Список використаних джерел

1. Гладка О. М., Карпович І. М., Сінчук А. М. Моделі економічної динаміки: посібник. Рівне: Університетська книга, 2019. 158 с. URL: https://core.ac.uk/download/pdf/297135502.pdf (дата звернення: 27.11.2018).

2. Клебанова Т. С., Дубровина Н. А., Полякова О. Ю. Моделювання економічної динаміки: посібник. Харків: Університетська книга, 2005. 244 с.

3. Січко Т. В., Рибак І. І. Системний підхід до аналізу організаційних структур. *Вісник Хмельницького національного університету. Технічні науки*. 2020, № 4. С. 70–74.

УДК 004.021

Левченко М. Р., здобувачка 2 курсу спеціальності 122 Комп'ютерні науки,

Вєтров О. С., старший викладач кафедри прикладної математики та кібербезпеки

РОЗГЛЯД ЗАСТОСУВАННЯ АЛГОРИТМІВ НА ГРАФАХ ДЛЯ ВИРІШЕННЯ ПРИКЛАДНИХ ЗАДАЧ

Донецький національний університет імені Василя Стуса, м. Вінниця

У сучасному інформаційному суспільстві, де величезний обсяг даних неперервно зростає, а взаємодії та зв'язки між об'єктами стають складнішими, застосування алгоритмів на графах є надзвичайно актуальним. Графи є відмінним математичним інструментом для моделювання та аналізу різноманітних систем, де об'єкти взаємодіють і мають складні структури.

З погляду комп'ютерних наук і дискретної математики графи є абстрактним методом представлення типів відносин, як-от дороги, що з'єднують міста, та інших видів мереж. Графи складаються з ребер та вершин. Вершина – це точка на графі, а ребро – це те, що з'єднує дві точки на графі [1].

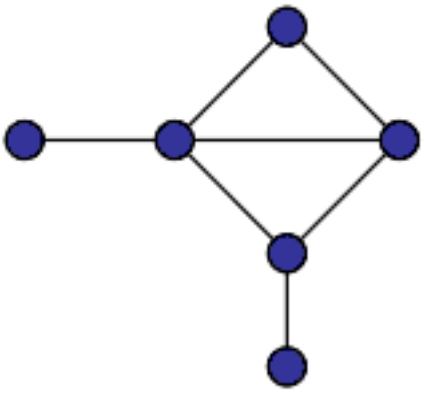

Рисунок 1. Загальний вигляд графу

Неорієнтований граф. У неорієнтованому графі ребра не мають напрямку. Це означає, що вони представляють просто зв'язок між двома вершинами, і цей зв'язок є взаємним. Якщо вершини A і B з'єднані ребром у неорієнтованому графі, то можна сказати, що A пов'язано з B, і навпаки. Формально ребро (A, B) еквівалентне ребру (B, A).

Орієнтований граф. У орієнтованому графі ребра мають напрямок. Кожне ребро вказує на одну з вершин як на вихідну (початкову) і на іншу як на вхідну (кінцеву). Такий граф може ілюструвати однобічні або однонапрямлені взаємозв'язки між об'єктами. Наприклад, якщо ребро (A, B) існує в орієнтованому графі, це не означає автоматично наявність ребра (B, A).

Пошук в ширину (BFS) – це алгоритм обходу або пошуку в графі, який починає з визначеної вершини (часто називається «початковою вершиною») і поступово відвідує всі сусідні вершини на поточному рівні перед переходом на наступний рівень.

У незваженому графі, де кожне ребро має однакову вагу (або вага не враховується взагалі), BFS знаходить найкоротший шлях між двома вершинами, тобто шлях, який має мінімальну кількість ребер. Алгоритм гарантує знаходження найкоротшого шляху, оскільки він відвідує вершини в порядку збільшення відстані від початкової вершини [3].

Алгоритм працює за $O(n + m)$, де n – кількість вершин, m – кількість ребер.

Основна ідея полягає в тому, що вершини додаються в чергу для обробки в порядку їх відстані від початкової вершини, і обробка продовжується доти, поки черга не стане порожньою.

Алгоритм пошуку в ширину (англ. breadth-first research, BFS) використовують також для:

1) вебсерфінгу;
2) пошуку друзів у соціальних мережах;
3) доступу до мережі Internet;
4) обрахунку кон'юнкції;
5) перевірки моделей у теорії автоматів та ін.

Загалом алгоритм пошуку в ширину розглядає простий зв'язний граф G = <V, E>. Задачею алгоритму є побудова маршруту від початкової і до всіх вершин графу, обхід графу [2].

Задача. Є лабіринт у вигляді сітки, де кожна клітина може бути або вільною, або перешкодою. Кожна вільна клітина – це вершина графу, а сусідні вільні клітини – це ребра, які їх з'єднують. Потрібно знайти найкоротший шлях від початкової до кінцевої точки лабіринту, обходячи перешкоди.

