Слайд 4:

Відомо декілька алгоритмів пошуку циклів швидко та з малим використанням пам'яті. В роботі розглянуто Алгоритм «черепахи і зайця» Роберта Флойда (переміщає два покажчики з різною швидкістю через послідовність значень, поки вони обидва не вказують на однакові значення.) , алгоритм Брента (заснований на ідеї експоненціального пошуку). І алгоритм Флойда, і Брента використовують лише постійну кількість комірок пам'яті і приймають ряд оцінок функції, пропорційну відстані від початку послідовності до першого повторення.

Алгоритм Р. В. Госпера знаходить період, але не є початковою точкою першого циклу. Його головна особливість полягає в тому, що він ніколи не створює резервного копіювання для переоцінки функції генератора, і є економічним як в просторі, так і в часі.

Проте в реалізації використано один із основних методів обходу вершин графа, який часто використовують для перевірки зв'язності, пошуку циклів і компонент сильної зв'язності та для топологічного сортування – обхід в глибину, або пошук в глибину.

Слайд 5:

Загальна ідея алгоритму полягає в наступному: для кожної не пройденої вершини необхідно знайти всі не пройдені суміжні вершини і повторити пошук для них.

Зачасту, простої інформації «були/не були в вершині» недостатньо для конкретних цілей. Тому в процесі алгоритму вершинам задаються деякі кольори:

- якщо вершина біла – ми в ній ще не були;

- сіра – вершина проходиться в даний момент;

- чорна – вершина, яку ми вже пройшли.

Цикл існує, якщо пошук в глибину знаходить ребро, кінець якого пофарбований в сірий колір.

То ж на слайді зображено сам алгоритм пошуку циклів за допомогою методу пошуку, обходу в глибину.

Перейдімо до самої програмної реалізації, перед початком роботи було отримано дані через об'єктно-реляційну систему керувань базами даних PostgreSQL. Програмна реалізація написана на мові програмування PHP.

Слайд 6:

Через те, що в графі є понад 70 тисяч ребер, інтерпретатор php 7.4.12 разом з сервером Аpache 2.4.46 не в стані опрацювати настільки великий стек рекурсії, навіть при вимкненні обмеження пам'яті. Тому за вхідний массив було прийнято таблицю graph, яка зображена на слайді. Це полегшує виконання алгоритму, адже швидкість обробки запитів PostgreSQL, як і інших баз даних, є на порядок вища, ніж обробка масива з такою самою кількістю даних.

В таблиці містяться наступні поля: id, source, target, distance, color. Поле id містить в собі ідентифікатор дуги. Поле source – ідентифікатор вершини, з якої виходить дуга. Поле target – ідентифікатор вершини, в яку входить дуга. Поле distance – довжина даного ребра. Color містить колір вершини.

Далі сам алгоритм реалізовано, і показано на наступному слайді.

Слайд 7:

Алгоритм пошуку всіх циклів – це перебір функції, яка наведена нижче для кожної вершини.

До роботи додається посилання на веб-сервіс для спільної розробки програмного забезпечення GitHub, на якому було розміщено програмну реалізацію даного алгоритму.

Слайд 8:

Для того аби побачити результат роботи нам достатньо обмежитись лише однією точкою, та вивести всі цикли для обраного аеропорту. Результат даної реалізації виведено в файл circles.js [12], де цикли представлені у вигляді географічних координат точок (рис. 27). Data(n) – номер виведеного циклу, в дужках записані порядок дуг циклу, географічні координати вхідної точки; географічні точки вихідної точки, і так поки не вернеться цикл в початкову точку.

Висновок:

Оскільки я вже говорила про кількість отриманих даних, можу сказати що за час виконання даної роботи прийшла до висновків, що пхп не для таких великих розрахунків, в цьому випадку треба було б обрати пітон, там ж є готові інструменти для пошуку циклів в графах.