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, напряму 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.

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

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

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.

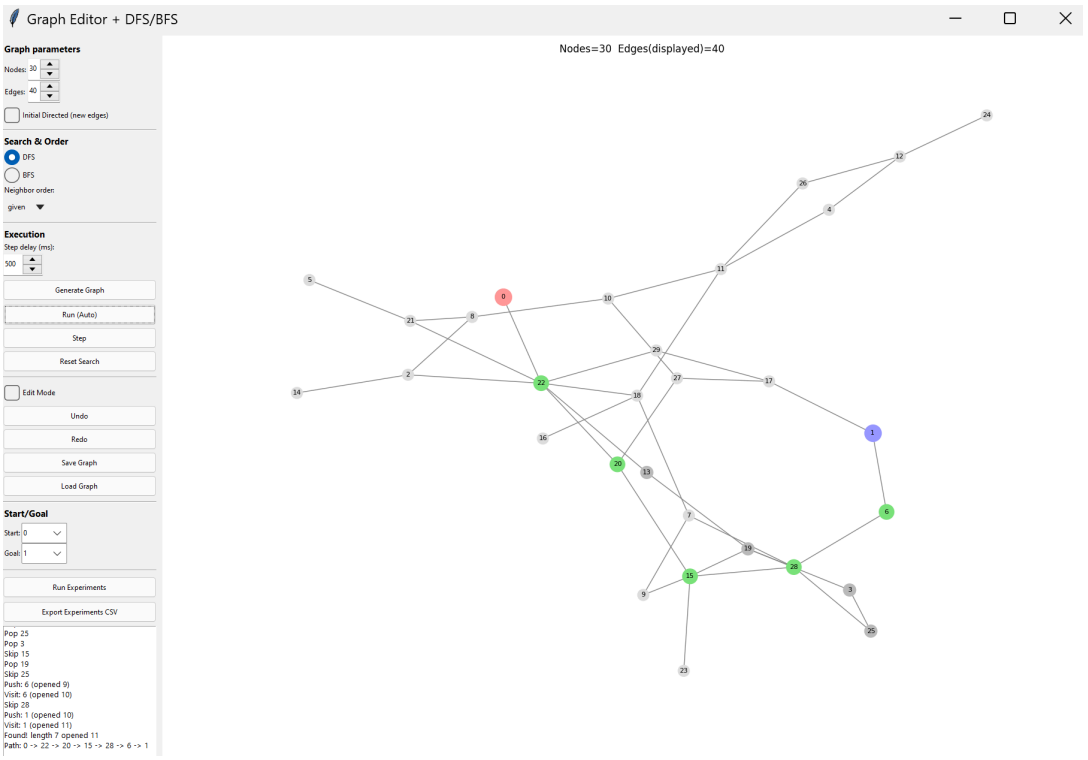

рис.1. Вплив різних напрямків обходу суміжних вершин при розкритті вершини (режим - given).

