Звіт

про виконання лабораторних та практичних робіт блоку № 6

На тему: «Динамічні структури (Черга, Стек, Списки, Дерево). Алгоритми обробки динамічних структур.»

з дисципліни: «Основи програмування»

до:

ВНС Лабораторної Роботи № 10

Алготестер Лабораторної Роботи № 5

Алготестер Лабораторної Роботи № 7-8

Практичних Робіт до блоку № 6

Виконав:

Студент групи ШІ-13

Скічко Михайло Вікторович

Львів 2024

Тема роботи:

Динамічні структури (Черга, Стек, Списки, Дерево). Алгоритми обробки динамічних структур.

Мета роботи:

Реалізувати різні динамічні структури і функції для роботи з ними.

Теоретичні відомості:

1) Теми, необхідні для виконання роботи:

1. Основи Динамічних Структур Даних:
    - Вступ до динамічних структур даних: визначення та важливість
    - Виділення пам'яті для структур даних (stack і heap)
    - Приклади простих динамічних структур: динамічний масив
2. Стек:
    - Визначення та властивості стеку
    - Операції push, pop, top: реалізація та використання
    - Приклади використання стеку: обернений польський запис, перевірка балансу дужок
    - Переповнення стеку
3. Черга:
    - Визначення та властивості черги
    - Операції enqueue, dequeue, front: реалізація та застосування
    - Приклади використання черги: обробка подій, алгоритми планування
    - Розширення функціоналу черги: пріоритетні черги
4. Зв'язні Списки:
    - Визначення однозв'язного та двозв'язного списку
    - Принципи створення нових вузлів, вставка між існуючими, видалення, створення кільця(circular linked list)
    - Основні операції: обхід списку, пошук, доступ до елементів, об'єднання списків
    - Приклади використання списків: управління пам'яттю, FIFO та LIFO структури
5. Дерева:
    - Вступ до структури даних "дерево": визначення, типи

- Бінарні дерева: вставка, пошук, видалення
- Обхід дерева: в глибину (preorder, inorder, postorder), в ширину
- Застосування дерев: дерева рішень, хеш-таблиці
- Складніші приклади дерев: AVL, Червоно-чорне дерево

6. Алгоритми Обробки Динамічних Структур:
   - Основи алгоритмічних патернів: ітеративні, рекурсивні
   - Алгоритми пошуку, сортування даних, додавання та видалення елементів

2) Джерела використані для ознайомлення з вищезазначеними темами:

- Всю інформацію до теоретичних відомостей я отримав на лекційних, практичних парах, та самостійне вивчення. Зокрема сайти https://acode.com.ua/ та https://www.w3schools.com/

Виконання роботи:

Опрацювання завдання та вимог до програм та середовища:

Завдання №1 – VNS Lab 10 Task 1 variant 23

Задача

Написати програму, у якій створюються динамічні структури й виконати їхню обробку у відповідності зі своїм варіантом.

Для кожного варіанту розробити такі функції:

1. Створення списку.
2. Додавання елемента в список (у відповідності зі своїм варіантом).
3. Знищення елемента зі списку (у відповідності зі своїм варіантом).
4. Друк списку.
5. Запис списку у файл.
6. Знищення списку.
7. Відновлення списку з файлу.

Порядок виконання роботи

1. Написати функцію для створення списку. Функція може створювати порожній список, а потім додавати в нього елементи.
2. Написати функцію для друку списку. Функція повинна передбачати вивід повідомлення, якщо список порожній.
3. Написати функції для знищення й додавання елементів списку у відповідності зі своїм варіантом.
4. Виконати зміни в списку й друк списку після кожної зміни.
5. Написати функцію для запису списку у файл.
6. Написати функцію для знищення списку.
7. Записати список у файл, знищити його й виконати друк (при друці повинне бути видане повідомлення "Список порожній").
8. Написати функцію для відновлення списку з файлу.
9. Відновити список і роздрукувати його.
10. Знищити список.

23. Запису в лінійному списку містять ключове поле типу *char (рядок символів). Сформувати двонаправлений список. Знищити елемент із заданим ключем. Додати К елементів після елемента із заданим ключем.

