Figure 3.3 Finding all nodes reachable from a particular node.

procedure explore\( (G, v) \)

Input: \( G = (V, E) \) is a graph; \( v \in V \)

Output: visited\( (u) \) is set to true for all nodes \( u \) reachable from \( v \)

visited\( (v) = \text{true} \)
previsit\( (v) \)
for each edge \( (v, u) \in E \):
  if not visited\( (u) \):
    explore\( (u) \)
postvisit\( (v) \)
**Figure 3.5** Depth-first search.

procedure \texttt{dfs}(G)

for all \( v \in V \):
    \( \text{visited}(v) = \text{false} \)

for all \( v \in V \):
    if not \( \text{visited}(v) \): \( \text{explore}(v) \)

**Figure 3.6** (a) A 12-node graph. (b) DFS search forest.

---

```
procedure \texttt{previsit}(v) \qquad procedure \texttt{postvisit}(v)
\begin{align*}
\text{pre}[v] &= \text{clock} \\
\text{clock} &= \text{clock} + 1 \\
\text{post}[v] &= \text{clock} \\
\text{clock} &= \text{clock} + 1
\end{align*}
```

**Property** For any nodes \( u \) and \( v \), the two intervals \( [\text{pre}(u), \text{post}(u)] \) and \( [\text{pre}(v), \text{post}(v)] \) are either disjoint or one is contained within the other.

**Why?** Because \( [\text{pre}(u), \text{post}(u)] \) is essentially the time during which vertex \( u \) was on the stack. The last-in, first-out behavior of a stack explains the rest.
**Figure 3.5** Depth-first search.

```plaintext
procedure dfs(G)
    for all \( v \in V \):
        visited(v) = false
    for all \( v \in V \):
        if not visited(v): explore(v)
```

**Figure 3.7** DFS on a directed graph.

Fig. 3.7 has two forward edges, two back edges, and two cross edges. Because of DFS exploration strategy, vertex \( u \) is an **ancestor** of vertex \( v \) when \( u \) is discovered first and \( v \) is discovered during `explore(u)`, i.e., \( \text{pre}(u) < \text{pre}(v) < \text{post}(v) < \text{post}(u) \).