Тема 6. Алгоритмы на графах

6.8. Фундаментальное множество циклов

Пусть $T = (V, E_T)$ – остовное дерево неориентированного графа $G = (V, E)$. Если к остовному дереву T добавить произвольное ребро $e \in E - E_T$, то полученный подграф $(V, E_T \cup \{e\})$ содержит точно один цикл. Такой цикл является элементом *фундаментального множества* циклов графа G относительно дерева T. Известно, что каждое остовное дерево графа G включает $|V| - 1$ ребер. Следовательно, в фундаментальном множестве циклов относительно любого остовного дерева графа G имеется $|E| - |V| + 1$ циклов. Важной особенностью любого цикла из фундаментального множества является то, что он содержит только одно ребро из множества $E - E_T$ и более это ребро не присутствует ни в одном другом цикле фундаментального множества.

Полезность отыскания фундаментального множества циклов обусловлена тем, что это множество полностью определяет циклическую структуру графа, т. е. любой цикл в графе может быть представлен комбинацией циклов из фундаментального множества. Для этого используется операция *симметрической разности*, которая для произвольных множеств A и B определяется следующим образом:

$A \oplus B = (A \cup B) - (A \cap B)$.

Пусть $F = \{C_1, C_2, \ldots, C_{|E|-|V|+1}\}$ – фундаментальное множество циклов, где каждый цикл C_i является подмножеством ребер графа G, т. е. $C_i \subseteq E$. Тогда произвольный цикл C графа G можно однозначно представить как симметрическую разность некоторого числа циклов фундаментального множества, т. е. $C = C_{i_1} \oplus C_{i_2} \oplus \ldots \oplus C_{i_t}$. Следует обратить внимание на то, что цикл фундаментального множества нельзя выразить через симметрическую разность других циклов множества. На рис. 6.9 показаны граф (*а*), его остовное дерево (*б*) и фундаментальное множество циклов относительно этого дерева (*в*). Например, цикл (*a, f, b, a*) есть $C_1 \oplus C_2$, а цикл (*b, c, d, g, f, b*) есть $C_2 \oplus C_3 \oplus C_4$. Следует отметить, что не каждая такая симметрическая разность является циклом. Например, $C_2 \oplus C_4$ состоит из двух не связанных между собой циклов.

Очевидным подходом к нахождению фундаментального множества циклов является использование поиска в глубину, который строит остовное дерево (*DFS*-дерево), и каждое обратное ребро порождает цикл относительно этого дерева. Когда поиск в глубину достигает обратного ребра (*v, w*), цикл состоит из ребер дерева, идущих от вершины *w* к вершине *v*, и обратного ребра (*v, w*). Ясно, что для хранения пути от *w* к *v* необходим стек, т. е. стек всегда содержит последовательность вершин из пути, идущего от корня к исследуемой в данный момент вершине. Поэтому, если анализируемое ребро (*v, w*) является обратным ребром остовного дерева, то цикл будет состоять из ребра (*v, w*) и ребер, соединяющих вершины из верхней группы элементов стека, начиная с вершины *v*. При этом верхняя группа элементов не должна удаляться из стека, поскольку одно и то же ребро может входить во многие циклы.

Рассмотренный метод построения фундаментального множества циклов $F = \{C_1, C_2, \ldots, C_{|E|-|V|+1}\}$ неориентированного графа $G = (V, E)$ представлен алгоритмом 6.17. Операции со стеком *S* используют индексацию элементов стека для реализации доступа к верхней группе элементов без удаления их из стека, т. е. операции детализированы до уровня работы с указателем *t* вершины стека.

