**Текстовое описание предметной области**

[Основные понятия и определения 1](#_Toc157560735)

[Виды графов 2](#_Toc157560736)

[Представление графа 5](#_Toc157560737)

[Область применения 7](#_Toc157560738)

[Алгоритмы на графах и соответствующие им задачи 8](#_Toc157560739)

[Визуализация алгоритмов 12](#_Toc157560740)

[Список использованной литературы 13](#_Toc157560741)

**Основные понятия и определения**

Граф – непустое множество вершин и набор неупорядоченных и упорядоченных пар вершин вида (v,w). Обычно граф обозначают как G (V, E); количество вершин и ребер обозначается, соответственно, n(G) и m(G). [1]

Неупорядоченная пара вершин называется ребром {v,w}, упорядоченная пара – дугой (v,w). [1]

Граф, содержащий только ребра, называется неориентированным (обозначается G). Граф, содержащий только дуги – ориентированным (или орграфом) (обозначается ). [1]

Пара вершин может быть соединена двумя или более рёбрами (или, соответственно, дугами одного направления), такие рёбра (или дуги) называются кратными. Дуга (или ребро) может начинаться и заканчиваться в одной и той же вершине, в этом случае соответствующая дуга (или ребро) называется петлёй. [1]

Граф без кратных рёбер и петель называется простым графом. [1]

Граф, все n вершин которого являются изолированными, называется нулевым (пустым) и обозначается . [1]

Простой граф, любые две вершины которого являются смежными, называется полным. Полный граф с n вершинами обозначается . [1]

Вершины, соединённые ребром или дугой, называются смежными. Рёбра, имеющие общую вершину, тоже называются смежными. Ребро (или дуга) и любая из его вершин называются инцидентными. [1]

Говорят, что ребро {v,w} соединяет вершины v и w. Для орграфов: дуга (v,w) начинается в вершине v (исходит из вершины v) и заканчивается в вершине w (заходит в вершину w), или идет из вершины v в вершину w. [1]

Степенью вершины v графа G называется число deg v рёбер графа G, инцидентных вершине v, при этом петли учитываются дважды. Вершина графа, имеющая степень 0, называется изолированной, а степень 1 – тупиковой (висячей, концевой). [1]

Путем или цепью в графе называют конечную последовательность вершин, в которой каждая вершина (кроме последней) соединена со следующей в последовательности вершин ребром. Циклом называют путь, в котором первая и последняя вершины совпадают. Путь или цикл называют простым, если ребра в нем не повторяются. Если в графе любые две вершины соединены путем, то такой граф называется связным. [2]

**Виды графов**

1. Ориентированные и неориентированные графы

Неориентированный граф – граф, ребрам которого не присвоено заданное направление. [3]

Изображение выглядит как линия, круг, диаграмма

Автоматически созданное описание

Рисунок 1 – Неориентированный граф

Ориентированный граф – граф, каждому ребру которого приписана ориентация. [3]

Изображение выглядит как линия, круг

Автоматически созданное описание

Рисунок 2 – Ориентированный граф

1. Пустой граф – граф, который состоит только из голых вершин. [2]

Изображение выглядит как круг, зарисовка

Автоматически созданное описание

Рисунок 3 – Пустой граф

1. Неориентированный граф, без петель и кратных ребер называется обыкновенным. [2]
2. Полный граф – граф, в котором каждая вершина соединена со всеми другими вершинами. [4]

Изображение выглядит как линия, круг, треугольник, диаграмма

Автоматически созданное описание

Рисунок 4 – Полный граф

1. Двудольный граф – граф, вершины которого можно разбить на две части. При этом ребра будут проходить только между частями, но никогда внутри одной из них. [5]

Изображение выглядит как круг, зарисовка, линия, рисунок

Автоматически созданное описание

Рисунок 5 – Двудольный граф

1. Эйлеровым графом называется граф, в котором есть эйлеров цикл. Эйлеровым циклом называется простой цикл, содержащий все ребра. [6]

Изображение выглядит как линия, круг, белый, зарисовка

Автоматически созданное описание

Рисунок 6 – Эйлеров граф

1. Регулярный граф – связный граф, все вершины которого имеют одинаковую степень k. Число вершин регулярного графа k-й степени не может быть меньше k+1. У регулярного графа нечётной степени может быть лишь чётное число вершин. [2]

Изображение выглядит как линия, круг, диаграмма, зарисовка

Автоматически созданное описание

Рисунок 7 – Регулярный граф

1. Гамильтонов граф – граф, содержащий гамильтонов цикл. Гамильтоновым циклом называется простой цикл, который проходит через все вершины рассматриваемого графа. [2]

Изображение выглядит как линия, круг, белый, диаграмма

Автоматически созданное описание

Рисунок 7 – Гамильтонов граф

1. Взвешенный граф – граф, вершины или рёбра которого характеризуются некоторой дополнительной информацией – весами вершин или рёбер. [7]

Изображение выглядит как диаграмма, линия, круг

Автоматически созданное описание

Рисунок 8 – Взвешенный граф

1. Дерево – связный граф без циклов.

Изображение выглядит как круг, зарисовка, диаграмма, белый

Автоматически созданное описание

Рисунок 9 – Дерево

**Представление графа**

Визуальное представление графа, в том числе ориентированного, хорошо понятно человеку. Но компьютер не понимает такую структуру, поэтому в программировании графы реализуют иначе. Их кодируют с помощью одного из двух способов: матрицы смежности или списка смежности. [8]

Матрица смежности – это способ представить граф в виде таблицы. У нее N столбцов и N строк, где N — количество вершин в графе. Столбцы и строки имеют те же названия или номера, что и вершины. Заполняется матрица так: если из вершины A в вершину B ведет ребро, то на пересечении строки A и столбца B в таблицу ставится единица. Такое повторяется для всех ребер, а оставшиеся ячейки заполняются нулями. [8]



Рисунок 10 – Матрицы смежности для неориентированного и ориентированного графов

Списки смежности – это второй способ. Он менее нагляден, но занимает меньше места в памяти. Его используют, если количество ребер намного меньше, чем количество вершин в квадрате, — так называемый разреженный граф. Матрица будет заполненной нулями. Для хранения в памяти компьютера это критично: полезной информации мало, зато структура отнимает много ресурсов. Поэтому при малом количестве ребер используют список. [8]

Список смежности – это N массивов данных, где N – количество вершин. На каждую вершину приходится один массив: в нем содержится имя этой вершины и список ее «соседей». То есть у каждой вершины есть список тех, что соединены с ней ребром. В этом способе представления удобнее узнавать о соседях вершины. Также он занимает меньше памяти, и алгоритмы проходят по нему быстрее. Но он плохо подходит для плотных графов. [8]

Второй распространенный синтаксис для преобразования графов в матрицы – это матрица инцидентности. В ней граф G с множеством вершин V и множеством ребер E преобразуется в матрицу размером V на E. Строки и столбцы обозначаются как вершины и ребра соответственно. Внутри матрицы мы снова видим, что все элементы обозначены либо как 0, либо как 1. На этот раз единица означает связь между вершиной в строке и ребром в столбце. [9]

Изображение выглядит как снимок экрана, круг

Автоматически созданное описание

Рисунок 11 – Граф и его матрица инцидентности

**Область применения**

На основе графов создают модели тех или иных процессов, событий, взаимоотношений или объектов. Графы как концепция используются в точных, естественных и гуманитарных науках: от физики и химии до экономики или социологии. С их помощью описывают разнообразные процессы, связи, структуры. Это может быть и схема работы алгоритма, и визуализация социальных взаимоотношений – практически любая структура, которую можно представить как набор элементов и связей между ними. [8] Широкое применение теории графов в компьютерных науках и информационных технологиях можно объяснить понятием графа как структуры данных. В компьютерных науках и информационных технологиях граф можно описать, как нелинейную структуру данных. Линейные структуры данных характеризуются тем, что связывают элементы отношениями типа «простого соседства». Линейными структурами данных являются, например, массивы, таблицы, списки, очереди, стеки, строки. В противоположность им нелинейные структуры данных – такие, в которых элементы располагаются на различных уровнях иерархии и подразделяются на три вида: исходные, порождённые и подобные. Итак, граф – нелинейная структура данных. [10]

**Алгоритмы на графах и соответствующие им задачи**

Так как инструмент создаётся для учебных целей, рассмотрим популярные алгоритмы и задачи для них, которые разбираются на дисциплинах «Дискретная математика» и «Алгоритмы и структуры данных».

1. Теорема Эйлера и Алгоритм Флёри

В городе Кёнигсберге были расположены семь мостов на реке Преголь, и спрашивалось, можно ли, выйдя из дома, вернуться обратно, пройдя в точности один раз по каждому мосту. Задача о кёнигсбергских мостах на языке теории графов формулируется так: существует ли в мультиграфе хотя бы один цикл, содержащий все ребра этого графа? В 1736 г. Л. Эйлер в трудах петербургской академии наук доказал, что не существует цикла, включающего каждое ребро графа по одному разу. Цикл (цепь) в графе называется эйлеровым, если он (она) содержит все ребра графа. Связный граф, в котором есть эйлеров цикл (цепь), называется эйлеровым графом. Такой граф можно нарисовать, не отрывая карандаша от бумаги и не повторяя линий. [11]

Теорема Эйлера: Связный граф является эйлеровым тогда и только тогда, когда степени всех его вершин четны. [11]

Наиболее простым алгоритмом нахождения Эйлерова цикла является алгоритм Флёри: пусть G – эйлеров граф, тогда следующая процедура всегда возможна и приводит к эйлеровому циклу графа G. Выходим из произвольной вершины и идём по рёбрам графа произвольным образом, соблюдая лишь следующие правила:

1. Стираем рёбра по мере их прохождения и стираем также изолированные вершины, которые при этом образуются.
2. На каждом этапе идём по мосту только тогда, когда нет других возможностей. (Мост – это ребро, которое после удаления делает граф не связным). [12]
3. Минимальное остовное дерево. Алгоритм Крускала. Алгоритм Прима

Остовным деревом графа называется дерево, которое можно получить из него путём удаления некоторых рёбер. Для взвешенных графов существует понятие веса остовного дерева, которое определено как сумма весов всех рёбер, входящих в остовное дерево. Из него натурально вытекает понятие минимального остовного дерева – остовного дерева с минимальным возможным весом. Для нахождения минимального остовного дерева графа существуют два основных алгоритма: алгоритм Крускала и алгоритм Прима. Они оба имеют сложность O(MlogN). [13]

Алгоритм Крускала был описан Крускалом (Kruskal) в 1956 г. Алгоритм Крускала изначально помещает каждую вершину в своё дерево, а затем постепенно объединяет эти деревья, объединяя на каждой итерации два некоторых дерева некоторым ребром. Перед началом выполнения алгоритма все рёбра сортируются по весу (в порядке возрастания). Затем начинается процесс объединения: перебираются все рёбра от первого до последнего (в порядке сортировки), и если у текущего ребра его концы принадлежат разным поддеревьям, то эти поддеревья объединяются, а ребро добавляется к ответу. По окончании перебора всех рёбер все вершины окажутся принадлежащими одному поддереву, и ответ найден. [14]

Алгоритм Прима назван в честь американского математика Роберта Прима (Robert Prim), который открыл этот алгоритм в 1957 г. Сам алгоритм имеет очень простой вид. Искомый минимальный остов строится постепенно, добавлением в него рёбер по одному. Изначально остов полагается состоящим из единственной вершины (её можно выбрать произвольно). Затем выбирается ребро минимального веса, исходящее из этой вершины, и добавляется в минимальный остов. После этого остов содержит уже две вершины, и теперь ищется и добавляется ребро минимального веса, имеющее один конец в одной из двух выбранных вершин, а другой – наоборот, во всех остальных, кроме этих двух. И так далее, т. е. всякий раз ищется минимальное по весу ребро, один конец которого – уже взятая в остов вершина, а другой конец – ещё не взятая, и это ребро добавляется в остов (если таких рёбер несколько, можно взять любое). Этот процесс повторяется до тех пор, пока остов не станет содержать все вершины (или, что то же самое, n-1 ребро). В итоге будет построен остов, являющийся минимальным. Если граф был изначально не связен, то остов найден не будет (количество выбранных рёбер останется меньше n-1). [15]

1. Кратчайший путь между вершинами графа. Алгоритм Дейкстры. Алгоритм Флойда

В таблице 1 приведена сравнительная информация двух алгоритмов нахождения кратчайшего пути между вершинами графа.

Таблица 1 – Сравнение алгоритма Дейкстры и алгоритма Флойда

| **Критерий** | **Алгоритм Дейкстры** | **Алгоритм Флойда** |
| --- | --- | --- |
| Описание | Алгоритм кратчайшего пути из одного источника, используемый для нахождения кратчайших путей от одной исходной вершины ко всем остальным вершинам взвешенного графа | Алгоритм кратчайшего пути для всех пар, используемый для нахождения кратчайших путей между всеми парами вершин во взвешенном графе |
| Алгоритм | Жадный подход | Динамическое программирование |
| Структура данных | Обычно используется очередь с приоритетом или куча | Двумерный массив |
| Отрицательные ребра | - | + |
| Временная сложность |  |  |

1. Задача топологической сортировки и Алгоритм Демукрона

Дан ориентированный граф с n вершинами и m рёбрами. Требуется пронумеровать его вершины таким образом, чтобы каждое ребро вело из вершины с меньшим номером в вершину с большим. Для решения этой задачи можно воспользоваться алгоритмом Демукрона [16]:

1. Найти все источники и пронумеровать их.
2. Удалить вершины-источники и исходящие из них дуги.
3. В новом графе проделать то же самое.

Ограничение: если в графе есть цикл, то отсортировать его нельзя.

1. Обход графа

Поиск в глубину. При поиске в глубину посещается первая вершина, затем необходимо идти вдоль ребер графа, до попадания в тупик. Вершина графа является тупиком, если все смежные с ней вершины уже посещены. После попадания в тупик нужно возвращаться назад вдоль пройденного пути, пока не будет обнаружена вершина, у которой есть еще не посещенная вершина, а затем необходимо двигаться в этом новом направлении. Процесс оказывается завершенным при возвращении в начальную вершину, причем все смежные с ней вершины уже должны быть посещены. Основная идея поиска в глубину – когда возможные пути по ребрам, выходящим из вершин, разветвляются, нужно сначала полностью исследовать одну ветку и только потом переходить к другим веткам (если они останутся нерассмотренными). [3]

Алгоритм:

1. Всем вершинам графа присваивается значение не посещенная. Выбирается первая вершина и помечается как посещенная.
2. Для последней помеченной как посещенная вершины выбирается смежная вершина, являющаяся первой помеченной как не посещенная, и ей присваивается значение посещенная. Если таких вершин нет, то берется предыдущая помеченная вершина.
3. Повторить шаг 2 до тех пор, пока все вершины не будут помечены как посещенные. [3]

Также часто используется нерекурсивный алгоритм поиска в глубину. В этом случае рекурсия заменяется на стек. Как только вершина просмотрена, она помещается в стек, а использованной она становится, когда больше нет новых вершин, смежных с ней. Временная сложность зависит от представления графа. Если применена матрица смежности, то временная сложность равна , а если нематричное представление – : рассматриваются все вершины и все ребра. [3]

Поиск в ширину. При поиске в ширину, после посещения первой вершины, посещаются все соседние с ней вершины. Потом посещаются все вершины, находящиеся на расстоянии двух ребер от начальной. При каждом новом шаге посещаются вершины, расстояние от которых до начальной на единицу больше предыдущего. Чтобы предотвратить повторное посещение вершин, необходимо вести список посещенных вершин. Для хранения временных данных, необходимых для работы алгоритма, используется очередь. Основная идея поиска в ширину заключается в том, что сначала исследуются все вершины, смежные с начальной вершиной (вершина, с которой начинается обход). Эти вершины находятся на расстоянии 1 от начальной. Затем исследуются все вершины на расстоянии 2 от начальной, затем все на расстоянии 3 и т. д. Обратим внимание, что при этом для каждой вершины сразу находятся длина кратчайшего маршрута от начальной вершины. [3]

Алгоритм:

1. Всем вершинам графа присваивается значение не посещенная. Выбирается первая вершина и помечается как посещенная (и заносится в очередь).
2. Посещается первая вершина из очереди (если она не помечена как посещенная). Все ее соседние вершины заносятся в очередь. После этого она удаляется из очереди.
3. Повторяется шаг 2 до тех пор, пока очередь не пуста. [3]

Сложность поиска в ширину при нематричном представлении графа равна , ибо рассматриваются все n вершин и m ребер. Использование матрицы смежности приводит к оценке [3]

**Визуализация алгоритмов**

Визуализация алгоритмов – это процесс представления работы алгоритмов в форме наглядного, последовательного объяснения, представленного в виде изображений. Цель визуализации – помочь людям понять работу алгоритма путем визуального представления шагов его выполнения. Поэтому пользователи должны иметь возможность управлять скоростью показа визуализации, должны иметь возможность перемотать показ визуализации вперёд или назад. Так как инструмент будет предназначен для студентов, необходимо сопровождать визуализацию подробными объяснениями-описанием применения алгоритма на заданном графе. При создании инструмента выгодно использовать событийно-ориентированный подход к визуализации, как наиболее соответствующий образовательным целям (изучению алгоритмов).

**Список использованной литературы**

1. Теория графов. Дискретная математика [Электронный ресурс]. URL: https://siblec.ru/informatika-i-vychislitelnaya-tekhnika/diskretnaya-matematika/3-teoriya-grafov#3.1 (Дата обращения: 29.01.2024).
2. Основные понятия теории графов [Электронный ресурс]. URL: https://skysmart.ru/articles/mathematic/osnovnye-ponyatiya-teorii-grafov (Дата обращения: 29.01.2024).
3. Галина Ваныкина, Татьяна Сундукова Структуры и алгоритмы компьютерной обработки данных: учебное пособие. : Интернет-Университет Информационных Технологий (ИНТУИТ), Ай Пи Ар Медиа, 2020.
4. Полный граф [Электронный ресурс]. URL: https://foxford.ru/wiki/matematika/polnyj-graf (Дата обращения: 29.01.2024).
5. Двудольные графы [Электронный ресурс]. URL: https://ru.hexlet.io/courses/graphs/lessons/bipartite/theory\_unit (Дата обращения: 29.01.2024).
6. Эйлеровы графы [Электронный ресурс]. URL: https://foxford.ru/wiki/matematika/eylerovy-grafy (Дата обращения: 29.01.2024).
7. Л.Л.Босова, А.Ю.Босова Информатика 9 класс: учебник. М.: БИНОМ. Лаборатория знаний, 2017. 208 с.
8. Ориентированный граф [Электронный ресурс]. URL: https://blog.skillfactory.ru/glossary/orientirovannyj-graf/ (Дата обращения: 29.01.2024).
9. Нотации — Теория графов [Электронный ресурс]. URL: https://ru.hexlet.io/courses/graphs/lessons/notation/theory\_unit (Дата обращения: 29.01.2024).
10. Теория графов. Основные понятия теории графов. Способы представления графа в памяти компьютера. [Электронный ресурс]. URL: https://teletype.in/@gamedev\_academy\_feedback/waq9yvgPpw\_ (Дата обращения: 29.01.2024).
11. В. Н. Степанов Дискретная математика: графы и алгоритмы на графах: учебное пособие. Омск: Издательство ОмГТУ, 2010. 120 с.
12. Ш.Ф.Арасланов ТЕОРИЯ ГРАФОВ ЛЕКЦИИ И ПРАКТИЧЕСКИЕ ЗАНЯТИЯ: учебное пособие. Казань: Казанск. гос. архитект.-строит. ун-та, 2013. 87 с
13. Минимальное остовное дерево. Алгоритм Прима. Алгоритм Крускала [Электронный ресурс]. URL: https://brestprog.by/topics/mst/ (Дата обращения: 29.01.2024).
14. Минимальное остовное дерево. Алгоритм Крускала [Электронный ресурс]. URL: https://e-maxx.ru/algo/mst\_kruskal (Дата обращения: 29.01.2024).
15. Минимальное остовное дерево. Алгоритм Прима [Электронный ресурс]. URL: https://e-maxx.ru/algo/mst\_prim (Дата обращения: 29.01.2024).
16. Топологическая сортировка [Электронный ресурс]. URL: https://e-maxx.ru/algo/topological\_sort (Дата обращения: 30.01.2024).