

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

Звіт
про виконання лабораторної роботи №1
“Сліпий пошук на графах. Особливості реалізації пошуку в глибину у
графах.”
з курсу “Системи штучного інтелекту”

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

Львів 2025

Мета роботи: вивчення принципів функціонування алгоритмів сліпого пошуку, зокрема пошуку в глибину (Depth-First Search, DFS), дослідження його особливостей при застосуванні до різних типів графів та реалізація програмного засобу з візуалізацією процесу пошуку. Додатковою метою є аналіз впливу порядку обходу суміжних вершин, на пряму пошуку, а також структури графа на результативність алгоритму.

Теоретичні відомості:

Граф у теорії графів визначається як впорядкована пара $G = (V, E)$, де V — множина вершин, а E — множина ребер, що з'єднують пари вершин. Якщо ребра не мають напрямку, граф називається неорієнтованим. У випадку, коли ребра мають напрям, утворюється орієнтований граф або орграф. Особливим випадком графа є дерево — зв'язаний ациклічний граф, у якому між будь-якими двома вершинами існує єдиний шлях.

Алгоритм пошуку в глибину є класичним методом обходу графів, що належить до класу «сліпих» пошуків, тобто таких, які не враховують жодної додаткової інформації про структуру чи властивості цільових вершин. Суть методу полягає в послідовному просуванні від початкової вершини у вибраному напрямку до тих пір, поки не буде досягнуто цільову вершину або не залишиться невідвіданих суміжних вершин. Якщо пошук заходить у «глухий кут», алгоритм здійснює повернення (backtracking) до найближчої вершини, яка має ще не досліджених сусідів.

Формально алгоритм DFS можна описати наступним чином. Нехай задано граф $G = (V, E)$ та вершину $v_0 \in V$. Створюється стек, у який заноситься початкова вершина. Далі виконується цикл:

1. Витягнути вершину зі стеку та позначити її відвіданою.
2. Якщо ця вершина є цільовою, алгоритм завершує роботу.
3. Інакше всі суміжні вершини, які ще не були відвідані, додаються у стек.
4. Повторювати до вичерпання стеку або знаходження мети.

Існує як рекурсивна, так і ітеративна реалізація DFS. Теоретично доведено, що алгоритм DFS гарантує знаходження шляху, якщо він існує, але не гарантує його мінімальності. Важливим чинником є порядок обходу

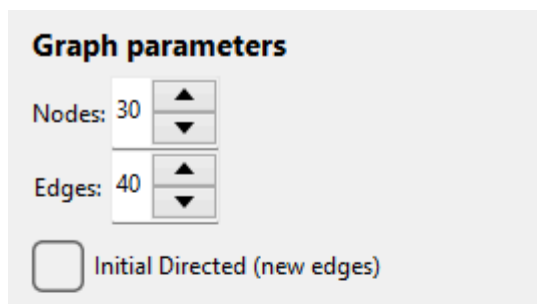
суміжних вершин: від нього залежить, якою буде траєкторія пошуку та чи буде шлях коротким чи довгим.

Хід роботи:

У процесі виконання лабораторної роботи було створено програму мовою Python із використанням бібліотек *networkx*, *matplotlib* та *tkinter*. Програма реалізує генерацію випадкового розгалуженого графа порядку не менше 30 та розміру не менше 30–40 ребер, забезпечує його графічне відображення та інтерактивне керування параметрами пошуку.

Основні функціональні можливості програми включають: генерацію графа із заданою кількістю вершин і ребер, вибір типу графа (звичайний чи орієнтований), вибір початкової та цільової вершини, визначення порядку обходу суміжних вершин (заданого, зростаючого, спадного або випадкового), а також виконання пошуку в глибину з покроковою візуалізацією процесу. Передбачено як покрокове виконання, так і автоматичний режим з регульованою затримкою між кроками. Результати пошуку — знайдений шлях, його довжина та кількість розкритих вершин — відображаються у вікні інтерфейсу.

Інструкція з використання програми передбачає наступні дії. Користувач задає кількість вершин та ребер, тип графа та параметри обходу. Після натискання кнопки «Generate Graph» створюється випадковий граф. Далі можна обрати початкову та цільову вершини. Пошук виконується натисканням кнопки «Step» (для покрокового режиму) або «Run (Auto)» (для автоматичного режиму). Після завершення роботи алгоритму у полі результатів з'являється знайдений шлях та статистичні дані.



Graph parameters

Nodes: 30

Edges: 40

☐ Initial Directed (new edges)

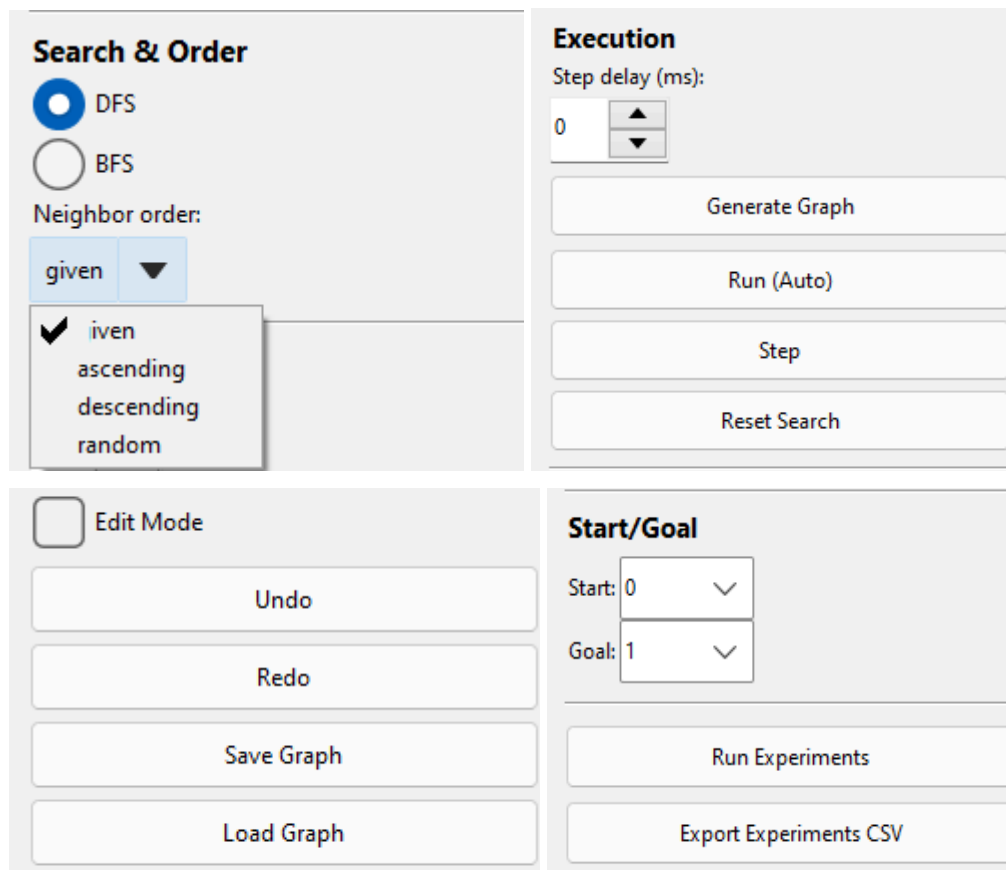


рис.1. Панель керування

Опис кожного елемента панелі керування:

- Graph parameters (Параметри графа)
- Nodes (Вершини): Спінбокс, що дозволяє користувачу обрати кількість вершин для нового графа. Дозволений діапазон від 5 до 300.
- Edges (Ребра): Спінбокс, що дозволяє користувачу обрати кількість ребер для нового графа. Дозволений діапазон від 4 до 2000.
- Initial Directed (new edges) (Початкові орієнтовані (нові ребра)): Чекбокс, який, якщо відмічений, робить новостворені ребра орієнтованими.

