МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

КРЕМЕНЧУЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

ІМЕНІ МИХАЙЛА ОСТРОГРАДСЬКОГО

НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ЕЛЕКТРИЧНОЇ ІНЖЕНЕРІЇ

ТА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

Кафедра комп’ютерної інженерії та електроніки

ЗВІТ З ПРАКТИЧНИХ РОБІТ

з навчальної дисципліни

«Алгоритми та методи обчислень»

Тема «Алгоритми на рядках.»

Студентка гр. КН-23-1 ПІБ Варич А.І

Викладач к. т. н., доц. В. М. Сидоренко

Кременчук 2024

**ЗМІСТ**

[1 Завдання 3](#_Toc167121908)

[2 Контрольні запитання 5](#_Toc167121909)

# Завдання

Варіант 6

**Постановка задачі:** Маємо дві короткі послідовності символів: «XYBAC» і «ABXC». Знайти найдовшу спільну підпослідовності символів, використовуючи алгоритм Хаббарда.

1. Створюю матрицю C розміром (m+1) × (n+1), де m і n - довжини двох послідовностей. Всі елементи матриці ініціалізуються нулями.
2. Якщо символи в обох послідовностях співпадають (X[i-1] == Y[j-1]), то C[i][j] = C[i-1][j-1] + 1.
3. Інакше C[i][j] = max(C[i-1][j], C[i][j-1]).
4. Після заповнення матриці, відновлюю найдовшу спільну підпослідовність, рухаючись від правого нижнього кута матриці до лівого верхнього, слідуючи шляхом, який визначається значеннями в матриці.
5. Дано дві послідовності: X = "XYBAC" та Y = "ABXC".

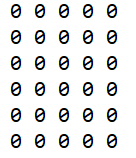


рис. 1 - Нульова матриця

1. Порівнюю кожен символ з кожним і заповнюю матрицю:

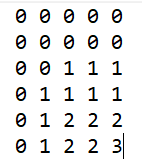


рис. 2 - Заповнення матриці

1. Кроки самого заповнення:

X[0] = 'X' не дорівнює Y[0] = 'A', тому C[1][1] = max(C[0][1], C[1][0]) = 0.

X[0] = 'X' не дорівнює Y[1] = 'B', тому C[1][2] = max(C[0][2], C[1][1]) = 0.

X[0] = 'X' дорівнює Y[2] = 'X', тому C[1][3] = C[0][2] + 1 = 1.

X[0] = 'X' не дорівнює Y[3] = 'C', тому C[1][4] = max(C[0][4], C[1][3]) = 1.

1. Відновлення LCS: Починаю з C[5][4] = 3 і відновлюю шлях

X[4] = 'C' дорівнює Y[3] = 'C', тому додаю 'C' до LCS.

Переміщаю до C[4][3] = 2.

X[3] = 'A' не дорівнює Y[2] = 'X', тому переміщаю до C[4][2] = 2.

X[3] = 'A' дорівнює Y[1] = 'B', тому додаю 'A' до LCS.

Переміщаю до C[3][1] = 1.

X[2] = 'B' дорівнює Y[1] = 'B', тому додаю 'B' до LCS.

Отримую LCS = "BAC".

1. Підсумую:

Найдовша спільна підпослідовність для «XYBAC» і «ABXC» є "BAC".

# Контрольні запитання

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

Задача полягає у визначенні найдовшої послідовності символів, яка є підпослідовністю двох заданих послідовностей. Ця підпослідовність повинна зберігати порядок появи символів в обох послідовностях.

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

Динамічне програмування.

Рекурсивні алгоритми з мемоізацією.

Метод Галгера та Лонґеста.

Алгоритм Хаббарда.

1. Як працює алгоритм динамічного програмування для знаходження LCS?

Алгоритм використовує двовимірну матрицю для зберігання довжин LCS для підпослідовностей. Матриця заповнюється шляхом порівняння символів двох послідовностей і вибору максимального значення з можливих варіантів. Результат зберігається в правому нижньому куті матриці.

1. Як працює алгоритм Хаббарда для знаходження LCS?

Алгоритм Хаббарда також використовує матрицю для зберігання проміжних результатів. Якщо символи збігаються, значення збільшується на одиницю від діагонального сусіда. Інакше вибирається максимальне значення зліва або зверху. Результат відновлюється, рухаючись від кінця до початку матриці.

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

Переваги: Динамічне програмування: висока ефективність, точність, і універсальність.

Алгоритм Хаббарда: простота реалізації, зрозумілість.

Недоліки: Обидва алгоритми можуть споживати багато пам'яті для великих послідовностей.

Можуть бути повільними для дуже довгих послідовностей без оптимізацій.

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

Вирівнювання послідовностей ДНК у біоінформатиці.

Порівняння та виявлення схожості між текстовими файлами.

Розробка систем контролю версій програмного забезпечення.

Виявлення схожості між музичними творами.

Оптимізація пошукових алгоритмів для пошуку підпослідовностей у базах даних.