зазвичай забезпечують високу швидкодію, масштабованість та горизонтальне розподілення даних.

Об'єктно-орієнтовані бази даних використовують об'єктно-орієнтовану модель, де дані представлені у вигляді об'єктів з методами та властивостями. Ця модель дозволяє зберігати складні дані зі зв'язками та спадковістю. Об'єктно-орієнтовані бази даних широко використовуються в областях, де моделювання об'єктів є важливим, наприклад, в розробці програмного забезпечення.

Ієрархічні бази даних використовують деревоподібну структуру для зберігання даних, де кожен запис має батьківський елемент. Ця модель часто використовується для зберігання ієрархічних даних, наприклад, в системах керування файлами або структурах організацій.

Графові бази даних засновані на графовій моделі, де даніпредставлені у вигляді вузлів (вершин) та ребер (зв'язків) між ними. Це дозволяє зберігати та оптимізовано опрацьовувати дані, які мають складні взаємозв'язки та залежності.

Кожен тип бази даних має свої переваги та обмеження і підходить для конкретних вимог та сценаріїв використання. Вибір правильного типу бази даних залежить від різних факторів, таких як природа даних, масштаб системи, потреби в швидкодії та доступності, а також вимоги до забезпечення безпеки та цілісності даних.

Список літератури:

1. *Мулеса О. Інформаційні системи та реляційні бази даних. Ужгород, 2018. 118 с.*
2. *Завадський І.О.. Основи баз даних. Київ, 2011. 193 с.*
3. *Ткачук І. Типи баз даних: особливості, відмінності та приклади URL: https://dou.ua/lenta/articles/types-of-databases/*
4. *М'якшило О.М., Загоровська Л.Г., Самсонов В.В. Організація баз даних. Теоретичні основи. Моделювання. Реалізація. НУХТ, 2019. 84 с.*

УДК 004.6

Афанасьєва Д. С., студентка 1 курсу спеціальності 122 «Комп'ютерні науки»
Науковий керівник:
Гончар В. М., асистент кафедри інформаційних технологій

АЛГОРИТМИ ПОШУКУ МІНІМАЛЬНОГО КІСТЯКОВОГО ДЕРЕВА

Донецький національний університет імені Василя Стуса, м. Вінниця

Сучасний світ інформаційних технологій неможливо уявити без застосування графів. Граф – це специфічна математична структура, яка утворюється з набору вершин, між якими існує певний зв'язок у вигляді ребер. Ребра можуть мати певну вагу, яка залежить від поставленої задачі, та орієнтацію (напрямок). Графи

застосовують для планування оптимальних маршрутів у логістиці, моделювання соціальних мереж, визначення структур баз даних, створення штучного інтелекту тощо.

Одним із важливих понять у теорії графів є кістякове дерево (з англ. «spanning tree»). Кістякове дерево у зв'язному неорієнтованому графі – це зв'язний підграф, який має в собі усі вершини початкового графа, певну підмножину його ребер та не містить циклів. Рухаючись цими ребрами можна потрапити з однієї вершини в іншу. Вони застосовуються для побудови оптимальних маршрутів у мережах зв'язків і логістиці, а також для генерування анімацій та графіки. [1]

Початковий граф може містити в собі велику кількість кістякових дерев, котрі будуть відрізнятись між собою довжиною (сумою усіх ваг ребер графа). Тому для їх ефективного застосування вводиться поняття мінімального кістякового дерева – кістякове дерево найменшої довжини. (рис. 1) [1]