Завдання №2 – Algotester Lab 5 variant 2

Задача

Limits: 1 sec., 256 MiB

В пустелі існує незвичайна печера, яка є двохвимірною. Її висота це N, ширина - M.

Всередині печери є пустота, пісок та каміння. Пустота позначається буквою O , пісок S і каміння X;

Одного дня стався землетрус і весь пісок посипався вниз. Він падає на найнижчу клітинку з пустотою, але він не може пролетіти через каміння.

Ваше завдання сказати як буде виглядати печера після землетрусу.

Input

У першому рядку 2 цілих числа N та M - висота та ширина печери

У N наступних рядках стрічка row_i яка складається з N цифер - i-й рядок матриці, яка відображає стан печери до землетрусу.

Output

N рядків, які складаються з стрічки розміром M - стан печери після землетрусу.

Constraints

$1 \leq N, M \leq 1000$

$|row_i| = M$

$row_i \in \{X, S, O\}$

Завдання №3 – Algotester Lab 7-8 variant 1

Задача

Limits: 2 sec., 256 MiB

Ваше завдання - власноруч реалізувати структуру даних "Двозв’язний список".
Ви отримаєте Q запитів, кожен запит буде починатися зі слова-ідентифікатора, після якого йдуть його аргументи.

Вам будуть поступати запити такого типу:

- **Вставка**:
  Ідентифікатор - $insert$
  Ви отримуєте ціле число $index$ елемента, на місце якого робити вставку.
  Після цього в наступному рядку рядку написане число N - розмір списку, який треба вставити.
  У третьому рядку N цілих чисел - список, який треба вставити на позицію $index$.
- **Видалення**:
  Ідентифікатор - $erase$
  Ви отримуєте 2 цілих числа - $index$, індекс елемента, з якого почати видалення та n - кількість елементів, яку треба видалити.
- **Визначення розміру**:
  Ідентифікатор - $size$
  Ви не отримуєте аргументів.
  Ви виводите кількість елементів у списку.
- **Отримання значення** i-го елементу
  Ідентифікатор - get
  Ви отримуєте ціле число - $index$, індекс елемента.
  Ви виводите значення елемента за індексом.
- **Модифікація значення** i-го елементу
  Ідентифікатор - set
  Ви отримуєте 2 цілих числа - індекс елемента, який треба змінити, та його нове значення.
- **Вивід списку на екран**
  Ідентифікатор - $print$
  Ви не отримуєте аргументів.
  Ви виводите усі елементи списку через пробіл.
  Реалізувати використовуючи перегрузку оператора $<<$

Input

Ціле число Q - кількість запитів.
У наступних рядках Q запитів у зазначеному в умові форматі.

Output

Відповіді на запити у зазначеному в умові форматі.

Constraints

$0 \le Q \le 10^3$
$0 \le l_i \le 10^3$
$\|l\| \le 10^3$

Завдання №4 – Class Practice Work

Задача

Зв’язаний список

Задача №1 - Реверс списку (Reverse list)

Реалізувати метод реверсу списку: Node* reverse(Node *head);
Умови задачі:

- використовувати цілочисельні значення в списку;
- реалізувати метод реверсу;
- реалізувати допоміжний метод виведення вхідного і обернутого списків;

Задача №2 - Порівняння списків

bool compare(Node *h1, Node *h2);
Умови задачі:

- використовувати цілочисельні значення в списку;
- реалізувати функцію, яка ітеративно проходиться по обох списках і порівнює дані в кожному вузлі;
- якщо виявлено невідповідність даних або якщо довжина списків різна (один список закінчується раніше іншого), функція повертає ***false***.

Задача №3 – Додавання великих чисел

Node* add(Node *n1, Node *n2);
Умови задачі:

- використовувати цифри від 0 до 9 для значень у списку;
- реалізувати функцію, яка обчислює суму двох чисел, які збережено в списку; молодший розряд числа записано в голові списка (напр. 379 $\Longrightarrow$ 9→7→3);
- функція повертає новий список, передані в функцію списки не модифікуються.

