Data Structures & Algorithms

Graph

Graph Traversal/Search

The goal of a graph traversal, generally, is to systematically explore all vertices reachable from a given start vertex in some particular order.

There are mainly two approach for graph traversal.

Depth-First Traversal/Depth-first Search (DFS).
Breadth-First Traversal/Breadth-first Search (BFS).

Breadth First Search (BFS)

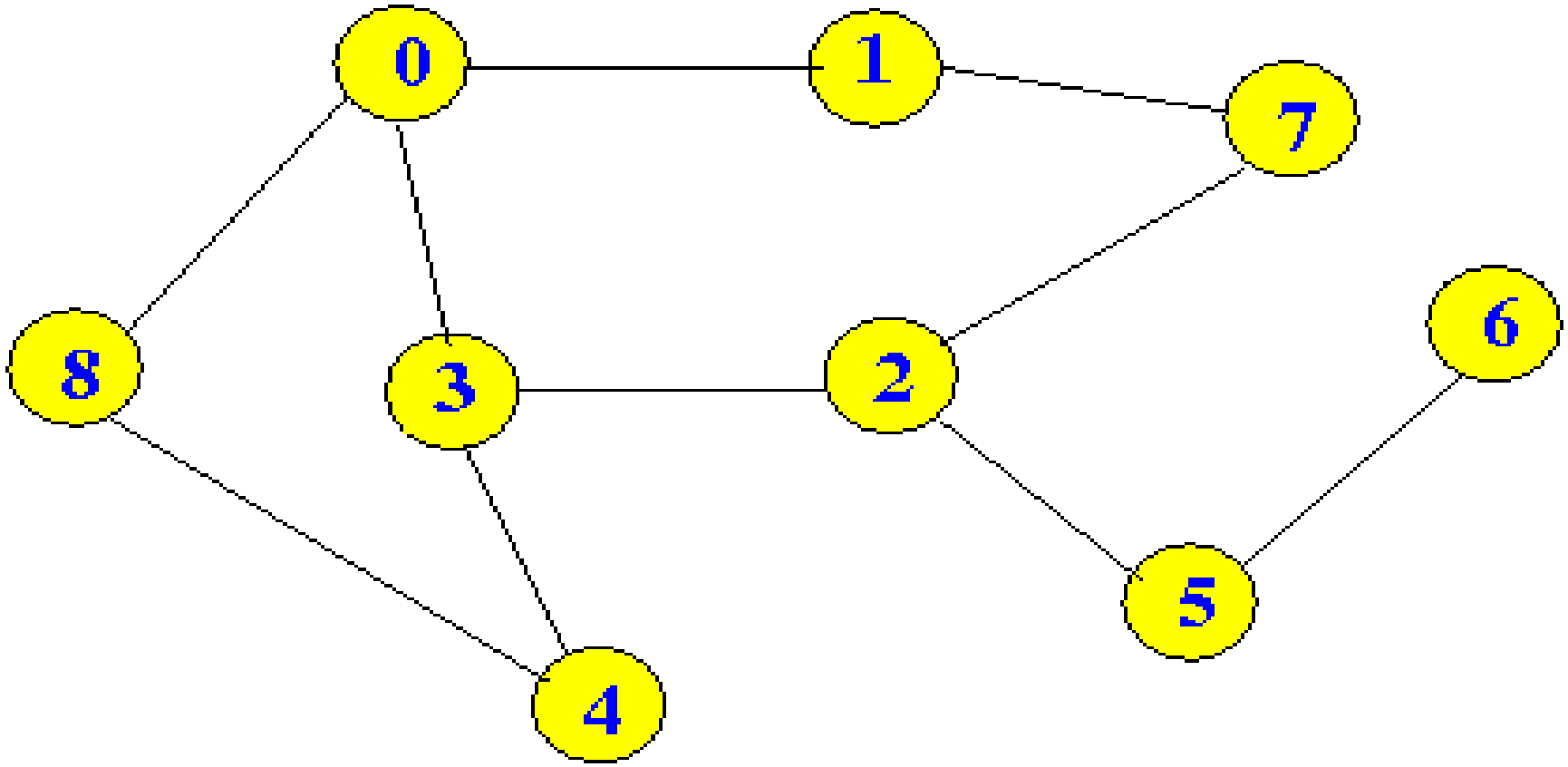
Given a connected, undirected and unweighted graph G = (V, E) and a designated source vertex s, breadth-first search systematically traverse every vertex of G that is reachable from s.

Idea: BFS logically subdivides the vertices into levels/layers. Proceed the graph level by level (one level at a time, left to right within a level). Visit all vertices on one level before moving on to the next level.
BFS visit all vertices at level k-1 before those at level k (or visit all vertices at level k before all those of level k+1).

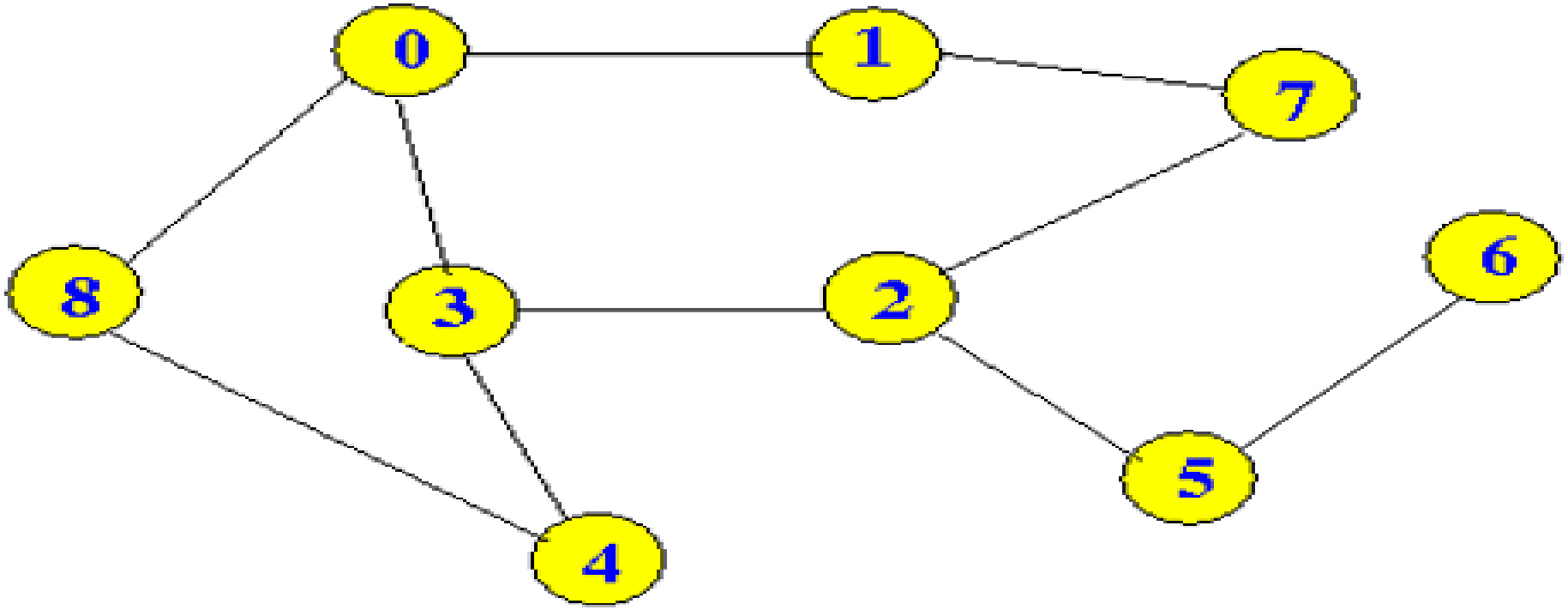
Queue is used in BFS traversal.

Breadth First Search (BFS)

Consider following undirected graph. Start from node 0. Compute the BFS tree. Traverse the following graph using BFS and give BFS traversal sequence.



Breadth First Search (BFS)

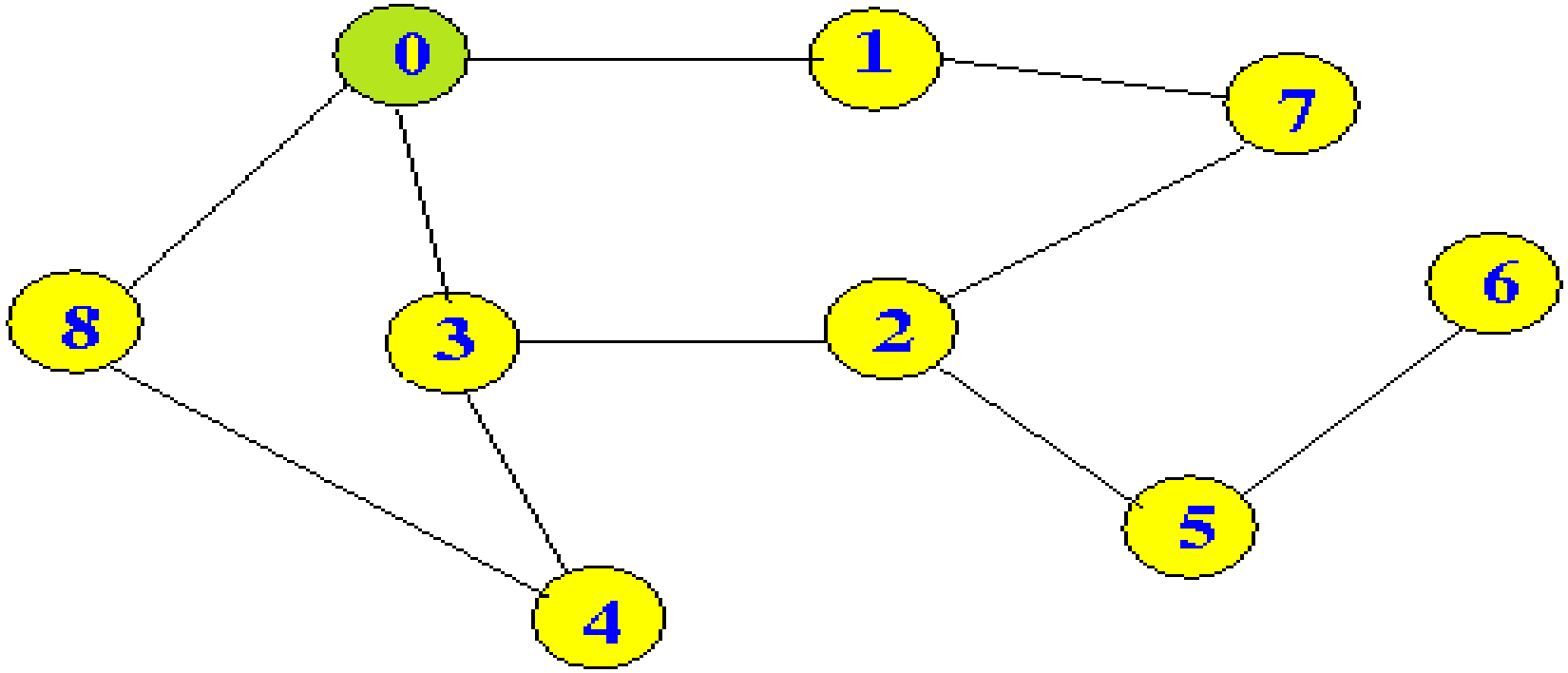


So, visited[]

0	1	2	3	4	5	6	7	8
0	0	0	0	0	0	0	0	0

Q = []

Breadth First Search (BFS)

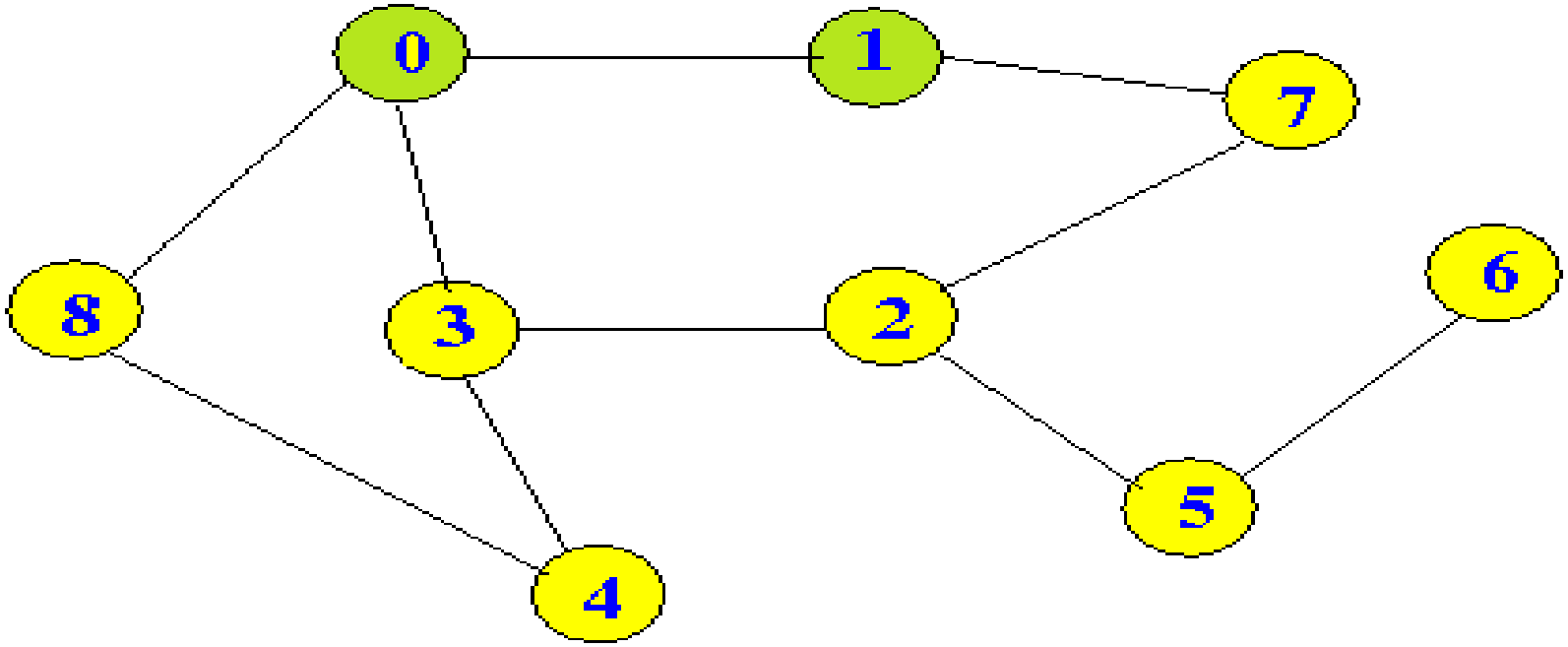


visited[]

0	1	2	3	4	5	6	7	8
1	0	0	0	0	0	0	0	0

Q = [0]
v = 0

Breadth First Search (BFS)

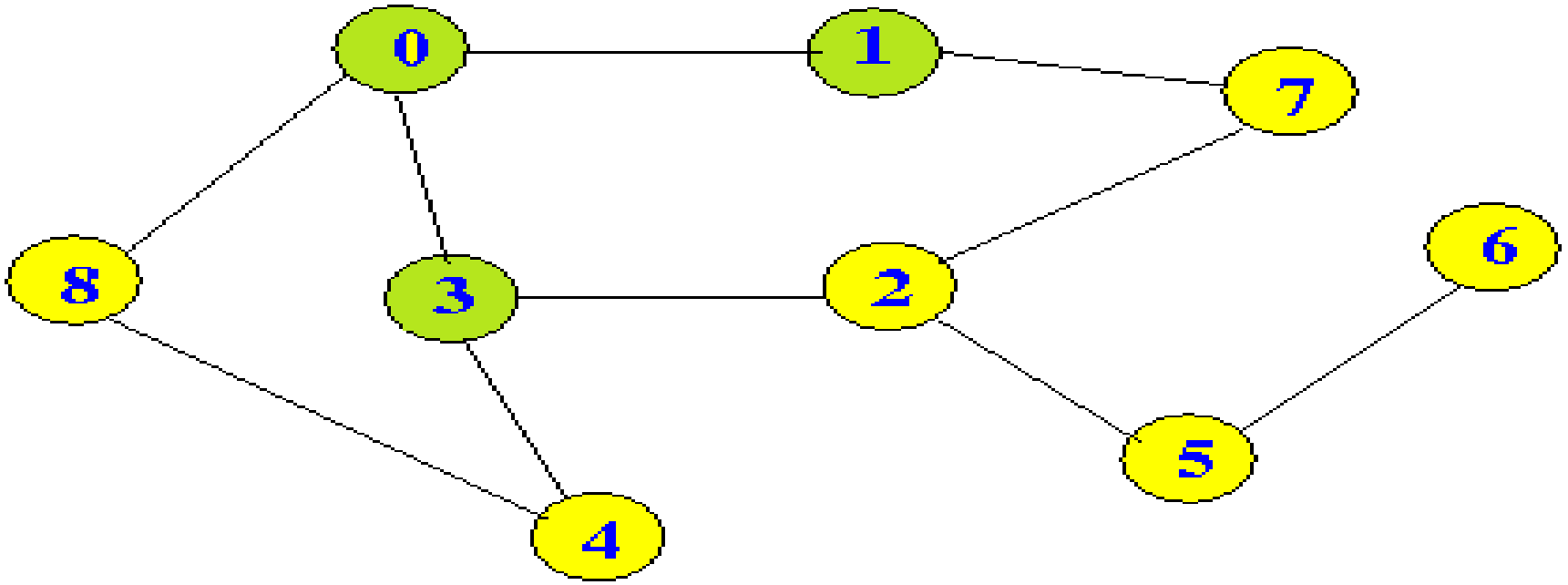


visited[]

0	1	2	3	4	5	6	7	8
1	1	0	1	0	0	0	0	1

Q = [1, 3, 8]
v = 1

Breadth First Search (BFS)

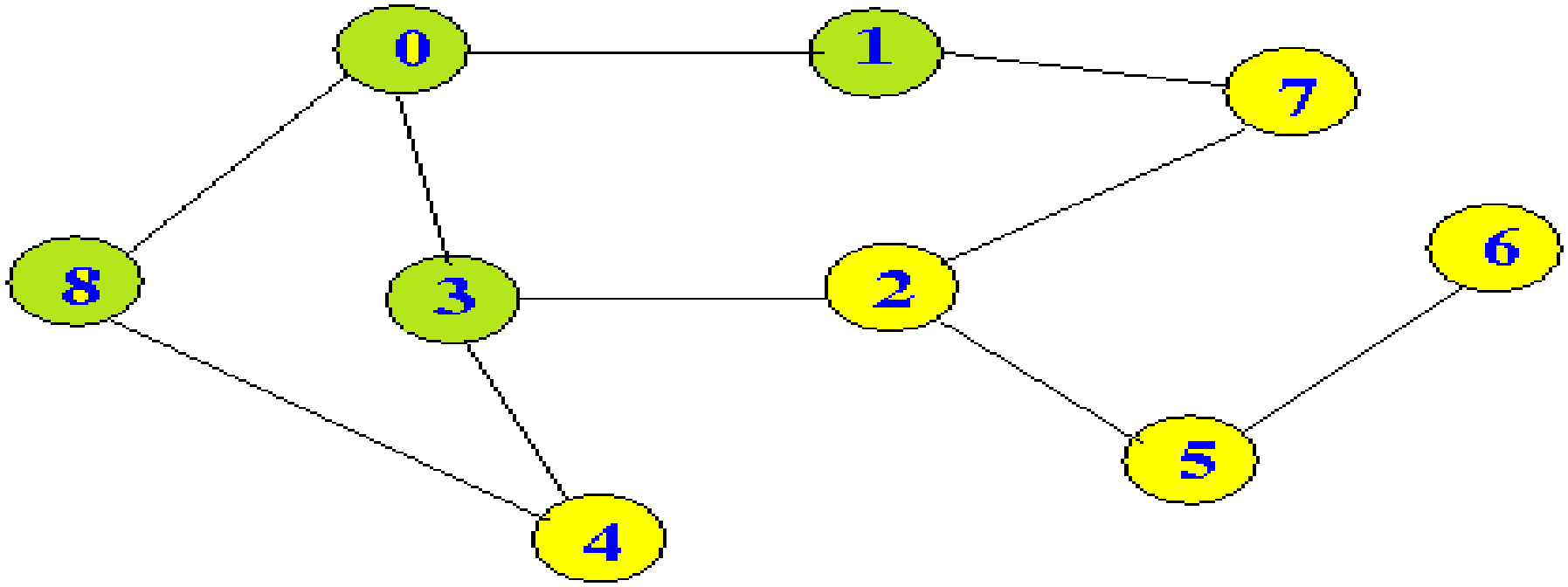


visited[]

0	1	2	3	4	5	6	7	8
1	1	0	1	0	0	0	1	1

Q = [3, 8, 7]
v = 3

Breadth First Search (BFS)

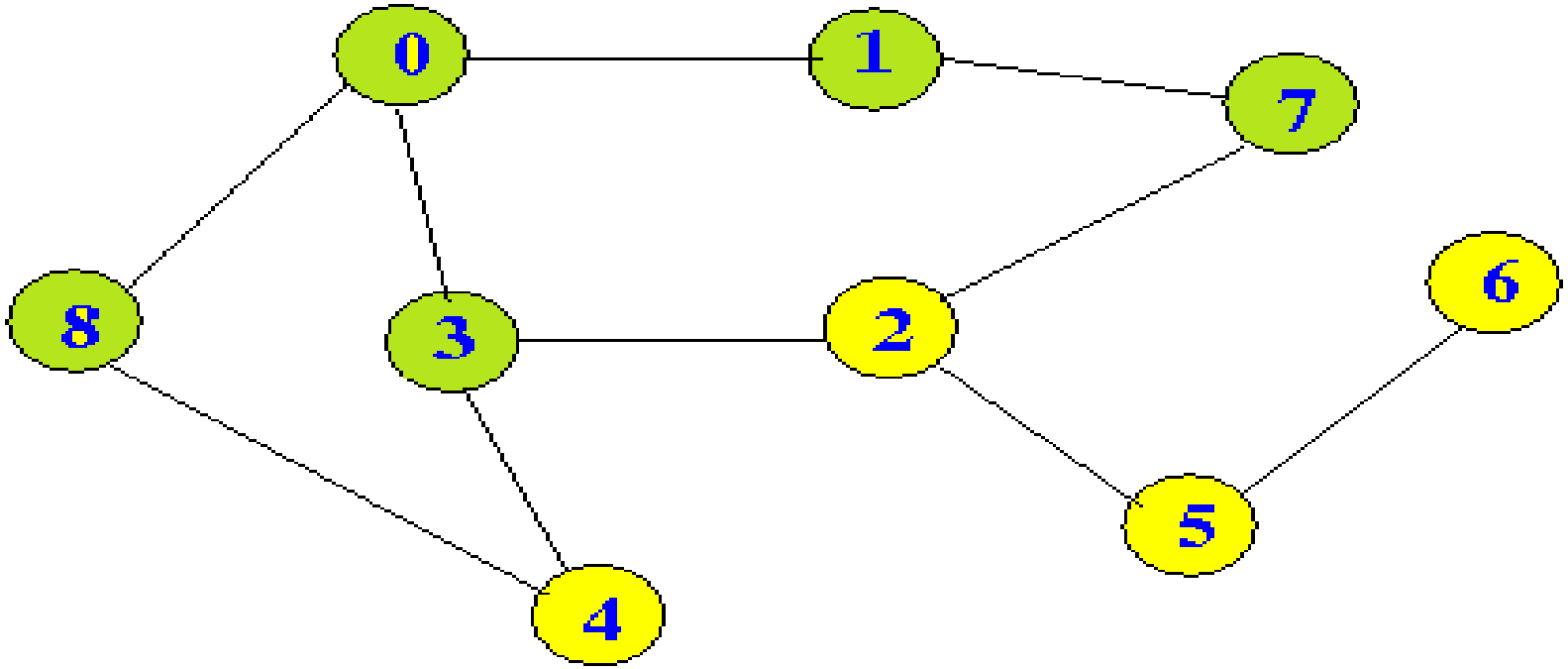


visited[]

0	1	2	3	4	5	6	7	8
1	1	1	1	1	0	0	1	1

Q = [8, 7, 2, 4]
v = 8

Breadth First Search (BFS)

