Тема 6. Алгоритмы на графах

6.6. Топологическая сортировка

Топологической сортировкой называется способ нумерации вершин ациклического орграфа $G = (V, E)$ целыми числами 1, 2, …, $|V|$ так, что если из вершины с номером i в вершину с номером j идет ориентированное ребро (дуга), то $i < j$. Ясно, что если орграф содержит цикл, то его топологическая сортировка невозможна. Особенностью ациклического орграфа является то, что он обязательно имеет, по крайней мере, одну вершину, в которую не входят дуги, и одну вершину, из которой не выходят дуги.

Одно из решений этой задачи заключается в следующем. Выбирается произвольная вершина графа, из которой не выходят дуги. Выбранной вершине присваивается наибольший номер $|V|$. Затем эта вершина удаляется из графа вместе с входящими в нее дугами. Поскольку оставшийся орграф также ациклический, процесс повторяется и присваивается следующий наибольший номер $|V| - 1$ вершине, из которой не выходят дуги, и т. д. Чтобы сделать топологическую сортировку более эффективной, нужно, чтобы при поиске вершины, из которой не выходят дуги, избежать просмотра каждого модифицированного орграфа. В этом может помочь поиск в глубину. Причем производится единственный поиск в глубину на данном ациклическом орграфе G. Дополнительно требуется массив *label* размера $|V|$ для записи номеров топологически отсортированных вершин. Если в орграфе G имеется дуга (v, w), то $label(v) < label(w)$.

Процедура топологической сортировки ациклического орграфа $G = (V, E)$ приведена в алгоритме 6.13. Ациклический ориентированный граф, заданный структурой смежности, и результат его топологической сортировки приведены на рис. 6.8, *а* и *б*.

for $x \in V$ **do** {// инициализация; $new(x) \leftarrow$ **true**; $label(x) \leftarrow 0$}

$i \leftarrow |V| + 1$

for $x \in V$ **do**

  if $new(x)$ **then** {// x не имеет пронумерованного предка; $TOPSORT(x)$}

procedure $TOPSORT(v)$

  $new(v) \leftarrow$ **false**

  for $w \in \mathrm{Adj}(v)$ **do** {// обработать всех потомков w; **if** $new(w)$ **then** $TOPSORT(w)$ **else if** $label(w) = 0$ **then** // G имеет цикл}

  // v пронумеровано числом, меньшим числа,

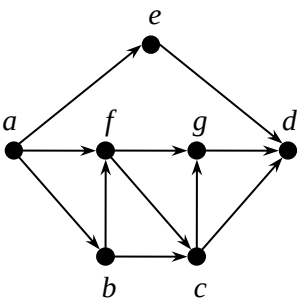
  // приписанного любому потомку

  $i \leftarrow i - 1$

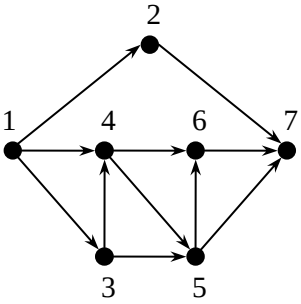
  $label(v) \leftarrow i$

return

Алгоритм 6.13. Топологическая сортировка орграфа



v	Adj(v)
a	b, e, f
b	c, f
c	d, g
d	–
e	d
f	c, g
g	d



а *б*

Рис. 6.8. Топологическая сортировка ациклического орграфа:
а – орграф и его структура смежности;
б – результат топологической сортировки орграфа

i: 8

	a	b	c	d	e	f	g
new(x)	+	+	+	+	+	+	+
$v = a$	–	+	+	+	+	+	+
$v = b$	–	–	+	+	+	+	+
$v = c$	–	–	–	+	+	+	+
$v = d$	–	–	–	–	+	+	+
$v = g$	–	–	–	–	+	+	–
$v = f$	–	–	–	–	+	–	–
$v = e$	–	–	–	–	–	–	–
label(x)	0	0	0	0	0	0	0
$v = d$ $i = 7$	0	0	0	7	0	0	0
$v = g$ $i = 6$	0	0	0	7	0	0	6
$v = c$ $i = 5$	0	0	5	7	0	0	6
$v = f$ $i = 4$	0	0	5	7	0	4	6
$v = b$ $i = 3$	0	3	5	7	0	4	6
$v = e$ $i = 2$	0	3	5	7	2	4	6
$v = a$ $i = 1$	1	3	5	7	2	4	6

Время работы алгоритма топологической сортировки равно $O(|V| + |E|)$, так как каждая дуга проходится один раз и для каждой вершины один раз вызывается *TOPSORT*.

