Breadth-First Search

(BFS)

Joseph Chuang-Chieh Lin (林莊傑)

Department of Computer Science & Engineering,
National Taiwan Ocean University

Fall 2024


Outline

Outline

Breadth First Search (BFS) (1/2)

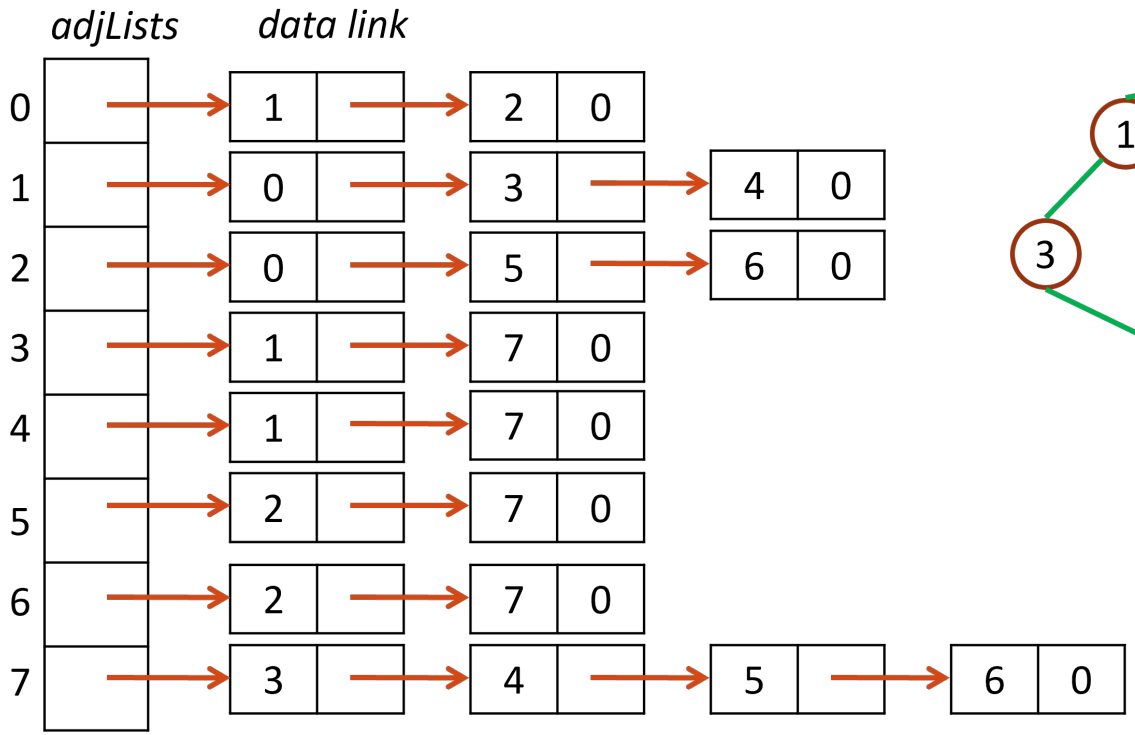
- The algorithm starts at vertex v and marks it as visited.
- Then visiting each of the vertices on v's adjacency list.
- When we have visited all the vertices on v's adjacency list, we visit all the unvisited vertices that are adjacent to the first vertex on v's adjacency list.
- To implement this scheme, as we visit each vertex we place the vertex in a queue.

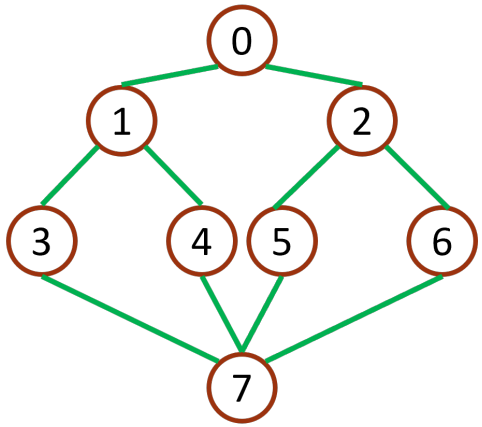
Breadth-First Search (BFS) (2/2)

- When we have exhausted an adjacency list, we remove a vertex from the queue and proceed by examining each of the vertices on its adjacency list.
- Unvisited vertices are visited and placed on the queue; visited are ignored.
- Finish the search when the queue is empty.

BFS Example

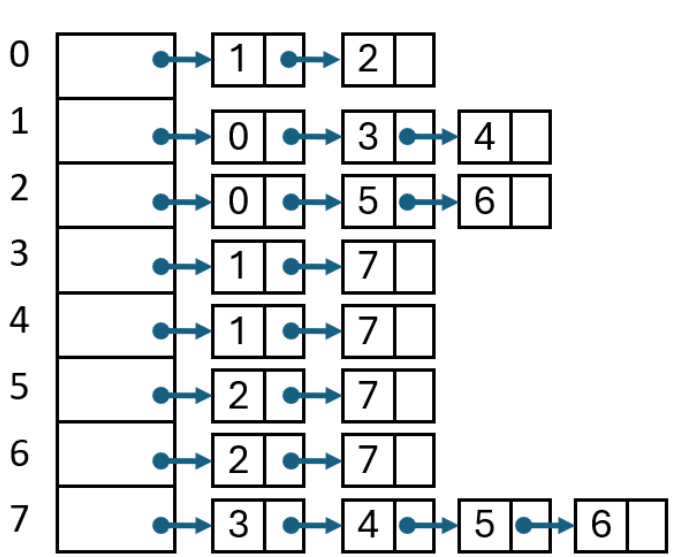
- Using a queue.
  - It resembles the level-order tree traversal.

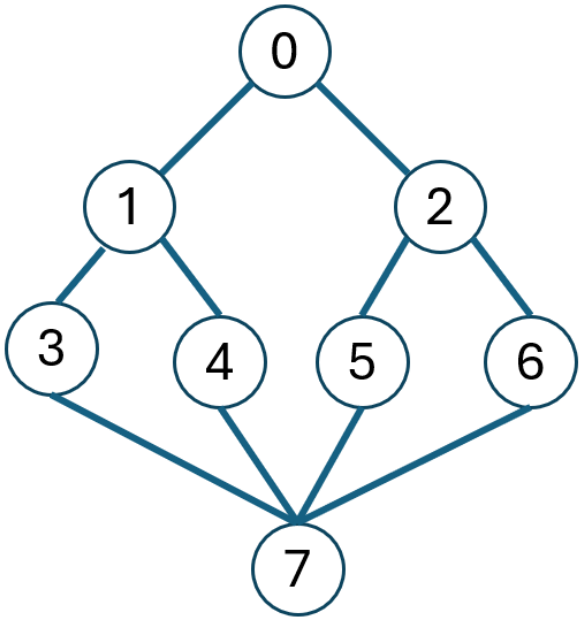




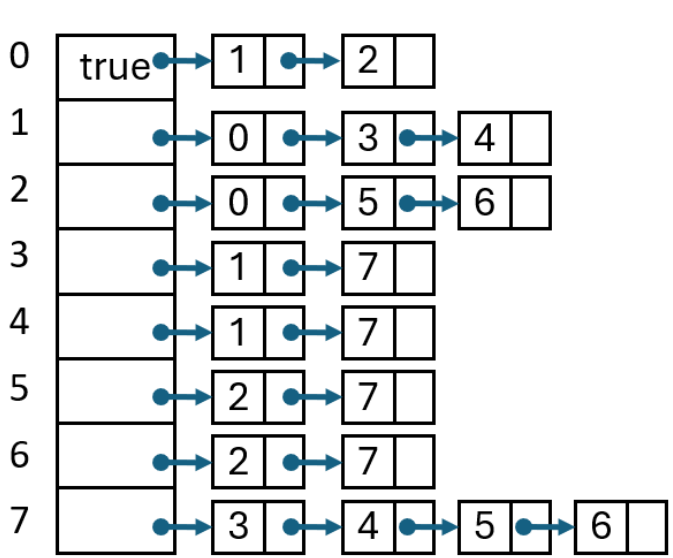
- The DFS order: $v_0 \rightarrow v_1 \rightarrow v_2 \rightarrow v_3 \rightarrow v_4 \rightarrow v_5 \rightarrow v_6 \rightarrow v_7$.

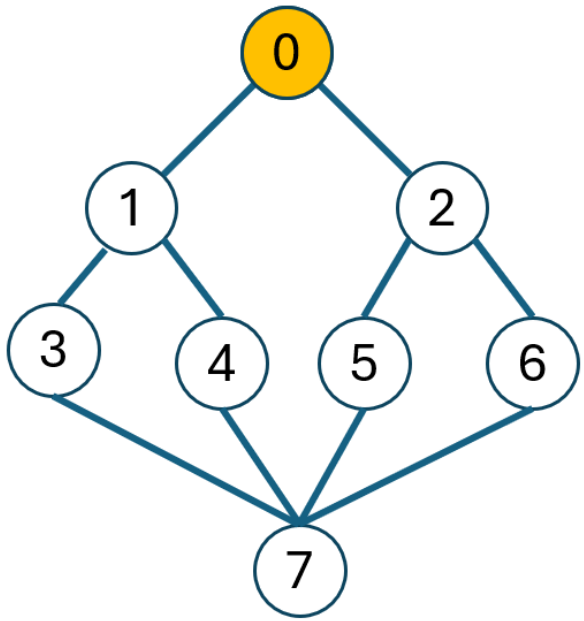
Example (Illustration)



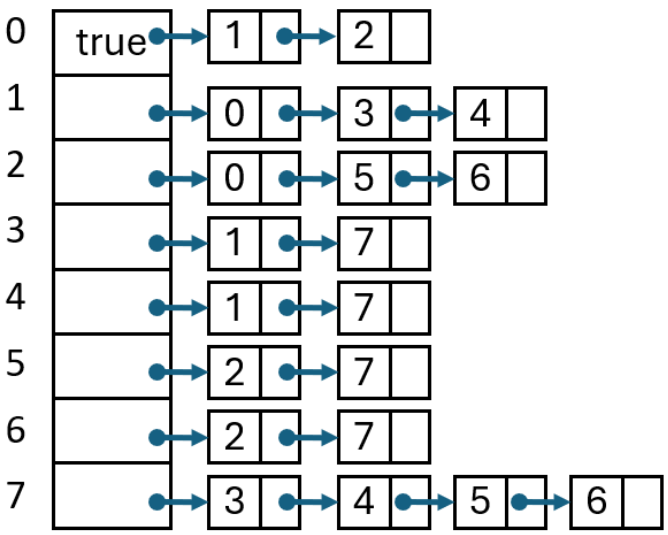


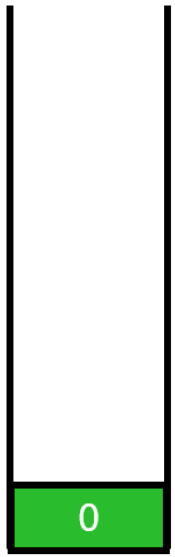
Example (Illustration)

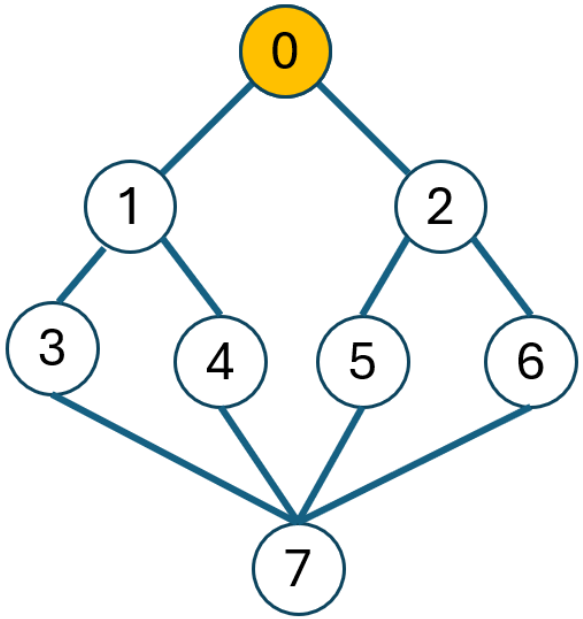




Example (Illustration)

