які ведуть до утворення мінімального кістякового дерева. Алгоритм Крускала будує паралельно кілька дерев, які зливаються в одне ціле лише в кінці. Також, варто зазначити, що цей алгоритм не обмежений у виборі ребер, тобто розглядаються всі ребра графа та надається перевага мінімальному, що не утворює цикл.

Як працює алгоритм Крускала? На початку обирається певна кількість дерев, які складаються з однієї вершини. Кожен наступний крок містить операцію сполучення двох дерев, використовуючи найменшої ваги ребро. Виконання алгоритму продовжується, доки не отримаємо єдине дерево, що містить в собі всі вершини та не містить циклів [3]. Роботу алгоритму зображено на(Рис.2)

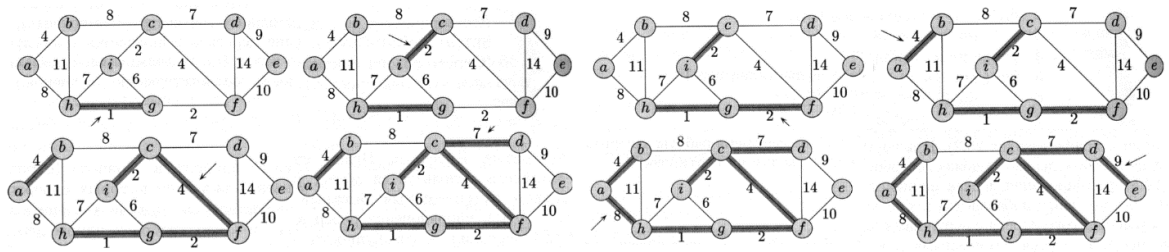

Рис. 2 Приклад роботи алгоритму Крускала

Отже, знаходження мінімального покривного дерева стає простішим, якщо використовувати алгоритми описані вище. Вони є зручними для використання, допомагають значно скоротити час та полегшити виконання самого завдання.

Список літератури

1. *Мінімальне покривне дерево та алгоритми обчислення, URL: http://surl.li/hcizc*
2. *Алгоритм Прима, URL: https://www.mathros.net.ua/algorytm-pryma.html*
3. *Алгоритм Крускала, URL:https://www.mathros.net.ua/algorytm-kruskala.html*

УДК 004.6

Козачок А.О., студентка 1 курсу
Спеціальність 122 «Комп'ютерні науки»
Науковий керівник:
Гончар В. М., асистент кафедри
інформаційних технологійф

АЛГОРИТМИ ЗНАХОДЖЕННЯ НАЙКОРОТШОГО ШЛЯХУ В ГРАФАХ ЗІ ЗВАЖЕНИМИ РЕБРАМИ З ОДНІЄЮ НЕГАТИВНОЮ.

Донецький національний університет імені Василя Стуса, м. Вінниця

Графи є важливим інструментом для моделювання різноманітних систем, від комунікаційних мереж до соціальних мереж і транспортних мереж. Однією з ключових задач при роботі з графами є знаходження найкоротшого шляху між

двома вершинами. У багатьох випадках ребра графа мають ваги, які відображають складність переміщення між вершинами. У таких випадках може виникнути ситуація, коли деякі ребра мають негативну вагу. Пошук найкоротшого шляху в таких графах є більш складним завданням, ніж у графах з додатними вагами ребер. Існує декілька алгоритмів для знаходження найкоротшого шляху в графах зі зваженими ребрами з однією негативною вагою. Далі про них.

Перший – це алгоритм Беллмана-Форда. Був заснований в 1956 році, та отримав свою назву від двох своїх засновників Річарда Беллмана та Лестера Форда. Цей алгоритм визначає найкоротші шляхи від однієї вершини графу яка є вхідною до всіх інших вершин в зваженому орієнтованому графі. Він емулює найкоротші шляхи від однієї вихідної вершини до всіх інших вершин у зваженому орграфі. Він повільніший, ніж алгоритм Дейкстри для тієї ж задачі, але він більш гнучкий, оскільки може обробляти графи з деякими вагами ребер, які є від'ємними числами. Його основна ідея полягає в тому, щоб обійти граф k разів, де k — кількість вершин у графі, і на кожному кроці оновлювати найкоротші шляхи між усіма парами вершин. Якщо найкоротший шлях до заданої вершини оновлюється на останньому кроці, це вказує на наявність у графі циклу з негативною вагою [1].

