МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение

высшего образования

**«КУБАНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»**

**(ФГБОУ ВО «КубГУ»)**

**Кафедра вычислительных технологий**

**КУРСОВАЯ РАБОТА**

**АНАЛИЗ АЛГОРИТМОВ НАХОЖДЕНИЯ АБСОЛЮТНОГО ЦЕНТРА ГРАФА**

Работу выполнил С.В. Мереуца

(подпись, дата) (инициалы, фамилия)

Факультет компьютерных технологий и прикладной математики 3 курс

Направление 02.03.02 – «Фундаментальная информатика и информационные технологии»

Научный руководитель

доц., канд.ф-м.н., Е.В. Кособуцкая

(подпись, дата) (инициалы, фамилия)

Нормоконтролер

доц., канд.ф-м.н., Е.В. Кособуцкая

(подпись, дата) (инициалы, фамилия)

Краснодар 2018

Содержание

[ВВЕДЕНИЕ 3](#_Toc515870977)

[1. Основные понятия теории графов 4](#_Toc515870978)

[1.1 Кратчайшие пути в графе 5](#_Toc515870979)

[1.2 Центр графа 6](#_Toc515870980)

[2. Использованные алгоритмы 9](#_Toc515870981)

[2.1 Алгоритмы поиска кратчайших путей 9](#_Toc515870982)

[2.1 Алгоритмы поиска абсолютного центра 10](#_Toc515870983)

[3. Описание программной реализации модифицированного алгоритма Хакими нахождения абсолютного центра графа. 14](#_Toc515870984)

[3.1 Описание программы 14](#_Toc515870985)

[3.4 Сравнение работы программы при различных выходных данных 15](#_Toc515870986)

[ЗАКЛЮЧЕНИЕ 18](#_Toc515870987)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 19](#_Toc515870988)

# ВВЕДЕНИЕ

Графы нашли применение практически во всех отраслях научных знаний: физике, биологии, химии, математике, истории, лингвистике, социальных науках, технике и т.п. Наибольшей популярностью теоретико-графовые модели используются при исследовании коммуникационных сетей, систем информатики, химических и генетических структур, электрических цепей и других систем сетевой структуры.

Довольно часто в практической деятельности возникает необходимость размещения предприятий и служб или какого-либо оборудования в сети. В таком случае важную роль играет минимизация расстояния располагаемого объекта от самой удаленной вершины, то есть оптимизация худшего варианта. Для решения такого рода задач были созданы алгоритмы поиска центра графа.

Если предположить, что располагаемый объект может находиться не только в каком-то районе, то есть вершине, а вообще в любой точке графа, то

следует применять алгоритмы поиска абсолютного центра, то есть точки на графе, которая не обязательно является вершиной графа.

Целью данной курсовой работы является анализ алгоритмов нахождения абсолютного центра графа, реализация модифицированного алгоритма Хакими на языке программирования C#, а также оценка времени выполнения алгоритма для графов разной размерности и плотности.

# 1. Основные понятия теории графов

Граф – абстрактный математический объект, представляющий собой множество вершин графа и набор рёбер, то есть соединений между парами вершин.

Существует много способов представить граф:

1. Перечисление элементов. Исходя из определения, для того, чтобы задать граф, достаточно перечислить его вершины и ребра (т.е. пары вершин).

2. Изображение. Если граф невелик, его можно нарисовать. В неориентированном графе ребра изображаются линиями, в ориентированном – стрелками.

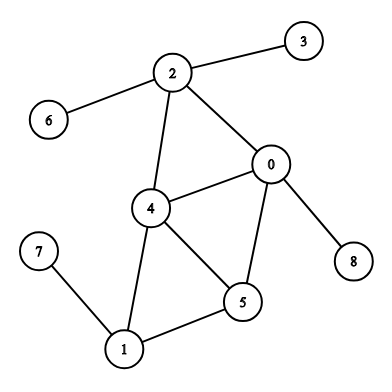


Рисунок 1 – Пример представления графа в виде изображения

3. Матрица смежности. Пусть G – граф с n вершинами, пронумерованными числами от 1 до n. Матрица смежности – это таблица с n строками и n столбцами, в которой элемент, стоящий на пересечении строки с номером i и столбца с номером j, равен 1, если вершины с номерами i и j смежны, и 0, если они не смежны.

4. Матрица инцидентности. Пусть G – граф, вершины которого пронумерованы числами от 1 до n, а ребра – числами от 1 до m. В матрице инцидентности строки соответствуют вершинам, а столбцы – ребрам. На пересечении строки с номером i и столбца с номером j стоит 1, если вершина с номерами i инцидентна ребру с номером j смежны, и 0 в противном случае.

