**Лекция 8**

**Математические основы сетевого планирования**

**Основные понятия теории графов**

Граф – это математическая модель, с помощью которой удобно представлять бинарное отношение. Хотя теория графов получила свое развитие задолго до появления теории множеств как самостоятельной дисциплины, большое число задач теории отношений формулируются и решаются в рамках именно этой теории.

Рассмотрим множество , которое будем назвать ***множеством*** ***вершин***, и набор  упорядоченных пар , называемый ***множеством дуг***. Пара  называется ***графом***. Запись  обозначает граф с именем , состоящий из множества вершин  и множества дуг .

Обычно вершины графа изображаются в виде точек, а дуги в виде соединяющих эти точки стрелок (рис 1.1).



Рис. 1.1. Изображение графа

На рис. 1.1 изображен граф , где  и .

Две соединенные дугой вершины графа, называются ***смежными вершинами***. Например,  и  смежные вершины графа , т.к. есть дуга .

Считается, что дуга  ***выходит из вершины***  и ***входит в вершину*** . При этом говорят, что вершины  и  ***инцидентны*** дуге , а дуга  инцидентна вершинам  и . Например, вершины  и  инцидентны дуге , выходящей из  и входящей в .

Вершины  и  дуги  называют соответственно ***начальной*** и ***конечной*** вершинами этой дуги.

Дуги, которые выходят и входят в одну и ту же вершину, называются ***петлями***. Например, дуга  является петлей.

Вершины, не имеющие смежных, называются ***изолированными вершинами***. Например,  – изолированная вершина.

Очевидно, что множество дуг  можно интерпретировать как бинарное отношение, а  – множество, на котором это бинарное отношение строится.

Если множество дуг  не является симметричным отношением, то такой граф называется ***ориентированным графом***. Граф, изображенный на рис. 1.1, является ориентированным графом.

Если множество дуг – симметричное отношение, то соответствующий граф называется ***неориентированным*** ***графом***. Симметричность отношения предполагает, что для каждой дуги  графа найдется дуга , имеющая противоположное направление.

На рис. 1.2 изображен граф, представляющий симметричное отношение , которое может быть описано следующей матрицей (матрица смежности):



Матрица симметричного бинарного отношения симметричная.



Рис. 1.2. Граф, представляющий симметричное отношение

Для того, чтобы разгрузить рисунок при изображении неориентированного графа, принято пару противоположных дуг изображать линией без стрелок, как это изображено на рис. 1.3.



Рис. 1.3. Неориентированный граф

Для того, чтобы подчеркнуть, что порядок вершин в неориентированном графе не имеет значения, при обозначении пар вершин, соединенных двумя противоположными дугами, используется запись с круглыми скобками: , а сами такие пары называют ***ребрами***.

В дальнейшем нас будут интересовать только ориентированные графы, т.к. именно они используются для моделирования сетевых графиков. Поэтому в дальнейшем изложение основ теории графов будет посвящено ориентированным графам.

***Способы представления графов***

Существует три основных способа представления графов: матрица смежности, матрица инцидентности и списки смежных вершин.

***Матрица смежности***

***Матрица инцидентности*** имеет размерность *m*×*n*, где  – количество вершин, а  – количество дуг. Каждый  элемент матрицы принимает одно из трех значений:  – вершина  не инцидентна дуге ;  – дуга  выходит из вершины ; – дуга  входит в вершину . Например, при заданной индексом нумерации ребер графа, изображенного на рис. 1.1 , его матрица инцидентности  будет следующей:



Следует обратить внимание, что петле в матрице  соответствует столбец с одной положительной единицей, что не всегда удобно при вычислениях. Например, при подсчете количества входящих в вершину дуг следует всегда учитывать, что может быть петля, которой нет соответствующей . Поэтому матрицы инцидентности применяются редко, особенно, если граф может иметь петли.

***Списки смежных вершин*** представляют собой набор из  множеств, соответствующих  вершинам графа. Множество, соответ-ствующее вершине ,  либо пустое, либо содержит номера вершин, в которые входит дуга, выходящая из вершины *i*. Например, списки смежных вершин для графа на рис. 1.1 можно представить в виде следующих пяти множеств: ; ;  . Следует отметить, что этот способ является наиболее удобным при программировании большинства задач теории графов.

