# Graph Traversals.

→ It is a systematic way of visiting the nodes in a specific order.

→ Two types
- Breadth first Traversals
- Depth first "

## Breadth first Traversals

* starts from unvisited node 'u', then unvisited vertex w of adjacent 'u'.

$$u \rightarrow v_i \rightarrow w_j.$$

$v_i$ adjacent of $u$     adjacent of $v_i$

* It uses queue data structure.

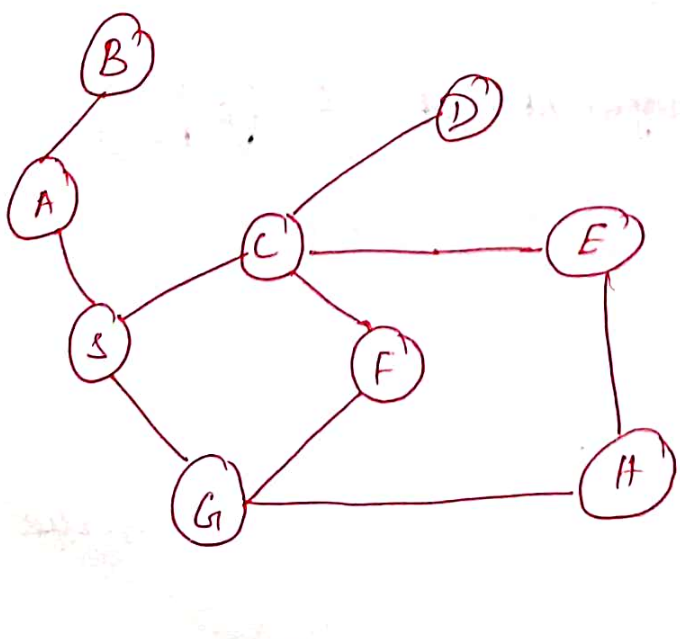
steps:

* choose source node.
* Use adjacency matrix to find all unvisited nodes & enqueue if its not visited, already visited node should be dequeued.
* Repeat step 2.

Algorithm..

```
void Bfs (Vertex u)
{ initialize queue Q;
  visited [u] = 1;
  Enqueue (u, Q);
  while (! is empty (Q))
  {
    u = Dequeue (Q);
    print u;
    for all vertices v adjacent to u do
     if (visited [v] == 0 ) then
      {
        Enqueue (v, Q);
        visited [v] = 1;
      }
   }
}
```

Eg:



| | S | A | B | C | D | E | F | G | H |
|---|---|---|---|---|---|---|---|---|---|
| S | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 1 | 0 |
| A | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| B | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| C | 1 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 0 |
| D | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| E | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| F | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 |
| G | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 |
| H | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 |

Procedure:

① Source Node 's'

Adjacent of S = A, C, G

① Enqueue (S) | S | | |

Dequeue (S)

② Enqueue adjacent of S | A | C | G |

Mark visited of $\boxed{S = 1}$

② Source 'A'.

Already Enqueued, so dequeue 'A'.

| C | G. | |

adjacent of 'A' : B so enqueue

| C | G | B |
|---|---|---|

A is visited $\boxed{A=1}$

(3) Source 'C'.

* Dequeue 'C'

| G | B |
|---|---|

* 'C' adjacent - D, S, E, F
already visited

| G | B | D | E | F |
|---|---|---|---|---|

C is visited $\boxed{C=1}$

(4) Source 'G'

→ Already enqueued, so dequeue 'G'

| B | D | E | F |
|---|---|---|---|

→ Adjacents of 'G' : S, F, H
already in queue
already visited

| B | D | E | F | H |
|---|---|---|---|---|

visited [G] = 1

Source 'B'

Dequeue 'B' | D | E | F | H |

Adjacent of 'B' is 'A'
↳ Already visited

B is visited $\boxed{B=1}$

Source 'D'

Dequeue 'D' | E | F | H |

Adjacent of D is 'C'
↳ Already visited

'D' is visited $\boxed{D=1}$

Source 'E'

Dequeue 'E' | F | H |

Adjacent of 'E' is C, H → Already in queue
↑
Already visited

'E' is visited $\boxed{E=1}$

Source 'F'

Dequeue 'F' | H |

adjacent of 'F' is C, G → Already visited

'F' is visited $\boxed{F=1}$

Source 'H'

Dequeue 'H'

Adjacent of H is $\overbrace{G, E}^{\text{Already visited}}$

H is visited $\boxed{H = 1}$