CSCI 241

Scott Wehrwein

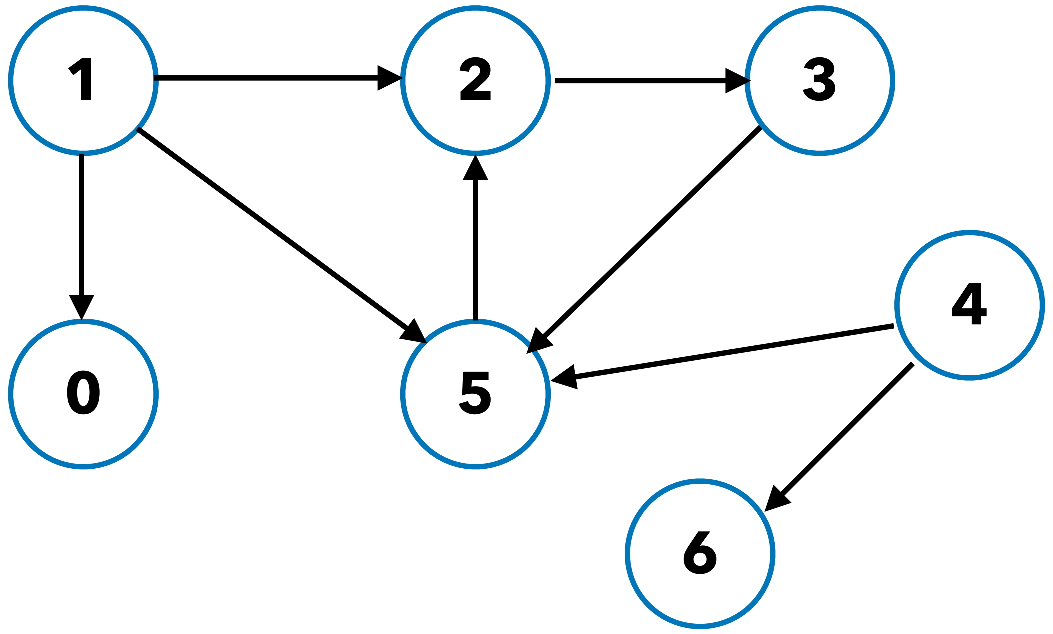
Graph Traversals:
Depth-First Search (Iteratively)

Goals

Be able to implement DFS iteratively using a stack.

Depth-first Search: Iteratively

```
/** Visit all nodes explorable from u.
  * Pre: u is unvisited. */
public static void dfs(int nodeID) {
  Stack s = (nodeID); // Not Java!
  // inv: all nodes to be visited are
  //      explorable from some node in s
  while (s is not empty) {
    u = s.pop();
    if (u has not been visited) {
      visit u;
      for each edge (u, v) from u:
        s.push(v);
    }
  }
}
```

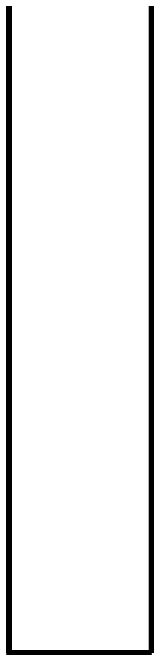


$$\begin{tikzcd} 1 \arrow[r] \arrow[d] \arrow[dr] & 2 \arrow[r] & 3 \arrow[dl] & \\ 0 & 5 \arrow[u] & & 4 \arrow[ll] \arrow[dl] \\ & & 6 & \end{tikzcd}$$

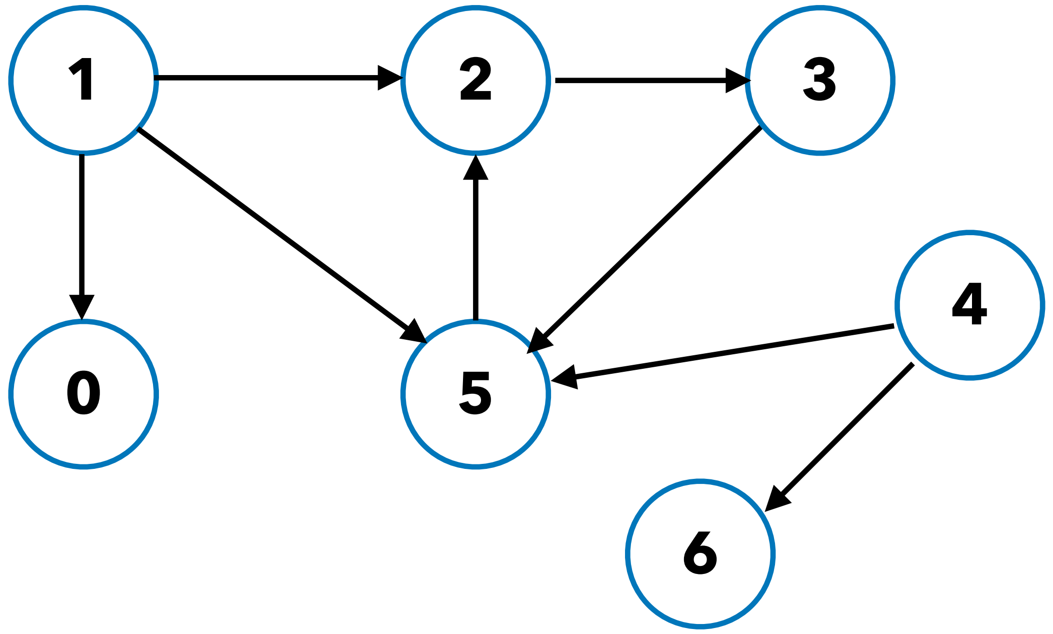
Depth-first Search: Iteratively

```
/** Visit all nodes explorable from u.
  * Pre: u is unvisited. */
public static void dfs(int nodeID) {
  Stack s = (nodeID); // Not Java!
  // inv: all nodes to be visited are
  //      explorable from some node in s
  while (s is not empty) {
    u = s.pop();
    if (u has not been visited) {
      visit u;
      for each edge (u, v) from u:
        s.push(v);
    }
  }
}
```

Stack s:



