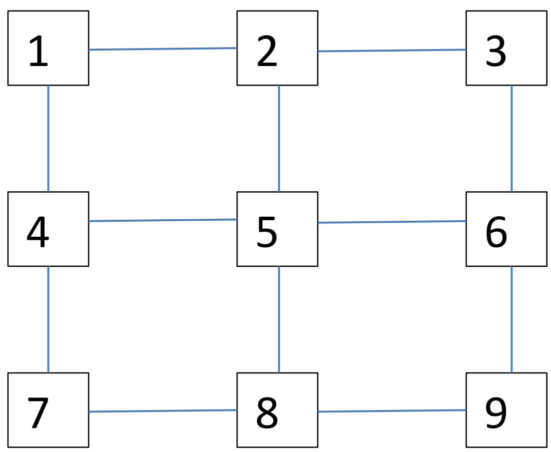
BFS

Breadth-First Search in Graphs:

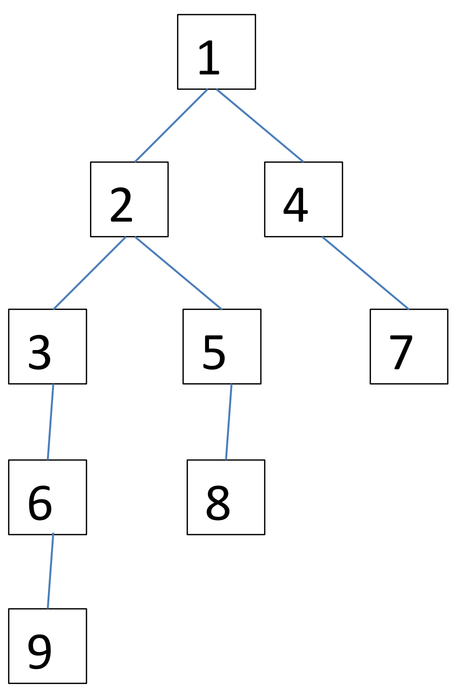
BFS is similar to level-order traversal of a tree

```
BFS (Graph G) {
	choose start vertex;
	Queue Q = new Queue( );
	Q.enqueue (start);
	boolean seen[1…n];
	for (k=1; k<=n; k++)  seen[k] = false;
	seen[start] = true;
	while (! Q.isEmpty( )) {
		x = Q.dequeue( );
		visit (x);
		for each vertex y such that (x,y) is an edge {
			// find each y by traversing adjacency list of x, or
			// by iterating across row x of adjacency matrix
			if (! seen[y]) {
				Q.enqueue (y);
				seen[y] = true;
				// optionally add edge (x,y) to the BFS tree;
			}
		}
	}
}
```

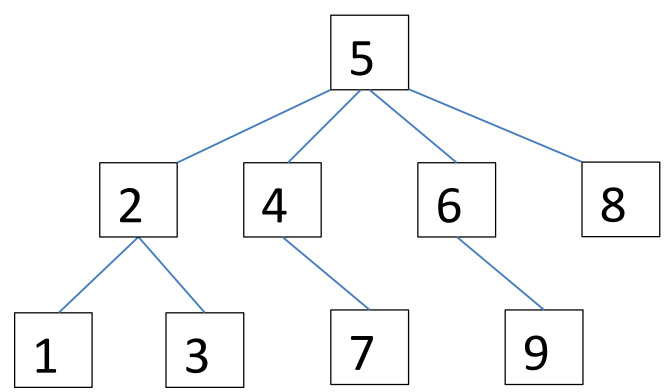
Example: Undirected Graph



BFS starting at vertex 1:



BFS starting at vertex 5:

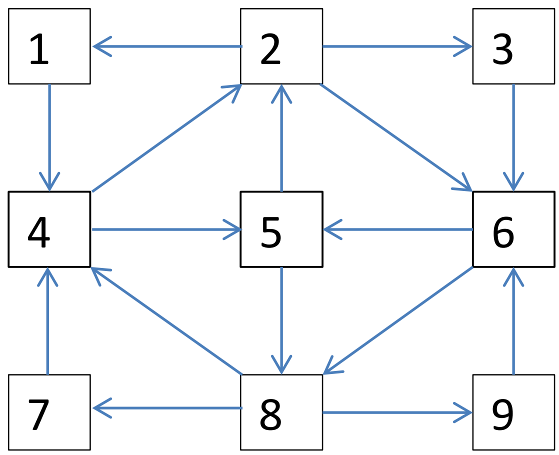


BFS order: 1,2,4,3,5,7,6,8,9

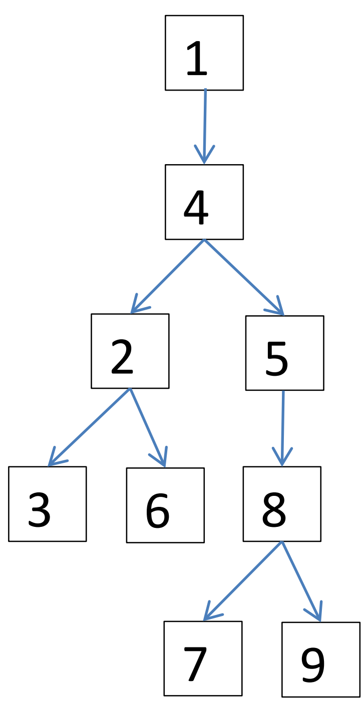
BFS order: 5,2,4,6,8,1,3,7,9

(BFS order is the same as level-order of the BFS tree)