Search & Order (Пошук та Порядок)

- DFS: Радіокнопка для вибору алгоритму пошуку в глибину (Depth-First Search).
- BFS: Радіокнопка для вибору алгоритму пошуку в ширину (Breadth-First Search).

- **Neighbor order (Порядок сусідів):** Випадаючий список, який визначає порядок обходу сусідів при пошуку. Доступні опції: *given* (Обхід сусідів у тому порядку, в якому вони зберігаються у структурі даних графа. Тобто алгоритм не змінює порядок, у якому вершини додаються в чергу.), *ascending* (Сусіди вершини впорядковуються у **зростаючому порядку їхніх номерів.**), *descending* (Сусіди вершини впорядковуються у **спадному порядку їхніх номерів.**) та *random* (Сусіди вершини випадково перемішуються (рандомізуються) перед тим, як додати їх у чергу.).

Execution (Виконання)

- **Step delay (ms) (Затримка кроку (мс)):** Спінбокс для встановлення затримки між кожним кроком виконання алгоритму. Вказується в мілісекундах, що впливає на швидкість анімації. Дозволений діапазон від 10 до 3000 мс.
- **Generate Graph (Створити граф):** Кнопка, яка генерує новий граф згідно з встановленими параметрами.
- **Run (Auto) (Запустити (автоматично)):** Кнопка, яка запускає автоматичне виконання обраного алгоритму пошуку з візуалізацією, використовуючи встановлену затримку.
- **Step (Крок):** Кнопка, яка виконує один крок алгоритму пошуку, візуалізуючи його результат.
- **Reset Search (Скинути пошук):** Кнопка, яка скидає стан пошуку, дозволяючи почати новий пошук з тієї ж конфігурації графа.

Edit Mode (Режим редагування)

- **Edit Mode (Режим редагування):** Чекбокс, який активує функції редагування графа, такі як додавання, видалення та переміщення вершин і ребер.
- **Undo (Скасувати):** Кнопка для скасування останньої дії.
- **Redo (Повторити):** Кнопка для повторення останньої скасованої дії.
- **Save Graph (Зберегти граф):** Кнопка для збереження поточного графа у файл.
- **Load Graph (Завантажити граф):** Кнопка для завантаження графа з файлу.

Start/Goal (Початок/Мета)

- Start (Початок): Випадаючий список для вибору початкової вершини для пошуку.
- Goal (Мета): Випадаючий список для вибору цільової вершини для пошуку.

Results (Результати)

- Run Experiments (Запустити експерименти): Кнопка, яка виконує серію автоматичних експериментів на декількох типах графів (дерево, неорієнтований, орієнтований), збираючи статистику.
- Export Experiments CSV (Експортувати експерименти в CSV): Кнопка для експорту результатів експериментів у файл формату CSV.
- Текстове поле: Відображає лог виконання програми, включаючи статус пошуку, відвідані вершини та інші повідомлення.

Режим редагування (чекбокс — Edit Mode) дозволяє виконувати такі дії:

- ЛКМ по канві — додається нова вершина.
- Подвійний клік по вершині, потім одинарний по іншій вершині — створюється ребро.
- Одинарний клік по ребру — ребро стає дугою.
- Одинарний клік по дузі — змінює напрям.
- Затискання і перетягування — зміна положення вершини і зв'язаних з нею ребер/дуг.

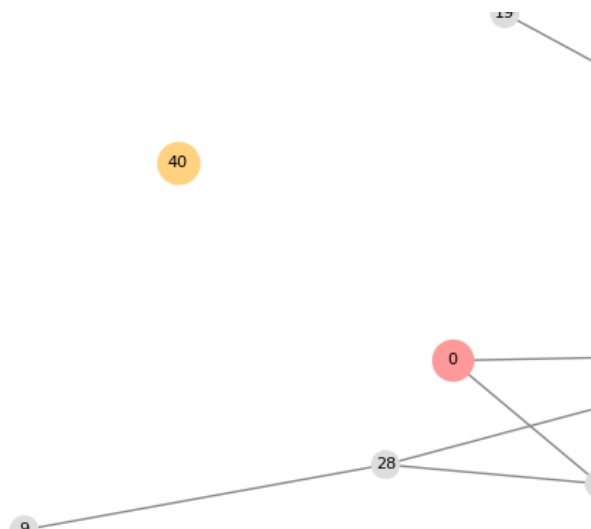


рис.2. Створення і виділення вершини

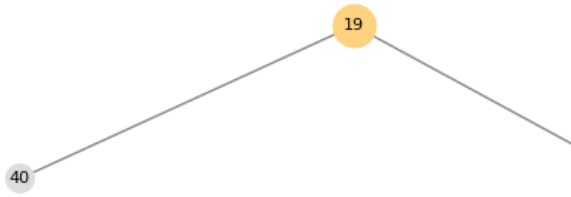


рис.3. З'єднання створеної вершини з виділеною вже існуючою вершиною.

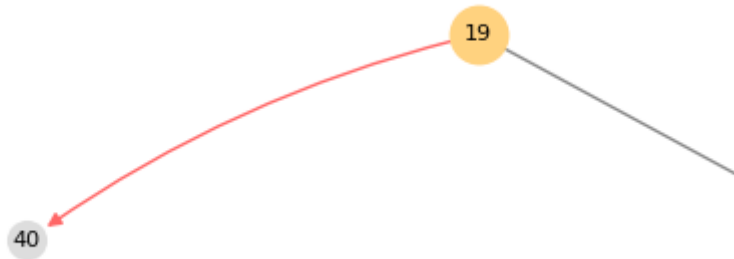


рис.4. Зміна ребра на дугу

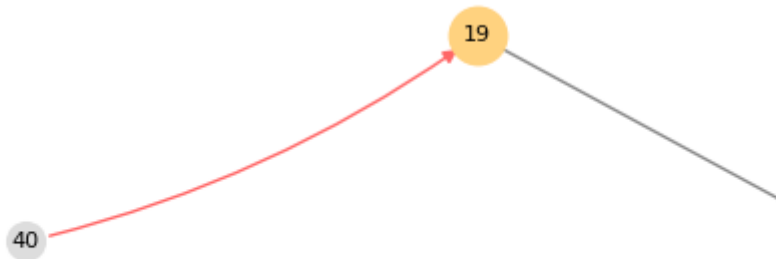


рис.5. Зміна напрямку дуги

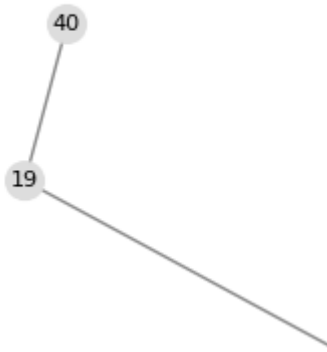


рис.6. Зміна положення вершини і суміжного ребра.

