Algorithm computing the blocks of a graph

**Input:** connected, undirected graph \( G = (V, E) \)

**Output:** set \( A \) of cut nodes, set \( B \) of blocks

```
procedure bicon(node v, node pv)
begin // v is current node, pv its parent
  visited[v] := true;
  int c := 0; // counts children of v in DFS-tree
  for all edges e = \{v, w\} of v do
    if visited[w] = false then // tree edge
      S.push(e); c++; bicon(w, v);
      low[v] = min(low[v], low[w]);
    if low[w] ≥ pre[v] and (v ≠ s or c = 2) then
      A := A ∪ \{v\}; fi; // v is cut node
    if low[w] ≥ pre[v] then C := ∅;
      repeat f := S.pop(); C := C ∪ \{f\};
      until f = e;
      B := B ∪ \{C\}; fi; // C is the next block
  else if pre[w] < pre[v] and w ≠ pv then //backward edge
    S.push(e); low[v] = min(low[v], pre[w]); fi
  od; end;
```

// main program

```
visited[v] := false for all v ∈ V;
s := starting node; current := 1; A := B := ∅; S := empty stack;
bicon(s, s);
```