Author: Ahmed Nouralla - Group: BS19-02 - a.shaaban@innopolis.university

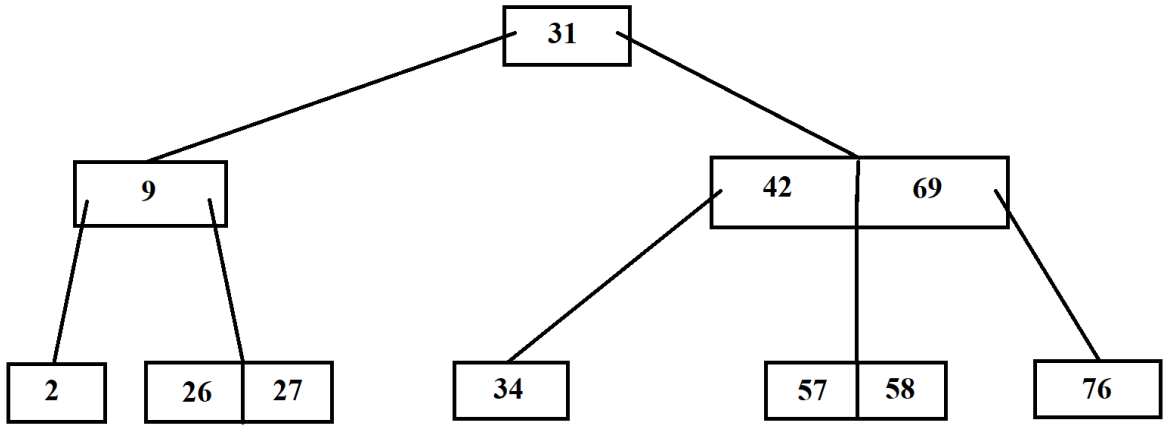
3.1 Constructing B-tree

Show the resultant B-tree (minimum degree $t = 2$) after inserting the following keys in order:

9, 31, 42, 69, 2, 26, 34, 57, 76, 27, 58

Number of keys in a single node ∈ [t-1, 2t-1] = [1, 3], Max no. of children for a node is 4

This diagram follows Cormen's book algorithm, which splits before inserting, it doesn't create a temporary 4-node.



3.2 The square graph:

Give algorithms and their respective time complexities using the Big-O notation for constructing the square graph of G.

Algorithm 1: Graph is represented as adjacency list: List G[]

```
Graph SquareGraph (Graph G) {

      foreach vertex v in G

            int[] distance = BFS(v); // BFS will return a lookup table
            // distance[x] contains the shortest distance from v to x.

            foreach vertex w in G
                  if distance[w] = 2 then
                        G.addEdge(v, w) // Will add the edge if not present
                  endif
            endfor

      endfor

      // Returns the squared graph.
      return G
}
```

Time complexity, BFS will run in O(V((V + E) + V)) = **O(V(V+E))**. If the graph is connected, this will be simplified to **O(VE)**.

Algorithm 2: Graph is represented as adjacency matrix: int G[][]

Where G[a][b] is either 1 if there is an edge from a to b, or 0 if there is no edge.

```
Graph SquareGraph (Graph G) {

      // Assuming vertices are indexed from 1 to V
      // Modify the graph to have non-zero entries on the main diagonal.
      // Required for the algorithm to work correctly, Linear time operation.

      foreach i from 1 to V
            G[i][i] = 1
      endfor

      // Do the matrix multiplication on the adjacency matrix G
      // Store the result in G itself.
      // A standard algorithm for matrix multiplication takes Cubic time.

      G = G * G

      // Modify the resulting matrix to have 1 in all non-zero entries.

      foreach i from 1 to V
            foreach j from 1 to V
                  if G[i][j] is not 0 then
                        G[i][j] = 1
                  endif
            endfor
      endfor

      // Returns the squared graph.
      return G
}
```

Overall time complexity: $\mathbf{O (V + V^3 + V^2) = O(V^3)}$

In practice: It will be close to $O(V^2)$, Because boolean matrix multiplication can be done efficiently by exiting when we get a true value.