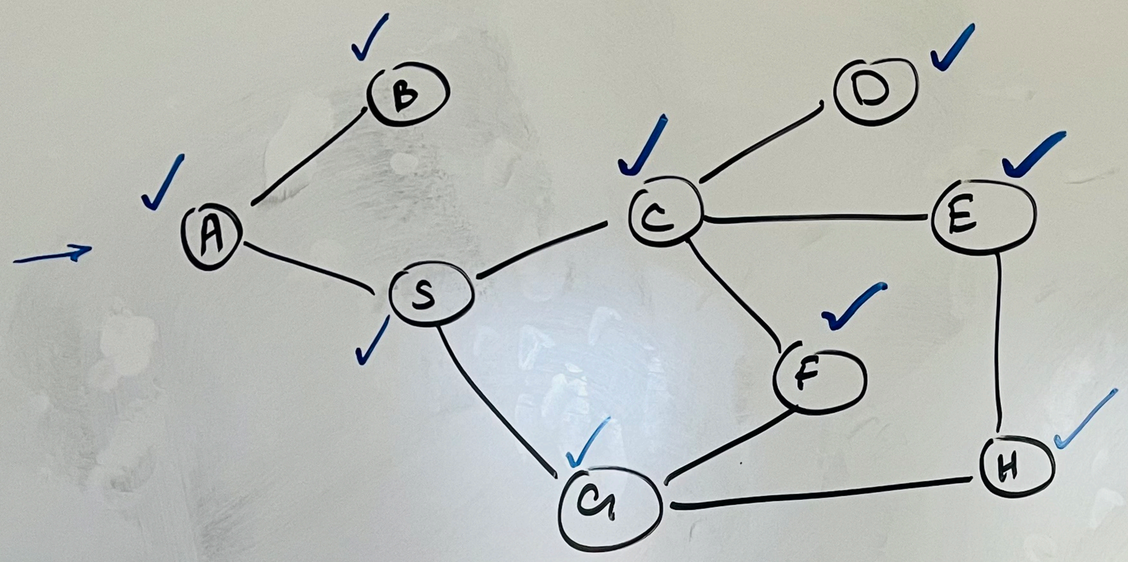
Breadth First Search (BFS) ⟶ Queue (FIFO)

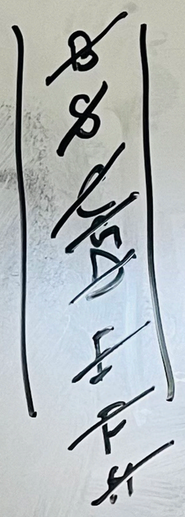


Queue Operations:

enqueue: insert

dequeue: remove

Queue Status



Output: A B S C G D E F H

Working node: ~~A~~ ~~B~~ ~~S~~ ~~C~~ ~~G~~ ~~D~~ ~~E~~ ~~F~~ H

Current working node is A. We will add it to the output sequence and mark it as visited. For BFS, we will check all the adjacent, unvisited nodes to A. We have B and S. We will enqueue them alphabetically.

We will enq
and mark
add it to th

Now, we ha
We will
B so we
any ad
the qu

We w
S ha
the n
outp

N
S
u

We will enque B first, add it to the output sequence and mark it as visited. Similarly, we will enque S, add it to the output sequence, and mark it as visited.

Now, we have to update the current working node. For that, we will check the first element of the queue. We have B so we will update it to B and dequeue it. Do we have any adjacent unvisited node to B? If no, we will update the current working node.

C G D E F H

~~B S C G D E F~~ We will update the current node to S and dequeue it. Now, S has C and G as adjacent unvisited nodes. So will enqueue them in alphabetically. So, we will enqueue C first, add it to the output sequence, and mark it as visited.

Now, will enqueue G, add it to the output, and mark it as visited. Since all of S's adjacent unvisited nodes have been explored, we will update the current working node to the first element of the queue. So we will update it to C and dequeue it.

Now, C h
We will en
"
"
"
We wi
then
and
Nou
"

Now, C has D, E, and F as adjacent, unvisited nodes.

We will enqueue D, add it to the output, and mark it as visited.

			E								
			F								

We will update the working node. We will update it to G and then dequeue it. So we will enqueue H, add it to output, and mark it as visited.

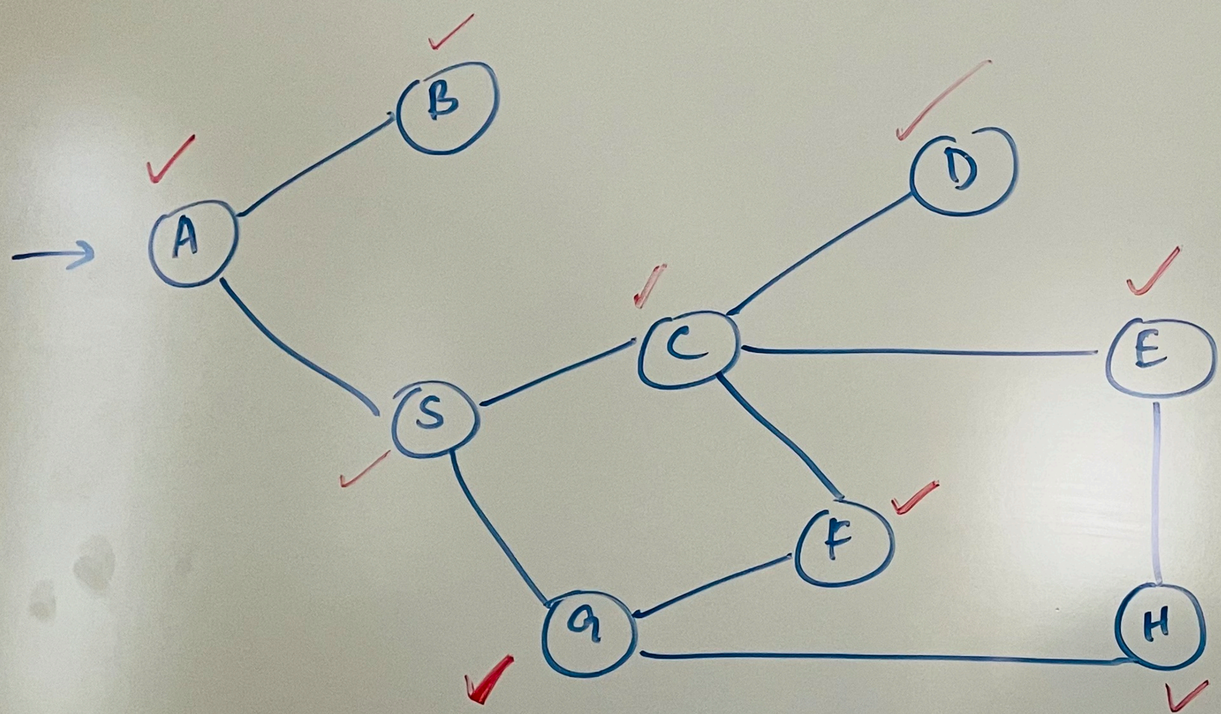
Now G is finished. We will update working node to D and dequeue it.

	D								E		
	E								F		
	F								H		

Now our queue is empty. The algorithm terminates.

late

Breadth First Search (BFS) — Queue (FIFO)

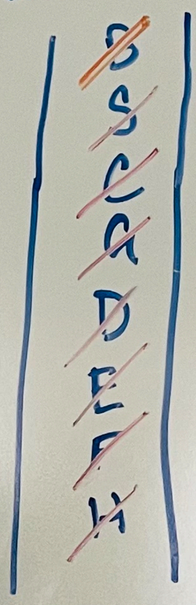


Queue operations:

enqueue : inserting

dequeue : removing

Queue Status



Output : A B S C G D E F H

working node : ~~A~~ ~~B~~ ~~S~~ ~~C~~ ~~G~~

~~D~~ ~~E~~ ~~F~~ H

Status

B S C G D E F H

: ~~A~~ ~~B~~ ~~S~~ ~~C~~ ~~G~~

~~D~~ ~~E~~ ~~F~~ H

Currently working node is A.
We will add A to the output, and mark it as visited. For BFS, we have to check all the adjacent, unvisited nodes of the current working node. We have B and S. We will enqueue B and S alphabetically.

We will enqueue B first, add it to the output and mark it as visited. Then we will enqueue S, add it to the output, and mark it as visited.

Now, we have to update the current working node. For that we will check the first element of the queue. We have B so we will update the working node to B, and dequeue. Does B have any adjacent, unvisited node? No. That's why we will update the working node.

We will update the current working node to S.
and dequeue S. Now, S has C and G as
adjacent, unvisited nodes. We will enqueue C and
G alphabetically. We will add C first, add it
to output, and mark it as visited.

Then we will enqueue G, add it to the output,
and mark it as visited. Since all of S's adjacent
unvisited nodes has been explored, we will update
the working node (pointer to C, and dequeue it.

Now C has D, E, F as adjacent unvisited nodes, so we
will enqueue them alphabetically.

We will enqueue D, add it to the output, and mark it as visited.
" " " E, " " " " , " " " " " .
" " " F, " " " " " , " " " " " .

We will update working node to G and dequeue G.

So, we will enque H, add it to the output. And mark it as visited.

Now, we will update working node to D, and dequeue it.

" " " " " E, " " ".

" " " " " F " " ".

" " " " " H, " " ".

H doesn't have any adjacent, unvisited node. And our queue is empty. So, the algorithm terminates

BFS Flowchart

① Set starting node to working node.
add it to output
mark it as visited.

② Explore all adjacent, unvisited nodes of working node.
Enqueue them alphabetically.
After we enqueue each time, we add it to output, and mark as visited.

③ When adjacent nodes of the working node is explored, we update the working node with the first element of the queue. After updating the working node, dequeue the node from the queue.

Have we explored all adjacent unvisited node of the working node?

Yes

No

④ When all elements in the queue has been dequeued, the algorithm terminates.

BFS Flowchart

① See starting node to working node
add it to output
mark it as visited.

② Explore all adjacent, unvisited nodes of working node.
Enqueue them alphabetically.
After each enqueue, add to output, and mark as visited.

③ When adjacent nodes of the working node is explored,
update the working node with the first element of the queue,
& dequeue the node from the queue.

Have we explored all adjacent unvisited node of the working node?

Yes

NO

④ When all elements of the queue has been dequeued, the algo terminates.