Національний технічний університет України

«Київський політехнічний інститут імені Ігоря Сікорського»

Факультет інформатики та обчислювальної техніки

Кафедра автоматики та управління в технічних системах

**Лабораторна робота №8**

З дисципліни «Алгоритми та структури даних-2»

Тема: «Дослідження операцій на графах»

Виконали: Перевірив:

студенти групи ІТ-03 Смолій В. В.

Митєв А. Ю.

Яремчук Д. В.

Шевчук Д. Д.

Дата здачі: 27.05.21

Захищено з балом \_\_\_\_\_\_\_\_

Київ 2021

**Тема**: «Дослідження операцій на графах»

**Мета**: дослідити операції на графах, їх трудомісткість та ефективність

**Завдання:**

Реалізувати відповідні структури даних для представлення *ациклічних нерегулярних графів зі ступенем до 4* та тестовий граф (не менш ніж 20 вершин та 30 ребер), визначити показники продуктивності розроблених процедур з реалізації операцій.

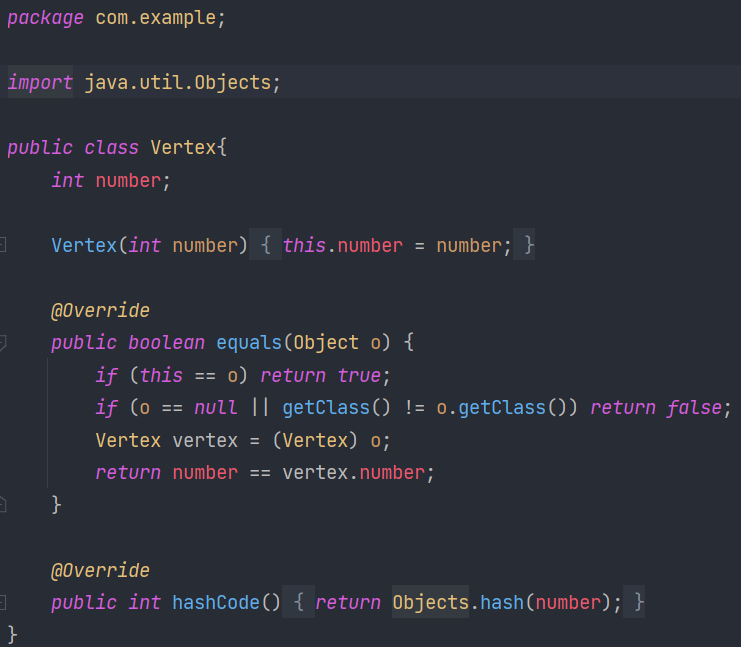
Відповідно реалізувати операції:

* додавання та видалення вершини;
* додавання та видалення ребра;
* отримання базових характеристик графу та визначеної вершини;
* пошуку визначеної вершини;
* обхід графу;
* визначення найменшого предку двох визначених вершин;
* пошук відстані між двома визначеними вершинами.

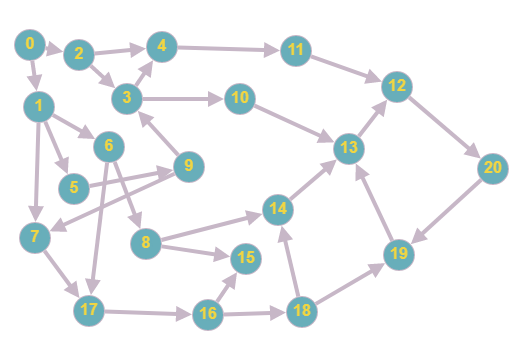
Варіанти завдань

|  |  |  |
| --- | --- | --- |
| Варіант | Метод обходу | Тип графу |
| 1 | *Inorder* | Орієнтований |
| 2 | *Preorder* | Неорієнтований |
| 3 | *Postorder* | Орієнтований |
| 4 | *Inorder* | Неорієнтований |
| **5** | ***Preorder*** | **Орієнтований** |
| 6 | *Postorder* | Неорієнтований |

**Хід роботи**

1. Ми вибрали список суміжності для реалізації графу. Це подання порівняно важко створити і воно менш ефективне для запитів. Однак він пропонує кращу просторову ефективність та його доволі легше зрозуміти. Список суміжності – це асоціативний масив, ключем якого є вершина, а значенням – список вершин, до яких є ребро від ключа.
2. Наш варіант передбачає реалізацію орієнтованого графа із preorder варіантом обходження графу. Але оскільки preorder, postorder та inorder traversal використовується лише для дерев, то ми не можемо його тут реалізувати. Є лише два методи обходження графу: depth first search, breadth first search. Хоча для обходу необхідний неорієнтований граф, ми все таки реалізували їх, адже обхід графу – це одна із найголовніших операцій щодо графа. Також оскільки наш граф орієнтований, то ми не можемо реалізувати метод пошуку найменшого спільного предка для двох вершин та пошуку відстані між ними, бо для цього нам треба повертатися з вершини до її батька, а це неможливо зробити в орієнтованому графі.
3. Для зручності ми створили клас Vertex для репрезентації вершини.

Для того щоб можна було працювати з вершинами в списку суміжності, ми перевизначаємо методи equals, hashcode. Для універсальнішого обходу ми винесли їх методи в окремий клас GraphTraversal, який також містить метод для перетворення орієнтованого графа в неорієнтований. Оскільки мова програмування Java при присвоєнні використовує посилання, а не значення, то після обходів ми створюємо новий такий самий граф.

1. Тестовий граф (21 вершина, 28 ребер):
2. Результат роботи