```
NumberAxis x = new NumberAxis();
x.setLabel("Аргумент");
NumberAxis y = new NumberAxis();
y.setLabel("Значення функції");
LineChart lch = new LineChart(x, y);
lch.setTitle("Line Chart");
XYChart.Series sr = new XYChart.Series();
sr.getData().add(new XYChart.Data(1.2, 1));
sr.getData().add(new XYChart.Data(2.1, -1));
sr.getData().add(new XYChart.Data(2.75, 1));
numberLineChart.getData().add(sr);
Scene scene = new Scene(numberLineChart, 600,600);
primaryStage.setScene(scene);
primaryStage.show();
```

Отже, JavaFX – бібліотека, яка надає широкий спектр можливостей розробникам для створення обширних застосунків, та наповнення їх різноманітними графічними елементами. Бібліотека підтримує CCS і FXML, що дає змогу налаштувати кожен елемент до мілких деталей.

Список літератури

1. *Graphical user interface. Wikipedia. / [Електронний ресурс] Режим доступу: https://en.wikipedia.org/wiki/Graphical_user_interface*
2. *Основи JavaFX / [Електронний ресурс] Режим доступу: https://schoolboyprog10.blogspot.com/p/16-javafx_12.html*
3. *Зелінська О.В., Волонтир Л.О., Денисюк В.О., Комп'ютерна графіка. Методичні вказівки для для проведення практичних занять та самостійної роботи для здобувачів вищої освіти першого (бакалаврського) рівня галузі знань 12 «Інформаційні технології» спеціальності 122 «Комп'ютерні науки» денної та заочної. Вінниця, 2020. URL: http://repository.vsau.org/card.php?lang=en&id=26657*

УДК 004.4

Юстименко Є. А. студент
Труханська В. О. студент
Мартьянова Т.А. старший викладач
кафедри інформаційних технологій

ДОСЛІДЖЕННЯ РОБОТИ РІЗНИХ АЛГОРИТМІВ ЗНАХОДЖЕННЯ ШЛЯХУ

Донецький національний університет імені Василя Стуса, м. Вінниця

В наш час стрімкий розвиток технологій різних напрямків привів до того, що для дослідження даних в більшості з них, ми маємо побудувати математичну модель. Однією з таких моделей є граф, яку можна застосувати до широкого спектру даних. Такі моделі, на основі графів, використовуються в комп'ютерних мережах, проектуванні ЕОМ, файлових системах, алгоритмах і структурах даних, тощо. Найчастіше на графі виконують алгоритм пошуку оптимального

шляху. Шлях між двома точками графу може означати найшвидший, найдешевший, найкоротший шлях між двома об’єктами моделі.

Першим дослідженням з теорії графів є стаття Ейлера, випущена у 1736 році. Першим алгоритмом пошуку найкоротшого шляху є алгоритм Дейкстри, створений у 1956 році.[1]

Алгоритм знаходження найкоротшого шляху – пошук шляху, який також називають ланцюгом, між двома вершинами графу. Основна задача алгоритму – знайти шлях з мінімальною загальною вагою ребер графу, які проходить алгоритм при досяганні кінцевої вершини. Алгоритм починається з одної вершини, переходить до сусідніх вершин, порівнюючи вагу ребер, поки не дійде до кінцевої вершини.

