УДК 519.7:316.472.4]:004.832.2-042.2

Афанасьєва Д. С., *здобувачка 2 курсу спеціальності 122 Комп'ютерні науки,*
Вєтров О. С., *старший викладач кафедри інформаційних технологій*

ПОРІВНЯННЯ АЛГОРИТМІВ ПОШУКУ В ШИРИНУ ТА В ГЛИБИНУ ДЛЯ ОБХОДУ ГРАФІВ У СОЦІАЛЬНИХ МЕРЕЖАХ

Донецький національний університет імені Василя Стуса, м. Вінниця

Соціальні мережі перетворилися на невичерпне джерело даних і відкрили нові можливості для аналізу поведінки користувачів, передбачення трендів та взаємозв'язків між людьми. Це спонукає до необхідності розробки та оптимізації алгоритмів обходу графів, які є важливим складником для ефективного аналізу й використання даних у соціальних мережах. Обрання правильного алгоритму стає ключовим етапом якісної та швидкої обробки великих масивів даних.

Алгоритм пошуку в ширину (BFS) розглядає граф як структуру, де вершини відвідуються у порядку росту відстані від початкової вершини. Цей алгоритм має схожість з обходом бінарного дерева, єдина відмінність у тому, що на відміну від них, графи можуть містити цикли. Для пошуку в ширину розглядається простий зв'язний граф з числом вершин V та числом ребер E [1]. Тому для початку обирається перша вершина, з якої починається алгоритм, далі відвідуються усі вершини, які знаходяться на відстані одного ребра від початкової, потім усі вершини на відстані двох ребер і так далі [2]. Під час обходу графу спершу вершини додаються в чергу і після їх проходу заносяться в масив відвіданих вершин. Часова складність алгоритму залежатиме від кількості вершин та ребер у графі й буде становити O(V+E). На рис. 1 показано реалізацію цього алгоритму мовою Python.

```python
def bfs(graph, start):
    visited = set()
    queue = []
    visited.add(start)
    queue.append(start)
    visited_nodes = []

    while queue:
        vertex = queue.pop(0)
        visited_nodes.append(vertex)

        for neighbor in graph[vertex]:
            if neighbor not in visited:
                visited.add(neighbor)
                queue.append(neighbor)

    return visited_nodes
```

Рис. 1. Реалізація алгоритму пошуку в ширину [2]

Для алгоритму пошуку в глибину (DFS) розглядається простий зв'язний граф з числом вершин V та числом ребер E [1]. Обхід графу розпочинається з вибору будь-якої вершини графу, яку ми поміщаємо до стека. Потім з вершини з верху стека ми беремо першу вершину, додаємо її до списку відвіданих і додаємо всі її сусідні вершини, які ще не були відвідані, до вершини стека. Ці кроки повторюються, поки стек не стане пустим, тобто всі вершини будуть відвідані. Часова складність алгоритму буде такою ж, як в алгоритму пошуку в ширину. Вона залежатиме від кількості вершин та ребер у графі й буде становити O(V+E). Отже, різниця в реалізації цих алгоритмів полягає в тому, що у BFS використовується черга для обходу вершин графу, а у DFS використовується стек, переходячи максимально можливо далеко вглиб від стартової вершини, перш ніж повернутися до інших сусідніх вершин. На рис. 2 показано реалізацію DFS мовою Python.

```python
def dfs(graph, start):
    visited = []
    stack = [start]

    while stack:
        current_vertex = stack.pop()
        if current_vertex not in visited:
            visited.append(current_vertex)
            print(current_vertex)

        for neighbor in graph[current_vertex]:
            if neighbor not in visited:
                stack.append(neighbor)

    return visited
```

Рис. 2. Реалізація алгоритму пошуку в глибину [3]

У контексті соціальних мереж застосування BFS та DFS залежить від конкретних потреб та постановки задачі. DFS має менші вимоги до пам'яті, порівняно з BFS, оскільки потрібно зберігати лише шлях від початкової вершини до поточної. Проте це може зробити DFS вразливим до нескінчених циклів, якщо граф містить такі цикли [4].