6.7. Транзитивное замыкание

В прикладных интерпретациях теории графов часто возникают задачи, требующие ответа на вопрос, какие вершины графа связаны между собой. Если граф неориентированный, то задача сводится к определению его связных компонент (см. алгоритм 6.10). Если же граф ориентированный, то задача формулируется как определение наличия ориентированных путей между вершинами орграфа. В этом случае решение задачи становится более сложным и сводится к построению *транзитивного замыкания* орграфа.

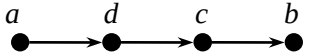
Орграф является одним из способов представления бинарного отношения. Бинарное отношение R на некотором множестве M есть набор упорядоченных пар элементов этого множества. Если xRy, $x, y \in M$, то отношение R представляется в графе дугой (x, y). Отношение R является транзитивным, если для любых элементов $x, y, z \in M$ выполняется условие: если xRy и yRz, то xRz. Транзитивное замыкание бинарного отношения R – это отношение R^*, определяемое следующим образом: xR^*y, если существует последовательность из k элементов $x_1 = x, x_2, x_3, \ldots, x_k = y$, в которой между соседними элементами выполнено отношение R: $x_1Rx_2, x_2Rx_3, \ldots, x_{k-1}Rx_k$. Очевидно, что если xRy, то xR^*y, т. е. $R \subseteq R^*$.

Пусть $G = (V, E)$ – орграф, представляющий отношение E. Орграф $G^* = (V, E^*)$, представляющий транзитивное замыкание E^* отношения E, называется транзитивным замыканием графа G. Дуга (x, y), $x \neq y$, принадлежит E^* тогда и только тогда, когда в графе G существует ориентированный путь из вершины x в вершину y. Очевидно, что петля $(x, x) \in E^*$ тогда и только тогда, когда в графе G существует ориентированный цикл, включающий вершину x.

Пусть орграф $G = (V, E)$ представлен матрицей смежности $A = [a_{ij}]$ размера $|V| \times |V|$, где $a_{ij} = 1$, если имеется дуга $(i, j) \in E$, и $a_{ij} = 0$ в противном случае, $i, j \in V$. Пусть $G^* = (V, E^*)$ транзитивное замыкание графа G. Тогда матрица смежности $A^* = [a^*_{ij}]$ графа G^* представляет собой матрицу связности (достижимости) графа G, в которой $a^*_{ij} = 1$, если в графе G имеется ориентированный путь из i в j, и $a^*_{ij} = 0$, если такого пути нет. Пусть вершины графа обозначаются числами 1, 2, …, $|V|$. Построение транзитивного замыкания G^* графа G сводится к вычислению матрицы A^* по матрице A. Для этого определяется последовательность матриц $A^0 = [a^0_{ij}]$, $A^1 = [a^1_{ij}]$, …, $A^{|V|} = [a^{|V|}_{ij}] = A^*$ следующим образом:

$$a^0_{ij} = a_{ij},$$
$$a^l_{ij} = a^{l-1}_{ij} \vee (a^{l-1}_{il} \wedge a^{l-1}_{lj}),$$

т. е. матрица A^l ($l \geq 1$) получается из матрицы A^{l-1} после обработки вершины l в графе, соответствующей матрице A^{l-1}. Очевидно, что $a^l_{ij} = a^{l-1}_{ij} \vee (a^{l-1}_{il} \wedge a^{l-1}_{lj})$ равно единице тогда и только тогда, когда либо $a^{l-1}_{ij} = 1$ (существует путь из i в j, использующий вершины только из множества $\{1, 2, \ldots, l-1\}$), либо a^{l-1}_{il} и a^{l-1}_{lj} одновременно равны единице (имеются пути из i в l и из l в j, использующие вершины только из множества $\{1, 2, \ldots, l-1\}$).



A^0	a	b	c	d
a	0	0	0	1
b	0	0	0	0
c	0	1	0	0
d	0	0	1	0

A^a	a	b	c	d
a	0	0	0	1
b	0	0	0	0
c	0	1	0	0
d	0	0	1	0

A^b	a	b	c	d
a	0	0	0	1
b	0	0	0	0
c	0	1	0	0
d	0	0	1	0

A^c	a	b	c	d
a	0	0	0	1
b	0	0	0	0
c	0	1	0	0
d	0	1	1	0

A^d	a	b	c	d
a	0	1	1	1
b	0	0	0	0
c	0	1	0	0
d	0	1	1	0

$$a_{ij}^0 = a_{ij},$$
$$a_{ij}^l = a_{ij}^{l-1} \vee (a_{il}^{l-1} \wedge a_{lj}^{l-1}),$$

Рассмотренное соотношение полезно для понимания способа вычисления A^*, но оно неудобно для практических вычислений, так как желательно преобразовать A в A^* на месте, используя фиксированный объем дополнительной памяти. Пусть дана матрица A^{l-1}. Ее можно преобразовать в матрицу A^l следующим образом. Если $a_{il}^{l-1} = 0$, то $a_{ij}^l = a_{ij}^{l-1}$. Если $a_{il}^{l-1} = 1$, то $a_{ij}^l = a_{ij}^{l-1} \vee a_{lj}^{l-1}$. Обозначив i-ю строку матрицы A через $a_{i,*}$, получим

$$a_{i,*}^l = \begin{cases} a_{i,*}^{l-1}, \text{ если } a_{il}^{l-1} = 0, \\ a_{i,*}^{l-1} \vee a_{l,*}^{l-1}, \text{ если } a_{il}^{l-1} = 1. \end{cases}$$

Для $i \neq l$ значение $a_{i,*}^{l-1}$ используется только при вычислении $a_{i,*}^l$, поэтому его значения могут меняться, не влияя на вычисления $a_{k,*}^l$ для $k \neq i$. Элемент $a_{l,*}^{l-1}$, хотя и используется для вычисления других строк, но $a_{l,*}^l = a_{l,*}^{l-1}$, поэтому никакие ложные дуги не добавляются.

Рассмотренный способ вычисления транзитивного замыкания ориентированного графа $G = (V, E)$ реализован алгоритмом 6.14, известным как алгоритм Уоршелла (Warshall). Очевидно, что данный алгоритм требует $O(|V|^3)$ операций.

Если строка матрицы A помещается в машинное слово, то самый внутренний цикл **for** с параметром k можно выполнить за одну операцию логического сложения двух строк матрицы. Экономия получается, даже если строка требует нескольких слов. Соответствующая модификация алгоритма представлена алгоритмом 6.15 и заключается в замене цикла **for** с параметром k на одну операцию присваивания $a_{i,*} \leftarrow a_{i,*} \vee a_{l,*}$. Данный алгоритм является в общем случае наиболее эффективным алгоритмом вычисления транзитивного замыкания.

```
for l ← 1 to |V| do
  for i ← 1 to |V| do
    if a_il = 1 then for k ← 1 to |V| do a_ik ← a_ik ∨ a_lk
```

Алгоритм 6.14. Алгоритм Уоршелла вычисления транзитивного замыкания

```
for l ← 1 to |V| do
  for i ← 1 to |V| do if a_il = 1 then a_i,* ← a_i,* ∨ a_l,*
```

Алгоритм 6.15. Модификация алгоритма Уоршелла

	a	в	c	d
a				1
в				
c		1		
d			1	

	a	в	c	d
a		(1)	(1)	1
в				
c		1		
d		(1)	1	

Неправильно

	a	в	c	d
a		?	(1)	1
в				
c		1		
d		(1)	1	

Алгоритм Уоршелла обрабатывает все дуги, заходящие в вершину, до начала обработки следующей вершины, т. е. обрабатывает матрицу A по столбцам (алгоритм, ориентированный по столбцам). Модификацией алгоритма Уоршелла для обработки матрицы A по строкам является алгоритм Уоррена (Warren) (алгоритм 6.16). Этот алгоритм является двухпроходным строкоориентированным алгоритмом. В нем при обработке вершины, например i, в первом проходе обрабатываются только дуги, связанные с вершинами, меньшими i, во втором проходе – только дуги, связанные с вершинами, большими i. Иными словами, алгоритм преобразует матрицу смежности A графа G в матрицу смежности A^* графа G^*, обрабатывая в первом проходе только элементы матрицы, расположенные ниже ее главной диагонали, а во втором проходе – только элементы матрицы, расположенные выше ее главной диагонали. Таким образом, при каждом проходе обрабатывается не более $|V|(|V|-1)/2$ дуг.

```
// 1-й проход
for i := 2 to |V| do
   for j := 1 to i − 1 do
      if a_ij = 1 then for k := 1 to |V| do a_ik := a_ik ∨ a_jk
// 2-й проход
for i := 1 to |V| − 1 do
   for j := i + 1 to |V| do
      if a_ij = 1 then for k := 1 to |V| do a_ik := a_ik ∨ a_jk
```

Алгоритм 6.16. Алгоритм Уоррена вычисления транзитивного замыкания

Очевидно, что алгоритм Уоррена требует $O(|V|^3)$ операций, т. е. имеет такую же сложность, что и алгоритм Уоршелла. Однако алгоритм Уоррена будет выполняться быстрее алгоритма Уоршелла для больших разреженных орграфов, особенно при страничной организации памяти. Ясно, что для существенного повышения эффективности алгоритм Уоррена можно модифицировать, используя тот же способ, что и для алгоритма Уоршелла при получении алгоритма 6.15, т. е. заменив самые внутренние циклы **for** операциями логического сложения двух строк матрицы.