Graph Traversals

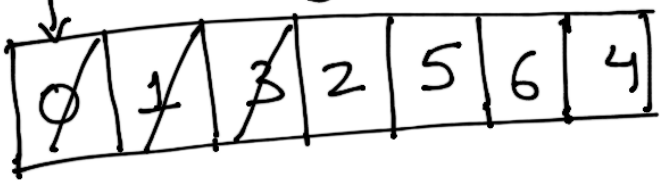
→ BFS (or level order traversal)

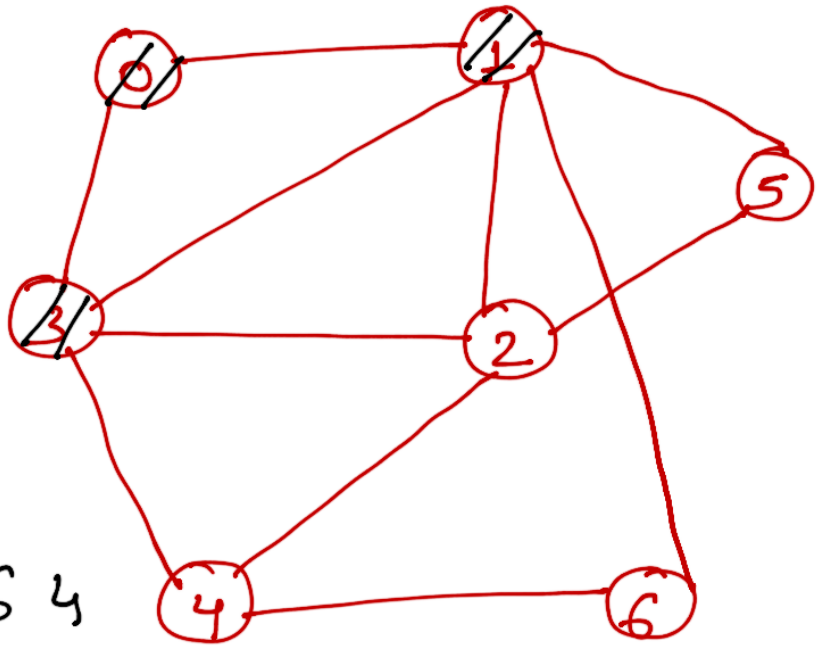
→ DFS

BFS

↳ Queue is used

Taking '0' as starting node



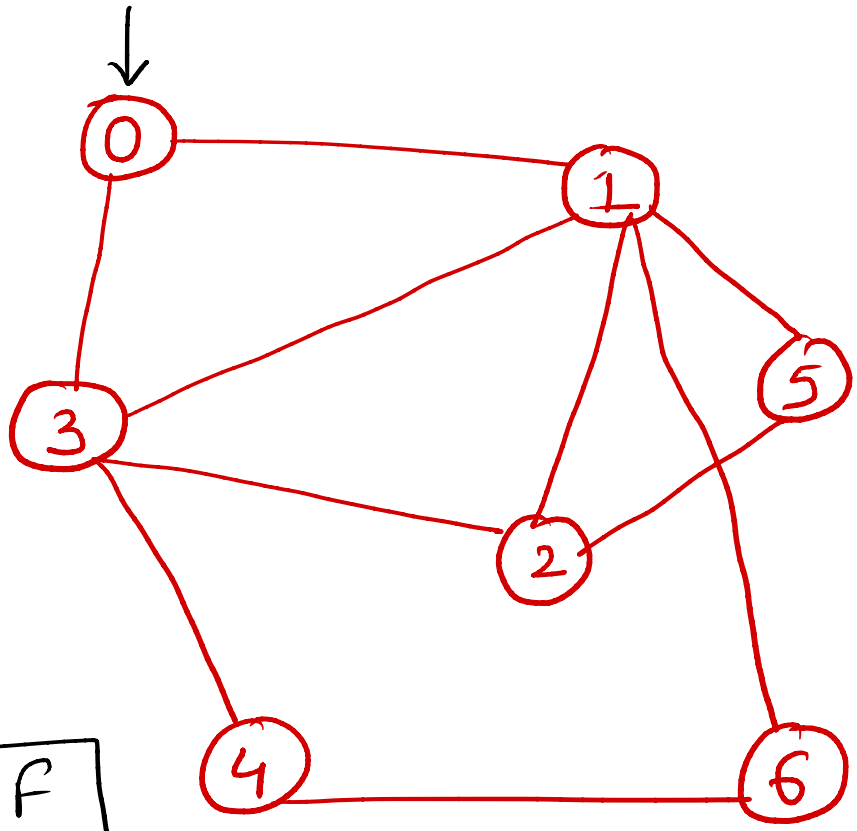


Result → 0 1 3 2 5 6 4

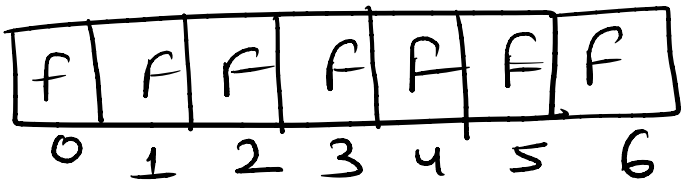
One possible BFS traversal

BFS (or level order traversal)

→ Take start vertex as '0'.

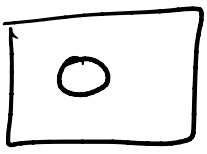


Queue Q

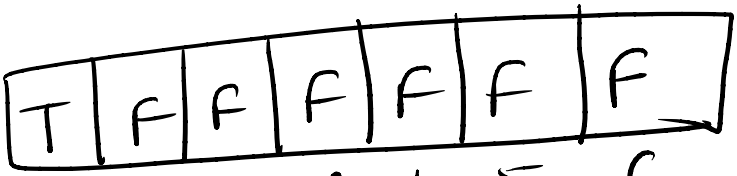


visited

↓ Dequeue

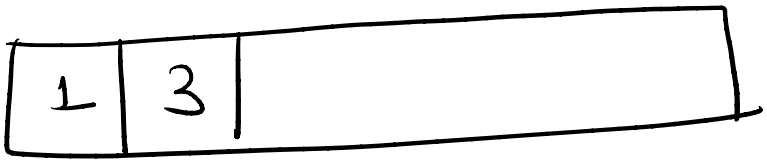
i) v 

(iii) Adj. u of '0' = {1, 3}

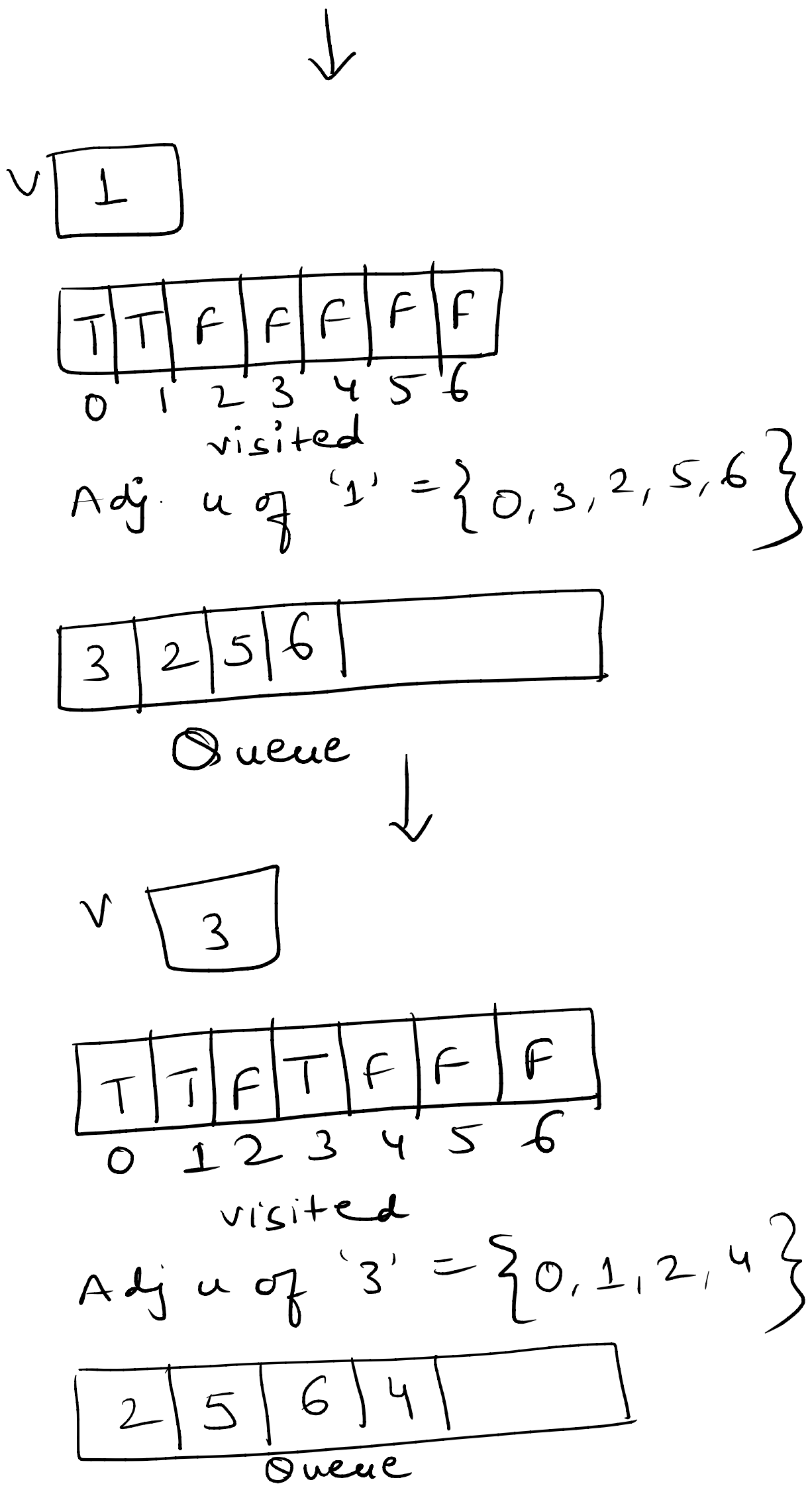
ii) 

visited

If u NOT visited & NOT in the Queue already

(iv) 

Queue


v
1
T T F F F F F
0 1 2 3 4 5 6
visited
Adj. u of '1' = {0, 3, 2, 5, 6}
3 2 5 6
Queue
v
3
T T F T F F F
0 1 2 3 4 5 6
visited
Adj u of '3' = {0, 1, 2, 4}
2 5 6 4
Queue

v | 2 |

T	T	T	T	F	F	F
0	1	2	3	4	5	6

visited

Adj u of '2' = { 3, 1, 5 }

5	6	4		

Queue

v | 5 |

T	T	T	T	F	T	F
0	1	2	3	4	5	6

visited

Adj u of '5' = { 1, 2 }

6	4		

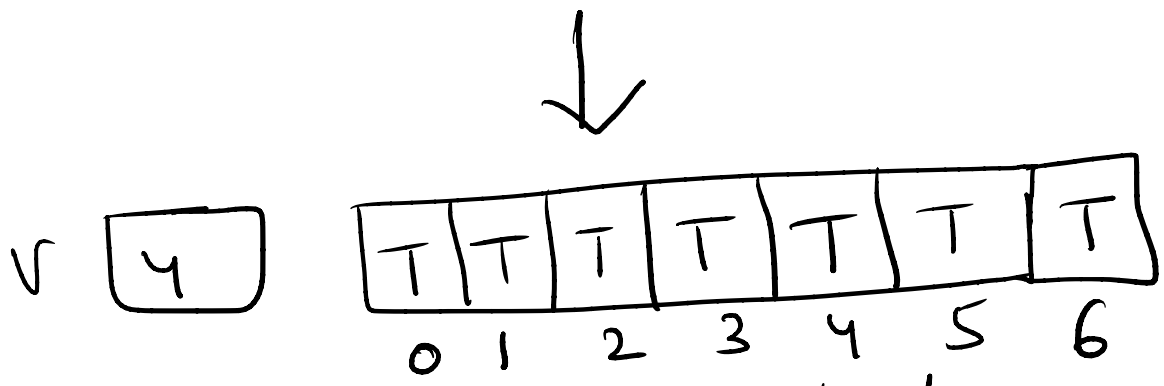
Queue

v | 6 |

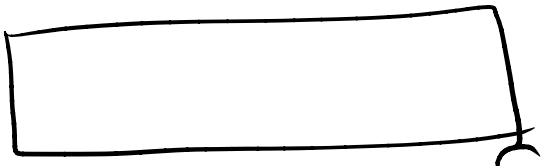
T	T	T	T	F	T	T
0	1	2	3	4	5	6

visited

Adj of 6 = { 4, 1 }



Adj of 4 = { 3, 6 }

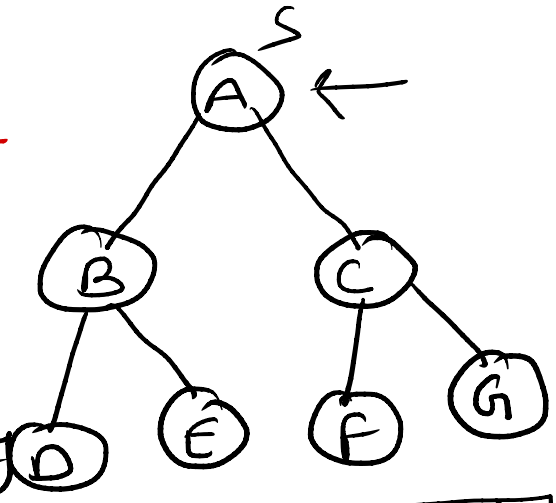


Queue is Empty

Result: 0, 1, 3, 2, 5, 6, 4

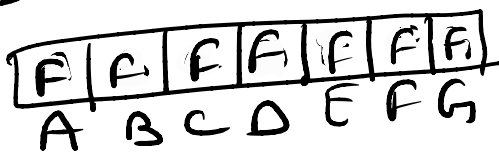
One possible BFS traversal

Pseudocode for BFS



BFS (G, S)

{ Initialize visited Array — O(1)



q.enqueue(s); — O(1)

while (!q.empty())

{ v = q.dequeue(); print 'v' — O(1)

visited [v] = true; — O(1)

for all u adjacent to v

{ if (u is not visited & u not already in queue)

q.enqueue(u);

}

}

$O \to c + E_0$

$1 \to c + E_1$

$2 \to c + E_2$

$V \to c + E_V$

$Vc + E_1 + E_2 + E_3 + \dots + E_V$

$= Vc + \sum E_i$

Time complexity: $O(V + E)$

Space Complexity: $O(V+E)$

visited Array = V

Queue = max V

Adj. List $O(V + E)$

$3V + E = O(V+E)$