ВВЕДЕНИЕ. ЦЕЛИ И ЗАДАЧИ ДИСЦИПЛИНЫ

Все математические дисциплины можно условно разделить на *д и с к р е т н ы е* и *н е п р е р ы в н ы е*. Дискретная математика – это та часть математики, главной особенностью которой является изучение отдельных объектов, без привлечения понятия непрерывности, т.е. дискретность – это антипод непрерывности. В дискретной математике отсутствует понятие предельного перехода, присущее классической, «непрерывной» математике. Она занимается изучением дискретных структур, которые возникают как внутри математики, так и в ее приложениях. Однако она зародилась в глубокой древности, раньше, чем непрерывная математика, хотя особую значимость приобрела только в последние десятилетия, в связи с повсеместным внедрением в практику информационных технологий.

Таким образом, в широком смысле дискретная математика включает в себя все разделы математики, в которых не используются топологические методы, в частности понятие непрерывности. Это – все разделы алгебры, математическая логика, почти вся теория чисел (в том числе всевозможные компьютерные арифметики), многие разделы экономико-математических методов, комбинаторика и многие другие дисциплины. В более узком смысле дискретная математика – это те разделы математической логики, алгебры, теории чисел и математической кибернетики, которые непосредственно составляют теоретический фундамент информатики. В этом узком смысле дискретная математика включает в себя теорию булевых функций и их минимизацию, теорию графов и многие разделы теоретической кибернетики, теорию автоматов и формальных грамматик, комбинаторику, теорию алгоритмов (в том числе теорию сложности вычислений), криптографию и теорию кодирования.

Некоторые из вышеперечисленных разделов имеют не только многочисленные «внутренние» (с точки зрения специалиста по информационным системам или вычислительной техники) приложения, используемые, к примеру, при построении различных дискретных устройств, в программировании и т.д., но их результаты и методы применяются также при решении многих нужных для практики задач. Например, при рассмотрении транспортных задач, для нахождения оптимальных решений в управлении, для выделения «узких мест» при планировании и разработке проектов, при составлении оптимальных расписаний, а также при моделировании сложных технологий и процессов различной природы.

Целью изучения дисциплины является ознакомление студентов с системой понятий и некоторыми наиболее важными в приложениях методами теории множеств, математической логики, теории булевых функций и теории графов. Знания и навыки, полученные при ее изучении, используются в дисциплинах: «Информатика», «Программирование», «Структуры и алгоритмы обработки данных в ЭВМ», «Базы данных», «Экспертные и интеллектуальные системы» и т.д. Но в особенности знания по дискретной математике пригодятся

при изучении дисциплин, связанных с функциональным и логическим программированием, кодированием и защитой информации.

Основная задача состоит в том, чтобы будущие специалисты чётко освоили основные понятия и приёмы работы с булевыми функциями и графами: построение таблиц значений; поиск и исключение фиктивных переменных; приведение булевых функций к стандартной форме (д.н.ф., к.н.ф., многочлен Жегалкина); основные методы минимизации булевых функций; построение диаграммы (рисунка) графа по его матрицам смежности и инцидентности и обратная задача; установление изоморфизма (одинаковости) графов; определение основных характеристик и свойств графов (векторы степеней, планарность, эйлеровость, гамильтоновость и т.п.); изучение важного частного случая графов – деревьев и их свойств.

За недостатком места о приложениях говорится относительно мало. Однако такие примеры содержатся в литературе.

Данное пособие предназначено в основном для изучения основ именно дискретной математики в узком понимании слова, хотя при этом затронуты основополагающие разделы математической логики – исчисление высказываний и исчисление предикатов. Однако математическую логику настоятельно рекомендуется изучать по более фундаментальным источникам, например, [1, 11,15,16,19,23,29]. В то же время, многие разделы дискретной математики в узком смысле слова в данном пособии никак не отражены, в частности, теория кодирования и криптография, теория алгоритмов и теория сложности вычислений. Это связано, в первую очередь, с ограниченностью отводимого времени для изучения дисциплины в учебных планах у студентов, обучающихся информационным технологиям и использованию вычислительной техники. Курс лекций будет также полезен будущим специалистам по прикладной математике, в частности по математическому и компьютерному моделированию.

