ЛЬВІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ імені ІВАНА ФРАНКА

Факультет прикладної математики та інформатики

Теорія алгоритмів

ЛАБОРАТОРНА РОБОТА №2

Тема: «Побудова алгоритмів із заданими оцінками»

Виконав:

Ст. *Лук'янчук Денис*

Група ПМІ-23

2025

Тема: «Побудова алгоритмів із заданими оцінками»

Мета роботи: Розробка та реалізація ефективних алгоритмів для обробки відсортованих масивів із заданими обмеженнями на часову складність. Вдосконалення навичок побудови алгоритмів для об'єднання, пошуку та порівняння даних, зберігаючи лінійну складність O(m + n) або подібну, а також на практичне застосування цих алгоритмів у програмуванні.

Завдання 1: Об'єднання двох неспадних масивів

Хід роботи:

1. Визначити розміри масивів A та B

2. Створити порожній результуючий масив C для зберігання m + n елементів

3. Ініціалізувати два вказівники: i = 0 для A, j = 0 для B

4. Порівнювати елементи A[i] та B[j]:

   - Якщо один масив закінчився, додати всі залишки іншого

   - Інакше додати менший елемент до C і зсунути відповідний вказівник

5. Повторювати, доки не оброблені всі елементи

```python
def merge_arrays(A, B):
    m, n = len(A), len(B)
    C = []
    i = j = 0
    while i < m or j < n:
        if i >= m:
            C.append(B[j])
            j += 1
        elif j >= n:
            C.append(A[i])
            i += 1
        else:
            if A[i] <= B[j]:
                C.append(A[i])
                i += 1
            else:
                C.append(B[j])
                j += 1
    return C
```

Завдання 2: Найменше спільне число у трьох неспадних масивах

Хід роботи:

1. Визначити розміри масивів A, B, C

2. Ініціалізувати три вказівники: i = 0 для A, j = 0 для B, k = 0 для C

3. Порівняти A[i], B[j], C[k]:

- Якщо всі рівні, повернути це число як найменше спільне

- Інакше зсунути вказівники для найменшого значення

4. Повторювати, доки не знайдено спільне число або не закінчився один із масивів

```python
def smallest_common(A, B, C):
    m, n, p = len(A), len(B), len(C)
    i = j = k = 0
    while i < m and j < n and k < p:
        if A[i] == B[j] == C[k]:
            return A[i]

        min_val = min(A[i], B[j], C[k])
        if A[i] == min_val:
            i += 1
        if B[j] == min_val:
            j += 1
        if C[k] == min_val:
            k += 1
    return None
```

Завдання 3: Пошук числа у прямокутному масиві

Хід роботи:

1. Визначити розміри масиву A (m рядків, n стовпців).

2. Почати з правого верхнього кута: i = 0, j = n - 1

3. Порівняти A[i][j] із шуканим числом x:

- Якщо дорівнює, повернути "знайдено"

- Якщо більше, зсунути вліво (j -= 1)

- Якщо менше, зсунути вниз (i += 1)

4. Повторювати, доки не вийдемо за межі або не знайдемо x

5. Повернути результат ("так" або "ні")

```python
def search_in_matrix(A, x):
    m, n = len(A), len(A[0])
    i, j = 0, n - 1
    while i < m and j >= 0:
        if A[i][j] == x:
            return True
        elif A[i][j] > x:
            j -= 1
        else:
            i += 1
    return False
```

Завдання 4: Чи можна отримати B з A, викресливши елементи

Хід роботи:

1. Визначити розміри масивів A та B

2. Ініціалізувати вказівники: i = 0 для A, j = 0 для B

3. Порівняти A[i] і B[j]:

   - Якщо рівні, зсунути обидва вказівники (i += 1, j += 1)

   - Якщо A[i] < B[j], пропустити елемент A (i += 1)

   - Якщо A[i] > B[j], повернути "ні"

4. Повторювати, доки не закінчиться A або B

5. Перевірити, чи весь B збігся (j == n), і повернути результат

```python
def can_obtain_B_from_A(A, B):
    m, n = len(A), len(B)
    i = j = 0
    while i < m and j < n:
        if A[i] == B[j]:
            i += 1
            j += 1
        elif A[i] < B[j]:
            i += 1
        else:
            return False
    return j == n
```

Приклад Unit-тесту(без помилок):

```
PS C:\Users\denys> & C:/Users/denys/AppData/Local/Programs/Python/Python313
.............
----------------------------------------------------------------------
Ran 13 tests in 0.001s

OK
PS C:\Users\denys>
```

Приклад Unit-тесту(з помилкою):

```
PS C:\Users\denys> & C:/Users/denys/AppData/Local/Programs/Python/Python313/python.exe "c:/Теорія ал
..........F.
======================================================================
FAIL: test_smallest_common_first_common (__main__.TestAlgorithms.test_smallest_common_first_common)
----------------------------------------------------------------------
Traceback (most recent call last):
  File "c:\Теорія алгоритмів\Unittest_2.py", line 22, in test_smallest_common_first_common
    self.assertEqual(smallest_common([7, 2, 3], [1, 2, 4], [1, 3, 5]), 1)
    ~~~~~~~~~~~~~~~~~~~^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
AssertionError: None != 1

----------------------------------------------------------------------
Ran 13 tests in 0.001s

FAILED (failures=1)
PS C:\Users\denys>
```

Висновок: У результаті виконання завдання були розроблені та реалізовані чотири алгоритми для обробки відсортованих масивів із заданими обмеженнями на часову складність. Кожен алгоритм (об’єднання двох масивів, пошук найменшого спільного числа у трьох масивах, пошук у прямокутному масиві та перевірка можливості отримання одного масиву з іншого) має лінійну складність O(m + n) або аналогічну, що відповідає умовам задачі. Реалізація на Python підтвердила коректність алгоритмів через юніт-тести, які охопили базові, крайові та спеціальні випадки. Завдання дозволило поглибити розуміння роботи з відсортованими структурами даних, оптимізувати алгоритми за часом виконання та practically застосувати теоретичні знання у програмуванні. Отримані результати демонструють ефективність запропонованих рішень та їхню відповідність поставленим вимогам.