Логические алгоритмы поиска крИТИЧЕСКИХ путей в графе Корчажкина О.М. ([olgakomax@gmail.com](mailto:olgakomax@gmail.com))

*ФИЦ «Информатика и управление» РАН, г. Москва*

Аннотация

В статье делается краткий обзор функций, заложенных в логических алгоритмах работы с графами по поиску критических путей, оптимизирующих различные параметры задачи. Алгоритмы представлены в описательной форме, позволяющей без привязки к конкретным языкам программирования установить границы их применимости для решения задач, доступных для освоения на уроках информатики школьниками старших классов. Приводятся ссылки на оригинальные отечественные и переводные зарубежные источники по рассматриваемым логическим алгоритмам с методическими рекомендациями для учителя, а также примерами упражнений и решения задач в виде схем алгоритмов и «листингов» программных кодов.

Язык графов (а сейчас уже можно говорить о технологии применения графов для решения многих классов переборных задач как об особом языке дискретной математики) помогает рассматривать проблемы оптимизации реальной системы по определённым параметрам через представление её упрощённой визуальной модели в виде графа – планиметрической сети, состоящей из некоторого числа узлов (вершин), попарно соединённых рёбрами (путями), по которым можно построить множество маршрутов между узлами. Подобные модели в виде графа используются для решения большого числа «жизненных» задач.

Известно, что первой проблемой, к решению которой была привлечена теория графов как отдельная ветвь математической науки, стала проблема мостов Кёнигсберга. Она возникла ещё в XVIII веке и была решена гениальным швейцарским математиком Леонардом Эйлером (1707-1783), с 1727 по 1741 год и с 1766 года до конца жизни работавшим в России. Будучи в Кёнигсберге, Эйлер прогуливался по мостам столицы тогдашней Восточной Пруссии, перекинутым через речку Прегель (по другим источникам прогуливался не Эйлер, а Иммануил Кант), и, желая обойти все мосты, пройдя по каждому из них только один раз, так и не смог исполнить свою задумку.

Отметим, что в Кёнигсберге к 1730-му году таких мостов насчитывалось семь (рис. 1), а к настоящему времени из старых мостов сохранилось только четыре, поскольку два моста, разрушенные во время Великой Отечественной войны, так и не были восстановлены, третий мост был перестроен, а два коротких моста объединены в один [1, с. 65]. Тем не менее, классическая задача о кёнигсбергских мостах сохранилась с условием XVIII века [1, с. 63; 2, c. 234-235; 3, с. 66]. Соответствующий граф изображён на рис. 2, где узлы обозначают участки суши, а рёбра – мосты, то есть пути передвижения между участками суши.

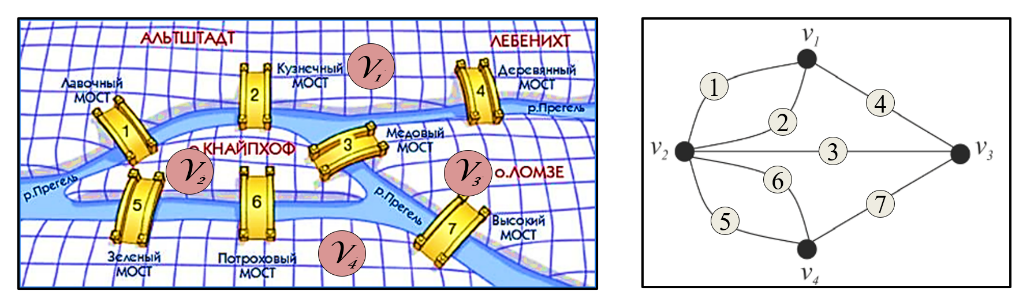


Рис. 1. Мосты Кёнигсберга Рис. 2. Граф «Мосты Кёнигсберга»

Введя понятие графа в 1735 году, Эйлер доказал, что все семь мостов Кёнигсберга невозможно обойти в «один заход», поскольку все четыре узла (участка суши) имеют нечётную степень: к участкам *V1*, *V3*и *V4* ведут по три моста, а к участку *V2*– пять.

Рассмотрим несколько алгоритмов решения задач, в которых требуется найти оптимальный маршрут, исходя из различных начальных условий, с помощью теории графов, не акцентируя ориентацию на конкретные языки программирования. При решении таких задач вводится обобщённый термин «критический путь» [4, с. 58-59], вобравший в себя множество значений параметров, которые предполагается подвергнуть оптимизации в ходе решения задачи – физических, материальных или энергетических затрат, сроков выполнения заданий, объёмов продаж, времени поездки, пройденного расстояния и пр.

