CSCI 241

Scott Wehrwein

Graph Traversals:
Breadth-First Search

Goals

Be able to execute and implement breadth-first search to search or traverse a graph.

Depth-first Search

```
/** 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);
    }
  }
}
```

Depth-first Search

```
/** 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);
    }
  }
}
```

Time for a magic trick…

Depth-first Search

```
/** 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);
    }
  }
}
```

Depth-first Search

```
/** 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);
    }
  }
}
```

Time for a magic trick...

Breadth-first Search

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

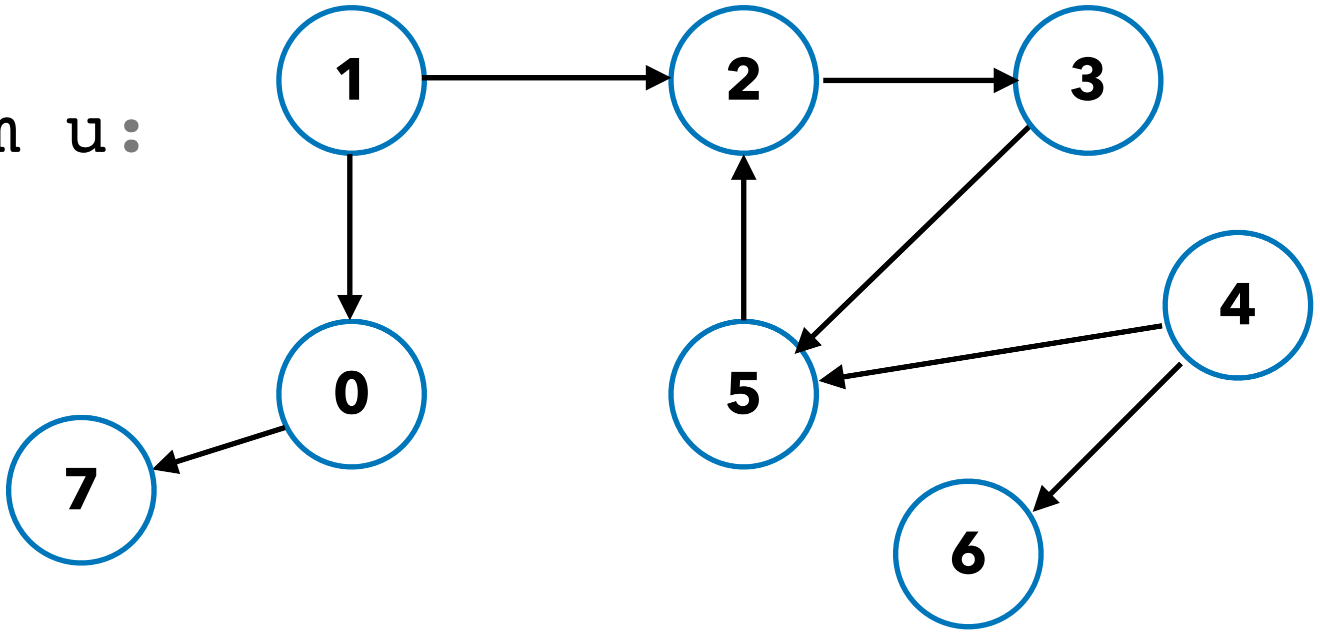
Breadth-first Search

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

Queue q:

1

bfs(1)