Пособие – это существенно поработанный и дополненный вариант пособий [20,21].

ЧАСТЬ ПЕРВАЯ. ЭЛЕМЕНТЫ ТЕОРИИ МНОЖЕСТВ И МАТЕМАТИЧЕСКОЙ ЛОГИКИ

См. лекции 1-5

ЧАСТЬ ВТОРАЯ. БУЛЕВЫ ФУНКЦИИ И ИХ МИНИМИЗАЦИЯ

См. лекции 6-11

ЧАСТЬ ТРЕТЬЯ. ЭЛЕМЕНТЫ ТЕОРИИ ГРАФОВ

Графы имеют многочисленные применения в самых различных областях человеческой деятельности. И это естественно, ибо если граф относительно небольшой (размерность задачи маленькая), то мы имеем возможность нарисовать его, точнее изобразить его схему или диаграмму. В этом случае во многих ситуациях задача становится практически понятной. С другой стороны, для работы с большими графами можно с успехом применять ЭВМ. С этой целью разработаны и удобные способы представления графов в ЭВМ, и разнообразные алгоритмы, позволяющие решать широкий круг задач.

11 ОСНОВНЫЕ ПОНЯТИЯ ТЕОРИИ ГРАФОВ

12 ОБХОДЫ ГРАФОВ. ПЛАНАРНОСТЬ. ХАРАКТЕРИЗАЦИЯ ДЕРЕВЬЕВ. РАСКРАСКИ

См. лекции 12-14

13 ОПИСАНИЕ НЕКОТОРЫХ АЛГОРИТМОВ НА ГРАФАХ

В этом разделе описываются относительно простые алгоритмы, решающие некоторые задачи на взвешенных графах. Все эти алгоритмы относятся, к так называемым, *ж а д н ы м* алгоритмам – на каждом шаге таких алгоритмов выбирается объект с наилучшими характеристиками из всех, что доступны на этом шаге.

3.1 Нахождение остова (каркаса) наибольшего (наименьшего) веса

Задача о нахождении остова (каркаса) наибольшего (наименьшего) веса ставится во взвешенном связном неориентированном графе. К ней сводятся некоторые прикладные задачи. Опишем её для случая построения каркаса максимального веса (при нахождении остова минимального веса поступаем точно также с заменой «максимум» на «минимум»).

Итак, нам нужно в данном взвешенном графе найти его подграф максимального веса так, чтобы он был деревом и содержал в себе все вершины графа.

13.1.1 О п и с а н и е а л г о р и т м а П р и м а. Первым берётся ребро графа наибольшего веса, затем отыскивается следующее ребро наибольшего веса, образующее вместе с построенной частью дерево и т.д., до тех пор, пока все вершины графа не окажутся в построенной части. Данный алгоритм достаточно легко применяется при визуальной работе с графами – наглядно видно, образуется ли дерево или нет.

Машинная реализация его также несложна, хотя и требует некоторой сноровки. В программу вводятся данные о графе – начальная, конечная вершина каждого ребра и веса всех рёбер (например, матрицей весов или списком взвешенных рёбер – см. подраздел 11.3). Затем ребра сортируются по убыванию веса. Из отсортированного списка рёбер последовательно выбираются те, у которых не задействована ещё ровно одна вершина, за исключением, разумеется, первого в списке (задействованные вершины помещаются в отдельный список). Таким образом, удается избежать возникновения циклов, а значит, построенный граф является деревом.

13.1.2 О п и с а н и е а л г о р и т м а К р а с к а л а. Этот алгоритм отличается от алгоритма Прима тем, что добавляемое на очередном шаге ребро может не образовывать с уже построенной частью дерева. Вследствие этого, после того как все вершины графа окажутся в построенной части, нужно либо убедиться в том, что построенный подграф – связный, либо добавить ещё рёбра наибольшего веса из оставшихся, чтобы связать отдельные построенные куски. Программная реализация этого алгоритма немного сложнее, чем у предыдущего.