5. Списки смежности. Этот способ часто используется для компьютерного представления графов. Состоит он в том, что для каждой вершины задается список всех смежных с ней вершин. В структурах данных, применяемых в программировании, списки смежности могут быть реализованы как массив линейных списков.

Граф G называется связанным, если любая пара его вершин соединена простой цепью.

## 1.1 Кратчайшие пути в графе

Задача поиска центра графа напрямую связана с кратчайшими путями в графе.

Длина маршрута равна количеству ребер в порядке их прохождения. Длина кратчайшей простой цепи, соединяющей вершины vi и vj в графе G, называется расстоянием d (vi, vj) между vi и vj.

Задача о кратчайшем пути – задача поиска самого короткого пути (цепи) между двумя вершинами на графе, в которой минимизируется сумма весов рёбер, составляющих путь. Она является одной из важнейших классических задач теории графов.

Значимость данной задачи определяется её различными практическими применениями. Например, в GPS-навигаторах осуществляется поиск кратчайшего пути между двумя перекрестками. В качестве вершин выступают перекрестки, а дороги являются ребрами, которые лежат между ними. Если сумма длин дорог между перекрестками минимальна, тогда найденный путь самый короткий.

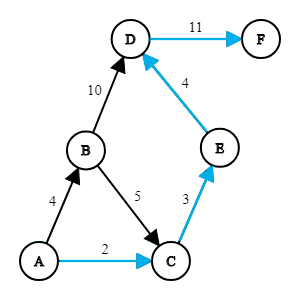


Рисунок 2 – Кратчайший путь между вершинами A и F во взвешенном ориентированном графе

## 1.2 Центр графа

Числа so(vi)и st(vi) называются соответственно числом внешнего разбиения и числом внутреннего разбиения вершины vi в графе G = (V, E) :

(1)

и

, (2)

где каждая вершина , а ui – это вес вершины vi.

Следует отметить, что so(vi) является наибольшим числом в строке соответствующей вершине vi матрицы D’(G), которая путем умножения каждого столбца j матрицы расстояний D(G) на vj. В то же время, st(vi) является наибольшим числом в столбце соответствующем вершине vj матрицы D’’(G), которая путем умножения каждой строчки j матрицы расстояний D(G) на vi.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | x1 | x2 | x3 | x4 | x5 | x6 | so(xi) |
| x1 | 0 | 1 | 2 | 3 | 2 | 3 | 3 |
| x2 | 3 | 0 | 1 | 2 | 1 | 2 | 3 |
| x3 | 4 | 3 | 0 | 1 | 2 | 3 | 4 |
| x4 | 3 | 2 | 2 | 0 | 1 | 2 | 3 |
| x5 | 2 | 1 | 1 | 2 | 0 | 1 | 2\* |
| x6 | 1 | 2 | 3 | 4 | 3 | 0 | 4 |
| st(xi) | 4 | 3\* | 3\* | 4 | 3\* | 3\* |  |

Рисунок 3 – Пример матрицы расстояний и поиска чисел разделений по ней[1]

Вершина vi\*, для которой выполняется:

, (3)

где каждая вершина , называется внешним центром графа G и аналогично, если для вершины vi\* выполняется:

, (4)

где каждая вершина , называется внутренним центром графа G.

Число внешнего разделения вершины, являющейся внешним центром, является внешним радиусом, а число внутреннего разделения вершины, являющейся внутренним центром является внутренним радиусом.

В неориентированном графе so(vi)и st(vi) равны(далее s(vi)).

Если при выборе центра не ограничиваться множеством вершин графа, а также рассматривать точки на графе, то поиск числа разделения выглядит так:

, (5)

где каждая вершина . В таком случае поиск центра:

, (6)

где y – каждая точка графа G.

Такой центр называется абсолютным, а соответствующий ему радиус называется абсолютным радиусом.

# 2. Алгоритмы необходимые для поиска абсолютного центра графа

## 2.1 Алгоритмы поиска кратчайших путей

Задача о кратчайшем пути является одной из важнейших классических задач теории графов. Сегодня известно множество алгоритмов для её решения, однако чаще всего разные алгоритмы выполняют немного разные задачи. Например, алгоритм Дейкстры находит кратчайший путь от одной из вершин графа до всех остальных, тогда как алгоритм Флойда – Уоршелла находит кратчайшие пути между всеми вершинами взвешенного ориентированного графа.