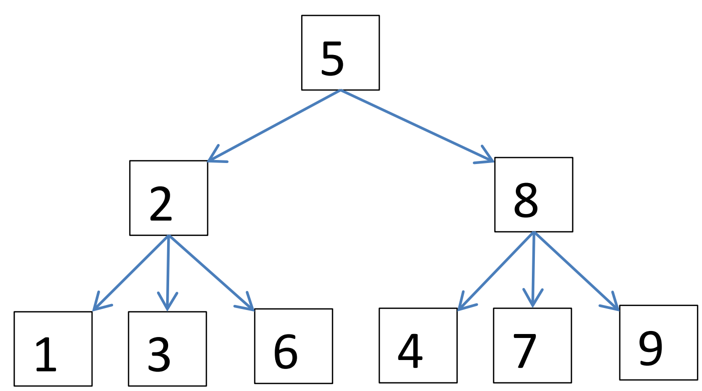
Example: Directed Graph



BFS starting at vertex 1:



BFS starting at vertex 5:



BFS order: 1,4,2,5,3,6,8,7,9

BFS order: 5,2,8,1,3,6,4,7,9

(BFS order is the same as level-order of the BFS tree)

Analysis of BFS:

If graph represented using adjacency matrix:

$\theta(n^2)$ time

If graph represented using adjacency lists:

$\theta(n+m)$ time

Simplifies to $\theta(m)$ time if graph is connected, because $m \geq n-1$ for connected graphs

Note: it is more efficient to use adjacency lists, because $m \leq n^2$ for all graphs

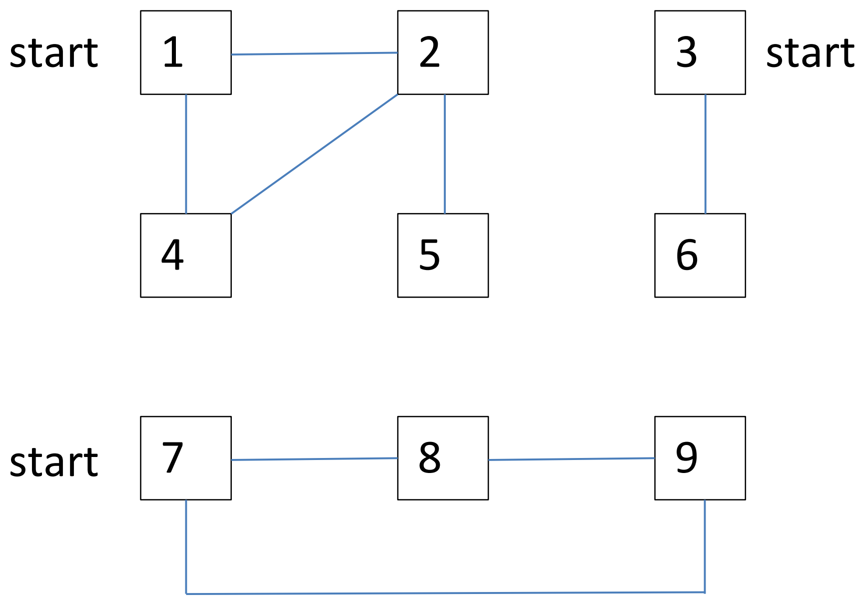
Applications of BFS:

- Shortest paths (fewest edges) from start vertex to each other vertex

- Connectedness of undirected graph:

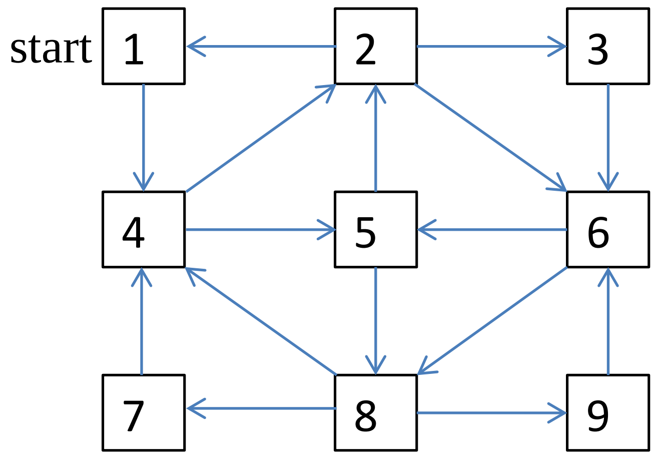
  Is the graph connected?

  If not, find all the connected components



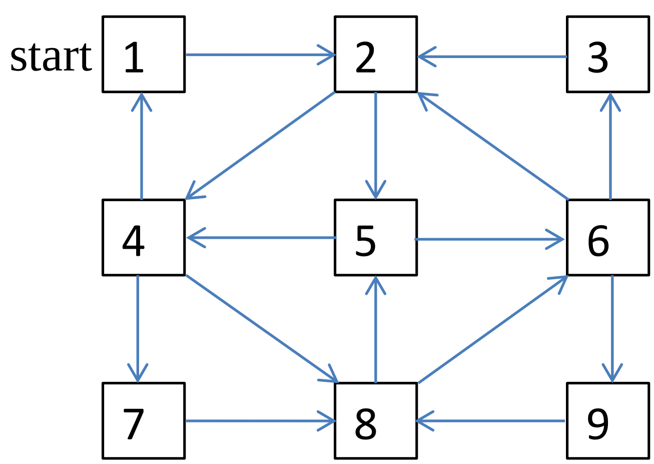
Components: {1,2,4,5} {3,6} {7,8,9}

- Strong connectedness of directed graph:
  For all x, y, do there exist paths from x to y and from y to x?



First do BFS from any chosen start vertex in given graph G.

Then construct the reverse of graph G to obtain G’. (How?)



Next do BFS in G’ using the same start vertex.

G (also G’) is strongly connected if and only if
both the BFS in G and the BFS in G’ reach every vertex;
that is, each BFS builds one BFS tree that has every vertex.