DFS може бути застосований у соціальних мережах для виявлення груп або спільнот користувачів. Наприклад, використання DFS може допомогти визначити, які користувачі утворюють тісні групи друзів або мають спільні інтереси. Також DFS може використовуватися для виявлення впливових осіб у мережі, які можуть мати важливий вплив на інших користувачів.

З іншого боку, BFS частіше використовується для вирішення конкретних завдань, як-от пошук найкоротшого шляху між користувачами або визначення рівня віддаленості між ними. Наприклад, за допомогою BFS можна знайти найшвидший спосіб зв'язку між двома користувачами або визначити, скільки зв'язків їх відокремлює. Такий аналіз може бути корисним для виявлення ступеня взаємодії між користувачами та оцінки їх впливу в мережі.

Отже, у цій роботі було досліджено два ключові алгоритми обходу графів: DFS (пошук в глибину) та BFS (пошук в ширину), і їх роль в аналізі соціальних

мереж. DFS дає змогу виявляти групи та спільноти користувачів, водночас, як BFS допомагає визначити найкоротший шлях між ними. Розуміння принципів роботи цих алгоритмів дає змогу краще аналізувати та розуміти структуру соціальних мереж, розподіл інформації та взаємозв'язки між користувачами.

Список використаних джерел

1. Кузьменко І. М. Теорія графів: навчальний посібник для здобувачів ступеня бакалавра за спеціальністю 122 «Комп'ютерні науки». Київ: КПІ ім. Ігоря Сікорського, 2020. 71 с. URL: https://ela.kpi.ua/server/api/core/bitstreams/fb0a4251-74d9-470b-88da-71abb4e85f93/content
2. Breadth First Search or BFS for a Graph. GeeksforGeeks. *Geeksforgeeks.* 26 Sep., 2024. URL: https://www.geeksforgeeks.org/breadth-first-search-or-bfs-for-a-graph/#breadth-first-search-or-bfs-for-a-graph (дата звернення: 15.04.2024).
3. Depth First Search (DFS). Programiz: вебсайт. URL: https://www.programiz.com/dsa/graph-dfs (дата звернення: 10.05.2024).
4. Graph traversal: DFS vs BFS. Hyperskill: вебсайт. URL: https://hyperskill.org/learn/step/35510#depth-first-search (дата звернення: 11.05.2024).

УДК: 004.9

Молодченко Д. В., *здобувач 2 курсу спеціальності 122 Комп'ютерні науки*
науковий керівник:
Потапова Н. А., *канд. екон. наук, доцент, доцент кафедри інформаційних технологій*

МОБІЛЬНИЙ ЗАСТОСУНОК ДЛЯ РОЗКЛАДУ ЗАНЯТЬ

Донецький національний університет імені Василя Стуса, м. Вінниця

Сучасна освітня система активно використовує корпоративні програми, що дають змогу здійснювати відеовиклики, листуватися та ділитися файлами, і все це можна переглянути навіть у транспорті, але до цієї групи застосунків досі не додали один тип програми – розклад. Здобувачі на початку кожного семестру стикаються з проблемою пошуку на телефоні файла, в якому міститься розклад. Аби подивитись номер аудиторії, в якій проходитиме заняття, доводиться долати шлях через файловий провідник телефона чи то ноутбука, який у здобувачів зазвичай запроваджений великою кількістю файлів. Функціонал застосунку має містити можливість додавати, редагувати та видаляти предмети, розклад, групи. До того ж у сучасному світі можливість отримати доступ до своїх даних на іншому пристрої стала вагомим плюсом для користувачів та сервісів що першими почали впроваджувати такий функціонал.

Мобільний застосунок – програмне забезпечення, платформою роботи якого є мобільні пристрої: смартфони та планшети. Зазвичай на цих пристроях встановлено одну з двох наступних операційних систем:

1. Android – відкрита операційна система, створена компанією Google у 2008 році на основі ядра Linux Kernel. Ця ОС стала надзвичайно популярною че-