МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
КРЕМЕНЧУЦЬКИЙ НАЦІОНАЛЬНИЙ
УНІВЕРСИТЕТ ІМЕНІ МИХАЙЛА
ОСТРОГРАДСЬКОГО

Кафедра комп’ютерної інженерії та електроніки

ЗВІТ З ПРАКТИЧНОЇ РОБОТИ №7

з навчальної дисципліни
«Алгоритми та методи обчислень»

Тема «Алгоритми на рядках. Динамічне програмування»

Студент гр. КІ-24-1, Варакута О.О
Викладач, Сидоренко В. М

Кременчук 2025

Тема: Алгоритми на рядках. Динамічне програмування

Мета роботи: Набути практичних навичок знаходження найдовшої спільної підпослідовності двох рядків за допомогою алгоритму динамічного програмування та оцінювання його асимптотичної складності.

Теоретичні відомості:

Підпослідовністю рядка називається послідовність символів, отримана шляхом видалення деяких символів без зміни порядку інших.
Найдовша спільна підпослідовність двох рядків — це максимальна за довжиною послідовність символів, яка є підпослідовністю кожного з рядків.

Для розв’язання задачі використовується алгоритм динамічного програмування, який будує таблицю оптимальних рішень для всіх підрядків двох рядків.

Хід Роботи

Завдання Варіант 5:

Маємо дві короткі послідовності символів:
X = «XYBAC» Y = «ABXC» .Потрібно знайти найдовшу спільну підпослідовність символів, використовуючи алгоритм динамічного програмування.

Ров’язання задачі :

Крок 1. Побудова таблиці

Нехай dp[i][j] — довжина найдовшої спільної підпослідовності для перших і символів рядка X та перших j символів рядка Y.

Розмір таблиці:

$(|X| + 1) \times (|Y| + 1) = 6 \times 5$

Перший рядок і перший стовпець таблиці заповнюються нулями.

Крк 2. Заповнення таблиці

Якщо $X[i-1] = Y[j-1]$, тоді

$dp[i][j] = dp[i-1][j-1] + 1$

Якщо X[i−1] ≠ Y[j−1], тоді
dp[i][j] = max(dp[i−1][j], dp[i][j−1])

Таблиця заповнюється зліва направо, зверху вниз.

Таблиця dp:

X \ Y	**–**	**A**	**B**	**X**	**C**
–	0	0	0	0	0
X	0	0	0	1	1
Y	0	0	0	1	1
B	0	0	1	1	1
A	0	1	1	1	1
C	0	1	1	1	2

Пояснення заповнення таблиці

У комірці dp[1][3] значення дорівнює 1, оскільки символ X збігається з символом X у другому рядку.

У комірці dp[3][2] значення дорівнює 1, оскільки символ B збігається з символом B.

У комірці dp[4][1] значення дорівнює 1, оскільки символ A збігається з символом A.

У комірці dp[5][4] значення дорівнює 2, оскільки символ C збігається з символом C, і dp[5][4] = dp[4][3] + 1.

Крок 3. Відновлення найдовшої спільної підпослідовності

Відновлення здійснюється шляхом руху з правого нижнього кута таблиці dp до початкової комірки.

Значення dp[5][4] = 2 означає, що довжина найдовшої спільної підпослідовності дорівнює 2.

Однією з можливих найдовших спільних підпослідовностей є:

X C

Також можлива підпослідовність:

A C

Обидві мають однакову максимальну довжину.

Результат:

Для рядківX = «XYBAC» Y = «ABXC»

довжина найдовшої спільної підпослідовності дорівнює 2.

Найдовша спільна підпослідовність: AC

Асимптотична складність алгоритму

Часова складність:
O(m · n), де m і n — довжини рядків X і Y відповідно.

Просторова складність:
O(m · n), оскільки використовується двовимірна таблиця.

Реалізація алгоритму мовою Python:

```
def longest_common_subsequence(x, y):
    m = len(x)
    n = len(y)

    dp = [[0] * (n + 1) for _ in range(m + 1)]

    for i in range(1, m + 1):
        for j in range(1, n + 1):
            if x[i - 1] == y[j - 1]:
                dp[i][j] = dp[i - 1][j - 1] + 1
            else:
                dp[i][j] = max(dp[i - 1][j], dp[i][j - 
1])

    lcs = []
    i, j = m, n
    while i > 0 and j > 0:
```

```
            if x[i - 1] == y[j - 1]:
                lcs.append(x[i - 1])
                i -= 1
                j -= 1
            elif dp[i - 1][j] > dp[i][j - 1]:
                i -= 1
            else:
                j -= 1

    lcs.reverse()
    return "".join(lcs)

x = "XYBAC"
y = "ABXC"
print(longest_common_subsequence(x, y))
```

Виведений результат роботи алгоритма: AC

Висновок:

У ході виконання практичної роботи було знайдено найдовшу спільну підпослідовність двох заданих послідовностей символів за допомогою алгоритму динамічного програмування. Побудовано таблицю dp, виконано ручний аналіз та отримано коректний результат, що підтверджує правильність застосованого алгоритму.

Контрольні питання:

1.У чому полягає задача знаходження найдовшої спільної підпослідовності (LCS)?

Задача знаходження найдовшої спільної підпослідовності полягає у визначенні максимально довгої послідовності символів, яка є підпослідовністю двох заданих послідовностей. Підпослідовність утворюється шляхом видалення деяких символів без зміни порядку решти. Метою задачі є визначення як довжини цієї підпослідовності, так і самих символів, що до неї входять. Задача LCS широко використовується для порівняння рядків і аналізу подібності даних.

2.Які головні методи можна використовувати для знаходження найдовшої спільної підпослідовності?

Для знаходження найдовшої спільної підпослідовності застосовуються такі основні методи:повний перебір усіх можливих підпослідовностей (теоретичний метод, практично не використовується через експоненційну складність); рекурсивні алгоритми без запам’ятовування проміжних результатів; алгоритми динамічного програмування; оптимізовані алгоритми, зокрема алгоритм Хіршберга (у деяких джерелах помилково називається алгоритмом Хаббарда). Найпоширенішим на практиці є метод динамічного програмування.

3.Як працює алгоритм динамічного програмування для знаходження LCS?

Алгоритм динамічного програмування для LCS базується на побудові двовимірної таблиці dp, де dp[i][j] зберігає довжину найдовшої спільної підпослідовності для перших і символів першого рядка та перших j символів другого рядка. Таблиця заповнюється поступово, використовуючи такі правила: якщо відповідні символи збігаються, значення комірки збільшується на одиницю відносно діагональної комірки; якщо символи не збігаються, вибирається максимальне значення з сусідніх комірок. Після заповнення таблиці виконується зворотний прохід для відновлення самої підпослідовності.

4.Як працює алгоритм Хаббарда для знаходження LCS?

Під алгоритмом Хаббарда у більшості навчальних матеріалів мається на увазі алгоритм Хіршберга. Це оптимізований варіант алгоритму динамічного програмування для LCS, який зменшує використання пам’яті. Алгоритм Хіршберга застосовує метод «розділяй і володарюй»: рядок ділиться на дві

частини, для яких обчислюються проміжні значення довжин LCS з використанням лише двох рядків таблиці dp. Потім рекурсивно відновлюється сама підпослідовність. Завдяки цьому алгоритм використовує лінійну пам’ять замість квадратичної.

5.Які переваги та недоліки алгоритмів динамічного програмування та Хаббарда для знаходження LCS?

Алгоритм динамічного програмування є простим для розуміння та реалізації, дозволяє легко відновити підпослідовність і має гарантований результат. Його основним недоліком є високе споживання пам’яті O(m·n), що може бути проблемою для великих рядків.

Алгоритм Хіршберга має значно менші вимоги до пам’яті O(m + n), що робить його ефективним для великих обсягів даних. Водночас він складніший у реалізації, має більшу кількість рекурсивних викликів і складнішу логіку відновлення підпослідовності.

6.Які існують практичні застосування для задачі знаходження найдовшої спільної підпослідовності?

Задача LCS має широке практичне застосування. Вона використовується у системах порівняння текстів і пошуку відмінностей між файлами, у системах контролю версій, у біоінформатиці для порівняння ДНК та білкових послідовностей, у лінгвістиці та обробці природної мови, а також у задачах аналізу подібності даних і розпізнавання шаблонів.