П р и м е р ы 13.1 Приведём решение этой задачи обоими алгоритмами для одного и того же графа G_{10}, изображённого на рисунке 13.1 слева. Здесь

они расписаны по шагам, и для наглядности размещены параллельно, но кратко без объяснения, почему на этом шаге нужно поступать именно так, а не иначе.

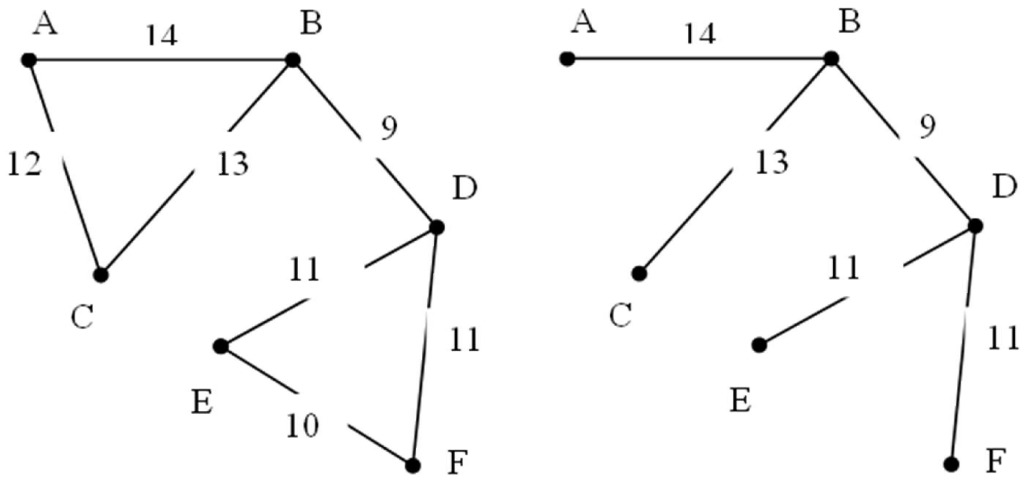

Рисунок 13.1 – Слева граф G_{10}, справа его каркас максимального веса

Таблица 13.1 – Нахождение максимального остова графа G_{10} двумя алгоритмами

Алгоритм Прима	Номер шага	Алгоритм Краскала
Выбираем ребро AB, вес 14.	1	Выбираем ребро AB, вес 14.
Присоединяем к нему ребро BC веса 13.	2	Присоединяем к нему ребро BC веса 13.
Хочется взять ребро AC веса 12, но нельзя (почему?), поэтому берём ребро BD веса 9 (а почему не другие – у них же вес больше?).	3	Хочется взять ребро AC веса 12, но нельзя (почему?), поэтому берём на выбор любое из рёбер DE или DF, у них одинаковый вес – 11, выбираем из них DE.
Можно взять на выбор любое из рёбер DE или DF, у них одинаковый вес – 11, выбираем из них DE.	4	Присоединяем ребро DF веса 11. Так как все вершины графа теперь включены в построенную часть, то хочется завершить работу, но нельзя (почему?).
Присоединяем ребро DF веса 11. Так как все вершины графа теперь включены в построенную часть, то алгоритм завершает работу.	5	Добавляем ребро BD веса 9 (а почему не другие – у них же вес больше?). Теперь можно завершить работу (почему?).

В результате работы обоих алгоритмов получился один и тот же остов: AB, BC, BD, DE, DF веса 14+13+9+11+11=58. Для данного графа каркас наибольшего веса единственный, однако, это бывает не всегда см. упражнение

13.1 ниже.

Упражнение 13.1 *Найдите, этими алгоритмами для графа* G_{10} *остов минимального веса. Вполне возможно, что Вы получите два разных каркаса, в этом нет ничего страшного – этот граф действительно два различных минимальных остова. Но какой бы разновидностью этих алгоритмов Вы ни пользовались, вес у этих каркасов должен быть одинаковый.*

13.2 Задача о кратчайшем пути в связном орграфе. Алгоритм Дейкстры

Наиболее распространёнными и востребованными в практических изысканиях являются задачи на отыскание экстремумов – минимальных или максимальных значений каких-то величин, или нахождению стратегии дающей оптимальное значение каких-то параметров. Многие из них можно свести к нахождению кратчайшего пути в ориентированном графе.