Каждому школьнику известно, что любую задачу можно решить разными способами – сложными, простыми, рациональными, нерациональными, красивыми и не очень. Алгоритмы, применяемые в информатике к решению одной и той же задачи, могут иметь не только разную степень сложности, но и разные границы применимости. Именно степень сложности/простоты алгоритма и границы, в рамках которых этот алгоритм действует, позволяет отнести задачу к сложной или простой, что определяется эффективностью найденного алгоритма, способного оптимизировать затраченное время, объём памяти, вычислительные мощности цифровых устройств, конфигурацию и глубину дерева, отображающего этапы алгоритма и определяющего время его работы, и другие параметры [3, с. 56, 72].

Алгоритм Беллмана-Форда – это алгоритм поиска кратчайшего пути из одной вершины до всех остальных во взвешенном ориентированном графе, то есть в графе, где каждый путь маркирован весовым коэффициентом и имеет одностороннюю направленность. Коэффициенты могут выбираться исходя из времени преодоления расстояния между двумя узлами за счёт скорости движения транспортных средств (минимизируется время пути), длины конкретного участка пути (минимизируется расстояние), стоимости поездки (минимизируются материальные затраты), наличия препятствий (минимизируются физические усилия человека или износ транспортного средства), расхода топлива (минимизируются энергетические затраты) и пр. Основные принципы реализации алгоритма Беллмана-Форда состоят в том, что на основе матрицы смежности (массива весовых коэффициентов) происходит обновление оценки для всех рёбер на каждом шаге итерации [2, с. 117]. В отличие от алгоритма Дейкстры (см. ниже) алгоритм Беллмана – Форда допускает рёбра с отрицательным весом, что может соответствовать некоторым дополнительным условиям задачи и в ряде ситуаций являться весьма полезным [2, с. 108] (см. подробнее [2, с. 114-118; 5, с. 114-119; 6, с. 226-227]).

Алгоритм Дейкстры поиска кратчайших путей в графе является модификацией алгоритма поиска в ширину по рёбрам графа. Фиксируется начало маршрута – некоторый стартовый узел (вершина-источник), затем узлы графа разбиваются на слои, отстоящие от стартового узла на расстояние 1, 2, 3 и т. д. [2, с. 106]. В каждой такой промежуточной точке ребра ставится «будильник», заведённый на время прибытия уже начатого движения. «Звонки» будильника сигнализируют о достижении некоторой вершины исходного графа, из которой начинаются новые движения по исходящим из неё рёбрам. При этом требуется «перезавести» будильники, расположенные на концах этих рёбер. Таким образом, алгоритм Дейкстры находит кратчайшие пути во взвешенных ориентированных графах с положительными целыми весами. В отличие от алгоритма Беллмана-Форда этот алгоритм не допускает возвращение к узлу, который был ранее помечен как посещённый даже в тех случаях, когда существует более короткий путь, поэтому алгоритм Дейкстры может применяться только в тех графах, где отсутствуют рёбра с отрицательным весом (см. подробнее [2, с. 108-113; 6, с. 227-228]).

Алгоритм Флойда-Уоршелла – это алгоритм динамического программирования, используемый для нахождения длин кратчайших путей между всеми парами вершин во взвешенном ориентированном графе. Создаётся матрица смежности графа, элементами которой являются длины кратчайших путей между соседними вершинами графа. При переходе через каждую новую вершину (при каждом новом шаге итерации) проверяется, можно ли сократить расстояние между исходными вершинами (значение каждого элемента матрицы): если ДА, то происходит обновление матрицы. По окончании движения формируется матрица, содержащая кратчайшие расстояния между всеми парами вершин в графе. Таким образом, алгоритм Флойда-Уоршелла основан на пошаговой проверке, какие из возможных промежуточных вершин могут улучшить расстояние между двумя другими вершинами (см. подробнее [2, с. 169-171; 5, с. 106-113; 6, с. 231-233]).

Алгоритм Крускала – «жадный» алгоритм построения минимального остовного (покрывающего) дерева – взвешенного связного неориентированного ациклического графа (графы такого вида и называются деревьями), имеющего минимальный возможный вес, где под весом дерева понимается сумма весов входящих в него рёбер. Остовным деревом графа называется связный граф, состоящий из одних вершин, в котором есть пути из любой вершины в любую другую вершину, но нет контуров. Такой алгоритм решает, например, задачу создания связного графа компьютерной сети, в которой вершины (компьютеры) соединены неориентированными рёбрами, имитирующими электронные (физические или беспроводные) связи между компьютерами для обмена данными и совместного использования ресурсов. Для каждого соединения задана стоимость обслуживания и требуется найти наиболее бюджетный вариант (вот почему алгоритм называется «жадным»). Согласно этому алгоритму на каждом шаге итерации выбирается ребро наименьшего веса, не создающее цикла. Поиск прекращается, когда в результате получается дерево с минимальной стоимостью (см. подробнее [2, с. 127-138]).

