Міністерство освіти і науки України
Львівський національний університет Івана Франка
Факультет електроніки та комп'ютерних технологій

Звіт
про виконання лабораторної роботи №3
"Одно-направлений пошук на графі. Дослідження особливостей використання хвильового алгоритму для знаходження найкоротшого шляху у одиничних графах методом одно-направленого пошуку. "
з курсу "Системи штучного інтелекту"

Виконала
Пильник С. А.
група ФЕІ-42
Перевірив
ас. Іжик О. Б.

Львів 2025

Мета роботи: вивчення принципів функціонування та особливостей застосування хвильового алгоритму (Breadth-First Search, BFS) для задачі одно-направленого пошуку найкоротшого шляху в одиничних графах. Дослідження впливу різних операторів переходу та розмірності графа на ефективність пошуку (час виконання, кількість розкритих вершин) та довжину знайденого шляху. Реалізація програмного засобу мовою Python з використанням tkinter для візуалізації процесу пошуку в матриці-лабіринті.

Теоретичні відомості:

В основі задачі пошуку шляху лежить математична структура, відома як **граф**. Формально, граф **G** визначається як пара **(V, E)**, де **V** є скінченною множиною **вершин** (або вузлів), а **E** — множиною **ребер** (або дуг), що з'єднують пари вершин. У контексті даної лабораторної роботи, множина вершин **V** представлена сукупністю *прохідних* комірок (позначених 0) у заданій матриці-лабіринті. Множина ребер **E** визначається неявно, на основі правил можливості переходу між двома вершинами-комірками. Кількість вершин (потужність |**V**|) називається **порядком** графу, тоді як кількість ребер (потужність |**E**|) — його **розміром**. Оскільки вартість переходу (вага ребра) між будь-якими двома суміжними вершинами приймається рівною одиниці, такий граф класифікується як **одиничний** (або незважений). Представлення задачі у вигляді матриці-лабіринту є формою подання простору станів, де непрохідні ділянки (-1) визначають обмеження топології графу.

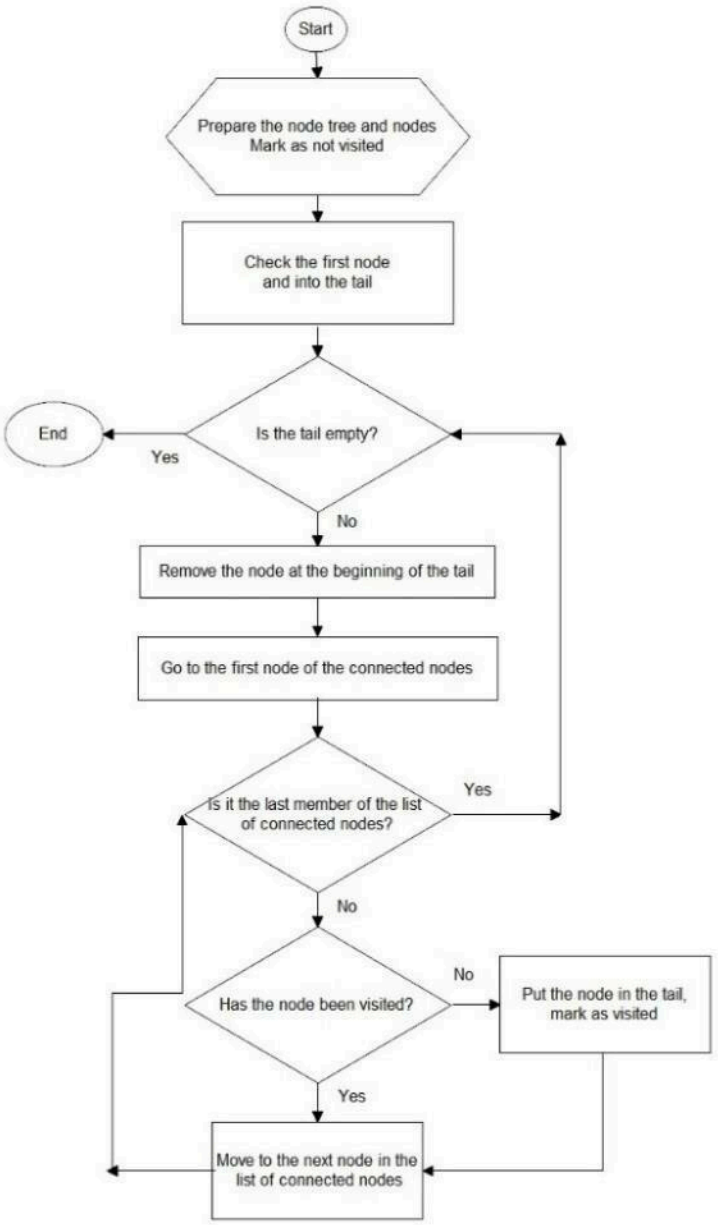
Структура множини ребер **E** та, відповідно, зв'язність графу визначаються **оператором переходу**. Цей оператор є формальним правилом, яке для довільної вершини (r, c) ідентифікує множину суміжних їй вершин. У роботі досліджується вплив трьох різних операторів: **Cardinal** (дозволяє переходи до 4 сусідів: вверх, вниз, вправо, вліво, що відповідає метриці Манхеттенської відстані), **Diagonal** (дозволяє переходи до 4 діагональних сусідів) та **Combined** (дозволяє переходи до 8 суміжних комірок). Вибір оператора переходу є критичним, оскільки він безпосередньо модифікує множину **E**, що впливає як на загальний розмір графу, так і на топологію та метричні характеристики (довжину) найкоротших шляхів у ньому.

Хвильовий алгоритм, що є прямою реалізацією **пошуку в ширину (Breadth-First Search, BFS)**, є фундаментальним методом обходу графу, який гарантовано знаходить шлях з мінімальною кількістю ребер

(найкоротший шлях) між початковою **V_start** та цільовою **V_goal** вершинами в одиничному графі. Алгоритм виконує систематичне дослідження вершин графу пошарово, поширюючись від **V_start** концентричними "хвилями".

Для реалізації одно-направленого хвильового пошуку використовується структура даних **черга (Queue)**, що працює за принципом FIFO (First-In, First-Out). Процес ініціалізується додаванням **V_start** до черги та її маркуванням як відвіданої. Алгоритм виконується ітеративно, доки черга не стане порожньою. На кожній ітерації (циклі розкриття) з початку черги вилучається вершина "current". Якщо "current" збігається з **V_goal**, пошук успішно завершується. В іншому випадку, алгоритм ідентифікує всіх суміжних до "current" сусідів (відповідно до обраного оператора переходу), які ще не були відвідані. Кожен такий сусід маркується як відвіданий, "current" записується як його "батьківська" вершина (для подальшого відновлення шляху), і сусід додається в кінець черги.

Гарантія знаходження найкоротшого шляху базується на властивостях одиничного графу та черги FIFO: вершини обробляються строго в порядку зростання їх відстані (кількості ребер) від **V_start**. Таким чином, будь-яка вершина, включно з **V_goal**, досягається вперше саме найкоротшим шляхом. Після знаходження **V_goal**, фактичний шлях відновлюється шляхом **зворотного трасування** (backtracking) від цільової вершини до початкової через збережені посилання на батьківські вершини.\



блок схема 1. Пошуку в Ширину (Breadth-First Search, BFS)

Хід роботи:

У процесі виконання лабораторної роботи було створено програму мовою Python із використанням вбудованої бібліотеки tkinter для реалізації графічного інтерфейсу користувача (GUI). Програма реалізує генерацію випадкового лабіринту (одиничного графу) у вигляді матриці та забезпечує виконання одно-направленого хвильового пошуку (BFS) з покроковою візуалізацією.

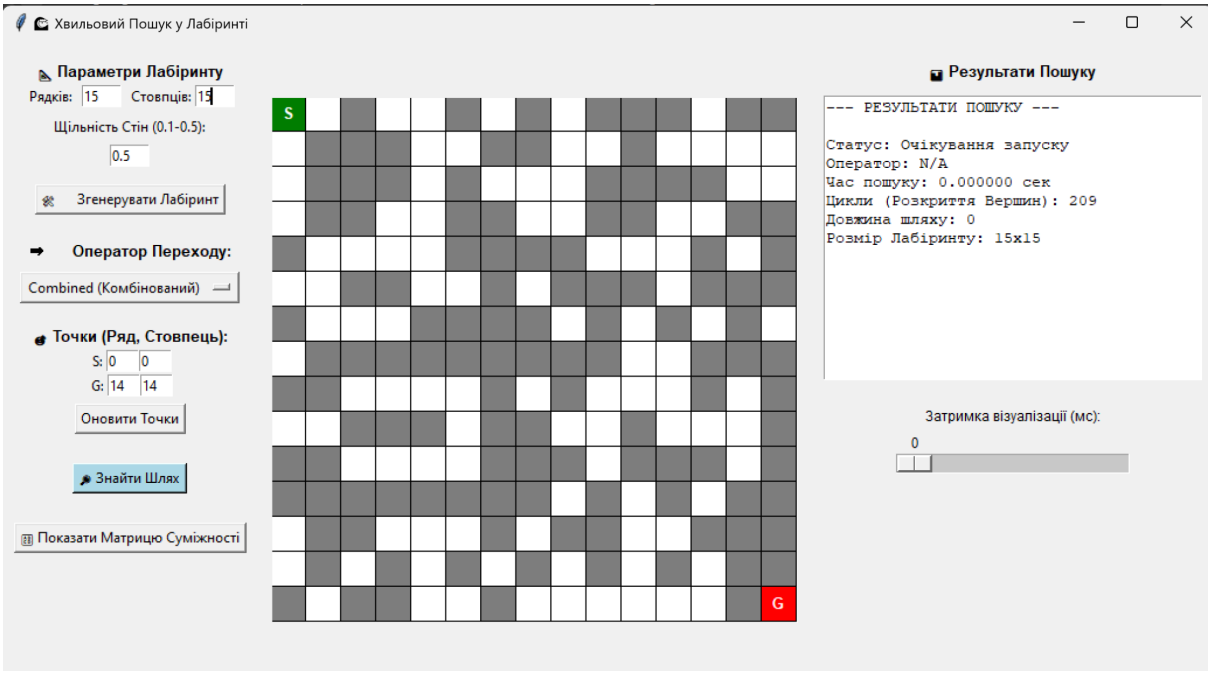


Рис. 1. Загальний вигляд інтерфейсу програми.

Основні функціональні можливості програми реалізовані через панель керування, яка логічно розділена на декілька блоків:

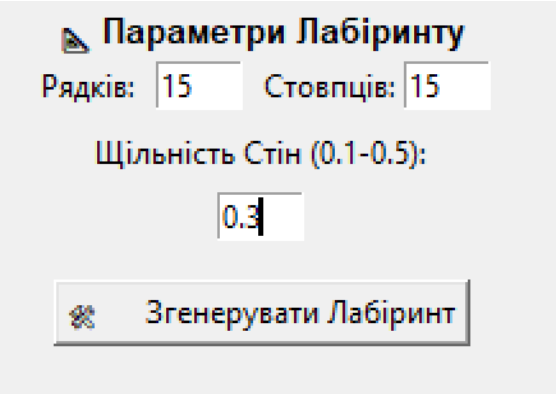


Рис. 2. Панель "Параметри Лабіринту"

1. Панель "Параметри Лабіринту"

- Рядків / Стовпців: Поля вводу для задання розмірності матриці-лабіринту (наприклад, 15x15).
- Щільність Стін (0.1-0.5): Поле вводу, що визначає ймовірність генерації перешкоди (-1) у кожній комірці.
- Згенерувати Лабіринт: Кнопка, що очищує поточне поле та створює новий випадковий лабіринт згідно із заданими параметрами.

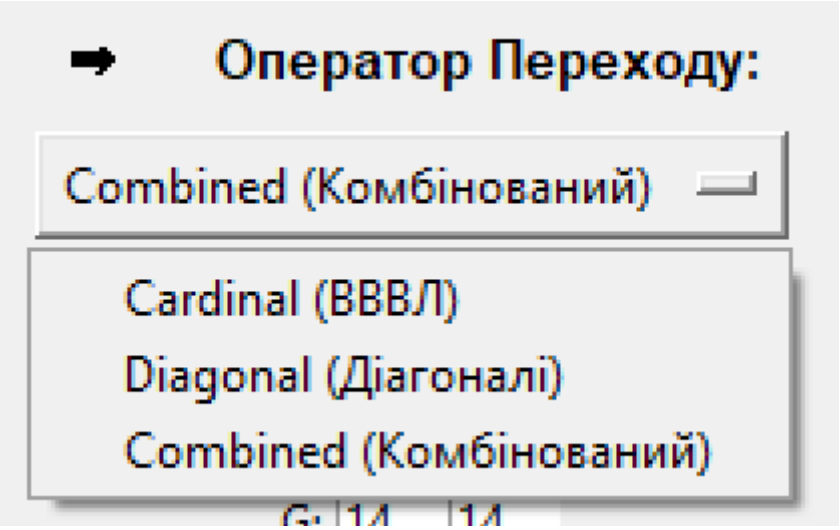


Рис. 3. Панель "Оператор Переходу"

2. Панель "Оператор Переходу"

- Випадаючий список: Дозволяє користувачу обрати один із трьох реалізованих операторів переходу:
    - Cardinal (ВВВЛ): Дозволяє рух лише до 4 сусідів (вверх, вниз, вліво, вправо).
    - Diagonal (Діагоналі): Дозволяє рух лише до 4 діагональних сусідів.
    - Combined (Комбінований): Дозволяє рух до 8 сусідів (комбінація двох попередніх).

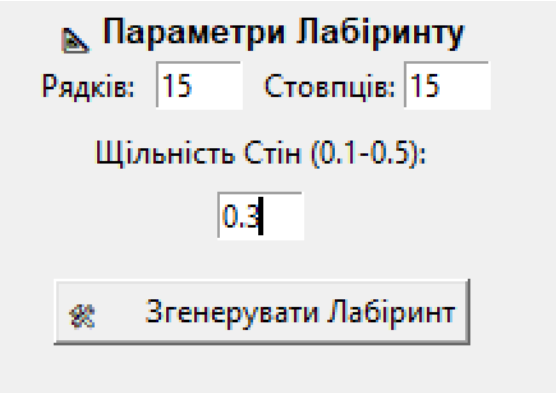


Рис. 4. Панель "Точки (Ряд, Стовпець)"

3. Панель "Точки (Ряд, Стовпець)"

- S (Старт): Два поля вводу для координат початкової вершини.
- G (Ціль): Два поля вводу для координат цільової вершини.

- Оновити Точки: Кнопка для застосування нових координат.

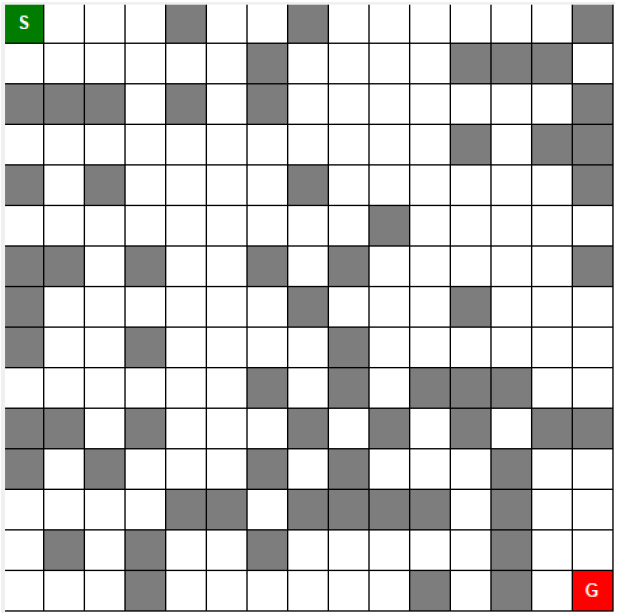


Рис. 5. Поле Візуалізації (Canvas)

4. Поле Візуалізації (Canvas)

Центральний елемент інтерфейсу, що відображає матрицю-лабіринт.

- Білі комірки: Прохідні ділянки (0).
- Сірі комірки: Стіни (-1).
- Зелена ('S'): Початкова вершина.
- Червона ('G'): Цільова вершина.
- Блакитні: Відвідані вершини (поширення "хвилі").
- Сині: Фінальний найкоротший шлях.
- Інтерактивність: Користувач може кліком лівої кнопки миші змінювати статус комірки (стіна ↔ прохід) для ручного редагування лабіринту.

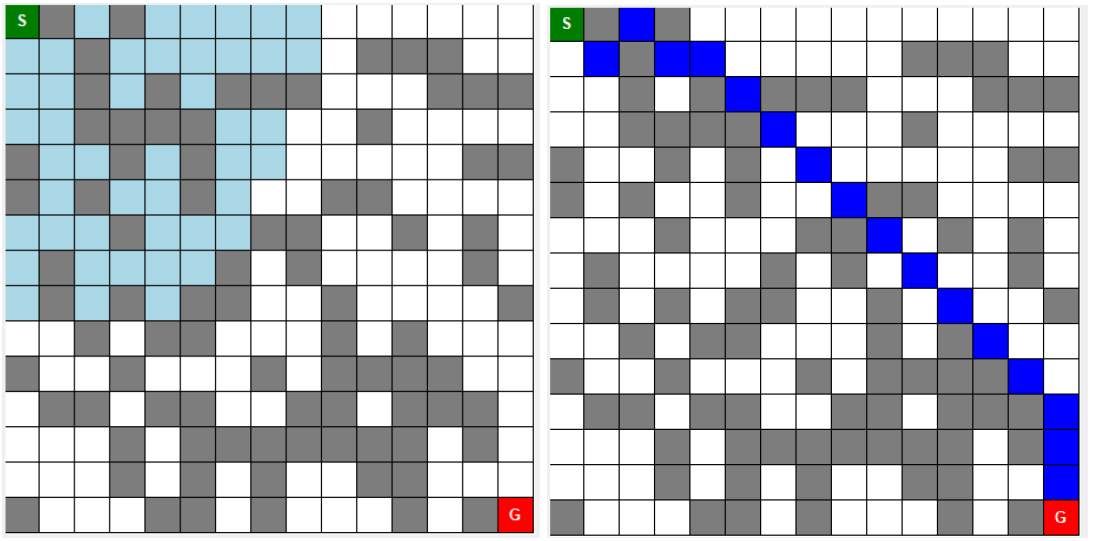


Рис. 6. Процес покрокової візуалізації пошуку та знайдений шлях.

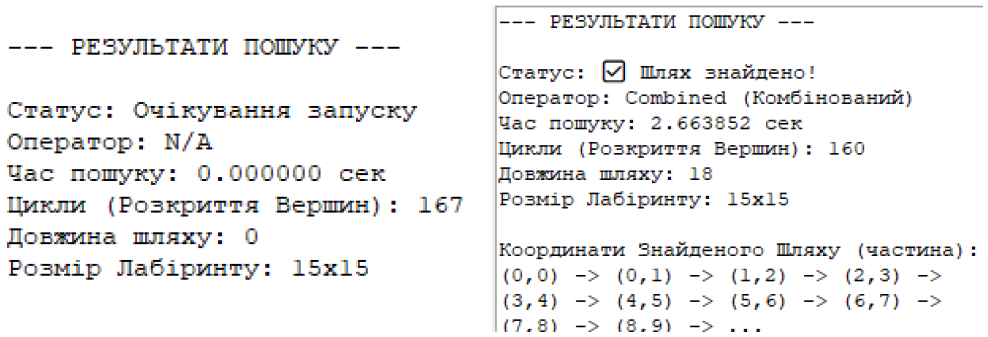


Рис. 7. Панель "Результати Пошуку"

5. Панель "Результати Пошуку"

- Текстове вікно: Окреме вікно, куди виводяться результати виконання алгоритму:
    - Статус: (Шлях знайдено / Шлях не знайдено).
    - Оператор: Використаний оператор переходу.
    - Час пошуку: Загальний час, витрачений на виконання алгоритму (в секундах).
    - Цикли (Розкриття Вершин): Кількість вершин, вилучених з черги (показник ефективності).
    - Довжина шляху: Кількість вершин у знайденому шляху.
    - Координати Знайденого Шляху: Послідовність координат від V_start до V_goal.

- Затримка візуалізації (мс): Повзунок, що дозволяє регулювати швидкість покрокової візуалізації для спостереження за процесом поширення хвилі.

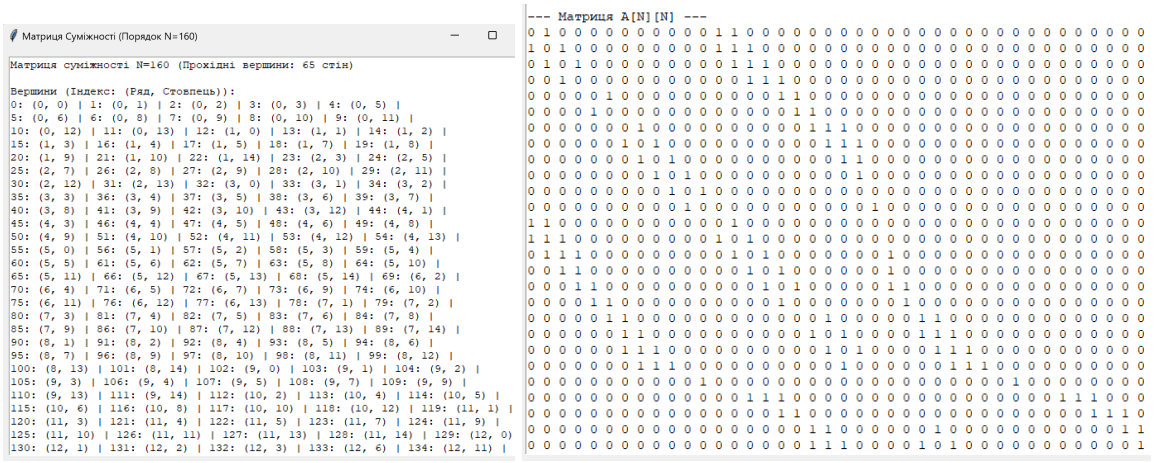


Рис. 8. Вікно Матриця Суміжності

6. Матриця Суміжності

- Показати Матрицю Суміжності: Кнопка, що відкриває нове вікно. У ньому граф представляється у вигляді формальної NxN матриці суміжності, де N — загальна кількість *прохідних* вершин у лабіринті. Це вікно також показує мапу відповідності індексів матриці (0...N-1) до координат лабіринту (r, c).

Дослідження

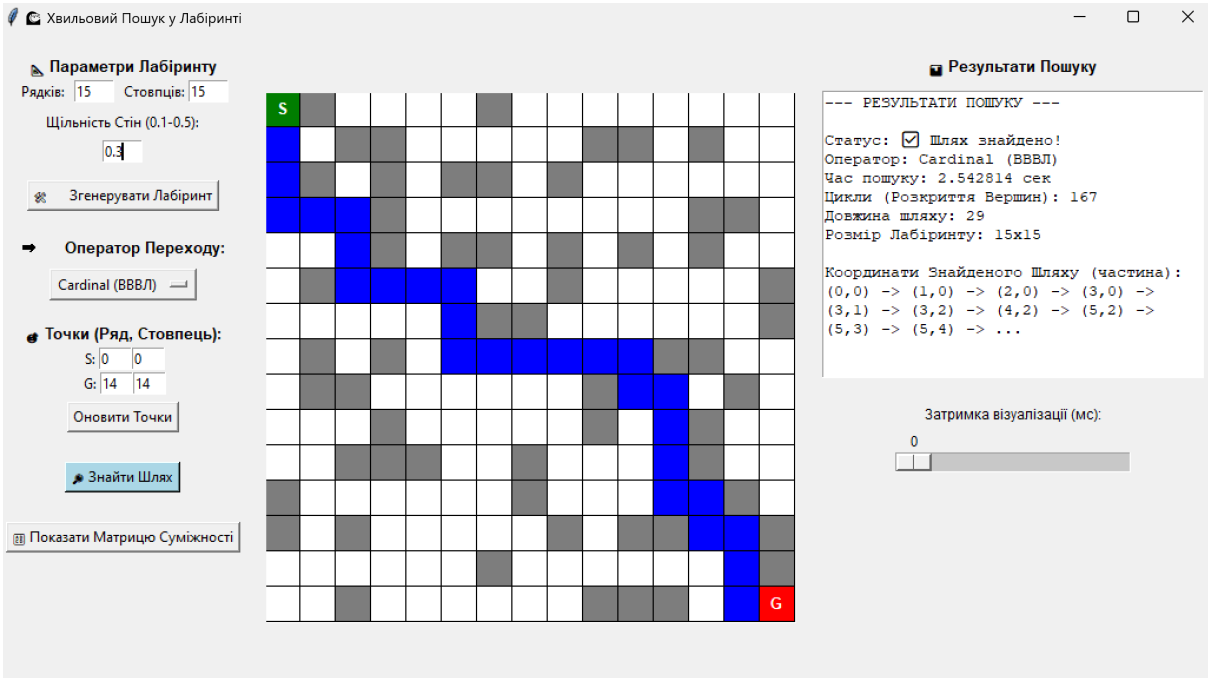
1. Дослідження Впливу Операторів Переходу

На першому етапі було зафіксовано параметри лабіринту (розмір 15x15, щільність стін 0.3) та обрано фіксовані початкову (0, 0) і цільову (14, 14) вершини. Пошук проводився з використанням трьох різних операторів переходу.

Оператор Переходу	Статус	Довжина Шляху (вузлів)	Цикли (Вершини)	Час, сек

Cardinal (ВВВЛ)	Шлях знайдено!	31	168	0.041
Diagonal (Діагоналі)	Шлях не знайдено.	0	95	0.025
Combined (Комбінований)	Шлях знайдено!	25	189	0.053

Таблиця 1. Результати пошуку при різних операторах переходу (Граф 15x15).



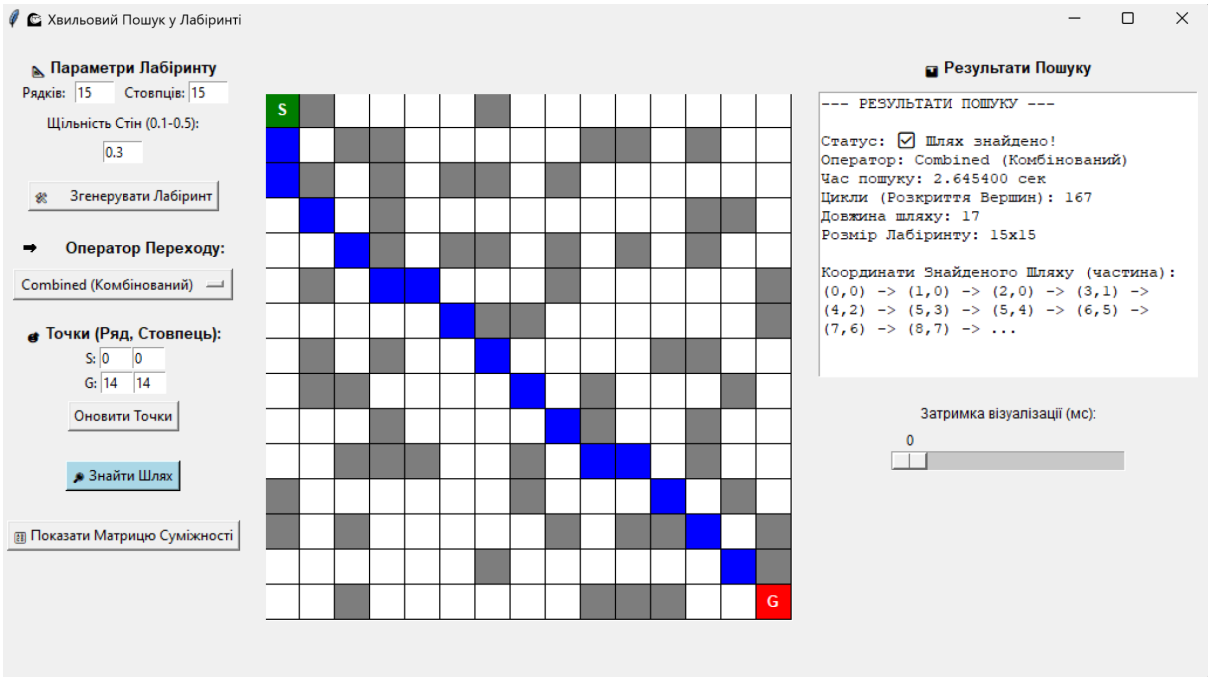


Рис. 9-10. Порівняння шляхів: Cardinal (довжина 29) та Combined (довжина 17).

Аналіз та Переваги/Недоліки:

1. **Cardinal (ВВВЛ):**
    - **Результат:** Завжди знаходить шлях, якщо він існує на 4-зв'язній сітці. Знайдений шлях (31) є найкоротшим з точки зору **Манхеттенської відстані**.
    - **Переваги:** Висока швидкість (168 циклів). Алгоритм перевіряє лише 4 сусідів на кожному кроці.
    - **Недоліки:** Знайдений шлях **не є** істинно найкоротшим у 8-зв'язному просторі.
2. **Diagonal (Діагоналі):**
    - **Результат:** Не зміг знайти шлях.
    - **Переваги:** Дуже швидкий (95 циклів), оскільки досліджує менше вершин.
    - **Недоліки:** Цей оператор є надто обмеженим. Він може знайти шлях лише у випадку, якщо від старту до цілі можна дістатися *виключно* діагональними кроками, що в лабіринті майже неможливо.
3. **Combined (Комбінований):**

- **Результат:** Знайшов **найкоротший** можливий шлях (25 вузлів), коректно використовуючи діагональні "зрізи".
- **Переваги: Гарантує знаходження істинно найкоротшого шляху** (за кількістю кроків) у 8-зв'язному просторі.
- **Недоліки: Є найбільш обчислювально дорогим** (189 циклів, час 0.053с). Оскільки на кожному кроці алгоритм змушений перевіряти 8 сусідів замість 4, загальна кількість операцій та розкритих вершин зростає.

2. Дослідження Впливу Розміру Графу

На другому етапі було зафіксовано оператор переходу ("Combined" як найбільш точний) та щільність стін (0.25). Досліджувався вплив зміни розмірності матриці-лабіринту (і, відповідно, **порядку графу**) на ефективність пошуку, дотримуючись діапазону порядку, вказаного у завданні (10-20 вершин).

Розмір Лабіринту	Загальна к-сть Вершин (Порядок)	Статус	Довжина Шляху (вузлів)	Цикли (Вершини)	Час, сек
3 x 4	12	Шлях знайдено!	5	11	0.001
4 x 4	16	Шлях знайдено!	6	15	0.002
4 x 5	20	Шлях знайдено!	7	19	0.004
5 x 5	25	Шлях знайдено!	9	24	0.005

Таблиця 2. Вплив розміру графу на ефективність (Оператор "Combined").

Аналіз:

Результати в Таблиці 2 демонструють пряму залежність ефективності пошуку від розміру графу, навіть у межах малих порядків. Хвильовий

алгоритм (BFS) у найгіршому випадку повинен відвідати всі доступні вершини V та ребра E (складність O(V+E)). У матриці-лабіринті N xM кількість вершин (порядок) V = N xM.

- При збільшенні порядку з 12 (3 x4) до 20 (4 x5) — тобто зростанні V приблизно в 1.7 раза — кількість циклів (розкритих вершин) зросла з 11 до 19 (приблизно в 1.7 раза).
- При переході до графу порядку 25 (5 x5) кількість циклів зросла до 24.
- Час виконання також показує чітку поліноміальну (близьку до лінійної відносно V) залежність, хоча на малих графах він залишається дуже низьким.
- Це підтверджує, що збільшення розміру графу (простору пошуку) прямо та передбачувано збільшує обчислювальні витрати на знаходження шляху, оскільки алгоритму необхідно дослідити більше вершин для поширення "хвилі".

Посилання на GitHub репозиторій з кодом проєкту: