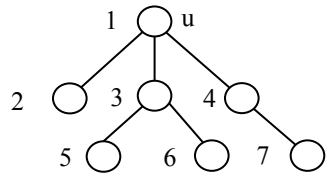
Breadth-first search
David Gries

Breadth-first search of a graph visits all nodes of a graph that are reachable along unvisited paths from node u in the following order:



First u.
Then all nodes that are 1 edge from u.
Then all nodes that are 2 edges from u,
And so forth.

Here is the iterative depth-first search algorithm that we developed earlier:

```
/** Visit every node reachable along a path of unvisited nodes from node u.
      Precondition: u has not been visited. */
public static void dfsIterative(Node u) {
      Stack s= (u); // Not Java!
      // Invariant: all nodes (and only those nodes) that have to be visited are
      // reachable along a path of unvisited nodes from some node in s.
      while (s is not empty) {
          u= s.pop();
          if (u is not visited) {
                Visit u;
                For each neighbor w of u:
                   s.push(w);
          }
      }
}
```

We change it into a breadth-first search simply by changing s from a stack to a queue!

```
/** Visit every node reachable along a path of unvisited nodes from node u.
      Precondition: u has not been visited. */
public static void bfs(Node u) {
      Queue s= (u); // Not Java!
      // Invariant: all nodes (and only those nodes) that have to be visited are
      // reachable along a path of unvisited nodes from some node in s.
      while (s is not empty) {
          u= s.remove(); // remove first element of queue and store it in u
          if (u is not visited) {
                Visit u;
                For each neighbor w of u:
                   s.add(w); // append w to queue
          }
      }
}
```

We explain why this results in a breadth-first search. First, for any integer $i \geq 0$, nodes that are i edges from u are put in the queue before nodes that are i+1 edges from u. Second, nodes are removed from the *front* of the queue and visited (if not yet visited), so those closer to u are visited first.