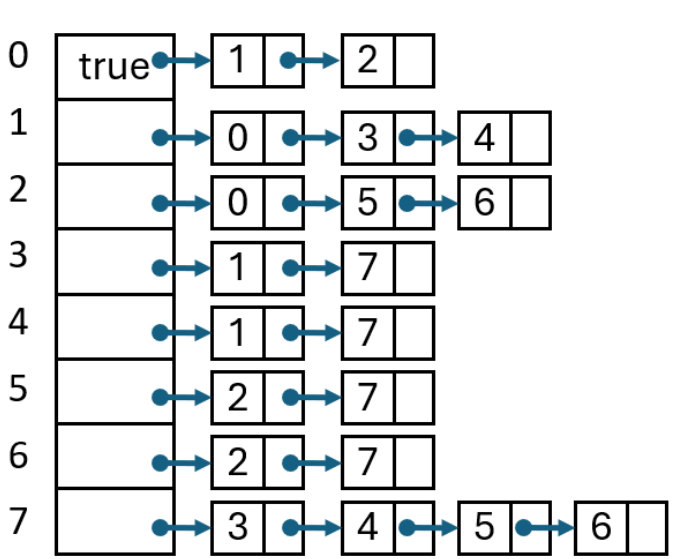


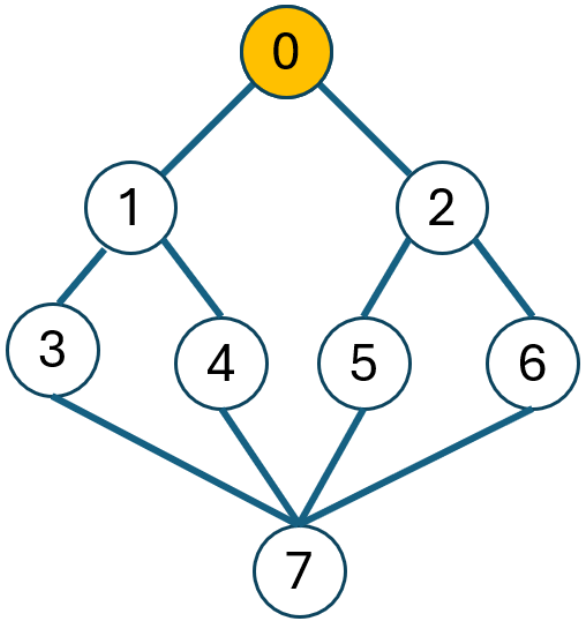




$0 \rightarrow$

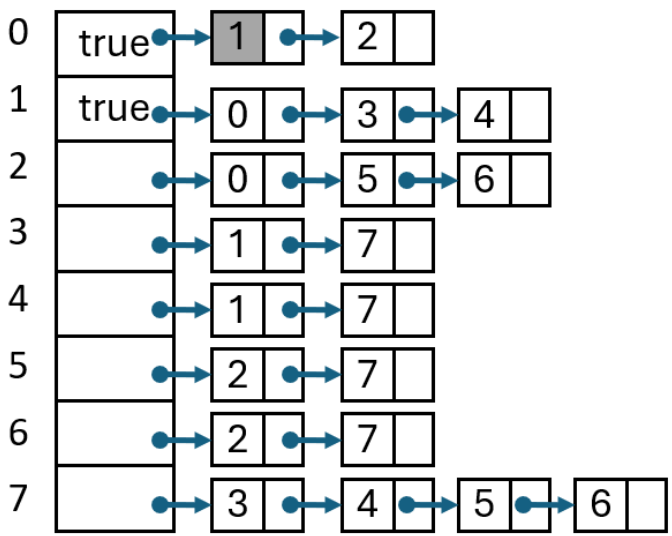
Example (Illustration)

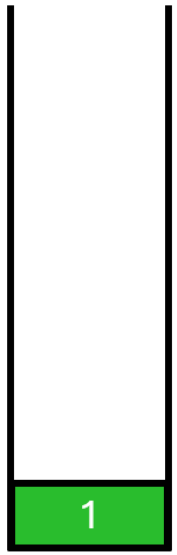


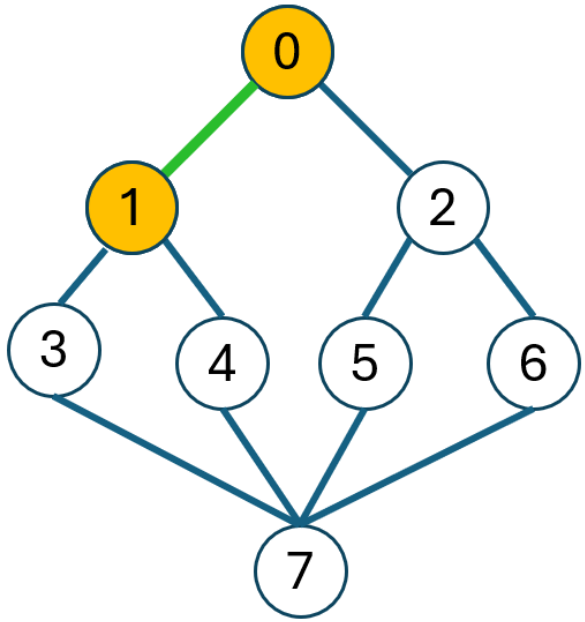


$0 \rightarrow$

Example (Illustration)

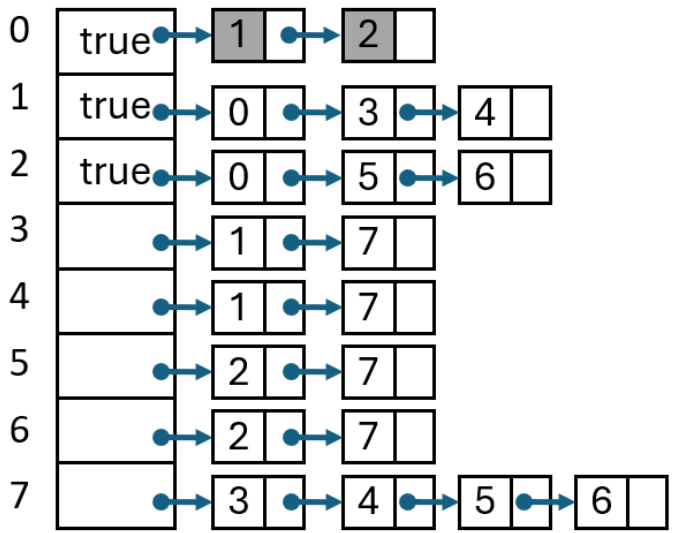


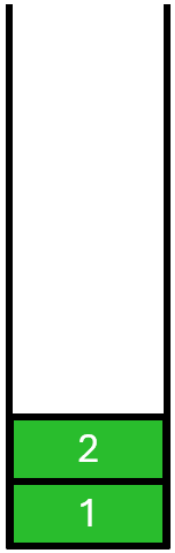


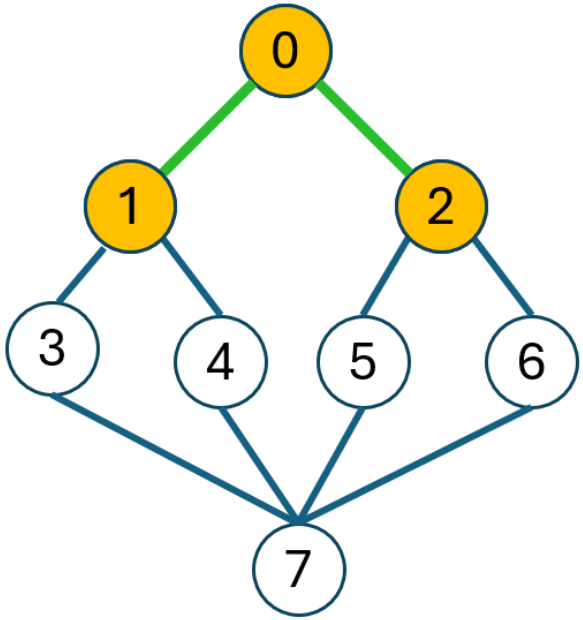


$0 \rightarrow 1 \rightarrow$

Example (Illustration)

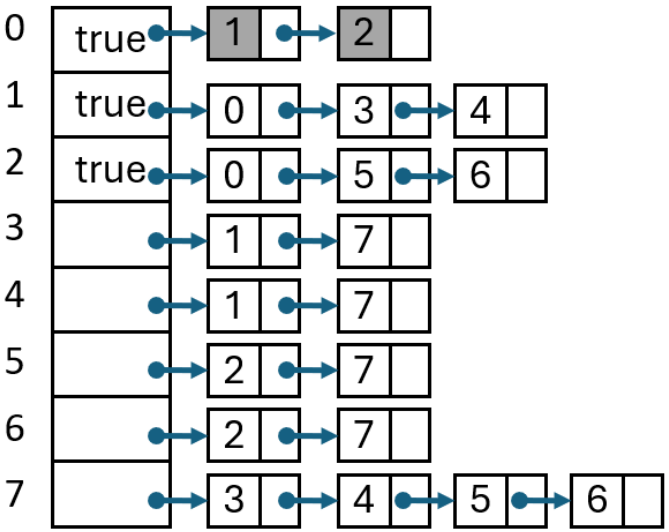


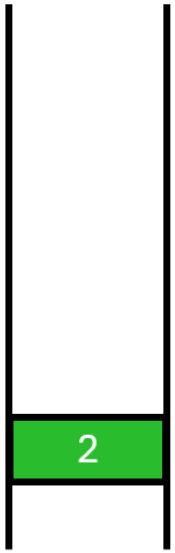


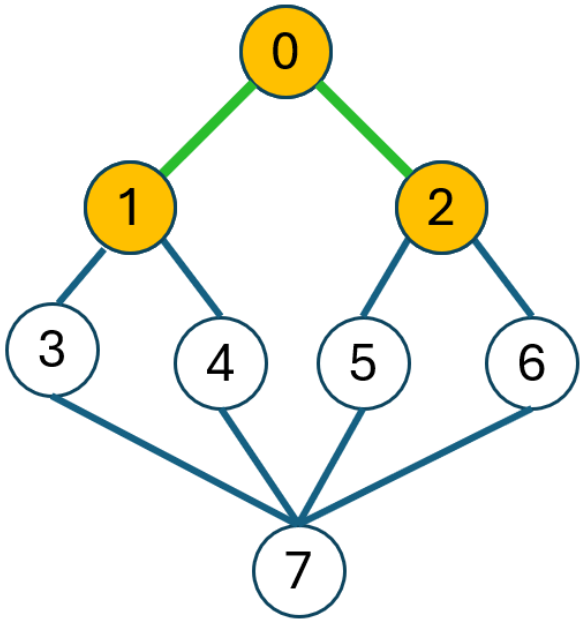


$0 \rightarrow 1 \rightarrow 2 \rightarrow$

Example (Illustration)

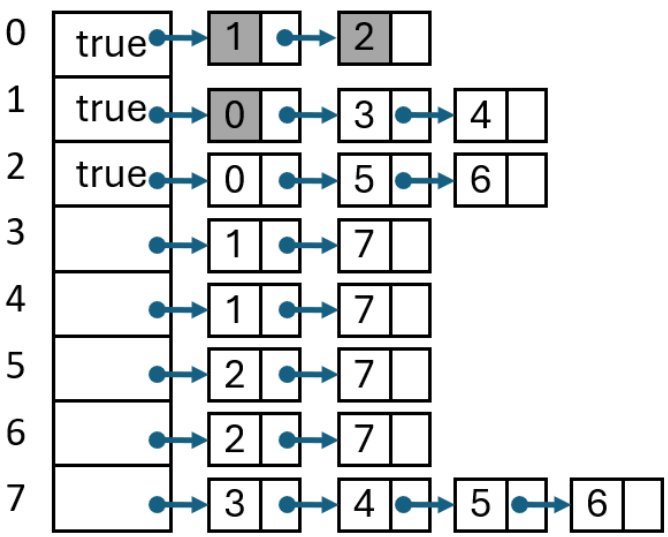


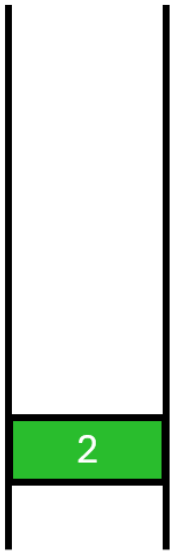


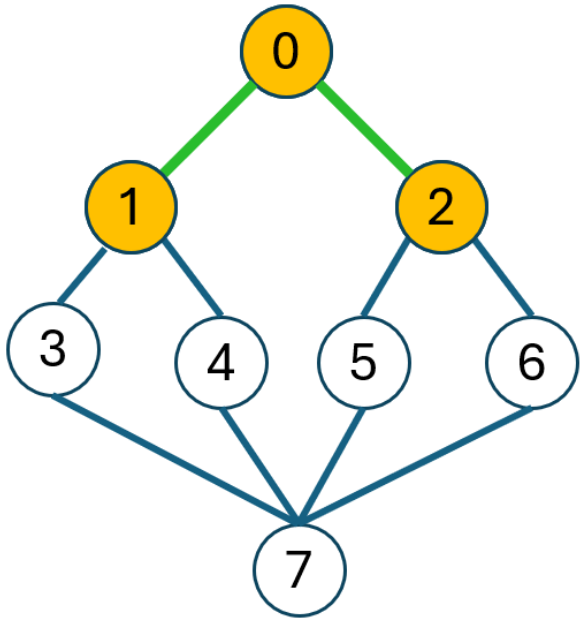


$0 \rightarrow 1 \rightarrow 2 \rightarrow$

Example (Illustration)

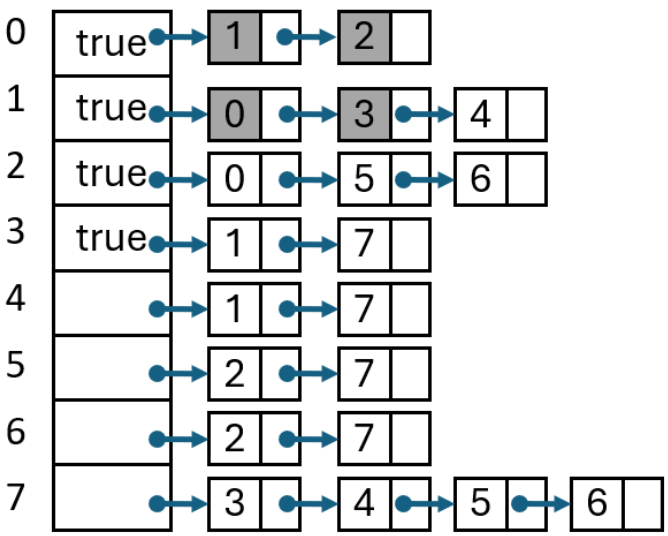


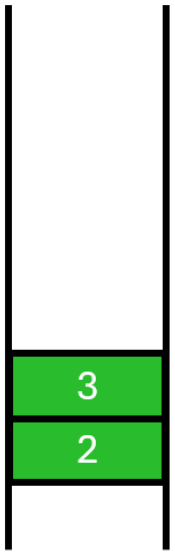


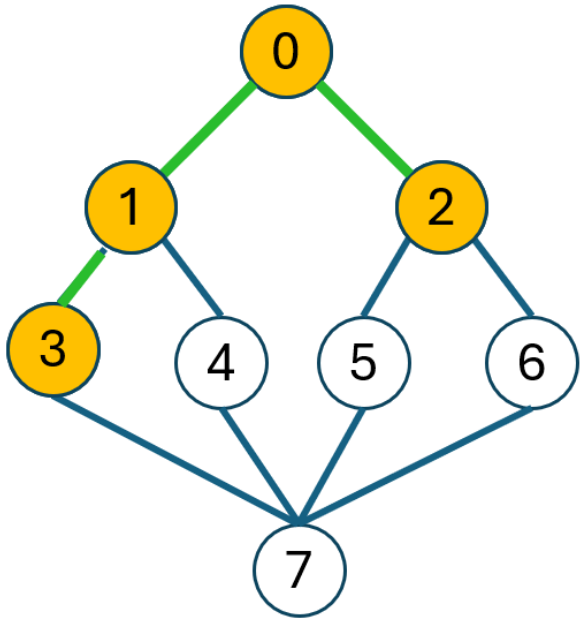


$0 \rightarrow 1 \rightarrow 2 \rightarrow$

Example (Illustration)

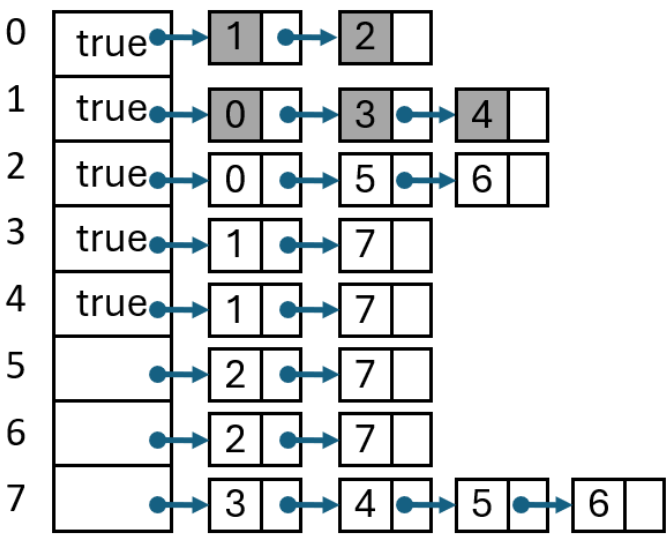


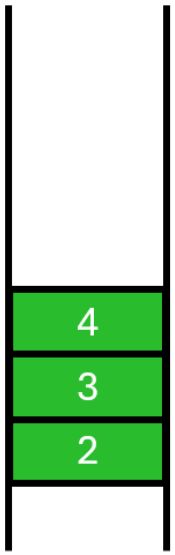


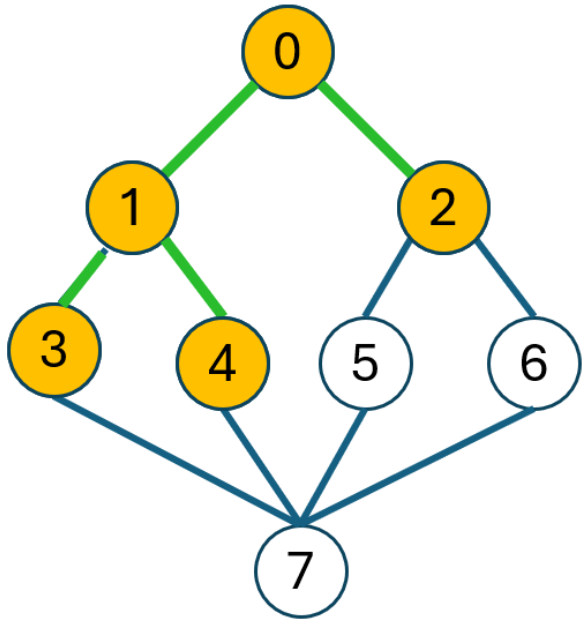


$0 \rightarrow 1 \rightarrow 2 \rightarrow 3 \rightarrow$

Example (Illustration)

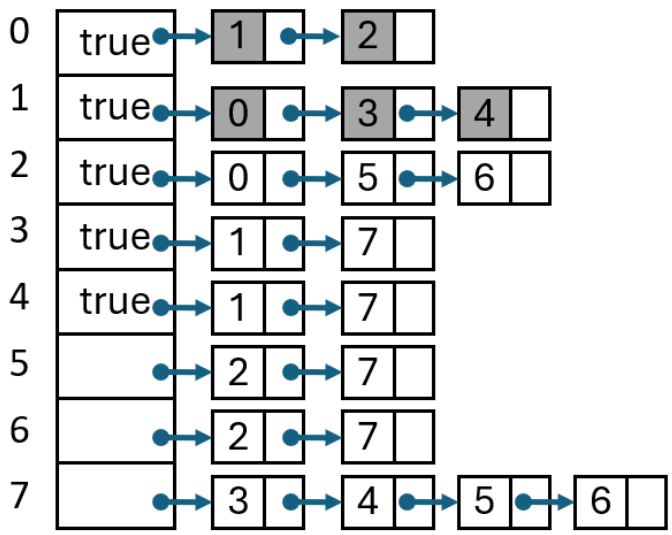


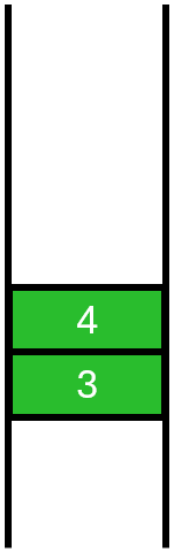


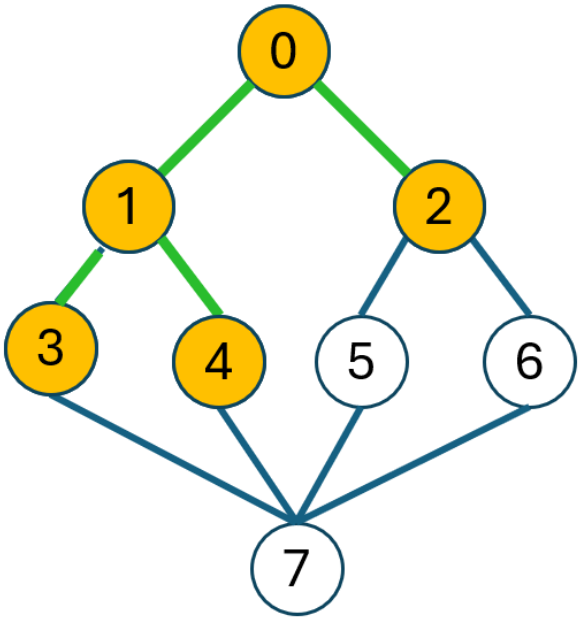


$0 \rightarrow 1 \rightarrow 2 \rightarrow 3 \rightarrow 4 \rightarrow$

Example (Illustration)

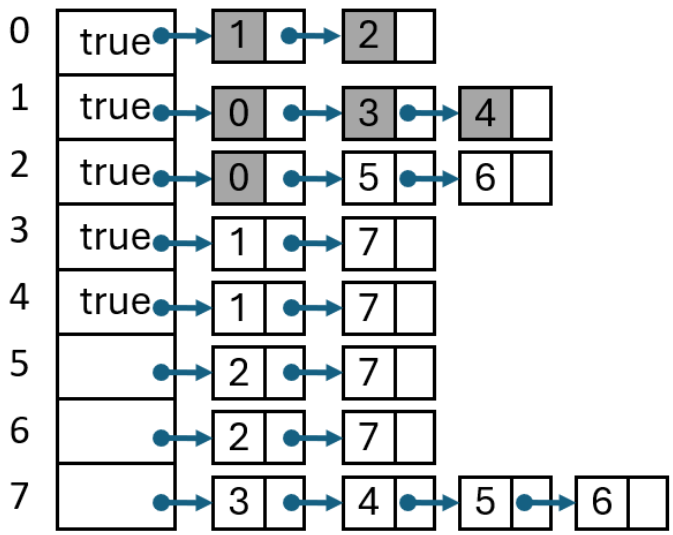


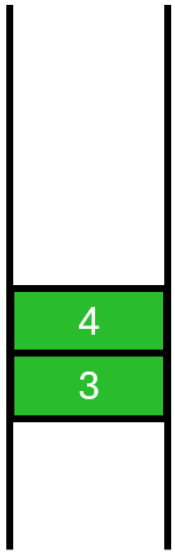


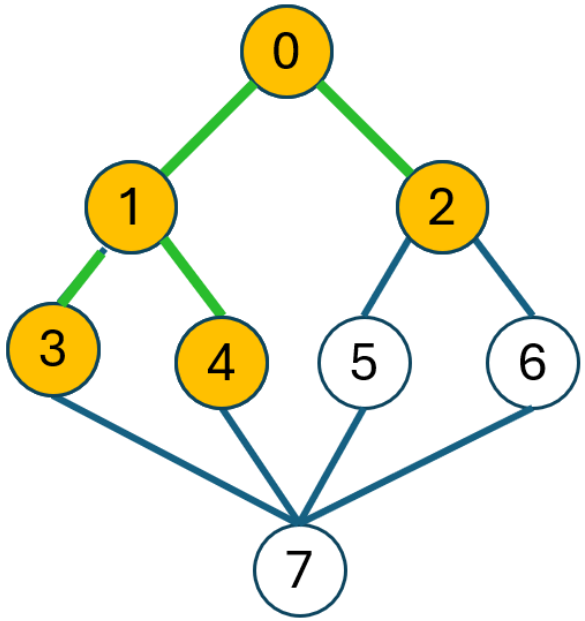


$0 \rightarrow 1 \rightarrow 2 \rightarrow 3 \rightarrow 4 \rightarrow$

Example (Illustration)

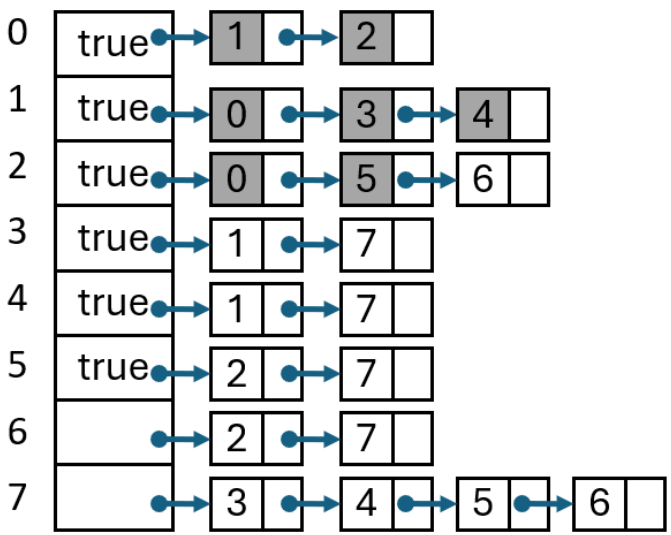


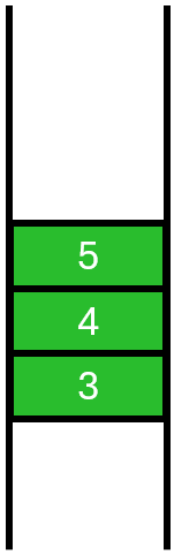


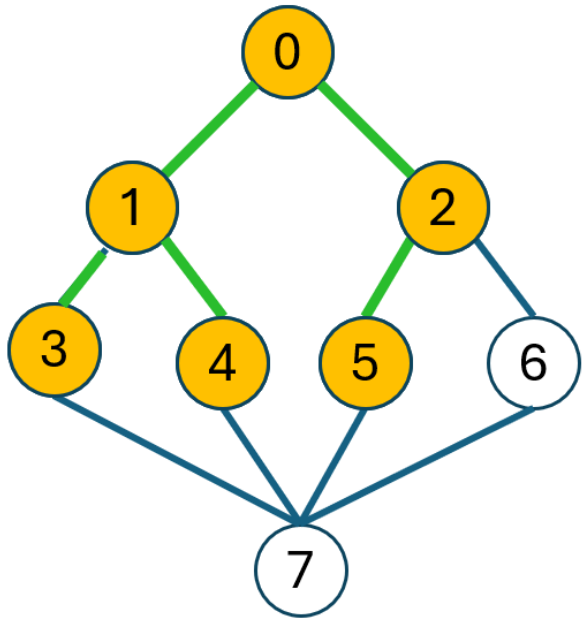


$0 \rightarrow 1 \rightarrow 2 \rightarrow 3 \rightarrow 4 \rightarrow$

Example (Illustration)

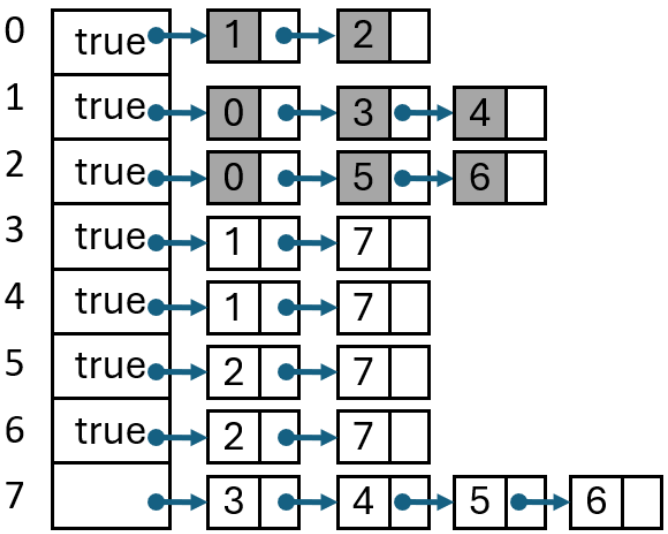


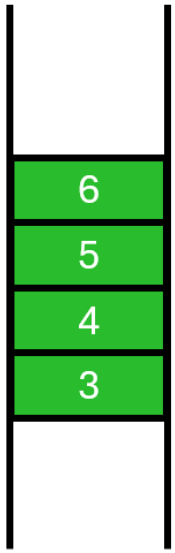


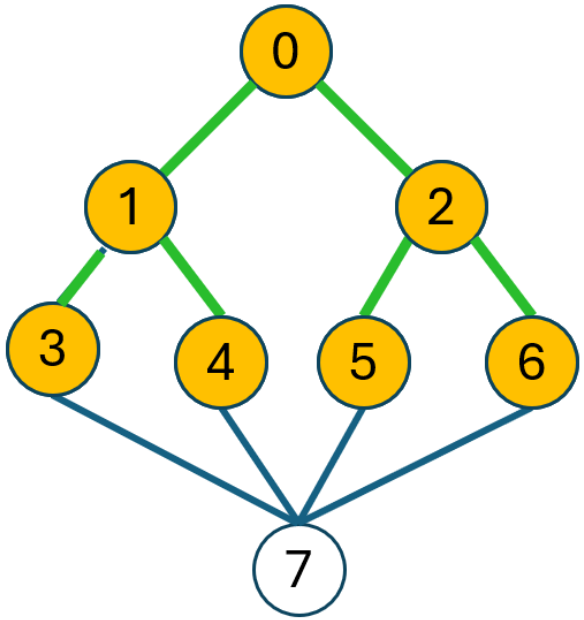


$0 \rightarrow 1 \rightarrow 2 \rightarrow 3 \rightarrow 4 \rightarrow 5 \rightarrow$

Example (Illustration)

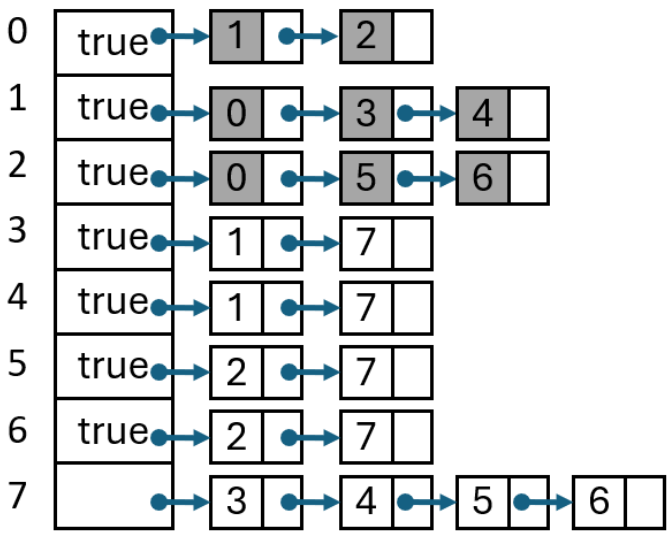


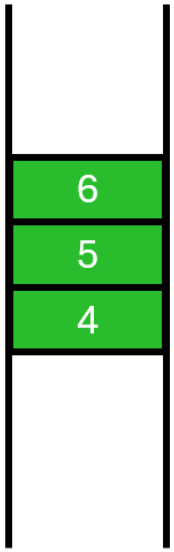


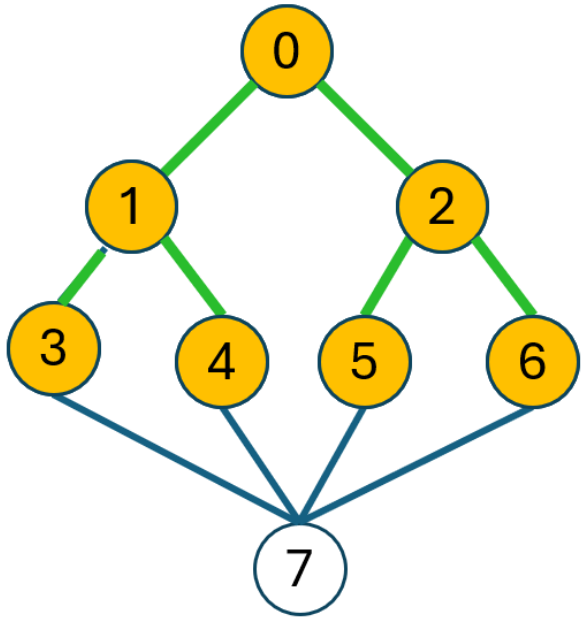


$0 \rightarrow 1 \rightarrow 2 \rightarrow 3 \rightarrow 4 \rightarrow 5 \rightarrow 6 \rightarrow$

Example (Illustration)

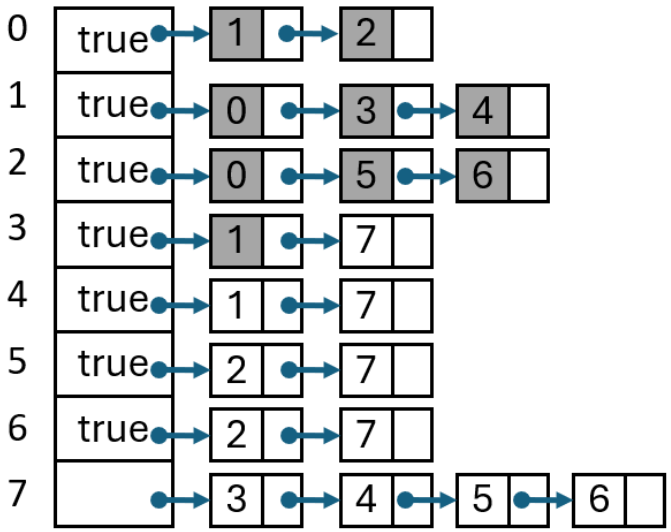


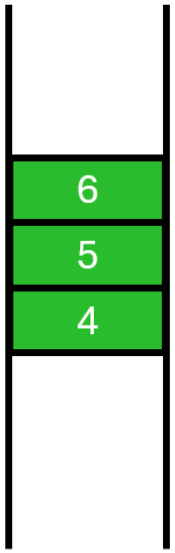


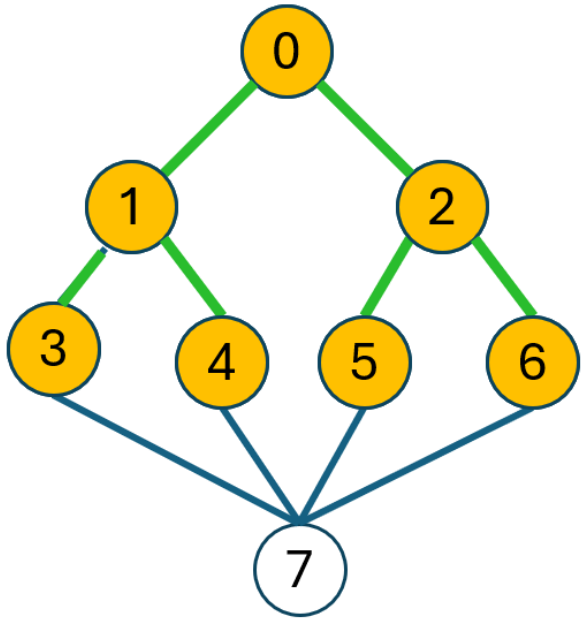


$0 \rightarrow 1 \rightarrow 2 \rightarrow 3 \rightarrow 4 \rightarrow 5 \rightarrow 6 \rightarrow$

Example (Illustration)

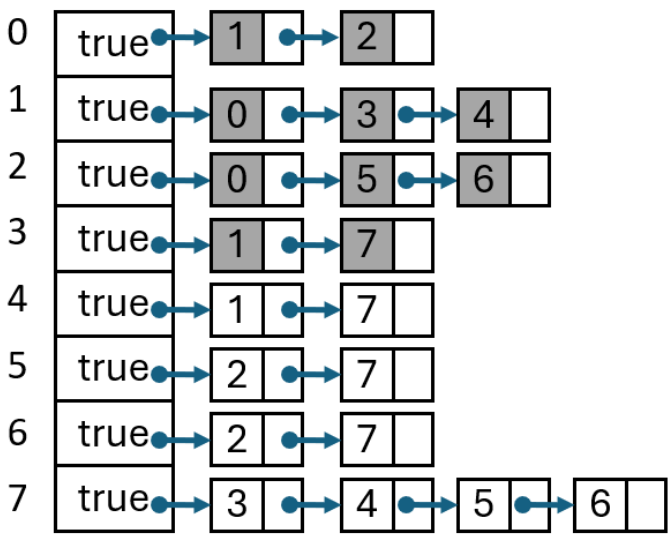


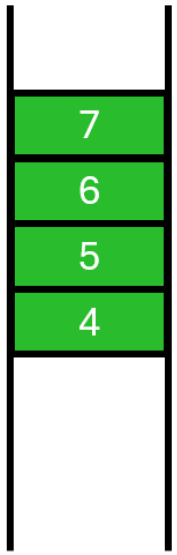


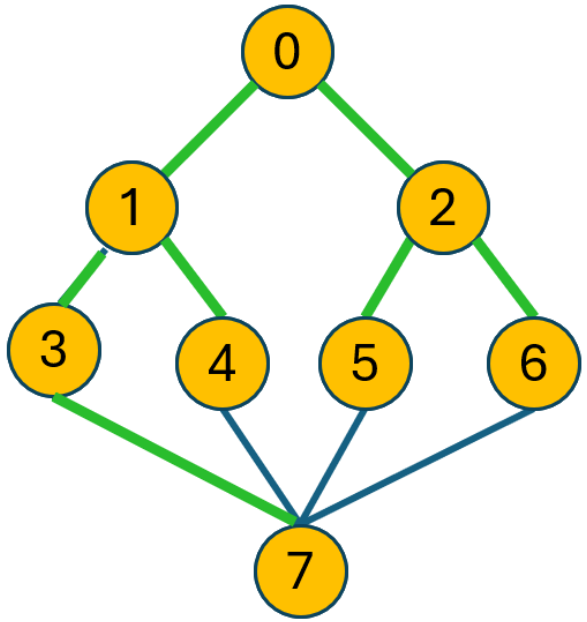


$0 \rightarrow 1 \rightarrow 2 \rightarrow 3 \rightarrow 4 \rightarrow 5 \rightarrow 6 \rightarrow$

Example (Illustration)

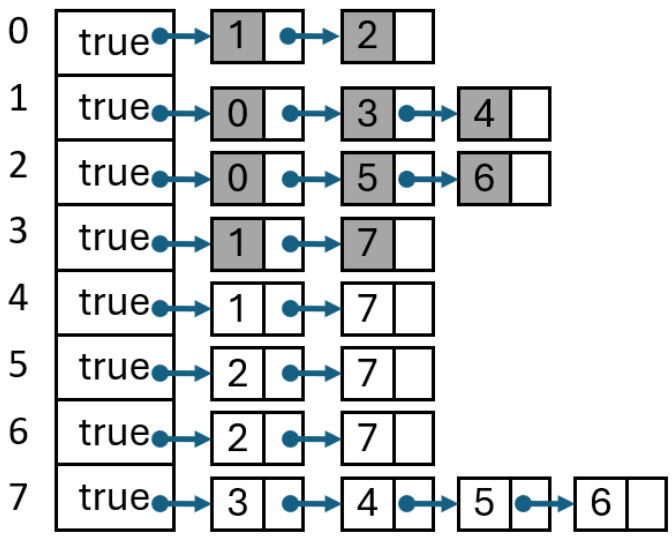


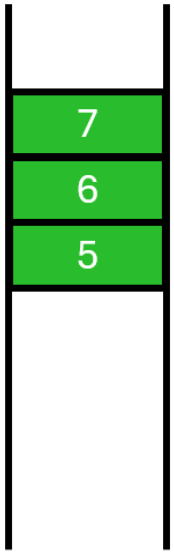


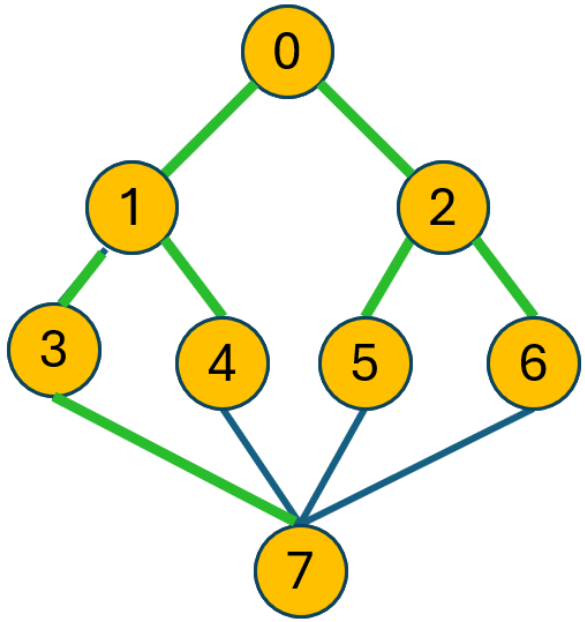


$0 \rightarrow 1 \rightarrow 2 \rightarrow 3 \rightarrow 4 \rightarrow 5 \rightarrow 6 \rightarrow 7$

Example (Illustration)

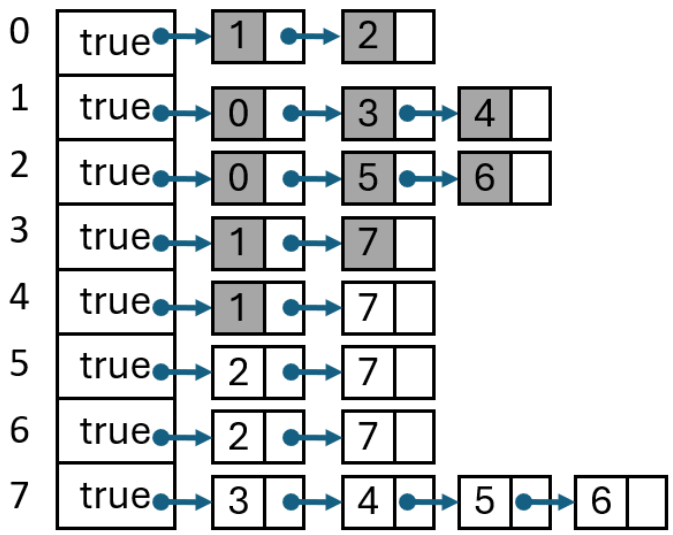


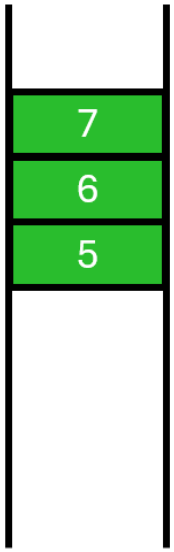


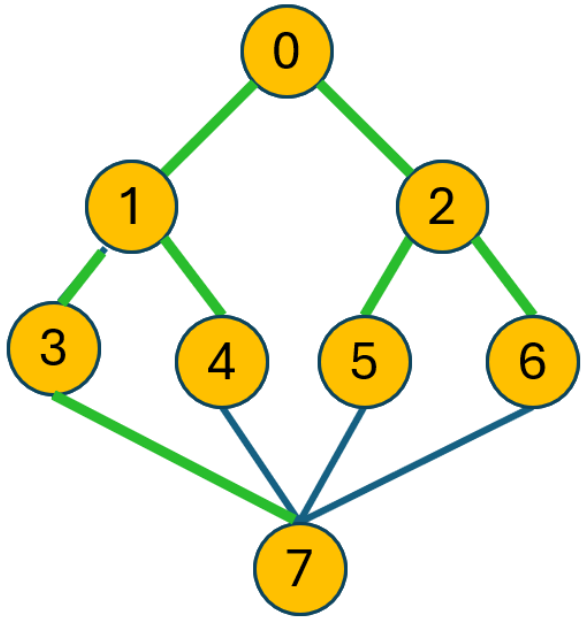


$0 \rightarrow 1 \rightarrow 2 \rightarrow 3 \rightarrow 4 \rightarrow 5 \rightarrow 6 \rightarrow 7$

Example (Illustration)

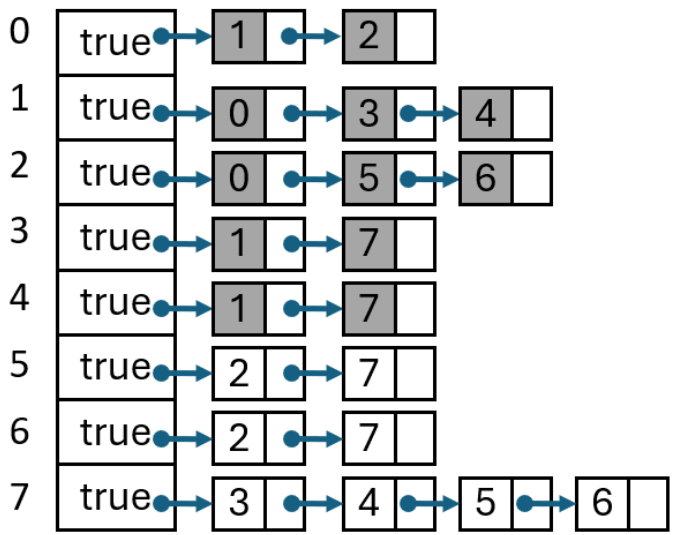


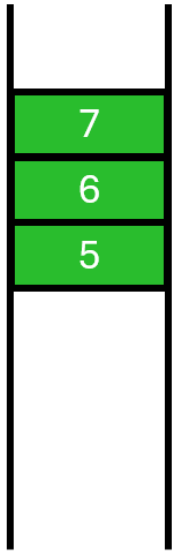


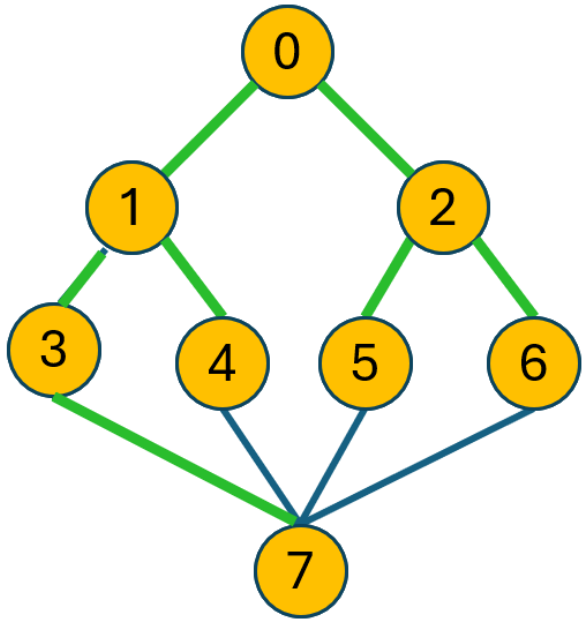


$0 \rightarrow 1 \rightarrow 2 \rightarrow 3 \rightarrow 4 \rightarrow 5 \rightarrow 6 \rightarrow 7$

Example (Illustration)

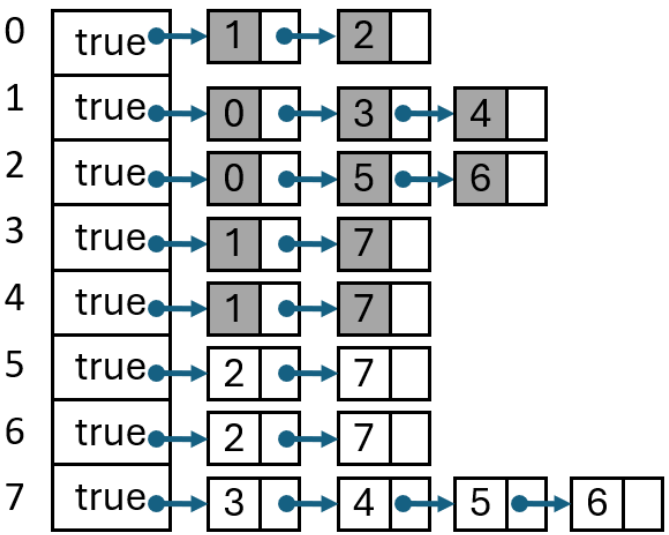


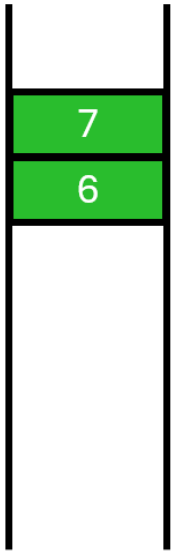


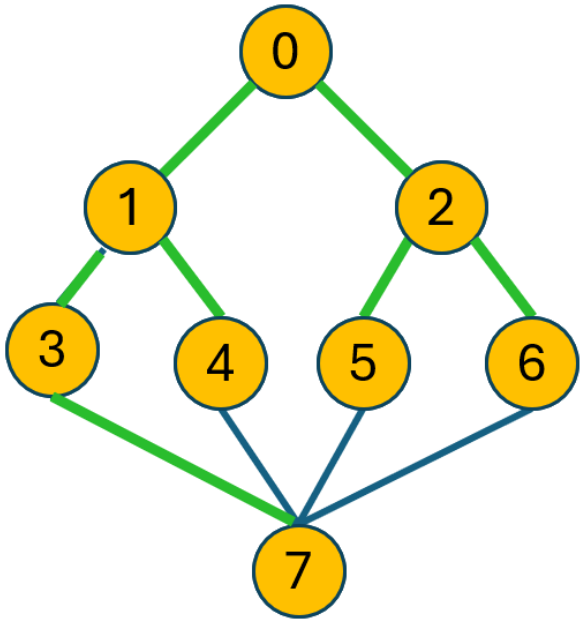


$0 \rightarrow 1 \rightarrow 2 \rightarrow 3 \rightarrow 4 \rightarrow 5 \rightarrow 6 \rightarrow 7$

Example (Illustration)

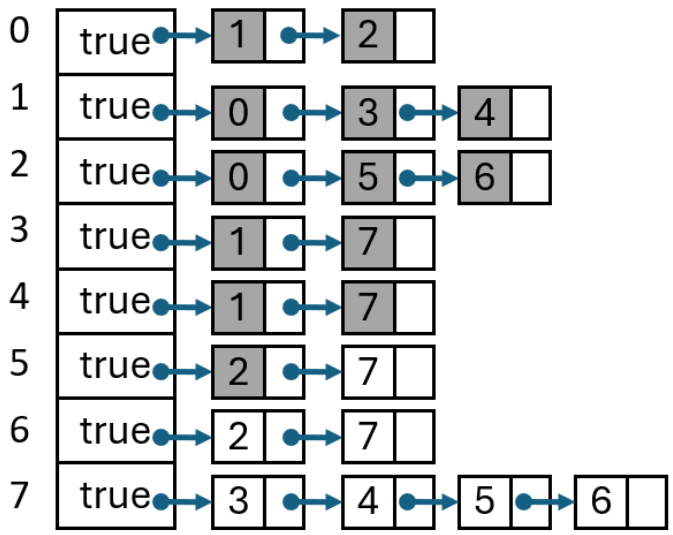


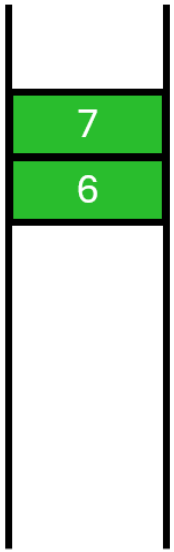


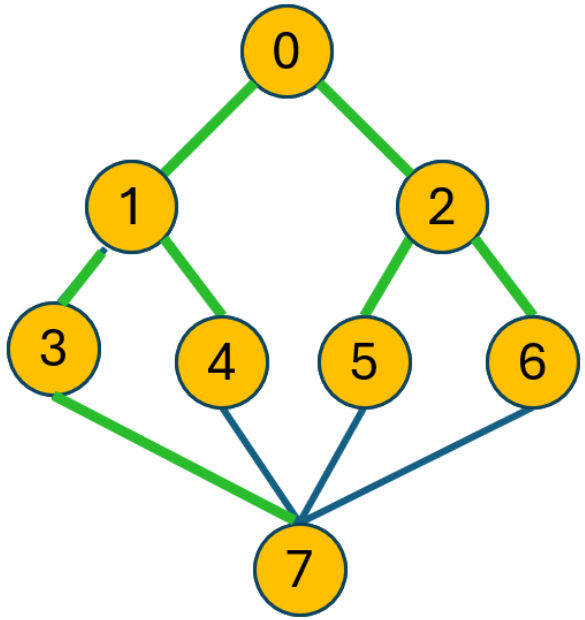


$0 \rightarrow 1 \rightarrow 2 \rightarrow 3 \rightarrow 4 \rightarrow 5 \rightarrow 6 \rightarrow 7$

Example (Illustration)

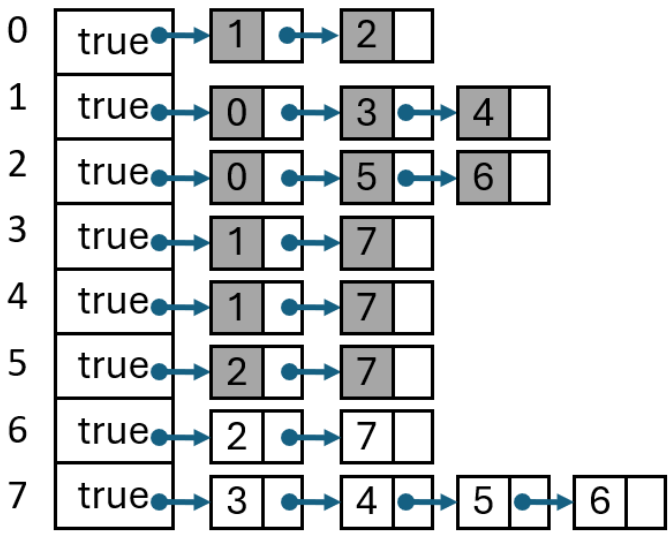


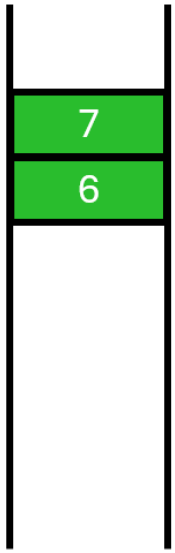


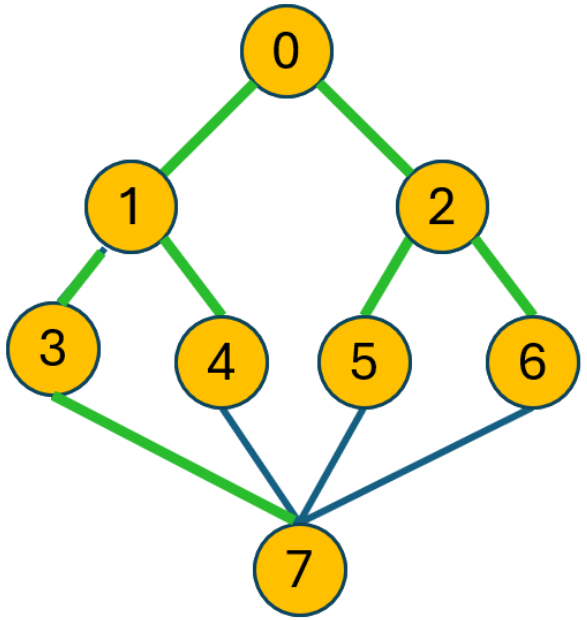


$0 \rightarrow 1 \rightarrow 2 \rightarrow 3 \rightarrow 4 \rightarrow 5 \rightarrow 6 \rightarrow 7$

Example (Illustration)

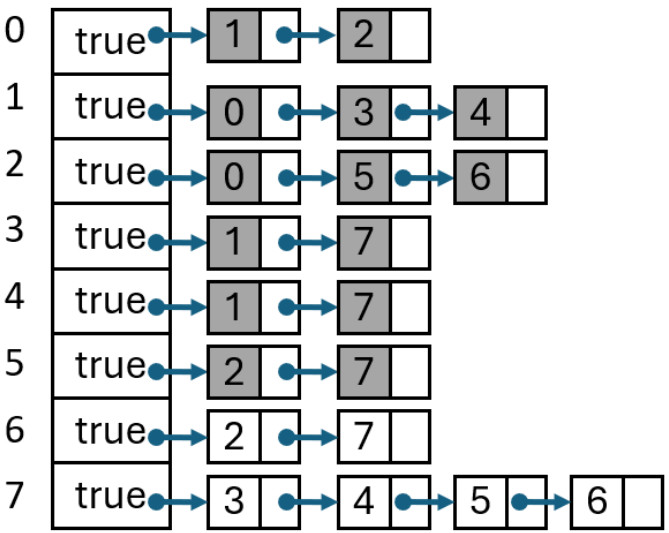


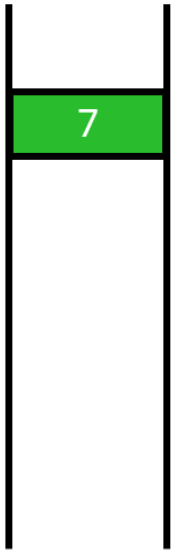


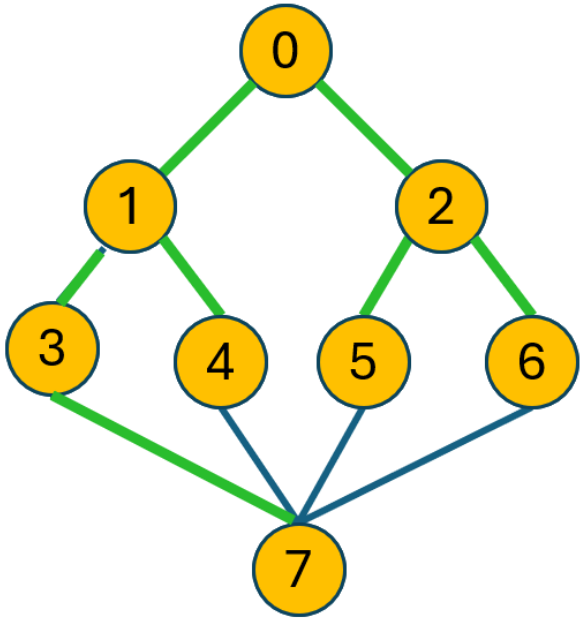


$0 \rightarrow 1 \rightarrow 2 \rightarrow 3 \rightarrow 4 \rightarrow 5 \rightarrow 6 \rightarrow 7$

Example (Illustration)

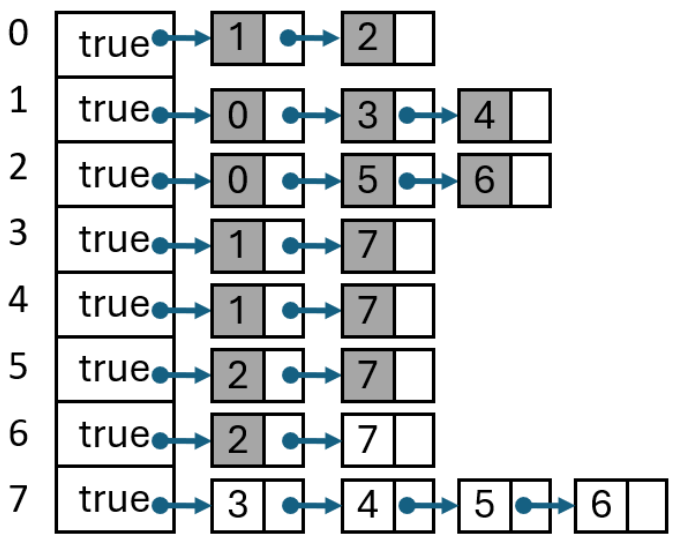


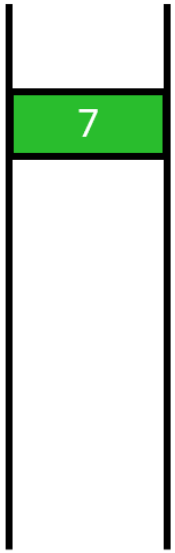


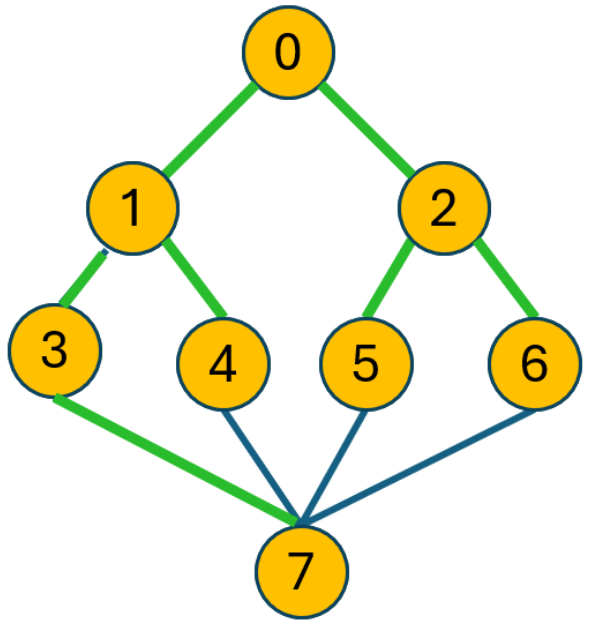


$0 \rightarrow 1 \rightarrow 2 \rightarrow 3 \rightarrow 4 \rightarrow 5 \rightarrow 6 \rightarrow 7$

Example (Illustration)

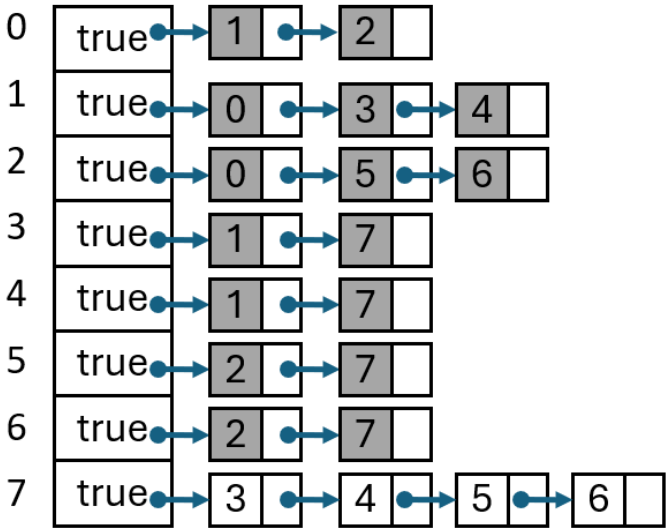


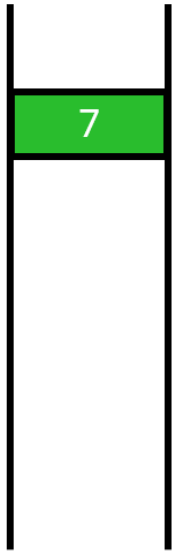


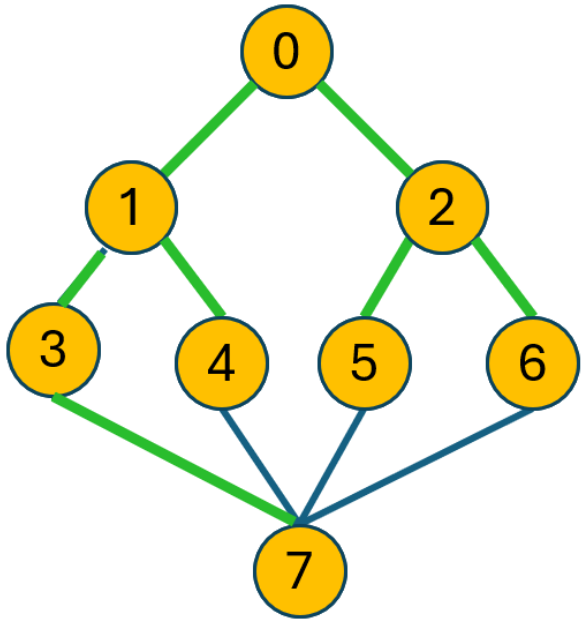


$0 \rightarrow 1 \rightarrow 2 \rightarrow 3 \rightarrow 4 \rightarrow 5 \rightarrow 6 \rightarrow 7$

Example (Illustration)

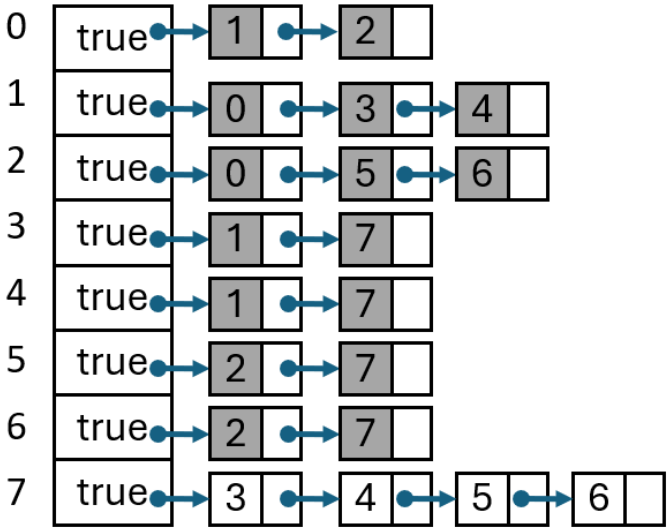


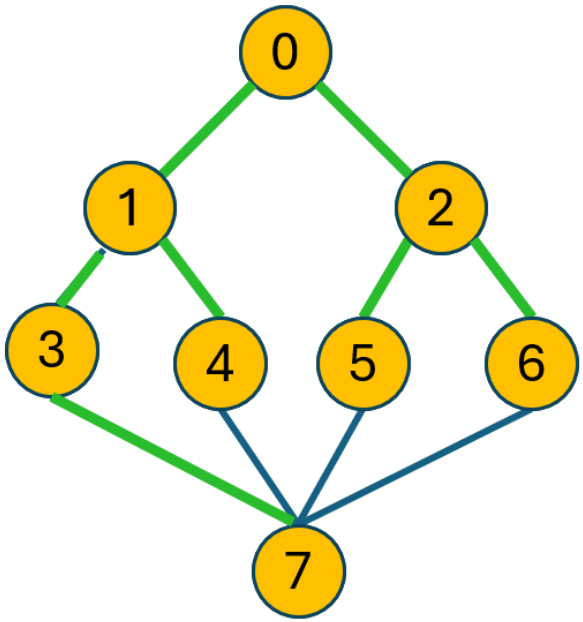




$0 \rightarrow 1 \rightarrow 2 \rightarrow 3 \rightarrow 4 \rightarrow 5 \rightarrow 6 \rightarrow 7$

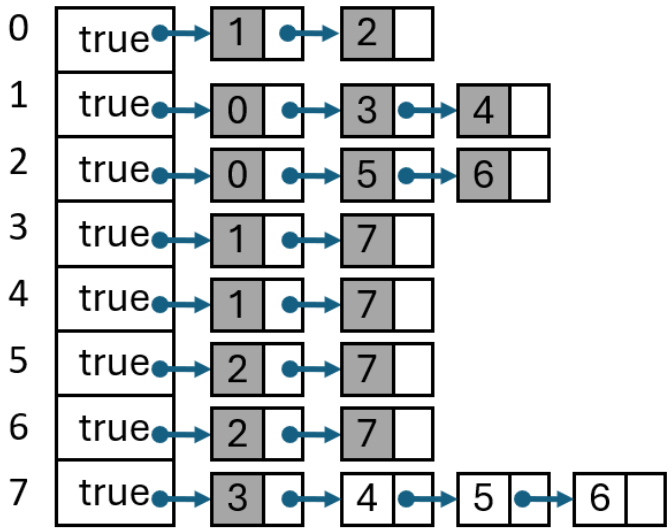
Example (Illustration)

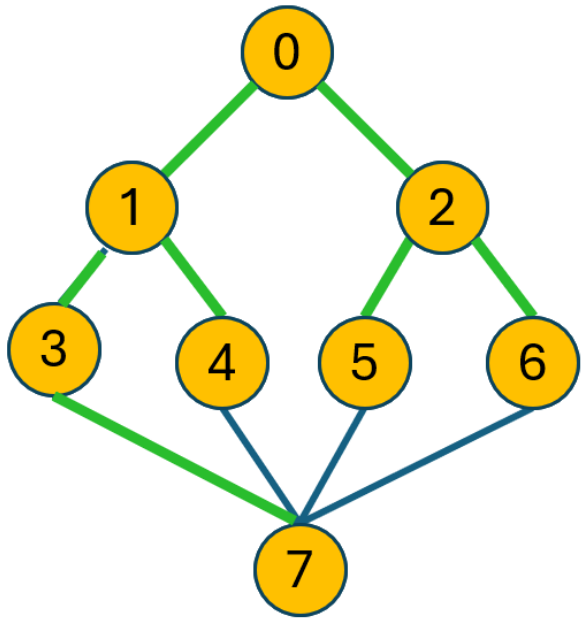




$0 \rightarrow 1 \rightarrow 2 \rightarrow 3 \rightarrow 4 \rightarrow 5 \rightarrow 6 \rightarrow 7$

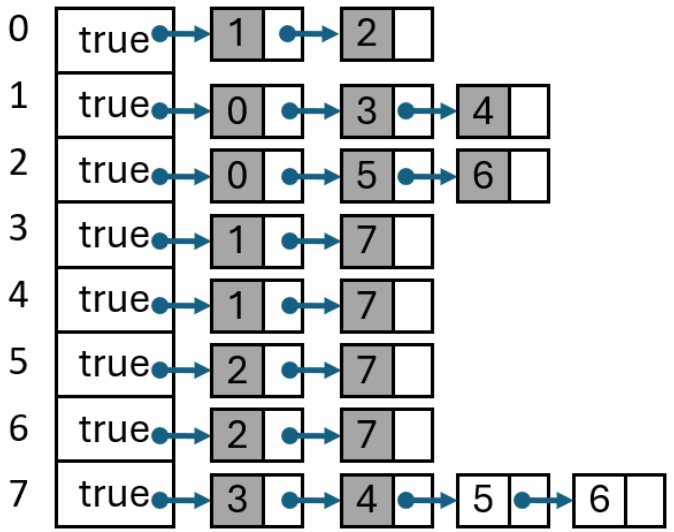
Example (Illustration)

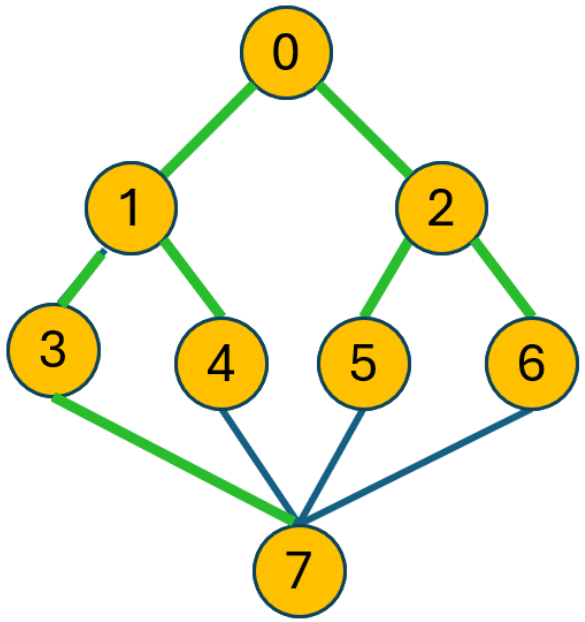




$0 \rightarrow 1 \rightarrow 2 \rightarrow 3 \rightarrow 4 \rightarrow 5 \rightarrow 6 \rightarrow 7$

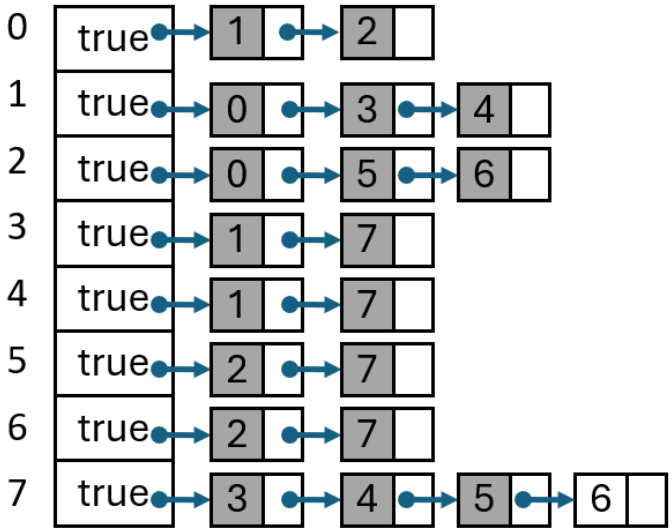
Example (Illustration)

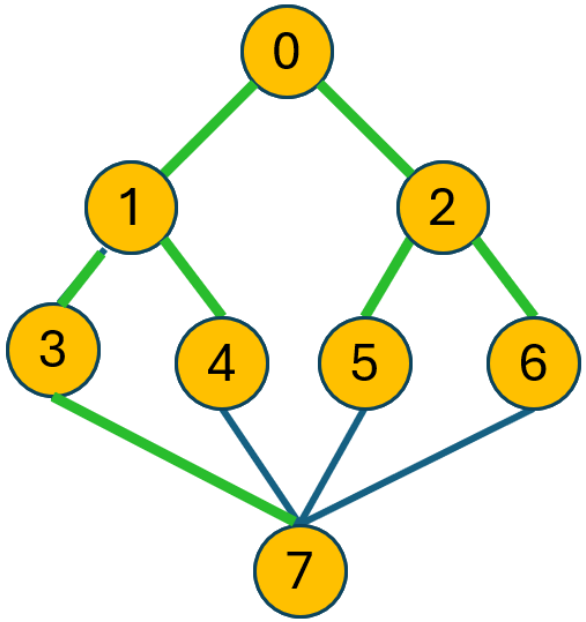




$0 \rightarrow 1 \rightarrow 2 \rightarrow 3 \rightarrow 4 \rightarrow 5 \rightarrow 6 \rightarrow 7$

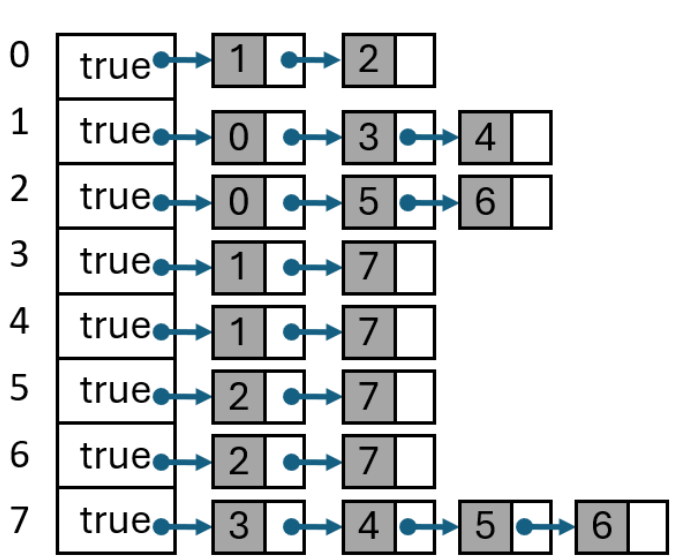
Example (Illustration)

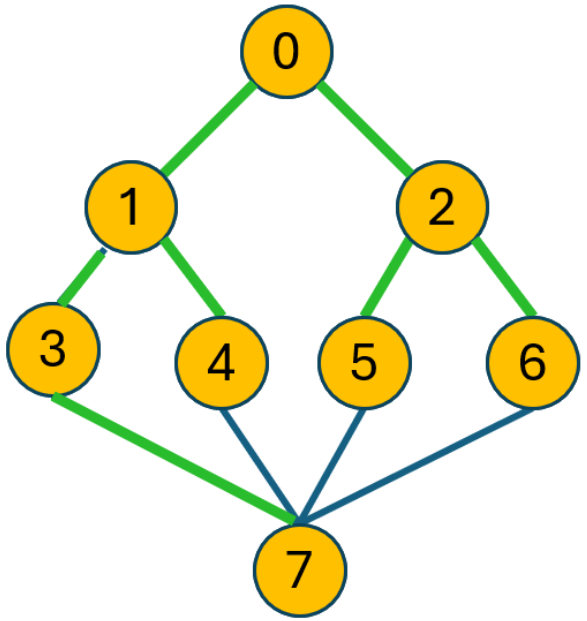




$0 \rightarrow 1 \rightarrow 2 \rightarrow 3 \rightarrow 4 \rightarrow 5 \rightarrow 6 \rightarrow 7$

Example (Illustration)

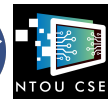




$0 \rightarrow 1 \rightarrow 2 \rightarrow 3 \rightarrow 4 \rightarrow 5 \rightarrow 6 \rightarrow 7$

The Pseudocode of BFS

```
BFS(G, u) { // let Q be the queue
    Q.enqueue(u)
    u.visited = True
    while (Q.empty() == False) { // when Q is not empty
        v = dequeue(Q)
        for all w in N(v) {
            if (w.visited == False) {
                Q.enqueue(w)
                w.visited = True
            }
        }
    }
}
driving main () {
    for each u in G
        u.visited = false
    BFS(G, u)
}
```

BFS in C

```
void bfs(int v) {
    nodePointer w;
    front = rear = NULL;  /* initialize queue */
    printf("%5d",v);
    visited[v] = TRUE;
    addq(v);

    while (front) {
        v = dequeue();
        for (w = graph[v]; w ; w->link)
            if (!visited[w->vertex]) {
                printf("%5d", w->vertex);
                enqueue(w->vertex);
                visited[w->vertex] = TRUE;
            }
    }
}
```

Analysis of BFS

- Since each vertex is placed on the queue exactly once, the while loop is iterated at most n times.

Analysis of BFS

- Since each vertex is placed on the queue exactly once, the while loop is iterated at most n times.
  - For the adjacency list representation, this loop has a total cost of $d_0 + d_1 + \ldots + d_{n-1} = O(e)$, where $d_i = \text{degree}(v_i)$.

Analysis of BFS

- Since each vertex is placed on the queue exactly once, the while loop is iterated at most n times.
  - For the adjacency list representation, this loop has a total cost of $d_0 + d_1 + \ldots + d_{n-1} = O(e)$, where $d_i = \text{degree}(v_i)$.
- For the adjacency matrix representation, the while loop takes $O(n)$ time for each vertex visited.
  - Therefore, the total time is $O(n^2)$.

As was true of DFS, all vertices visited, together with all edges incident to them, form a connected component of G.

Discussions