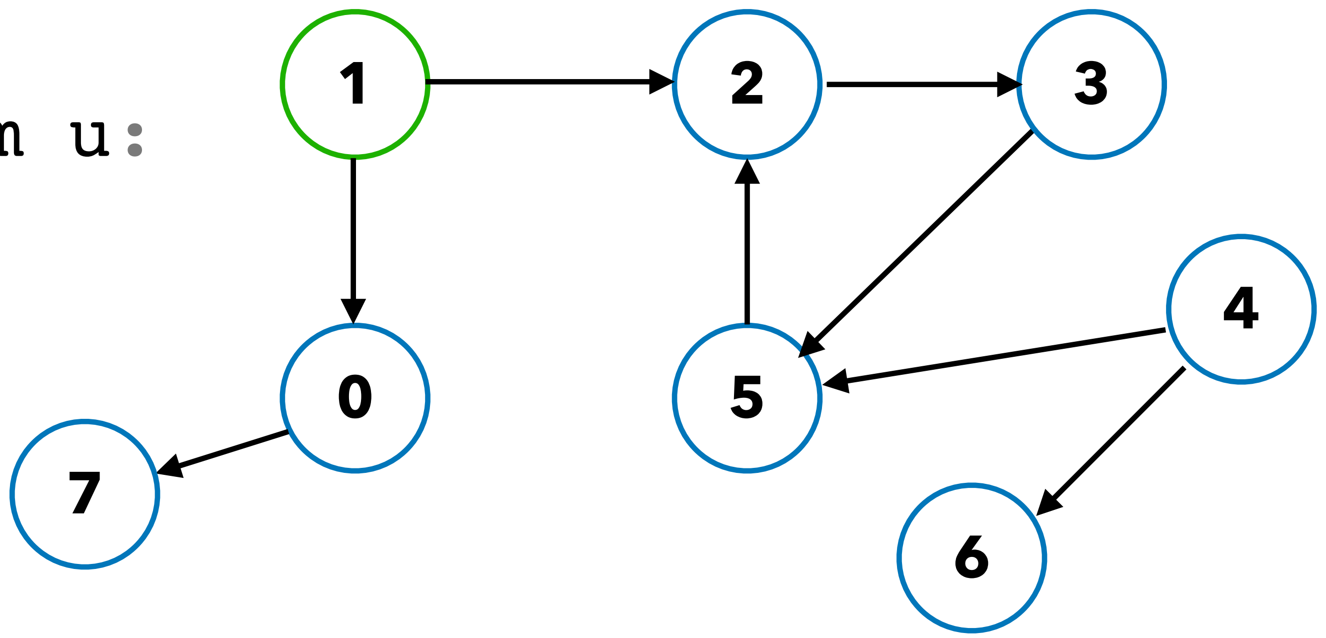
Breadth-first Search

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

Queue q:

0 2

bfs(1)



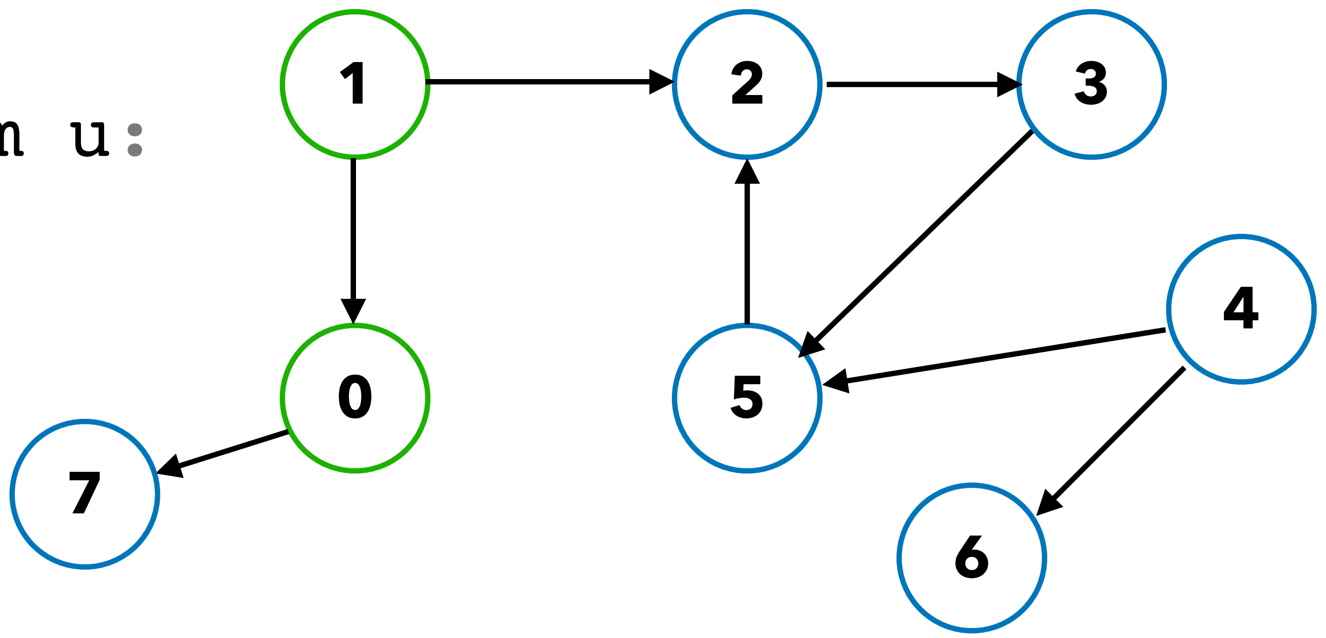
Breadth-first Search

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

Queue q:

```
2 7
```

bfs(1)



