МИНИСТЕРСТВО ОБЩЕГО И ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ

САМАРСКИЙ ГОСУДАРСТВЕННЫЙ АЭРОКОСМИЧЕСКИЙ
УНИВЕРСИТЕТ имени академика С.П. КОРОЛЕВА

АЛГОРИТМЫ ТЕОРИИ ГРАФОВ

Методические указания
к курсовой работе

САМАРА 1997

Составитель В.А. Колдоркина

УДК 519.(075)

Алгоритмы теории графов: Метод. указания к курсовой работе / Самар. гос. аэрокосм. ун-т; Сост. В.А.Колдоркина. Самара, 1997. 20 с.

В методических указаниях изложены некоторые алгоритмы теории графов и способы их реализации на ЭВМ.

Предназначены для студентов третьего курса вечернего отделения СГАУ 6-го факультета. Используются при выполнении курсовых работ по курсу "Дискретная математика". Выполнены на кафедре "Прикладная математика"

Печатается по решению редакционно-издательского совета Самарского государственного аэрокосмического университета имени академика С.П.Королева.

- рецензент А.Ф.Тарасюк

Способы представления графа

Множество самых разнообразных задач формулируется в терминах теории графов. Так, например, могут быть сформулированы задачи составления расписаний в исследовании операций, анализа сетей в электротехнике, установления структуры молекул в органической химии, сегментации программ в программировании, анализа цепей Маркова в теории вероятностей и т.д.. В задачах, возникающих в реальной жизни, оказывается, что соответствующие графы часто так велики, что их анализ не осуществим без ЭВМ, и поэтому эффективные алгоритмы решения задач теории графов имеют большое практическое значение.

Простота использования, так же как и эффективность алгоритма на графе, зависит от подходящего выбора представления графа.

Одним из наиболее распространенных представлений простого графа является матрица смежностей. Матрица смежностей графа $G = (V, E)$ есть $|V| \times |V|$ — матрица $A = [a_{ij}]$, в которой $a_{ij} = 1$, если в G существует ребро, идущее из i-ой вершины в j-ю, и $a_{ij} = 0$ в противном случае. У неориентированного графа матрица смежностей симметрична, и для ее представления достаточно хранить только верхний треугольник.

Матрица весов. Граф, в котором ребру (i, j) сопоставлено число μ_{ij}, называется взвешенным графом, а число μ_{ij} — весом ребра (i, j).

В сетях связи или транспортных сетях эти веса представляют некоторые физические величины, такие, как стоимость, расстояние, эффективность, емкость или надежность соответствующего ребра. Простой взвешенный граф может быть представлен своей матрицей весов $M = [\mu_{ij}]$, где μ_{ij} есть вес ребра, соединяющего вершины i и j.

Список ребер. Если граф является разреженным $|E| = o(|V|^2)$, то более эффективно представлять ребра графа парами вершин. Это представление можно реализовать двумя массивами

$$g = (g_1, g_2, \ldots g_{|E|})$$

и

$$h = h(h_1, h_2, \ldots g_{|E|}).$$

Каждый элемент в массиве есть метка вершины, а i-е ребро графа выходит из вершины g_i и входит в вершину h_i.

Структуры смежности. В ориентированном графе вершина y называется последователем другой вершины x, если существует ребро, направленное из x в y; вершина x называется тогда предшественником y. В случае неориентированного графа две вершины называются соседями, если между ними есть ребро.

Граф может быть описан его структурой смежности, т.е. списком всех последователей (соседей) каждой вершины. Структуры смежности могут быть удобно реализованы массивом из $|V|$ линейно связанных списков, где каждый список содержит последователей некоторой вершины x ($Adj(x)$).

Хранение списков смежности в виде связанного списка желательно для алгоритмов, в которых в графе добавляются или удаляются вершины.

Поиск в глубину

Рассмотрим один из методов обхода всех вершин и ребер графа. Этот метод исходит из процедуры прохождения графа методом поиска с возвращением. На неориентированном графе $G(V, E)$ поиск в глубину осуществляется следующим образом:

1. Движение начинаем с произвольной вершины $v \in V$ и идем по одному из ребер (v, w), инцидентному v. Возможны следующие варианты дальнейшего движения:
   а) Если вершина w уже пройдена (посещалась ранее), возвращаемся в v и выбираем другое ребро;
   б) Если вершина w еще не пройдена, то заходим в нее и применяем процесс рекурсивно к w;
   в) Если все ребра, инцидентные вершине v, уже просмотрены, идем назад по ребру (v, u), по которому пришли в v, и продолжаем исследование ребер, инцидентных u.
2. Обход заканчивается, когда все ребра будут помечены и произойдет возвращение в начальную вершину v.

Рассмотрим пример исследования поиском в глубину неориентированного графа, представленного структурой смежности.

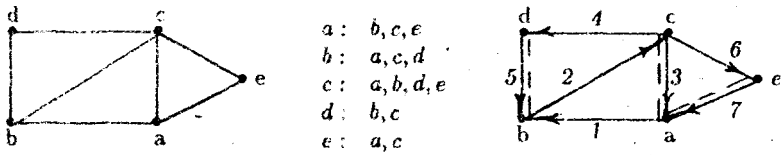


Рис. 1. Поиск в глубину на графе

Начинаем с вершины "a" и идем по первому встретившемуся ребру (a, b). Так как вершину "b" раньше не посещали, остаемся в "b" и проходим по первому непройденному ребру, инцидентному "b", а именно, (b, c).

