МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«МАГНИТОГОРСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
ИМ. Г. И. НОСОВА»
(ФГБОУ ВО «МГТУ ИМ. Г.И. НОСОВА»)

Кафедра вычислительной техники и программирования

КУРСОВАЯ РАБОТА

по дисциплине «Алгоритмы и теория сложности»

на тему: «Оптимальное размещение: центра на нагруженном неориентированном взвешенном графе»

Исполнитель: Варламов М.Н. студент 3 курса, группа АВб-19-1

Руководитель: Файнштейн С.И, старший преподаватель каф. ВТиП

Работа допущена к защите «____» __________2021 г. ______________

Работа защищена «____» ___________ 2021 г. с оценкой ___________

Магнитогорск, 2021

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«МАГНИТОГОРСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
ИМ. Г.И. НОСОВА»
(ФГБОУ ВО «МГТУ ИМ. Г.И. НОСОВА»)

Кафедра вычислительной техники и программирования

ЗАДАНИЕ НА КУРСОВУЮ РАБОТУ

Тема: «Оптимальное размещение: центра на нагруженном взвешенном неориентированном графе»

Обучающемуся Варламову Максиму Николаевичу

Исходные данные: <u>необходимо реализовать алгоритм нахождения центра на нагруженном взвешенном неориентированном графе (рёбрам приписаны положительные длины, вершинам – неотрицательные веса)</u>

Срок сдачи: «_____» __________ 2021 г.

Руководитель: __________________ /С.И. Файнштейн/

Задание получил: __________________ /М.Н. Варламов/

Магнитогорск, 2021

Содержание

1 ТЕОРЕТИЧЕСКОЕ ОБОСНОВАНИЕ РЕАЛИЗАЦИИ АЛГОРИТМА ПОИСКА ЦЕНТРА

1.1 Задача размещения станции пожаротушения

Необходимо разместить станцию пожаротушения так, чтобы расстояние от станции до самого удаленного жилого дома было наименьшим. Также необходимо учитывать вероятность возгорания каждого дома.

1.2 Математическая постановка задачи

Дан нагруженный взвешенный неориентированный граф G = $<V, E>$. Рёбрам приписаны положительные длины, вершинам – неотрицательные веса.

Требуется разместить в одном из районов станцию скорой помощи так, чтобы расстояние от станции до самого удалённого района было оптимально.

Результатом работы алгоритма является оптимально размещенный центр графа.

1.3 Описание алгоритма поиска центра графа

На рисунке 1.1 представлен модельный граф.

