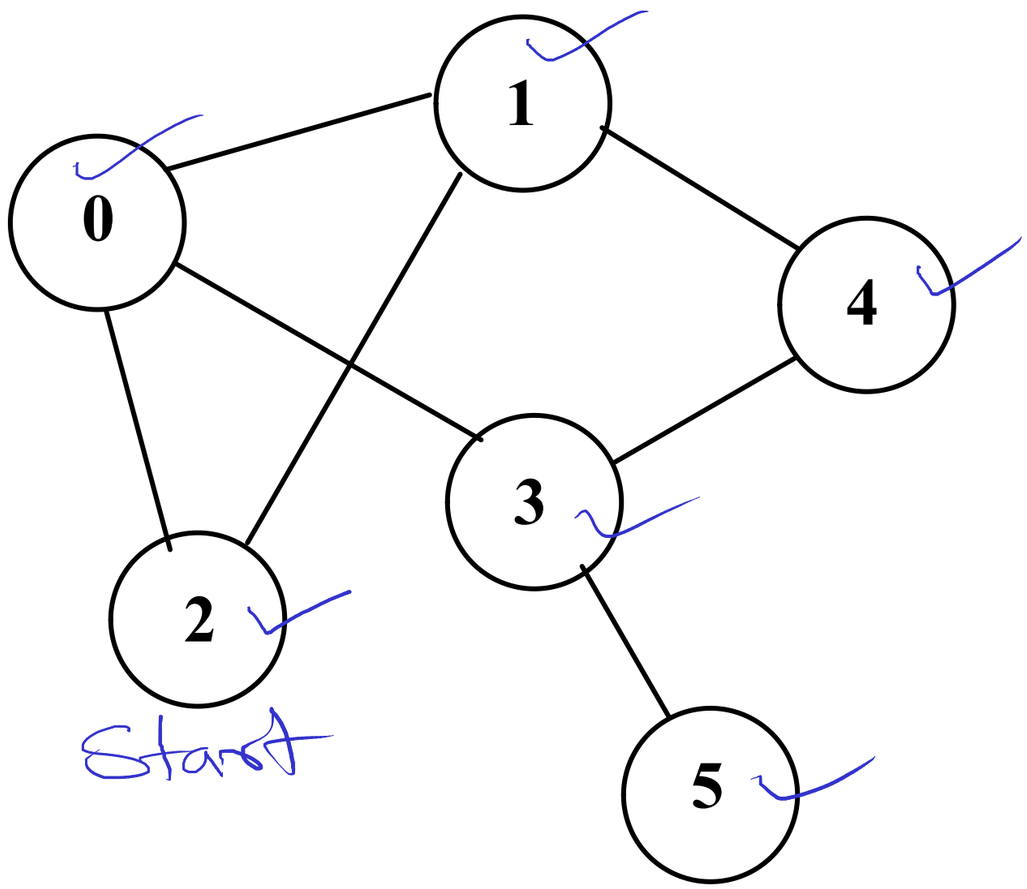
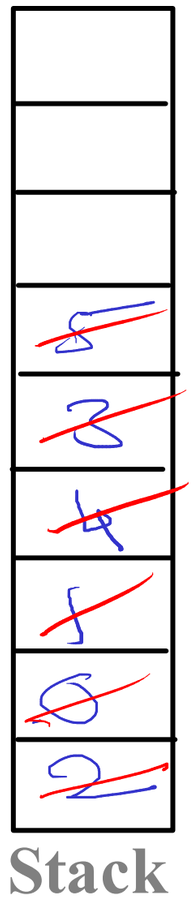
DFS Traversal

