Тема 6. Алгоритмы на графах

6.4. Остовные деревья

6.4.4. Остовные деревья ориентированных графов

Очевидно, остовное дерево (лес) орграфа является ориентированным. Рассмотренные выше алгоритмы построения остовных деревьев неориентированных графов могут находить ориентированные остовные деревья и для орграфов. Главное отличие заключается только в том, что ребра (дуги) орграфа проходятся в соответствии с их ориентацией.

Задача усложняется, если необходимо не только построить остовное дерево, но и разбить множество ребер орграфа на классы. Если поиск в глубину в неориентированном графе разбивает его ребра на два класса (ребра остовного дерева и обратные ребра), то в орграфе ребра (дуги) разбиваются на четыре класса. Непросмотренное ребро (v, w), встречающееся, когда процесс поиска в глубину находится в вершине v, можно классифицировать следующим образом:

1. Вершина w еще не пройдена. В этом случае (v, w) – *ребро дерева*, идущее к новой вершине.

2. Вершина w уже была пройдена:

а) если w потомок v в *DFS*-дереве (а не в орграфе), то ребро (v, w) называется *прямым ребром*, т. е. прямые ребра идут от предков к потомкам, но не являются ребрами дерева;

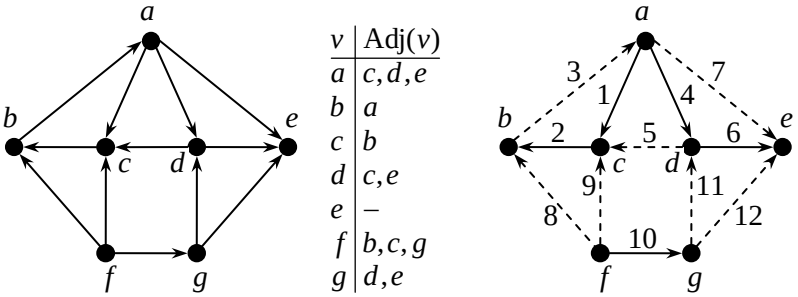
б) если w предок v в *DFS*-дереве, то ребро (v, w) называется *обратным ребром*, т. е. обратные ребра идут от потомков к предкам;

в) если v и w не являются ни предками, ни потомками друг друга (несоотносимы) в *DFS*-дереве, то ребро (v, w) называется *поперечным ребром*.

Как и в случае построения *DFS*-дерева для неориентированного графа, чтобы отличать разные классы ребер, необходимо использовать нумерацию исследуемых вершин с помощью элементов $num(v)$. Ребро (v, w) при $num(w) > num(v)$ является либо ребром дерева, либо прямым ребром. В процессе поиска в глубину различить эти ребра просто – ребро дерева всегда ведет к новой вершине. Ребро (v, w) при $num(w) < num(v)$ является либо обратным, либо поперечным ребром. Критерий их отличия основан на следующем свойстве поиска в глубину: если к моменту рассмотрения ребра (v, w) $num(w) < num(v)$, но еще не все исходящие из w ребра исследованы (просканированы), то это означает, что w является корнем поддерева, которое содержит вершину v, т. е. v – потомок w. Таким образом, ребро (v, w) при $num(w) < num(v)$ является обратным ребром, если вершина w еще не полностью просканирована к моменту анализа ребра (v, w), в противном случае – поперечным ребром.

Необходимо отметить еще одну особенность *DFS*-дерева для ориентированных графов. Она заключается в том, что в общем случае, даже если орграф связен, его *DFS*-дерево может быть несвязным, т. е. *DFS*-лесом.

Орграф, заданный структурой смежности, и построенный для него *DFS*-лес представлены на рис. 6.6. Ребра дерева изображены сплошными линиями, а прочие ребра – штриховыми. Числа около ребер показывают порядок исследования ребер по заданной структуре смежности. Как видно из рисунка, для связного орграфа полученный *DFS*-лес несвязный и состоит из двух деревьев: дерева с вершинами a, b, c, d, e (корень a) и дерева с вершинами f и g (корень f). Ребро (b, a) обратное, ребро (a, e) прямое, ребра (d, c), (f, b), (f, c), (g, d) и (g, e) – поперечные.



а *б*

Рис. 6.6. Остовный лес орграфа:
а – орграф и его структура смежности; *б* – *DFS*-лес орграфа;
Ребро (*b*, *a*) обратное, ребро (*a*, *e*) прямое, ребра (*d*, *c*), (*f*, *b*), (*f*, *c*), (*g*, *d*) и (*g*, *e*) – поперечные

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

Процесс построения *DFS*-дерева (леса) $T = (V, E_T)$ для ориентированного графа $G = (V, E)$ с разбиением ребер графа на четыре класса представлен алгоритмом 6.9. В отличие от алгоритма 6.3 построения *DFS*-дерева для неориентированного графа, в него добавлены элементы *scan* (*v*) для всех $v \in V$, чтобы отмечать, полностью просканирована вершина *v* или нет. Элементами множеств E_F и E_C являются соответственно прямые и поперечные ребра. Очевидно, что временная сложность алгоритма равна $O(|V| + |E|)$.

```
for x ∈ V do { num(x) ← 0
             { scan(x) ← true
i ← 0
E_T ← E_B ← E_F ← E_C ← ∅
for x ∈ V do if num(x) = 0 then DFSTO(x)

procedure DFSTO(v)
   i ← i + 1
   num(v) ← i
   for w ∈ Adj(v) do
      if num(w) = 0
         then { // (v, w) – ребро дерева
              { E_T ← E_T ∪ {(v, w)}
              { DFSTO(w)
         else if num(w) > num(v)
                 then { // (v, w) – прямое ребро
                      { E_F ← E_F ∪ {(v, w)}
                 else if num(w) < num(v) and scan(w)
                         then { // (v, w) – обратное ребро
                              { E_B ← E_B ∪ {(v, w)}
                         else { // (v, w) – поперечное ребро
                              { E_C ← E_C ∪ {(v, w)}
   scan(v) ← false
return
```

Алгоритм 6.9. Построение *DFS*-дерева для орграфа