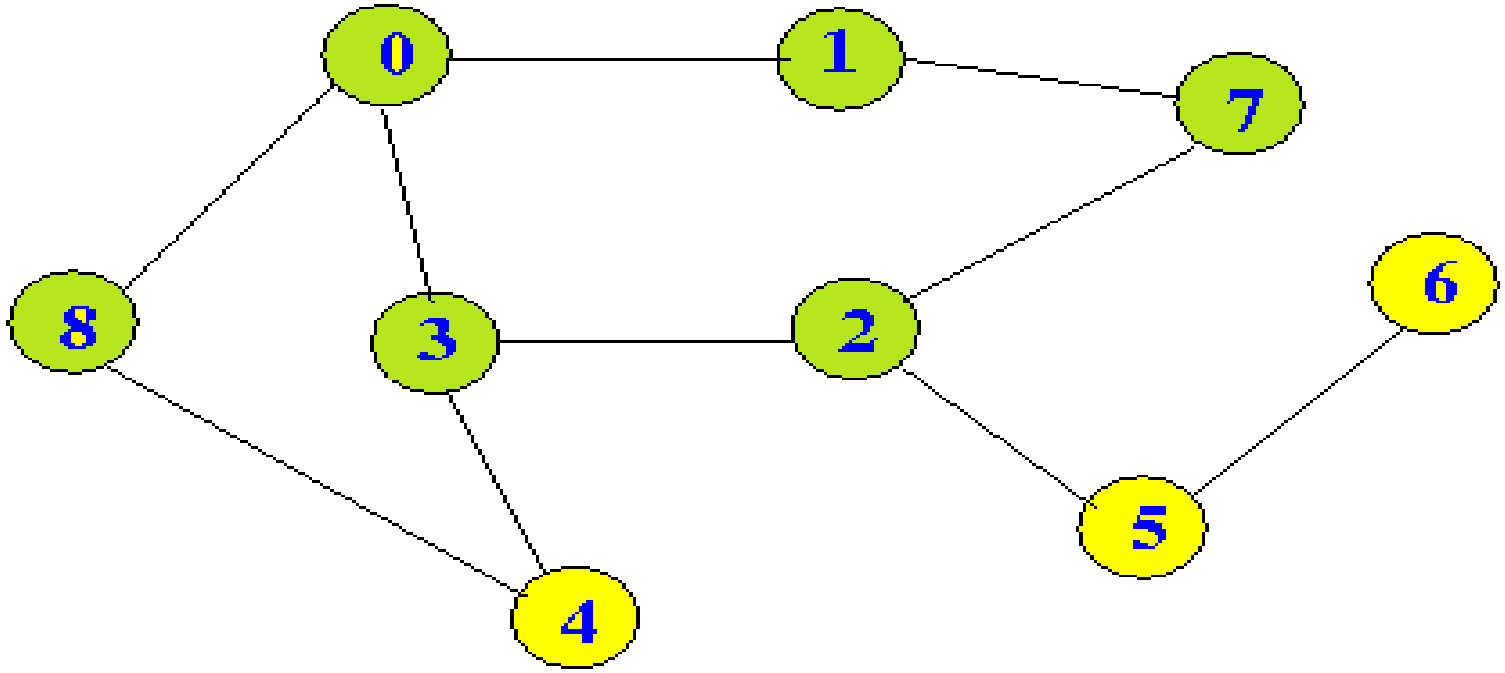


visited[]

0	1	2	3	4	5	6	7	8
1	1	1	1	1	0	0	1	1

Q = [7, 2, 4]
v = 7

Breadth First Search (BFS)