При программировании реальных задач теории графов, как правило, применяются матрица смежности и списки смежных вершин. При этом часто этой информации бывает недостаточно, т.к. она отражает только структуру графа. Если с вершинами и/или дугами графа связаны какие-то дополнительные характеристики, необходимо предусмотреть возможность их хранения.

***Кратчайшие и максимальные пути между вершинами графа***

Пусть  – ориентированный граф, на множестве дуг которого определена функция . Функция  ставит каждой дуге  графа  в соответствие действительное число , которое часто называют ***весом дуги***. При этом саму функцию  обычно называют ***весовой функцией***, а граф – ***взвешенным графом***.

В общем случае граф  может содержать несколько путей , , …  из вершины  в вершину . ***Весом***  ***пути***  ***взвешенного графа*** *называется сумма весов дуг, составляющих этот путь: *.

***Кратчайшим путем  из вершины  в вершину *** *называется путь с минимальной весом* ******. В общем случае в одном графе может быть несколько минимальных путей.

Поиск кратчайшего пути между двумя вершинами графа является одной из часто используемых в приложениях задач. Наиболее известными способами решения этой задачи являются ***алгоритмы Дейк-стры***, ***Беллмана*** *–* ***Форда*** и ***Флойда – Уоршолла***. Здесь будет рассмотрен только алгоритм Дейкстры (Эдсгер Вибе Дейкстра, известный голландский ученый-математик, 1930–2002), с другими алгоритмами можно ознакомится в [9, 10].

Алгоритм Дейкстры решает задачу о кратчайших путях во взвешенном графе  с дугами неотрицательного веса , ,  из вершины  во все достижимые вершины этого графа. Алгоритм последовательно преобразовывает исходный граф  в дерево кратчайших путей . ***Дерево кратчайших путей*** – это граф , обладающий следующими свойствами:

1. ;
2.  – ориентированное дерево с корнем ;
3.  – множество достижимых вершин графа  из ;
4. для каждой вершины  путь из  в  в дереве  является кратчайшим путем в графе .

Алгоритм Дейкстры предполагает, что граф  задан с помощью списков смежных вершин, а также известна весовая функция , заданная на множестве *V×V*. Функция  ставит в соответствие каждой паре вершин  вес  дуги, соединяющей эти вершины, или , если такой дуги нет в графе . Кроме того, задана вершина , относительно которой определяются все кратчайшие пути.

Для пояснения работы алгоритма Дейкстры будем использовать следующие обозначения.

 – множество вершин графа , смежных с вершиной . По сути ,  – это списки смежных вершин графа , с помощью которых задан граф . Иначе  – это перечень всех вершин, которые являются конечными вершинами дуг с началом в .

– верхняя оценка кратчайшего пути из вершины  в вершину . В начале работы алгоритма , а для всех остальных  оценка .

 – вершина, предшествующая вершине  в дереве кратчайших путей. В начале работы алгоритма для всех  , где – специальный символ, обозначающий пустоту. После завершения работы алгоритма множество значений  позволяет построить дерево кратчайших путей.

– множество вершин . Вначале множество  пустое: . В процессе работы алгоритма  пополняется вершинами из , для которых уже определен кратчайший путь из вершины . После окончания работы алгоритма .

 – множество вершин . Вначале . В процессе работы алгоритма элементы этого множества – вершины, не добавленные во множество : . После окончания работы алгоритма .

 – процедура извлечения элемента из множества  с наименьшим текущим значением . При сравнении предполагается, что , где  – любое действительное число.

 – процедура релаксации, которая определена для произвольных двух вершин  графа . Релаксация состоит в следующем: значение  уменьшается до  в том случае, если значение  меньше текущего значения . При этом операция сложения предполагает, что , где  – любое действительное число или . Кроме того, если  уменьшилось, то значение  становится равным .

 – кратчайший путь из вершины  в вершину .