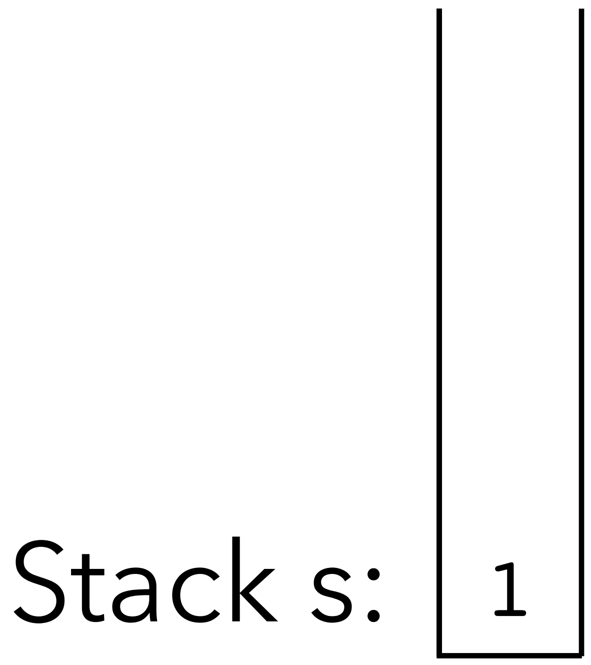
```
dfs(1)
```

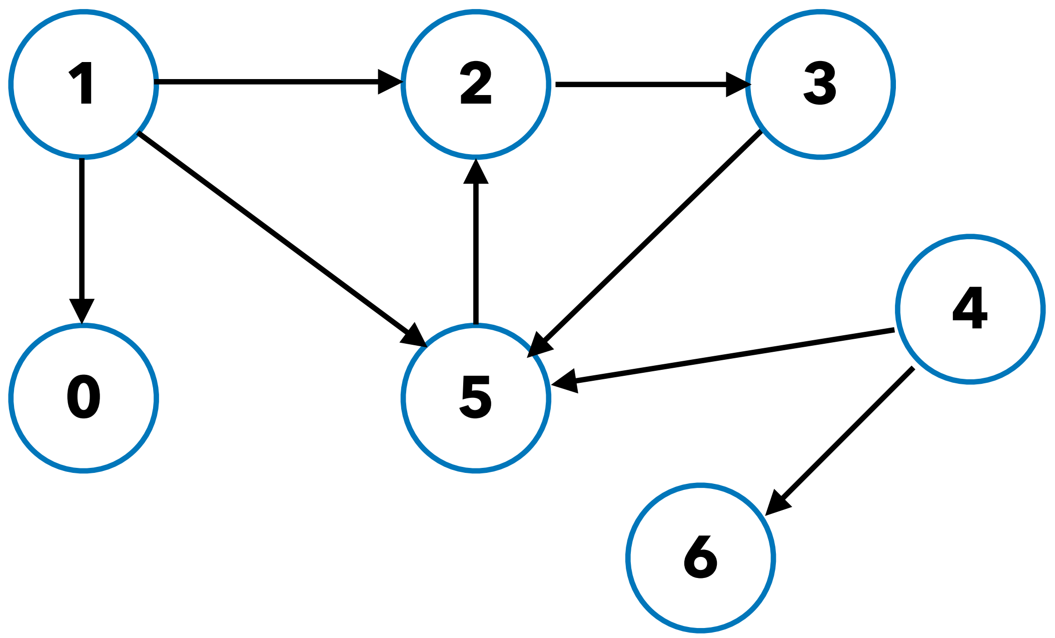


Depth-first Search: Iteratively

```
/** Visit all nodes explorable from u.
  * Pre: u is unvisited. */
public static void dfs(int nodeID) {
→ Stack s = (nodeID); // Not Java!
  // inv: all nodes to be visited are
  //      explorable from some node in s
  while (s is not empty) {
    u = s.pop();
    if (u has not been visited) {
      visit u;
      for each edge (u, v) from u:
        s.push(v);
    }
  }
}
```

dfs(1)

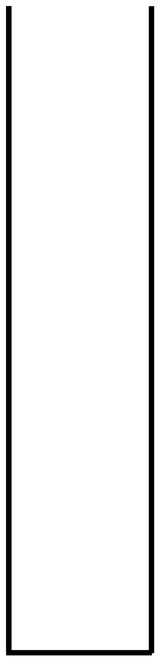




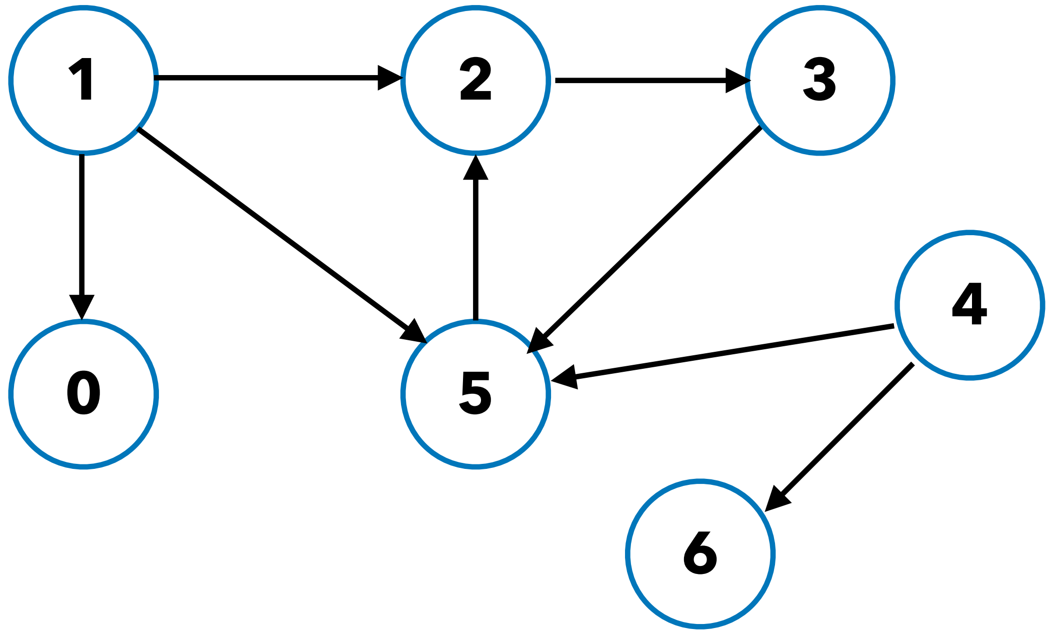
Depth-first Search: Iteratively

```
/** Visit all nodes explorable from u.
  * Pre: u is unvisited. */
public static void dfs(int nodeID) {
  Stack s = (nodeID); // Not Java!
  // inv: all nodes to be visited are
  //      explorable from some node in s
  while (s is not empty) {
→   u = s.pop();
    if (u has not been visited) {
      visit u;
      for each edge (u, v) from u:
        s.push(v);
    }
  }
}
```

Stack s:



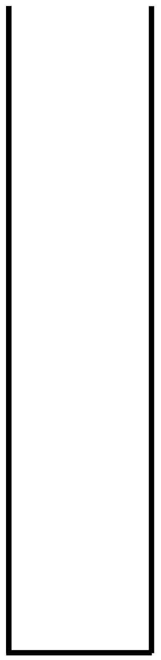
dfs(1)



