Лекция 9. КРАТЧАЙШИЕ ПУТИ В ГРАФЕ

План лекции:
1. Постановка задачи о кратчайшем пути.
2. Задача о кратчайшем пути в ненагруженном графе и волновой алгоритм ее решения.
3. Задача о кратчайшем пути в нагруженном графе и ее решение на основе алгоритма Дейкстры.

1. Постановка задачи о кратчайшем пути

В практических приложениях имеет большое значение задача о нахождении кратчайшего пути между двумя вершинами связного неориентированного графа.

Формулировка задачи может иметь несколько вариантов.

Вариант 1. Дана сеть автомобильных дорог, например, Тульской области. Найти кратчайшие пути от Тулы до каждого города области (если двигаться можно только по дорогам).

Вариант 2. Имеется некоторое количество авиарейсов между городами мира. Для каждого известна стоимость. Найти маршрут минимальной стоимости (возможно с пересадками), например, из Торонто в Новосибирск.

Вариант 3. Есть план города с нанесенными на него местами расположения пожарных частей. Определить ближайшую к каждому дому пожарную станцию.

В математике разработан ряд методов для решения подобных задач. Однако в большинстве случаев методы, основанные на использовании графов, оказываются наименее трудоемкими.

2. Задача о кратчайшем пути в ненагруженном графе и волновой алгоритм ее решения

Рассмотрим связный ненагруженный граф $G = (X, U)$, ребра которого имеют одинаковый вес (длину), принимаемый за единицу. Решим для этого графа следующую задачу: найти кратчайшую цепь, связывающую заданную начальную вершину a с заданной конечной вершиной b.

Поскольку рассматриваемые графы сравнительно просты, то кратчайший путь нетрудно найти просто путем перебора возможных путей. Однако для сложных графов должен быть найден систематический метод.

Общее правило для нахождения кратчайшего пути в графе состоит в том, чтобы каждой вершине x_i присвоить индекс λ_i, равный длине кратчайшего пути из данной вершины в начальную. Присваивание индексов производится в следующем порядке:

1) начальной вершине a присваивается индекс $\lambda_a = 0$;
2) всем вершинам, смежным с вершиной a, присваиваем индекс 1;
3) всем вершинам, смежными с вершинами, имеющими индекс 1, присваиваем индекс 2, и так далее, пока не будет помечена вершина b. Сам кратчайший путь найдем, если будем двигаться из конечной вершины в направлении убывания индексов.

На рис. 1 показаны индексы, которые получают вершины в процессе работы алгоритма. Так как вершина b получила отметку 7, то длина кратчайшей цепи от a до b равна 7. Эта цепь выделена на рисунке.

Описанный алгоритм называют *волновым*, так как процесс расстановки отметок напоминает распространение возмущения, которое возникает в вершине a и движется со

скоростью одно ребро в единицу времени. Вершины, имеющие одинаковые отметки, представляют собой фронт волны.

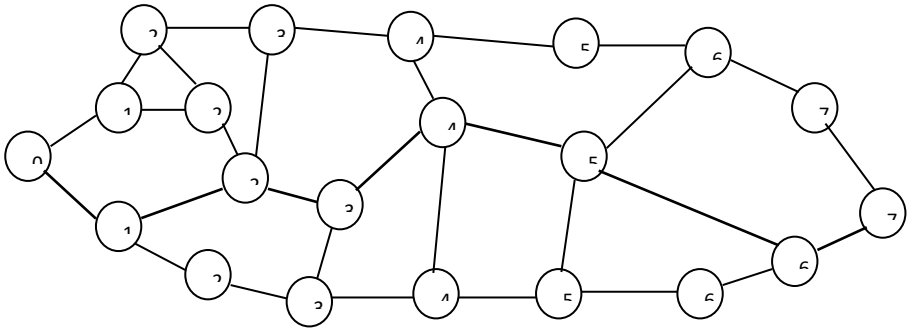


Рис. 1. Отыскание кратчайшего пути в ненагруженном графе

3. Задача о кратчайшем пути в нагруженном графе и ее решение на основе алгоритма Дейкстры

Поставим в соответствие каждому ребру u связного графа $G=(X,\ U)$ целое число $l(u)\geq 0$ и будем называть его весом (длиной, стоимостью) ребра u. В результате получим граф с нагруженными ребрами или просто нагруженный граф, который обозначается $G=(X,\ U,\ L)$, где $L=[l_{ij}]$ – матрица весов. Для любой цепи μ ее вес равен сумме весов составляющих ее ребер: $l(\mu)=\sum_{u\in\mu} l(u)$.

Решим следующую задачу: в связном графе $G=(X,\ U,\ l)$ найти кратчайшую цепь, связывающую две заданные вершины a и b.

Алгоритм решения этой задачи, как и предыдущий, состоит в вычислении по определенным правилам индексов вершин. Индекс вершины x обозначим λ_x. После окончания процесса вычисления индексов они должны удовлетворять *условиям оптимальности*, которые заключаются в следующем:

$1^\circ.\ \lambda_a=0$.

$2^\circ.\ \forall u=\{x,\ y\}:\ \lambda_y-\lambda_x\leq l(x,\ y)$.

$3^\circ.\ \forall y\exists x:\lambda_y-\lambda_x=l(x,\ y)$, где x – смежная с y вершина.

При выполнении этих условий длина кратчайшей цепи между вершинами a и b равна λ_b, а сама кратчайшая цепь проходит по таким вершинам

$$a=x_0\to x_1\to\ldots\to x_{i-1}\to x_i\to x_{i+1}\to\ldots\to x_{k-1}\to x_k,$$

для которых $\lambda_{x_i}-\lambda_{x_{i-1}}=l(x_{i-1},\ x_i)$, $i=1,\ \ldots,\ k$.

Для доказательства этого утверждения построим цепь, используя свойство 3°. Длина такой цепи равна

$$l(a,\ x_1)+l(x_1,\ x_2)+\ldots+l(x_{k-1},\ b)=\lambda_{x_1}-\lambda_a+\lambda_{x_2}-\lambda_{x_1}+\ldots+\lambda_b-\lambda_{x_{k-1}}=\lambda_b-\lambda_a=\lambda_b.$$

Покажем, что длина любой другой цепи между вершинами a и b не меньше, чем λ_b.

Пусть

$$a=y_0\to y_1\to\ldots\to y_{i-1}\to y_i\to y_{i+1}\to\ldots\to y_{s-1}\to y_s$$

произвольная из таких цепей. В силу свойства 2°:

$$\begin{cases} \lambda_{y_1} - \lambda_a \le l(a,\ y_1); \\ \lambda_{y_2} - \lambda_{y_1} \le l(y_1,\ y_2); \\ \quad\quad ... \\ \lambda_b - \lambda_{y_{s-1}} \le l(y_{s-1},\ b). \end{cases}$$

Сложив эти неравенства, получим

$$\lambda_b \le l(a,\ y_1) + l(y_1,\ y_2) + ... + l(y_{s-1},\ b)\,.$$

Рассмотрим далее алгоритм Дейкстры, который обеспечивает присвоение вершинам графа отметок, удовлетворяющих условиям 1°–3°.

Алгоритм состоит из двух этапов:

1) инициализация алгоритма, которая заключается в начальной расстановке отметок;

2) циклически повторяющаяся процедура исправления отметок.

На каждом шаге алгоритма все отметки делятся на *предварительные* и *окончательные*. Алгоритм обрабатывает только предварительные отметки и заканчивает работу, когда все отметки станут окончательными.

Рассмотрим алгоритм Э. Дейкстры, который позволяет определить расстояние от одной из вершин графа до всех остальных.

Каждой вершине $x \in X$ графа $G = (X,\ U,\ l)$ поставим в соответствие метку – минимальное известное расстояние от вершины x до начальной вершины a. Алгоритм работает пошагово – на каждом шаге он «посещает» одну вершину и пытается уменьшить метки. Работа алгоритма завершается, когда все вершены посещены.

Инициализация (начальная расстановка отметок). Полагаем

$$\lambda_a = 0,\ \lambda_x = \infty\,.$$

Символ ∞ отражает тот факт, что расстояния от a до других вершин пока неизвестны. Все отметки объявляем предварительными.

Шаг алгоритма (исправление отметок). Из еще не просмотренных вершин выбираем вершину x, имеющую минимальную метку λ_x. Для каждой вершины y, смежной с x и имеющей предварительную отметку λ_y, исправляем отметку λ_y по правилу

$$\lambda_y^{'} = \min\{\ \lambda_y,\ \lambda_x + l(x,\ y)\}\,.$$

Объявляем отметку вершины x окончательной и повторяем шаг алгоритма для следующей вершины. Число шагов равно числу вершин графа.

Для графа, изображенного на рис. 2, показаны окончательные отметки и кратчайшая цепь.