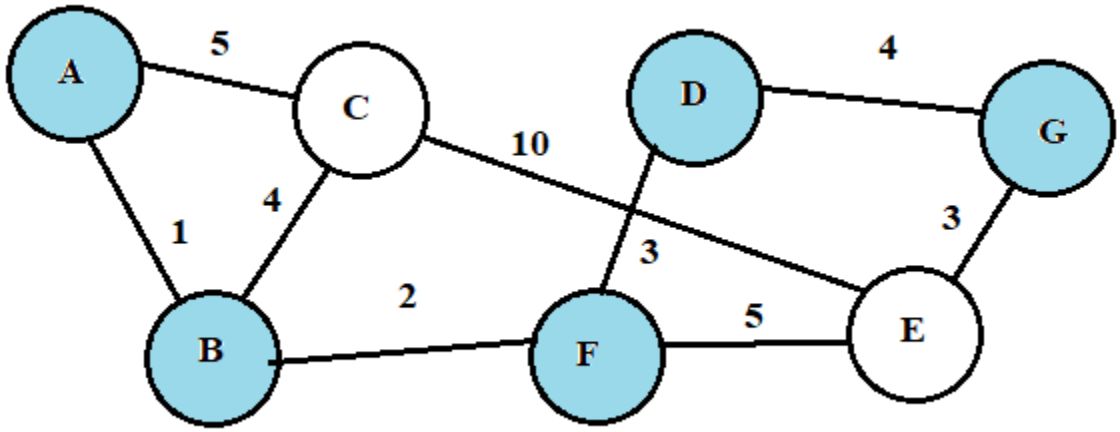

Рис 1. Приклад роботи алгоритму пошуку шляху

На рис. 1 показана робота алгоритму пошуку найкоротшого шляху між вершинами A і G, які розміщені в орієнтованому графі з вагами. Вага шляху – 10 одиниць.

Розглянемо основні алгоритми пошуку шляху:

Алгоритм Дейкстри. Найвідоміший і найпоширеніший алгоритм, який знаходить коротший маршрут від однієї вершини, до всіх інших. Алгоритм не буде працювати, якщо граф матиме ребра з вагою менше нуля. Цей алгоритм є одним із найпростіших. У випадку з графом, де кількість вершин у графі може сягати кількох тисяч використання даного алгоритму не буде оптимальним вибором. Якщо застосувати алгоритм до графа з від’ємними ребрами, алгоритм покаже неправильний(збитковий) маршрут.

Обчислювальна складність алгоритму - $O(n^2 + m)$, де n – вершини, m – ребра.[2]

Алгоритм Беллмана-Форда. Алгоритм пошуку найкоротшого шляху у зваженому графі, який знаходить шлях від однієї вершини до решти. На відміну від алгоритму Дейкстри, алгоритм Беллмана Форда допускає наявність у графі

ребер з негативною вагою. У випадку з графом, з великою кількістю вершин, алгоритм використає повний перебір всіх вершин графа, що призведе до великої втрати часу та займе більший обсяг пам'яті обчислювальної машини.

Обчислювальна складність алгоритму – O(n * m) , де n – вершини, m – ребра.[2]

Алгоритм пошуку A *. Даний алгоритм по суті є розширенням алгоритму Дейкстри, але досягає вищої продуктивності за рахунок введення у роботу алгоритму евристичної функції. Алгоритм A* є алгоритмом пошуку за кращим збігом на графі, який знаходить маршрут з найменшою вагою ребер від початкової вершини до кінцевої. Даний алгоритм крок за кроком переглядає всі шляхи, що ведуть від початкової вершини в кінцеву, поки не знайде мінімальний. Порядок обходу вершин визначається евристичною функцією «відстань + вартість» та евристичною оцінкою відстані від розглянутої вершини до кінцевої.

Обчислювальна складність – O(log h(x)), де h – евристична функція[2]

Отже, з представлених алгоритмів найкращу продуктивність показує A*, але при великій кількості вершин, A*, як і інші представлені алгоритми, покаже поганий результат по витратам часу і об’єму використаної пам’яті ЕОМ.

Алгоритми пошуку найкоротшого шляху – дуже важливий розділ в теорії графів. Такі алгоритми можна використовувати для обробки або дослідження даних в різних галузях, які виходять за межі інформаційних технологій.

Список літератури

1. *Dijkstra Algorithm. Wikipedia. / [Електронний ресурс]*
   Режим доступу: https://en.wikipedia.org/wiki/Dijkstra%27s_algorithm
2. *Pathfinding. Wikipedia. / [Електронний ресурс]*
   Режим доступу: https://en.wikipedia.org/wiki/Pathfinding
3. *Мультиагентна система маршрутизації на основі алгоритмів пошуку найкоротшого шляху в графі/ [Електронний ресурс]*
   Режим доступу: https://ela.kpi.ua/bitstream/123456789/24087/1/Tishkov_magistr.pdf