В вершине "c" первым непройденным ребром оказывается ребро (c, a). Проходим (c, a) и обнаруживаем, что вершина "a" уже посещалась ранее. Поэтому возвращаемся в "c" и помечаем ребро (c, a) как обратное (пунктирной линией).

Вернувшись в "c", ищем другое непройденное ребро и идем по первому встретившемуся из них (c, d).

Так как "d" — новая вершина, остаемся в ней и идем по ребру (d, b). Поскольку вершина "b" посещалась ранее, помечаем ребро (d, b) как обратное и возвращаемся в "d". Из-за того, что непройденных ребер, инцидентных "d", не осталось, возвращаемся в "c".

В "c" находим непройденное ребро (c, e) и проходим его. Поскольку "e" — новая вершина, остаемся в "e" и ищем непройденное инцидентное ей ребро — (e, a). Так как "a" — уже пройденная вершина, помечаем ребро (e, a) как обратное.

Теперь вершина "e" полностью просмотрена, возвращаемся в "c". В "c" нет ребер, оставшихся непройденными, поэтому движемся назад в "b" и, наконец, в "a". Все ребра, инцидентные "a", пройдены, что означает конец процедуры.

Как видно из рис. 1, поиск в глубину превращает исходный неориентированный граф в орграф, индуцируя на каждом ребре направление. Числа около ребер графа указывают порядок, в котором проходились ребра.

Ребра графа, приводящие во время прохождения в новые вершины, образуют ориентированное остовное дерево (лес в случае несвязного графа), т.е. дерево T, являющееся остовом графа $G(V, E)$, в котором имеется выделенная вершина r, называемая корнем, и такая,

что имеется ориентированный путь из r в любую вершину T, но не существует ориентированного пути из любой вершины T в корень r.

Если в ориентированном остовном дереве имеется путь из вершины x в вершину y, то говорим, что x — предок y, y — потомок x.

Для реализации процедуры поиска в глубину необходимо отличать уже пройденные вершины от еще непройденных. Этого можно достигнуть путем постепенной нумерации вершин числами от 1 до $|V|$ по мере того, как в них попадаем.

Время поиска в глубину на неориентированном графе равно $O(|V| + |E|)$.

Поиск связных компонент графа

Подграфом графа $G(V, E)$ называется граф $G_1(V_1, E_1)$, у которого $V_1 \subset V$ и $E_1 \subset E$.

Подграф графа G, индуцированный подмножеством S множества V вершин графа G, — это подграф, который получается в результате удаления всех S вершин из V и всех ребер, инцидентных им.

Неориентированный граф G связен, если существует хотя бы один путь в G между каждой парой вершин v_i и v_j.

Ориентированный граф G связен, если неориентированный граф, полученный из G путем удаления ориентации ребер, является связным.

Ориентированный граф сильно связен, если для каждой пары вершин v_i и v_j существует, по крайней мере, один ориентированный путь из v_i в v_j и, по крайней мере, один из v_j в v_i.

Максимальный связный подграф графа G называется связной компонентой или просто компонентой G.

Несвязный граф состоит из двух или более компонент.

Максимальный сильно связный подграф называется сильно связной компонентой.

Простейшим случаем применения техники поиска в глубину на неориентированном графе является алгоритм отыскания связных компонент неориентированного графа.

В этом алгоритме в процессе реализации обхода вершин и ребер графа согласно поиску в глубину присваивается общий номер $N(v)$ каждой вершине w, принадлежащей компоненте, в которую попадает очередная исследуемая вершина v.

В случае примера на рис. 1 все вершины исследуемого графа получат один номер.

Если в графе несколько компонент, то алгоритм поиска в глубину реализуется для каждой из них, при этом $|N(v)|$ равно числу компонент связности исследуемого графа.

Топологическая сортировка вершин графа

Простейшим случаем использования техники поиска в глубину на орграфах является способ пометки вершин ациклического орграфа $G(V, E)$ числами $1, 2, \ldots, |V|$ так, что если из вершины i в вершину j идет ориентированное ребро, то $i < j$. Такой способ пометки называется топологической сортировкой вершин графа G.

Например, вершины орграфа на рис. 2*а* топологически отсортированы, а на рис. 2*б* — нет.

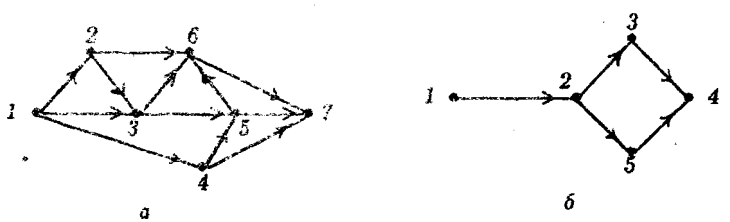


Рис. 2. Ациклические орграфы: топологически отсортированный (*а*) и топологически неотсортированный (*б*)

Топологическая сортировка может рассматриваться как процесс отыскания линейного порядка, в который может быть вложен данный частичный порядок.