Выполнение алгоритма Дейкстры состоит из двух этапов. На первом этапе инициализируются , ,  и  значениями, описанными выше. Второй этап алгоритма опишем с помощью мнемокода, как показано на рис. 1.6. Для удобства строки мнемокода пронумерованы.

//-- Dijkstra

//-- вычисления кратчайших путей в графе

1 **while **

2 **{**

3 = 

4 ****

5 **for для всех **

6 **{**

7 ****

8  **}**

9  **}**

Рис. 1.6. Второй этап алгоритма Дейкстры

Основной цикл алгоритма (строки 1–9) на рис. 1.6 выполняется до тех пор, пока все вершины графа не будут извлечены из множества  (строка 1). Извлечение вершин из множества  осуществляется с помощью процедуры  (строка 3). Извлеченная вершина помещается сначала в переменную  затем во множество  (строка 4). Далее выполняется внутренний цикл (строки 5–8), в котором для всех вершин, имеющих входящие дуги с начальной вершиной , выполняется процедура релаксации.

Результатом выполнения алгоритма Дейкстры являются массивы  и , состоящие из  элементов. Массив  позволяет построить граф кратчайших путей, а каждый элемент массива  содержит вес кратчайшего пути между вершинами  и .

На рис. 1.7 приведен пример решения задачи поиска кратчайшего пути в графе с помощью алгоритма Дейкстры. На рис. 1.7, а изображен исходный граф и проинициализированные массивы , ,  и , назначение которых разъяснялось выше. В качестве меток для вершин графа используются числа от  до . Рис. 1.7, б – 1.7, е отражают процесс решения задачи. Каждый рисунок отражает состояние массивов после завершения очередной итерации основного цикла алгоритма.

В примере на рис. 1.7 осуществляется поиск кратчайших путей из вершины  до всех остальных вершин графа. По мере решения задачи, метки вершин графа перемещаются из массива  в массив . В массиве  формируется вес пути для каждой вершины, а в массиве  – список предшествующих вершин. Результат решения представлен на рис 1.7, ж и представляет собой дерево кратчайших путей. В этом дереве из вершины  до любой другой вершины графа существует единственный путь, который является кратчайшим.

При расчете временных характеристик сетевого графика, необходимо найти критический путь, определяющий минимальное время выполнения проекта. Отыскание критического, максимального, пути в графе сводится к поиску пути с самым большим весом, называемого максимальным путем в графе.

***Максимальным путем  называется путь с максимальным весом ***. В общем случае, в одном графе может быть несколько максимальных путей.

Построение максимального пути ****** во взвешенном ориентированном графе  возможно, если в нем нет контуров с положительным весом. Если в графе есть такой контур, то некоторые пути могут иметь сколь угодно большой вес, т.к. каждый обход контура увеличивает вес пути на величину веса этого контура.

Будем предполагать, что граф  безконтурный и его вершины пронумерованы так, что у любой дуги  конечная вершина всегда имеет больший номер, чем начальная: . Выше рассматривалась процедура ранжирования, основанная на алгоритме Демукрона, позволяющая привести любой безконтурный граф такому виду. Пусть все вершины графа  имеют номера от  до .

Рассматриваемый алгоритм основывается на рекуррентном выражении , где  – множество начальных вершин дуг, входящих в вершину . При этом предполагается, что для всех вершин  ранга  значение .

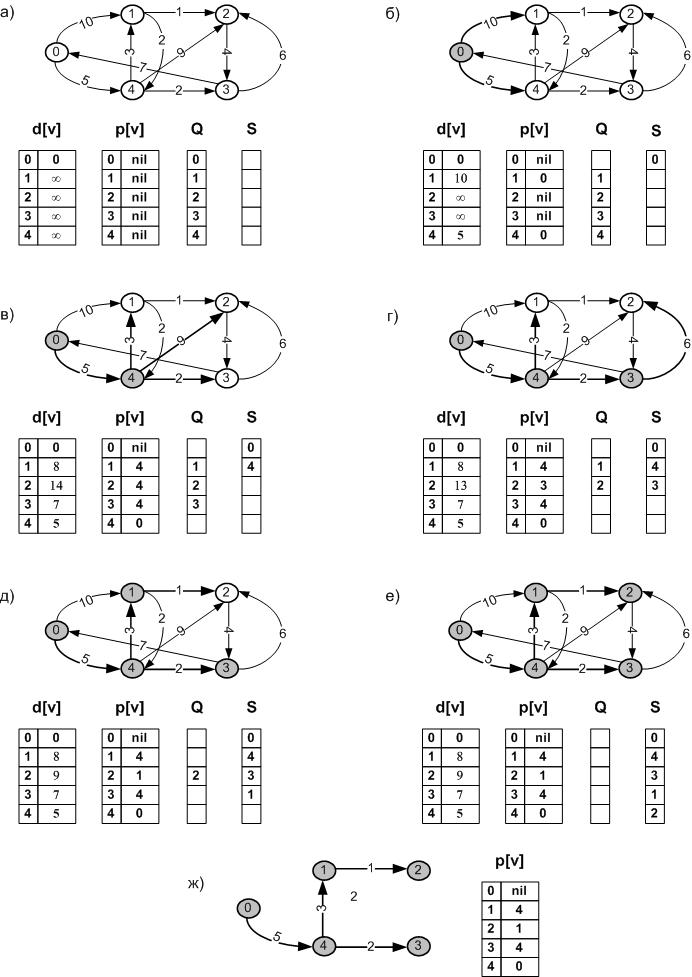


Рис. 1.7. Процесс построения дерева кратчайших путей алгоритмом Дейкстры

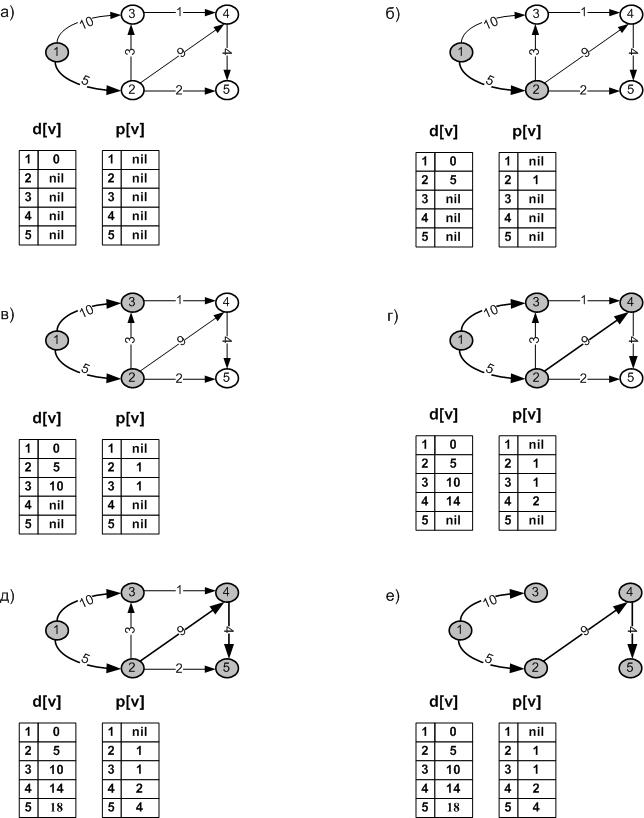


Рис. 1.8. Процесс построения максимального пути в графе

Вычисление значений  необходимо выполнить для каждой вершины графа в порядке возрастания номера. Каждое полученное значение  представляет собой максимальный вес пути в графе  с конечной вершиной .

Цикл вычисления  сопровождается построением массива элементов , которые формируются по тому же принципу, что и в алгоритме Дейкстры. Каждый элемент  соответствует вершине графа . Значение  – вершина (или ее номер), предшествующая вершине  в пути максимального веса с конечной вершиной .

На рис. 1.8 представлен пример решения задачи поиска максимального пути в графе.

На рис. 1.8, а изображен заданный граф и проинициализированные массивы  и . На рисунке 1.8, е представлен результат работы алгоритма. Для построения максимального пути в графе необходимо найти максимальный элемент в  (в нашем случае – это 18) и обратным порядком построить все предшествующие  вершины по массиву  (в нашем случае – это вершины: , , , ).