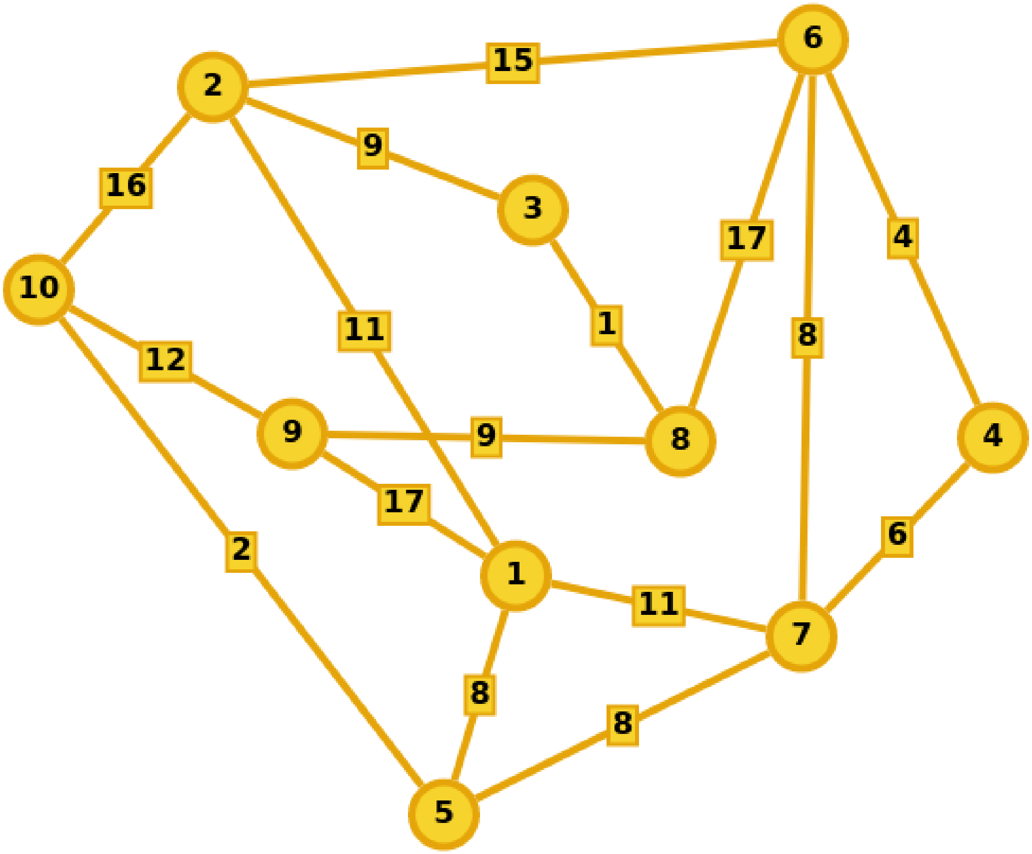

Рисунок 1.1 — Модельный граф

Данном графу соответствует матрица неотрицательных весов, представленная в таблице 1.1.

Таблица 1.1 — Матрица неотрицательных весов графа

0	11	0	0	8	∞	11	∞	17	∞
11	0	9	∞	∞	15	∞	∞	∞	16
∞	9	0	∞	∞	∞	∞	1	∞	∞
∞	∞	∞	0	∞	4	6	∞	∞	∞
8	∞	∞	∞	0	∞	8	∞	∞	2
∞	15	∞	4	∞	0	8	17	∞	∞
11	∞	∞	6	8	8	0	∞	∞	∞
∞	∞	1	∞	∞	17	∞	0	9	∞
17	∞	∞	∞	∞	∞	∞	9	0	12
∞	16	∞	∞	2	∞	∞	∞	12	0

Алгоритм поиска центра на неориентированном графе состоит из следующих шагов [1]:

1) алгоритмом Флойда найдём матрицу расстояний между всеми парами вершин;

2) умножим элементы каждой строки матрицы на вектор весов;

3) найдём в каждой строке максимальный элемент (расстояние от центра до самой удалённой вершины) и запишем его в массив Max;

4) найдём в столбце Max наименьший элемент;

2 ПРАКТИЧЕСКАЯ РЕАЛИЗАЦИЯ АЛГОРИТМА ПОИСКА ЦЕНТРА

2.1 Листинг реализации алгоритма

Разработка алгоритма осуществлялась на языке Java.

Функция, реализующая алгоритм Флойда:

```
public static int[][] floyd(int[][] graph) {

    int v = graph.length;
    int[][] g = Arrays.copyOf(graph, v);

    for (int i = 0; i < v; i++)
        g[i][i] = 0;
    for (int i = 0; i < v; i++)
       for (int j = 0; j < v; j++)
           for (int k = 0; k < v; k++)
               g[i][j] = Math.min(g[i][j], g[i][k + g[k][j]);

    return g;
}
```

Умножение элементов каждой строки матрицы на вектор весов:

```
    for (int i = 0; i < 10; i++)
       for (int j = 0; j < 10; j++)
          rasst[i][j] *= vesa[j];
```

Поиск в каждой строке максимального элемента:

```
    int[] max = new int[10];
         for (int i = 0; i < 10; i++) {
             int temp = 0;
             for (int j = 0; j < 10; j++)
                 if (rasst[i][j] > temp)
                     temp = rasst[i][j];

             max[i] = temp;
     }
```

Поиск в столбце Max наименьшего элемента:

```
    int res = Arrays.stream(max).min().getAsInt();
```

2.2 Результаты работы программы

На рисунке 2.1 представлена матрица расстояний между всеми парами вершин.

```
[0, 11, 20, 17, 8, 19, 11, 21, 17, 10]
[11, 0, 9, 19, 18, 15, 22, 10, 19, 16]
[20, 9, 0, 28, 27, 18, 26, 1, 10, 22]
[17, 19, 28, 0, 14, 4, 6, 21, 30, 16]
[8, 18, 27, 14, 0, 16, 8, 28, 14, 2]
[19, 15, 18, 4, 16, 0, 8, 17, 26, 18]
[11, 22, 26, 6, 8, 8, 0, 25, 22, 10]
[21, 10, 1, 21, 28, 17, 25, 0, 9, 21]
[17, 19, 10, 30, 14, 26, 22, 9, 0, 12]
[10, 16, 22, 16, 2, 18, 10, 21, 12, 0]
```

Рисунок 2.1 — Матрица расстояний между всеми парами вершин

На рисунке 2.2 представлен результат перемножения строк матрицы расстояний и вектора весов, а также наибольший элемент в строке.

```
max[0, 11, 40, 51, 32, 95, 66, 147, 136, 90] = 147
max[0, 0, 18, 57, 72, 75, 132, 70, 152, 144] = 152
max[0, 9, 0, 84, 108, 90, 156, 7, 80, 198] = 198
max[0, 19, 56, 0, 56, 20, 36, 147, 240, 144] = 240
max[0, 18, 54, 42, 0, 80, 48, 196, 112, 18] = 196
max[0, 15, 36, 12, 64, 0, 48, 119, 208, 162] = 208
max[0, 22, 52, 18, 32, 40, 0, 175, 176, 90] = 176
max[0, 10, 2, 63, 112, 85, 150, 0, 72, 189] = 189
max[0, 19, 20, 90, 56, 130, 132, 63, 0, 108] = 132
max[0, 16, 44, 48, 8, 90, 60, 147, 96, 0] = 147
```

Рисунок 2.2 — Результат перемножения строк матрицы расстояний и вектора весов

На рисунке 2.3 представлен результат работы программы для поиска центра графа

```
Матрица расстояний
[0, 11, 20, 17, 8, 19, 11, 21, 17, 10]
[11, 0, 9, 19, 18, 15, 22, 10, 19, 16]
[20, 9, 0, 28, 27, 18, 26, 1, 10, 22]
[17, 19, 28, 0, 14, 4, 6, 21, 30, 16]
[8, 18, 27, 14, 0, 16, 8, 28, 14, 2]
[19, 15, 18, 4, 16, 0, 8, 17, 26, 18]
[11, 22, 26, 6, 8, 8, 0, 25, 22, 10]
[21, 10, 1, 21, 28, 17, 25, 0, 9, 21]
[17, 19, 10, 30, 14, 26, 22, 9, 0, 12]
[10, 16, 22, 16, 2, 18, 10, 21, 12, 0]

Перемножение строк матрицы на вектор весов вершин
max[0, 11, 40, 51, 32, 95, 66, 147, 136, 90] = 147
max[0, 0, 18, 57, 72, 75, 132, 70, 152, 144] = 152
max[0, 9, 0, 84, 108, 90, 156, 7, 80, 198] = 198
max[0, 19, 56, 0, 56, 20, 36, 147, 240, 144] = 240
max[0, 18, 54, 42, 0, 80, 48, 196, 112, 18] = 196
max[0, 15, 36, 12, 64, 0, 48, 119, 208, 162] = 208
max[0, 22, 52, 18, 32, 40, 0, 175, 176, 90] = 176
max[0, 10, 2, 63, 112, 85, 150, 0, 72, 189] = 189
max[0, 19, 20, 90, 56, 130, 132, 63, 0, 108] = 132
max[0, 16, 44, 48, 8, 90, 60, 147, 96, 0] = 147

Оптимальный центр: 9 вершина со значением 132
```

Рисунок 2.3 — Результат работы программы для поиска центра графа

ЗАКЛЮЧЕНИЕ

В результате выполнения курсовой работы был реализован алгоритм оптимального размещения: центра на нагруженном взвешенном неориентированном графе (рёбрам приписаны положительные длины, вершинам – неотрицательные веса).

Если учитывать алгоритм Флойда – сложность алгоритма поиска центра составляет $O(n^5)$.

Если считать, что матрица расстояний уже каким-то образом получена (не учитывать Флойда), то сложность будет составлять $O(n^2)$.

Таким образом, алгоритм поиска центра графа можно использовать для графов большой размерности.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. Алгоритмы на сетях и графах, Миков А.Ю., Файнштейн С.И., МГТУ им. Носова, 2016 г.