Бінарні дерева

Задача №4 - Віддзеркалення дерева

TreeNode *create_mirror_flip(TreeNode *root);
Умови задачі:

- використовувати цілі числа для значень у вузлах дерева
- реалізувати функцію, що проходить по всіх вузлах дерева і міняє місцями праву і ліву вітки дерева
- функція повертає нове дерево, передане в функцію дерево не модифікується

Задача №5 - Записати кожному батьківському вузлу суму підвузлів

void tree_sum(TreeNode *root);
Умови задачі:

- використовувати цілочисельні значення у вузлах дерева;
- реалізувати функцію, яка ітеративно проходить по бінарному дереві і записує у батьківський вузол суму значень підвузлів
- вузол-листок не змінює значення
- значення змінюються від листків до кореня дерева

Завдання №5 – Self Practice Work Algotester task 1 (Algotester Lab 7-8 variant 3)

Задача

Limits: 1 sec., 256 MiB

Ваше завдання - власноруч реалізувати структуру даних "Двійкове дерево пошуку".
Ви отримаєте Q запитів, кожен запит буде починатися зі слова-ідентифікатора, після якого йдуть його параметри.

Вам будуть поступати запити такого типу:

- **Вставка**:
  Ідентифікатор - $insert$
  Ви отримуєте ціле число $value$ - число, яке треба вставити в дерево.
- **Пошук**:
  Ідентифікатор - $contains$
  Ви отримуєте ціле число $value$ - число, наявність якого у дереві необхідно перевірити.
  Якщо $value$ наявне в дереві - ви виводите Yes, у іншому випадку No.
- **Визначення розміру**:
  Ідентифікатор - $size$
  Ви не отримуєте аргументів.
  Ви виводите кількість елементів у дереві.
- **Вивід дерева на екран**
  Ідентифікатор - $print$
  Ви не отримуєте аргументів.
  Ви виводите усі елементи дерева через пробіл.
  Реалізувати використовуючи перегрузку оператора $<<$

Input

Ціле число Q - кількість запитів.
У наступних рядках Q запитів у зазначеному в умові форматі.

Output

Відповіді на запити у зазначеному в умові форматі.

Constraints

$0 \le Q \le 10^3$
$0 \le t_i \le 10^3$

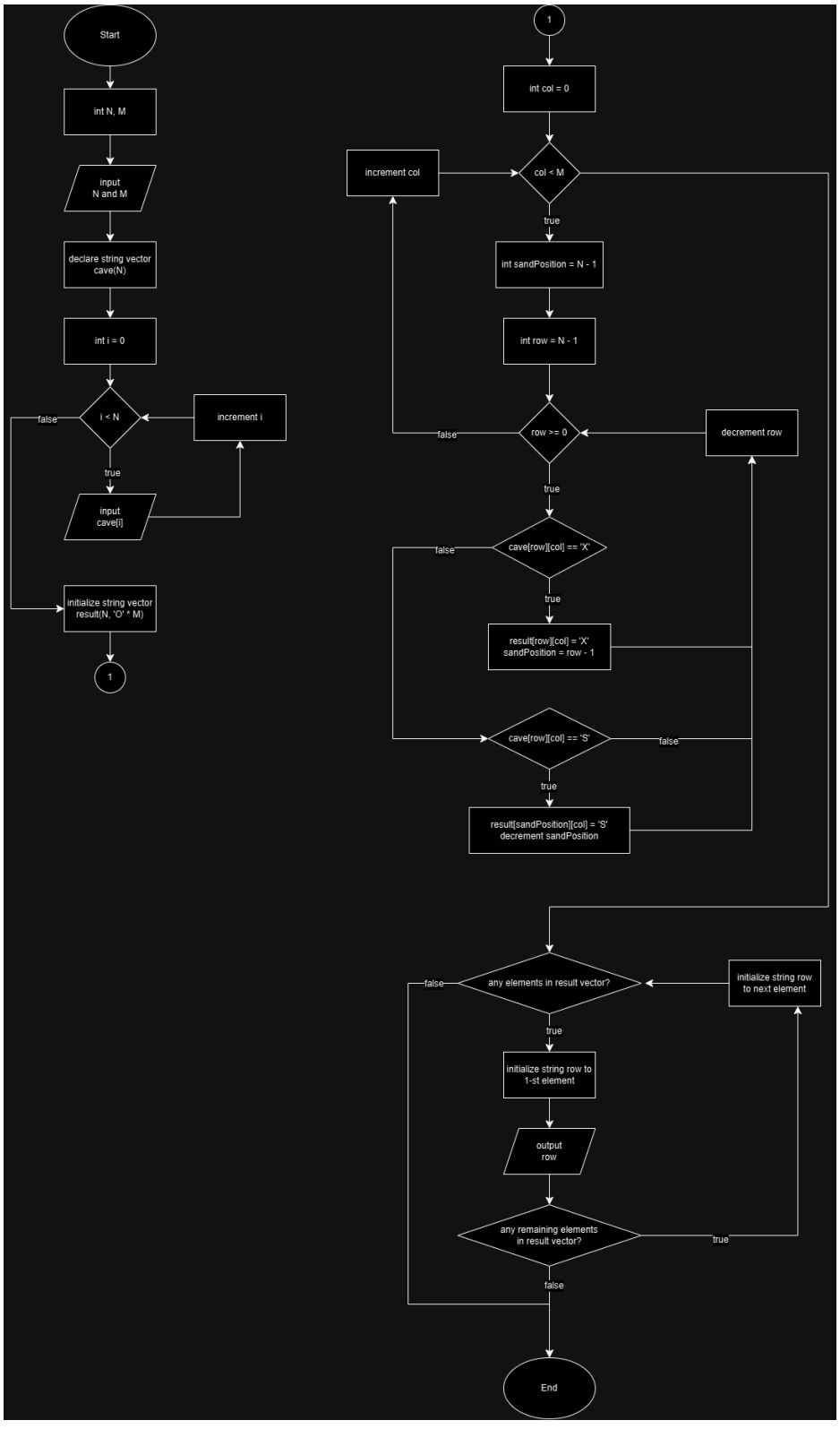
Дизайн та планована оцінка часу виконання завдань:

Завдання №1 – VNS Lab 10 Task 1 variant 23

Планований час виконання: 40 хв

Завдання №2 – Algotester Lab 5 variant 2

Планований час виконання: 30 хв



Завдання №3 – Algotester Lab 7-8 variant 1

Планований час виконання: 50 хв

Завдання №4 – Class Practice Work

Планований час виконання: 90 хв

Завдання №5 – Self Practice Work Algotester task 1 (Algotester Lab 7-8 variant 3)

Планований час виконання: 50 хв

Код програм з посиланням на зовнішні ресурси:

Завдання №1 – VNS Lab 10 Task 1 variant 23

файл vns_lab_10_task_1_variant_23_mykhailo_skichko.cpp

Завдання №2 – Algotester Lab 5 variant 2

файл algotester_lab_5_variant_2_mykhailo_skichko.cpp

Завдання №3 – Algotester Lab 7-8 variant 1

файл algotester_lab_7_8_variant_1_mykhailo_skichko.cpp

Завдання №4 – Class Practice Work

файл practice_work_team_tasks_1_2_3_mykhailo_skichko.cpp
файл practice_work_team_tasks_4_5_mykhailo_skichko.cpp

Завдання №5 – Self Practice Work Algotester task 1 (Algotester Lab 7-8 variant 3)

файл self_practice_work_algotester_task_1_mykhailo_skichko.cpp

Результати виконання завдань, тестування та фактично затрачений час:

Завдання №1 – VNS Lab 10 Task 1 variant 23

фактично затрачений час: 70 хв

```
List created.
Element "one" added.
Element "two" added.
Element "zero" added at the beginning.
Element "four" added at the beginning.
Element "five" added at position 2.

After Insertions:
Forward List: four five zero one two

First element deleted.
Last element deleted.
Element "five" deleted.

After Deletions:
Forward List: zero one

List saved to file "DLL.txt".
The list is cleared.
The list is empty.

The list is cleared.
Element "zero" added.
Element "one" added.
List restored from file "DLL.txt".
Forward List: zero one

The list is cleared.
```