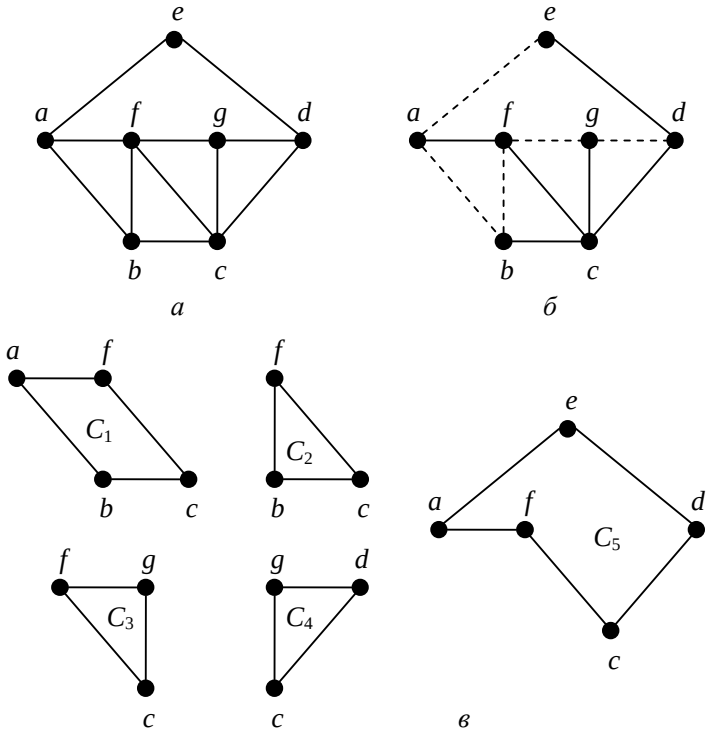
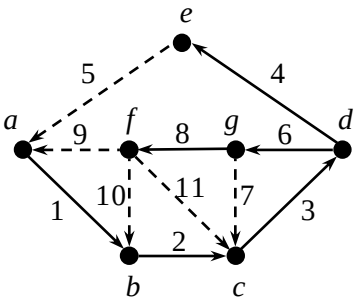


Рис. 6.9. Фундаментальное множество циклов графа:
а – неориентированный граф; *б* – остовное дерево графа;
в – фундаментальные циклы графа

$i \leftarrow j \leftarrow t \leftarrow 0$
// j – номер цикла C_j, t – указатель вершины стека S
$S \leftarrow \varnothing$ // стек S пуст
for $x \in V$ **do** $num(x) \leftarrow 0$ // инициализация
for $x \in V$ **do if** $num(x) = 0$ **then** $CYCLE(x, 0)$

procedure $CYCLE(v, u)$
  $i \leftarrow i + 1$
  $num(v) \leftarrow i$
  $t \leftarrow t + 1$
  $S_t \leftarrow v$ // поместить v в вершину стека S
  for $w \in \mathrm{Adj}(v)$ **do**
    if $num(w) = 0$
      then $\begin{cases} CYCLE(w, v) \\ t \leftarrow t - 1 \quad \text{// исключить элемент из стека } S \end{cases}$
      else if $num(w) < num(v)$ **and** $w \neq u$
        then $\begin{cases} \text{// } (v, w) \text{ – обратное ребро} \\ j \leftarrow j + 1 \\ \text{// сохранить полученный цикл в } C_j \\ C_j \leftarrow (w, S_t, S_{t-1}, \ldots, S_k) \\ \text{// здесь } S_k = w, \text{ а } S_t = v \end{cases}$
return

Алгоритм 6.17. Нахождение фундаментального множества циклов графа



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

Сформированные циклы:
$C_1 = (a, e, d, c, b, a)$; $C_2 = (c, g, d, c)$; $C_3 = (a, f, g, d, c, b, a)$; $C_4 = (b, f, g, d, c, b)$; $C_5 = (c, f, g, d, c)$.

Вычислительная сложность алгоритма, не считая операции сохранения циклов C_j, как и во всех алгоритмах, основанных на поиске в глубину, равна $O(|V| + |E|)$. Дополнительно необходимо учесть следующее. В цикле **for** каждое ребро (v, w) просматривается дважды, но одно и то же ребро входит во многие циклы C_j, т. е. на самом деле просматривается много раз. Таким образом, число операций равно $O(|V| + |E| + l)$, где l – суммарная длина всех порожденных циклов. Величина l зависит от графа и от порядка вершин в списках смежностей. Поскольку каждый цикл имеет длину не более, чем $|V| - 1$, то очевидно, что $l \le (|V| - 1)(|E| - |V| + 1)$, т. е. $l = O(|V||E|)$. Общая сложность алгоритма равна $O(|V||E| + |V| + |E|)$ или $O(|V||E|)$.