n, а шаблоном P — масив $P[1 \dots m]$ довжини $m \le n$. Вважаємо, що елементи масивів T та P — символи деякого скінченного алфавіту Σ (наприклад, $\Sigma = \{a, b, \dots, z\}$). Алгоритмом пошуку підрядків будемо називати алгоритм, який знаходить усі позиції входження шаблону до тексту. У сучасному світі існує велика кількість алгоритмів, які розв'язують поставлену задачу. Серед них відмітимо наступні:

- найпростіший алгоритм перевірки входження шаблона P у текст T полягає у послідовній перевірці рівності $P[1 \dots m] = T[s \dots s + m - 1]$ для кожного із $n - m$ можливих значень s. Часова оцінка такого алгоритму дорівнює $O(n^2)$;
- алгоритм Рабіна—Карпа пошуку підрядків, незважаючи на свою найгіршу оцінку $O((n - m + 1) * m)$, в середньому працює достатньо швидко;
- для розв'язку поставленої задачі велика кількість алгоритмів на початку своєї роботи будують автомат, який потім знаходить в тексті T усі входження шаблону P;
- алгоритми Кнута—Моріса—Пратта та Бойєра—Мура вирішують задачу пошуку за лінійний час.

Операції редагування

Оскільки поняття відстані між рядками визначається як мінімальна кількість перетворень, то далі дамо означення цим перетворенням. Їх будемо розглядати в контексті операцій редагування тексту.

Означення 1. Нехай A та B — два символьних рядки. a_i — i-й символ у рядку A, b_i — i-й символ у рядку B. Визначимо наступні операції редагування на рядках:

Subst (A, k, b_j) — заміна символа, який знаходиться в k-й позиції рядка A, символом b_j.

Insert (A, c, k) — вставка символа c у рядок A у k-ту позицію (перед k-ю позицією);

Delete (A, k) — вилучення символа з k-ї позиції у рядку A.

При цьому операцію заміни можна реалізувати через операції вставки та видалення:

$$\text{Subst}\ (A, k, b_j) \Leftrightarrow \text{Delete}\ (A, k),\ \text{Insert}\ (A, c, k).$$

Для спрощення подальшого викладення операцію заміни будемо також позначати через Subst (a_i, b_j) — заміна літери a_i літерою b_j.

Приклад 1. Для перетворення рядка GTAGT у рядок TAGG необхідно зробити наступні операції редагування:

Delete (GTAGT, 1) = TAGT,

Subst (TAGT, 4, G) = TAGG.

Означення 2. Перетворенням рядка A у рядок B будемо називати послідовність операцій редагування, що перетворюють A у B, і будемо її позначати через $P(A, B)$.

Якщо позначити A = GTAGT, B = TAGG, то $P(A, B)$ = {Delete $(A, 1)$, Subst $(A, 4, G)$}.

Означення 3. Вагою перетворення $W_{P(A,B)}$ будемо називати загальну суму ваг операцій перетворень, задіяних у $P(A, B)$.

Ваги операцій редагування, наприклад, можна визначити наступним чином:

1. $W_{\text{Delete}} = W_{\text{Insert}} = 1,$

2. $W_{\text{Subst}} = \begin{cases} 1, & \text{якщо } a_i \ne b_j \\ 0, & \text{якщо } a_i = b_j. \end{cases}$

Означення 4. Відстанню Левенштейна між двома рядками A і B називається сума ваг найменшої трансформації $P(A, B)$:

$$D(A, B) = \min_{P(A,B)} \{W_{P(A,B)}\}.$$

Властивості відстані $D(A, B)$:

1. $D(A, B) = 0 \Leftrightarrow \mathrm{A} = \mathrm{B}$; очевидно;
2. $D(A, B) = D(B, A)$; операції редагування є симетричними;
3. $\forall\ C\ D(A, B) \le D(A, C) + D(C, B)$.

Приклад 2. Відстань Левенштейна між рядками GTAGT та TAGG дорівнює 2.

Обчислення відстані між рядками

Означення 5. Дано два рядки A і B довжиною n та m відповідно. Позначимо через $D(i, j)$, $1 \le i \le n$, $1 \le j \le m$ відстань між префіксами $\mathrm{A}_{1 \dots i}$ та $\mathrm{B}_{1 \dots j}$. Матрицю D будемо називати *матрицею відстаней* між рядками.