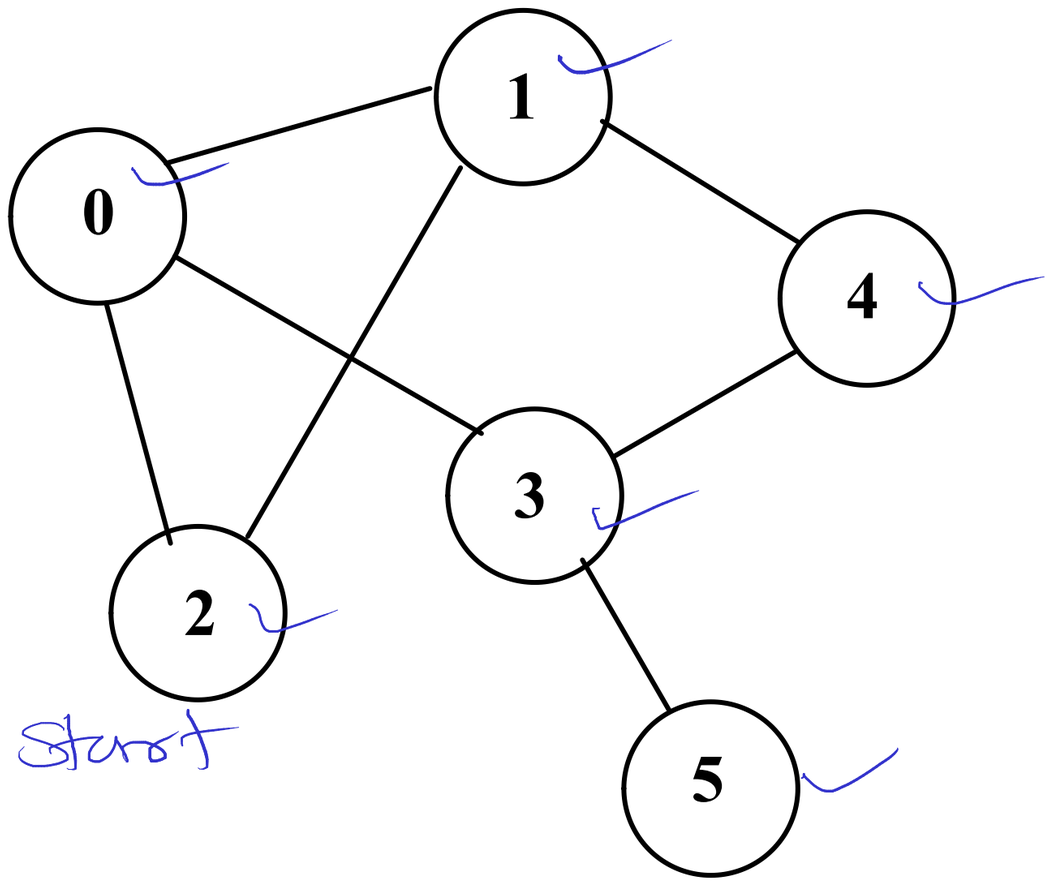


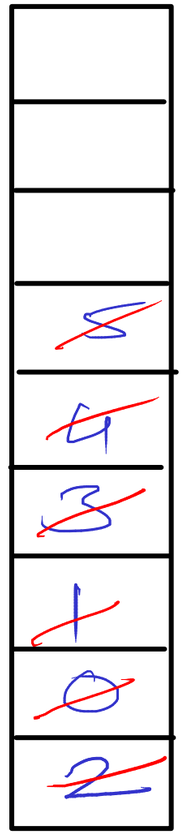


DFS: 2, 1, 4, 3, 5, 0

//1. Choose a vertex as start vertex.
//2. Push start vertex on stack & mark it.
//3. Pop vertex from stack.
//4. Print the vertex.
//5. Put all non-visited neighbours of the vertex
//on the stack and mark them.
//6. Repeat 3-5 until stack is empty.

