4. Січко Т. В., Нескородєва Т. В. Методичні вказівки щодо виконання лабораторних робіт з дисципліни «Методи оптимізації та дослідження операцій» для студентів СО «Бакалавр» денної та заочної форм навчання спеціальностей 122 «Комп'ютерні науки», 113 «Прикладна математика». Вінниця: ДонНУ імені Василя Стуса. 2020, 104 с.

УДК 004.021

Поліщук В. С., здобувач 2 курсу спеціальності 122 Комп'ютерні науки,

Потапова Н. А., канд. екон. наук, доцент, доцент кафедри інформаційних технологій

АЛГОРИТМ ПОШУКУ У ГЛИБИНУ

Донецький національний університет імені Василя Стуса, м. Вінниця

Алгоритм пошуку в глибину є одним з найважливіших обчислень для графів і дерев. Він використовує методологію «глибокого» погляду, щоб почати з дослідження якомога більшої відстані в глибину графу, а потім повернутися назад. Найчастіше для актуалізації обчислень використовується рекурсивний підхід або стек.

Пошук у глибину (DFS) – це алгоритм для обходу або пошуку структур даних дерева або графу. Алгоритм починається з кореневого вузла і досліджує, наскільки це можливо, уздовж кожної гілки перед зворотним відстеженням. Додаткова пам'ять (зазвичай стек) потрібна для відстеження вузлів, знайдених на цей момент уздовж певної гілки. Це допоможе вам відстежити графік. Варіант пошуку в глибину досліджувався як стратегія вирішення лабіринту французьким математиком Шарлем-П'єром Тремо в XIX ст.

Часовий і просторовий аналіз DFS різниться залежно від області його застосування. У цьому налаштуванні часові та просторові межі такі ж, як і для пошуку в ширину, і вибір того, який із цих двох алгоритмів використовувати, залежить більше від різниці в порядку згенерованих вершин, ніж від складності. Це залежить від характеристик. Для програм DFS, пов'язаних із певним доменом, як-от пошук рішень із допомогою штучного інтелекту або вебсканування, граф, який потрібно пройти, часто занадто великий, щоб бути повністю доступним, або нескінченний (з можливістю того, що DFS ніколи не завершиться). У таких випадках пошук здійснюватиметься лише на обмеженій глибині. Через обмежені ресурси, як-от пам'ять і дисковий простір, структури даних зазвичай не використовуються для відстеження набору всіх раніше відвіданих вершин. Під час виконання пошуку на обмеженій глибині час залишається лінійним із

розширеною кількістю вершин і ребер хоча деякі вершини можуть шукатися кілька разів, тому це число може змінюватися на графі. Проте складність простору цього варіанта DFS пропорційна лише межі глибини, тому вона набагато менша, ніж простір, необхідний для пошуку на таку саму глибину з допомогою пошуку в ширину. У таких програмах DFS також підходить евристичний метод для вибору ймовірних гілок. Якщо відповідні обмеження глибини невідомі заздалегідь, ітеративний пошук у глибину неодноразово застосовує DFS зі зростаючими наборами обмежень. У режимі аналізу штучного інтелекту, коли коефіцієнт розгалуження більший за 1, кількість вузлів на рівень геометрично збільшується, тому ітераційне поглиблення зменшує час виконання до постійного значення, порівняно зі знанням точного обмеження глибини. Ви також можете використовувати DFS для збору зразків вузлів графу. Однак неповна DFS, як і неповна BFS, спрямована на вузли вищого порядку.

Пошук у глибину також можна використовувати для лінійного впорядкування вершин графу або дерева. Це можна зробити чотирма можливими способами. Попереднє впорядкування – це список вершин у тому порядку, в якому до них вперше звернувся алгоритм пошуку в глибину. Це компактний і природний спосіб описати процес пошуку, як це було зроблено в першій частині цієї статті. Попереднє впорядкування дерева виразів – це вираз у польській нотації. Заднє впорядкування – це список вершин у порядку, в якому алгоритм востаннє відвідував вершини. Задній порядок дерева виразів – це вираз у зворотному польському записі. Зворотне попереднє впорядкування – це зворотне попередньому впорядкуванню. Це означає, що список вершин знаходиться у порядку, зворотному порядку першого відвідування. Зворотне попереднє впорядкування – це не те ж саме, що і пост-впорядкування. Зворотний пост-впорядкування – це зворотне пост-впорядкування, тобто список вершин у зворотному порядку до попереднього відвідування. Зворотне пост-впорядкування не тотожне попередньому впорядкуванню. Для бінарних дерев також існує впорядкування і зворотне впорядкування.

Для реалізації алгоритму знадобиться відзначати, в яких вершинах був дослідник, а в яких – ні. Позначку робитимемо в списку Visited, де Visited [i] = True для відвіданих вершин, і Visited [i] = False для невідвіданих. Позначка «про відвідування вершини» ставиться під час входу в цю вершину. Алгоритм обходу в глибину оформимо у вигляді рекурсивної процедури DFS, де start – номер вершини, з якої запускається обхід.

У цьому алгоритмі *n* – число вершин у графі, вершини нумеруються числами від 1 до *n*, зберігає матрицю суміжності. Для запуску алгоритму, наприклад, для вершини з номером start необхідно викликати DFS. Після цього виклику всі вершини, доступні із start, будуть позначені в списку Visited.

```
procedure DFS(start:integer);
        var i: integer;
begin
                Visited[start]:=true;
                for i:=1 to n do
                        if (a[start, i]=1) and (Visited[i]=false)
                                then DFS(i)
        end;
```

Рисунок 1. Код алгоритму пошуку в глибину

Візуалізація алгоритму пошуку в глибину наведена на рис. 2.