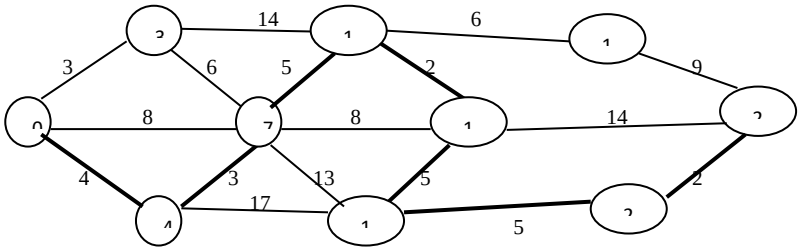


Рис. 2. Окончательные отметки и кратчайшая цепь в нагруженном графе

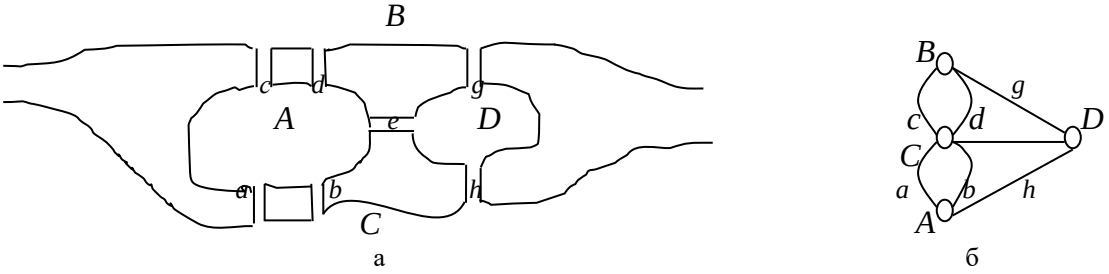
Лекция 10. ЦИКЛОМАТИКА ГРАФОВ

План лекции:
4. Эйлеровы циклы и цепи.
5. Цикломатическое число графа.
6. Понятие о гамильтоновых циклах.

4. Эйлеровы циклы и цепи

Началом математической теории графов послужила *задача о Кёнигсбергских мостах*, поставленная Леонардом Эйлером (1707 – 1783).

Кёнигсберг (теперь Калининград) расположен на обоих берегах реки Преголя и на двух островах этой реки. Берега реки и два острова соединены семью мостами, как показано на рис.1. Эйлер в 1736 г. поставил вопрос: можно ли, начав с некоторой точки, совершить прогулку и вернуться в исходную точку, пройдя по каждому мосту ровно один раз. Если каждый берег и острова считать вершинами графа, а каждый мост – ребром, то карту города можно представить в виде мультиграфа (рис. 1).



а б

Рис. 1. Карта города (а) и эквивалентный ей граф (б)

Эйлеровым циклом (*цепью*) в мультиграфе G называется цикл (цепь), содержащий все ребра мультиграфа по одному разу. Граф, имеющий эйлеров цикл, также будем называть э*йлеровым* . Граф, содержащий эйлерову цепь, называется *полуэйлеровым.* Эйлеров граф можно нарисовать на бумаге, не отрывая от нее карандаша. Замкнутые линии, которые можно получить, не отрывая карандаша от бумаги, проходя при этом каждый участок один раз, называются *уникурсальными*. С этими линиями связана задача минимизации холостого хода пера графопостроителя.

Эйлер установил, что задача кёнигсбергских мостов неразрешима, и этот результат ознаменовал возникновение теории графов.

Т е о р е м а . Мультиграф обладает эйлеровым циклом тогда и только тогда, когда он связный и все его локальные степени четны.

Д о к а з а т е л ь с т в о . Н е о б х о д и м о с т ь . Пусть мультиграф G обладает эйлеровым циклом. Тогда связность мультиграфа очевидна, так как в эйлеров цикл входят все ребра, а значит, и все вершины. Следовательно, любые две вершины соединены цепью. Далее, каждый раз, когда эйлеров цикл проходит через какую-то вершину, он должен войти в нее по одному ребру и выйти по другому, поэтому условие четности степеней вершин также необходимо.

Д о с т а т о ч н о с т ь докажем индукцией по числу ребер m мультиграфа. При $m = 2$ теорема справедлива. Пусть утверждение теоремы верно для всех мультиграфов с числом ребер, не превосходящем m. Для мультиграфа с числом ребер $m+1$ рассмотрим произвольный простой цикл. Хотя бы один такой цикл обязательно существует для графов с четными степенями вершин. Обозначим его μ'. Далее удалим из G все пройденные ребра. Получим мультиграф G', в котором все вершины по-прежнему имеют четные степени, но

он будет несвязным. Пусть $G_1', ..., G_p'$ – компоненты связности G'. Каждая из этих компонент представляет собой связный мультиграф с четными степенями и с числом ребер, меньшим, чем $m+1$, поэтому по предположению индукции она обладает эйлеровым циклом. Обозначим эйлеровы циклы компонент $\mu_1', ..., \mu_p'$. Присоединяя к циклу μ' эйлеровы циклы $\mu_1', ..., \mu_p'$, получаем эйлеров цикл мультиграфа G. ■

Следствия.

1°. Связный мультиграф, имеющий ровно две вершины с нечетной степенью, является полуэйлеровым графом.

Действительно, добавим к мультиграфу ребро, соединяющее вершины с нечетной степенью. В результате степени всех вершин станут четными. Построим в новом графе эйлеров цикл, а затем удалим добавленное ребро: цикл разорвется и станет цепью. Эта цепь будет начинаться и заканчиваться в вершинах нечетной степени.

2°. В общем случае число вершин мультиграфа, имеющих нечетную степень, всегда четно. Если мультиграф имеет $2 \cdot s$ таких вершин, то все его ребра можно включить в s цепей. Другими словами, мультиграф можно нарисовать, $s-1$ раз отрывая карандаш от бумаги.

Обоснование этого утверждения аналогично предыдущему.

Ребро графа $G = (X, U)$, через которое проходит хотя бы один цикл, называется *цикловым*. Ребро, которое не входит ни в один цикл, называется *перешейком* или *мостом*. Например, в графе, изображенном на рис. 2, ребра u_1 и u_2 – перешейки, а остальные ребра цикловые.

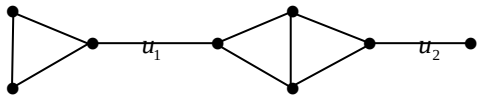


Рис. 2. Пример разбиения ребер графа на цикловые и перешейки

Справедливо следующее очевидное утверждение: при удалении из связного графа циклового ребра он остается связным; при удалении из связного графа перешейка граф распадается на две компоненты. Из этого утверждения следует, что при удалении из связного графа циклового ребра число связных компонент не изменяется. При удалении перешейка число связных компонент увеличивается на единицу.

5. Цикломатическое число графа

Рассмотрим (n, m)-граф G, имеющий p компонентов связности. Величина

$$\rho = n - p$$

называется *коцикломатическим числом* графа. Оно равно общему числу ребер в остовах каждой из p связных компонент графа G. *Цикломатическим числом* (*дефектом* или *первым числом Бетти*) называется величина

$$\lambda = m - \rho = m - n + p.$$

Теорема. $\forall G: \lambda \geq 0$.

Доказательство. Будем удалять из графа по одному ребру и следить за изменением величины λ. Параметры исходного графа обозначим m, n, p, а после удаления ребра – m', n', p'. В процессе удаления ребер возможны две ситуации:

1°. Удаляемое ребро цикловое. Тогда

$$m' = m-1,\ n' = n,\ p' = p;\ \lambda' = m' - n' + p' = \lambda - 1.$$

2°. Удаляемое ребро – перешеек. В этом случае

$$m' = m-1,\ n' = n,\ p' = p+1;\ \lambda' = m' - n' + p' = \lambda.$$

Итак, при удалении ребра величина λ либо не изменяется, либо уменьшается на единицу. После удаления всех ребер получим пустой граф, для которого $m_0 = 0$, $n_0 = n$, $p_0 = n$, то есть $\lambda_0 = 0$. Следовательно, в исходном графе $\lambda \geq 0$. ■

Из теоремы следует, что при $\lambda > 0$ в графе имеется, по крайней мере, один цикл.

6. Понятие о гамильтоновых циклах

Простой цикл в графе G, проходящий через каждую вершину графа один и только один раз, называется *гамильтоновым*.

Распространенная интерпретация задачи о гамильтоновых циклах состоит в следующем. Обед накрыт на круглом столе. Среди гостей некоторые являются друзьями. При каких условиях можно рассадить всех так, чтобы по обе стороны каждого из присутствующих сидели его друзья?

Подобная задача сформулирована и решена Киркманом. Одиннадцать министров ежедневно садятся за круглый стол. Как их рассаживать, если желательно, чтобы каждый из них имел на каждом заседании новых соседей? Сколько дней это может продолжаться?

Эта задача сводится к отысканию наибольшего числа гамильтоновых циклов без общих ребер в полном графе с одиннадцатью вершинами $a, b, c, d, e, f, g, h, i, j, k$. В данном случае существует только пять циклов без общих ребер:

$$a\ b\ c\ d\ e\ f\ g\ h\ i\ j\ k\ a,$$
$$a\ c\ e\ b\ g\ d\ i\ f\ k\ h\ j\ a,$$
$$a\ e\ g\ c\ i\ b\ k\ d\ j\ f\ h\ a,$$
$$a\ g\ i\ e\ k\ c\ j\ b\ h\ d\ f\ a,$$
$$a\ i\ k\ g\ j\ e\ h\ c\ f\ b\ d\ a.$$

Эти циклы получаются вращением линии, проведенной на рис. 3 в направлении стрелок. В более общем случае $2 \cdot n + 1$ вершин можно найти n гамильтоновых циклов без общих ребер.