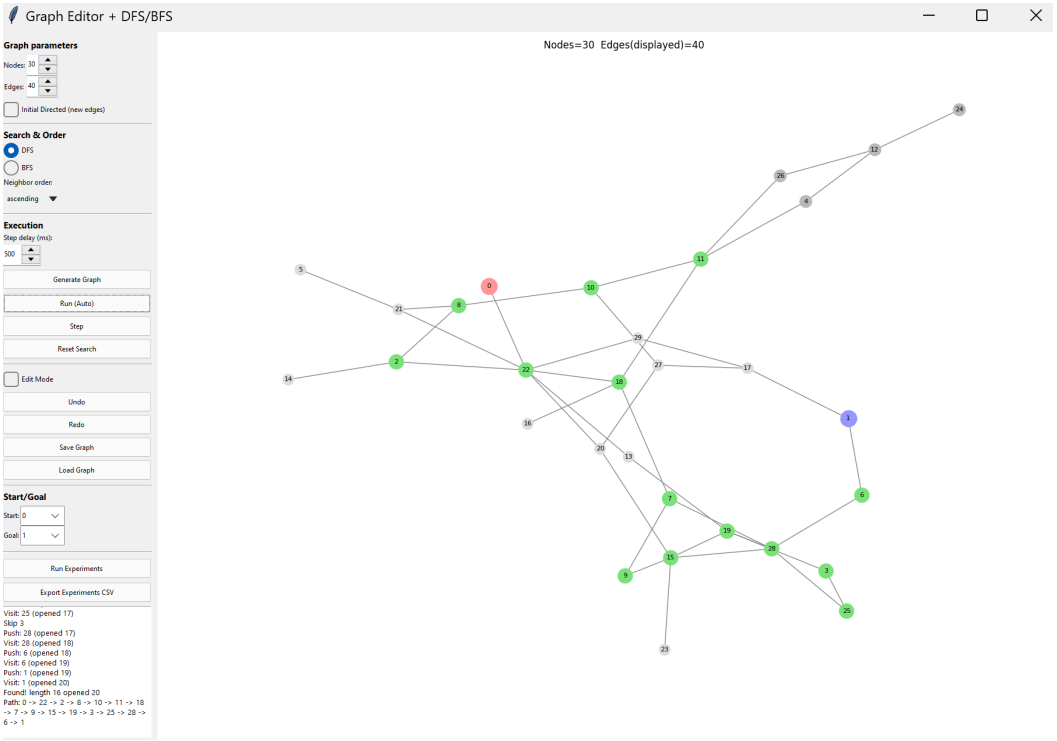

рис.2. Вплив різних напрямків обходу суміжних вершин при розкритті вершини (режим - ascending).

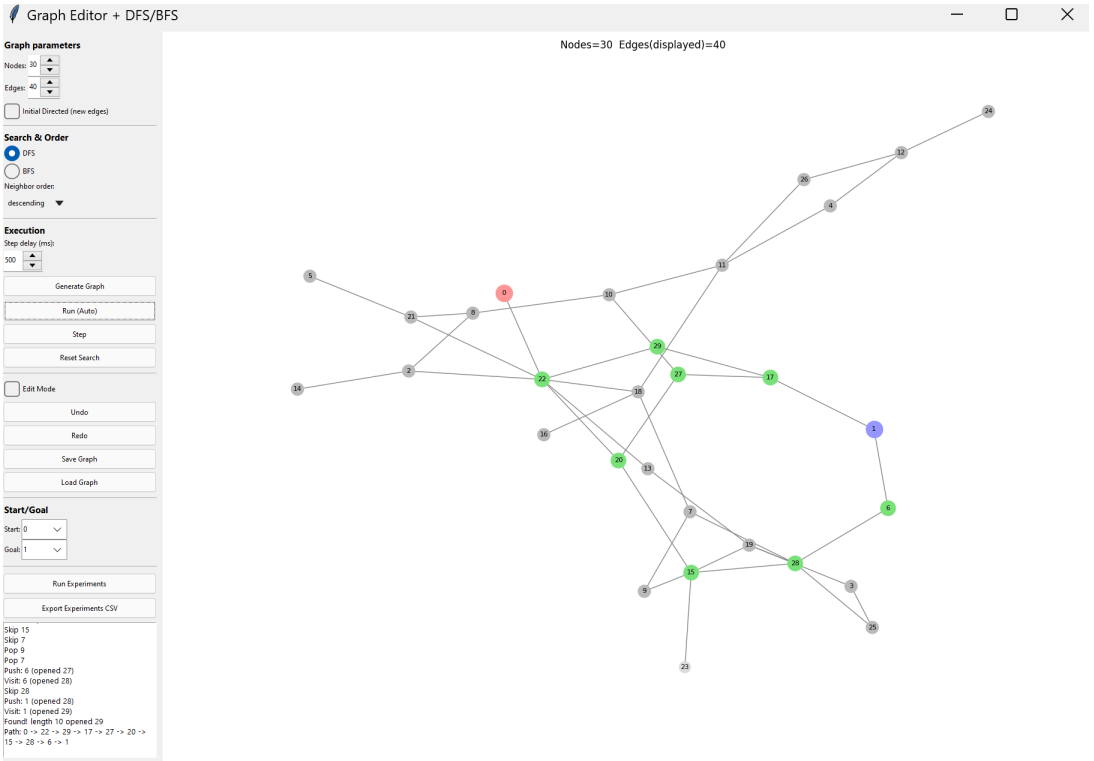

рис.3. Вплив різних напрямків обходу суміжних вершин при розкритті вершини (режим - descending).

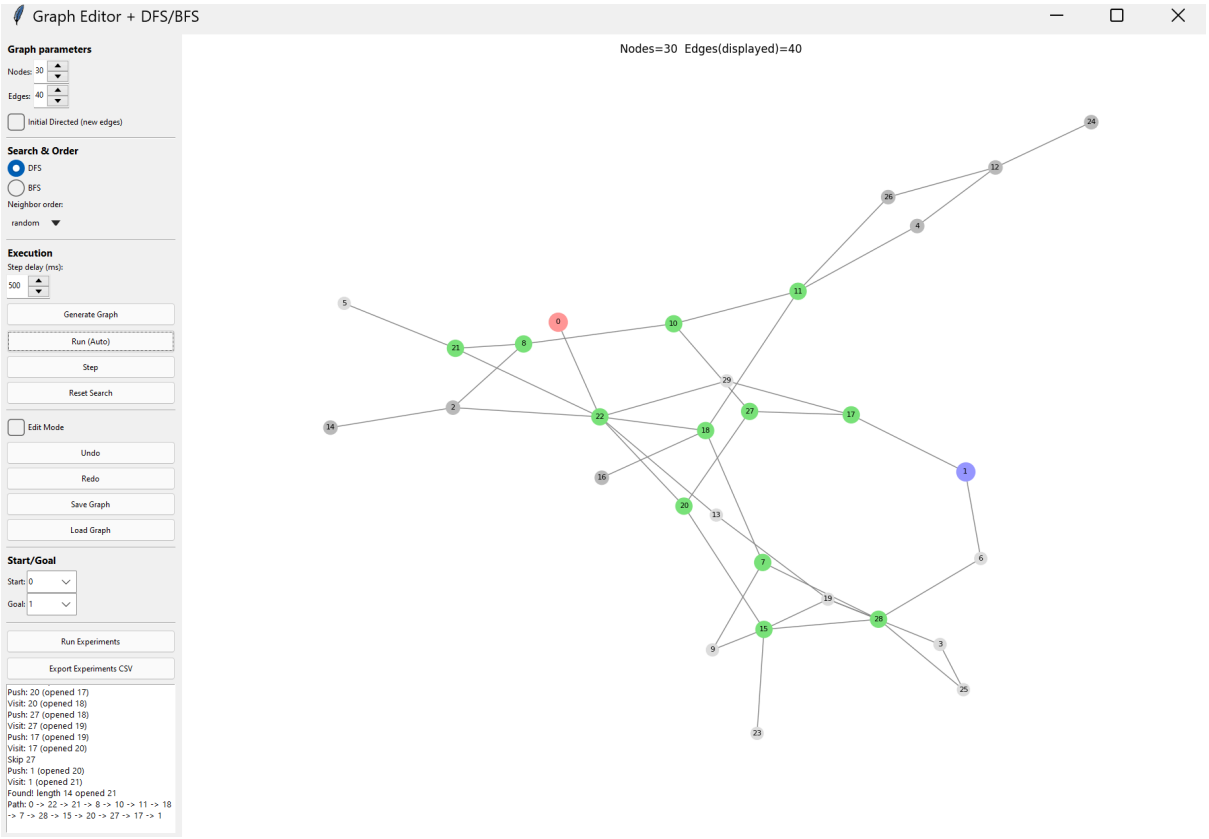

рис.4. Вплив різних напрямків обходу суміжних вершин при розкритті вершини (режим - random).

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 A до вершини B, завжди мав дзеркальний аналог від B до A. 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 A до B 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.

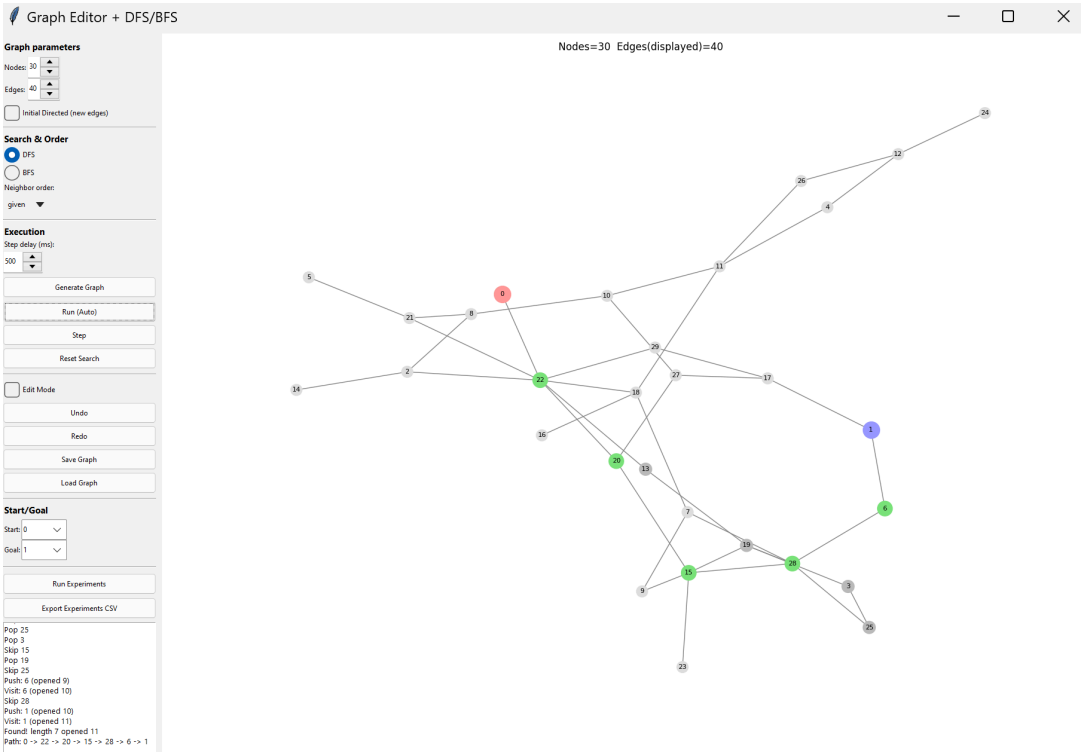

рис.5. Дзеркальна заміна початкової та цільової вершини у неорієнтованому графі (початкова - 0, цільова -1).

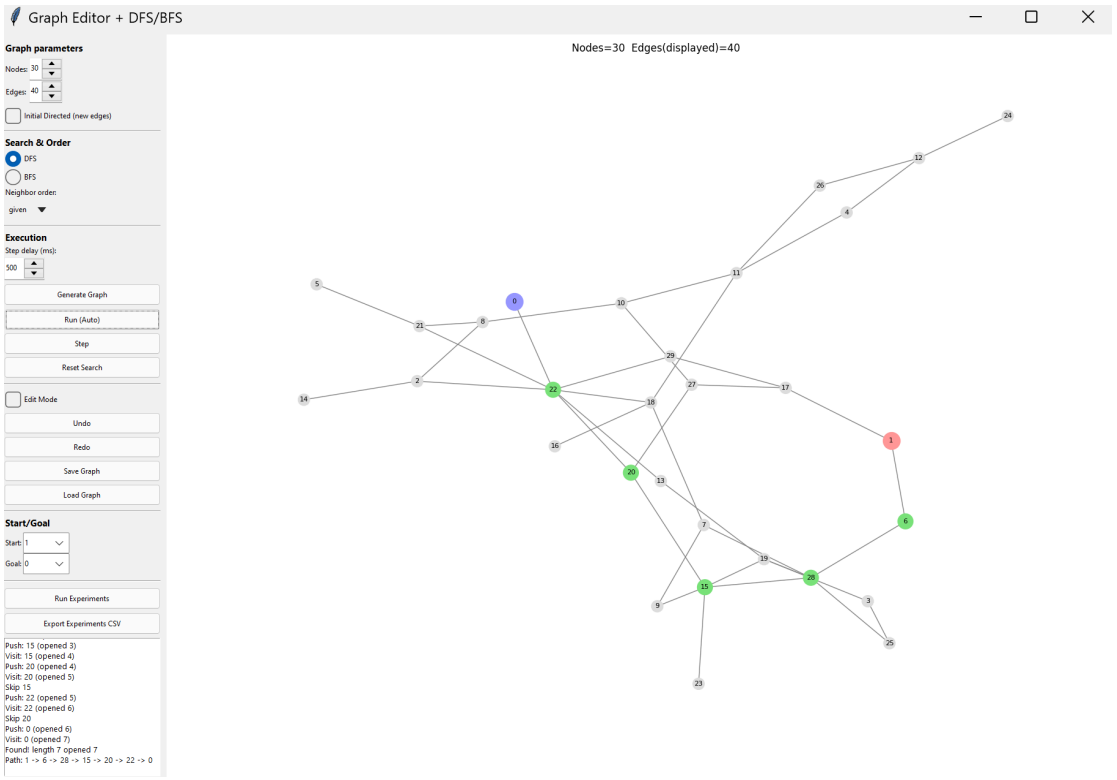

рис.6. Дзеркальна заміна початкової та цільової вершини у неорієнтованому графі (початкова - 1, цільова -0).

Було проаналізовано і вплив зміни розмірів та порядку графа. У невеликих графах з кількістю вершин близько 10–15 DFS швидко знаходив шлях, і навіть випадковий порядок обходу не призводив до суттєвої перевитрати ресурсів. Проте у графах порядку 50 і більше, особливо при густині близько 80–100 ребер, алгоритм розкривав значно більшу кількість вершин, перш ніж знаходив ціль, або ж знаходив її на довгому обхідному шляху. Це підтвердило, що DFS менш ефективний на великих і густих графах, оскільки він не гарантує пошуку найкоротшого маршруту і часто «занурюється» у глибокі гілки, які не ведуть до мети. Водночас BFS, який буде вивчено у наступній роботі, у таких випадках працює ефективніше, оскільки знаходить мінімальний шлях.

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.