Завдання №2 – Algotester Lab 5 variant 2

фактично затрачений час: 20 хв

Created	Compiler	Result	Time (sec.)	Memory (MiB)	Actions
5 hours ago	C++ 23	Accepted	0.026	1.992	View

Завдання №3 – Algotester Lab 7-8 variant 1

фактично затрачений час: 40 хв

Created	Compiler	Result	Time (sec.)	Memory (MiB)	Actions
5 hours ago	C++ 23	Accepted	0.008	1.309	View

Завдання №4 – Class Practice Work

фактично затрачений час: 120 хв

```
List 1: 2 -> 1 -> 4 -> 3 -> 5 -> 6 -> NULL
List 2: 1 -> 2 -> 4 -> 3 -> 5 -> 6 -> NULL
Are List 1 and List 2 equal? -- Not Equal
Reversed List 1: 6 -> 5 -> 3 -> 4 -> 1 -> 2 -> NULL
Number 1: 9 -> 7 -> 3 -> NULL
Number 2: 6 -> 8 -> NULL
Sum: 5 -> 6 -> 4 -> NULL
```

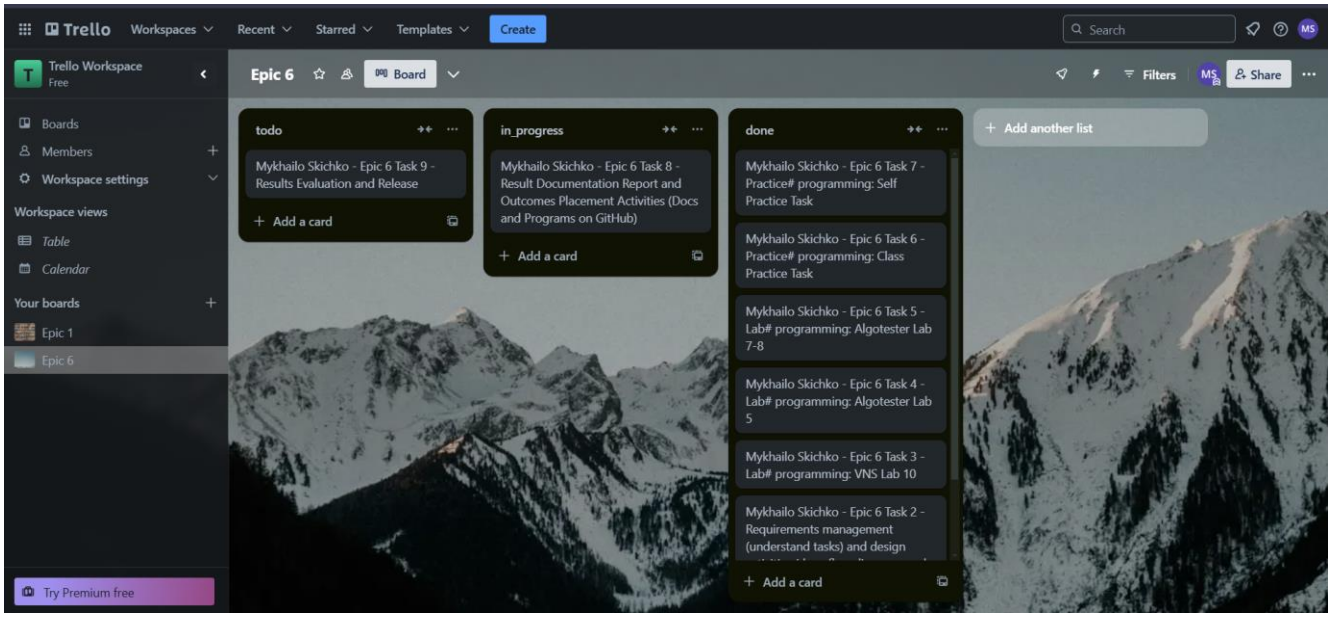
```
Initial tree:
5
3 7
2 4 6 8
1 9


Mirrored tree:
5
7 3
8 6 4 2
9 1


After tree summation:
20
5 15
1 4 6 9
1 9
```