У цьому коді використовується BFS (пошук в ширину) для знаходження найкоротшого шляху у лабіринті. Лабіринт представлений у вигляді двовимірного списку (maze), де 0 відповідає вільній клітині, а 1 – перешкоді.

```
maze = [
    [0, 1, 0, 0, 0],
    [0, 1, 0, 1, 0],
    [0, 0, 0, 1, 0],
    [1, 1, 1, 1, 0],
    [0, 0, 0, 0, 0]
]
```

Рисунок 2. Двовимірний масив

Спочатку створюємо двовимірний список visited, який використовується для відстеження відвіданих клітин у лабіринті. Кожен елемент цього списку визначається значенням False, яке вказує на те, що відповідна клітина ще не була відвідана. Після цього визначаємо напрямки руху в лабіринті, представлені координатами (dx, dy) для руху праворуч, вниз, ліворуч та вгору. Ці напрямки використовуються для визначення сусідніх клітин під час обходу лабіринту.

```
def shortest_path_in_maze(maze, start, end):
    rows, cols = len(maze), len(maze[0])
    visited = [[False] * cols for _ in range(rows)]
    directions = [(0, 1), (1, 0), (0, -1), (-1, 0)]
```

Рисунок 3. Двовимірний масив та напрямок руху в лабіринті

Далі створюємо чергу queue з початковою клітиною та відстанню 0 і позначаємо початкову клітину як відвідану.

```
queue = deque([(start[0], start[1], 0)])
visited[start[0]][start[1]] = True
```

Рисунок 4. Черга з початковою клітиною

Поки черга не порожня: виймаємо клітину з черги, перевіряємо, чи досягли кінцевої точки, додаємо сусідні клітини до черги і позначаємо їх як відвідані.

```
while queue:
    x, y, distance = queue.popleft()

    if (x, y) == end:
        return distance

    for dx, dy in directions:
        new_x, new_y = x + dx, y + dy

        if is_valid_move(new_x, new_y):
            queue.append((new_x, new_y, distance + 1))
            visited[new_x][new_y] = True
```

Рисунок 5. Реалізація пошуку в ширину

Повертаємо довжину найкоротшого шляху або – 1, якщо досягти кінцевої точки неможливо.

```
result = shortest_path_in_maze(maze, start_point, end_point)
print("Найкоротший шлях:", result)
```

Рисунок 6. Виведення результату

```
Найкоротший шлях: 12

Process finished with exit code 0
```

Рисунок 7. Результат програми

Підсумовуючи, варто зазначити, що використання алгоритмів на графах для вирішення прикладних завдань виявляється дуже ефективним і практичним напрямом. Розглянуті алгоритми, як-от пошук найкоротшого шляху в лабіринті з допомогою BFS, можуть бути застосовані в різноманітних областях, зокрема в оптимізації маршрутів кур'єрських служб у місті. Застосування теорії графів та алгоритмів на графах дає змогу розв'язувати складні завдання шляхом моделювання взаємозв'язків між об'єктами. Цей підхід виявляється важливим і у сучасному світі, де оптимізація та аналіз мережевих структур стають ключовими вимогами для ефективного вирішення прикладних задач.

Список використаних джерел

1. Medium. Графи: основи, алгоритми, теорії. URL: https://medium.com/nuances-of-programming/графы-основы-теории-алгоритмы-поиска-b93672f597472

2. Теорія графів. URL: https://ela.kpi.ua/bitstream/123456789/35854/1/Teoriia_hrafiv.pdf

3. Алгоритми оптимізації на графах. URL: https://ami.lnu.edu.ua/wp-content/uploads/2014/02/DO_CH2_Alhorytmy-optymizatsii-na-hrafakh.pdf

УДК 004.77

Малінін А. О., здобувач 3 курсу спеціальності 122 Комп'ютерні науки,

Волонтир Л. О., канд. техн. наук, доцент доцент кафедри інформаційних технологій

ВИСОКОТЕХНОЛОГІЧНІ ІНСТРУМЕНТИ У ВЕБДИЗАЙНІ ТА ВЕБПРОГРАМУВАННІ ЯК ДВИГУНИ ІННОВАЦІЙ У ЦИФРОВОМУ ПРОСТОРІ

Донецький національний університет імені Василя Стуса, м. Вінниця

У стрімкому розвитку цифрового світу високотехнологічні інструменти вебдизайну та програмування відіграють важливу роль у формуванні та прискоренні інноваційних аспектів інтернету. Метою цього дослідження є детальний аналіз важливих аспектів впливу та ролі інструментів у сучасній веброзробці, а також того, як ці інструменти діють як творчі каталізатори та інформують про технологічний прогрес й інноваційні підходи [1]. Основні принципи цієї ініціативи визначають високотехнологічні інструменти не лише як інструменти для вирішення завдань, а й як творчі інструменти, які сприяють і стимулюють інноваційний розвиток. Важливо підкреслити, що ці інструменти не тільки автоматизують рутинні завдання, але й слугують джерелом нових ідей, розширюють горизонти і збагачують технічні та творчі сфери веброзробки. Одним із важливих моментів цього дослідження є визначення впливу високотехнологічних інструментів на розвиток вебпростору. Вони не тільки відповідають сучасним вимогам, а й визначають тенденції розвитку. Високотехнологічні рішення у сфері вебдизайну дають змогу створювати інтерфейси, які є не тільки естетично привабливими, але й функціонально багатими, що відображають сучасні тенденції та враховують очікування користувачів. Ще один важливий аспект, який необхідно розглянути більш детально, – це вплив високотехнологічних інструментів на безпеку та стандартизацію веброзробки. Їх внесок у забезпечення кібербезпеки та створення загальних систем стандартів відіграє важливу роль у формуванні довіри користувачів до цифрових платформ [2].

Також розглядається вплив високотехнологічних інструментів на розвиток електронної комерції та динаміку взаємодії бізнесу з клієнтами. Сучасні веб-