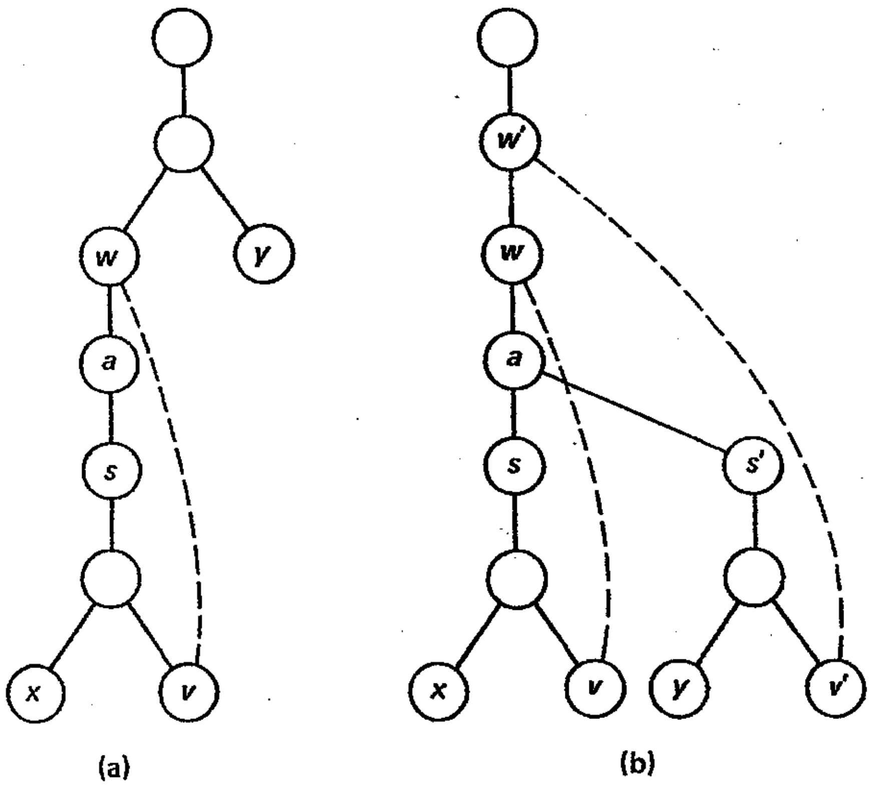

Fig. 5.10 Counterexample paths.

```
procedure SEARCHB(v):
begin
    mark v "old";
    DFNUMBER[v] ← COUNT;
    COUNT ← COUNT + 1;
    LOW[v] ← DFNUMBER[v];
    for each vertex w on L[v] do
        if w is marked "new" then
            begin
                add (v, w) to T;
                FATHER[w] ← v;
                SEARCHB(w);
                if LOW[w] ≥ DFNUMBER[v] then a biconnected
                        component has been found;
                LOW[v] ← MIN(LOW[v], LOW[w])
            end
        else if w is not FATHER[v] then
            LOW[v] ← MIN(LOW[v], DFNUMBER[w])
end
```

Fig. 5.11. Depth-first search with LOW computation.

```
        begin
1.          S ← {v₀};
2.          D[v₀] ← 0;
3.          for each v in V − {v₀} do D[v] ← l(v₀, v);
4.          while S ≠ V do
                begin
5.                  choose a vertex w in V − S such that D[w] is a minimum;
6.                  add w to S;
7.                  for each v in V − S do
8.                      D[v] ← MIN(D[v], D[w] + l(w, v))
                end
        end
```

Fig. 5.24. Dijkstra's algorithm.

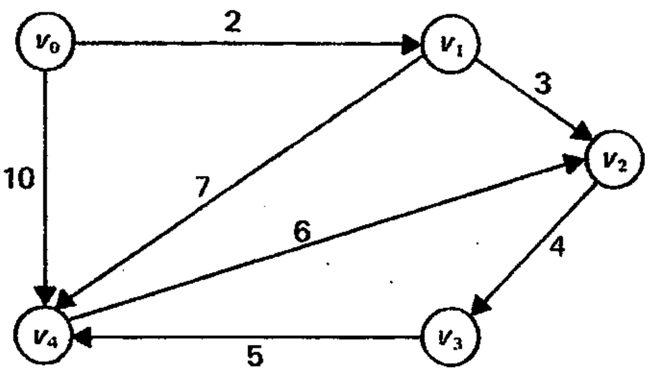

Fig. 5.25 A graph with labeled edges.

Iteration	S	w	$D[w]$	$D[v_1]$	$D[v_2]$	$D[v_3]$	$D[v_4]$
Initial	$\{v_0\}$	—	—	2	$+\infty$	$+\infty$	10
1	$\{v_0, v_1\}$	v_1	2	2	5	$+\infty$	9
2	$\{v_0, v_1, v_2\}$	v_2	5	2	5	9	9
3	$\{v_0, v_1, v_2, v_3\}$	v_3	9	2	5	9	9
4	All	v_4	9	2	5	9	9

Fig. 5.26. Computation of Algorithm 5.6 on graph of Fig. 5.25.

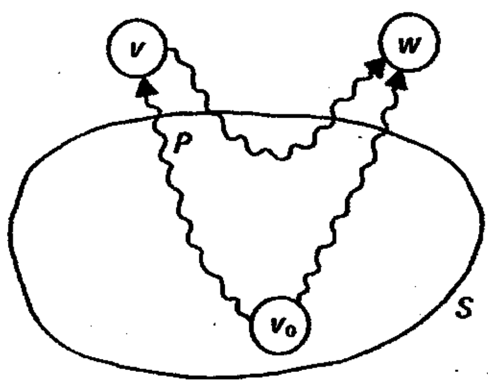

Fig. 5.27 Paths to vertex v.