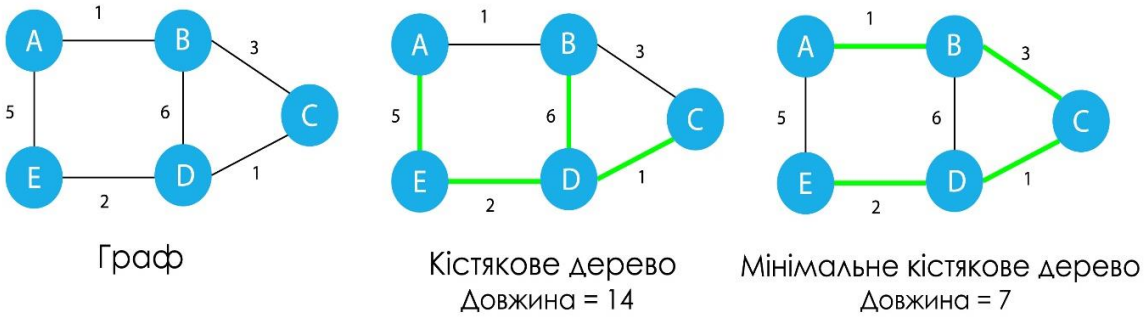

Рисунок 11. Приклад кістякового та мінімального кістякового дерев графа

Для того аби визначити мінімальне кістякове дерево у графі існують певні алгоритми: алгоритм Прима, алгоритм Крускала, алгоритм Борувки, алгоритм двох китайців. У даній науковій роботі дослідимо принцип роботи перших двох зазначених вище способів.

Алгоритм Прима використовує «жадібний» підхід для розв'язку задачі, тобто здійснює вибір найкращого варіанту з точки зору миттєвого вигляду. У випадку алгоритму Прима, під час кожного етапу розглядаються ребра, які можна приєднати до вже існуючої частини кістякового дерева, і з цієї групи обирається ребро з найменшою вагою. Початок роботи алгоритму починається з вибору будь-якої вершини графа та її додавання до множини пройдених вершин. Далі обираються всі ребра, які інциденті до початкової вершини. З-поміж множини визначених ребер потрібно відшукати те, що має найменшу довжину (вагу). На наступному кроці необхідно обрати вершину, яка інцидентна до шуканого ребра і не належить до тих вершин, які вже були обрані, і додати до пройдених вершин. Далі повторюємо попередні кроки доти, доки множина відвіданих вершин не буде рівна кількості вершин початкового графа. У результаті отримуємо мінімальне кістякове дерево початкового графа. Приклад застосування цього алгоритму до початкового графа. (рис. 2) [2]