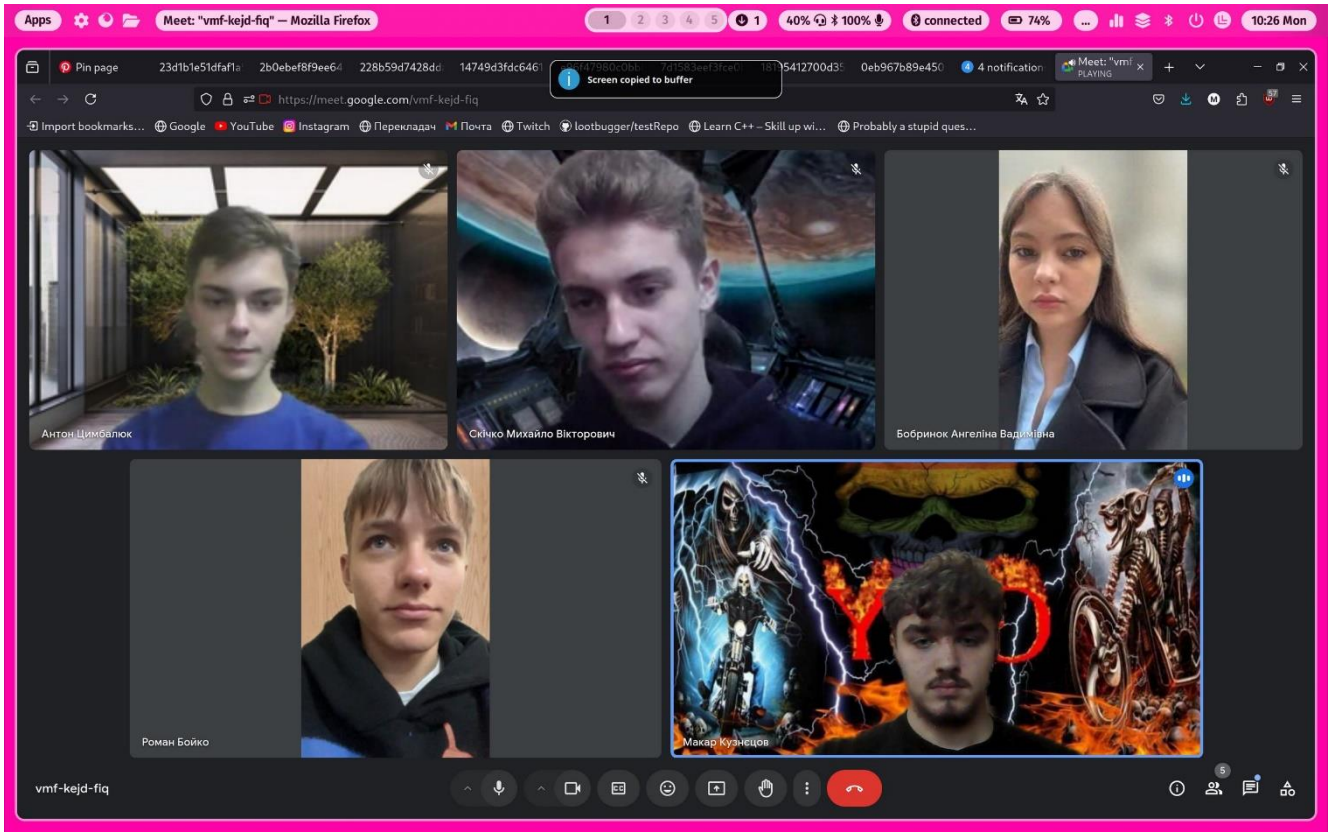
Завдання №5 – Self Practice Work Algotester task 1 (Algotester Lab 7-8 variant 3)

фактично затрачений час: 50 хв

Created	Compiler	Result	Time (sec.)	Memory (MiB)	Actions
4 hours ago	C++ 23	Accepted	0.008	1.348	View

Кооперація з командою:





Висновки:

У результаті виконання роботи я реалізував зв'язні списки та бінарні дерева. Це дало змогу глибше зрозуміти принципи роботи з динамічними структурами та алгоритмами обробки даних, зокрема для вставки, пошуку та обходу елементів. Здобуті навички допоможуть у подальшій роботі з алгоритмами та у програмуванні.

Посилання на pull request: