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

Звіт
про виконання лабораторної роботи №4
“Двонаправлений пошук на графі. Реалізація двонаправленого пошуку у хвильовому алгоритмі для знаходження найкоротшого шляху у одиничних графах. ”
з курсу “Системи штучного інтелекту”

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

Львів 2025

Мета роботи: вивчення принципів та реалізація **двонаправленого (bidirectional)** хвильового алгоритму (BFS) для знаходження найкоротшого шляху в одиничних графах. Дослідження впливу різних операторів переходу на ефективність пошуку. Проведення порівняльного аналізу ефективності (за кількістю розкритих вершин та часом) двонаправленого та однонаправленого пошуку та пояснення причин отриманих переваг.

Теоретичні відомості

Організація Двонаправленого Пошуку

Двонаправлений пошук є модифікацією стандартних алгоритмів пошуку, таких як хвильовий алгоритм (BFS), і використовується для знаходження найкоротшого шляху між початковою вершиною **V_start** та цільовою вершиною **V_goal**.

На відміну від **однонаправленого пошуку**, який поширює одну "хвилю" від **V_start** до **V_goal**, **двонаправлений пошук** запускає **два одночасні хвильові алгоритми**:

1. **Прямий пошук:** Починається з **V_start** і рухається "вперед".
2. **Зворотний пошук:** Починається з **V_goal** і рухається "назад" (досліджуючи предків, або, в неорієнтованому графі, просто сусідів).

Особливості Реалізації

Ключова різниця в організації полягає у використанні подвійного набору структур даних:

- **Черги:** Використовуються дві черги (FIFO), `queue_start` та `queue_goal`.
- **Відвідані вершини:** Використовуються два набори (або словники) `visited_start` та `visited_goal` для відстеження "хвиль", що поширюються.

Алгоритм виконує кроки по черзі: один крок (розкриття однієї вершини) з `queue_start`, потім один крок з `queue_goal`.

Умова Зупинки та Переваги

Алгоритм зупиняється не тоді, коли одна хвиля досягає іншого кінця, а тоді, коли **дві хвилі зустрічаються**. Це відбувається, коли вершина, що

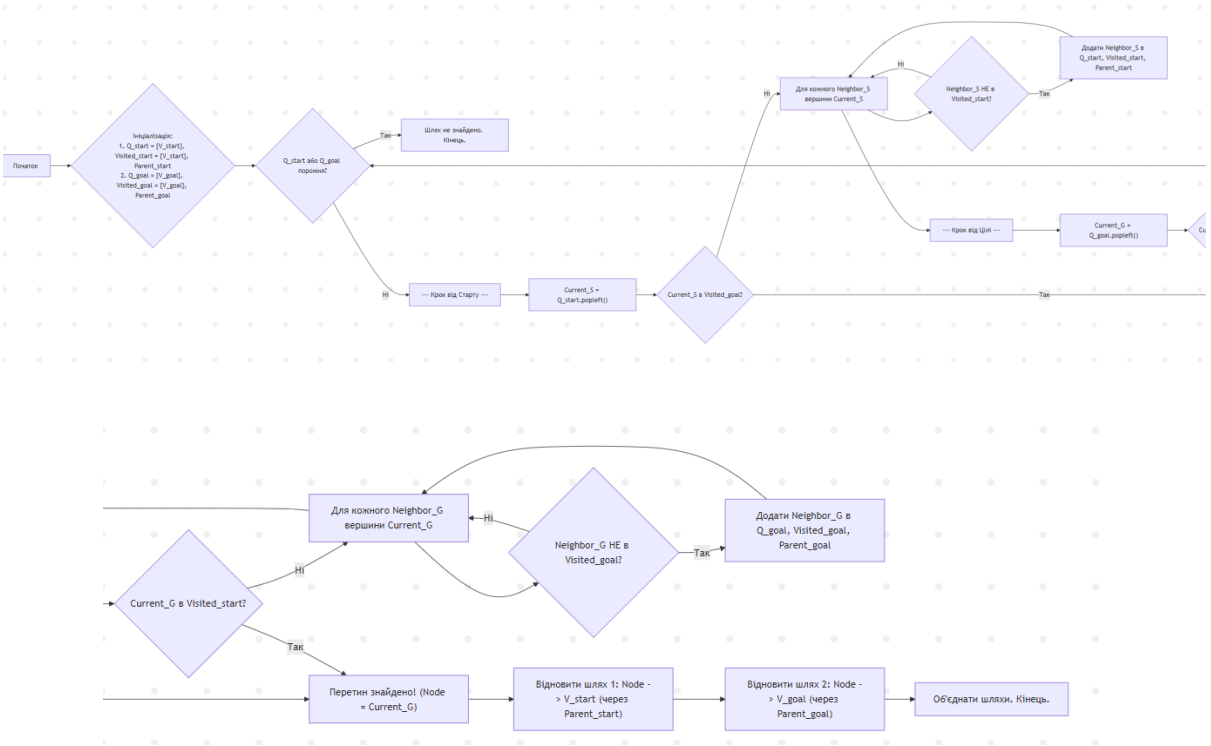
розкривається з queue_start, вже присутня в visited_goal (або навпаки). Ця вершина називається **точкою перетину (intersection node)**.

