Київський національний університет імені Тараса Шевченка  
Факультет комп’ютерних наук та кібернетики  
Кафедра системного аналізу та теорії прийняття рішень

Звіт  
з лабораторної роботи № 1  
на тему:

**«Рядки і послідовності»**

Студента другого курсу

групи К-23(2)

Міщука Романа Андрійовича

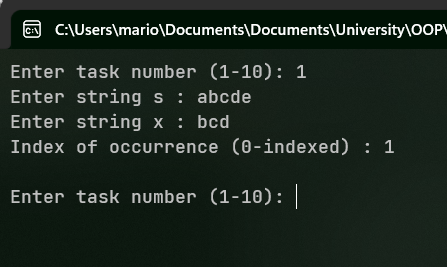
Факультету комп’ютерних наук

та кібернетики

Київ – 2023

## Пошук точного підрядка в рядку

**Опис алгоритму:** знаходження точного входження рядка x в рядок s. Для цього використовується алгоритм Моріса-Прата. Він є покращенням наївного алгоритму. Це покращення досягається за допомогою попереднього обрахування довжини зрушення у випадку виявлення розбіжності зразка.

**Результат роботи:**

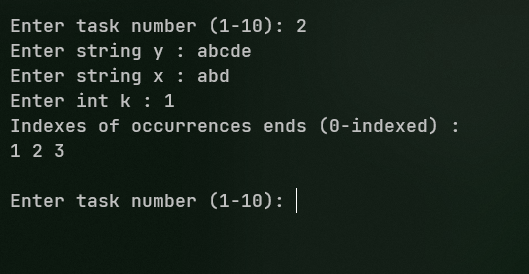
01234

abcde

bcd

## Нечіткий пошук (близькі за відстанню)

**Опис алгоритму:** знаходження входжень зразку x у рядок y, за умови, що входження може мати k відмінностей від зразка. Для цього використовується алгоритм Ландау-Вішкіна. Алгоритм ґрунтується на перефразуванні наївного методу динамічного програмування, що дозволяє використовувати суфіксні дерева для пришвидшення. Проте через це необхідно буде перед запуском алгоритму побудувати дерево для y#x$.

**Результат роботи:**

01234

***abcd***e

ab d

01234

***abc***de

abd

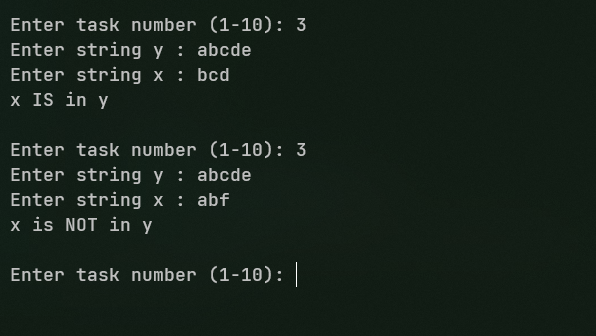
01234

***ab***cde

abd

## Перевірка на підпослідовність

**Опис алгоритму:** перевірка, чи можливо видаленням деякої кількості символів із рядка y отримати рядок x. Алгоритм реалізовано простим скануванням символів у рядку y.

**Результат роботи:**

01234

abcde

abf

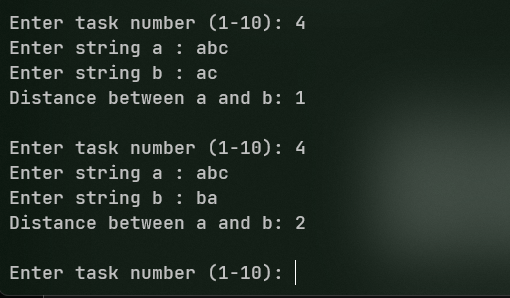
01234

abcde

bcd

## Загальні підпослідовності. Відстань

**Опис алгоритму:** знаходження відстані (кількості відмінностей) між рядками a та b. Застосовується алгоритм Вагнера-Фішера. Для виконання завдання алгоритм методом динамічного програмування рахує відстані між різними комбінаціями префіксів a та b.

**Результат роботи:**

012

abc

ba

012

abc

a c

## Пошук lis та lcs

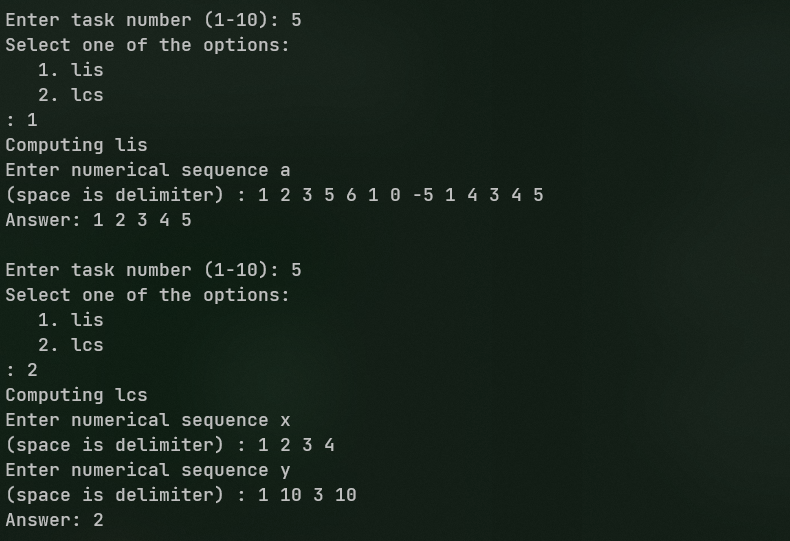
*lis* – longest incremental subsequence – найбільша зростаюча підпослідовність;

lcm – longest common subsequence – найбільша спільна підпослідовність;

**Опис алгоритмів:**

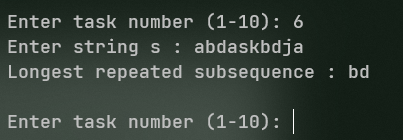
Для знаходження lis послідовності чисел a, застосовується алгоритм Робінсона-Шенстеда. Він ґрунтується на застосуванні методу динамічного програмування (зберігання найменшого члену зростаючих підпослідовностей певної довжини) для прискорення роботи.

Для знаходження lcm двох послідовностей чисел x та y також використовується метод динамічного програмування. Він рахує довжину спільної підпослідовності для різних комбінацій префіксів x та y.

**Результат роботи:**

## Максимальний повторюваний підрядок

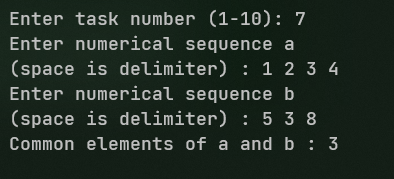
**Опис алгоритму:** знаходження такого підрядку максимальної довжини, який би повторювався хоча б 2 рази. Ця задача є досить легко виконуваною, використовуючи методи, реалізовані в пункті 2 лабораторної роботи. Адже завдання відповідає пошуку внутрішної вершини суфіксного дерева рядку. Критерієм пошуку є довжина підрядку, якому відповідає ця вершина.

**Результат роботи:**

abdaskbdja

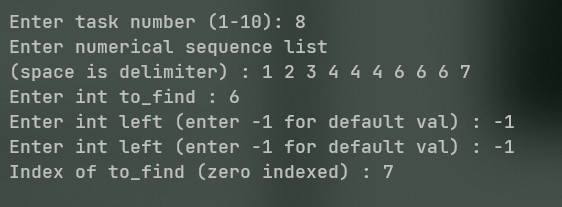
## Загальні елементи двох масивів

**Опис алгоритму:** знаходження таких чисел, які б одночасно належали послідовностям a та b. Для розв’язання використовується сортування елементів послідовностей, з подальшим ітеруванням по ним.

**Результат роботи:**

## Бінарний пошук

**Опис алгоритму:** знаходження індекса числа to\_find у відсортованій послідовності чисел.

**Результат роботи:**

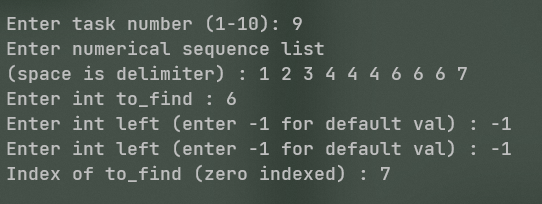
*0 1 2 3 4 5 6 7 8 9*

1 2 3 4 4 4 6 6 6 7

6

## Інтерполяційний пошук

**Опис алгоритму:** знаходження індекса числа to\_find у відсортованій послідовності чисел. Схожий за принципом роботи на бінарний пошук. Основна відмінність полягає у стрибках, які робить цей алгоритм. Інтерполяційний пошук ділить відрізок не навпіл, а інтерполює його (вважаючи послідовність арифметичною прогресією) і переходить ймовірного індекса елемента. Це дозволяє прискорити пошук.

**Результат роботи:**

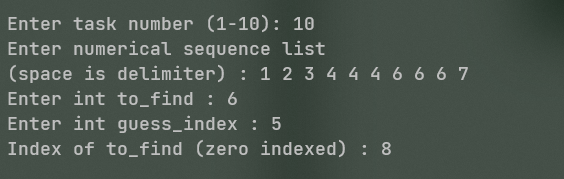
*0 1 2 3 4 5 6 7 8 9*

1 2 3 4 4 4 6 6 6 7

6

## Бінарний пошук з визначенням найближчих вузлів

**Опис алгоритму:** знаходження індекса числа to\_find у відсортованій послідовності чисел, за зразком бінарного пошуку. Проте крім послідовності та числа, цей алгоритм також отримує додаткову інформацію у вигляді приблизного місця знаходження числа guess\_index. Це дозволяє запускати бінарний пошук на вужчому відрізку, і, відповідно, прискорити алгоритм.

**Результат роботи:**

*0 1 2 3 4 5 6 7 8 9*

1 2 3 4 4 4 6 6 6 7

6