Обчислити значення елемента матриці $D(i, j)$ можна, розмірковуючи наступним чином (можливі лише наступні варіанти):

- замінити a_i на b_j та рекурсивно обчислити $D(i - 1, j - 1)$;

- видалити b_j та обчислити $D(i, j - 1)$;
- видалити a_i та обчислити $D(i - 1, j)$.

Таким чином отримаємо рекурсивну формулу обчислення відстані між префіксами (значення $D(n, m)$ точно дорівнює відстані між рядками A та B):

$$D(i, j) = \min\{D(i - 1, j) + W_{\text{Delete}}, D(i - 1, j - 1) + W_{\text{Subst}}, D(i, j - 1) + W_{\text{Insert}}\} \qquad (1)$$

Початковими умовами можна вважати наступні:

$D(0, 0) = 0$;

$D(i, 0) = i * W_{\text{Delete}}, \ 1 \le i \le n$;

$D(0, j) = j * W_{\text{Insert}}, \ 1 \le j \le m$.

Беручи до уваги запропоновані у попередньому розділі ваги операцій редагування (зауважимо, що їх можна було визначити й інакше), отримаємо рекурсивну формулу для обчислення елемента матриці $D(i, j)$.

Означення 6. *Графом редагування* будемо називати граф, кількість вершин якого дорівнює $m * n$, де m, n — розміри матриці D, а з вершини, яка відповідає елементу $D(i, j)$, йде ребро в ту (або ті) з вершин $D(i - 1, j)$, $D(i - 1, j - 1)$, $D(i, j - 1)$, з якої досягається мінімум при обчисленні функції (1).

П р и к л а д 3. Побудувати матрицю відстаней для рядків X = ACTGTA та Y = CTCAGTA. Якщо ми хочемо отримати з рядка X рядок Y, то запишемо X зліва по вертикалі, а Y — зверху по горизонталі. Запрограмувавши функцію (1), отримаємо наступну таблицю відстаней:

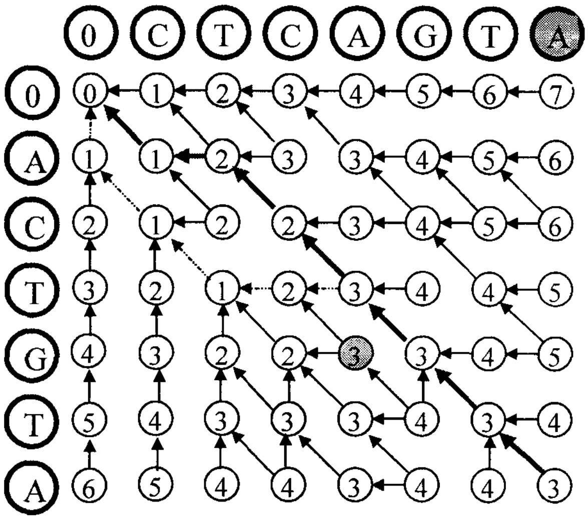
	0	C	T	C	A	G	T	A
0	0	1	2	3	4	5	6	7
A	1	1	2	3	3	4	5	6
C	2	1	2	2	3	4	5	6
T	3	2	1	2	3	4	4	5
G	4	3	2	2	3	3	4	5
T	5	4	3	3	3	4	3	4
A	6	5	4	4	3	4	4	3

Наприклад, при обчисленні значення $D(4, 4)$ маємо:

$$D(4, 4) = \min\{D(3, 3) + W_{\text{Subst}(G, A)}, D(3, 4) + 1, D(4, 3) + 1\} = \min\{2 + 1, 3 + 1, 2 + 1\} = \min\{3, 4, 3\} = 3.$$

Наведемо текстове перетворення, яке відповідає клітинці $D(4, 4)$ матриці. Оскільки оптимальне значення $D(4, 4)$ може бути обчислене в результаті руху з $D(3, 3)$ чи $D(4, 3)$, то для отримання зі слова ACTG слова CTCA за мінімальну кількість дій останньою операцією повинна бути операція редагування Subst(CTCG, 4, A) = CTCA або Insert(CTC, A, 4) = CTCA.

Граф редагування має наступний вигляд:



Використовуючи граф редагування, знайдемо послідовність операцій, які необхідно застосувати до слова ACTGTA, щоб отримати CTCAGTA. З матриці відстаней можна встановити, що найменша кількість операцій для такого перетворення дорівнює 3 (тому що $D(6, 7) = 3$). Виділимо жирними стрілками один із можливих шляхів від $D(6, 7)$ до $D(0, 0)$ на графі редагування. Перетворенню будуть відповідати ті і тільки ті стрілки, що з'єднують вершини, в яких записані різні значення.

1. $D(1, 1) \to D(0, 0)$ Subst (A, 1, C) = C:
   Subst (ACTGTA, 1, C) = CCTGTA;
2. $D(1, 2) \to D(1, 1)$ Insert (C, T, 2) = CT:
   Insert (CCTGTA, T, 2) = CTCTGTA;
3. $D(2, 3) \to D(1, 2)$ перетворення немає;
4. $D(3, 4) \to D(2, 3)$ Subst (CTCT, 4, A) = CTCA :
   Subst (CTCTGTA, 4, A) = CTCAGTA;
5. $D(4, 5) \to D(3, 4)$ перетворення немає;
6. $D(5, 6) \to D(4, 5)$ перетворення немає;
7. $D(6, 7) \to D(5, 6)$ перетворення немає.

Очевидно, що таке можливе оптимальне перетворення не є однозначним. Кількість мінімальних перетворень за кількістю операцій редагування дорівнює кількості шляхів від $D(6, 7)$ до $D(0, 0)$ на графі редагування.

Наприклад, якщо рухатися спочатку по жирних, а потім по пунктирних стрілках, то отримаємо наступну послідовність перетворень:

1. Delete (ACTGTA, 1) = CTGTA;
2. Insert (CTGTA, C, 3) = CTCGTA;
3. Insert (CTCGTA, A, 4) = CTCAGTA.

Найдовша спільна підпослідовність

Означення 7. Будемо говорити, що послідовність $A = a_1a_2 \ldots a_m$ є підпослідовністю $B = b_1b_2 \ldots b_n$, $(m \le n)$, якщо існує таке відображення $f: \{1 \ldots m\} \to \{1 \ldots n\}$, для якого $f(i) = j$ тоді і тільки тоді, коли $a_i = b_j$ і f є монотонно зростаючою функцією.

Означення 8. Підпослідовність L називається *найбільшою підпослідовністю* послідовностей A і B, якщо вона є підпослідовністю як A, так і B з найбільшою можливою довжиною.

Нехай $A_{1...n}$ та $B_{1...m}$ — дві послідовності. Якщо через $S(i, j)$ позначити довжину найбільшої підпослідовності послідовностей префіксів $A_{1...i}$ та $B_{1...j}$, то це значення можна отримати наступним чином:

- якщо $A[i] = B[j]$, то $S(i, j) = S(i-1, j-1) + 1$: треба шукати найдовшу підпослідовність у $A_{1...i-1}$ та $B_{1...j-1}$ та додати до цього значення 1;
- якщо $A[i] \neq B[j]$, то $S(i, j) = \max\{S(i-1, j), S(i, j-1)\}$; треба шукати найдовшу підпослідовність у $A_{1...i-1}$ та $B_{1...j}$ та $A_{1...i}$ та $B_{1...j-1}$ і взяти найбільшу із них.

Очевидно, що найдовша підпослідовність слова та порожнього слова дорівнює нулю. Таким чином, для обчислення найдовшої спільної підпослідовності отримаємо наступну рекурсивну формулу:

$$S(i, j) = \max\{S(i-1, j), S(i-1, j-1) + $$
$$+ 1, S(i, j-1)\}.$$

Початкові умови визначаються наступним чином:

$$S(i, 0) = 0;\ 0 \le i \le n;$$
$$S(0, j) = 0;\ 0 \le i \le n.$$

Отже, для знаходження найбільшої спільної підпослідовності достатньо поміняти вагові значення для операцій редагування у задачі обчислення відстані між рядками (сама формула залишається тією ж самою).

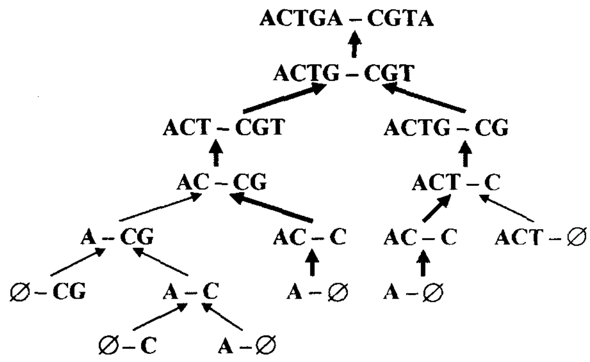
Означення 9. *Деревом обчислення* найбільшої спільної підпослідовності будемо називати бінарне дерево, у вузлах якого будуть знаходитися дві поточні послідовності, для яких розв'язується задача. Якщо останні літери двох послідовностей у вершині однакові, то така вершина має одного сина (такий зв'язок будемо називати *вертикальним* згідно з положенням стрілки), якщо ні — то двох синів, що визначаються відповідно до правил обчислення найдовшої підпослідовності.

Для знаходження найдовшої спільної підпослідовності необхідно знайти такий шлях від кореня до листа, який містить найбільшу кількість вертикальних зв'язків, оскільки кожен з них відповідає літері, яка входить до спільної підпослідовності.

П р и к л а д 4. Знайти найбільшу підпослідовність у рядках ACTGA та CGTA.

Найбільшу кількість вертикальних зв'язків (три) містять шляхи, виділені жирними стрілками. Відповіддю буде або CGA (якщо рухатися по правому піддереву), або CTA (при русі по лівому піддереву) — найбільша спільна підпослідовність для послідовностей ACTGA та CGTA.

Дерево обчислень має вигляд:



Зіставлення

Означення 10. Зіставленням двох рядків A і B будемо називати таку вставку символів проміжку в середину чи в кінець рядків, що коли записати рядок B під рядком A, то кожна літера одного рядка зіставляється або з такою ж літерою іншого рядка, або з проміжком.

Зауваження: проміжок не може зіставлятися з проміжком.

П р и к л а д 5. Рядки ACTGTA та CTCAGTA можуть бути зіставлені наступним чином (символ проміжку позначено через —):

A C T G T A — — —

— C T — C A G T A.

При цьому три літери C, T, A (які утворюють одну із найбільших підпослідовностей, приклад 4) другого слова знаходяться під такими ж літерами першого слова.

Означення 11. *Вирівнюванням* двох рядків A і B будемо називати такі рядки A' та B', що кількість літер у A' та B' однакова ($|A'| = |B'|$), а після вилучення усіх проміжків з A' отримаємо A, а з B' отримаємо B.

Для прикладу 5 вирівнюванням рядків $X =$ $=$ ACTGTA та $Y =$ CTCAGTA будуть рядки $X' =$ $=$ ACTGT—A——— та $Y' =$ —CT——CAGTA.

Означення 12. *Вагою зіставлення* літер A і B будемо називати число, яке дорівнює 1 — кількість операцій редагування, необхідних для отримання B з A.

$$\text{Тобто, } W_c(A, B) = \begin{cases} 0, & \text{якщо } A \neq B \\ 1, & \text{якщо } A = B. \end{cases}$$

Звідси зазначимо, що вага зіставлення проміжку з довільною літерою дорівнює 1: $W_c(A, —) =$ $= W_c(—, A) = 1$ для довільної літери A.

Означення 13. *Вагою зіставлення* вирівнених рядків A' та B' будемо називати суму ваг зіставлень літер, що знаходяться на однакових місцях:

$$W_c(A', B') = \sum_{i=1}^{n} W(A'_i, B'_i),$$

де $n = |A'| = |B'|$.

Означення 14. *Оптимальним зіставленням* (або оптимальним вирівнюванням) рядків A та B будемо називати таке їх зіставлення, при якому його вага набуває найбільшого значення.

Очевидно, що вага зіставлення рядків A та B (в розумінні означення 12) буде тим більшою, чим більше однакових символів буде знаходитися один під іншим при записі B' під A'. Тобто зіставлення A та B буде оптимальним, якщо буде знайдено найбільшу підпослідовність у рядках A та B й при їх вирівнюванні всі символи цієї найбільшої підпослідовності знаходитимуться один під іншим.

Приклад 6. Рядки ACTGA та CGTA мають два оптимальні зіставлення вагою 3, оскільки ці рядки мають найбільшу підпослідовність або CGA, або CTA. Підпослідовності CGA буде відповідати зіставлення

$$\begin{array}{c} A\ C\ T\ G - A - C - G\ T\ A, \end{array}$$

а послідовності CTA — зіставлення

$$A\ C - T\ G\ A - C\ G\ T - A.$$

1. *Gusfield Dan.* Algorithms on Strings, Trees, and Sequences.— Cambridge University Press, 1997.

2. *Hirschberg D. S.* Algorithms for the longest common subsequence problem // J. ACM.—1977,—N24.—P. 664—675.

Medvedev M.

PROBLEMS, CONNECTEN WITH LEVINSHTEIN DISTANCE

The problem of Levenshtein distance evaluation is presented in this article. The problems of finding the greatest common subsequence and optimal allignment between the words are researched.