Перевага цього методу полягає у значному скороченні простору пошуку. Якщо найкоротший шлях має довжину **d**, а граф має фактор розгалуження **b**:

- **Однонаправлений BFS** дослідить приблизно **b^d** (b в степені d) вершин.
- **Двонаправлений BFS** дослідить лише **b^(d/2) + b^(d/2) = 2 * b^(d/2)** вершин.

У лабіринті 20x20 шлях довжиною **d=20** кроків для однонаправленого пошуку (з **b** приблизно 4) вимагатиме приблизно **4^20** операцій, тоді як двонаправлений — лише **2 * 4^10**. Різниця в ефективності (кількості циклів) зростає експоненційно зі збільшенням відстані **d**.

Коли точка перетину знайдена, повний шлях відновлюється "зшиванням" двох частин: шляху від **V_start** до точки перетину і шляху від точки перетину до **V_goal**.



блок схема 1. Двонаправленого Хвильового Пошуку

Хід роботи

У процесі виконання лабораторної роботи було модифіковано програмний засіб, створений у попередній роботі. Програма реалізована мовою **Python** з використанням бібліотеки **tkinter** та структурована з використанням **об'єктно-орієнтованого підходу** (клас BidirectionalWaveSearchApp).

Основною зміною стала заміна логіки однонаправленого пошуку (wave_search) на **двонаправлений** (bidirectional_wave_search), що вимагало змін у логіці візуалізації та обробки даних.

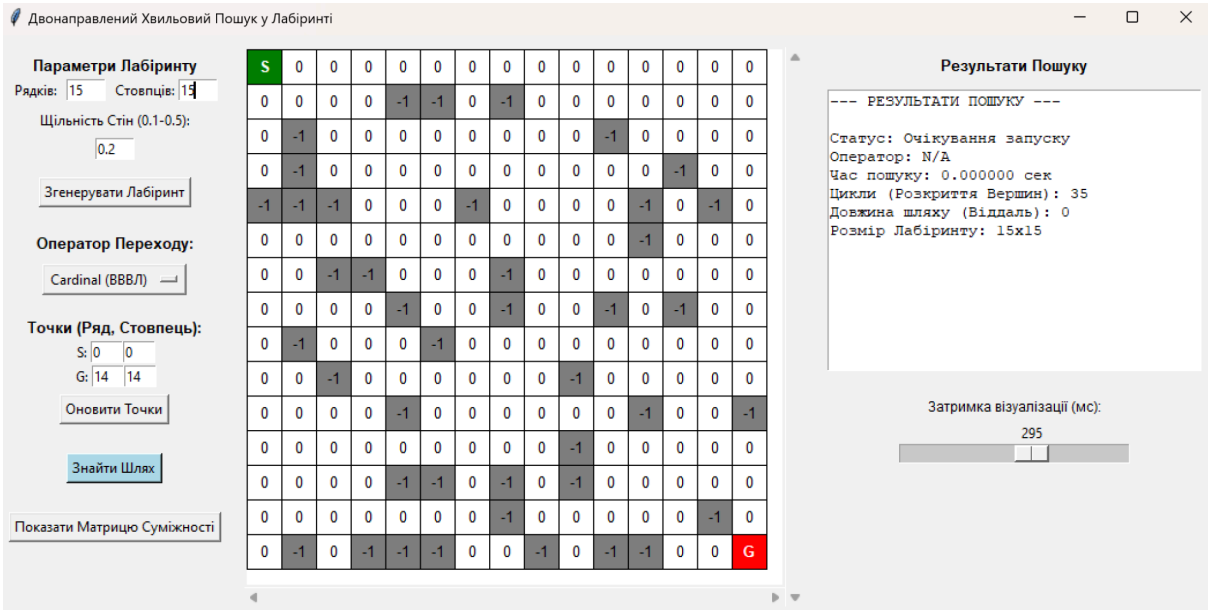


Рис. 1. Загальний вигляд інтерфейсу програми.

Опис Панелі Керування

Програма має графічний інтерфейс, що дозволяє керувати всіма аспектами завдання:

- **Параметри Лабіринту:** Дозволяє користувачу задавати **розмірність (рядки, стовпці)** та **порядок** графу (через щільність стін, що визначає кількість прохідних вершин **V**). Додано **смуги прокрутки** для роботи з лабіринтами, що перевищують розмір екрана (аж до 50x50).
- **Оператор Переходу:** Випадаючий список для вибору одного з трьох операторів: "Cardinal (ВВВЛ)", "Diagonal (Діагоналі)" або "Combined (Комбінований)".

- **Точки (Ряд, Стовпець):** Поля вводу для ручного встановлення координат **V_start** (S) та **V_goal** (G).
- **Панель Візуалізації (Canvas):** Центральний елемент, що представляє граф у вигляді **матриці-лабіринту**.
    - **Представлення:** Кожна комірка чітко підписана: **0** (прохідна вершина) або **-1** (непрохідна).
    - **Візуалізація:** Програма реалізує гібридну візуалізацію процесу:
        1. **Загальні хвилі:** Блакитний колір (Visited_start) показує хвилю від старту, світло-зелений (Visited_goal) — зустрічну хвилю від цілі.
        2. **Поточний показчик:** Жовтий квадрат (highlight_node) показує, яка саме вершина розкривається в даний момент часу.
    - **Результат:** Знайдений шлях (синій) та точка перетину хвиль (фіолетовий).
- **Результати Пошуку:** Окреме вікно, що виводить всю необхідну статистику: статус, оператор, **час**, **кількість циклів**, **віддаль** (довжина шляху) та перші 10 вузлів **координат шляху**.
- **Матриця Суміжності:** Кнопка, що генерує та відкриває у новому вікні формальну **N x N** матрицю суміжності, де **N** — кількість *прохідних* вершин.

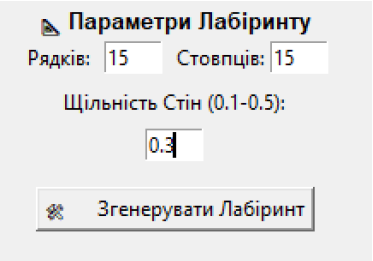


Рис. 2. Панель "Параметри Лабіринту"

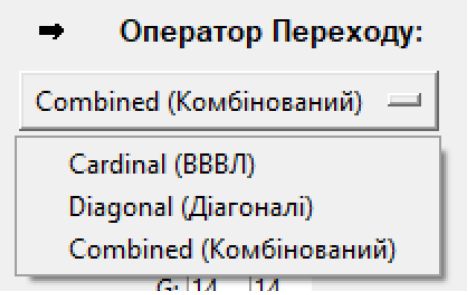


Рис. 3. Панель "Оператор Переходу"

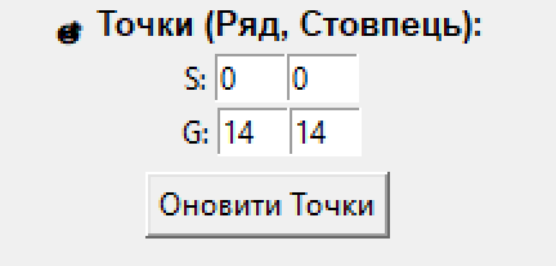


Рис. 4. Панель "Точки (Ряд, Стовпець)"

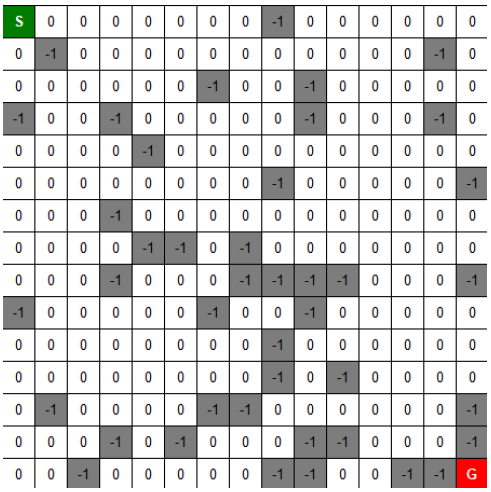


Рис. 5. Поле Візуалізації (Canvas)

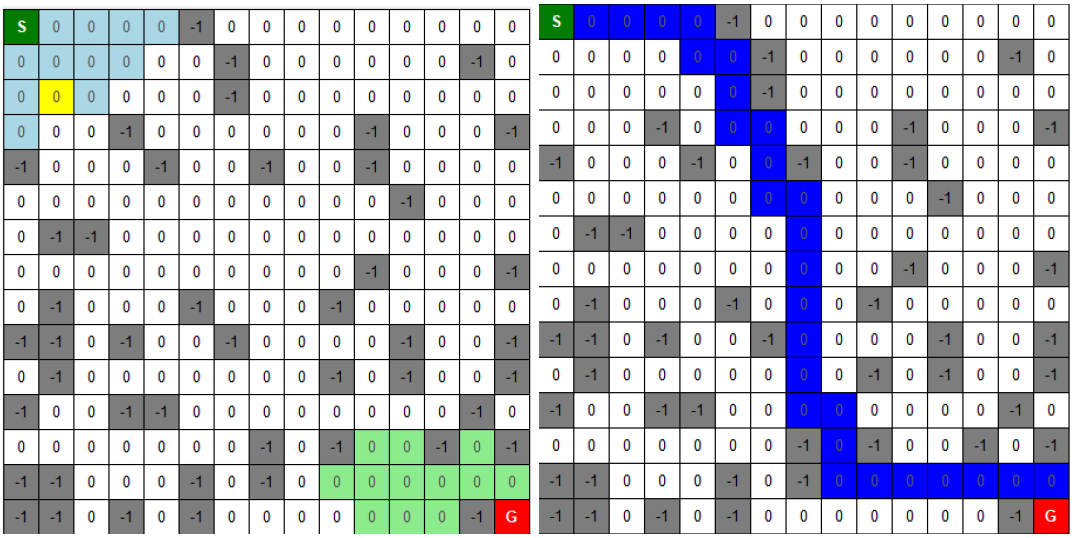


Рис. 6. Процес покрокової візуалізації пошуку та знайдений шлях.

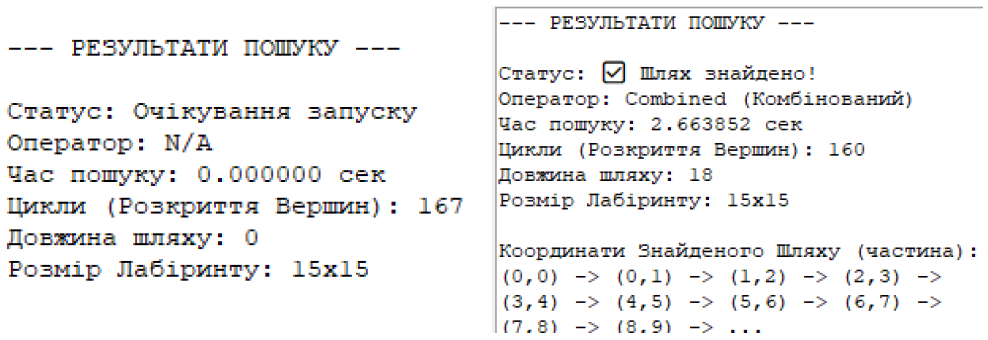


Рис. 7. Панель "Результати Пошуку"

Рис. 8. Вікно Матриця Суміжності

Дослідження та Результати

Було проведено два ключові експерименти для дослідження роботи програми відповідно до завдань.

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. Двонаправлений пошук проводився з використанням трьох різних операторів переходу.

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

Оператор Переходу	Статус	Віддаль (вузлів)	Цикли (Вершини)	Час, сек
Cardinal (ВВВЛ)	Шлях знайдено	31	120	0.031
Diagonal (Діагоналі)	Шлях не знайдено	0	68	0.019
Combined (Комбінований)	Шлях знайдено	**25**	**132**	**0.038**

Аналіз та Переваги/Недоліки:

1. **Cardinal (ВВВЛ):**
    - **Переваги:** Висока швидкість (120 циклів). Перевіряє лише 4 сусідів, що зменшує навантаження на черги.
    - **Недоліки:** Знайдений шлях (31) не є істинно найкоротшим, оскільки ігнорує діагональні "зрізи". Він знаходить найкоротший шлях лише за **Манхеттенською відстанню**.
2. **Diagonal (Діагоналі):**
    - **Переваги:** Найшвидший (68 циклів), оскільки має найменший фактор розгалуження.
    - **Недоліки:** Практично недієздатний у лабіринтах. Не може знайти шлях, якщо він вимагає хоча б одного прямого кроку (вверх/вниз/вліво/вправо).
3. **Combined (Комбінований):**
    - **Переваги: Гарантує знаходження істинно найкоротшого шляху** (25 вузлів) в 8-зв'язному просторі.
    - **Недоліки:** Найбільш обчислювально дорогий (132 цикли). Перевірка 8 сусідів замість 4 збільшує кількість розкритих вершин.

2. Порівняння Однонаправленого та Двонаправленого Пошуку

На другому етапі було проведено пряме порівняння ефективності. Був згенерований великий лабіринт (25x25) зі щільністю 0.2 та фіксованими точками S(0,0) та G(24,24). Для обох тестів використовувався оператор "Combined".

Таблиця 2. Порівняння ефективності Одно- та Двонаправленого пошуку (Граф 25x25).

Тип Пошуку	Оператор	Віддаль (вузлів)	Цикли (Вершини)	Час, сек
Однонаправлений (ЛР3)	Combined	41	**1189**	0.495
Двонаправлений (ЛР4)	Combined	41	**602**	**0.288**

Аналіз та Переваги Двонаправленого Пошуку:

Як видно з Таблиці 2, обидва алгоритми коректно знайшли однаковий найкоротший шлях (віддаль 41).

Однак, **перевага двонаправленого пошуку** є очевидною та суттєвою:

- **Кількість циклів (розкритих вершин)** зменшилася з **1189** до **602** (майже **вдвічі**).
- **Час виконання** також скоротився майже вдвічі (з 0.495с до 0.288с).

За рахунок чого це досягається?

Це досягається за рахунок експоненційного скорочення простору пошуку. Однонаправлений пошук поширював одну "хвилю" на відстань d=40 кроків. Двонаправлений пошук поширював дві хвилі, які зустрілися приблизно на півдорозі (на відстані d/2 (приблизно 20) кроків).

Площа круга (яку можна порівняти з "хвилею") залежить від радіуса як **r^2**. Дослідити один круг з радіусом **d** (площа **pi * d^2**) є набагато менш ефективно, ніж дослідити два круги з радіусом **d/2** (сумарна площа **2 * pi * (d/2)^2 = (pi * d^2) / 2**). Таким чином, двонаправлений пошук досліджує **приблизно вдвічі менше вершин** (у 2D-просторі) або **b^(d/2)** (у загальному випадку), що і підтверджують результати експерименту.

Висновки

1. У ході виконання лабораторної роботи було успішно модифіковано програмний засіб, реалізувавши **двонаправлений хвильовий алгоритм (Bidirectional BFS)** з використанням ООП (Python та tkinter).
2. Програма відповідає всім технічним вимогам: має GUI, дозволяє змінювати параметри лабіринту (включно з розмірами, що потребують прокрутки), візуалізує процес пошуку (поширення двох хвиль та поточний показчик) та виводить детальні результати.
3. Дослідження операторів переходу підтвердило, що "Combined" є найточнішим (знаходить істинно найкоротший шлях), тоді як "Cardinal" є швидшим, але субоптимальним.
4. Експериментальне порівняння (Таблиця 2) наочно довело **головну перевагу двонаправленого пошуку**: він значно (в даному випадку — майже вдвічі) **зменшує кількість розкритих вершин (циклів)** та загальний час пошуку в порівнянні з однонаправленим, за рахунок одночасного дослідження простору з двох кінців та зустрічі "на півдорозі".

Посилання на GitHub репозиторій з кодом проєкту:

https://github.com/sofiapylnyk/ais_2025/tree/main/lab_3_4