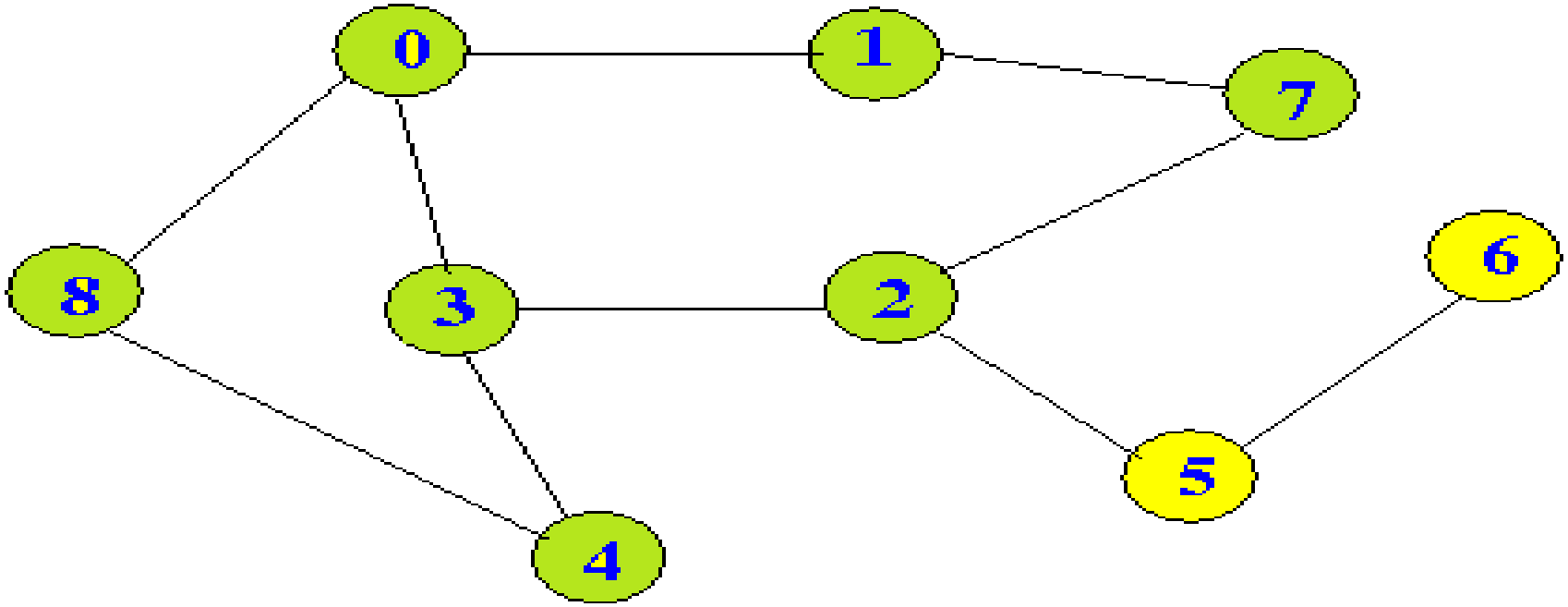


visited[]

0	1	2	3	4	5	6	7	8
1	1	1	1	1	0	0	1	1

Q = [2, 4]
v = 2

Breadth First Search (BFS)

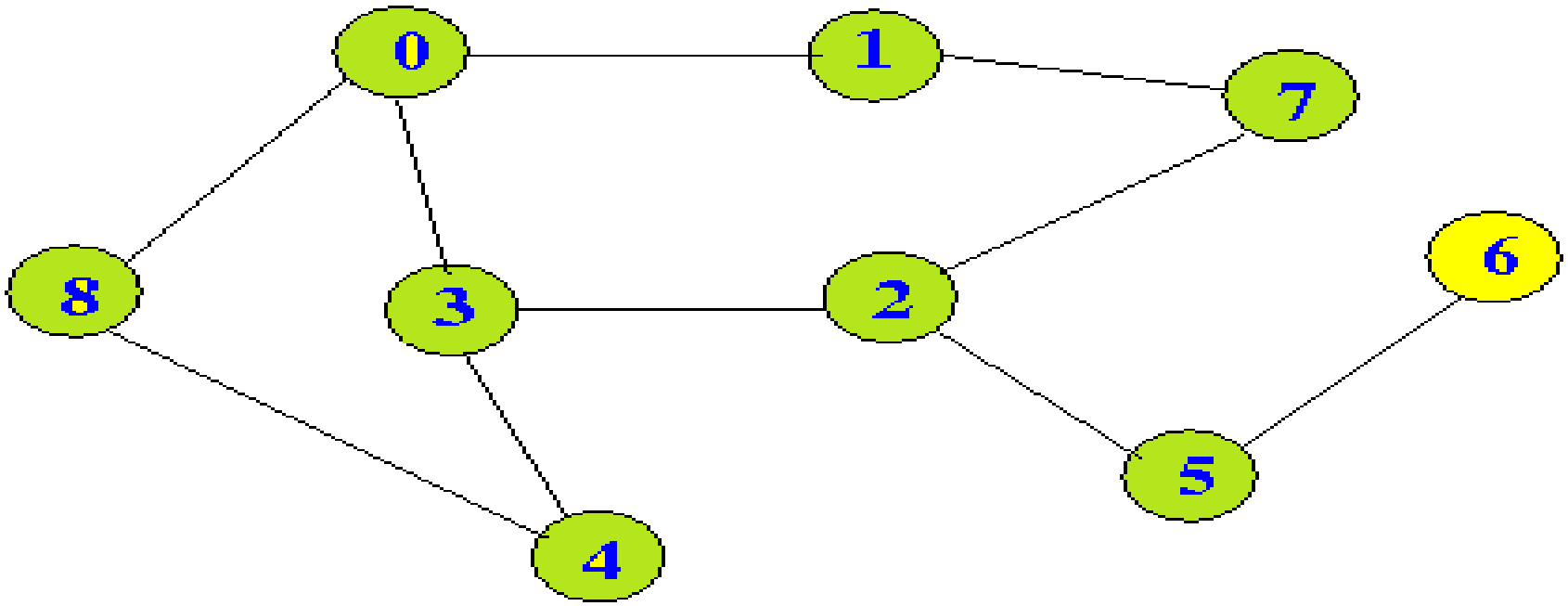


visited[]

0	1	2	3	4	5	6	7	8
1	1	1	1	1	1	0	1	1

Q = [4, 5]
v = 4

Breadth First Search (BFS)

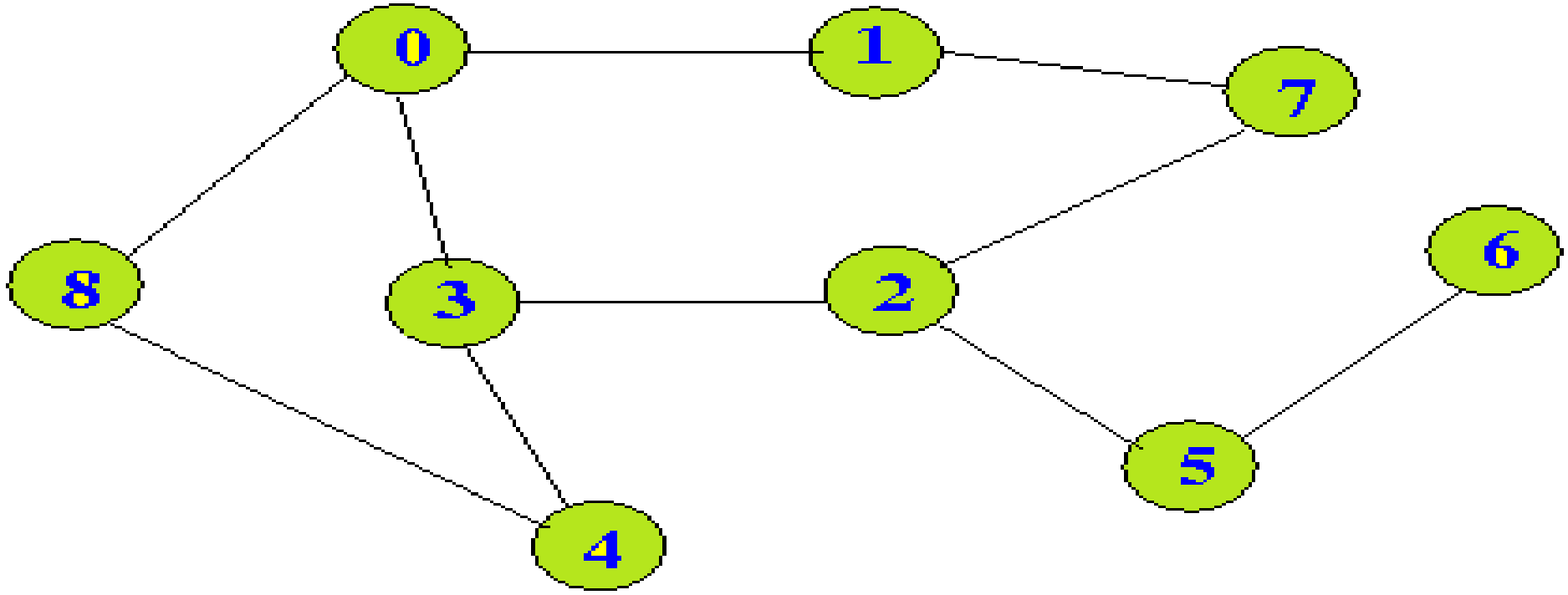


visited[]

0	1	2	3	4	5	6	7	8
1	1	1	1	1	1	0	1	1

Q = [5]
v = 5

Breadth First Search (BFS)



visited[]

0	1	2	3	4	5	6	7	8
1	1	1	1	1	1	1	1	1

Q = [6]

v = 6

Breadth First Search (BFS)

Now, Q = []

So, the algorithm terminates.

So, BFS traversal sequence is 0-1-3-8-7-2-4-5-6.

The BFS traversal sequence is not unique. Traversal sequence depend on the order that neighbor(s) are chosen.

Homework : Now, for the same undirected graph, start from node 2. Compute the BFS tree. Give BFS traversal sequence.

Breadth First Search (BFS)

```
// G(V, E) is given undirected graph represented by adjacency matrix
// visited [i] is a binary array of size V, initialized by 0, indicates if node i is visited.
// neighbor(i) indicates {set of nodes to which node i is connected} ;   Q[V] = [] ,  queue Q of size V is empty initially
BFS-traversal(s)
{     // s is start node
   visited[s] = 1   // Keep the status of the start node unchanged
   Q = {s}
   while Q is not empty
       v = dequeue(Q)  // dequeue a node, enqueue all its neighbors
       print(v)
       for each u in neighbor(v)      // for all adjacent vertices of vertex v
                if visited[u]  == 0
                     visited[u]  = 1  // Changed the status of each unmarked neighbor(s) of u , mark as visited
                      enqueue(Q, u) // each vertex enters Q exactly once
}
```

Breadth First Search (BFS)

Time complexity

Assuming that the graph is connected and represented by adjacency lists.

The operations of enqueuing and dequeuing take O(1) time, and so the total time devoted to queue operations is O(V). Because the procedure scans the adjacency list of each vertex only when the vertex is dequeued, it scans each adjacency list at most once. Since the sum of the lengths of all the adjacency lists is O(E), the total time spent in scanning adjacency lists is O(E). The overhead for initialization is O(V), and thus the total running time of the BFS procedure is O(V + E).