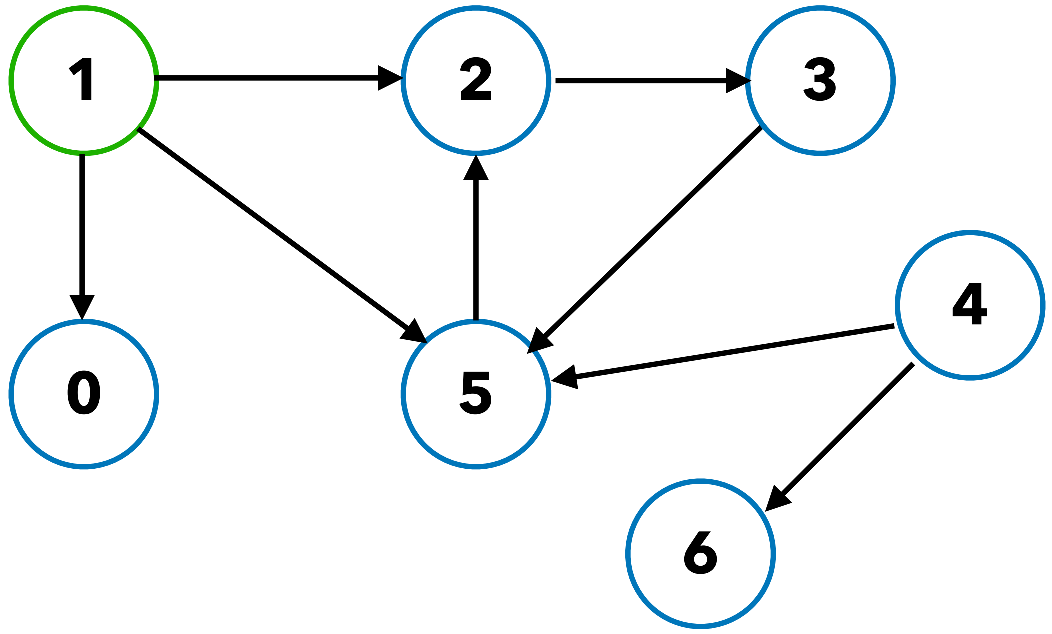
Depth-first Search: Iteratively

```
/** Visit all nodes explorable from u.
  * Pre: u is unvisited. */
public static void dfs(int nodeID) {
  Stack s = (nodeID); // Not Java!
  // inv: all nodes to be visited are
  //      explorable from some node in s
  while (s is not empty) {
    u = s.pop();
    if (u has not been visited) {
→     visit u;
      for each edge (u, v) from u:
        s.push(v);
    }
  }
}
```

Stack s:



dfs(1)



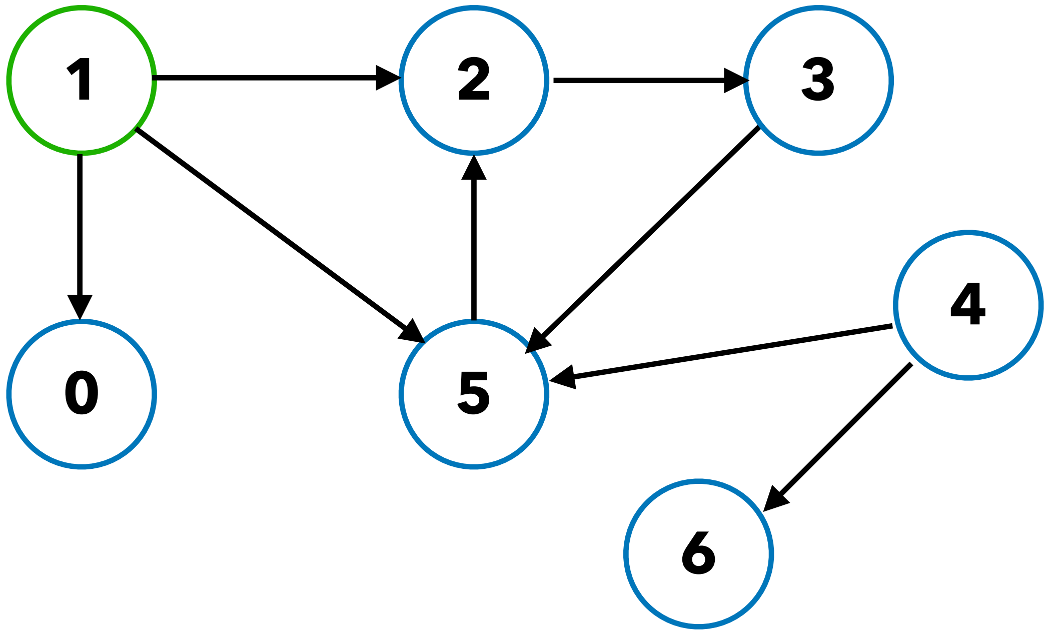
Depth-first Search: Iteratively

```
/** Visit all nodes explorable from u.
  * Pre: u is unvisited. */
public static void dfs(int nodeID) {
  Stack s = (nodeID); // Not Java!
  // inv: all nodes to be visited are
  //      explorable from some node in s
  while (s is not empty) {
    u = s.pop();
    if (u has not been visited) {
      visit u;
→     for each edge (u, v) from u:
        s.push(v);
    }
  }
}
```

dfs(1)

Stack s:

0
2
5



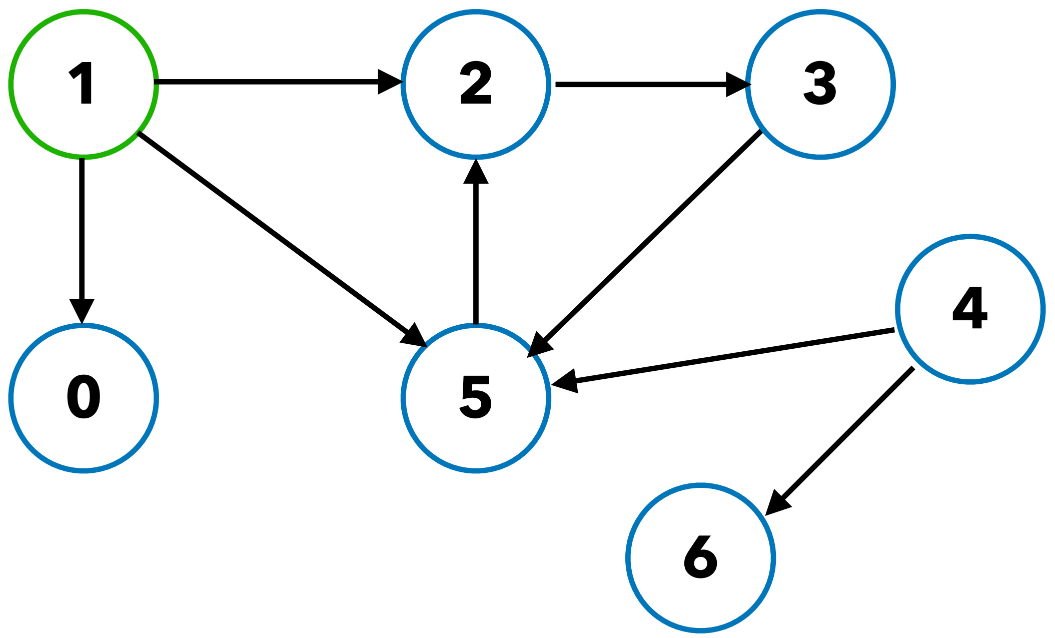
Depth-first Search: Iteratively

```
/** Visit all nodes explorable from u.
  * Pre: u is unvisited. */
public static void dfs(int nodeID) {
  Stack s = (nodeID); // Not Java!
  // inv: all nodes to be visited are
  //      explorable from some node in s
  while (s is not empty) {
→   u = s.pop();
    if (u has not been visited) {
      visit u;
      for each edge (u, v) from u:
        s.push(v);
    }
  }
}
```

Stack s:

2
5

dfs(1)



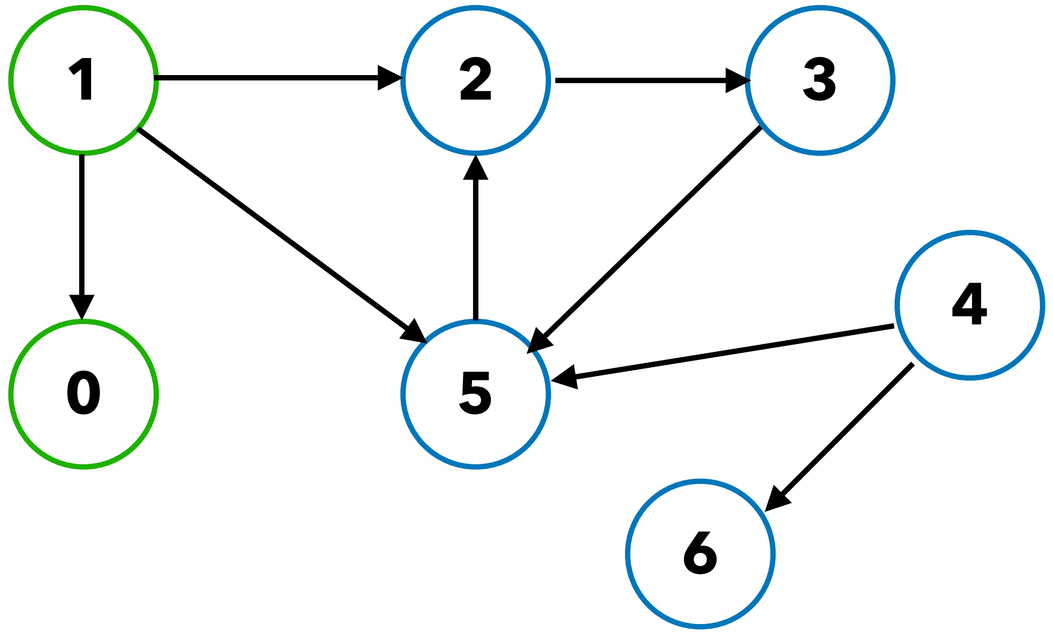
Depth-first Search: Iteratively

```
/** Visit all nodes explorable from u.
  * Pre: u is unvisited. */
public static void dfs(int nodeID) {
  Stack s = (nodeID); // Not Java!
  // inv: all nodes to be visited are
  //      explorable from some node in s
  while (s is not empty) {
    u = s.pop();
    if (u has not been visited) {
→     visit u;
      for each edge (u, v) from u:
        s.push(v);
    }
  }
}
```

Stack s:

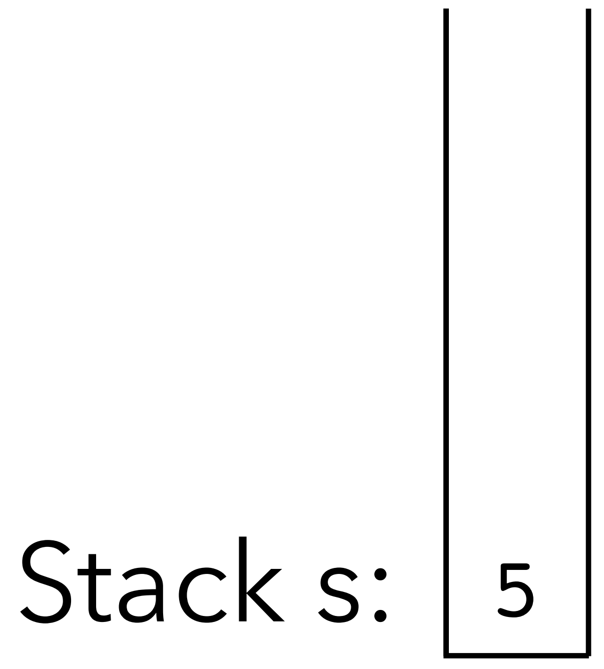
2
5

`dfs(1)`

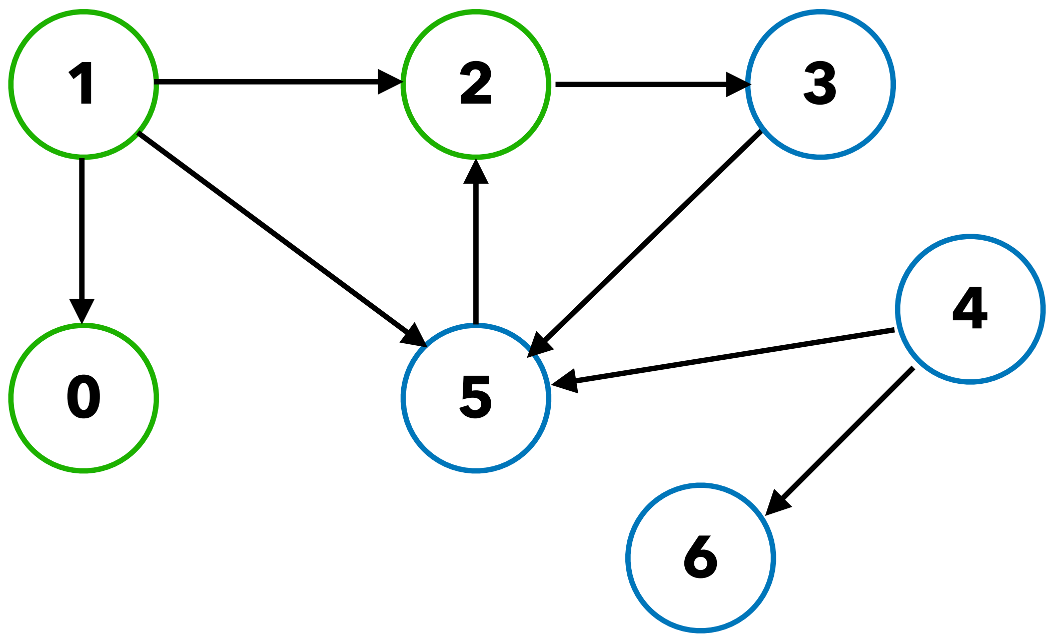


Depth-first Search: Iteratively

```
/** Visit all nodes explorable from u.
  * Pre: u is unvisited. */
public static void dfs(int nodeID) {
  Stack s = (nodeID); // Not Java!
  // inv: all nodes to be visited are
  //      explorable from some node in s
  while (s is not empty) {
    u = s.pop();
    if (u has not been visited) {
→     visit u;
      for each edge (u, v) from u:
        s.push(v);
    }
  }
}
```



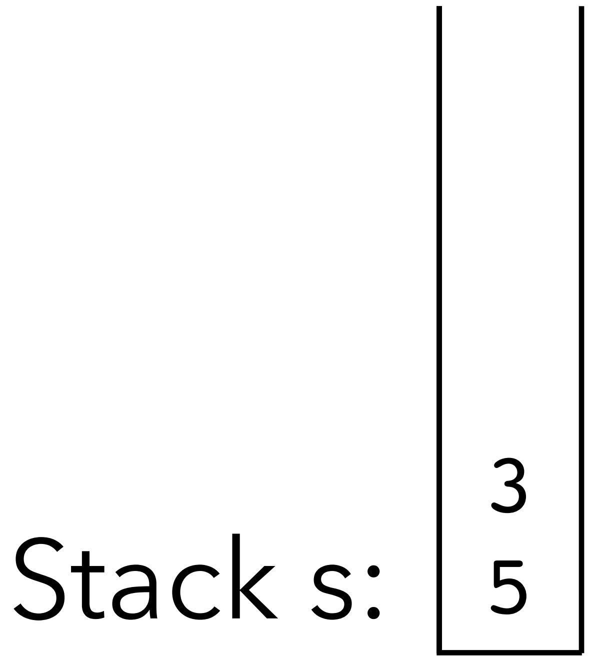
dfs(1)

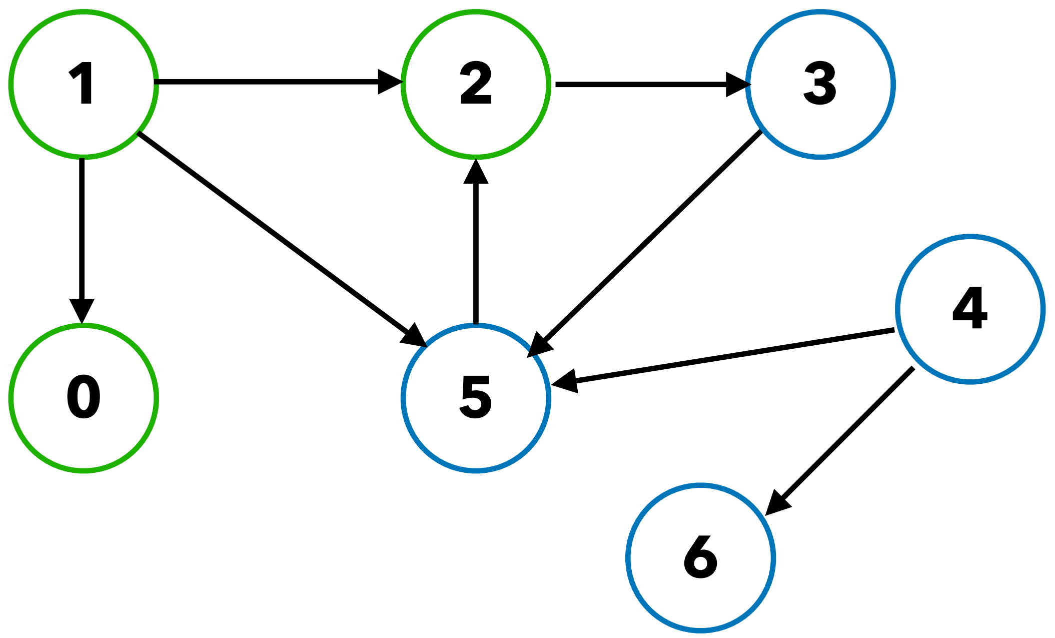


Depth-first Search: Iteratively

```
/** Visit all nodes explorable from u.
  * Pre: u is unvisited. */
public static void dfs(int nodeID) {
  Stack s = (nodeID); // Not Java!
  // inv: all nodes to be visited are
  //      explorable from some node in s
  while (s is not empty) {
    u = s.pop();
    if (u has not been visited) {
      visit u;
→     for each edge (u, v) from u:
        s.push(v);
    }
  }
}
```

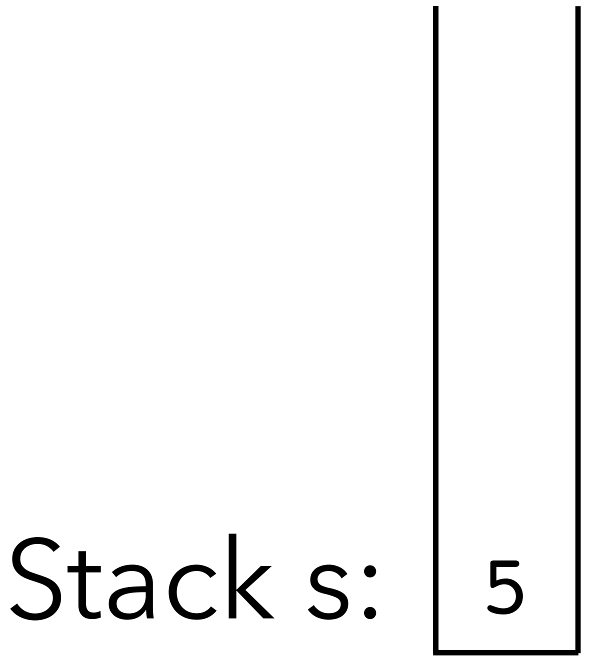
dfs(1)



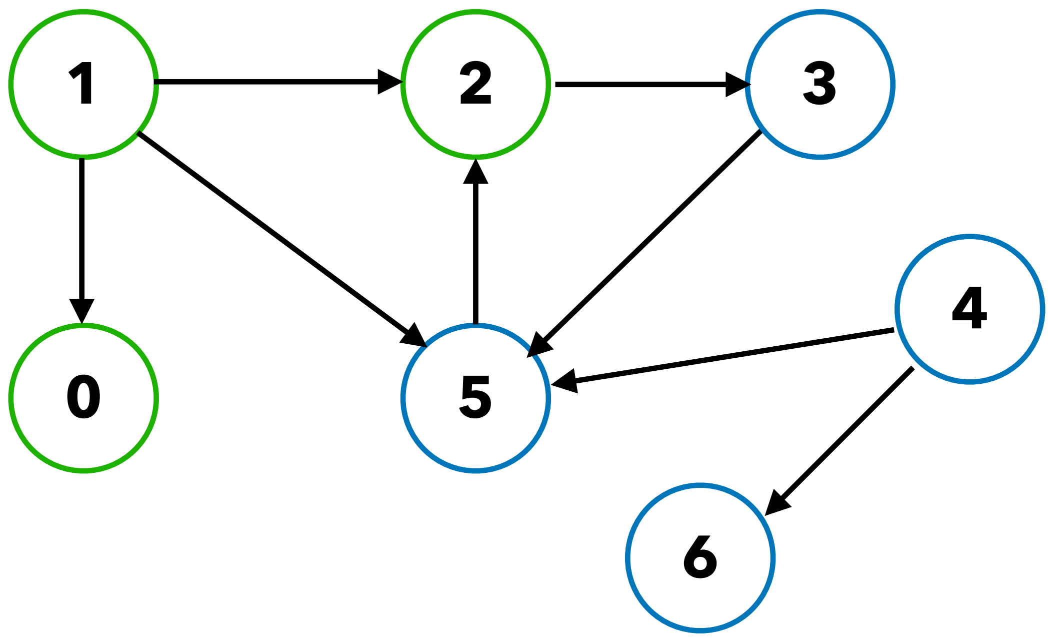


Depth-first Search: Iteratively

```
/** Visit all nodes explorable from u.
  * Pre: u is unvisited. */
public static void dfs(int nodeID) {
  Stack s = (nodeID); // Not Java!
  // inv: all nodes to be visited are
  //      explorable from some node in s
  while (s is not empty) {
    u = s.pop();
    if (u has not been visited) {
      visit u;
      for each edge (u, v) from u:
        s.push(v);
    }
  }
}
```



dfs(1)



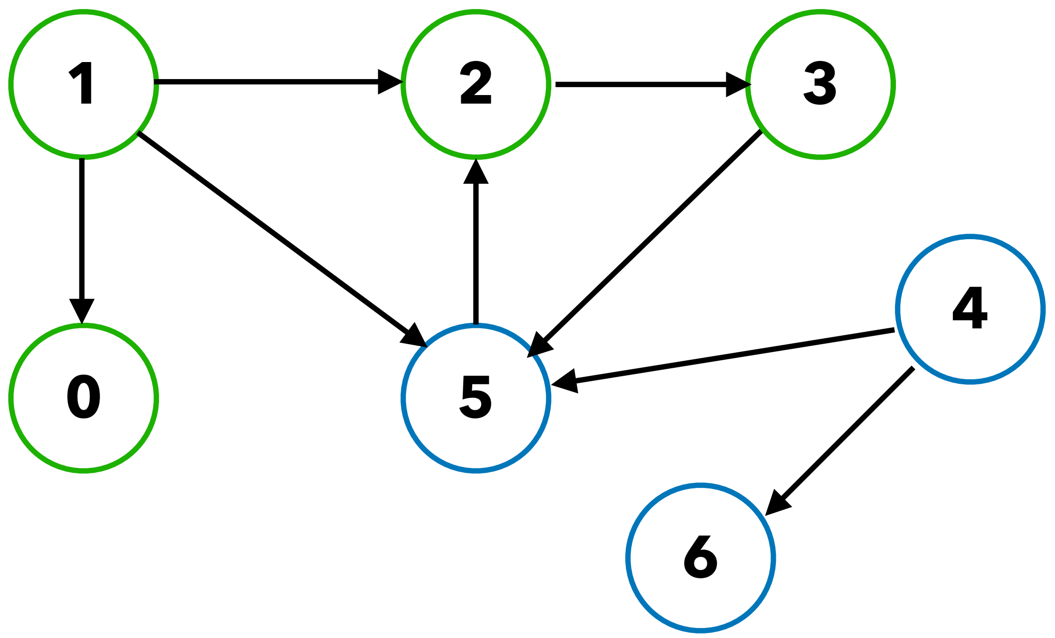
Depth-first Search: Iteratively

```
/** Visit all nodes explorable from u.
  * Pre: u is unvisited. */
public static void dfs(int nodeID) {
  Stack s = (nodeID); // Not Java!
  // inv: all nodes to be visited are
  //      explorable from some node in s
  while (s is not empty) {
    u = s.pop();
    if (u has not been visited) {
      visit u;
→     for each edge (u, v) from u:
        s.push(v);
    }
  }
}
```

Stack s:

5
5

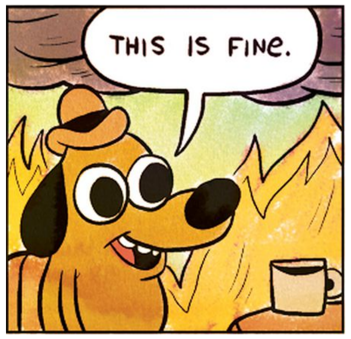
dfs(1)



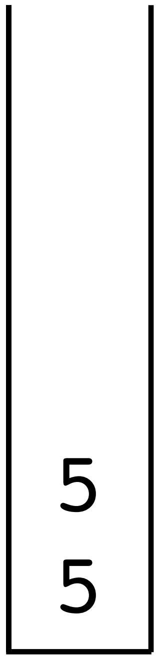
Depth-first Search: Iteratively

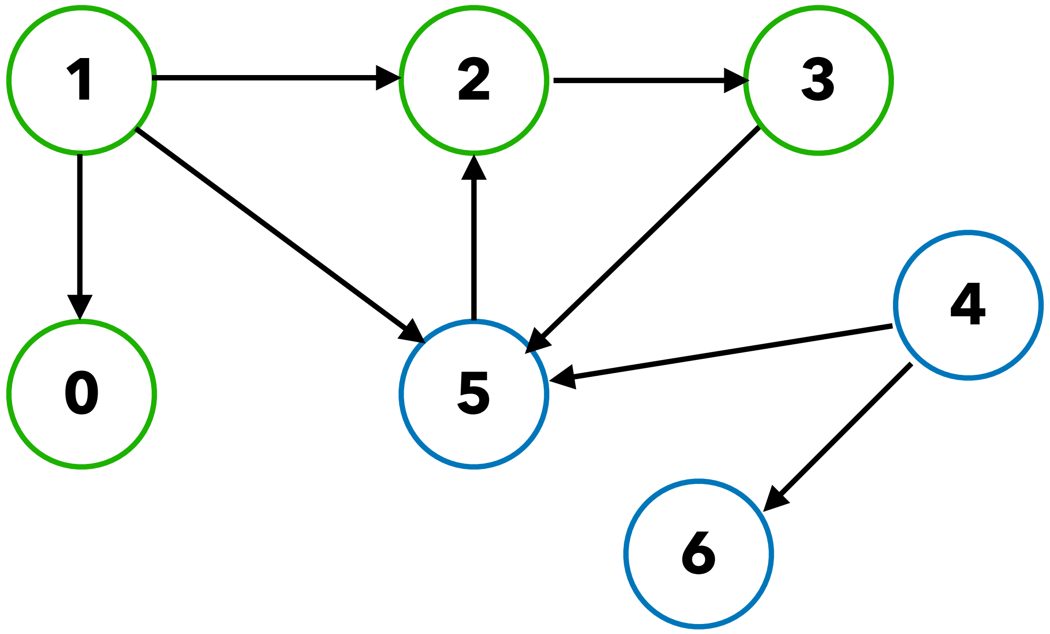
```
/** Visit all nodes explorable from u.
  * Pre: u is unvisited. */
public static void dfs(int nodeID) {
  Stack s = (nodeID); // Not Java!
  // inv: all nodes to be visited are
  //      explorable from some node in s
  while (s is not empty) {
    u = s.pop();
    if (u has not been visited) {
      visit u;
→     for each edge (u, v) from u:
        s.push(v);
    }
  }
}
```

dfs(1)



Stack s:





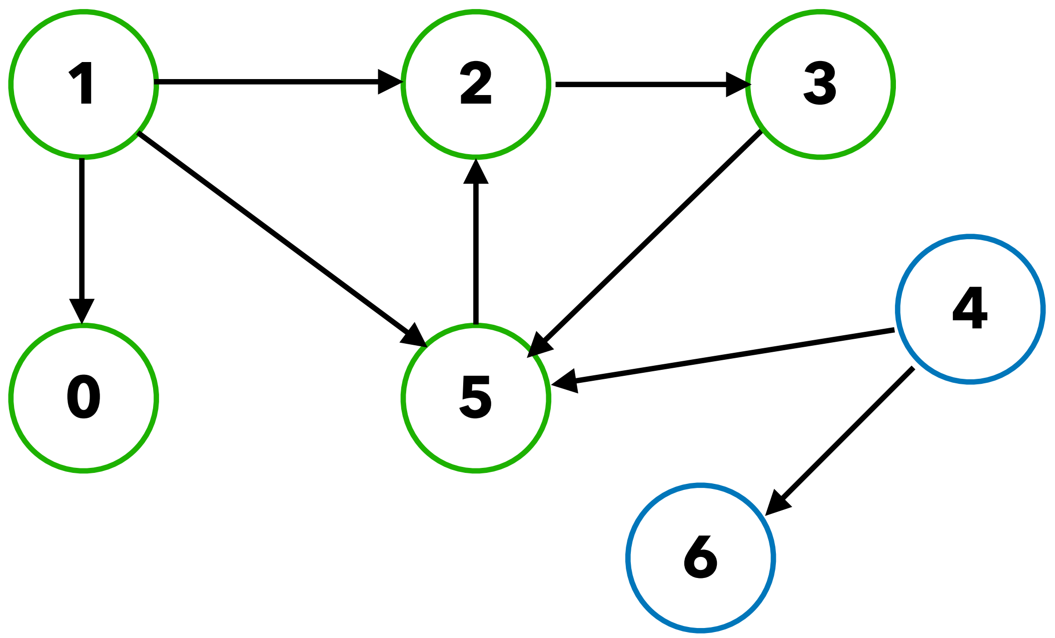
Depth-first Search: Iteratively

```
/** Visit all nodes explorable from u.
  * Pre: u is unvisited. */
public static void dfs(int nodeID) {
  Stack s = (nodeID); // Not Java!
  // inv: all nodes to be visited are
  //      explorable from some node in s
  while (s is not empty) {
    u = s.pop();
    if (u has not been visited) {
→     visit u;
      for each edge (u, v) from u:
        s.push(v);
    }
  }
}
```

Stack s:

5

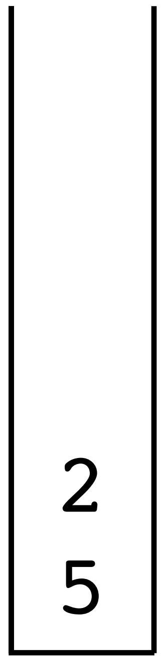
dfs(1)



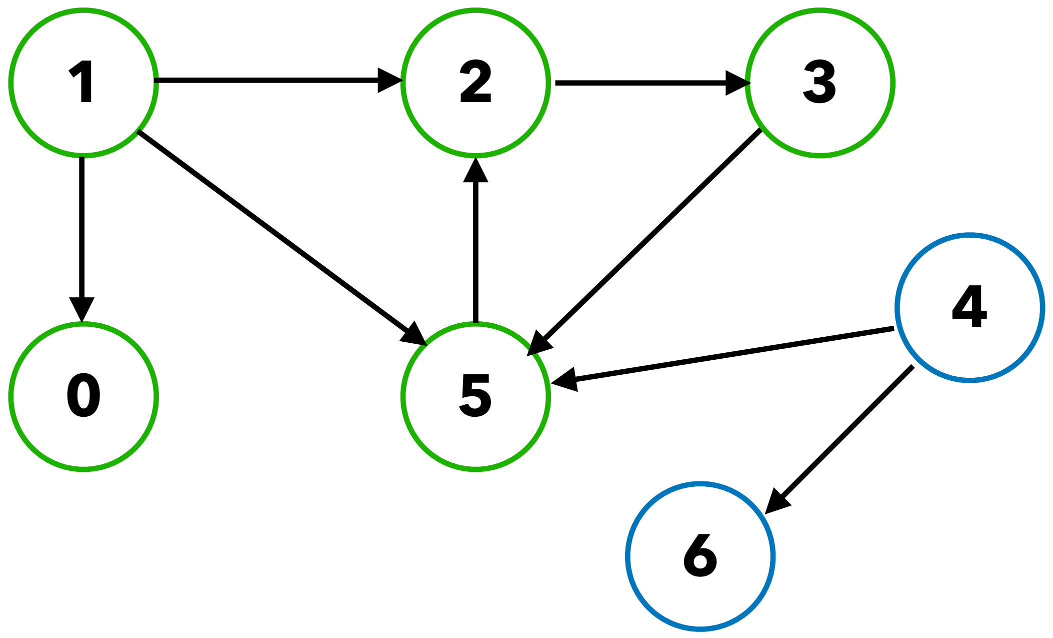
Depth-first Search: Iteratively

```
/** Visit all nodes explorable from u.
  * Pre: u is unvisited. */
public static void dfs(int nodeID) {
  Stack s = (nodeID); // Not Java!
  // inv: all nodes to be visited are
  //      explorable from some node in s
  while (s is not empty) {
    u = s.pop();
    if (u has not been visited) {
      visit u;
      for each edge (u, v) from u:
→       s.push(v);
    }
  }
}
```

Stack s:



dfs(1)

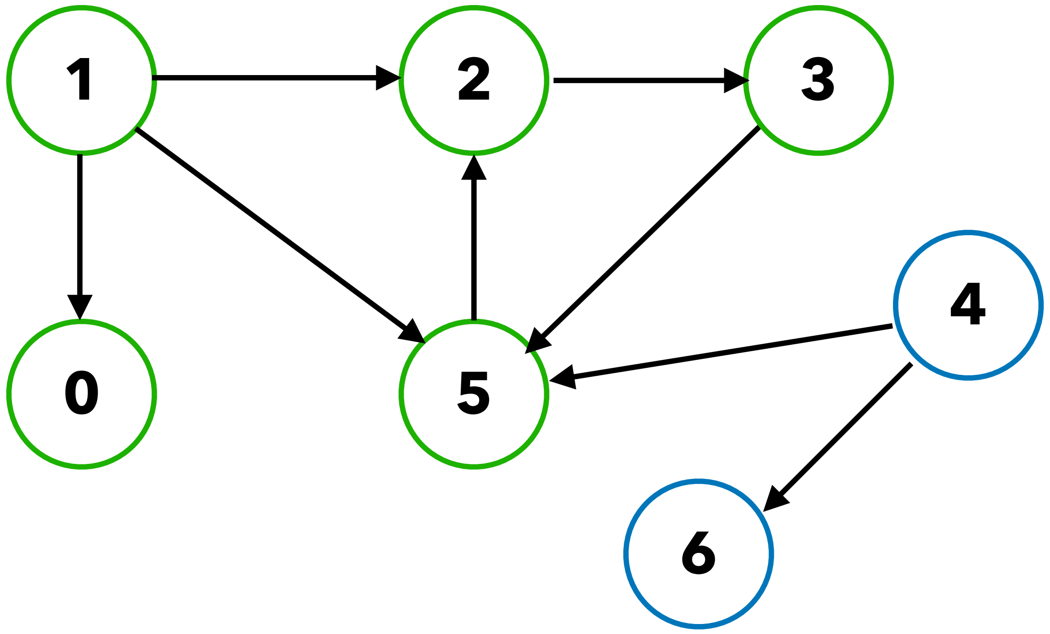


Depth-first Search: Iteratively

```
/** Visit all nodes explorable from u.
  * Pre: u is unvisited. */
public static void dfs(int nodeID) {
  Stack s = (nodeID); // Not Java!
  // inv: all nodes to be visited are
  //      explorable from some node in s
  while (s is not empty) {
    u = s.pop();
→   if (u has not been visited) {
      visit u;
      for each edge (u, v) from u:
        s.push(v);
    }
  }
}
```

