**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра ИС**

Курсовая РАБОТА

**по дисциплине «Алгоритмы и структуры данных»**

Тема: “Алгоритм поиска минимального остова на основе алгоритма Краскала”

|  |  |  |
| --- | --- | --- |
| Студент гр. 2372 |  | Васильев Ю.А. |
| Преподаватель |  | Пелевин М.С. |

Санкт-Петербург

2023

**ЗАДАНИЕ**

**на курсовую работу**

|  |  |  |
| --- | --- | --- |
| Студент Васильев Ю.А. | | |
| Группа 2372 | | |
| Тема работы: Алгоритм поиска минимального остова на основе алгоритма Краскала | | |
|  | | |
|  | | |
|  | | |
|  | | |
|  | | |
|  | | |
| Студент |  | Васильев Ю.А. |
| Преподаватель |  | Пелевин М.С. |

**содержание**

|  |  |  |
| --- | --- | --- |
|  | Введение | 4 |
| 1. | Граф | 5 |
| 1.1. | Что такое граф и как его представлять | 5 |
| 1.1.1 | Матрица смежности | 5 |
| 1.1.2 | Матрица инцидентности | 6 |
| 1.1.3 | Список смежности | 6 |
| 1.2. | Обходы графов | 7 |
| 1.2.1 | Обход в глубину (DFS) | 7 |
| 1.2.2 | Обход в ширину (BFS) | 7 |
| 2. | Система непересекающихся множеств | 9 |
| 3. | Минимальное остовое дерево | 10 |
| 3.1. | Что такое МОД и как его строить | 10 |
| 3.2. | Алгоритм Краскала (Крускала) | 10 |
|  | Заключение | 12 |
|  |  |  |
|  |  |  |

**введение**

**Цель работы:**

Реализовать алгоритм [поиска минимального остова](https://ru.wikipedia.org/wiki/Минимальное_остовное_дерево) на основе [алгоритма Краскала](https://ru.wikipedia.org/wiki/Алгоритм_Краскала) (Крускала).

На вход подаётся текстовый файл, содержащий матрицу смежности графа, где первая строка – названия вершин графов, а остальные строки – квадратная матрица, представляющая собой вес ребра между вершинами. В матрице значение 0 стоит, если ребра между вершинами нет, и положительное число, которое соответствует весу, когда ребро между вершинами существует.

Необходимо из найти минимальное остовое дерево из данного графа.

Результат в виде отсортированных по имени пар и их суммарный вес. Максимальный размер входных данных: 50 вершин. Вершины могут быть заданы любой текстовой последовательностью без пробелов. Вес ребра ограничен интервалом от 1 до 1023 включительно.

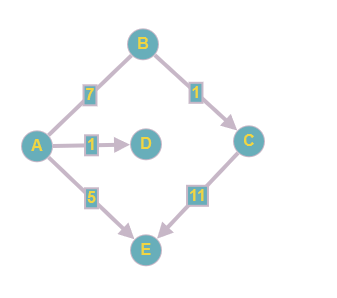
**Методы решения:**

Из представленного графа, ребра выписываются в отсортированном порядке, после чего они объединяются с помощью системы непересекающихся множеств (СНМ).

**1. Граф**

**1.1. Что такое граф и как его представить**

Граф – это абстрактная математическая структура, представляющая собой множество вершин (или узлов), соединенных рёбрами (или дугами). Графы используются для моделирования различных отношений и взаимосвязей в различных областях, таких как компьютерные науки, социология, транспортная логистика и многие другие.



*Рис.1 Граф с пятью вершинами*

Существует несколько способов представления графа в программировании, включая матрицы смежности, матрицы инцидентности и списки смежности.

Рассмотрим возможности представления графов:

**1.1.1** **Матрица смежности:**

Матрица смежности – это квадратная матрица, где строки и столбцы соответствуют вершинам графа, а элемент (i, j) равен весу ребра, если между вершинами i и j есть ребро, и 0 в противном случае, если ребра нет.

Пример матрицы смежности, представляющей граф на картинке выше:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | ***A*** | ***B*** | ***C*** | ***D*** | ***E*** |
| **A** | 0 | 7 | 0 | 1 | 5 |
| **B** | 7 | 0 | 1 | 0 | 0 |
| **C** | 0 | 0 | 0 | 0 | 11 |
| **D** | 0 | 0 | 0 | 0 | 0 |
| **E** | 0 | 0 | 0 | 0 | 0 |

*Табл.1 Пример матрицы смежности*

**1.1.2 Матрица инцидентности**

Матрица инцидентности имеет вид представления графа в виде матрицы, в которой каждый столбец задаёт отдельное ребро. Строки матрицы при этом задают вершины. Положительное число в столбце задаёт вершину, из которой выходит ребро, а отрицательное - в которое входит. Если оба числа положительные, то ребро является двусторонним.

Пример матрицы инцидентности, представляющей граф на картинке выше:

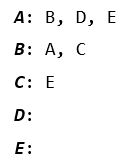
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | ***1*** | ***2*** | ***3*** | ***4*** | ***5*** |
| **A** | 1 | 5 | 7 | 0 | 0 |
| **B** | 0 | 0 | 7 | 1 | 0 |
| **C** | 0 | 0 | 0 | -1 | 11 |
| **D** | -1 | 0 | 0 | 0 | 0 |
| **E** | 0 | -5 | 0 | 0 | -11 |

*Табл.2 Пример матрицы инцидентности*

**1.1.3 Списки смежности:**

В списке смежности каждой вершине сопоставляется список соседних с ней вершин.