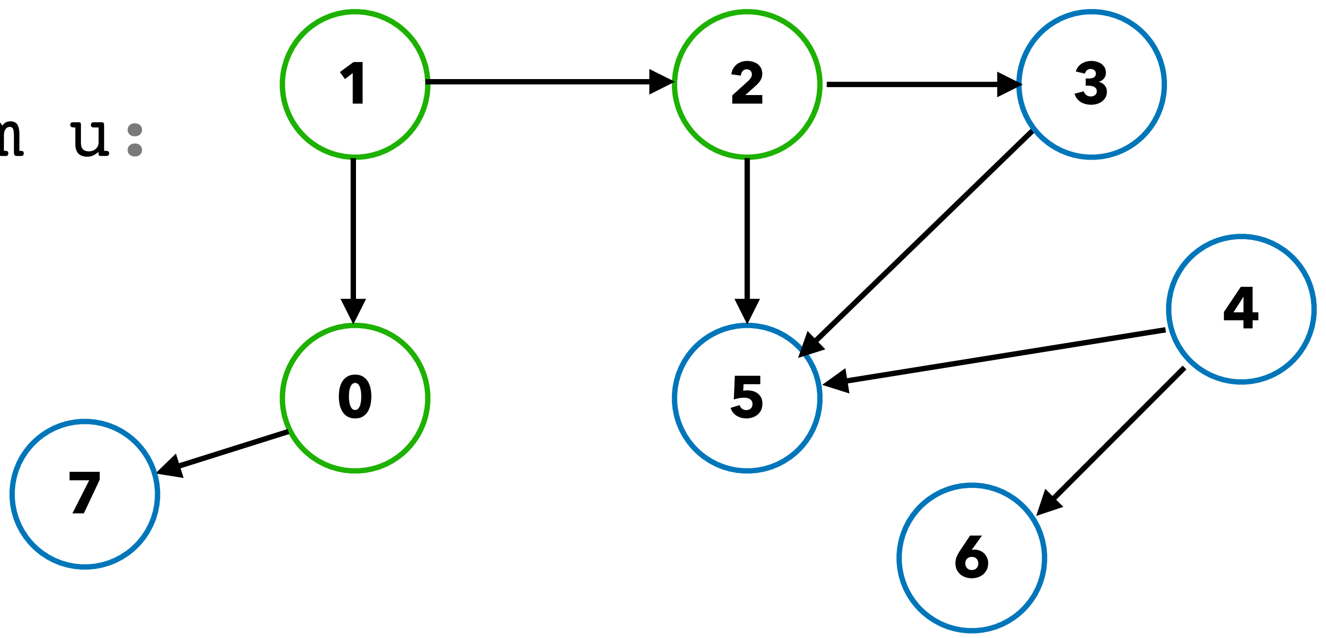
Breadth-first Search

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

Queue q:

7	3	5

bfs(1)