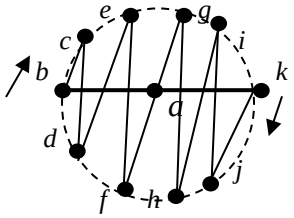


Рис. 3. Гамильтоновы циклы в задаче Киркмана

В применениях графов к играм вершины соответствуют различным позициям. Таким образом, существование гамильтонова цикла равносильно существованию циклической последовательности ходов, содержащей каждую позицию по одному разу. Примером является известная *задача о шахматном коне*: можно ли, начиная из произвольного поля на доске, ходить конем в такой последовательности, чтобы пройти через каждое из 64 полей и вернуться в исходное? Эта задача имеет несколько вариантов решений.

Гамильтоновой цепью в графе называется простая цепь, проходящая через все вершины по одному разу.

Если в графе не существует гамильтоновых циклов, то можно искать сумму непересекающихся простых циклов, проходящих через все вершины.

В орграфах можно искать орциклы, проходящие через каждую вершину по одному разу.

К гамильтоновым циклам относится так называемая *задача о коммивояжере*. Район, который должен посетить коммивояжер, содержит какое-то количество городов. Расстояния между ними известны, и нужно найти кратчайшую дорогу, проходящую через

все пункты и возвращающуюся в исходный. Эта задача имеет ряд приложений в исследовании операций, например в вопросах о наиболее эффективном использовании подвижного состава или оборудования.

В задаче о коммивояжере города можно представить как вершины графа G, в котором каждой паре вершин приписывается расстояние $\mu(a, b)$. Если вершины не инцидентны, то полагают $\mu(a, b) = \infty$. Тогда задача состоит в том, чтобы найти такой гамильтонов цикл P, для которого сумма

$$\mu(P) = \sum_{i=1}^{n} \mu(a_{i-1}, a_i)$$

минимальна. Так как обычно речь идет только о конечном числе вершин, задача может быть решена перебором. Однако, никакого эффективного алгоритма решения этой задачи до сих пор не известно. Имеются некоторые частные схемы для отдельных случаев. Например, частную задачу определения кратчайшей воздушной линии, соединяющей все столицы штатов в США, просчитали до конца Данциг, Фалкерсон и Джонсон.

В отличие от эйлеровых циклов критерий существования гамильтоновых циклов не известен. Более того, иногда даже для конкретных графов бывает очень трудно решить, можно ли найти такой цикл.

Лекция 11. ДЕРЕВЬЯ И ИХ СВОЙСТВА. ЗАДАЧА ПОИСКА КРАТЧАЙШЕГО ОСТОВА

План лекции:

7. Введение.
8. Определение дерева и его свойства.
9. Задача поиска кратчайшего остова.

1. Введение

Одним из наиболее важных понятий, которое часто используется в различных приложениях теории графов, является дерево. С помощью деревьев легко описывается структура самых различных объектов: организаций и учреждений, книг и документов, математических формул, химических соединений, компьютерных файловых систем, программ и многое другое.

Принято считать, что первым использовал понятие дерева Кирхгофф в 1847 г. при исследовании электрических цепей. Спустя десять лет химик Кэли ввел термин «дерево» при изучении структуры углеводородных соединений и получил первые важные результаты в этом разделе теории графов.

2. Определение дерева и его свойства

Граф без циклов называется *ациклическим* или *лесом*. Связный ациклический граф называется *деревом*. Если G – лес, то каждая его компонента является деревом. *Листом* называют вершину, степень которой равна 1, если она не рассматривается как корень. В качестве корня в неориентированном дереве можно принять любую вершину.

Существует несколько эквивалентных определений неориентированного дерева, каждое из которых отражает различные свойства последнего. Приведем некоторые из них.

Т е о р е м а . Следующие определения дерева эквивалентны:

1) дерево – это связный граф без циклов;
2) дерево – это связный граф, в котором каждое ребро является мостом;
3) дерево – это связный граф, цикломатическое число которого равно нулю;
4) дерево – это граф, в котором для любых двух вершин существует ровно одна соединяющая их цепь.

Эти утверждения выводятся одно из другого по схеме $1) \Rightarrow 2) \Rightarrow 3) \Rightarrow 4) \Rightarrow 5)$.

Из свойства 3) имеем: $\lambda = m - n + 1 = 0$ или $m = n - 1$, то есть в любом дереве число ребер на единицу меньше числа вершин.

Рассмотрим связный граф $G = (X,\ U)$ и будем из него удалять по одному цикловые ребра до получения ациклического подграфа. В результате получим *остовное* дерево $T = (X',\ U')$ графа G, для которого $X' = X$, $U' \subseteq U$.

Так как удаление цикловых ребер можно вести разными способами, то один и тот же граф в общем случае имеет несколько остовных деревьев. На рис. 1. представлен граф G и три его остовных дерева.

G T_1 T_2 T_3

Рис. 1. Граф G и его остовные деревья T_1, T_2 и T_3

Ребра графа G, не вошедшие в его остовное дерево T, называются *хордами* дерева T.

Л е м м а. В графе G для любого остовного дерева T и любой хорды h этого дерева существует единственный цикл, содержащий хорду h и не содержащий других хорд.

Д о к а з а т е л ь с т в о. Пусть $h=\{a, b\}$. В дереве T имеется единственная цепь, соединяющая вершины a и b. Присоединяя к этой цепи ребро h, получим требуемый цикл.

3. Задача поиска кратчайшего остова

Задача поиска кратчайшего остова состоит в следующем: для нагруженного графа G требуется построить остовное дерево T, сумма длин ребер которого минимальна.

Этой задаче можно дать такую интерпретацию: n пунктов на местности нужно связать сетью дорог, трубопроводов или линий телефонной связи. Для каждой пары пунктов i и j задана стоимость их соединения $l(i, j)$, которая представляет длину ребра $\{i, j\}$. Требуется построить связывающую сеть минимальной стоимости, которую называют *кратчайшей связывающей сетью*. Очевидно, что эта сеть будет остовным деревом графа G, при этом среди всех остовных деревьев она будет иметь минимальную сумму длин входящих в нее ребер.

Алгоритм построения кратчайшей связывающей сети состоит из $n-1$ шагов, на каждом из которых присоединяется одно ребро. Правило для выбора этого ребра следующее: среди еще не выбранных ребер берется самое короткое, не образующее цикла с уже выбранными ребрами.

Пример. Дана матрица расстояний $C=[c_{ij}]$, в которой элемент c_{ij} – вес ребра, который указывает в условных единицах затраты, необходимые для того, чтобы связать пункт i с пунктом j. Требуется с наименьшими затратами связать все пункты друг с другом.

$$C=\begin{pmatrix} 0&5&6&2&1&7&5&4&6\\ 5&0&7&7&5&1&4&6&5\\ 6&7&0&5&6&6&2&5&4\\ 2&7&5&0&1&7&6&4&5\\ 1&5&6&1&0&5&6&3&7\\ 7&1&6&7&5&0&4&5&5\\ 5&4&2&6&6&4&0&6&2\\ 4&6&5&4&3&5&6&0&7\\ 6&5&4&5&7&5&2&7&0 \end{pmatrix}.$$

Применение сформулированного выше алгоритма выглядит следующим образом. В матрице C отыскивается минимальный элемент, который вычеркивается, а соответствующее ему ребро заносится в сеть, если при этом не образуется цикл. Затем эти действия повторяются. Таким образом, на первых пяти шагах работы алгоритма будут выбраны ребра $\{1, 5\}$, $\{2, 6\}$, $\{4, 5\}$, $\{3, 7\}$, $\{7, 9\}$. Из оставшихся ребер минимальную длину имеет ребро $\{1, 4\}$, но в сеть оно не включается, так как образует цикл с уже выбранными ребрами. На следующих этапах алгоритма в сеть будут включены ребра $\{5, 8\}$, $\{2, 7\}$ и $\{1, 2\}$.

Для обоснования алгоритма предположим, что дерево T, которое он строит, состоит из ребер $u_1, \ldots, u_{n-1}$, для которых $l(u_1) \le \ldots \le u_{n-1}$.

Рассмотрим любое другое дерево T' и упорядочим его ребра по возрастанию длин. Пусть первые $k-1$ ребер дерева T' такие же, как в дереве T, а k-е ребро отличается от u_k $(1 \le k \le n-1)$. Присоединим к дереву T' ребро u_k. Тогда возникнет цикл, в который входит

ребро u_k и какие-то ребра из дерева T'. Среди этих ребер обязательно найдется ребро u, длина которого не меньше, чем длина ребра u_k: иначе ребро u_k образовало бы цикл с ребрами меньшей длины, что исключается правилом выбора очередного ребра в рассмотренном алгоритме. Удалим из дерева T' ребро u, заменив его ребром u_k. В результате получим дерево, длина которого не больше, чем длина дерева T'. Аналогичным путем вводим в дерево T' ребра $u_{k+1}, ..., u_{n-1}$, при этом всякий раз длина дерева не увеличится. Это означает, что дерево T действительно кратчайшее.

Лекция 12. ПРОСТРАНСТВО ЦИКЛОВ ГРАФА

План лекции:
1. Пространство остовных подграфов.
2. Квазициклы. Пространство циклов графа.
3. Размерность и базис пространства циклов.

7. Пространство остовных подграфов

Зафиксируем некоторое множество X и рассмотрим множество G_X всех графов с множеством вершин X. Буквой O будем обозначать пустой граф из этого множества: $O = G(X,\ 0)$.

Для графов $G_1 = (X,\ U_1)$ и $G_2 = (X,\ U_2)$ из G_X определим их *сумму по модулю 2* как граф $G_1 \oplus G_2 = (X,\ U_1 \oplus U_2)$, где $U_1 \oplus U_2$ обозначает симметрическую разность множеств U_1 и U_2. Иначе говоря, ребро принадлежит графу $G_1 \oplus G_2$ тогда и только тогда, когда оно принадлежит одному из графов G_1 и G_2. Пример показан на рис. 1.

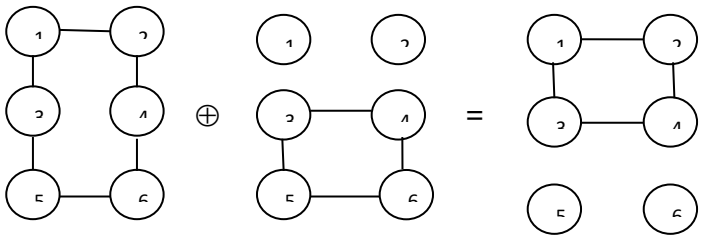


Рис. 1.

Введенная операция обладает следующими свойствами:

1°. Коммутативность: $\forall G_1, G_2 \Rightarrow G_1 \oplus G_2 = G_2 \oplus G_1$.

2°. Ассоциативность: $\forall G_1, G_2, G_3 \Rightarrow G_1 \oplus (G_2 \oplus G_3) = (G_1 \oplus G_2) \oplus G_3$.

3°. $\forall G \Rightarrow G \oplus O = G$.

4°. $\forall G \Rightarrow G \oplus G = O$.

Отсюда следует, что множество G_X относительно операции $\oplus$ образует *абелеву группу*. Нейтральным элементом («нулем») этой группы служит граф O, а противоположным к каждому графу является сам этот граф. Уравнение $G \oplus X = H$ с неизвестным X и заданными графами G и H имеет единственное решение $X = G \oplus H$. Благодаря свойству ассоциативности в выражении вида $G_1 \oplus G_2 \oplus ... \oplus G_k$ можно не использовать скобки для указания порядка действий. Очевидно, что ребро принадлежит графу $G_1 \oplus G_2 \oplus ... \oplus G_k$ тогда и только тогда, когда оно принадлежит нечетному количеству графов $G_1,\ G_2,\ ...,\ G_k$.

Рассмотрим множество из двух элементов $\{0,\ 1\}$. Оно является полем относительно операций умножения и сложения по модулю 2. Определим операцию умножения элементов этого поля на графы:

$$\forall G \Rightarrow 0 \cdot G = O,\ 1 \cdot G = G.$$

Множество G_X с введенными операциями сложения графов и умножения на элементы поля является линейным векторным пространством.

Зафиксируем некоторый граф $G \in G_X$ и рассмотрим множество всех его остовных подграфов, которое будем обозначать $S[G]$. Это множество состоит из 2^m элементов, среди них сам граф G и граф O. Оно замкнуто относительно сложения графов и умножения на

элементы поля, следовательно, является подпространством пространства G_X. Его называют *пространством остовных подграфов* графа G.

Остовному подграфу $G' = (X,\ U')$ можно поставить в соответствие двоичное слово $\tilde{\alpha} = \alpha_1 ... \alpha_m$, в котором нули указывают, какие ребра удалены, а единицы – какие оставлены:

$$G' \leftrightarrow \alpha(G') = \alpha_1\ ...\ \alpha_m,\ \alpha_i = \begin{cases} 1, \text{ если } u_i \in U', \\ 0, \text{ если } u_i \notin U', \end{cases} i = 1,...,\ m.$$

8. Квазициклы. Пространство циклов графа

Циклом будем называть граф, у которого одна компонента связности является простым циклом, а остальные – изолированными вершинами.

В пространстве остовных подграфов графа G выделим подпространство, содержащее все циклы графа G.

Определение. Остовный подграф, у которого степени всех вершин четны, называется *квазициклом*.

Отметим, что всякий цикл является квазициклом, в том числе и пустой граф.

Покажем, что множество квазициклов замкнуто относительно операции сложения.

Лемма. Сумма двух квазициклов есть квазицикл.

Доказательство. Пусть C_1 и C_2 – квазициклы. Рассмотрим произвольную вершину $x \in X$, и пусть ее степени в C_1 и C_2 равны соответственно s_1 и s_2. Тогда степень вершины x в графе $C_1 \oplus C_2$ будет равна $s = s_1 + s_2 - 2 \cdot s_{1,2}$, где $s_{1,2}$ – число вершин, с которыми x смежна в обоих графах C_1 и C_2. Отсюда видно, что s четно, если четны s_1 и s_2. ■

Из леммы следует, что множество квазициклов является линейным векторным пространством над полем {0, 1}, которое называется *пространством циклов* графа G. Обозначим это пространство через $C[G]$. Очевидно, что $C[G]$ является подпространством векторного пространства остовных подграфов.

9. Размерность и базис пространства циклов

Компактное представление пространства дает его базис. Если выписать все простые циклы графа G, то в большинстве случаев они не образуют базис, так как некоторые из этих циклов могут быть суммами других. Построить базис пространства $C[G]$, состоящий из простых циклов, можно следующим образом. Выберем в графе G какой-нибудь остов (каркас) T. Пусть $h_1,\ ...,\ h_s$ – все хорды графа G. Если добавить к T хорду h_i, то в графе образуется единственный цикл C_i. Таким образом, получим семейство из s циклов, которые называются *фундаментальными (базисными*) циклами относительно остова T.

Теорема. *Множество всех фундаментальных циклов относительно любого остова T графа G образует базис пространства циклов этого графа.*

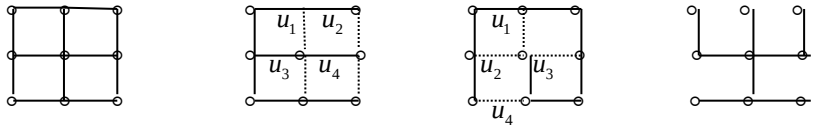
Доказательство.

Зафиксируем некоторый остов T и рассмотрим фундаментальные циклы $C_1,\ ...,\ C_s$ относительно этого остова. В каждом из этих циклов имеется хорда h_i, принадлежащая циклу C_i и не принадлежащая никакому из остальных. Поэтому при сложении этого цикла с другими фундаментальными циклами эта хорда будет присутствовать в суммарном графе. Следовательно, сумма различных фундаментальных циклов никогда не будет пустым графом, то есть фундаментальные циклы линейно независимы.

Покажем теперь, что любой квазицикл C графа G является суммой фундаментальных циклов. Пусть h_{i_1}, ..., h_{i_k} – все ребра C, не принадлежащие T (хорды графа C). Рассмотрим квазицикл $C' = C \oplus C_{i_1} \oplus ... \oplus C_{i_k}$. Каждое из ребер h_{i_j} ($j = 1, ..., k$) входит ровно в два слагаемых этой суммы – в C и в C_{i_j}. Следовательно, при сложении эти ребра уничтожатся. Все остальные ребра, присутствующих в графах-слагаемых, принадлежат T. Значит, C' – подграф графа T. Поскольку в T нет циклов, то $C' = O$. Отсюда $C = C_{i_1} \oplus ... \oplus C_{i_k}$.

Из этой теоремы следует, что размерность пространства циклов графа равна числу ребер, не входящих в его остов, то есть числу хорд. Так как остов содержит $n - p$ ребер, то эта размерность равна $m - n + p$. Таким образом, *размерность пространства циклов графа равна его цикломатическому числу.*

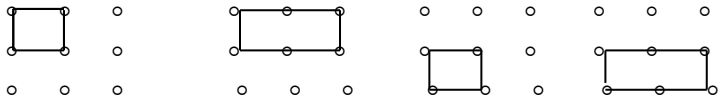
Пример. Построим систему базисных циклов для графа, представленного на рис. 2.



G T_1 T_2 T_3

Рис. 2. Граф G и его остовные деревья T_1, T_2 и T_3

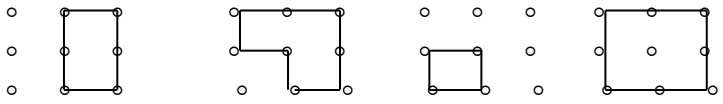
Выделим остовное дерево T_1. Присоединяя к дереву по очереди хорды u_1, u_2, u_3 и u_4, получим 4 базисных цикла (рис. 3).



C_1 C_2 C_3 C_4

Рис. 3. Базисные циклы графа G

Для дерева T_2 получим другую систему базисных циклов (рис. 4).



$\tilde{C}_1$ $\tilde{C}_2$ $\tilde{C}_3$ $\tilde{C}_4$

Циклы одной из систем можно выразить как линейные комбинации циклов из другой системы. В данном случае имеем:

$$\begin{cases} \tilde{C}_1 = C_1 \oplus C_2 \oplus C_3 \oplus C_4, \\ \tilde{C}_2 = \quad C_2 \oplus C_3 \oplus C_4, \\ \tilde{C}_3 = \quad\quad C_3 \oplus C_4, \\ \tilde{C}_4 = \quad C_2 \quad\quad \oplus C_4. \end{cases}$$

Обратное преобразование имеет вид:

$$\begin{cases} C_1 = \tilde{C}_1 \oplus \tilde{C}_2, \\ C_2 = \quad \tilde{C}_2 \oplus \tilde{C}_3, \\ C_3 = \quad \tilde{C}_2 \quad\quad \oplus \tilde{C}_4, \\ C_4 = \quad \tilde{C}_2 \oplus \tilde{C}_3 \oplus \tilde{C}_4. \end{cases}$$

Лекция 13. ПОТОКИ В СЕТЯХ

План лекции:
1. Задача о максимальном потоке.
2. Определение сети, потока и разреза.
3. Теорема Форда-Фалкерсона.

10. Задача о максимальном потоке

Одной из наиболее интересных и важных задач теории графов является задача определения максимального потока, протекающего от некоторой вершины s графа (источника) к некоторой конечной вершине t (стоку). При этом каждой дуге $u_{ij} = (x_i, x_j)$ графа G приписана некоторая пропускная способность $q_{ij}(u_{ij})$, и эта пропускная способность определяет наибольшее значение потока, который может протекать по данной дуге. Эта задача и ее варианты могут возникать во многих практических приложениях, например при определении максимальной интенсивности транспортного потока между двумя пунктами на карте дорог, представленной графом. В этом примере решение задачи о максимальном потоке укажет также ту часть сети дорог, которая «насыщена» и образует «узкое место» в отношении потока между двумя указанными концевыми пунктами.

Метод решения *задачи о максимальном потоке* (от s к t) предложен Фордом и Фалкерсоном, и их «техника пометок» составляет основу других алгоритмов решения многочисленных задач, являющимися простыми обобщениями или расширениями указанной задачи. Рассмотрим возможные варианты задачи о максимальном потоке.

Задача нахождения допустимого потока минимальной стоимости. Допустим, что каждой дуге графа приписана не только пропускная способность q_{ij}, дающая верхнюю границу потока через дугу $u_{ij} = (x_i, x_j)$, но также пропускная способность r_{ij}, дающая нижнюю границу потока через эту дугу. В общем случае может существовать много потоков, удовлетворяющим требованиям о максимальной и минимальной пропускных способностях дуг. Если в дополнение к пропускным способностям заданы также стоимости единицы потока, протекающего по дуге, то возникает задача нахождения допустимого потока минимальной стоимости.

Задача о многопродуктовом потоке. Эта задача возникает, если в сети имеется несколько источников и стоков, между которыми протекают потоки различных продуктов. В этой задаче пропускная способность $q_{ij}(u_{ij})$ является ограничением для суммы всех потоков всех видов продукции через эту дугу.

Задача о потоках с выигрышами. Во всех рассмотренных выше случаях неявно допускалось, что поток на входе дуги такой же, как и на выходе. Если рассмотреть граф, в котором выходной поток дуги равен ее входному потоку, умноженному на некоторое неотрицательное число, то задачу о максимальном потоке называют задачей о потоках с выигрышами. В такой задаче потоки могут «порождаться» и «поглощаться» самим графом, так что поток, входящий в s, и поток, покидающий t, могут изменяться совершенно независимо.

11. Определение сети, потока и разреза

Граф, некоторые вершины которого выделены, называется *сетью*. Выделенные вершины называются *полюсами* сети. Например, дерево с корнем можно рассматривать как однополюсную сеть.

Вершины, отличные от полюсов, называются *внутренними* вершинами сети. Ребро, инцидентное хотя бы одному полюсу, называется *полюсным* ребром. Остальные ребра называются *внутренними*.

(k, l)-полюсником называется сеть с $(k+l)$ полюсами, разбитыми на два класса: k входных и l выходных полюсов. (1,1)-полюсник называется также *двухполюсной сетью.*

Далее будут рассматриваться только двухполюсные сети, которые будем называть просто сетями. Будем называть также цепью (без указания концов) элементарную цепь между полюсами сети. В противном случае будем указывать концы цепи и называть ее *цепочкой*.

Транспортной называется двухполюсная сеть, в которой каждой дуге u приписано целое неотрицательное число $c(u)$, называемое пропускной способностью дуги.

Можно дать различные интерпретации транспортной сети. Пусть, например, в полюсе s имеется неограниченный запас некоторого продукта и нужно организовать доставку этого продукта в полюс t по сети путей сообщения с некоторыми промежуточными вершинами. В этом случае пропускная способность дуги – количество груза, которое можно перевезти в единицу времени по данной дуге. Тогда возникает задача: организовать перевозки по сети таким образом, чтобы, не превышая пропускных способностей дуг, перевозить из s в t максимальное количество груза в единицу времени.

Для каждой вершины x сети S обозначим через U_x^+ множество всех дуг, входящих в x, а через U_x^- – выходящих из x. Для источника s и стока t имеем $U_s^+ = U_t^- = \varnothing$.

Потоком в сети S называется целочисленная функция $\varphi(u)$, определенная на дугах сети и удовлетворяющая условиям:

1) $0 \le \varphi(u) \le c(u),\ (u \in U)$; (1)

2) $\sum_{u \in U_x^+} \varphi(u) - \sum_{u \in U_x^-} \varphi(u) = 0,\ (x \in X,\ x \ne s,\ x \ne t)$. (2)

Второе условие называют *уравнением сохранения*. Оно представляет собой для каждой внутренней вершины сети закон Кирхгофа, согласно которому сумма значений потока по ребрам, входящим в вершину, равна сумме значений потока по ребрам, исходящим из вершины.

На рис. 1 приведен пример сети $S = (X, U, c(u), \varphi(u))$, в которой каждой дуге u_i приписана двойка $(c(u_i), \varphi(u_i))$.

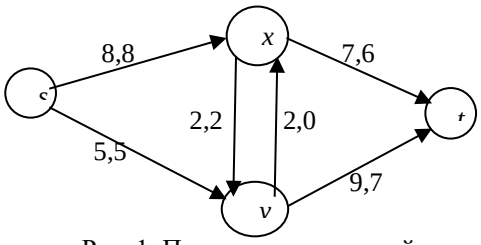


Рис. 1. Пример транспортной сети

Если сложить уравнения сохранения для всех вершин, то останутся только члены, соответствующие дугам U_s^- и U_t^+:

$$\sum_{u \in U_s^-} \varphi(u) - \sum_{u \in U_t^+} \varphi(u) = 0.$$

Таким образом, для любого потока величина груза, выходящего из источника s равна величине груза, прибывающего в сток t. Эту величину обозначают Ф и называют *величиной потока*:

$$\Phi = \sum_{u \in U_s^-} \varphi(u) = \sum_{u \in U_t^+} \varphi(u).$$

Поток в сети, имеющий наибольшую величину, называется *максимальным.*

Основная задача состоит в нахождении максимального потока для данной транспортной сети. Для ее решения используют специальные подмножества дуг сети, называемые *разрезами* или *сечениями*.

Разрезом сети называется множество ребер, при удалении которого сеть становится несвязной, причем полюсы попадают в разные компоненты связности. Очевидно, что любая цепь проходит через одно ребро разреза.

Пусть $A \subseteq X$ – некоторое подмножество вершин сети, для которого полюс $s \in A$, а другой полюс $t \notin A$. Обозначим $\overline{A} = X \setminus A$ – дополнение множества A до множества X. Тогда $s \notin \overline{A}$, а $t \in \overline{A}$. Множество дуг сети, имеющих начало в A, а конец в $\overline{A}$, называется *разрезом, порождаемым множеством вершин* A, и обозначается $(A,\ \overline{A})$. Например, для сети на рис. 1 выберем $A = \{s,\ x\}$. Тогда $\overline{A} = \{y,\ t\}$. Имеем разрез

$$(A,\ \overline{A}) = \{(x,\ t), (s,\ y), (x,\ y)\}.$$

Ребро разреза называется *прямым*, если оно ориентировано слева направо, и *обратным* – в противном случае. Сумма пропускных способностей всех прямых дуг разреза называется *пропускной способностью разреза* и обозначается $c(A,\ \overline{A})$. Разрезы, которые обладают наименьшей возможной пропускной способностью, называются *минимальными.*

12. Теорема Форда-Фалкерсона

В 1955 г. Форд и Фалкерсон доказали следующую теорему о максимальном потоке и минимальном разрезе.

Т е о р е м а. Максимальная величина потока $\Phi_{\max}$ в сети равна пропускной способности $c_{\min}$ любого минимального разреза.

Д о к а з а т е л ь с т в о. Докажем сначала, что величина любого потока не превосходит пропускной способности любого разреза: $\Phi \le c(A,\ \overline{A})$. Просуммируем уравнения сохранения (2) по всем вершинам $x \in A$. Получим

$$\sum_{u \in A} \varepsilon_u \cdot \varphi(u) = 0, \qquad (3)$$

где значения коэффициентов $\varepsilon_u \in \{-1,\ 0,\ 1\}$ зависят от расположения концов дуги $u = (x,\ y)$ относительно множеств A и $\overline{A}$. На рис. 2 показаны шесть возможных вариантов такого расположения.

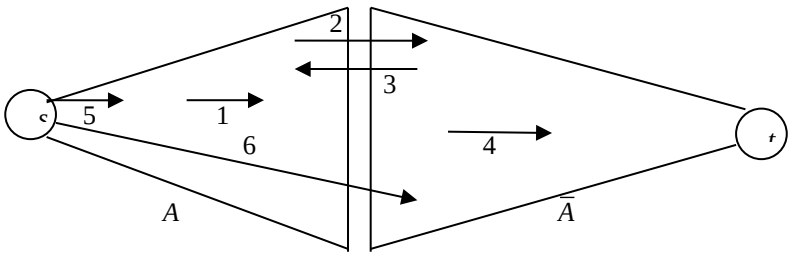


Рис. 2. Возможные расположения концов произвольной дуги $u = (x,\ y)$

1°. $x \in A,\ x \ne s,\ y \in A$. В этом случае $\varphi(u)$ в ходит в складываемые уравнения дважды: со знаком ”+” для вершины y и со знаком ”–” для вершины x. Следовательно, $\varepsilon_u = 0$.

2°. $x \in A,\ x \ne s,\ y \in \overline{A}$. В этом случае $\varphi(u)$ в ходит только в уравнение сохранения для вершины x, поэтому $\varepsilon_u = -1$.

3°. $x \in \overline{A},\ y \in A$. В этом случае $\varphi(u)$ в ходит только в уравнение сохранения для вершины y, поэтому $\varepsilon_u = 1$.

4°. $x \in \overline{A},\ y \in \overline{A}$. В этом случае $\varepsilon_u = 0$, т. к. $\varphi(u)$ нет в складываемых уравнениях.

5°. $x = s,\ y \in A$. Результат аналогичен п. 3°.

6°. $x = s,\ y \in \overline{A}$. Результат аналогичен п. 4°.

Для дуг шестого типа в сумму (3) добавим и вычтем $\varphi(u)$. Тогда $\varepsilon_u = 1$ для дуг, выходящих из источника ($u \in U_s^-$), и дуг, идущих против разреза $(u \in (\overline{A},\ A))$; $\varepsilon_u = -1$ для дуг разреза $(u \in (A,\ \overline{A}))$; $\varepsilon_u = 0$ для остальных дуг.

Перепишем уравнение (3) в виде

$$\sum_{u \in U_s^-} \varphi(u) - \sum_{u \in (A,\ \overline{A})} \varphi(u) + \sum_{u \in (\overline{A},\ A)} \varphi(u) = 0,$$

откуда для любого потока и любого разреза

$$\Phi = \sum_{u \in U_s^-} \varphi(u) = \sum_{u \in (A,\ \overline{A})} \varphi(u) - \sum_{u \in (\overline{A},\ A)} \varphi(u) \le \sum_{u \in (A,\ \overline{A})} \varphi(u) \le \sum_{u \in (A,\ \overline{A})} c(u) = c(A,\ \overline{A}). \qquad (4)$$

Отсюда следует, что максимальное значение потока в сети равно минимальному значению пропускных способностей разрезов этой сети:

$$\max_{\varphi} \Phi = \min_{A} c(A,\ \overline{A}). \ \blacksquare$$

Лекция 14. АЛГОРИТМ ПОСТРОЕНИЯ МАКСИМАЛЬНОГО ПОТОКА В ТРАНСПОРТНОЙ СЕТИ

План лекции:
1. Определение прибавляющей цепи.
2. Алгоритм построения максимального потока в транспортной сети.

1. Определение прибавляющей цепи

Из теоремы Форда-Фолкерсона следует, что максимальный поток в сети не превосходит минимальной пропускной способности разреза, то есть

$$\max_{\varphi} \Phi \le \min_{A} c(A, \bar{A}). \qquad (1)$$

Анализ неравенства (1) показывает, что величина Ф потока φ совпадает с пропускной способностью разреза $(A, \bar{A})$ тогда и только тогда, когда выполняется условие:

$$\varphi(u)= \begin{cases} c(u), \ \forall u \in (A, \bar{A}), \\ 0, \ \forall u \notin (A, \bar{A}). \end{cases} \qquad (2)$$

Алгоритм, который приводится ниже, направлен на построение такого разреза, для которого выполняется условие (2). Если удается построить такой разрез, то задача решена, если нет, то производится увеличение потока с помощью прибавляющих цепей.

Дадим определение прибавляющей цепи. Рассмотрим в сети цепь из источника в сток, то есть последовательность вершин

$$s = x_0, \ x_1, \ ..., x_i, \ x_{i+1,} \ ..., x_{k-1}, \ x_k = t$$

такую, что между вершинами x_i и x_{i+1} есть дуга ($i = 0, 1, ..., k-1$), которой может оказаться *прямой* $(x_i, \ x_{i+1})$ или *обратной* $(x_{i+1}, \ x_i)$.

Пусть в сети задан поток φ. Цепь из s в t называется *прибавляющей*, если для каждой ее прямой дуги выполняется строгое неравенство $\varphi(u) < c(u)$, а для каждой обратной дуги – строгое неравенство $\varphi(u) > 0$.

Предположим, что для потока φ удалось найти прибавляющую цепь. Тогда увеличивая φ на максимально возможное число n единиц на прямых дугах (с обеспечением условия $\varphi(u) \le c(u)$) и уменьшая на столько же единиц на обратных дугах (с обеспечением условия $\varphi(u) \ge 0$), получим новый поток, величина которого на n единиц больше, чем величина φ.

Пример 1. Пусть прибавляющая цепь имеет вид (рис. 1):

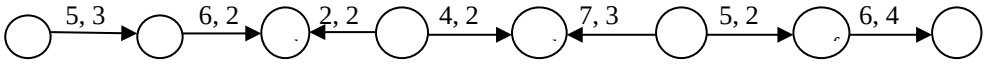


Рис.1.

С помощью это цепи можно перейти к новому потоку (рис. 2.):

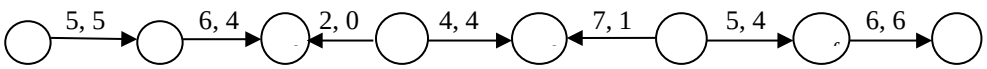


Рис. 1.

2. Алгоритм построения максимального потока в транспортной сети

Алгоритм состоит в последовательном просмотре вершин сети и присвоении им отметок. На каждом шаге алгоритма любая вершина находится в одном из трех состояний: а) не помечена; б) помечена, но не просмотрена; в) помечена и просмотрена.

0-й шаг. Зададим какой-нибудь, например, нулевой поток по сети.

1-й шаг. Помети источник s любой отметкой, например, звездочкой *. После этого вершина s помечена, но не просмотрена. Остальные вершины не помечены.

2-й шаг. Берем очередную помеченную, но не просмотренную вершину x. Просматриваем все дуги, инцидентные этой вершине. Если вторая вершины дуги не помечена, то помечаем ее отметкой "x" в следующих двух случаях:

а) дуга выходит из вершины x и поток по ней строго меньше пропускной способности;

б) дуга входит в вершину x и поток по ней строго больше нуля.

После завершения этого шага вершина x объявляется помеченной и просмотренной, а вершины, получившие при просмотре отметку "x", объявляются помеченными, но не просмотренными.

Шаг 2 циклически повторяется до тех пор, пока не произойдет одно из двух событий, рассматриваемых далее на 3-ем и 4-ом шагах.

3-й шаг. Сток t получил отметку, например, "y". Переходим из t в вершину y, по отметке вершины y отыскиваем следующую вершину и т. д. до тех пор, пока не дойдем до вершины s. В результате получаем прибавляющую цепь, с помощью которой увеличиваем текущий поток. Далее стираем отметки всех вершин и повторяем выполнение алгоритма с 1-го шага.

4-й шаг. Процесс расстановки отметок закончился тем, что все помеченные вершины просмотрены, но сток t при этом не помечен. Пусть A – множество помеченных вершин. Так как $t \notin A$, а $s \in A$, то можно определить разрез $(A, \overline{A})$. Для $\forall u \in (A, \overline{A})$, то есть дуги, идущей из помеченной вершины в непомеченную, $\varphi(u)=c(u)$, иначе другой конец этой дуги был бы помечен. По той же причине для $\forall u \notin (A, \overline{A})$ $\varphi(u)=0$. Следовательно, для построенного потока и разреза $(A, \overline{A})$, образованного помеченными вершинами, выполняются условия (1). В таком случае имеем максимальный поток.

Пример 2. Пусть задана транспортная сеть и поток по ней (рис. 2).