BFS Traversal



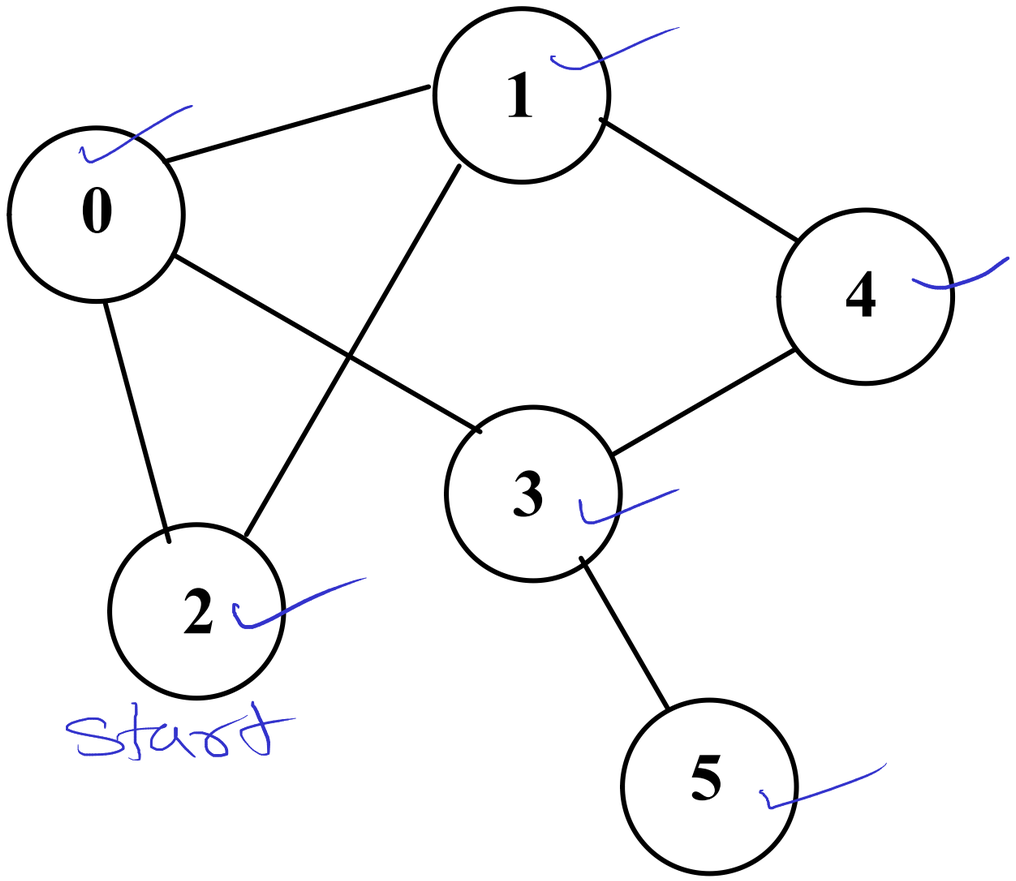


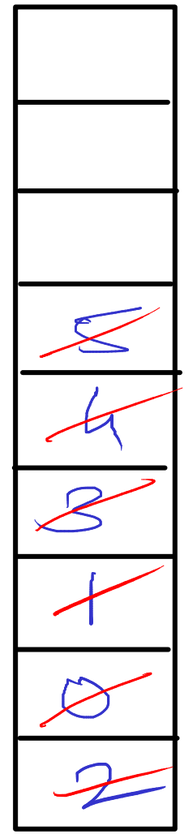
Queue

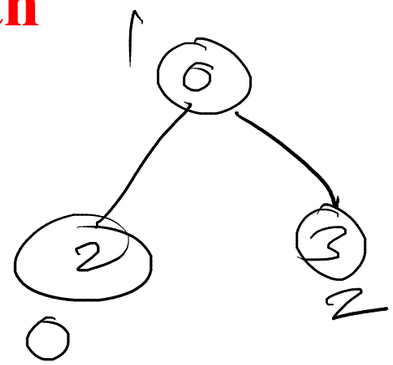
BFS: 2, 0, 1, 3, 4, 5

//1. Choose a vertex as start vertex.
//2. Push start vertex on queue & mark it
//3. Pop vertex from queue.
//4. Print the vertex.
//5. Put all non-visited neighbours of the vertex
//on the queue and mark them.
//6. Repeat 3-5 until queue is empty.