13.2.1 Вначале уточним некоторые термины и понятия, относящиеся к ориентированным графам.

Для ориентированных графов различают *сильную связность*, когда из каждой вершины существует маршрут в любую другую вдоль стрелок (по направлению рёбер), и *слабую связность*, если можно проходить против направлений дуг. Другими словами, слабо связный орграф (*слабый орграф*) – это орграф, который после дезориентации дуг будет связным.– орграф. Сильно связные орграфы называют ещё *сильными орграфами*.

На практике наиболее применимы орграфы, в некотором смысле промежуточные между сильными и слабыми орграфами. *Односторонне связный* орграф (или *односторонний орграф*) – это орграф, у которого любая пара вершин односторонне связна, т.е. из некоторых вершин можно дойти до всех остальных по направлению рёбер, а обратного пути может и не быть.

Многополюсная сеть (*многополюсник*) – орграф с выделенными вершинами – *полюсами*. Причём достаточно часто подразумевается, что этот орграф односторонне связен, точнее, что из некоторых выделенных вершин (из *начальных*) можно дойти до всех остальных по направлению стрелок. *Двухполюсная сеть* – сеть с двумя выделенными вершинами. Обычно подразумевается, что из одной этих вершин, из *начала*, достижима по стрелкам любая другая, а второй полюс – *концевой*, наоборот, достижим из всех остальных.

13.2.2 Задача о кратчайшем пути между двумя вершинами – начальной s и конечной t во взвешенном односторонне связном орграфе может решаться разными способами [1,2,4,17,21].

Здесь описывается один из самых простых – *алгоритм Дейкстры*, для случая когда все веса на рёбрах (дугах) – неотрицательные.

Пусть на ребре, идущем из вершины v в вершину p имеется вес $w(v,p)$, при этом если дуга из вершины v в вершину p отсутствует, то полагаем, что $w(v,p) = \infty$.

Описание алгоритма. На каждом шаге этого алгоритма всякая вершина v графа G имеет метку $l(v)$, которая может быть *постоянной* или *временной*. В первом случае $l(v)$ – вес кратчайшего пути из вершины s в вершину v (т.е. вес (s,v)-пути). Если же метка временная, то $l(v)$ – вес кратчайшего (s,v)-пути, проходящего только через вершины с постоянными метками. Таким образом, временная метка $l(v)$ является оценкой сверху для веса кратчайшего (s,v)-пути, и став на некотором шаге постоянной, она остаётся такой до конца. Кроме $l(v)$, с каждой вершиной графа, за исключением исходной s, связывается ещё одна метка – $\theta(v)$. На каждом шаге $\theta(v)$ является номером вершины, предшествующей v в (s,v)-пути, имеющем минимальный вес среди всех (s,v)-путей, проходящих через вершины, получившие к данному моменту постоянные метки. После того как вершина t получила постоянную метку, с помощью меток $\theta(v)$ обратным ходом восстанавливается последовательность вершин, составляющих кратчайший (s,t)-путь.

На нулевом шаге алгоритма вершине s присваивается постоянная метка $l(s) = 0$, остальным вершинам v – временные метки $\ell(v) = \infty$. Меток $\theta(v)$ пока нет. Каждый следующий шаг алгоритма состоит в следующем. Пусть p – вершина, получившая постоянную метку $l(p)$ на предыдущем шаге. Просматриваем все вершины графа v, имеющие временные метки $l(v)$ с целью уменьшения (если это возможно) этих меток. Метка $l(v)$ вершины v заменяется на $l(p)+w(p,v)$, если оказалось, что $l(v) > l(p)+w(p,v)$. В этом случае говорят, что вершина v получила свою метку l из вершины p (или, что мы пришли в v из вершины p), и полагают $\theta(v) = p$. Если же $l(v) \le l(p)+w(p,v)$, то метки θ и l вершины v не изменяются на данном шаге. Шаг заканчивается выбором такой вершины v_1, у которой метка $l(v)$ минимальная среди всех вершин с временными метками. Метки $l(v_1)$ и $\theta(v_1)$ вершины v_1 делают постоянными и переходят к следующему этапу.