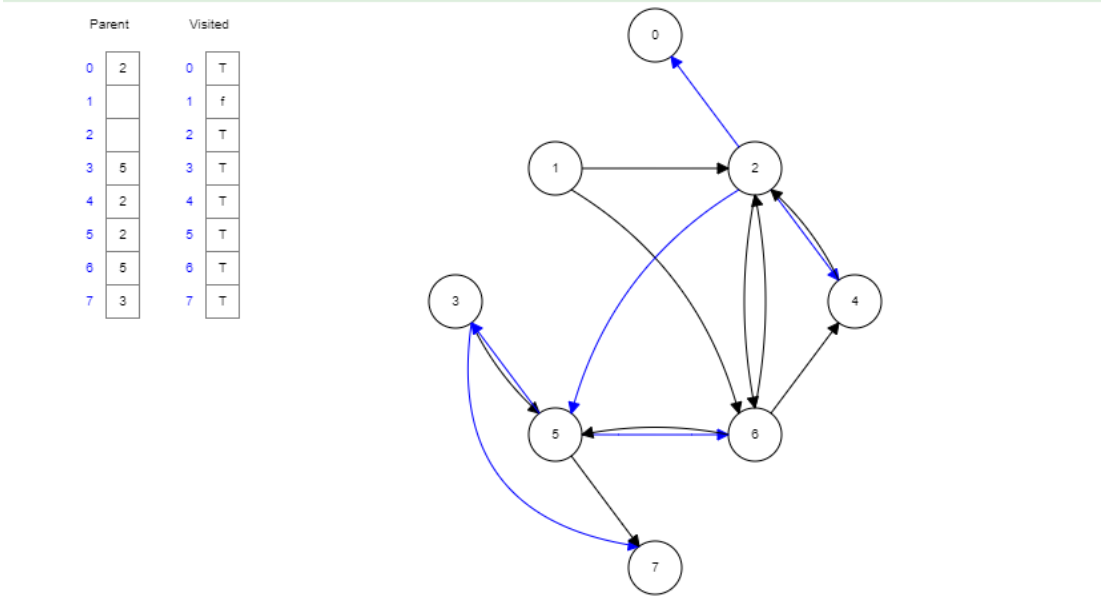

Рисунок 2. Реалізація алгоритму пошуку в глибину

Глибинний пошук у графі дає змогу систематично досліджувати структуру графу або дерева, проникаючи вглиб та перевіряючи всі можливі гілки. Це допомагає виявляти компоненти зв'язності, цикли, шляхи та інші характеристики графів. Важливо пам'ятати, що глибинний пошук не гарантує знаходження найкоротших шляхів у невагомих графах, і у деяких випадках може призвести до перевірки одних і тих самих вузлів кілька разів. Однак він є ефективним інструментом для визначення властивостей графів та дерев.

Список використаних джерел

1. Algoua. URL: https://algoua.com/algorithms/graphs/dfs/ (дата звернення: 28.11.2023).

2. Tilda. URL: https://grafi.tilda.ws/vglibiny (дата звернення: 28.11.2023).

3. Mathros. URL: https://www.mathros.net.ua/obhid-grafa-v-glybynu.html (дата звернення: 28.11.2023).

4. Biz. URL: http://www.ni.biz.ua/5/5_15/5_159663 (дата звернення: 28.11.2023).

УДК 004.021

Поліщук О. С., здобувачка 2 курсу спеціальності 122 Комп'ютерні науки,

Потапова Н. А., канд. екон. наук, доцент, доцент кафедри інформаційних технологій

СУТНІСТЬ І СКЛАДНИКИ РЕКУРЕНТНИХ АЛГОРИТМІВ

Донецький національний університет імені Василя Стуса, м. Вінниця

Рекурсія є фундаментальним поняттям у комп'ютерних науках, що використовується для вирішення різноманітних завдань. Вона базується на ідеї функції, що викликає саму себе, і відіграє важливу роль у розробці алгоритмів та програм.

Рекурсія – процес повторення чого-небудь самоподібним способом. Наприклад, вкладені віддзеркалення, утворені двома точно паралельними одне одному дзеркалами, є однією з форм нескінченної рекурсії. Цей термін має більш спеціальні значення в різних галузях знань – від лінгвістики до логіки. Кількість вкладених викликів функції або процедури називається глибиною рекурсії.

Перевага рекурсивного визначення об'єкта полягає в тому, що таке вичерпне визначення теоретично здатне описувати нескінченно велику кількість об'єктів. Із допомогою рекурсивної програми можливо описати нескінченне обчислення, причому без явних повторень частин програми [1].

Рекурсивну програму завжди можна перетворити в нерекурсивну (ітеративну, з використанням циклів), яка виконує ті ж обчислення. І навпаки, використовуючи рекурсію, будь-яке обчислення, що припускає використання циклів, можна реалізувати, не застосовуючи цикли. Рекурентне співвідношення є рекурсивною функцією з цілочисельними значеннями. Значення будь-якої такої функції можна визначити, обчислюючи усі її значення, починаючи з найменшого, використовуючи на кожному кроці раніше обраховані значення для підрахунку поточного значення. Рекурентні вирази використовуються, зокрема, для визначення складності рекурсивних обчислень [2].

Властивості рекурентних алгоритмів передбачають [2]:

1. Самовиклик (Self-calling). Рекурсія полягає у виклику функції або алгоритму самим собою, що покладено в основну ідею рекурсивного підходу.