Топологическая сортировка возможна тогда и только тогда, когда граф является ациклическим.

Топологическая сортировка полезна при анализе схем действия, где большой и сложный план представляется как орграф, вершины которого соответствуют целям плана, а ребра — действиям. Топологическая сортировка дает порядок, в котором могут быть достигнуты цели.

Алгоритм Т.С.:

1. Отыскивается вершина графа $G(V, E)$, из которой не выходят ребра (такая вершина существует, если G не имеет циклов).

2. Ей присваивается наибольший номер, а именно $|v|$, и она удаляется из графа G вместе с входящими в нее ребрами.

3. Повторяя этот процесс для графа, полученного в результате такого удаления, приписываем следующий наибольший номер $|v|-1$ вершине, из которой не выходят ребра, и т.д.

Для того, чтобы алгоритму потребовалось $O(|V|+|E|)$ *операций,* нужно избежать поиска в модифицированном орграфе вершины, из которой не выходят ребра. Это будет сделано, если провести единственный поиск в глубину на данном ациклическом орграфе G.

Двусвязность

Определение. Пусть $G(V, E)$ — связный неориентированный граф. Вершина a называется точкой сочленения графа G, или существуют такие вершины v и w, что v, w, a — различны и всякий путь между v и w содержит вершину a.

Иначе говоря, "a" — точка сочленения графа G, если удаление вершины "a" расщепляет граф G не менее, чем на две части.

Определение. Неориентированный связный граф двусвязен тогда и только тогда, когда в нем нет точек сочленения.

Определение. Максимальный двусвязный подграф графа называется двусвязной компонентой.

Определение. Граф, содержащий точку сочленения, называется разделимым.

Например, вершины b, f и i на рис. 3*а* являются точками сочленения.

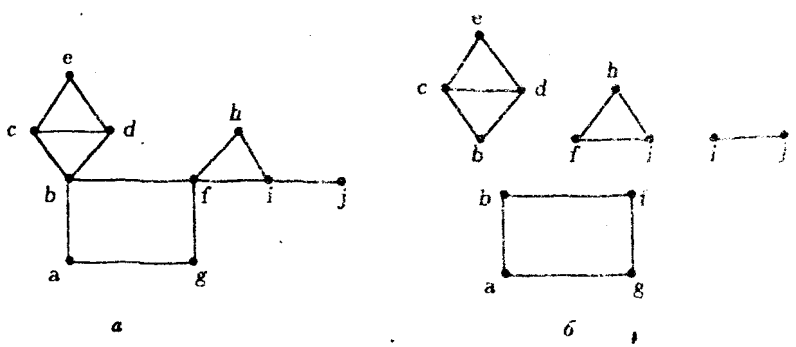


Рис. 3. Разделимый граф (*а*) и его двусвязные компоненты (*б*)

Отыскание точек сочленения и двусвязных компонент данного графа важно при изучении надежности коммуникационных и транспортных сетей. Это важно также и при определении других свойств графа, таких, например, как планарность.

Алгоритм поиска двусвязных компонент состоит в следующем:

1. Выбираем произвольную вершину v в V и строим глубинное остовное дерево (алгоритм "поиск в глубину") $T(V, E_1)$.
2. В процессе выполнения процедуры "поиск в глубину" при выборе вершины w помещаем ребро (v, w) в стек для ребер, если оно еще не там.
3. Если в процессе выполнения процедуры "поиск в глубину" обнаружена двусвязная компонента (найдена точка сочленения), то из стека выталкиваются все ребра, принадлежащие этой компоненте.

Для выполнения пункта 3 в процедуру "поиск" включается вычисление значения функции H для каждой вершины с помощью равенства:

$$H[v] = \min\left\{(v) \cup (H[s], \text{ где } s - \text{сын } v) \cup (w, (v, w) \in B)\right\}.$$

Здесь B — множество обратных ребер, полученных в результате реализации процедуры "поиск в глубину" (пунктирные на рис. 1).

Начальное значение функции $H[v]$ равно ее максимально возможному значению. В дальнейшем значение $H[v]$ корректируется. По значению $H[v]$ легко найти точки сочленения, а именно:

если обнаруживается пара (v, w), для которой w сын v и $H[w] \geqslant v$, то вершина v — есть точка сочленения. Выталкиваются все ребра вплоть до (v, w).

Рассмотрим пример на рис. 4.

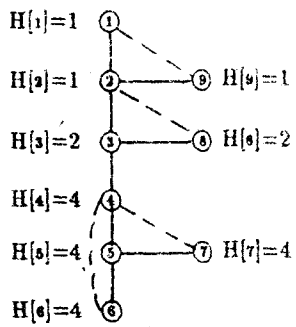


Рис. 4. Поиск точек сочленения с помощью функции H[v]

Здесь процедура "поиск" устанавливает, что H[6]=4, так как есть обратное ребро (6,4). Тогда полагается H[5]=4, поскольку 4 меньше начального значения H[5]=5. Закончив "поиск 5", обнаруживаем, что H[5]=4 и точка 4 — точка сочленения (H[w]≥v)

В этот момент стек содержит такие ребра (от дна к вершине): (1,2),(2,3),(3,4),(4,5),(5,6),(6,4),(5,7), (7,4).