Пример 13.2 Применим алгоритм Дейкстры для нахождения кратчайшего пути между вершинами $s = A_1$ и $t = A_6$ графа, изображённого на рисунке 13.2

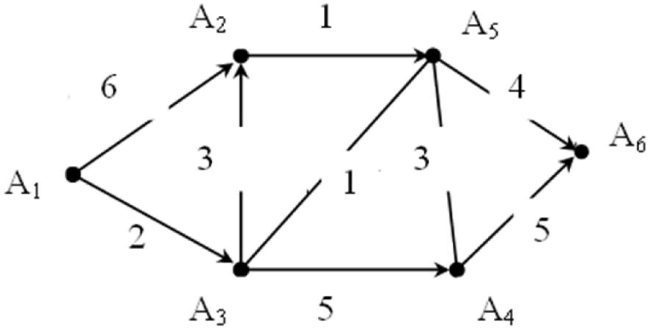

Рисунок 13.2 – Исходная сеть. Дуги без стрелок считаются двусторонними

Шаг 0. Присваиваем вершине A_1 постоянную метку $l = 0$, остальным вершинам – временные метки $l = \infty$.

Шаг 1. У вершин, которые непосредственно не связаны рёбрами с построенной частью (в данном случае – это вершина A_1), метки измениться не могут. Поэтому рассматриваем только

соседние с построенной частью вершины A_2 и A_3. Для A_2: $l(A_2) = \infty$, а $l(A_1)+w(A_1,A_2) = 0+6 = 6 < \infty$, поэтому метку $l(A_2)$ меняем на 6, кроме того вершине A_2 присваиваем метку $\theta = 1$. Точно также, $l(A_3) := 2$, $\theta(A_3) = 1$. Ищем $min\{l(A_2),l(A_3),l(A_4),l(A_5),l(A_6)\} = min\{6,2,\infty,\infty,\infty\}$. Он равен $l(A_3) = 2$. Ввиду этого метку $l(A_3) = 2$ объявляем постоянной.

Шаг 2. Соседними с вершинами, имеющими постоянные метки (с построенной частью), являются вершины A_2, A_4, A_5. Для A_2 имеем: $l(A_2) = 6$, $l(A_3) + w(A_3,A_2) = 2+3 = 5$, $min(6,5) = 5$, поэтому метку $l(A_2)$ заменяем на 5 и полагаем $\theta(A_2) = 3$.

Для A_4: $l(A_4)=\infty$, $l(A_3)+w(A_3,A_4)=2+5=7$, $min(\infty,7)=7 \Rightarrow$ новые метки: $l(A_4)=7$, $\theta(A_4) = 3$.

Для A_5: $l(A_5)=\infty$, $l(A_3)+w(A_3,A_5)=2+1=3$; $min(\infty,3)=3 \Rightarrow$ новые метки: $l(A_5)=3$, $\theta(A_5) = 3$.

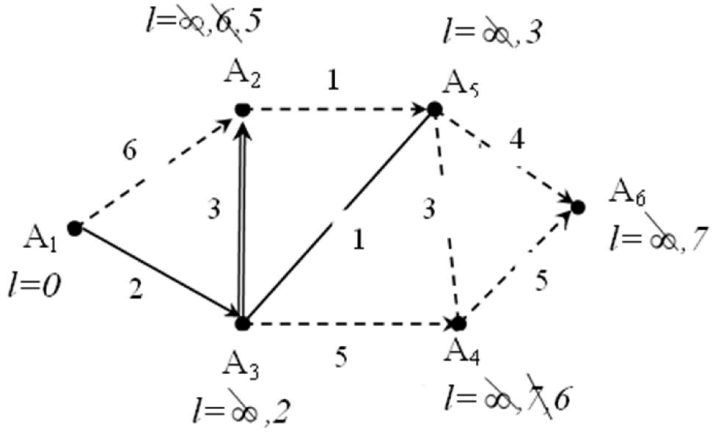

Рисунок 13.3 – Ситуация после шага 3, добавленное на этом шаге ребро выделено двойной стрелкой

Наконец, наименьшая временная метка l – у вершины A_5, её делаем постоянной и переходим к следующему шагу.

Шаг 3. Рассматриваем соседние с построенной частью вершины A_2, A_4 и A_6.

Для A_2 имеем: $l(A_2) = 5$, $l(A_5) + w(A_5,A_2) = 3 + \infty = \infty$; $min(5,\infty)=5 \Rightarrow$ метки $l(A_2)$ и $\theta(A_2)$ оставляем прежними.

Для A_4: $l(A_4) = 7$, $l(A_5) + w(A_5,A_4) = 3+3 = 6$, $min(7,6) = 6 \Rightarrow$ новые метки: $l(A_4) = 6$, $\theta(A_4) = 5$.

Для A_6: $l(A_6)=\infty$, $l(A_5) + w(A_5,A_4) = 3+4 = 7$, $min(\infty,7) = 7 \Rightarrow$ новые метки: $l(A_6) = 7$, $\theta(A_6) = 5$.

Минимальная среди временных меток – $l(A_2) = 5$, её делаем постоянной.

Шаг 4. Соседние с построенной частью вершины A_4 и A_6. Рассматриваем A_4: $l(A_4) = 6$, $l(A_2) + w(A_2,A_4) = 5+\infty = \infty$; $min(6,\infty) = 6 \Rightarrow$ метки $l(A_4)$ и $\theta(A_4)$ оставляем прежними. Для A_6: $l(A_6) = 7$, $l(A_2) + w(A_2,A_6) = 5+\infty = \infty$; $min(7,\infty) = 7 \Rightarrow$ метки $l(A_6)$ и $\theta(A_6)$ оставляем прежними. Наименьшая временная метка – $l(A_4) = 6$, делаем её постоянной.

Шаг 5. Осталась одна вершина A_6: $l(A_6)=7$, $l(A_4) + w(A_4,A_6) = 6+5 = 11$; $min(7,11) = 7 \Rightarrow$ метки $l(A_6)$ и $\theta(A_6)$ оставляем прежними, и $l(A_6) = 7$ объявляем постоянной. Алгоритм закончил работу.

Осталось по меткам θ восстановить кратчайший путь, (его длина уже известна – это $l(A_6) = 7$). Вспомним о смысле этих меток: $\theta(A_6) = 5$. Это

означает, что в вершину A_6 мы пришли из A_5, т.е. искомый маршрут имеет вид: $A_1 \Rightarrow \ldots \Rightarrow A_5 \Rightarrow A_6$. Далее, $\theta(A_5) = 3$, таким образом в A_5 нужно идти из A_3, и с учётом того, что $\theta(A_3) = 1$, окончательно получаем:

$A_1 \Rightarrow A_3 \Rightarrow A_5 \Rightarrow A_6$ – кратчайший путь из A_1 в A_6.

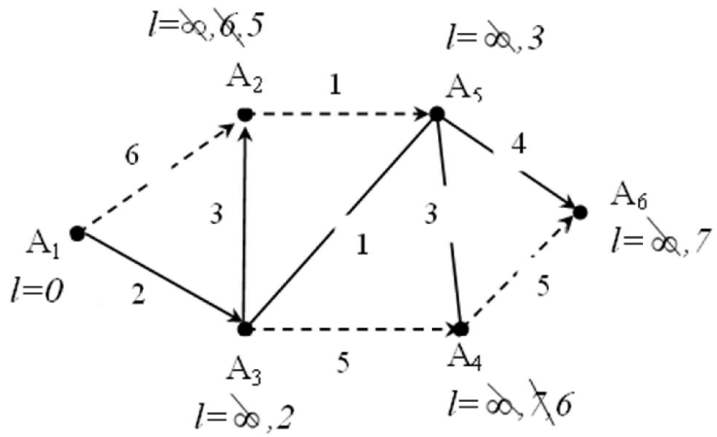

Рисунок 13.4 – Ситуация после окончания работы

Замечание. 13.1 Нетрудно увидеть, что попутно мы нашли кратчайшие пути из начальной вершины A_1 до всех вершин, которые к концу работы получили постоянные метки, в рассмотренном примере оказалось, что просто до всех остальных.