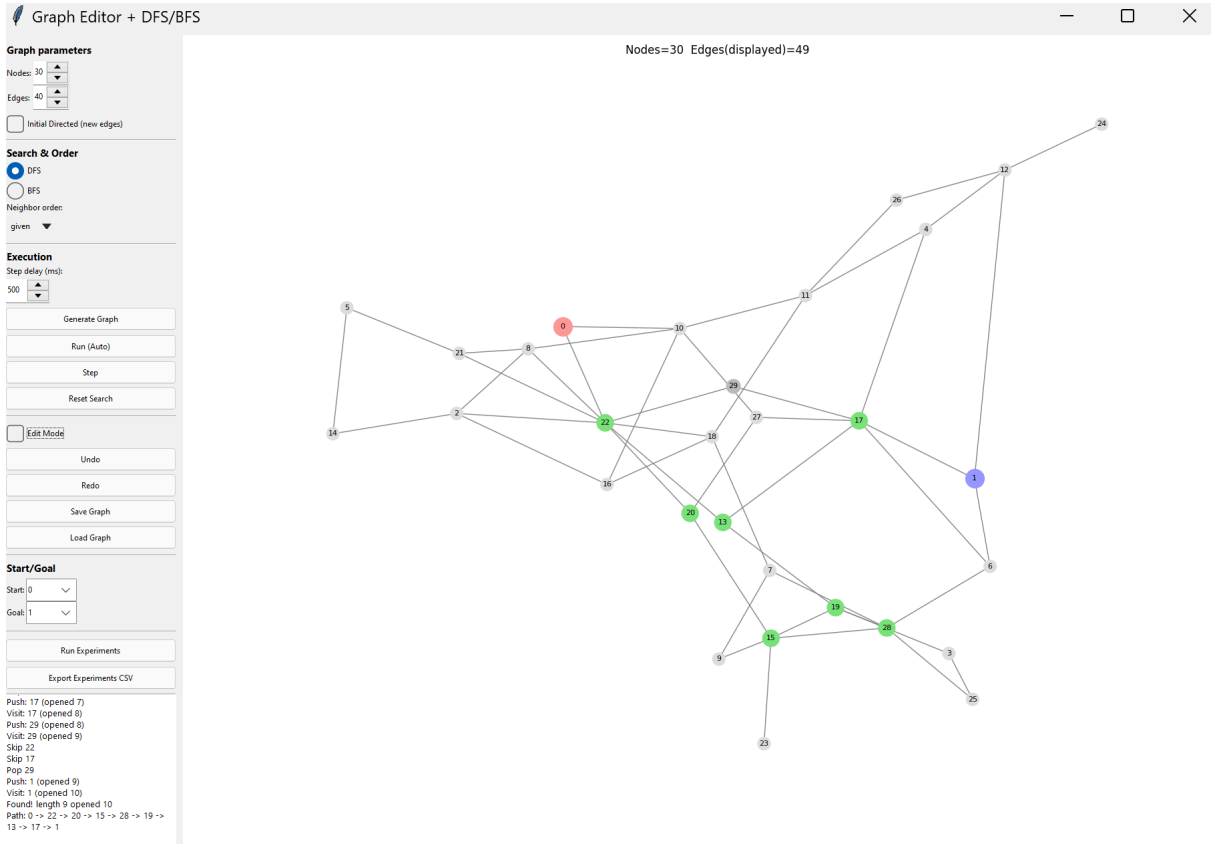

рис.7. Робота алгоритму на графі того ж розміру, але з додатковими ребрами (додані у режимі редагування).

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 проявилося 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.