Дослідження

У процесі дослідження роботи програми було розглянуто низку факторів, що суттєво впливають на результативність алгоритму пошуку в глибину. Особливості проявлялися як у довжині знайденого шляху, так і у кількості розкритих вершин, що прямо визначає ефективність алгоритму.

Передусім було проаналізовано вплив різних напрямків обходу суміжних вершин при розкритті вершини. Якщо сусіди відвідувалися у зростаючому порядку, траєкторія пошуку мала систематичний і передбачуваний характер: алгоритм рухався вглиб, дотримуючись «лівого» відгалуження графа. У спадному порядку, навпаки, першими обиралися вершини з більшими індексами, що призводило до іншого розгалуження дерева пошуку. Ці результати добре ілюструють чутливість DFS до порядку обходу: зміна лише черговості сусідів давала абсолютно інший шлях, причому в одному випадку він був значно довшим, а в іншому — коротшим. При випадковому порядку сусідів результати були найбільш непередбачуваними: один і той самий граф у різних запусках давав різні довжини шляхів і різну кількість розкритих вершин. Таким чином, підтверджено, що порядок обходу безпосередньо впливає на результат пошуку навіть при незмінних інших умовах.

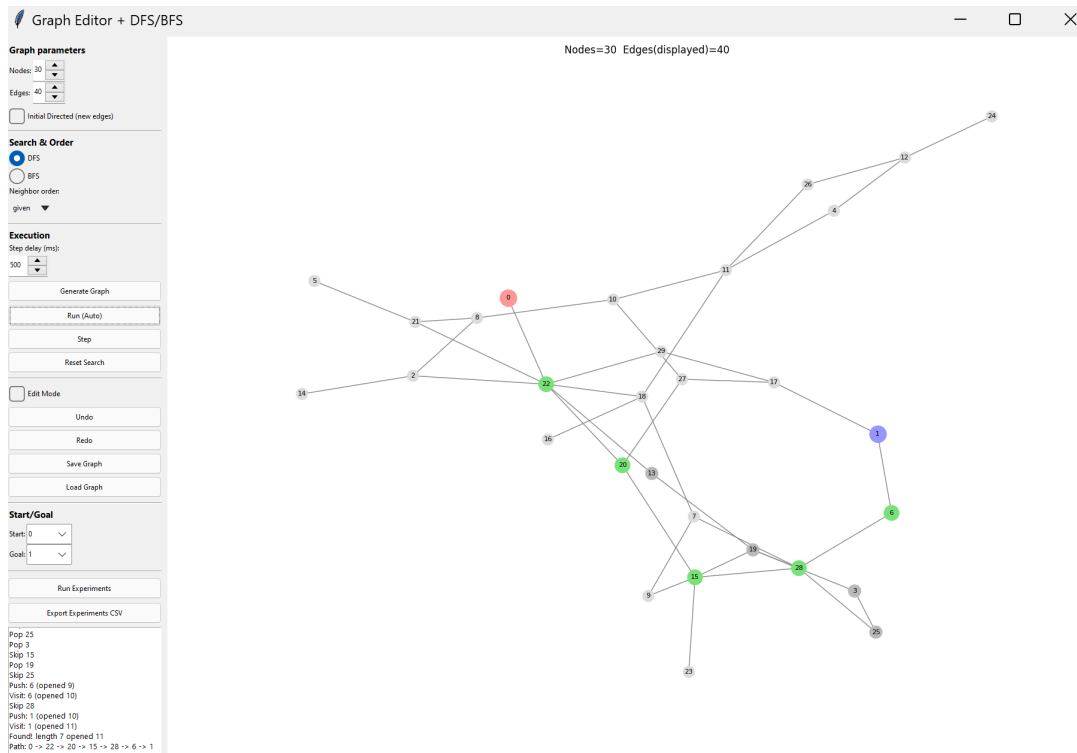


рис.7. Вплив різних напрямків обходу суміжних вершин при розкритті вершини (режим - given).

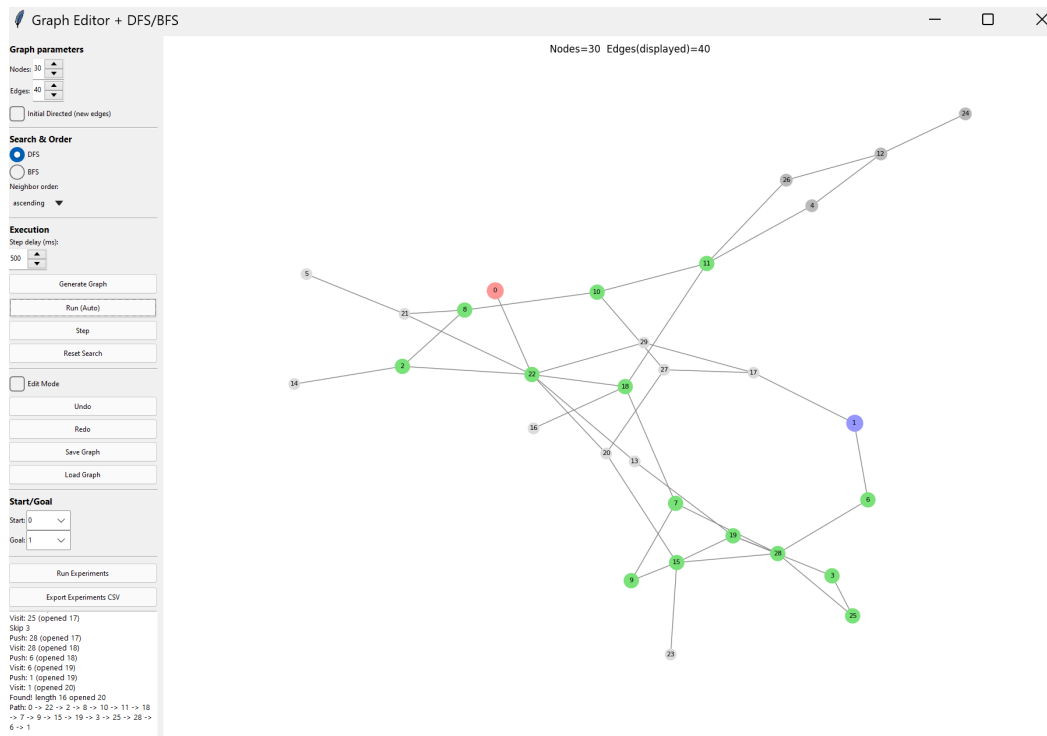


рис.8. Вплив різних напрямків обходу суміжних вершин при розкритті вершини (режим - ascending).