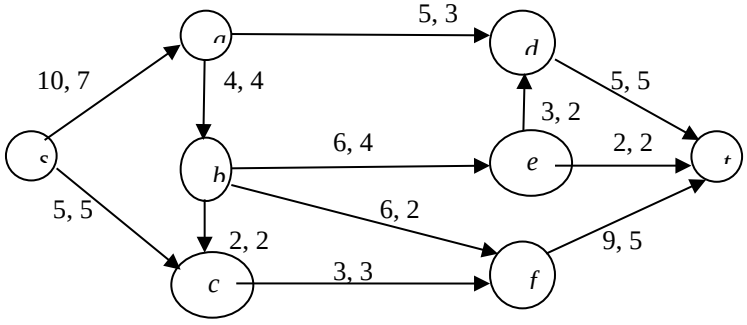


Рис. 2.

Процесс расстановки отметок показан в таблице 1.

Таблица 1

Номер шага	Отметки вершин							
	s	a	b	c	d	e	f	t
1	*							
2	*	"s"						
3	*	"s"			"a"			
4	*	"s"			"a"	"d"		
5	*	"s"	"e"		"a"	"d"		
6	*	"s"	"e"		"a"	"d"	"b"	
7	*	"s"	"e"		"a"	"d"	"b"	"f"

Поскольку сток получил отметку, то строим прибавляющую цепь (рис. 3).

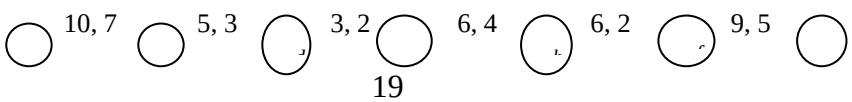


Рис. 3.

Увеличиваем поток на две единицы на прямых дугах этой цепи и уменьшаем на две единицы на обратных. В результате получаем поток, изображенный на рис. 4.

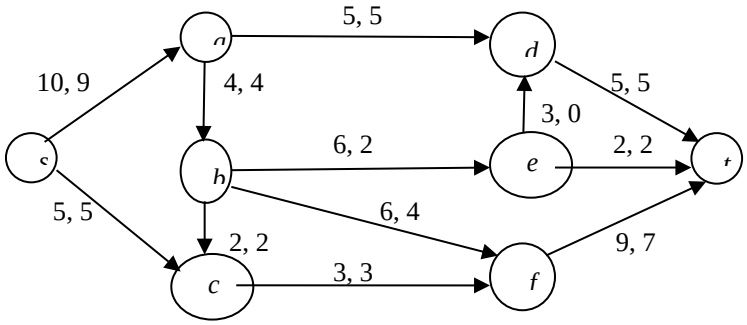


Рис. 4.

В процессе расстановки отметок для нового потока удается пометить только вершины s и a. Тогда $A=\{s,\ a\}$ – множество помеченных вершин, которое порождает минимальный разрез $(A,\ \bar{A})=\{(s,c),\ (a,b),\ (a,d)\}$. Его пропускная способность

$$c(A,\ \bar{A})=c(s,c)+c(a,b)+c(a,d)=5+4+5=14$$

совпадает с величиной потока

$$\Phi=\sum_{u\in U_s^-}\varphi(u)=c(s,\ a)+c(s,\ c)=9+5=14$$

или

$$\Phi=\sum_{u\in U_t^+}\varphi(u)=c(d,\ t)+c(e,\ t)+c(f,\ t)=5+2+7=14.$$

Лекция 16. СЕТЕВЫЕ ГРАФИКИ

План лекции:
1. Определение сетевого графика.
2. Алгоритм отыскания критического пути.
3. Определение резервов времени и коэффициентов напряженности работ.
4. Построение сетевого графика по заданной упорядоченности работ.

3. Определение сетевого графика

Примерно с 60-х годов получила широкое распространение система сетевого планирования (СПУ), основным элементом которой является сетевой график. *Сетевой график* представляет собой изображение хода проекта при помощи ациклической сети. Под сетью понимается ориентированный граф, в котором выделены две вершины: одна из них называется *началом* и не имеет входящих дуг, другая – *концом*, она не имеет выходящих дуг. Последовательность различных дуг, в которой начало каждой дуги совпадает с концом предыдущей, называется *путем*. Замкнутый путь называется *контуром* или ориентированным циклом. Если в сети нет ориентированных циклов, она называется *ациклической.* Дуги сети изображают отдельные работы, а вершины – события, состоящие в завершении одной или нескольких работ. Каждой дуге в сетевом графике приписано целое неотрицательное число – продолжительность соответствующей работы.

Обозначим $t_{i,j}$ – продолжительность работы $(i,\ j)$. Рассмотрим некоторый путь на сетевом графике: i_1 - i_2 - i_3 - ... - i_{k-1} - i_k. Длиной пути назовем сумму продолжительностей входящих в него работ: $t_{i_1,i_2} + t_{i_2,i_3} + ... + t_{i_{k-1},i_k}$.

Рассмотрим все пути из начальной вершины в конечную. Путь наибольшей длины называют *критическим*. Длина критического пути $l_{кр}$ – основная характеристика сетевого графика, смысл которой состоит в том, что если каждая работа $(i,\ j)$ будет начинаться в тот момент, когда произойдет событие i (раньше она начинаться не может), и выполняться точно за время $t_{i,j}$, то вся совокупность работ будет выполнена за время, равное длине критического пути.

4. Алгоритм отыскания критического пути

Опишем алгоритм для нахождения критического пути в сетевом графике. В процессе работы алгоритма для каждой вершины i рассчитывается величина $t_p(i)$ – максимальная длина пути из начала в вершину i.

1°. ***Правильная нумерация сети.*** Нумерация вершин сети называется *правильной*, если номер начала любой дуги сети меньше, чем номер ее конца. Правильная нумерация ациклической сети всегда возможна и производится следующим образом. Нумеруем начальную вершину сети нулем и удаляем ее из сети вместе со всеми выходящими из нее дугами. В получающейся сети непременно образуются вершины, не имеющие входящих дуг (это следует из ацикличности), которые назовем вершинами первого ранга и занумеруем их числами 1, 2, … Далее удалим все вершины первого ранга и выходящие из них дуги. Появившиеся вершины без входящих дуг назовем вершинами второго ранга и дадим им очередные номера. Этот процесс продолжается до тех пор, пока не будут занумерованы все вершины сети.

2°. ***Расстановка отметок.*** Пусть сеть правильно занумерована. Для каждой вершины i вычисляем отметку $t_p(i)$ по следующим правилам:

– полагаем $t_p(0) = 0$;

– просматриваем вершины в порядке их номеров и для j-ой вершины вычисляем $t_p(j)$ по формуле

$$t_p(j) = \max_i [t_p(i) + t_{ij}], \quad (1)$$

где максимум берется по всем вершинам i, имеющим дугу (i, j), направленную в вершину j.

По окончании процесса вычисления отметок величина $l_{кр}$ находится как отметка концевой вершины.

3°. ***Построение критического пути***. Начиная с вершины-конца, последовательно находим дуги (i, j), для которых $t_p(j) - t_p(i) = t_{ij}$. Эти дуги и образуют критический путь.

5. Определение резервов времени и коэффициентов напряженности работ

Параметр $t_p(i)$, вычисляемый при построении критического пути, называется «ранний срок свершения события i». Для каждого события можно рассчитать также поздний срок его свершения $t_n(i)$. Смысл этого параметра состоит в следующем: если событие i «запоздает, однако произойдет не позднее $t_n(i)$, то длина критического пути (время выполения всего проекта) не изменится.

Метод расчета $t_n(i)$ аналогичен методу расчета $t_p(i)$, если его «вывернуть наизнанку». Пусть для правильной сети концевая вершина получила номер N. Тогда для каждой вершины i вычисляем отметку $t_n(i)$ по следующим правилам:

– полагаем $t_n(N) = l_{кр}$;

– просматриваем вершины в обратном порядке их номеров и для i-ой вершины вычисляем $t_n(i)$ по формуле

$$t_n(i) = \min_j [t_n(j) - t_{ij}], \quad (2)$$

где минимум берется по всем вершинам j, в которые ведут дуги (i, j) из вершины i.

Резервом времени работы (i, j) называется величина

$$P(i, j) = t_n(j) - t_p(i) - t_{ij}. \quad (3)$$

Задержка в выполнении работы на величину, не превышающую $P(i, j)$, не изменяет длину критического пути, то есть срока выполнения проекта.

Резервы времени позволяют разбить всю совокупность работ на более важные и менее важные, что используется для управления выполнением проекта. Работы, лежащие на критическом пути, имеют резервы времени, равные нулю. Задержка в выполнении такой работы на время Δt ровно на такую же величину изменяет время выполнения всего проекта. Поэтому возможны варианты форсирования этих работ за счет работ, имеющих большой резерв времени.

При практическом применении сетевых графиков более удобными по сравнению с резервами времени являются *коэффициенты напряженности работ*, определяемые как отношение длины максимального пути из начала в конец, проходящего через данную работу, к длине критического пути:

$$k_н(i, j) = \frac{t_p(i) + t_{ij} + l_{кр} - t_n(j)}{l_{кр}} = 1 - \frac{P(i, j)}{l_{кр}}. \quad (4)$$

Из формулы (4) следует, что $0 < k_н(i, j) \le 1$, причем для работ с нулевым резервом (лежащих на критическом пути) $k_н(i, j) = 1$, а для работ с большим резервом времени –

$k_н(i,\ j) \approx 0$. На основании этого всю совокупность работ делят на зоны: *критическую* с $k_н(i,\ j) \approx 0,9 \div 1$, *подкритическую* с $k_н(i,\ j) \approx 0,6 \div 0,9$, и *резервную*.

6. Построение сетевого графика по заданной упорядоченности работ

Будем считать, что задан перечень работ, в совокупности составляющих некоторый проект:

$$U = \{1,\ 2,...,\ n\}.$$

(Работы пронумерованы и в дальнейшем ссылаемся на каждую работу по ее порядковому номеру).

Предположим далее, что для каждой работы j ($1 \le j \le n$) указаны ее непосредственные предшественники, то есть множество работ, выполнение которых является необходимым условием для начала работы j.

Пример. $U = \{1,\ 2,\ 3,\ 4,\ 5,\ 6,\ 7\}$, то есть весь комплекс работ, составляющих проект, разбит на 7 работ, соотношения между ними отображены в таблице 1.

Таблица 1

Работы	1	2	3	4	5	6	7
Предшественники	–	–	2	1,3	1,3	2	5,6

Требуется построить сетевой график, соответствующий заданной упорядоченности работ.

Процедура построения сетевого графика распадается на несколько этапов.

I. ***Транзитивное замыкание отношения предшествования***. Если работа i предшествует работе j, а работа j предшествует работе k, то, очевидно, работа i должна завершиться до начала работы k. Множество работ называется замкнутым (по транзитивности), если вместе с каждой работой оно содержит и всех ее предшественников.

Наименьшее замкнутое множество, содержащее множество работ M, называется *транзитивным замыканием M*.

На этапе I для каждого множества предшественников работы j строится его транзитивное замыкание. Оно будет состоять из всех работ i, для которых можно найти цепочку работ $j_1, j_2,\ldots j_p$, такую, что i предшествует j_1, j_1 предшествует $j_{2,\ \ldots}\ j_p$ предшествует j. Будем называть эти множества полными предшественниками работы j.

Продолжение примера. Полные предшественники для 7 работ рассматриваемого выше примера приведены в таблице 2.

Таблица 2

Работы	1	2	3	4	5	6	7
Полные предшественники	– P_1	– P_1	2 P_2	1,2,3 P_3	1,2,3 P_3	2 P_2	1,2,3,5,6 P_4

Среди построенных множеств встречаются одинаковые. Найдем все различные множества полных предшественников и обозначим их

$$P_1,\ P_2,\ ...,\ P_s. \qquad (5)$$

Продолжение примера. В рассматриваемом примере полагаем

$$P_1 = \varnothing,\ P_2 = \{2\},\ P_3 = \{1,\ 2,\ 3\},\ P_4 = \{1,\ 2,\ 3,\ 5,\ 6\}.$$

Построением множеств (1) заканчивается этап I. Каждому из множеств P_i в сетевом графике, который будет построен, соответствует вершина. Она так же будет обозначаться P_i.

II. ***Построение конституент для множеств полных предшественников***. Пусть $U = \{1,\ 2,\ldots,\ n\}$ – совокупность всех работ. Разбиение множества *U* на непересекающиеся подмножества

$$U = C_1 + C_2 + \ldots + C_t \qquad (6)$$

будем называть *разбиением на конституенты,* если каждое из множеств (5) полных предшественников можно представить в виде объединения некоторых конституент.

Продолжение примера. Пусть в рассматриваемом примере

$$C_1 = \{2\},\ C_2 = \{1,3\},\ C_3 = \{5,\ 6\},\ C_4 = \{4,\ 7\}.$$

Тогда

$$P_2 = C_1,\ P_3 = C_1 + C_2,\ P_4 = C_1 + C_2 + C_3, \qquad (7)$$

(P_1 представляется в виде пустой суммы).

Построением множеств (2) заканчивается этап II. Каждому из множеств C_i в сетевом графике, который будет построен, соответствует вершина. Она так же будет обозначаться C_i.

Ш. ***Построение сетевого графика***.

а) Вершинами сетевого графика служат построенные множества P_i $(1 \le i \le s)$ и C_j, ($1 \le j \le t$).

б) Работа k изображается дугой с началом в множестве полных предшественников работы k и ведущей в конституенту, содержащую работу k. На рис. 1 показан фрагмент сетевого графика, содержащий работу 4.

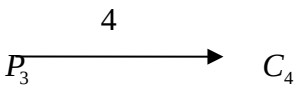


Рис. 1

в) Сетевой график содержит также пустые дуги, помогающие обеспечить требуемую упорядоченность работ. Эти дуги не соответствуют работам (или можно считать, что они изображают работы с нулевым временем выполнения) и называются фиктивными. Фиктивные дуги проводятся следующим образом: для каждого представления множеств P_i в виде суммы C_j:

$$P_i = C_{j1} + C_{j2} + \ldots + C_{jq}$$

проводятся фиктивные дуги из каждой вершины C_{j1}, C_{j2}, …, C_{jq} в вершину P_i.

Продолжение примера. Фиктивные дуги, соответствующие представлениям (7) для рассматриваемого нами примера показаны на рис. 2.

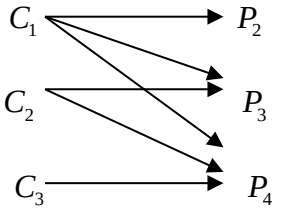


Рис. 2

Чтобы выяснить роль фиктивных дуг, рассмотрим, например, работу 4. Изображающая ее дуга начинается в вершине P_3. В силу (3) $P_3 = C_1 + C_2$. Фиктивные дуги "подводят" к вершине P_3 вершины C_1 и C_2, где заканчиваются дуги, изображающие работы 2 и 1, 3 – предшественники работы 4. Следовательно, работа 4 будет начинаться после завершения всех ей предшествующих.

IV. ***Упрощение графа***. Граф, построенный на предыдущем этапе можно упростить, удаляя из него некоторые фиктивные дуги (иногда все). Для этого применяются следующие два правила.

1°. *Если фиктивная дуга соединяет вершины, между которыми есть другой путь, то эту дугу следует удалить.*

2°. *Если единственная дуга, выходящая из вершины (входящая в вершину) является фиктивной, то эту дугу следует удалить и слить ее концы в одну вершину.*

Фактическое построение сетевого графика удобно совмещать с упрощениями 1°, 2°. Покажем как строится сетевой график в рассматриваемом примере. Работы помещаются на сетевой график в порядке их появления в исходной таблице, при этом одновременно производятся упрощения.

Продолжение примера. Работы 1 и 2 начинаются в P_1 и заканчиваются в C_2 и C_1 соответственно. Работа 3 начинается в P_2, согласно (3), надо провести фиктивную дугу из C_1 в P_2. На рис. 4 изображен фрагмент сетевого графика, построенный к этому моменту.

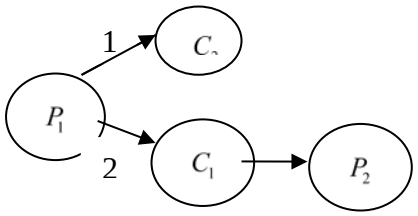


Рис. 4. Этап 1

Так как фиктивная дуга является единственной входящей в P_2 (и останется таковой до конца построения, потому что P_2 только один раз фигурирует в левых частях (7)), то вершины C_1 и P_2 можно слить в одну (рис.5).

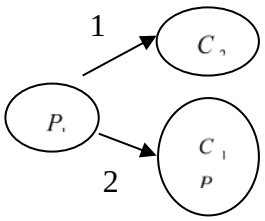


Рис. 5

В вершине (C_1, P_2) будет начинаться работа 3, которая заканчивается в вершине C_2. Чтобы изобразить работу 4, вводим новую вершину P_3 – начало работы 4. В силу (7) ее нужно соединить фиктивными дугами с вершинами C_1 и C_2 (рис. 6).

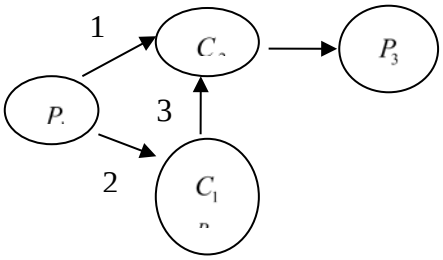


Рис. 6

Дугу от C_1 к P_3 можно удалить по правилу 1°, так как имеется путь $C_1 \to C_2 \to P_3$ между этими вершинами. После удаления дуга $C_2 - P_3$ будет единственной входящей в P_3 и вершины C_2 и P_3 можно слить в одну по правилу 2°. Действуя подобным образом, приходим окончательно к графу, показанному на рис. 7.

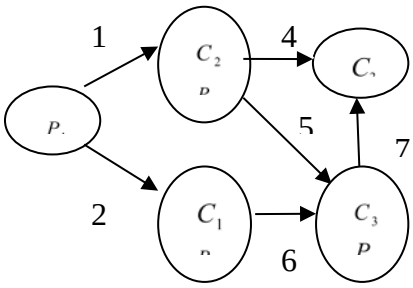


Рис. 7