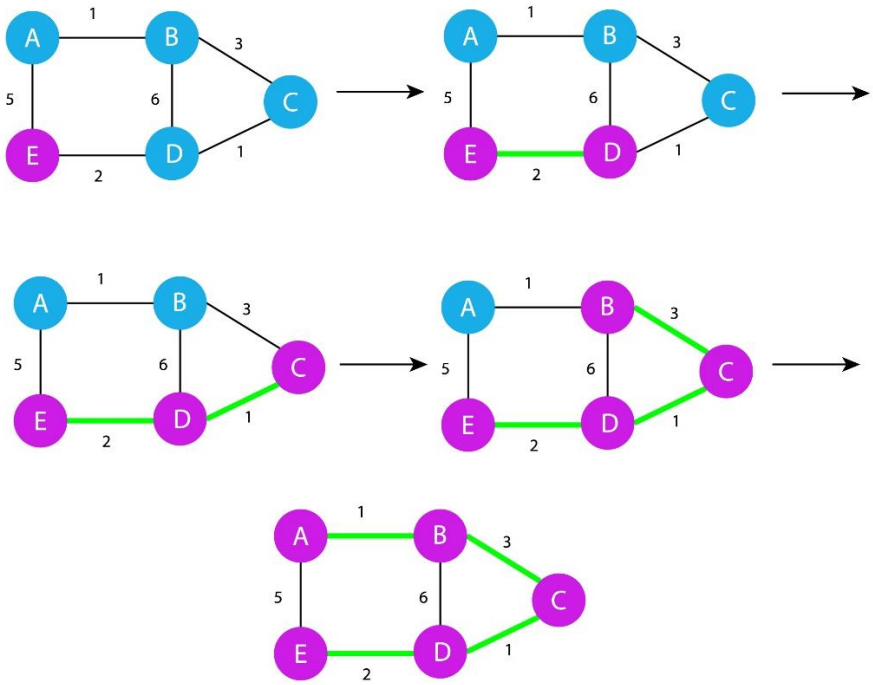

Рисунок 12. Приклад виконання алгоритму Прима

Алгоритм Крускала, як і алгоритм Прима, по своїй суті також є «жадібним», який працює з ребрами графа, але принцип дещо другий. Виконання процесу пошуку мінімального кістякового дерева не потребує обрання початкової вершини, адже цей спосіб протягом усього свого виконання паралельно створює дерева й об'єднує їх у єдине дерево наприкінці алгоритму. Спочатку всі ребра початкового графа сортуються в порядку зростання своєї ваги. З отриманого списку необхідно обрати ребро найменшої ваги та перевірити чи утворює воно цикл. Якщо ні, то ребро з інцидентними вершинами додається до дерева, якщо так – ребро відкидається і не береться до уваги в подальшій побудові. Попередні кроки повторюються до того часу, поки дерево не буде мати кількість вершин початкового графа та кількість ребер, що на одиницю менше за число вершин, адже лише тоді гарантовано можна сказати, що дерево є кістяковим і зв'язним. Застосуємо алгоритм до початкового графа. (рис. 3) [3]

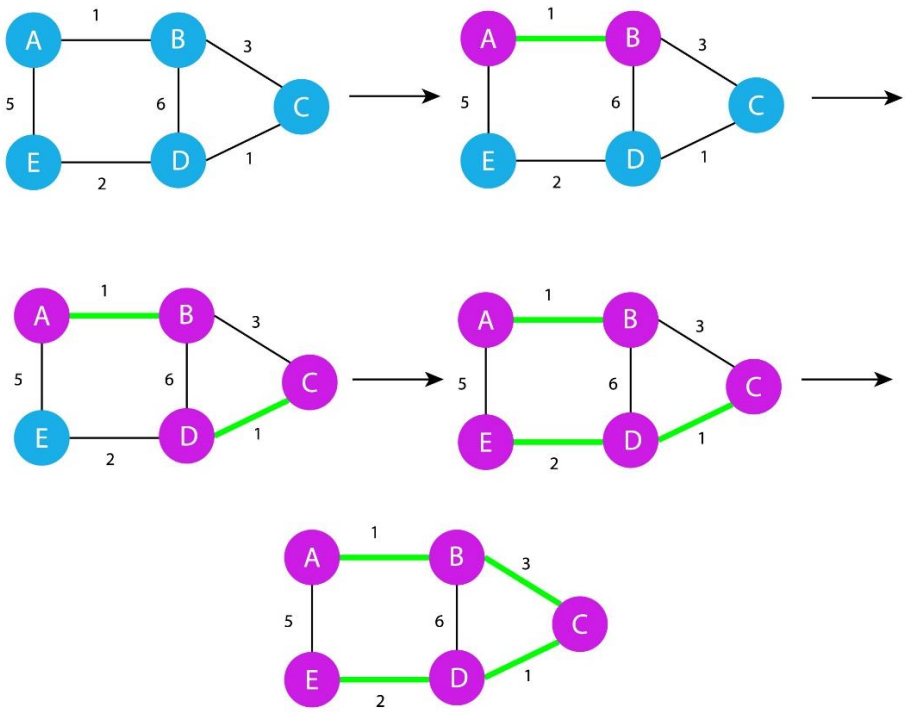

Рисунок 13. Приклад виконання алгоритму Крускала

Отже, мінімальні кістякові дерева графів мають широке застосування в реальному житті у сферах IT, логістики, математики, економіки, соціології тощо. Тому вміння швидко та точно їх знаходити особливо важливо. Для цього можна застосовувати алгоритми Прима та Крускала, які були наведені та описані в даній науковій роботі.

Список літератури

1. Minimum Spanning Tree, URL: https://www.hackerearth.com/practice/algorithms/graphs/minimum-spanning-tree/tutorial/

2. Знаходження дерева мінімальної довжини використовуючи алгоритм Прима, URL: https://www.mathros.net.ua/algorytm-pryma.html

3. Знаходження мінімального кістякового дерева графа за допомогою алгоритму, URL Крускала: https://www.mathros.net.ua/algorytm-kruskala.html

4. Kruskal's Minimum Spanning Tree (MST) Algorithm, URL: https://www.geeksforgeeks.org/kruskals-minimum-spanning-tree-algorithm-greedy-algo-2/