Алгоритм Прима является альтернативой алгоритму Крускала, поскольку также решает задачу создания минимального покрывающего дерева, для которого задаётся множества рёбер некоторого поддерева и множество вершин, принадлежащих этим рёбрам. На каждом шаге итерации к имеющемуся дереву добавляется дополнительное ребро наименьшего веса, а к множеству вершин – соответствующая этому ребру новая вершина. Поиск прекращается, когда в множество вершин «войдёт» вершина ребра с минимальным весом (см. подробнее [2, с. 138-139]). Нетрудно увидеть, что алгоритм Прима имеет практически тот же самый псевдокод, что и алгоритм Дейкстры, поэтому с ним можно ознакомить учащихся как с усложнённым вариантом алгоритма Дейкстры. Основным отличием двух алгоритмов является задание приоритета: в алгоритме Прима приоритет отдаётся вершине, «поступившей» от ребра с минимальным весом, а в алгоритме Дейкстры приоритетным является полный путь с минимальным суммарным весом.

В заключение отметим, что каждый алгоритм следует разбирать с учащимися пошагово, демонстрируя его работу на коротких и простых примерах: сначала в виде схемы, а затем переходить к «листингам». Особенно это касается алгоритмов Беллмана-Форда и Флойда-Уоршелла, которые авторы пособия по динамическому программированию [5, с. 106] советуют начинать после освоения базового материала этих алгоритмов на графах (то есть «вручную»): изучения способов представления графов, алгоритмов обхода и алгоритма Дейкстры, являющегося самым доступным для понимания школьников, введения понятий ориентированных и неориентированных, взвешенных и невзвешенных графов. Рекомендуется также изучить примеры реальных задач, решаемых с помощью теории графов: задачу о кёнигсбергских мостах, задачу коммивояжёра, задачу об оптимизации объёма продаж, о времени пребывания самолётов в аэропорту, о транспортной сети города, о сети коммуникаций и пр.

К алгоритмам Крускала и Прима, которые не входят в программу по информатике для 10-11-х классов, поскольку являются весьма сложными для учащихся массовой школы, рекомендуется переходить на факультативных занятиях или на уроках по углублённому курсу информатики в профильных классах. Возможно также знакомство учащихся с этими алгоритмами через решение реальных задач в качестве демонстрации потенциала теории графов и его реализации современными методами динамического и линейного программирования.

Старшеклассникам, изучающим информатику по углублённой программе и в качестве серьёзной подготовки к олимпиадам или к поступлению в профильные вузы под руководством опытных преподавателей, будут, несомненно, интересны упражнения из классического учебника по алгоритмам [2], позволяющие закрепить изученный материал по темам: «Пути в графах» [2, с. 74-82], «Жадные алгоритмы» [2, с. 148-155], «Динамическое программирование» [2, с. 175-183], «Линейное программирование» [2, с. 220-229].

В качестве источника дополнительных материалов для самостоятельной или кружковой работы можно также порекомендовать учебное пособие С.М. Окулова «Дискретная математика. Теория и практика решения задач по информатике» [6]. В нём помимо богатого теоретического материала приведены практические задания – упражнения и задачи по поиску критических путей в графе с «набросками» решений, к которым обращаются как к исходным данным для исследования алгоритмов решаемых задач [6, с. 234-235; 349-351]. Автор пособия предлагает в самых сложных случаях, «когда логика программной реализации неочевидна для читателя, <…> выполнять её ручную трассировку – составлять таблицу изменений значений основных переменных в процессе работы программы» [6, с. 263].

Литература

1. Мир математики: в 40 т. Т. 25: Хоакин Наварро. Неуловимые идеи и вечные теоремы. Великие задачи математики. / Пер с исп. М., Де Агостини, 2014. 160 с.
2. Дасгупта С., Пападимитриу Х., Вазирани У. Алгоритмы / Пер. с англ. под ред. А. Шеня. – 3-е изд., стереотип. М., МЦНМО, 2023. 320 с.
3. Мир математики: в 45 т. Т. 43: Луис Фернандо Ареан. Существуют ли неразрешимые проблемы? Математика, сложность и вычисление. / Пер с исп. М., Де Агостини, 2014. 144 с.
4. Мир математики: в 40 т. Т. 11: Клауди Альсина. Карты метро и нейронные сети. Теория графов. / Пер с исп. М., Де Агостини, 2014. 144 с.
5. Бутарев К.В., Павлов Д.И. Введение в динамическое программирование: учебно-методическое пособие. М., МПГУ, 2024. 144 с.
6. Окулов С.М. Дискретная математика. Теория и практика решения задач по информатике: учебное пособие. М., БИНОМ. Лаборатория знаний, 2008. 422 с.