Наступний алгоритм, алгоритм Флойда-Уоршелла, також відомий як алгоритм Флойда. Він був опублікований у тому вигляді, який ми знаємо сьогодні, Робертом Флойдом у 1962 році. Однак це практично той самий алгоритм, опублікований Бернардом Роєм у 1959 році та Стівеном Варшелом у 1962 році для пошуку транзитивного замикання в графі Цей алгоритм є алгоритмом динамічного програмування, який знаходить найкоротші шляхи між усіма парами вершин у зваженому графі, включаючи графи з від'ємними вагами. Основна ідея алгоритму полягає в поступовому вдосконаленні найкоротших шляхів між вершинами шляхом порівняння відстаней між вершинами через інші вершини.

Алгоритм Флойда-Уоршелла складається з трьох вкладених циклів, які обчислюють найкоротші шляхи між усіма парами вершин. Спочатку матриця відстаней заповнюється вагами ребер графа. Потім на кожному кроці алгоритм намагатиметься оновити відстані між кожною парою вершин, перевіряючи, чи використання додаткових вершин скоротить відстань між двома вершинами. Якщо так, відстань оновлюється [2].

Також можна застосовувати алгоритм Дейкстри. Його заснував видатний голландський вчений Едсгер Вібе Дейкстра в 1952 році, в результаті чого алгоритм отримав ім'я свого винахідника. Алгоритм Дейкстри з'явився, коли Дейкстра оцінював продуктивність комп'ютера ARCMAC. Завдяки алгоритму він зміг розрахувати оптимальний шлях передачі електричного струму в найважливіших елементах кола. Цей алгоритм для пошуку найкоротшого шляху в зваженому графі з однією вихідною вершиною. Він працює для зважених графів без циклів від'ємної ваги.

На кожному кроці алгоритму Дейкстри можна оновлювати відстані до вершин і змінювати поточну вершину. Алгоритм завершується, коли відвідано всі вершини та знайдено найкоротший шлях до кожної вершини.

Алгоритм Дейкстри дозволяє знайти найкоротший шлях від однієї вершини до всіх інших вершин графа. Він досить ефективний для вирішення задач пошуку найкоротших шляхів у зважених графах [3].

Отже, проаналізувавши ці три алгоритми, ми можемо зробити висновок, що алгоритм Беллмана-Форда є найбільш універсальним і ефективним рішенням для пошуку найкоротшого шляху у зважених граничних графах з однією негативною вагою.

Список літератури

1. *Наочне пояснення алгоритму Беллмана-Форда, URL: http://surl.li/ehrmh*
2. *Floyd Warshall Algorithm, URL: https://www.geeksforgeeks.org/floyd-warshall-algorithm-dp-16/*
3. *Пошук найкоротшого шляху на графі за допомогою клітинних автоматів, URL: https://ela.kpi.ua/bitstream/123456789/52540/1/Kuibida_magistr.pdf*

УДК 004.896

Корнієнко К. К.,
студент 3 курсу спеціальності 122
«Комп'ютерні науки»
Січко Т. В., к. т. н., доцент, доцент
кафедри інформаційних технологій

СУЧАСНІ ФРЕЙМВОРКИ ТА БІБЛІОТЕКИ ДЛЯ РОЗРОБКИ ВЕБ-ЗАСТОСУНКІВ

Донецький національний університет імені Василя Стуса

Сучасні фреймворки та бібліотеки для розробки веб-застосунків є різноманітними та можуть бути використані для різних цілей. До них відносяться React, Angular і Vue.

React – це універсальний і зручний фреймворк, який характеризується гнучкістю та простотою використання. Він дозволяє швидко розробляти складні користувацькі інтерфейси за допомогою компонентів, які можна легко змінити. Крім того, React демонструє високий рівень продуктивності завдяки використанню Virtual DOM (Document Object Model) і оптимізованому алгоритму оновлення сторінок [1]. Однією з найбільш помітних переваг є велика спільнота розробників, які розробляють і підтримують широкий спектр бібліотек і компонентів, що використовуються для розробки різноманітних програм. Важливою особливістю React є його простота з точки зору тестування. Він надає можливість легкого тестування окремих компонентів та їх функцій, що робить