Санкт-Петербургский Политехнический Университет Петра Великого
Физико-механический институт
Кафедра Прикладной математики и Информатики

Отчет по Лабораторной работе №4

Тема: Гамильтонов цикл

Предмет: Дискретная математика
Студент: Егоркин Станислав Дмитриевич
Группа: 5030102/20202
Преподаватель: Нахатович Михаил Алексеевич

1. **Требования:**

На вход программе подаётся граф. Требуется найти в графе и вывести любой гамильтонов цикл, если таковой имеется. В противном случае указать, что цикл не был найден.

Программа реализована на языке C++.

2. **Идея алгоритма:**

Гамильтонов цикл - замкнутый путь, который проходит через каждую вершину данного графа ровно по одному разу. Так как цикл проходит через каждую вершину ровно по 1 разу, значит можно выбрать любую вершину из графа, с которой будет начинаться цикл. У этой вершины будет ребро со следующей вершиной. Эта вершина будет иметь ребро с другой вершиной и так далее, до тех пор, пока не вернемся в начальную вершину. Если не этого не происходит, то у вершин с несколькими ребрами выбирается другое ребро.

3. **Алгоритмы программы:**

Цикл проходит через каждую вершину ровно по 1 разу, значит можно выбрать любую вершину из графа, с которой будет начинаться цикл. У этой вершины будет ребро со следующей вершиной. Эта вершина будет иметь ребро с другой вершиной и так далее, до тех пор, пока не вернемся в начальную вершину. Если не этого не происходит, то у вершин с несколькими ребрами выбирается другое ребро.

4. **Описание работы кода:**

1) Основная функция (findHamiltonianCycle):

Выбираем начальную вершину 0. От неё будем искать цикл. Если для вершины 0 не существует цикла, значит гамильтонова цикла в графе нет.

2) Поиск пути (hamCycleDFS):

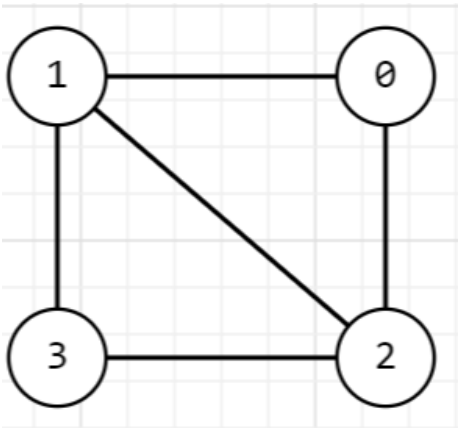
Рекурсивно добавляет вершину в цикл. Если после рекурсии путь не вернулся в начальную вершину – удаляется последняя вершина из пути.

3) Проверка добавления вершину в текущий цикл (isSafe):

Функция проверяет наличие ребра и то, что данная вершина не была ранее добавлена в цикл.

5. **Пример работы алгоритма:**

Рассмотрим пример графа, записанный в виде матрицы смежности:



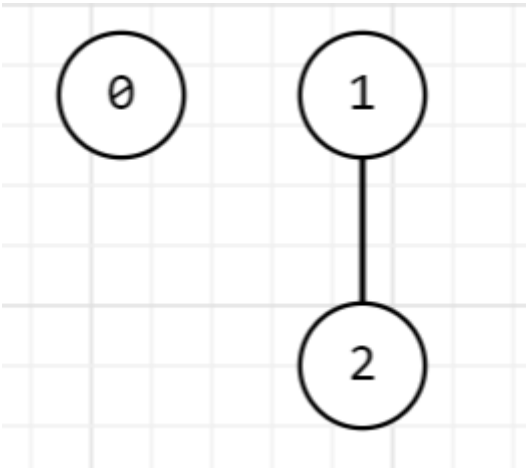
0	1	1	0
1	0	1	1
1	1	0	1
0	1	1	0

1) Начинаем с вершины 0.
2) Проходим по ребрам и получаем путь: 0-1-2-3
3) Путь не вернулся обратно в вершину 0. Удаляем вершины из пути, пока не будет найдено другое ребро: 0-1-2
4) Идем по другому ребру : 0-1-2-0
5) Вершина 3 не была посещена. Удаляем вершины из пути: 0-1
6) Идем по другому ребру: 0-1-3-2-0.
7) Получен гамильтонов цикл. 0-1-3-2-0.