Для поиска абсолютного центра необходима матрица расстояний для всех вершин, таким образом, можно сделать вывод, что наиболее подходящим алгоритмом для построения такой матрицы является, например, алгоритм Флойда – Уоршелла.

Алгоритм Флойда – Уоршелла – динамический алгоритм для нахождения кратчайших расстояний между всеми вершинами взвешенного ориентированного графа. Разработан в 1962 году Робертом Флойдом и Стивеном Уоршеллом. При этом алгоритм впервые разработал и опубликовал Бернард Рой.

Алгоритм Флойда – Уоршелла является эффективным для расчёта всех кратчайших путей в плотных графах, когда имеет место большое количество пар рёбер между парами вершин.

В данном алгоритме формируется матрица кратчайших путей по следующему принципу. Для каждой пары вершин va, vb длина пути из матрицы кратчайших путей сравнивается с длинной пути через каждую вершину vk, где vk – это вершина графа.

Три вложенных цикла содержат операцию, исполняемую за константное время , то есть алгоритм имеет кубическую сложность, при этом простым расширением можно получить также информацию о кратчайших путях — помимо расстояния между двумя узлами записывать в матрицу идентификатор первого узла в пути.

## 2.1 Алгоритмы поиска абсолютного центра

Существует несколько алгоритмов для поиска абсолютного центра. Одним из них является алгоритм Хакими.

Данный алгоритм состоит в следующем. Необходимо для каждого ребра в графе найти точку с наименьшим числом разделения. После этого из всех получившихся точек выбрать точку с наименьшим числом разделения.

Первый шаг осуществляется следующим образом.

(7)

где

, (8)

где – расстояние до от точки на ребре до одной из вершин инцидентных ребру.

Можно обозначить как ε, тогда если cab – вес ребра, то выбор числа разделения приобретает следующий вид:

(9)

Удобно рассматривать эту форму в виде двух выражений:

(10)

Рассмотрим применение функций вида (10) для графа, изображенного на рисунке 4.

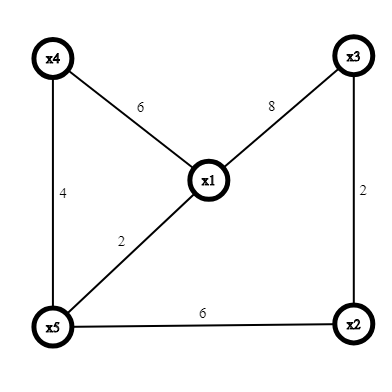


Рисунок 4 - Пример взвешенного неориентированного графа[1]

Первым шагом будет исследование графиков функций вида (10) для каждого ребра. Например, график для ребра (x1, x3) (рисунок 5).

В описанном выше методе Хакими поиск локального центра осуществляется вдоль всего ребра графа G. Если в графе много ребер, то время вычисления, требуемое для поиска, может оказаться чрезвычайно большим.

Для решения этой проблемы была придумана модификация метода Хакими, которая состоит в вычислении верхних и нижних оценок абсолютных локальных радиусов, соответствующих локальным центрам ребер, и в использовании полученных оценок для уменьшения числа ребер, участвующих в поиске.

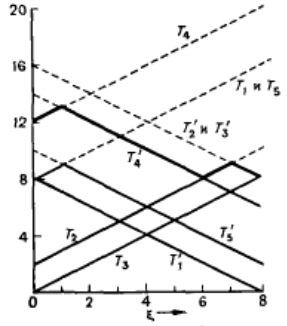


Рисунок 5 - График изменения расстояния до вершин для разных ε

Всякому локальному центру, расположенному на ребре (va, vb) соответствует его абсолютный локальный радиус, который не меньше, чем pab, где

. (11)

Таким образом, pab есть нижняя оценка абсолютного радиуса графа, если абсолютный центр лежит на ребре (va, vb).

Если предположить, что абсолютный центр графа расположен в середине ребра (va, vb), тогда абсолютный радиус равен:

, (12)

где us\* – вес той вершины, в которой достигается наибольшее значение величины pab.

Следовательно

, (13)

является обоснованной верхней оценкой абсолютного радиуса графа. Таким образом, каждое ребро, для которого нижняя оценка абсолютного радиуса не меньше обоснованной меньшей оценки, не может рассматриваться при поиске абсолютного центра.

Кроме рассмотренного алгоритма Хакими и его модифицированного варианта, существует итерационный алгоритм.

Пусть Qλ(vi) – множество всех таких точек y, лежащих на ребрах графа G, из которых вершина vi достижима со взвешенным расстоянием, не превосходящим λ. Таким образом,

, (14)

где y лежит на ребрах графа G. Отсюда, абсолютный радиус r графа является наименьшим λ, при котором из некоторой точки y графа G все вершины могут быть достигнуты со взвешенным расстоянием меньшим или равным λ. Следовательно, r – это такое наименьшее λ, что

. (15)

Таким образом, нужно начать с произвольного небольшого λ, строить множества Qλ(vi) для всех вершин и проверять выполняется ли соотношение (15). Если оно не выполняется, то необходимо увеличить λ, на небольшую величину и попробовать снова. Эту процедуру нужно повторять, пока не выполнится соотношение (15).

# 3. Описание программной реализации модифицированного алгоритма Хакими нахождения абсолютного центра графа.

Программная реализация модифицированного алгоритма Хакими для поиска абсолютного центра графа выполнена на языке C#.

## 3.1 Описание программы

В данной программе используется представление графа в виде матрицы, так как в этом случае прямой доступ к элементам ускоряет процесс выполнения программы. Для построения матрицы кратчайших путей программа использует алгоритм Флойда — Уоршелла, что также предполагает использование матрицы. Основной проблемой матричного представления является избыточные затраты памяти для графов с относительно малым количеством ребер. Но в данном случае граф достаточно плотный.

В программе присутствует несколько вспомогательных методов. Одним их них является метод для генерации графа при заданной размерности и вероятности присутствия для ребер графа. Данный метод случайным создает ребро между вершинами с длиной определяемой генератором случайных чисел в диапазоне от 0 до 1000. Затем вызывается метод для проверки связности графа основанный на поиске в глубину, который возвращает массив с вершинами, недостижимыми из вершины 1. Создается ребро между случайной достижимой вершиной и одной из недостижимых вершин. Затем снова вызывается метод для проверки связности и добавление новых ребер до тех пор, пока граф не станет связным.

Далее вызывается метод для построения матрицы кратчайших расстояний в графе. Данный метод основан на алгоритме поиска кратчайших расстояний Флойда — Уоршелла.

После нахождения кратчайших расстояний выполняется часть алгоритма, из-за которой метод и называется модифицированным. Программа вычисляет верхнюю оценку абсолютного радиуса графа. Она нужна для того, чтобы отбросить ребра, которые не следует проверять, так как абсолютный центр на них точно отсутствует. Таким образом, остается множество ребер, которые необходимо проанализировать.

Для каждого из полученных ребер выполняются следующие действия. Вычисляется некоторый достаточно маленький шаг. Данная программа в качестве шага выбирает 1% от веса ребра. Шаг определяет отдаление от одной из вершин предполагаемого абсолютного центра на каждой итерации, то есть сумма шагов – это положение проверяемой вершины на ребре. Выполняется поиск самой отдаленной от предполагаемого центра вершины и расстояние до нее. После этого происходит проверка, является ли данное расстояние наименьшим из всех найденных до этого. В итоге получено расположение абсолютного центра и абсолютный радиус графа.

Время замеряется для графов размерностью от 10 до 1000 вершин с интервалом 50 и плотностью ребер от 0.1 до 1 с интервалом 0.3.

## 3.4 Сравнение работы программы при различных выходных данных для различных алгоритмов

Для того, чтобы оценить работу алгоритма, производятся замеры времени выполнения данной реализации алгоритма Хакими и его модифицированной версии на графах разной размерности и при разном количестве ребер относительно полного графа.

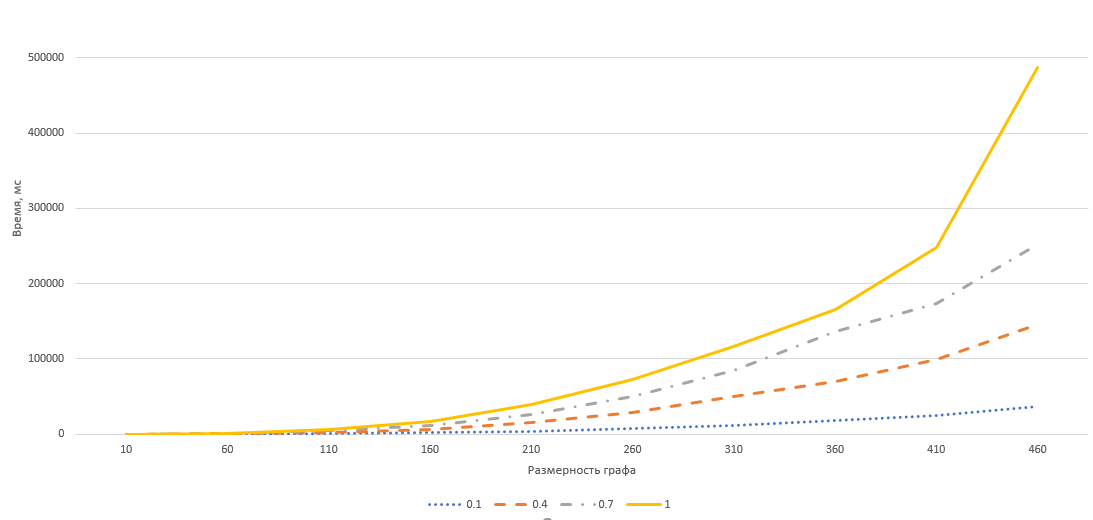


Рисунок 6 – График зависимости времени (в мс) от размерности при различной плотности графа для алгоритма Хакими (интервал 10 – 460)

Как видно на рисунке 6, скорость роста времени выполнения программы относительно размерности графа увеличивается с его плотностью. Так как графы и вес ребер в них генерировались случайным образом, то исходя из графика можно сказать, что вес ребер не влияет на время выполнения программы.

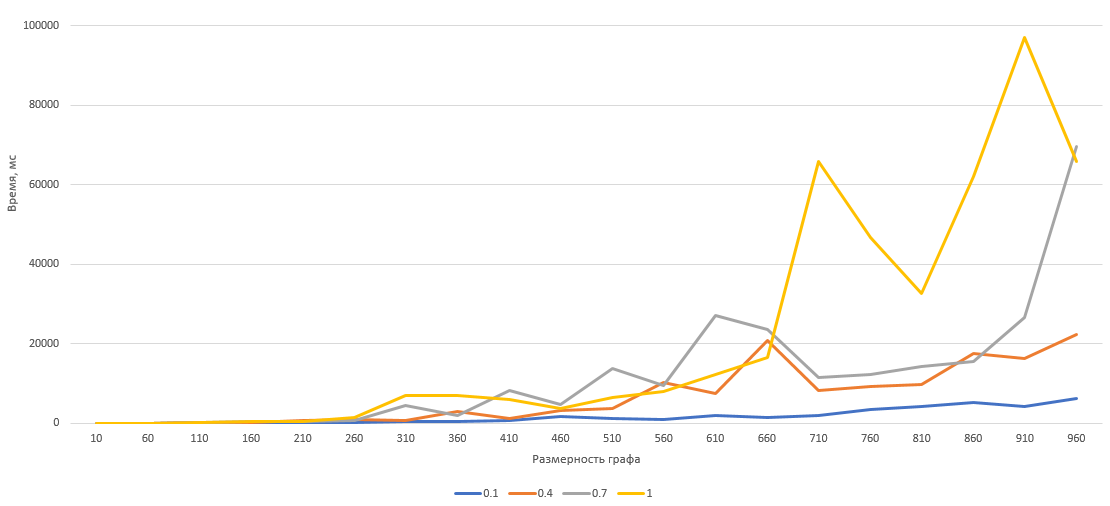


Рисунок 7 – График зависимости времени (в мс) от размерности при различной плотности графа для модифицированного алгоритма Хакими (интервал 10 – 960)

Модифицированный алгоритм Хакими для уменьшения времени работы использует отсеивание ребер, в которых абсолютного центра быть не может. Это видно на графике (рисунок 7) скорость роста времени выполнения программы в зависимости от размерности графа зависит не только от плотности графа, но и от весов ребер. Так как графы и вес ребер в них генерировались случайным образом, то и время выполнения растет с перепадами.

# ЗАКЛЮЧЕНИЕ

В ходе выполнения курсовой работы были изучены различные алгоритмы поиска абсолютного центра графа, а также реализован алгоритм Хакими на языке программирования C#.

Проведены исследования данной реализации модифицированного алгоритма Хакими для графов различной размерности и различным относительным количеством ребер. Выяснено, что в данной реализации время выполнения имеет слабую зависимость от относительного количества ребер, однако сильно зависит от размерности графа.

# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1 Кристофидес Н. Теория графов. Алгоритмический подход. – Мир 1978

2 Рой, Бернард Transitivité et connexité. – C. R. Acad. Sci. Paris 1959

3 Троелсен Э. Язык программирования C# 5.0 и платформа .NET 4.5. – “Вильямс, 2013