Пример списка смежности, представляющей граф на картинке выше:

******

*Рис.2 Пример списка смежности*

**1.2. Обходы графов**

Алгоритмы обхода графов в глубину (Depth-First Search, DFS) и в ширину (Breadth-First Search, BFS) используются для поиска и обхода вершин в графе. Оба алгоритма помогают определить связность графа, находить пути, обнаруживать циклы и выполнять другие операции.

Оба алгоритма гарантируют, что каждая вершина будет посещена ровно один раз. Выбор между DFS и BFS зависит от конкретных требований задачи. DFS обычно используется для поиска в глубину, например, при топологической сортировке, а BFS - для поиска в ширину, такого как нахождение кратчайшего пути между двумя вершинами.

**1.2.1 Обход в глубину (DFS)**

Алгоритм работает так

1. Выбирается начальная вершина.

2. Вершина отмечается как посещённая.

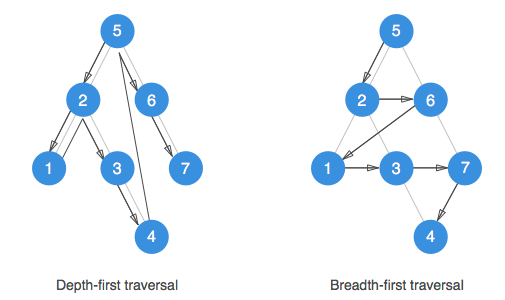
3. Рекурсивное повторение этого же процесса для всех соседних вершин, которые ещё не были посещены.

**1.2.2 Обход в ширину (BFS)**

1.Выбирается начальная вершина и она же помещается в очередь.

2. Пока очередь не пуста, извлекается вершина из начала очереди.

3. Вершина отмечается как посещённая, после чего все её не посещённые соседние вершины добавляются в конец очереди.



*Рис.3 Иллюстрирование работы алгоритмов DFT и BFT*

**2. Система непересекающихся множеств**

Система непересекающихся множеств – это структура данных, которая управляет множеством элементов, разбитых на несколько непересекающихся подмножеств. Она предоставляет около-константное время выполнения операций по добавлению новых множеств, слиянию существующих множеств и определению, относятся ли элементы к одному и тому же множеству.

Основные методы системы непересекающихся множеств:

- make\_set(x): создает новое множество, содержащее элемент x. Каждый элемент изначально является представителем своего собственного множества.

- find(x): возвращает представителя множества, к которому принадлежит элемент x. Этот метод позволяет определить, принадлежат ли два элемента одному множеству.

- union(x, y): объединяет два множества, к которым принадлежат элементы x и y. Этот метод позволяет объединять множества и создавать более крупные множества, объединяя их представителей.

**3. Минимальное остовое дерево**

**3.1. Что такое МОД и как его строить**

Минимальное остовое дерево (Minimum Spanning Tree, MST) в графе — это подграф, содержащий все вершины исходного графа и обладающий следующими свойствами:

1. Связность: Все вершины связаны между собой рёбрами. То есть, минимальное остовое дерево должно быть связным.
2. Ацикличность: Все рёбра остового дерева не образуют циклы.
3. Минимальный вес: Сумма весов рёбер остового дерева минимальна. Каждый раз, когда добавляется ребро, выбирается ребро с минимальным весом из доступных.

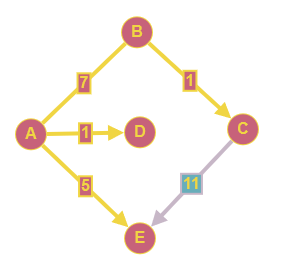
Различают два алгоритма поиска такого дерева, такие как Алгоритм Прима и Алгоритм Краскала (в этой работе разобран именно он).

**3.2. Алгоритм Краскала (Крускала)**

Механизм, по которому работает данный алгоритм, очень прост. На входе имеется пустой подграф, который и будем достраивать до потенциального минимального остового дерева.

1. Вначале мы производим сортировку рёбер по неубыванию по их весам.
2. Добавляем i-ое ребро в наш подграф только в том случае, если данное ребро соединяет две разные компоненты связности, одним из которых является наш подграф. То есть, на каждом шаге добавляется минимальное по весу ребро, один конец которого содержится в нашем подграфе, а другой - еще нет. (В работе на этом этапе используется СНМ)
3. Алгоритм завершит свою работу после того, как множество вершин нашего подграфа совпадет с множеством вершин исходного графа.

Пример МОД, на основании графа на картинке выше:

****

*Рис.4 Минимальное остовое дерево (отмечено жёлтым)*

**заключение**

В результате выполненной работы было успешно построено минимальное остовое дерево на основе алгоритма Краскала с использованием системы непересекающихся множеств (СНМ). Алгоритм Краскала, в сочетании с СНМ, обеспечил эффективное нахождение минимального остового дерева во взвешенном графе, сохраняя свойства связности и ацикличности.

Применение СНМ позволило эффективно управлять структурой множеств, обеспечивая оптимальное объединение вершин и предотвращение образования циклов.