Тема 6. Алгоритмы на графах

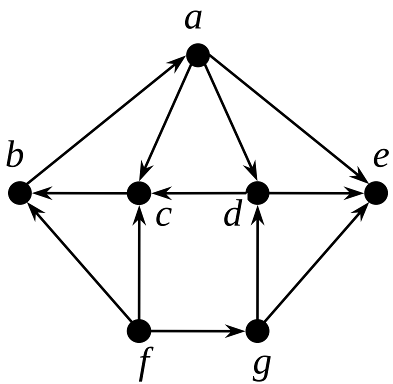
6.5. Связные компоненты

6.5.3. Сильно связные компоненты

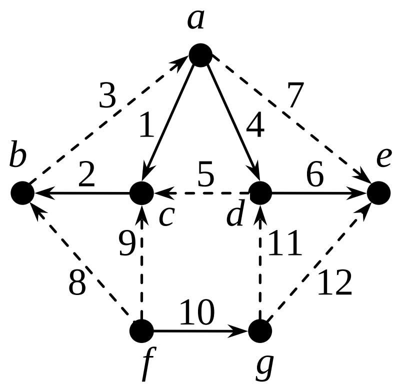
Понятие сильной связности определено для ориентированных графов. Орграф $G = (V, E)$ называется *сильно связным*, если для любых двух его вершин v и w существуют ориентированные пути из v к w и из w к v. Даже если орграф G не сильно связный, он может содержать сильно связные подграфы. Максимальные сильно связные подграфы орграфа G называются *сильно связными компонентами* этого графа, т. е. сильно связный орграф имеет только одну сильно связную компоненту.

Поиск в глубину позволяет находить сильно связные компоненты орграфа. Напомним, что процесс поиска в глубину разбивает ребра орграфа $G = (V, E)$ на четыре множества: E_T – множество ребер DFS-дерева (леса), E_B – множество обратных ребер, E_F – множество прямых ребер и E_C – множество поперечных ребер. При этом номера вершин пройденного орграфа представляют собой значения элементов $num(v)$ для всех $v \in V$. Разбиение ребер можно использовать для определения сильно связных компонент. Очевидно, что прямые ребра можно не рассматривать, так как они не влияют на сильную связность. Исходящие из вершины v обратные и поперечные ребра могут идти только в такие вершины w, для которых $num(v) > num(w)$.

Пусть $G_i = (V_i, E_i)$ – сильно связная компонента орграфа $G = (V, E)$, а $T = (V, E_T)$ – DFS-лес для G. Пусть $v, w \in V_i$, т. е. принадлежат одной и той же сильно связной компоненте G_i. Без потери общности будем считать, что $num(v) < num(w)$. По определению в G_i существует путь P из вершины v в вершину w. Пусть x – вершина на P с наименьшим номером (возможно, это сама вершина v). Путь P, дойдя до какого-нибудь потомка вершины x, уже не сможет выйти за пределы поддерева потомков вершины x, поскольку за пределы этого поддерева могут выходить лишь поперечные и обратные ребра, идущие в вершины с номерами, меньшими x. Следовательно, вершина w – потомок вершины x. Все вершины, номера которых заключены между $num(x)$ и $num(w)$, также являются потомками вершины x. Так как $num(x) \le num(v) < num(w)$, то вершина v – потомок вершины x. Таким образом, любые две вершины в G_i имеют общего предка в G_i. Следовательно, если G_i – сильно связная компонента орграфа G, то вершины G_i определяют дерево, которое является подграфом DFS-леса. Обратное утверждение неверно: не каждое поддерево DFS-леса соответствует сильно связной компоненте.



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



а　　　　*б*

Пример $P = c(2) \to b(3) \to a(1) \to d(4)$, в скобках значения num, $v = c$, $x = a$, $w = d$.

Сильно связные компоненты орграфа можно найти, определив корни поддеревьев, соответствующих этим компонентам. Для этого определим функцию

$lowlink(v) = \min(\{num(v)\} \cup \{num(w) \mid$ существует поперечное или обратное ребро из потомка вершины v в вершину w и w находится в той же самой сильно связной компоненте, что и $v\})$.

Другими словами, $lowlink(v)$ есть наименьший номер среди номеров вершин в той сильно связной компоненте, в которой находится вершина v; эти вершины можно достичь по пути из нуля или большего числа ребер, за которыми следует не больше одного поперечного или обратного ребра. В этом случае вершина v будет корнем сильно связной компоненты тогда и только тогда, когда $lowlink(v) = num(v)$.

Вычисление значения $lowlink(v)$ в процессе поиска в глубину предполагает следующие действия:

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

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

3. Когда рассматривается поперечное ребро (v, w), для которого v и w принадлежат одной и той же сильно связной компоненте, то $lowlink(v)$ присваивается наименьшее из его текущего значения $lowlink(v)$ и $num(w)$.

4. Когда осуществляется возврат в вершину v после полного сканирования сына w вершины v, то $lowlink(v)$ есть минимум из его текущего значения $lowlink(v)$ и значения $lowlink(w)$.

Процедура определения сильно связных компонент V_j ($j \geq 1$) орграфа $G = (V, E)$ представлена алгоритмом 6.12. Стек S используется для хранения вершин в порядке прохождения в глубину. Вершина w находится в стеке S тогда и только тогда, когда она принадлежит той же сильно связной компоненте, что и предок вершины v. Поэтому в момент, когда $lowlink(v) = num(v)$, вершины сверху стека S до вершины v включительно образуют сильно связную компоненту и исключаются из стека. В операторе «**if** $w \in S$» определяется наличие вершины w в стеке S. Для реализации такой проверки достаточно использовать булев массив, элементы которого сопоставлены вершинам графа G. В случае, если $w \in S$, то w находится в той же сильно связной компоненте, что и v, поскольку $w \in \mathrm{Adj}(v)$ и $w \in S$, т. е. существует путь из вершины w в вершину v.

```
for x ∈ V do num(x) ← 0
i ← j ← 0   //j – номер сильно связной компоненты V_j
S ← ∅   // стек S пуст
for x ∈ V do if num(x) = 0 then STRONG(x)

procedure STRONG(v)
   i ← i + 1
   num(v) ← lowlink(v) ← i
   S ⇐ v
   for w ∈ Adj(v) do
      if num(w) = 0
         then { // (v, w) – ребро дерева
              { STRONG(w)
              { lowlink(v) ← min(lowlink(v), lowlink(w))
         else  if num(w) < num(v)
                  then { // (v, w) – обратное/ поперечное ребро
                       { if w ∈ S
                       {    then lowlink(v) ← min(lowlink(v), num(w))
   if lowlink(v) = num(v)
      then { // v – корень сильно связной компоненты
           { j ← j + 1
           { V_j ← ∅
           { // выделить из стека компоненту V_j
           { // top(S) – верхний элемент стека S
           { while num(top(S)) ≥ num(v) do { x ⇐ S
           {                               { V_j ← V_j ∪ {x}
return
```

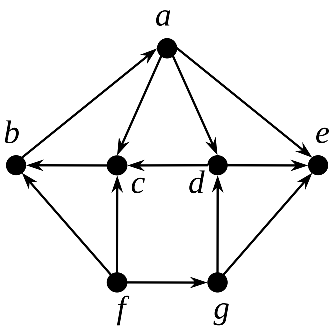
Алгоритм 6.12. Определение сильно связных компонент орграфа

Для орграфа, изображенного на рис. 6.6, *а*, алгоритм выделяет сильно связные компоненты в таком порядке: $V_1 = \{e\}$, $V_2 = \{a, b, c, d\}$, $V_3 = \{g\}$, $V_4 = \{f\}$. При этом получаются следующие значения элементов $num(v)$ и $lowlink(v)$:

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

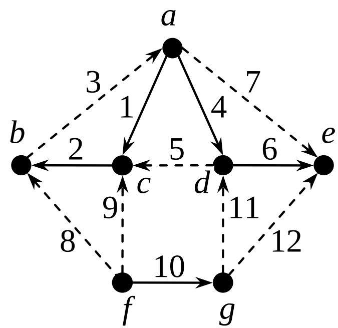
.

Алгоритм осуществляет поиск в глубину, в ходе реализации которого требуются некоторые дополнительные операции при прохождении каждого ребра графа. Каждая вершина включается в стек и исключается из него в точности один раз. Поэтому временная сложность алгоритма равна $O(|V| + |E|)$.



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

а



б