Тема 6. Алгоритмы на графах

6.5. Связные компоненты

6.5.1. Связные компоненты неориентированных графов

Неориентированный граф $G = (V, E)$ называется *связным*, если все его вершины связаны между собой, т. е. существует хотя бы один путь в G между каждой парой вершин. Отношение связности определяет разбиение множества V вершин графа на непересекающиеся подмножества $V_i \subseteq V$. Вершины одного и того же множества V_i связаны друг с другом, а вершины различных множеств V_i и V_j не связаны между собой, т. е. в графе G нет ребер, связывающих вершины из разных множеств V_i и V_j. Максимальные связные подграфы $G_i = (V_i, E_i)$ графа G называются *связными компонентами* графа. Связный граф представляет собой единственную связную компоненту. Несвязный граф состоит из двух или более связных компонент.

Для отыскания связных компонент неориентированного графа легко применить технику поиска в глубину. Соответствующая модификация алгоритма поиска в глубину представлена алгоритмом 6.10. Каждой вершине v графа $G = (V, E)$ сопоставлен элемент $compnum(v)$ для присваивания общего номера связной компоненты, в которую попадает вершина v. Таким образом, алгоритм осуществляет разбиение множества V вершин графа на непересекающиеся подмножества, вершины каждого такого подмножества имеют одинаковый номер в элементах $compnum(v)$, соответствующий номеру связной компоненты. Очевидно, что этот алгоритм требует $O(|V| + |E|)$ операций.

for $x \in V$ **do** $compnum(x) \leftarrow 0$

$c \leftarrow 0$ // счетчик компонент

for $x \in V$ **do if** $compnum(x) = 0$ **then** $\begin{cases} c \leftarrow c + 1 \\ COMP(x) \end{cases}$

procedure $COMP(v)$

  $compnum(v) \leftarrow c$

  for $w \in \mathrm{Adj}(v)$ **do if** $compnum(w) = 0$ **then** $COMP(w)$

return

Алгоритм 6.10. Определение связных компонент

6.5.2. Двухсвязные компоненты

Вершина v неориентированного графа $G = (V, E)$ называется *точкой сочленения*, если удаление этой вершины и всех инцидентных ей ребер ведет к разъединению оставшихся вершин, т. е. увеличивает число связных компонент графа. Граф, содержащий точку сочленения, называется *разделимым*. Связный граф без точек сочленения называется *двусвязным*. Максимальный двусвязный подграф графа называется *двусвязной компонентой,* или *блоком* этого графа.

Неориентированный связный разделимый граф и его двусвязные компоненты приведены на рис. 6.7. Точками сочленения являются вершины f и g.

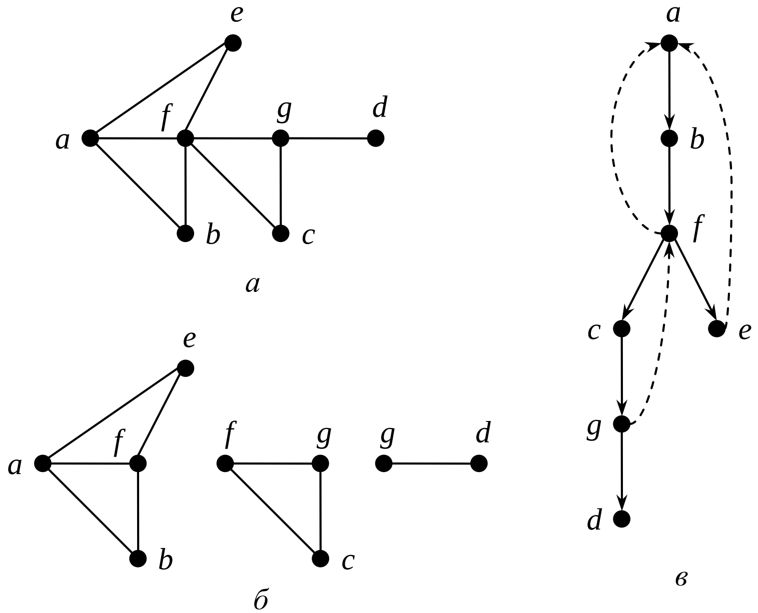


Рис. 6.7. Двусвязные компоненты неориентированного графа:
а – неориентированный граф; *б* – двусвязные компоненты графа;
в – *DFS*-дерево графа в традиционном древовидном представлении

Следует обратить внимание на то, что если $G_1 = (V_1, E_1)$ и $G_2 = (V_2, E_2)$ – два разных блока графа G, то $V_1 \cap V_2 = \varnothing$ или $V_1 \cap V_2 = \{v\}$, где v – точка сочленения графа G. Вершина v неориентированного связного графа является точкой сочленения тогда и только тогда, когда существуют две другие вершины x и y, такие, что любой путь между x и y проходит через вершину v; в этом случае удаление v и всех инцидентных ей ребер из графа G разрывает все пути между x и y, т. е. делает граф G несвязным. Нахождение точек сочленения и блоков графа является классической задачей, которая эффективно решается использованием техники поиска в глубину. Прежде всего необходимо определить критерий для распознавания точек сочленения.

Пусть $T = (V, E_T)$ – *DFS*-дерево (остовное дерево, построенное поиском в глубину) связного неориентированного графа $G = (V, E)$. Тот факт, что удаление точки сочленения v расщепляет граф G по крайней мере на две части, говорит о следующем. Одна из этих частей состоит из сына вершины v и всех его потомков в *DFS*-дереве. Следовательно, в *DFS*-дереве вершина v должна иметь сына w, потомки которого (включая и w) не соединены обратными ребрами с предками вершины v. Очевидно, что корень *DFS*-дерева является точкой сочленения только тогда, когда он имеет не менее двух сыновей. Для неориентированного графа (рис. 6.7, *а*) *DFS*-дерево в традиционном древовидном представлении показано на рис. 6.7, *в*. Точками сочленения являются вершины f и g. Вершина f имеет сына c, и ни из какого потомка вершины c не выходит обратное ребро к предкам вершины f. Аналогично вершина g имеет сына d, потомки которого не связаны обратными ребрами с предками вершины g.

Таким образом, можно сформулировать следующий критерий для распознавания точек сочленения: пусть $T = (V, E_T)$ – DFS-дерево с корнем r связного неориентированного графа $G = (V, E)$. Вершина $v \in V$ является точкой сочленения графа G тогда и только тогда, когда выполнено одно из условий:

а) $v = r$ и r имеет более одного сына;

б) $v \neq r$ и существует сын w вершины v, такой, что ни w, ни какой-либо его потомок не связаны обратным ребром ни с одним предком вершины v.

Чтобы включить в процедуру поиска в глубину сформулированный критерий, для каждой вершины $v \in V$ необходимо определить два параметра: $num(v)$ и $low(v)$. Первый параметр – это номер вершины v в порядке, в котором вершины посещаются при поиске в глубину. Если E_T и E_B – множества соответственно ребер DFS-дерева и обратных ребер, то

$$low(v) = \min(\{num(v)\} \cup \{num(w) \mid \text{существует такое обратное ребро } (x, w) \in E_B, \text{ что } x \text{ – потомок } v, \text{ а } w \text{ – предок } v \text{ в } DFS\text{-дереве}\}),$$

т. е. $low(v)$ есть наименьшее значение $num(x)$, где x – вершина графа, в которую можно попасть из v, проходя последовательность из нуля или более ребер дерева, за которой следует не более чем одно обратное ребро. Нумерация вершин в порядке прохождения в глубину обладает тем свойством, что если x – потомок вершины v, а (x, w) – обратное ребро, причем $num(w) < num(v)$, то w – предок вершины v. Следовательно, если v не корень, то v является точкой сочленения тогда и только тогда, когда имеет сына u, для которого $low(u) \geq num(v)$.

Переопределим $low(v)$ так, чтобы выразить через вершины, смежные с v, используя обратные ребра и значения $num(w)$ на сыновьях вершины v:

$$low(v)=\min(\{num(v)\}\cup\{low(w) \mid (v,w)\in E_T\}\cup\{num(w) \mid (v,w)\in E_B\}).$$

Тогда вычисление значения $low(v)$ предполагает следующие шаги:

1. Когда v проходится в первый раз, то $low(v)$ присваивается значение $num(v)$.