Выталкиваются все ребра вплоть до (4,5), т.е. на выход подаются ребра (7,4),(5,7),(6,4),(5,6),(4,5), которые являются ребрами первой найденной двусвязной компоненты, и т.д.

Закончив "поиск (2)", обнаруживаем, что H[2]=1 и опустошаем стек, хотя 1 — не точка сочленения. Это гарантирует, что двусвязная компонента, содержащая корень, тоже будет найдена.

Остов минимального веса

Деревом называется связный граф без циклов.

Теорема (об эквивалентных определениях дерева).

Если в графе выполняются любые два из перечисленных ниже условий, то выполняется и третье:

1) граф связен;

2) граф не имеет циклов;

3) число ребер графа на единицу меньше числа его вершин.

Определение. Граф $G_1(X_1, Y_1)$ называется суграфом графа $G(X, Y)$, если $X_1 = X$, а $Y_1 \subset Y$.

5 Определение. Остовным деревом графа $G(X, Y)$ называется его суграф — дерево, связывающее все вершины из множества X.

Рассмотрим следующую задачу: во взвешенном связном графе найти остов минимального веса, т.е. дерево, у которого сумма весов его ребер минимальна.

Эта задача возникает при проектировании линий электропередачи, трубопроводов, дорог и т.п., когда требуется заданные центры соединить некоторой системой каналов связи так, чтобы любые два центра были связаны либо непосредственно соединяющим их каналом, либо через другие центры и каналы, и чтобы общая длина (или, например, стоимость) каналов связи была минимальной.

Для решения этой задачи имеется эффективный алгоритм Краскала:

1. Выбираем в графе самое легкое ребро и относим его к остову.
2. Если ребра $u_1, \ldots, u_i$ уже выбраны, то из оставшихся выбираем то, которое не образует цикла с уже выбранными и имеет среди всех таких ребер наименьший вес.
3. Построение остова заканчиваем тогда, когда добавление любого из оставшихся ребер ведет к образованию цикла.

Пример реализации этого алгоритма рассмотрен на рис. 5.

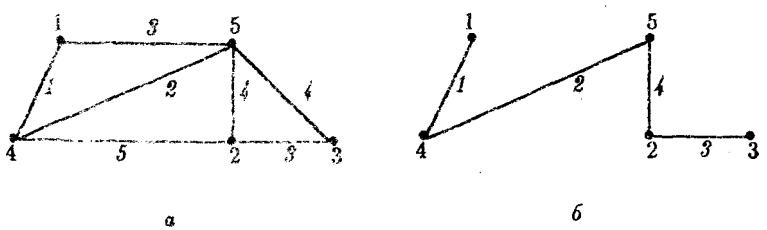


Рис. 5. Взвешенный граф (*а*) и его остов минимального веса (*б*)

При реализации этого алгоритма можно проделать следующее:

1. Построить таблицу весов ребер графа по возрастанию.
2. Используя таблицу весов по возрастанию как исходную, работать с набором вершин графа в цикле; при этом необходимо рассмотреть четыре случая:

   а) вершины очередного ребра не принадлежат ни одной компоненте связности из уже построенных. В этом случае формируется новая компонента связности;

   б) только одна из вершин вносимого ребра принадлежит к какой-нибудь компоненте связности. Ребро заносится в эту компоненту, а общее число компонент связности не меняется;

в) обе вершины входят в уже составленные компоненты связности, но в разные. Поскольку появилось связующее звено, то эти компоненты объединяются в одну. Число компонент связности при этом уменьшается на единицу;

г) обе вершины исследуемого ребра принадлежат одной компоненте. Ребро не вносится в остов, так как образует цикл.

По окончании каждого этапа цикла необходима проверка, нет ли в уже имеющихся компонентах связности общей вершины, что дает право объединять эти компоненты в одну, а их общее число при этом уменьшить на единицу.

Транзитивное замыкание орграфа

Определение. Ориентированным **маршрутом** в орграфе G называется такая последовательность $S = (v_0, x_1, v_1, x_2 \ldots x_n, v_n)$ его чередующихся вершин v_i и дуг x_j, что $x_i = (v_{i-1}, v_i)$ $(i = \overline{1,n})$.

Длиной маршрута называется число входящих в него дуг.

Определение. Маршрут называется **цепью**, если все входящие в него дуги различны, и **путем**, если все входящие в него вершины, кроме, возможно, крайних, различны.
Если в орграфе G нет параллельных дуг, то маршрут может быть задан последовательностью входящих в него вершин: $S = (v_0, v_1, \ldots, v_n)$. В любом случае маршрут можно задать последовательностью входящих в него дуг: $S = (x_1, x_2, \ldots, x_n)$.

Определение. Маршрут называется **циклическим**, если его первая и последняя вершины совпадают. Циклический путь называется **контуром**.

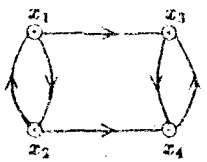
В том случае, когда необходимо установить лишь факт наличия в орграфе путей между вершинами, возникает задача о построении транзитивного замыкания орграфа.

Определение. Транзитивным замыканием орграфа G называется орграф G^* с тем же множеством вершин, что и G, но в котором дуга из x_i в x_j идет тогда и только тогда, когда вершина x_j достижима в G из x_i.

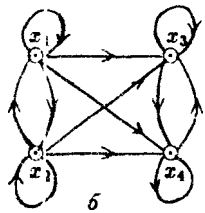
Матрицей смежностей транзитивного замыкания G^* орграфа G служит матрица достижимости $R(G)$ графа G.

Пример. Для графа G на рис.6 матрица достижимости $R(G)$ имеет вид

$$R(G) = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 \end{bmatrix}$$



$$A(G) = \begin{bmatrix} 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$



а *б*

Рис.6. Граф G и его матрица смежностей $A(G)$ (*а*); транзитивное замыкание G^* графа G (*б*)

Простейший, но не самый эффективный из известных алгоритмов построения матрицы достижимости состоит в последовательном изменении строк матрицы смежности до получения строки матрицы достижимости.

Можно вычислить матрицу A^* по матрице A, определив последовательность матриц $A^{(0)} = [a_{ij}^{(0)}]$; $A^{(1)} = [a_{ij}^{(1)}]$; ...; $A^{|V|} = [a_{ij}^{|V|}]$ следующим образом:

$$\begin{aligned} a_{ij}^{(0)} &= a_{ij}, \\ a_{ij}^{(\ell)} &= a_{ij}^{(\ell-1)} \vee (a_{i\ell}^{(\ell-1)} \wedge a_{\ell j}^{(\ell-1)}). \end{aligned} \qquad (1)$$

При этом $A^* = A^{(|V|)}$, V — множество вершин графа. Для того, чтобы понять, почему это так, заметим, что по индукции легко доказать, что $a_{ij}^{(\ell)} = 1$ тогда и только тогда, когда существует путь из x_i в x_j с промежуточными вершинами только из множества $\{x_1\ x_2, \ldots, x_\ell\}$.

Для $\ell = 0$ это очевидно (путь без промежуточных вершин).

Если это верно для $\ell - 1$, то $a_{ij}^{(\ell)} = 1$ тогда и только тогда, когда выполняется одно или оба из условий:

1. Существует путь от x_i до x_j, все промежуточные вершины которого принадлежат множеству $\{x_1, \ldots, x_{\ell-1}\}$.
2. Существуют пути от x_i до x_ℓ и от x_ℓ до x_j, все промежуточные вершины которых принадлежат множеству $\{x_1, \ldots, x_{\ell-1}\}$.

Итак, утверждение справедливо для ℓ.

С целью экономии объема памяти можно соотношение (1) рассматривать в другой форме.

Заметим, что если задана матрица $A^{(\ell-1)}$, то ее можно преобразовать в $A^{(\ell)}$ следующим образом. Если $a_{il}^{(\ell-1)} = 0$, то просто имеем $a_{ij}^{(\ell)} = a_{ij}^{(\ell-1)}$, и если $a_{il}^{(\ell-1)} = 1$, имеем $a_{ij}^{(\ell)} = a_{ij}^{(\ell-1)} \vee a_{\ell j}^{(\ell-1)}$. Обозначив i-тую строку матрицы A через $a_{i,*}$, получим:

$$a_{i,*}^{(\ell)} = \begin{cases} a_{i,*}^{(\ell-1)} & \text{, если} \quad a_{il}^{(\ell-1)} = 0 \\ a_{i,*}^{(\ell-1)} \vee a_{\ell,*}^{(\ell-1)} & \text{, если} \quad a_{il}^{(\ell-1)} = 1 \end{cases} \tag{2}$$

Здесь в самом внутреннем цикле (а их будет два — по ℓ и по i) основной является операция "или" двух строк матрицы A.

Поиск в ширину

Поиск в ширину следующим образом приписывает вершинам рассматриваемого графа номера $0, 1, 2, \ldots$.

1. Начиная с произвольной вершины, приписываем ей номер 0.
2. Каждой вершине из окружения вершины 0 приписываем номер 1.
3. Рассматриваем поочередно окружения всех вершин с номером 1 и каждой из входящих в эти окружения вершин (т.е. смежных с 1), еще не занумерованных, приписываем номер 2.
4. Рассматриваем окружения всех вершин с номером 2 и т.д., пока возможно.

Если исходный граф связен, то поиск в ширину занумерует все его вершины.

С помощью поиска в ширину можно решить следующие задачи:

1) разбить множество вершин графа на его области связности;
2) для несовпадающих вершин u и v связного графа найти кратчайшую (u, v)-цепь;
3) в ориентированном графе найти множество всех вершин, достижимых из заданной вершины v;
4) выяснить, исходный граф двудольный или нет.

Рассмотрим подробнее последнюю задачу.

Определение. Граф называется двудольным, если существует такое разбиение множества его вершин на две части (доли), что концы каждого ребра принадлежат разным частям.

Если при этом любые две вершины, входящие в разные доли, смежны, то граф называется полным двудольным.

Полный двудольный граф, доли которого состоят из p и из q вершин, обозначается $k_{p,q}$. Например, полный двудольный граф $k_{3,3}$ изображен на рис.7.

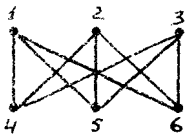


Рис.7. Полный двудольный граф $k_{3,3}$

Известен следующий критерий двудольности графа: теорема Кенига. Для двудольности графа необходимо и достаточно, чтобы он не содержал циклов нечетной длины.

Очевидно следствие. Граф является двудольным тогда и только тогда, когда он не имеет простых циклов нечетной длины.

С целью реализации критерия двудольности сначала все вершины графа нумеруются методом поиска в ширину.

Далее множество всех вершин графа разобьем на две части — A и B, отнеся к A все вершины с четными номерами, а к B — все остальные вершины, и рассмотрим порожденные подграфы $G(A)$ и $G(B)$.

Если оба они пусты, то исходный граф G — двудольный. В противном случае граф G не является двудольным.

Заметим, что для выяснения двудольности графа достаточно проверить, что все пары вершин с равными номерами по поиску в ширину не смежны.

На рис.8 изображен двудольный граф с вершинами, пронумерованными методом поиска в ширину. Легко видеть, что в нем все вершины с равными номерами не смежны.

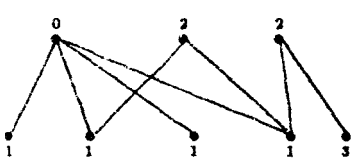


Рис.8. Двудольный граф, вершины которого пронумерованы методом поиска в ширину

Кратчайшие пути

Пусть $G(V, E)$ — ориентированный взвешенный граф.

Задача о кратчайшем пути состоит в отыскании пути минимального

веса, соединяющего заданные начальную и конечную вершины графа при условии, что хотя бы один такой путь существует.

Начальную и конечную вершины обозначим соответственно через s и t, (s,t) — путь минимального веса будем называть кратчайшим (s,t)-путем.

Эффективным алгоритмом построения кратчайшего пути в графе с неотрицательными весами дуг ($w(\ell)\geqslant 0$) является алгоритм Дейкстры.

На каждой итерации этого алгоритма всякая вершина v графа G имеет метку $\ell(v)$, которая может быть постоянной или временной. В первом случае $\ell(v)$ — вес кратчайшего (s,v)-пути. Если же метка $\ell(v)$ временная, то $\ell(v)$ — вес кратчайшего (s,v)-пути, проходящего через вершины, уже имеющие постоянные метки.

Таким образом, временная метка $\ell(v)$ является оценкой сверху для веса кратчайшего (s,v)-пути и, став на некоторой итерации постоянной, она остается такой до конца работы алгоритма.

Кроме $\ell(v)$ с каждой вершиной v графа G, за исключением s, связывается еще одна метка — $\theta(v)$. На каждой итерации $\theta(v)$ 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 $\theta(v)$ легко указать последовательность вершин, составляющих кратчайший (s,t)-путь.

Перед началом первой итерации алгоритма вершина s имеет постоянную метку $\ell(s)=0$, а метки всех остальных вершин равны ∞ и эти метки временные. На каждом шаге итерации метки меняются следующим образом:

1. Каждой вершине v, не имеющей окончательной метки, присваивается новая временная метка — наименьшая из ее временной метки $\ell(v)$ и чисел $w(p,v)$ + окончательная метка p, где p – вершина, которой присвоена постоянная метка на предыдущем шаге. Таким образом,

$$\ell(v)^{(i+1)\,max} = \min_{p\forall v}\left\{\ell(v)^{(i)\,max};\ \left(\ell(p)^{(i)\,max} + w(p,v)\right)\right\}.$$

2. Находится наименьшая из всех временных меток, которая и становится постоянной меткой рассматриваемой вершины. В случае равенства выбирается любая из них.

Алгоритм заканчивает работу, когда метка $l(t)$ становится постоянной.

Первой помеченной вершиной будет s, которая находится на нулевом расстоянии от себя. Следующей, получившей постоянную метку, будет вершина, ближайшая к s. Третьей вершиной, получившей постоянную метку, будет вторая по близости к s, и т.д.. Легко видеть, что постоянная метка каждой вершины — это кратчайший путь от s до этой вершины.

Будем считать, что граф G задан матрицей весов.

Итак, алгоритм Дейкстры состоит в следующем:

1. Положить $\ell(s) := 0$ и считать эту метку постоянной. Положить $\ell(v) := \infty$ для всех $v \in V$ графа $G(V, E)$, $v \neq s$ и считать эти метки временными. Положить $p := s$.
2. Для всех v, смежных с p и имеющих временные метки, выполнить: если $\ell(v) > \ell(p) + w(p, v)$, то $\ell(v) := \ell(p) + w(p, v)$ и $\theta(v) := p$. Иначе $\ell(v)$ и $\theta(v)$ не меняется.
3. Пусть V' — множество вершин с временными метками ℓ. Найти вершину v^*, такую, что $\ell(v^*) = \min\limits_{v \in V'} \ell(v)$. Считать метку $\ell(v^*)$ постоянной меткой вершины v^*.
4. $p := v^*$. Если $p = t$, то перейти к п.5 [$\ell(t)$ — вес кратчайшего пути]. Иначе, перейти к п.2.
5. $p^* := (s, \ldots, \theta^3(t), \theta^2(t), \theta(t), t)$, где p^* — кратчайший путь.

Замечание. Легко видеть, что этот алгоритм применим и к неориентированным графам. Для этого достаточно неориентированное ребро uv графа, имеющее вес $w(u, v)$, рассматривать как пару дуг (u, v) и (v, u) того же веса.

Замечание. Если п.4 модифицировать так, чтобы алгоритм заканчивал работу только после получения всеми вершинами постоянных меток, то он будет строить кратчайшие пути из s в каждую из остальных вершин.

Если вместе с превращением метки вершины v^* в постоянную запосить дугу $(\theta(v^*), v^*)$ в множество A^*, то после окончания работы алгоритма граф $D = (V, A^*)$ будет корневым ориентированным остовным деревом. Это дерево называется деревом кратчайших путей из s графа G.
Для любой вершины $v \in V$ графа G единственный (s, v)–путь в дереве D является кратчайшим (s, v)–путем в графе G.

Замечание. Рассмотренный алгоритм, модифицированный так, как указано в предыдущем замечании, можно рассматривать как алгоритм построения дерева D кратчайших путей из вершины s графа G.

Пример.

На рис.9 изображены пять копий графа G, каждая из которых отражает ситуацию, сложившуюся после очередной итерации алгоритма. Около каждой дуги написан ее вес. Вершинам приписаны метки, полученные ими в результате предыдущих итераций.

Некоторые дуги обведены жирными линиями, т.е. отмечены. Добавление такой дуги (a, b) при переходе к $(k+1)$-й итерации означает, что вершина "b" получила свою метку $l(b)$ из a и эта метка стала постоянной на $(k+1)$-й итерации. Вершина t в этом примере получает постоянную метку последней, и отмеченные дуги в последней копии G образуют дерево кратчайших путей из s.

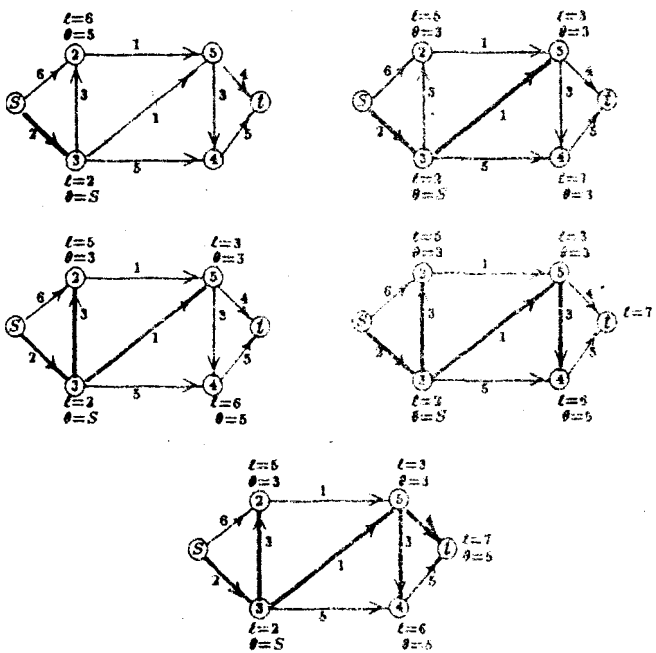


Рис.9. Пять копий графа G, отражающих ситуацию, сложившуюся после каждой очередной итерации алгоритма Дейкстры.

Клики

Определение. Подмножество V' вершин графа $G(V, E)$ называется кликой, если любые две входящие в него вершины смежны, т.е. если порожденный подграф $G(V', E')$ является полным.

Определение. Клика называется максимальной, если она не содержится в клике с бо́льшим числом вершин, и наибольшей, если число вершин в ней наибольшее среди всех клик. Число вершин в наибольшей клике графа G называется его плотностью (или кликовым числом) и обозначается через $\varphi(G)$.

На рис. 10 представлен граф и его клики.

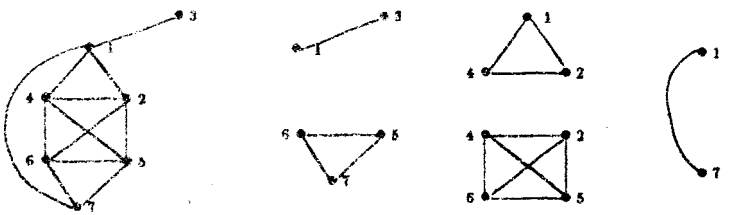


Рис.10. Граф G и его максимальные клики

Определение клик графа полезно в таких областях, как информационный поиск и социология.

Рассмотрим следующий алгоритм порождения максимальных клик:

1) Берем очередную вершину (a), начиная с первой, и оформляем ее в клику.
2) Выбираем ближайшую по номеру вершину, смежную с (a), и заносим ее в ту же клику.
3) Выбираем следующую по номеру вершину (b), смежную с (a), и сравниваем ее со всеми вершинами в уже существующей клике от (a).