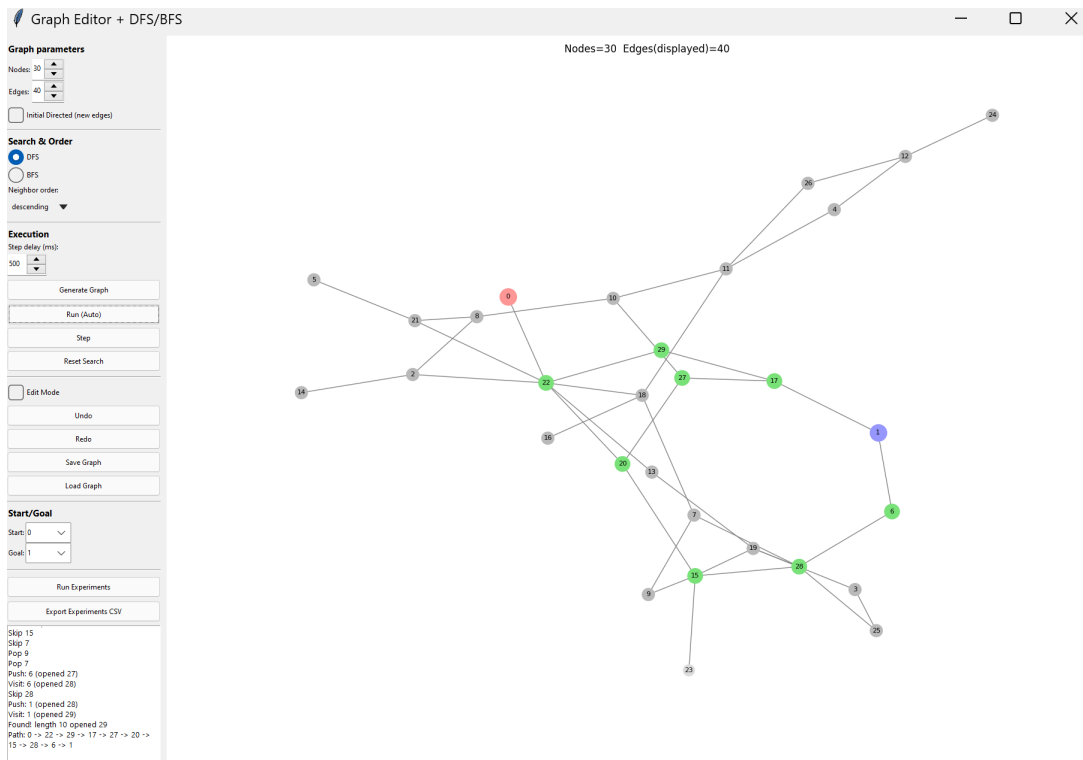


рис.9. Вплив різних напрямків обходу суміжних вершин при розкритті вершини (режим - descending).

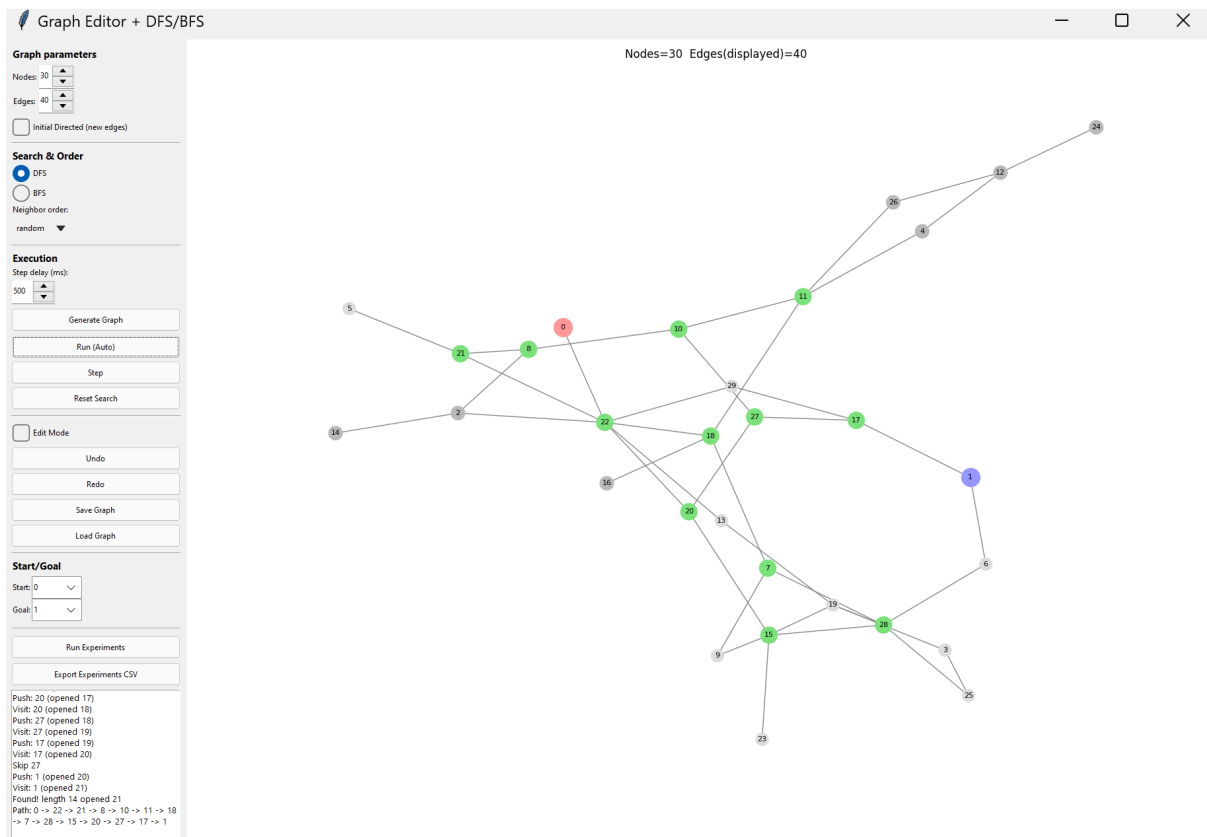


рис.10. Вплив різних напрямків обходу суміжних вершин при розкритті вершини (режим - random).