Stack s: 5

dfs(1)

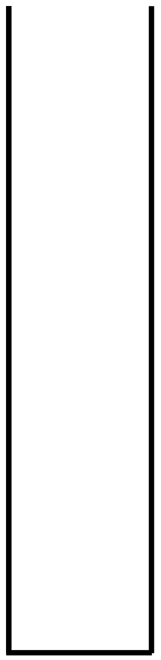


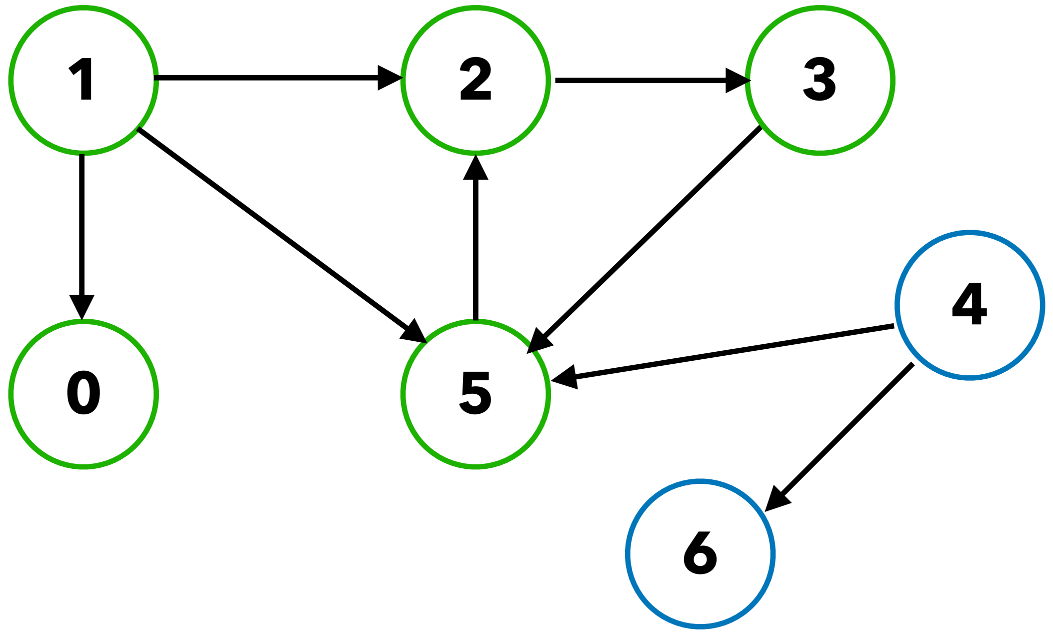
Depth-first Search: Iteratively

```
/** Visit all nodes explorable from u.
  * Pre: u is unvisited. */
public static void dfs(int nodeID) {
  Stack s = (nodeID); // Not Java!
  // inv: all nodes to be visited are
  //      explorable from some node in s
  while (s is not empty) {
→   u = s.pop();
    if (u has not been visited) {
      visit u;
      for each edge (u, v) from u:
        s.push(v);
    }
  }
}
```

dfs(1)

Stack s:



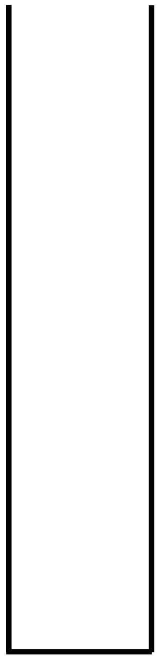


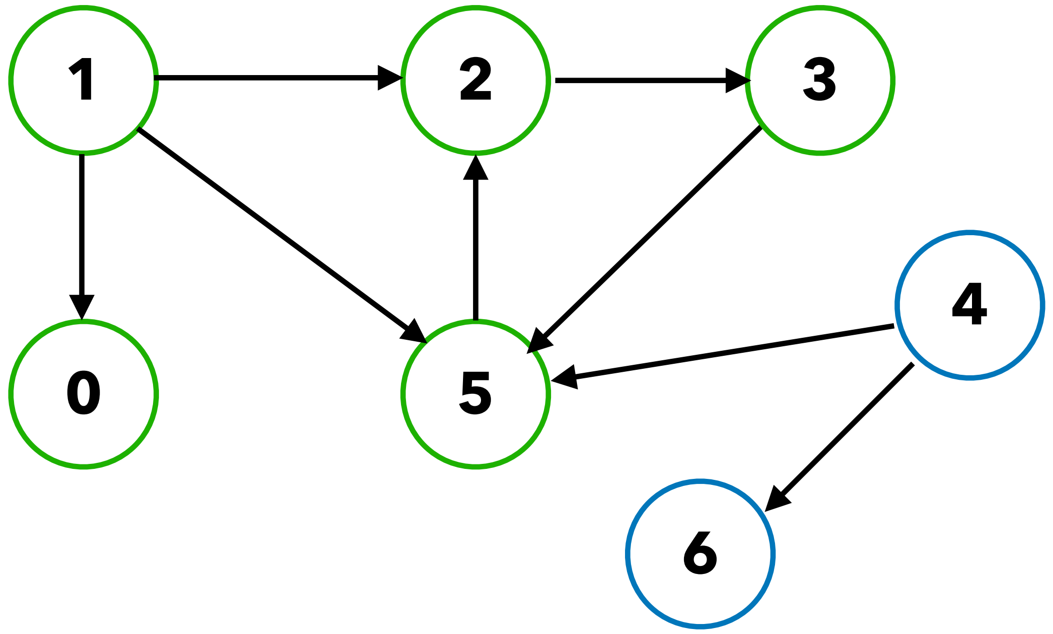
Depth-first Search: Iteratively

```
/** Visit all nodes explorable from u.
  * Pre: u is unvisited. */
public static void dfs(int nodeID) {
  Stack s = (nodeID); // Not Java!
  // inv: all nodes to be visited are
  //      explorable from some node in s
  while (s is not empty) {
    u = s.pop();
→   if (u has not been visited) {
      visit u;
      for each edge (u, v) from u:
        s.push(v);
    }
  }
}
```

dfs(1)

Stack s:





Depth-first Search: Iteratively

```
/** Visit all nodes explorable from u.
  * Pre: u is unvisited. */
public static void dfs(int nodeID) {
  Stack s = (nodeID); // Not Java!
  // inv: all nodes to be visited are
  //      explorable from some node in s
→ while (s is not empty) {
    u = s.pop();
    if (u has not been visited) {
      visit u;
      for each edge (u, v) from u:
        s.push(v);
    }
  }
}
```

Stack s:

dfs(1)