   Если вершина (b) смежна со всеми вершинами рассматриваемой клики, то добавляем ее в эту клику.

   Если вершина (b) смежна лишь с некоторыми из вершин рассматриваемой клики, оформляем это подмножество вершин и вершину (b) в отдельную клику.

4) Повторяем п. 3, пока не просмотрим все вершины, смежные с (a).

5) Повторяем пункты 1,2,3,4, пока не просмотрим все вершины.

6) Исключаем те клики, которые являются подмножествами более крупных клик.

Этот алгоритм представляет собой естественный поиск клик с возвращением, т.е. поиск, в котором не делается никаких попыток упростить дерево поиска.

Каждый узел в дереве поиска соответствует полному подграфу графа, и каждое ребро соответствует вершине графа.

Сын данного узла c получается добавлением к c вершины $x \notin c$, которая смежна с каждой вершиной из c. Ребро, идущее от c к сыну $c \cup \{x\}$, соответствует вершине x.

На рис. 11 показаны некоторый граф G и дерево поиска T, которое проходится в процессе естественного поиска с возвращением.

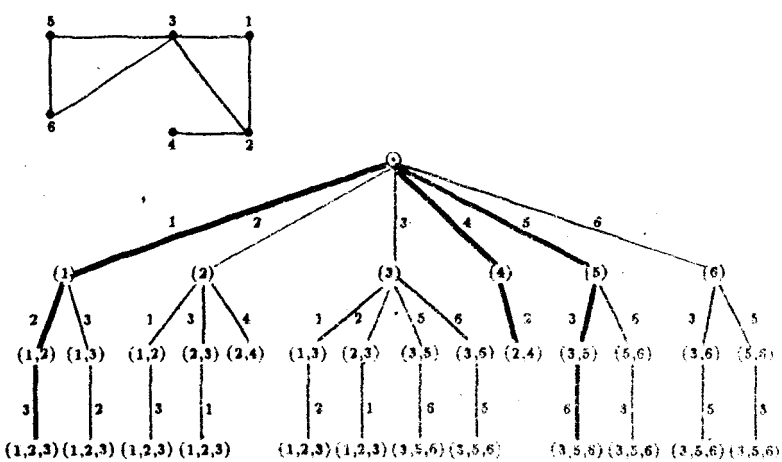


Рис. 11. Граф G и результат поиска с возвращением без ограничений клик G

Заметим, что при реализации рассматриваемого алгоритма каждая клика порождается много раз: каждое из подмножеств $\{1,2,3\}$ и $\{3,5,6\}$ порождается шесть раз, а подмножество $\{2,4\}$ — дважды. В общем случае клика размером k порождается $k!$ раз.

На рис. 11 все ребра, изображенные тонкими линиями, можно оборвать, если воспользоваться следующим известным результатом:

1. Поддерево с корнем в $S \cup \{v_i\}$ не нужно исследовать, если поддерево с корнем $S \cup \{x\}$ уже исследовалось и $v_i \in Adj(x)$.
2. Пусть S — узел в дереве поиска T, и пусть $\widehat{S} \subset S$ — собственный предок S в T. Если все поддеревья узла $\widehat{S} \cup \{x\}$ уже исследованы, так что порождены все клики, включающие $\widehat{S} \cup \{x\}$, то все неисследованные поддеревья с корнями $S \cup \{x\}$ можно проигнорировать.

Таким образом, если воспользоваться результатами 1,2, то дерево поиска T будет значительно урезано, и каждая максимальная клика будет порождаться только один раз.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. **Евстигнеев В.А.** Применение теории графов в программировании. М.: Наука, 1985.
2. **Кузнецов О.П., Адельсон-Вельский Г.М.** Дискретная математика для инженеров. М.: Энергоиздат, 1988.
3. **Оре О.** Теория графов. М.: Наука, 1980.
4. **Рейнгольд и др.** Комбинаторные алгоритмы. М.: Мир, 1980.
5. **Емеличев Е.А. и др.** Лекции по теории графов. М.: Наука, 1990.
6. **Ахо Х. и др.** Построение и анализ вычислительных алгоритмов. М.: Мир, 1979.
7. **Свами М., Тхуласираман К.** Графы, сети и алгоритмы. М.: Мир, 1984.

СОДЕРЖАНИЕ

АЛГОРИТМЫ ТЕОРИИ ГРАФОВ

Составитель Колдоркина Валентина Александровна

Редактор Т.К.Кретинина
Техн. редактор Г.А.Усачева
Корректор Т.К.Кретинина

Подписано в печать 12.03.96. Формат 60x84 1/16
Бумага офсетная. Печать офсетная.
Усл. печ. л. 1,62. Усл. кр.-отт. 1,74. Уч.-изд. л. 1,7.
Тираж 75 экз. Заказ 99. Арт. С – 43/97.

Самарский государственный аэрокосмический университет имени академика С.П.Королева. 443086 Самара, Московское шоссе, 34.

РИО Самарского государственного аэрокосмического университета им. академика С.П.Королева. 443001 Самара, ул. Молодогвардейская, 151.