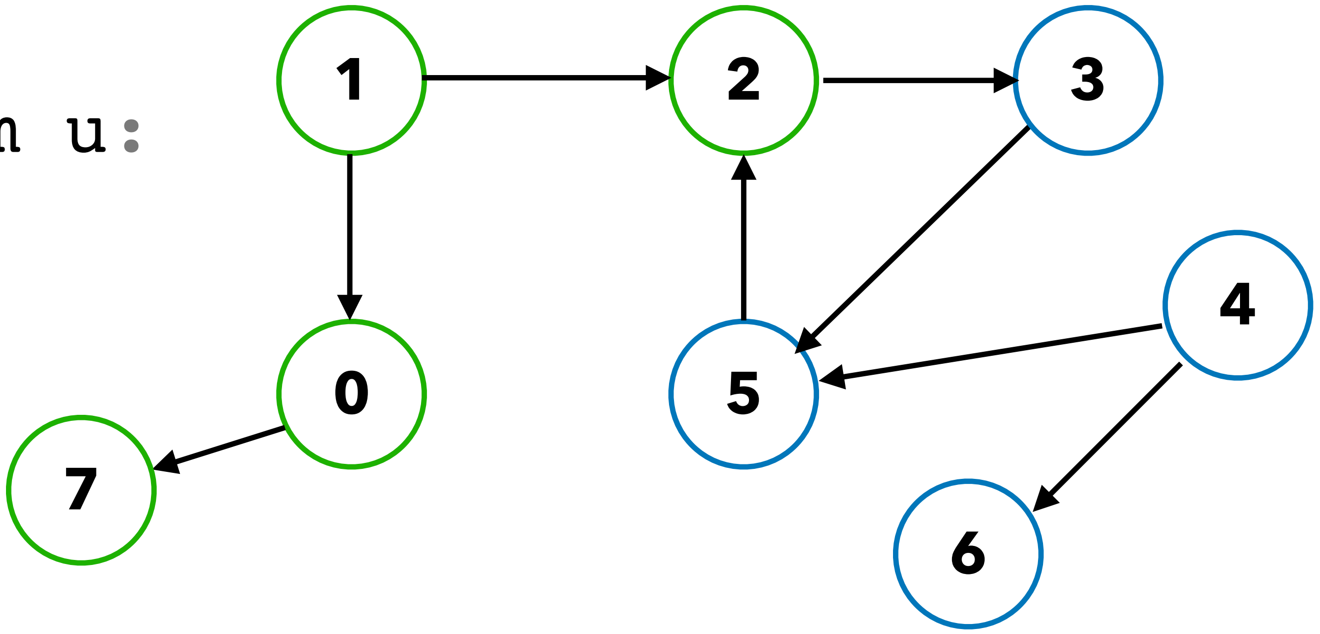
Breadth-first Search

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

Queue q:

3	5

bfs(1)



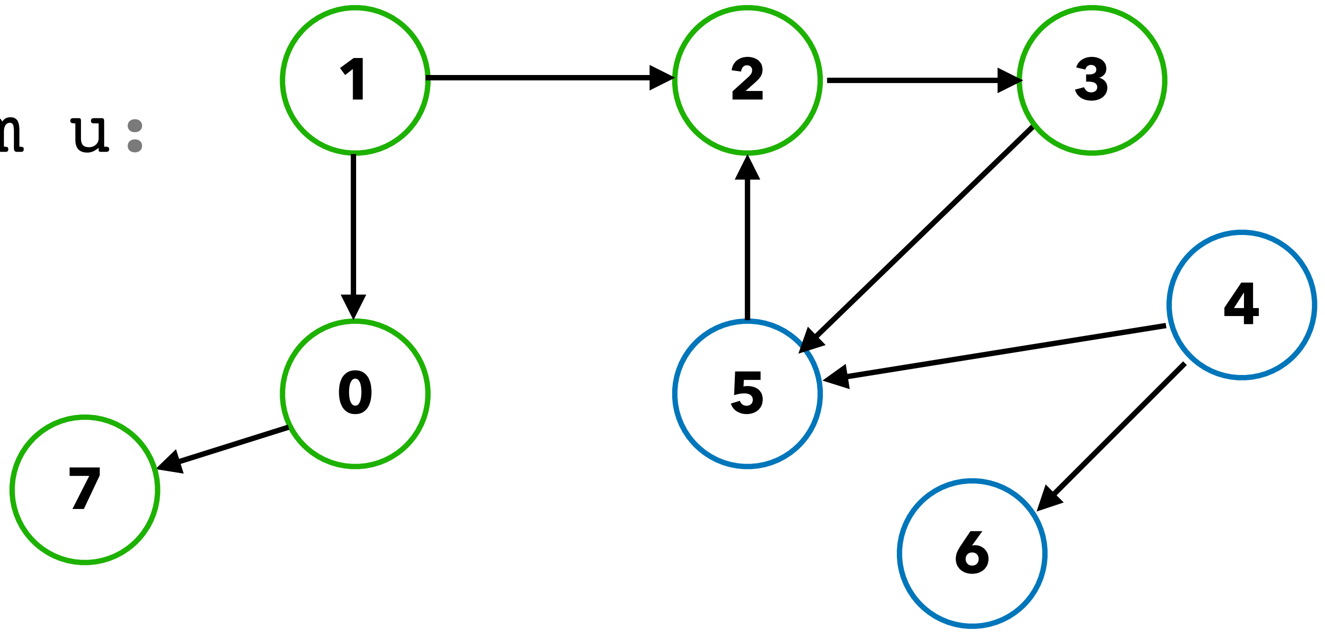
Breadth-first Search

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

Queue q:

```
5 5
```

bfs(1)

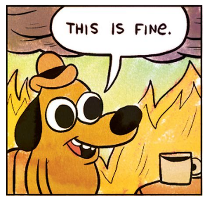


Breadth-first Search

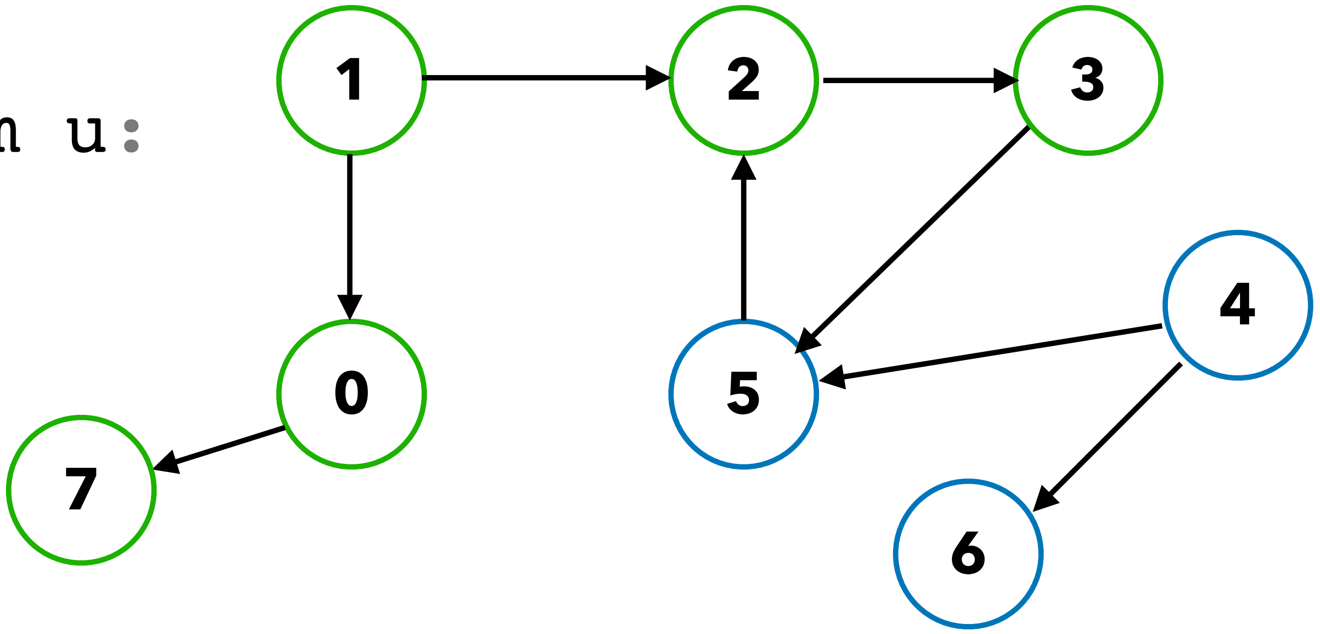
```
/** Visit all nodes explorable from u.
  * Pre: u is unvisited. */
public static void bfs(int nodeID) {
  Queue q = (nodeID); // Not Java!
  // inv: all nodes to be visited are
  //       explorable from some node in q
  while (q is not empty) {
    u = q.dequeue();
    if (u has not been visited)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

Queue q:

5 5



bfs(1)



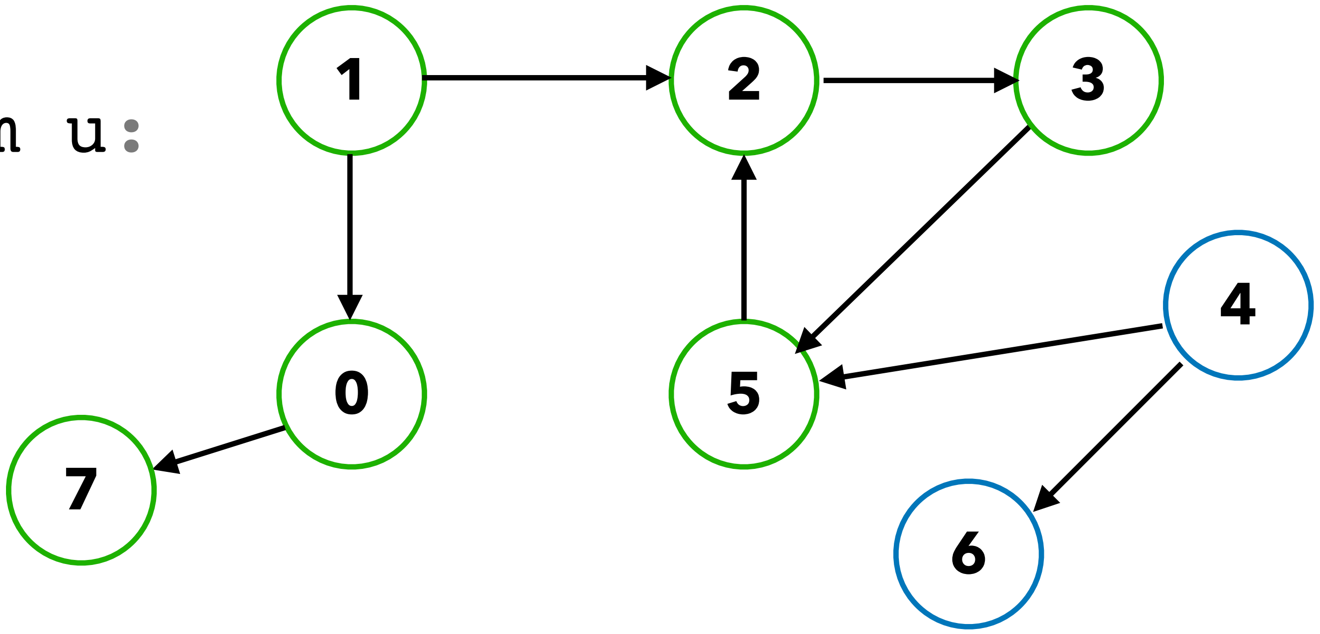
Breadth-first Search

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

Queue q:

5

bfs(1)



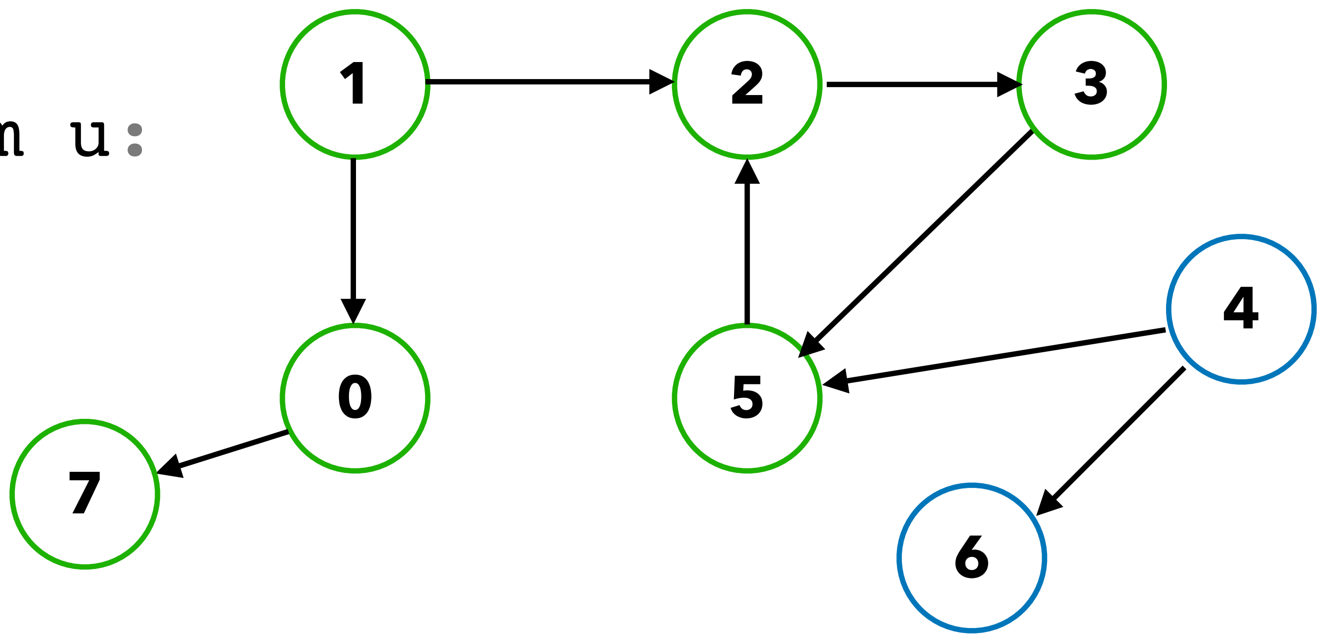
Breadth-first Search

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

Queue q:

5 2

bfs(1)



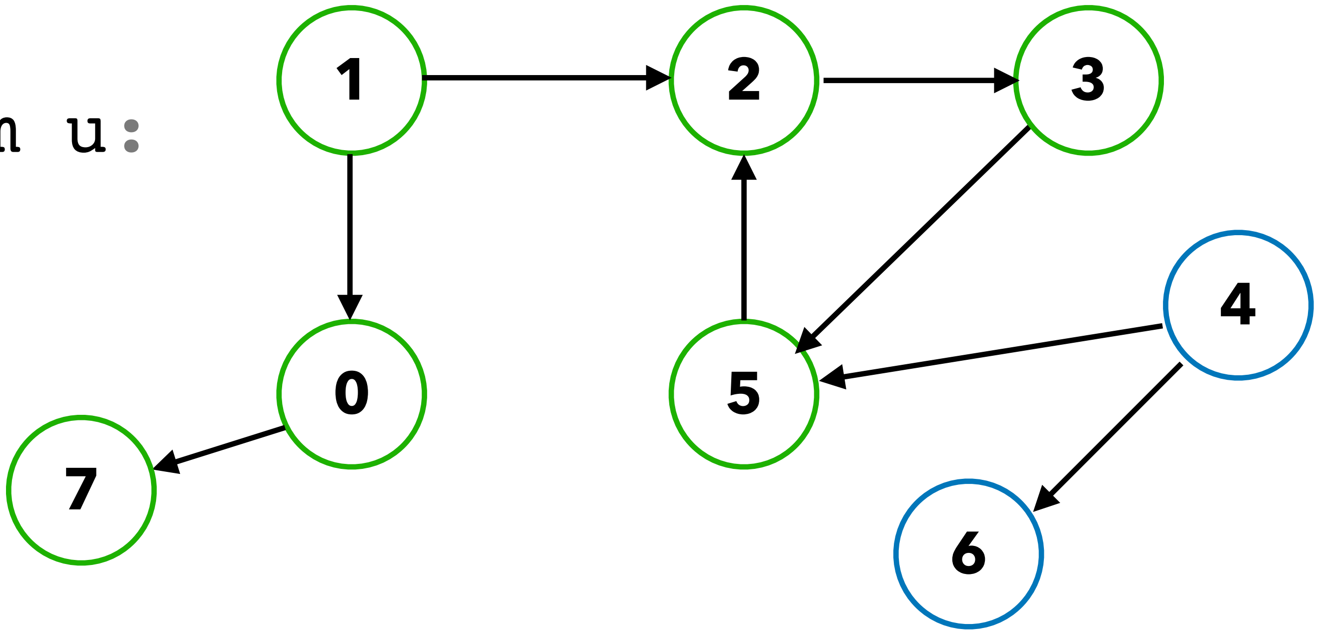
Breadth-first Search

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

Queue q:

2

bfs(1)



Breadth-first Search

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

Queue q:

bfs(1)