6. Тесты:

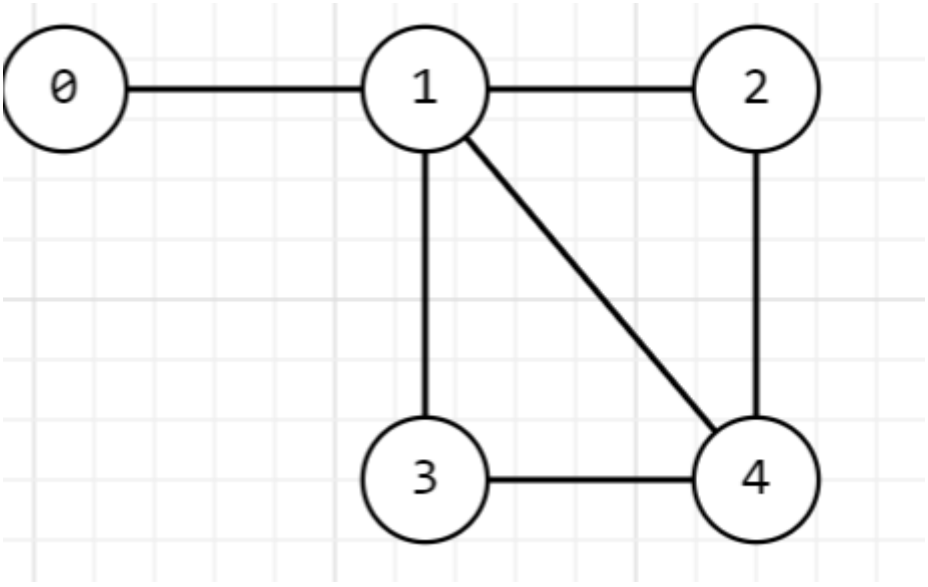
1) Пример из отчета – Гамильтонов цикл найден: 0 1 3 2 0
2) Пример с вершиной без ребер – Гамильтонов цикл не найден**.**

0	0	0
0	0	1
0	1	0



3) Пример без гамильтонова цикла –

0	1	0	0	0
1	0	1	1	1
0	1	0	0	1
0	1	0	0	1
0	1	1	1	0



Гамильтонов цикл не найден**.**

7. **Формат данных**
   Данные указываются в виде матрицы смежности, в которой значение 1 в колонках и столбцах показывают наличие ребра между вершинами под номерами колон и столбцов соответственно и наоборот.
8. **Формат выходных данных**
   Алгоритм выводит данные в формате, если цикл найден, сообщение «Гамильтонов цикл найден:» и все посещенные вершины, начиная с вершины под номером 0 (соответствует первой строке\столбцу матрицы смежности) и продолжая номерами всех остальных

посещенный в цикле вершин до начальной вершины под номером 0, через пропуск. Если цикла не найден, то выводится сообщение «Гамильтонов цикл не найден.»

9. **Представление графа**
   В программе граф задаётся в виде матрицы смежности для удобства пользования в функции поиска пути. (проверка наличия ребра в графе, представленным в виде матрицы смежности имеет сложность O(1)).
10. **Область применимости**
    Количество вершин задаётся в формате неотрицательных целочисленных чисел в пределах до 2147483647, матрица смежности в формате квадратной матрицы размером количества вершин и заполняются данными 0 и 1.
11. **Сложность алгоритма**
    Проверка на наличии ребре – O(1)
    Проверка уникальности вершины в пути (в худшем случаи) – O(n)
    Общее количество перестановок – O((n-1)!)
    Для каждой перестановки проверяется уникальности и наличие ребра. Значит итоговая сложность алгоритма – O(n*(n-1)!)
12. **Вывод:**

Данный алгоритм найти гамильтонов цикл, если он есть в графе. Сложность алгоритма – O(n*(n-1)!).

Источники информации:

1. https://www.geeksforgeeks.org/hamiltonian-cycle/
2. https://ru.wikipedia.org/wiki/Гамильтонов_граф