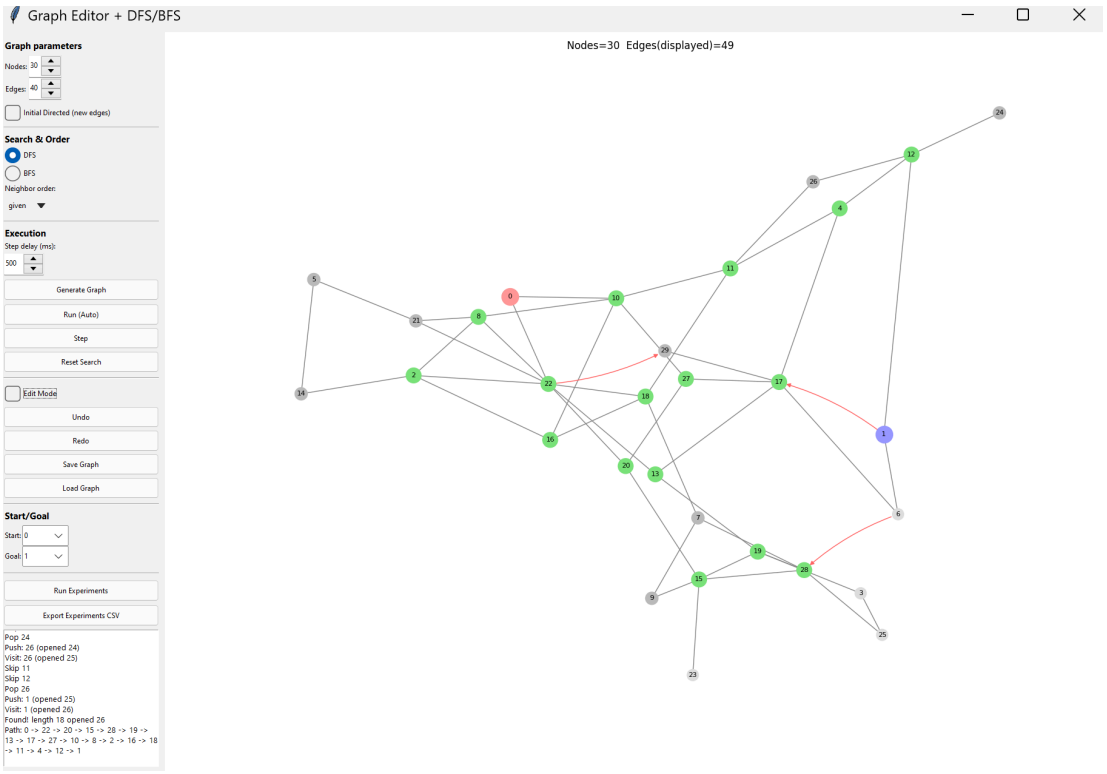

рис.8. Вплив заміни частини ребер на дуги - невеликий відсоток ребер стає орієнтованим (зміна у режимі редагування).

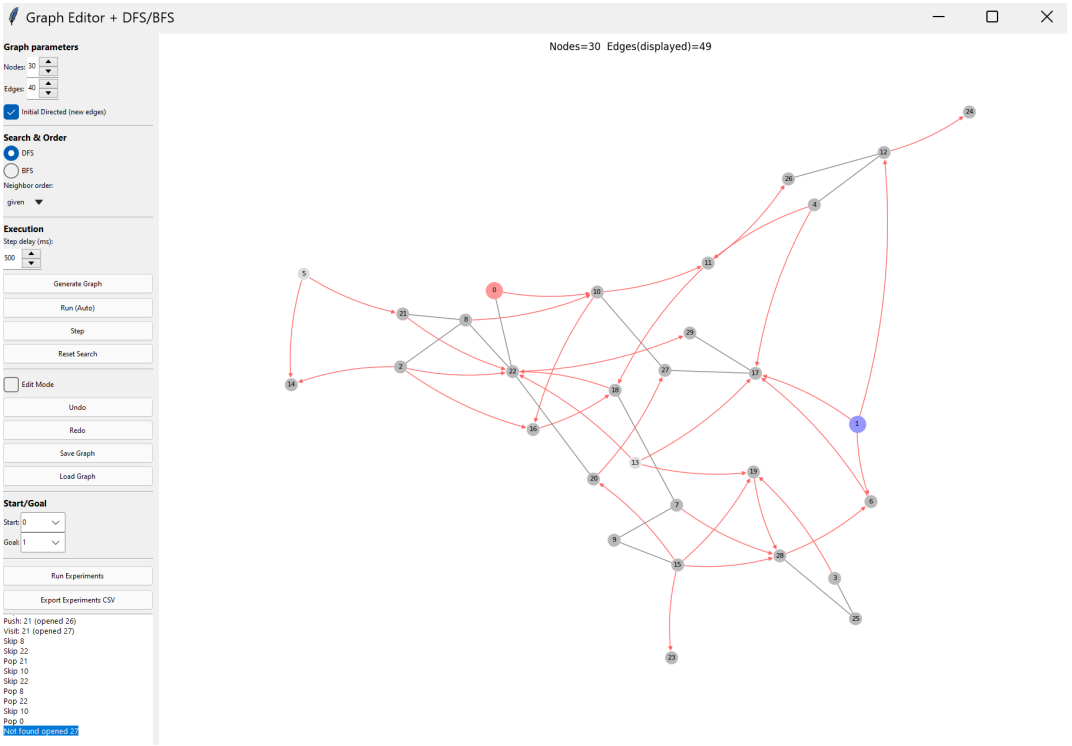

рис.9. Вплив заміни частини ребер на дуги - коли певні вершини стають недосяжними.

Таким чином, проведені дослідження дозволили на практиці переконатися у теоретичних особливостях пошуку в глибину. Було показано, що порядок обходу сусідів визначає ефективність пошуку, заміна напрямку пошуку змінює результати залежно від виду графа, а структура графа (розмірність, розгалуженість, наявність додаткових зв'язків або орієнтованих ребер) суттєво впливає як на кількість відвіданих вершин, так і на можливість знаходження цілі. Отже, DFS є потужним інструментом для обходу графів, але його ефективність сильно залежить від контексту, у якому він застосовується.

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.