Single Source Path length



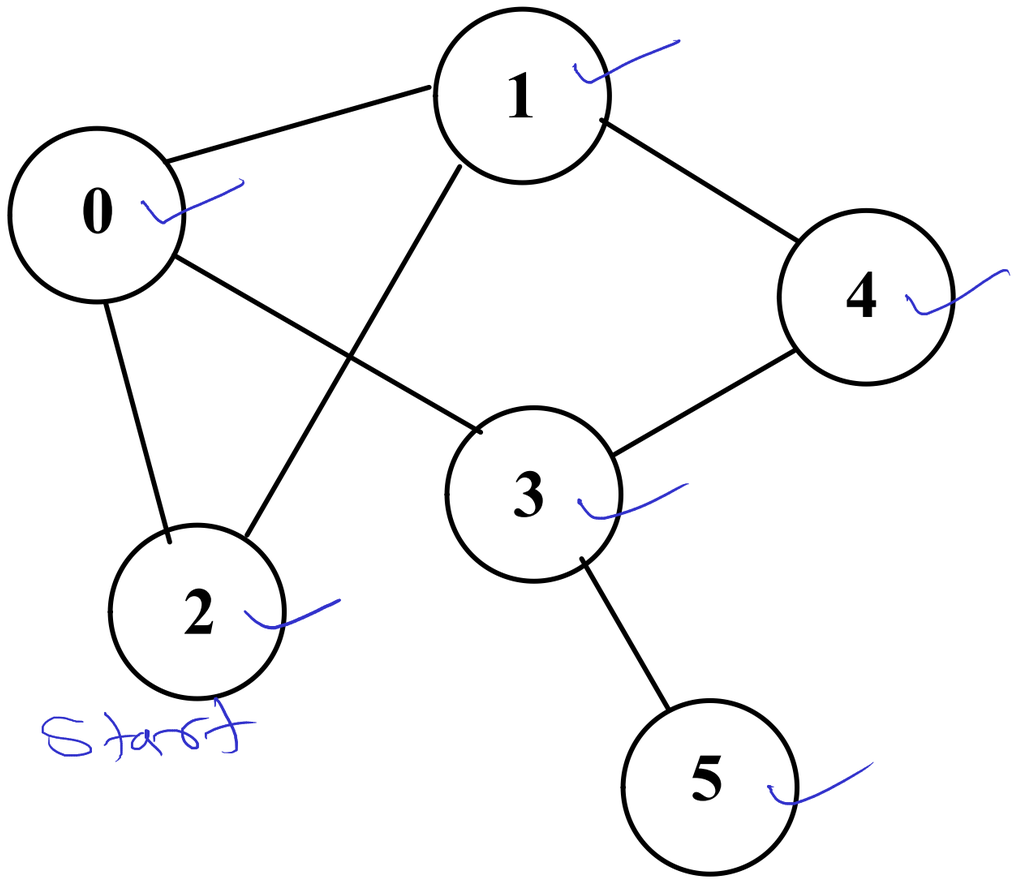


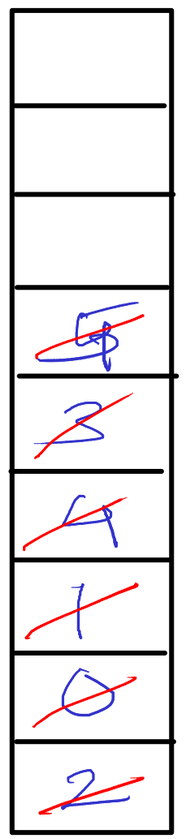


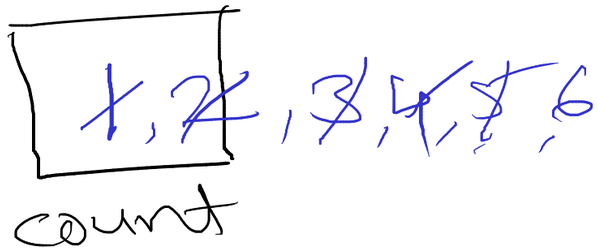
2 0 1 3

//1. Create path length array to keep distance of vertex from start vertex.
//2. push start on queue & mark it.
//3. pop the vertex.
//4. push all its non-marked neighbors on the queue, mark them.
//5. For each such vertex calculate distance as dist[neighbor] = dist[current] + 1
//6. print current vertex to that neighbor vertex edge.
//7. repeat steps 3-6 until queue is empty.
//8. Print path length array.

Check Connected-ness







2, 1, 4, 3, 5, 0

//1. push start on stack & mark it.
//2. begin counting marked vertices from 1.
//3. pop and print a vertex.
//4. push all its non-marked neighbors on the stack, mark them and increment count.
//5. if count is same as number of vertex, graph is connected (return).
//6. repeat steps 3-5 until stack is empty.
//7. graph is not connected (return)