2. Когда рассматривается обратное ребро (v, w), инцидентное v, то $low(v)$ присваивается наименьшее из текущего значения $low(v)$ и $num(w)$.

3. Когда поиск в глубину возвращается к v после полного сканирования сына w этой вершины, то $low(v)$ присваивается наименьшее из текущего значения $low(v)$ и значения $low(w)$.

Необходимо обратить внимание на то, что для любой вершины v вычисление $low(v)$ заканчивается при завершении ее сканирования.

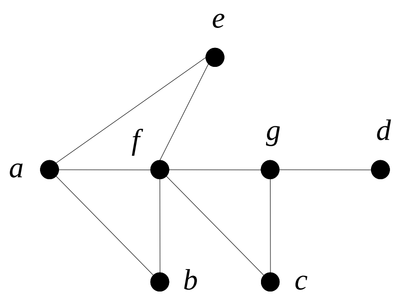
Процедура определения двусвязных компонент E_j ($j \geq 1$) неориентированного графа $G = (V, E)$ представлена алгоритмом 6.11. Для выделения ребер, принадлежащих двусвязным компонентам, используется стек S. Вначале стек пуст. По мере просмотра ребер они добавляются в стек. Пусть поиск в глубину возвращается в вершину v после полного сканирования сына w этой вершины. В этот момент завершается вычисление $low\,(w)$. Предположим, что $low\,(w) \geq num\,(v)$. Тогда v – точка сочленения. Ребро (v, w) вместе с ребрами, инцидентными w и его потомкам, образуют двусвязную компоненту. Эти ребра являются в точности теми ребрами, которые находятся в верхней части стека S, включая ребро (v, w). Исключение этих ребер из стека приводит к тому, что алгоритм продолжает работать с графом G', который получается из графа G удалением ребер уже выделенной двусвязной компоненты.

```
for x ∈ V do num(x) ← 0
i ← j ← 0   // j – номер двусвязной компоненты E_j
S ← ∅   // стек S пуст
for x ∈ V do if num(x) = 0 then BICOMP(x, 0)

procedure BICOMP(v, u)
   i ← i + 1
   num(v) ← low(v) ← i
   for w ∈ Adj(v) do
      if num(w) = 0
              ⎧ // (v, w) – ребро дерева
              ⎪ S ⇐ (v, w)
              ⎪ BICOMP(w, v)
              ⎪ low(v) ← min(low(v), low(w))
              ⎪ if low(w) ≥ num(v)
              ⎪          ⎧ // v – корень или точка сочленения.
              ⎪          ⎪ // Верхняя часть стека S до (v, w)
        then ⎨          ⎪ // включительно содержит
              ⎪          ⎪ // двусвязную компоненту
              ⎪          ⎪
              ⎪     then ⎨ (x, y) ⇐ S
              ⎪          ⎪ j ← j + 1
              ⎪          ⎪ E_j ← {(x, y)}
              ⎪          ⎪ // выделить из стека компоненту E_j
              ⎪          ⎪ while (x, y) ≠ (v, w) do ⎧ (x, y) ⇐ S
              ⎩          ⎩                          ⎩ E_j ← E_j ∪ {(x, y)}
        else if num(w) < num(v) and w ≠ u
                  ⎧ // (v, w) – обратное ребро
             then ⎨ S ⇐ (v, w)
                  ⎩ low(v) ← min(low(v), num(w))
return
```

Алгоритм 6.11. Определение двусвязных компонент графа

Для неориентированного графа (см. рис. 6.7, *a*), алгоритм выделяет двусвязные компоненты в таком порядке: $E_1 = \{(g, d)\}$, $E_2 = \{(f, c), (c, g), (g, f)\}$ и $E_3 = \{(a, b), (b, f), (f, a), (f, e), (e, a)\}$. При этом получаются следующие значения элементов $low(v)$:



v	a	b	c	d	e	f	g
$num(v)$	1	2	4	6	7	3	5
$low(v)$	1	1	3	6	1	1	3

Определим вычислительную сложность алгоритма. Поскольку он осуществляет поиск в глубину с конечным объемом дополнительной работы при прохождении каждого ребра, требуемое время выполнения равно $O(|V| + |E|)$.