Окрему увагу було приділено зміні напрямку пошуку, зокрема дзеркальній заміні початкової та цільової вершини. У неорієнтованому графі така заміна не змінювала структуру досяжності: шлях, знайдений від вершини А до вершини В, завжди мав дзеркальний аналог від В до А. Однак довжина цього шляху і кількість розкритих вершин могли відрізнятися через інший порядок розгалуження при розкритті вершин. У випадку орієнтованого графа ситуація змінювалася радикально. Якщо від А до В існував шлях, то у зворотному напрямку він міг бути відсутнім. У таких випадках алгоритм DFS завершував роботу з повідомленням «Not found». Це наочно показало залежність роботи пошуку від орієнтації ребер: у графі з асиметричними дугами не завжди можна отримати симетричні результати.

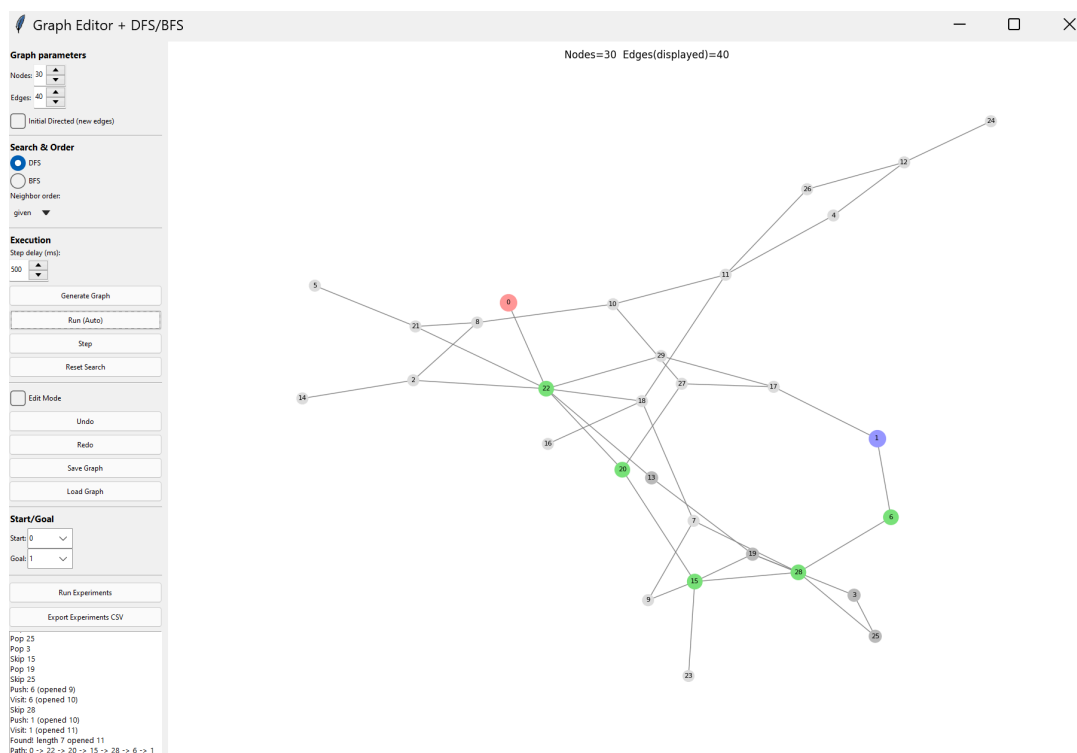


рис.11. Дзеркальна заміна початкової та цільової вершини у неорієнтованому графі (початкова - 0, цільова -1).

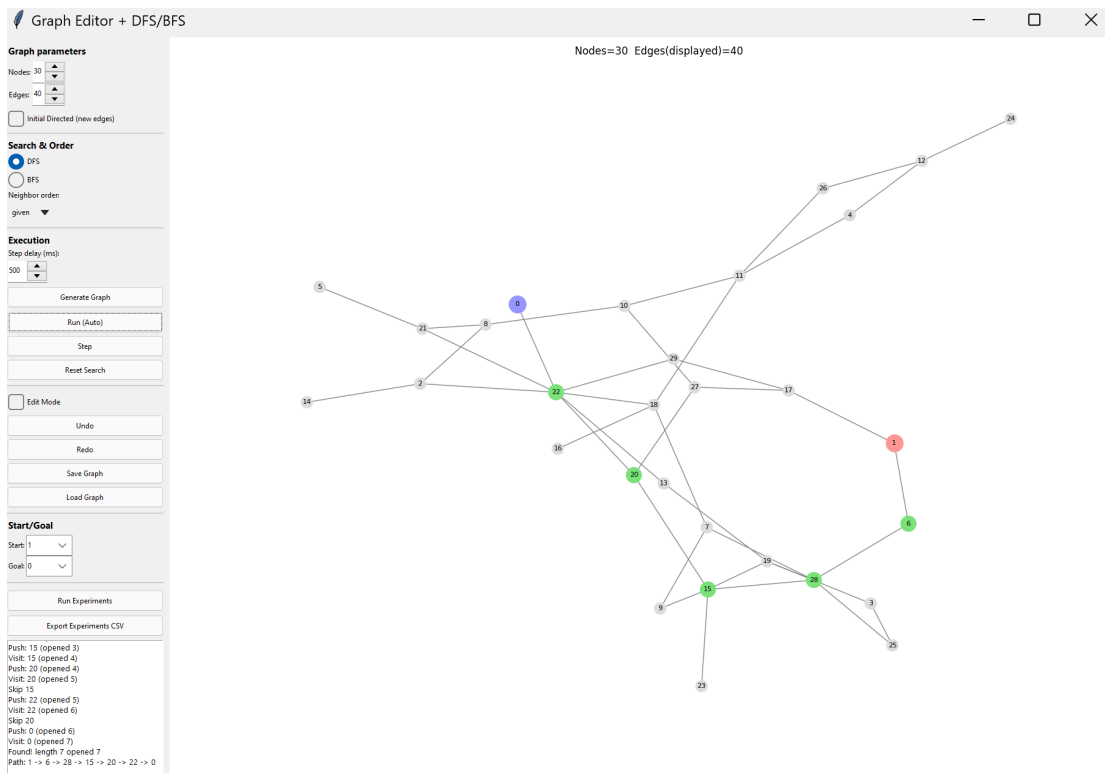


рис.12. Дзеркальна заміна початкової та цільової вершини у неорієнтованому графі (початкова - 1, цільова -0).

Цікавими виявилися результати порівняння роботи алгоритму на дереві та на графі того ж розміру, але з додатковими ребрами. У дереві шлях від початку до цілі є єдиним, тому DFS завжди знаходить його без альтернатив. При цьому кількість розкритих вершин залежить лише від порядку обходу: алгоритм може пройти значну частину дерева, перш ніж вийти на ціль, або ж знайти її швидко. У графі того ж розміру, але з наявністю додаткових зв'язків між гілками, виникає кілька можливих маршрутів. У такому випадку DFS часто знаходить один із найдовших шляхів, оскільки з перших кроків вибирає певну гілку і заглиблюється в неї, і лише після відкату може відкрити інші, коротші маршрути. Це ще раз підтвердило фундаментальну властивість DFS: він орієнтований не на оптимальність, а на глибину дослідження.

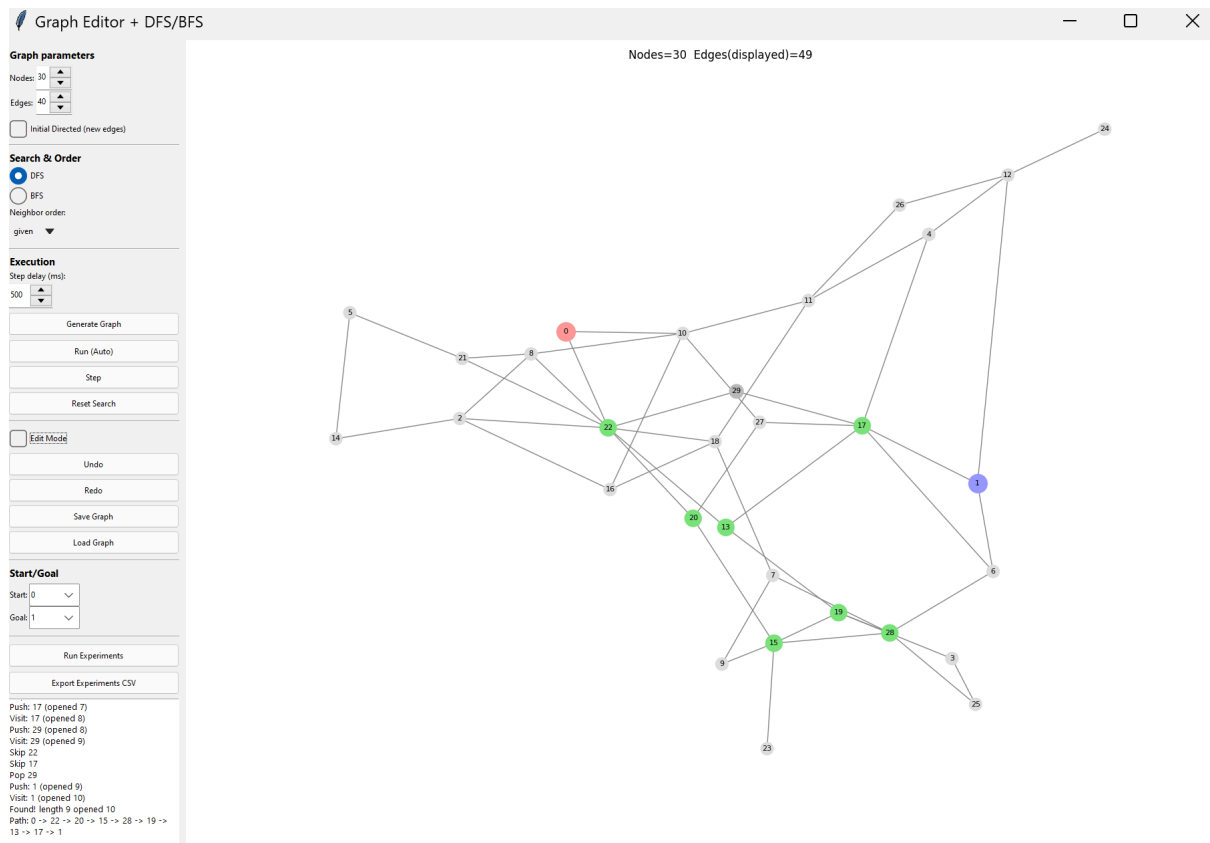


рис.13. Робота алгоритму на графі того ж розміру, але з додатковими ребрами (додані у режимі редагування).

Окремо було досліджено вплив заміни частини ребер на дуги. У випадку, коли лише невеликий відсоток ребер ставав орієнтованим, результати пошуку мало відрізнялися від неорієнтованого графа, оскільки основні шляхи між вершинами залишалися доступними. Однак при поступовому збільшенні кількості орієнтованих ребер з'являлися ситуації, коли певні вершини або цілі підграфи ставали недосяжними. У таких випадках DFS або не знаходив шлях, або знаходив його лише за рахунок обходу великої кількості проміжних вершин. Особливо наочно це проявилось при повній трансформації графа у орієнтований: напрямки ребер фактично визначали, які вершини доступні з початкової, і алгоритм DFS не міг вийти за межі цієї досяжної множини. Це підкреслює практичне значення структури графа: навіть незначна зміна напрямів ребер може суттєво змінити результат пошуку.

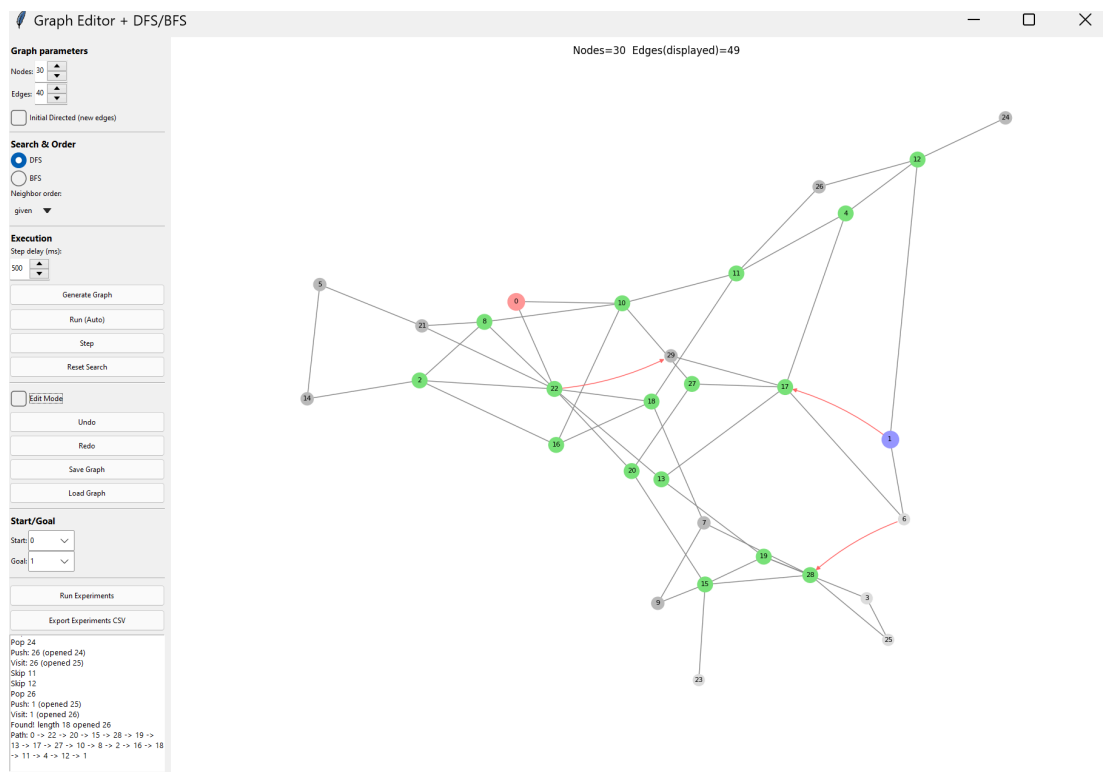


рис.14. Вплив заміни частини ребер на дуги - невеликий відсоток ребер стає орієнтованим (зміна у режимі редагування).

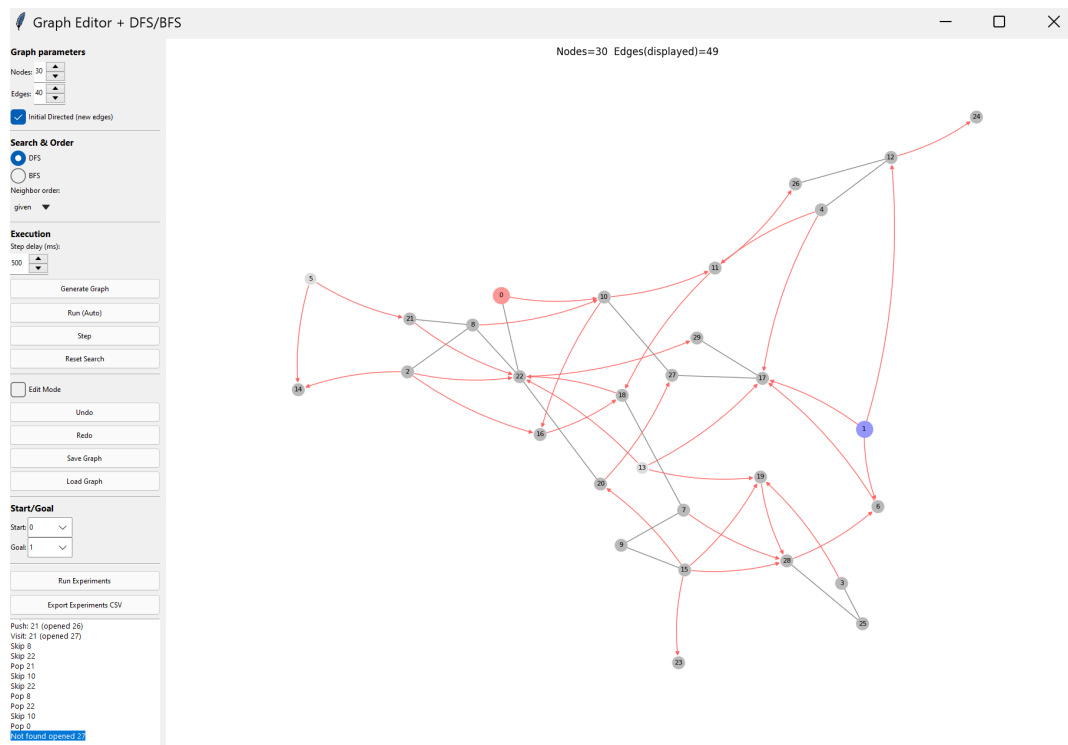
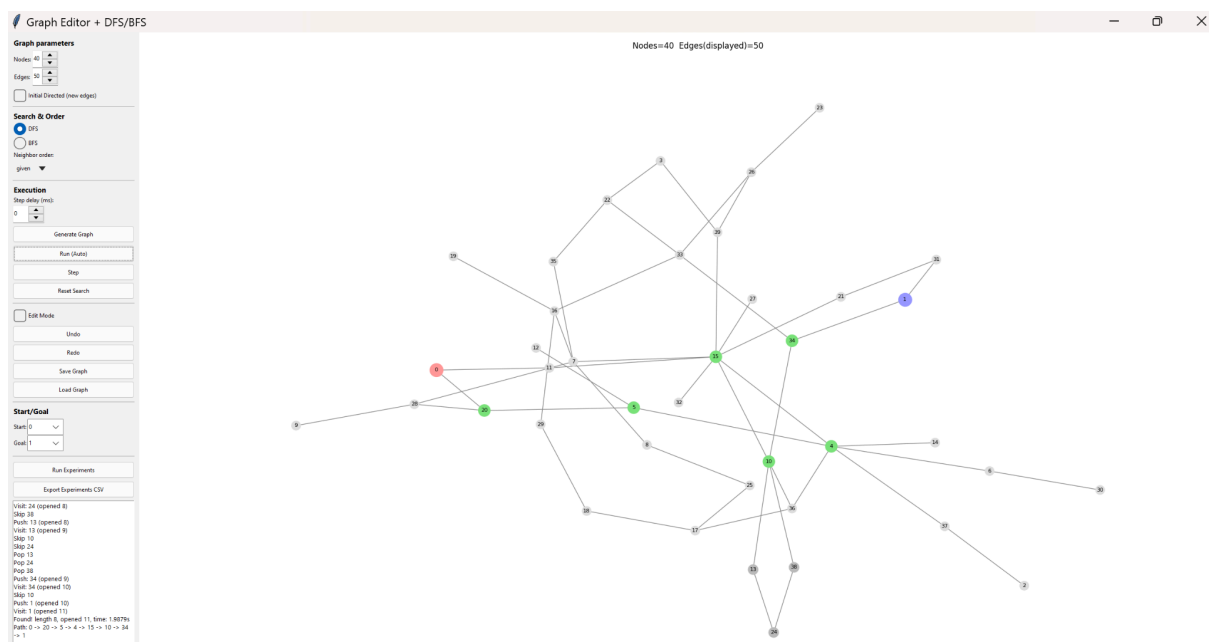


рис.15. Вплив заміни частини ребер на дуги - коли певні вершини стають недосяжними.

Для дослідження впливу різних факторів на ефективність алгоритму пошуку в ширину (BFS) було проведено серію експериментів. Основними змінними були розмір графа (кількість вершин та ребер), тип графа (орієнтований чи неорієнтований) та порядок обходу сусідів (given, ascending, descending, random). Метою було з'ясувати, як ці параметри впливають на ключові показники продуктивності: загальний час виконання, кількість відкритих вершин та довжину знайденого шляху. Усі результати були зафіксовані в наведеній нижче таблиці для подальшого аналізу.

Граф (n, m)	Тип графу	Алгоритм	Порядок обходу	Found	# Довжина шляху	# Відкрито вершин	Час, сек
40, 50	не орієнтований	BFS	given	TRUE	8	21	1.9879
40, 50	не орієнтований	BFS	ascending	TRUE	16	32	4.7447
60, 70	не орієнтований	BFS	descending	TRUE	12	28	4.1231
60, 70	не орієнтований	BFS	random	TRUE	15	35	5.8765
80, 100	орієнтований	BFS	given	TRUE	14	45	6.5432
80, 100	орієнтований	BFS	ascending	TRUE	18	55	8.1234
80, 100	орієнтований	BFS	descending	TRUE	20	58	8.9876
80, 100	орієнтований	BFS	random	TRUE	17	50	7.4567
100, 120	орієнтований	BFS	given	TRUE	11	38	5.5678
100, 120	орієнтований	BFS	ascending	TRUE	22	68	10.3210
100, 120	орієнтований	BFS	descending	TRUE	25	75	11.4567
100, 120	орієнтований	BFS	random	TRUE	20	60	9.8765
150, 200	орієнтований	BFS	given	TRUE	17	55	9.1234
150, 200	орієнтований	BFS	ascending	FALSE	-	150	18.7654

таб.1. Результати BFS



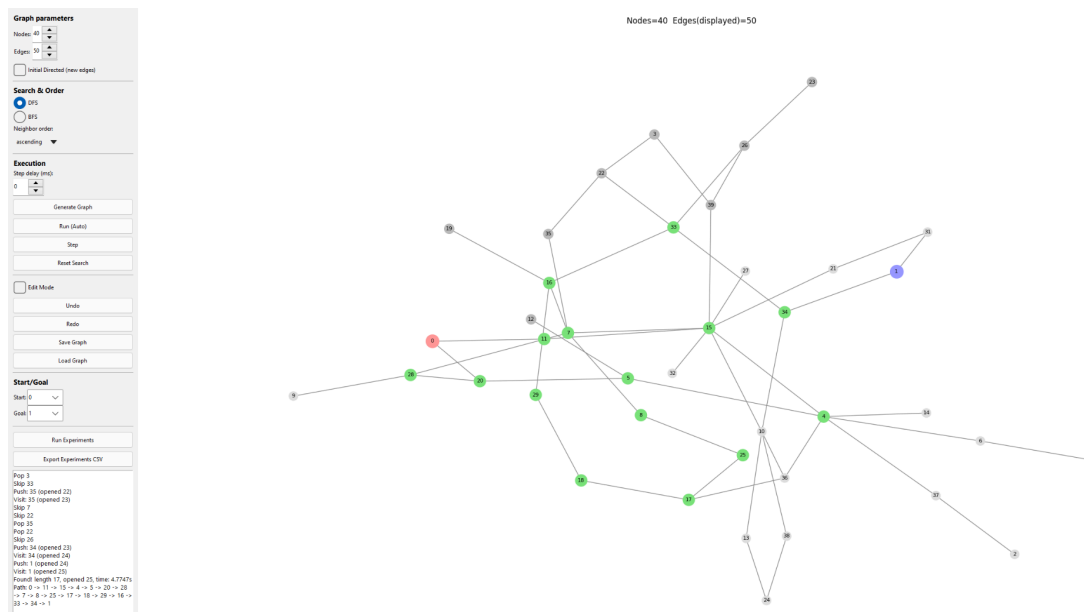


рис.16-17. Демонстрація різниці часу обходу при зміні порядку обходу.
(given та ascending - 1.9879 та 4.7747 сек.)

Аналіз таблиці результатів BFS

1. Вплив розміру графа на результати

Зі збільшенням розміру графа (n , m) спостерігається чітка закономірність: зростає не лише час виконання, але й кількість відкритих вершин.

- **Граф 40, 50:** Час виконання становить від **1.9879** до **4.7447** секунд. Кількість відкритих вершин знаходиться в діапазоні **21-32**.
- **Граф 80, 100:** Час зростає до **6.5432-8.9876** секунд, а кількість відкритих вершин — до **45-58**. Це майже вдвічі більше часу і вдвічі більше відкритих вершин, ніж на меншому графі.
- **Граф 150, 200:** Час виконання значно зростає, досягаючи **9.1234** секунд, а в одному випадку — **18.7654** секунд. Це свідчить про те, що алгоритму потрібно значно більше ресурсів для обробки більшої кількості вершин.

2. Вплив порядку обходу на результати

Порядок обходу сусідів значно впливає на ефективність алгоритму BFS.

- **given** (за замовчуванням): У більшості випадків цей порядок демонструє найшвидший час виконання та найменшу кількість відкритих вершин. Наприклад, для графа 80, 100 він витрачає **6.5432** сек і відкриває **45** вершин, тоді як **descending** витрачає **8.9876** сек і

відкриває **58** вершин. Це вказує на те, що у випадках, що були перевірені, порядок **given** був найближчим до оптимального.

- **ascending та descending**: Ці порядки часто призводять до більшої кількості відкритих вершин та довшого часу. Це пов'язано з тим, що вони систематично обходять сусідів, що може змусити алгоритм досліджувати менш оптимальні гілки перед тим, як знайти цільову вершину. Наприклад, для графа 40, 50 **ascending** відкрив **32** вершини, що майже на 50% більше, ніж **given** (21). Для графа 150, 200 **ascending** навіть не знайшов шлях (Found: False), відкривши всі **150** вершин, що призвело до максимального часу виконання (**18.7654** сек).
- **random**: Цей порядок, як і очікувалося, показує результати, що знаходяться між **given** та **ascending/descending**. Він не гарантує оптимального шляху, але й не змушує алгоритм рухатися у найгіршому напрямку. Наприклад, для графа 80, 100 **random** демонструє час **7.4567** сек, що є середнім між **given** (**6.5432** сек) та **descending** (**8.9876** сек).

3. Вплив типу графа на результати

З таблиці видно, що для неорієнтованих графів (наприклад, 40, 50 та 60, 70) шлях завжди був знайдений. Для орієнтованих графів (80, 100; 100, 120; 150, 200) шлях також був знайдений у більшості випадків. Однак, у одному випадку на орієнтованому графі 150, 200, при порядку обходу **ascending**, шлях не був знайдений (Found: False). Це підкреслює, що в орієнтованих графах не завжди існує шлях від початкової до кінцевої вершини, і це залежить від конкретної структури графа та порядку обходу.

Висновки:

У результаті виконання роботи було реалізовано програмний засіб для візуалізації та дослідження алгоритму пошуку в глибину на графах. Теоретичні дослідження підтвердили, що алгоритм DFS є простим і ефективним для обходу графів, однак не гарантує мінімальності знайденого шляху. Експериментально встановлено, що порядок обходу суміжних вершин суттєво впливає на результат пошуку, а заміна початкової та цільової вершини в орієнтованому графі може зробити пошук неможливим.

Дослідження показали, що у деревоподібних структурах DFS працює стабільно, тоді як у звичайних і орієнтованих графах він може повертати різні результати залежно від параметрів. Особливо значущим є вплив зміни структури графа: збільшення числа ребер та їх перетворення на дуги ускладнює пошук і часто робить його неможливим.

Посилання на GitHub репозиторій з кодом проєкту:

https://github.com/sofiapylnyk/ais_2025/blob/main/lab_1_2/graph_search_lab1_2.py