DISTRIBUTED SYSTEMS LABORATORY

ASSIGNMENT 1,2,3

NAME:SUDHANSU SEKHAR SWAIN

ROLL:118CS0689

LAB 1

AIM OF THE EXP.: Design a network topology (Star, Ring, Tree) with arbitrary number of nodes. Determine the adjacency matrix, list and degree of nodes.

PROCEDURE(PSEUDOCODE):

1. Declare two variables numNodes (integer type) and topologyType (enum type) and take them as input.

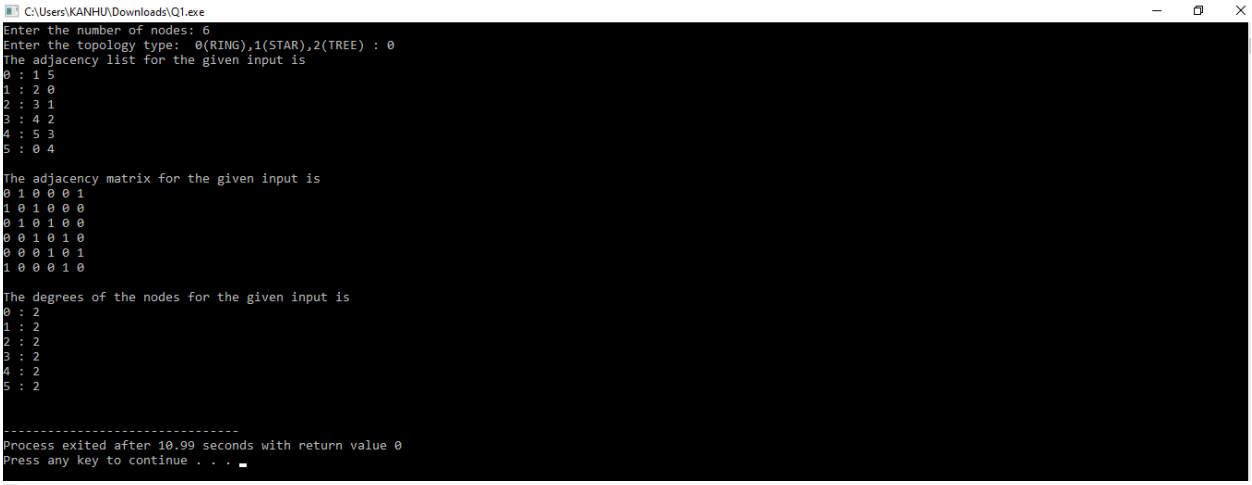
from the user

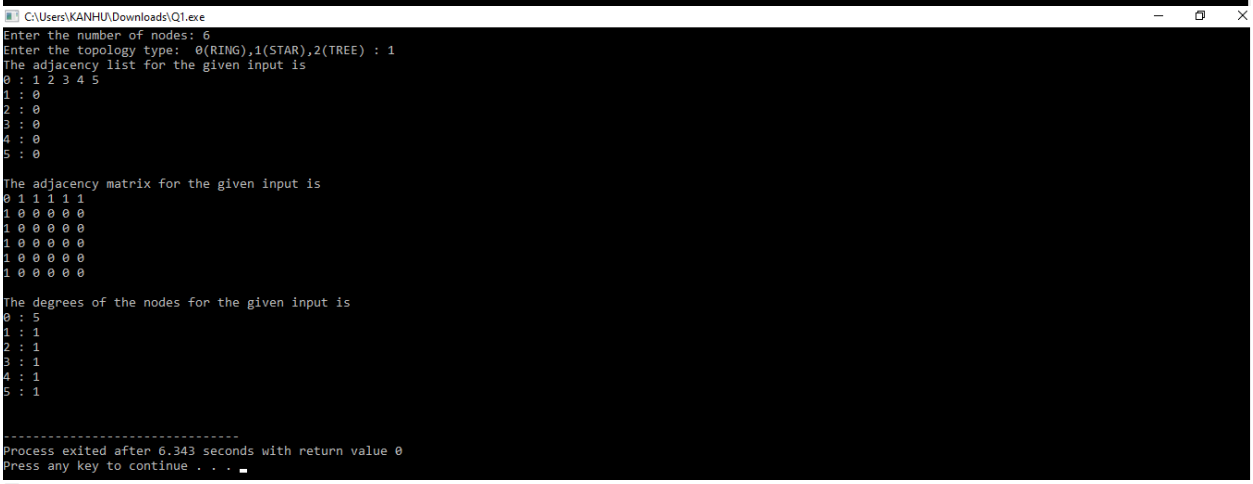
2. If the topologyType "STAR" is selected,
    2.1. For each i from 1 to n-1
        i. Add an edge between node i and node 0.

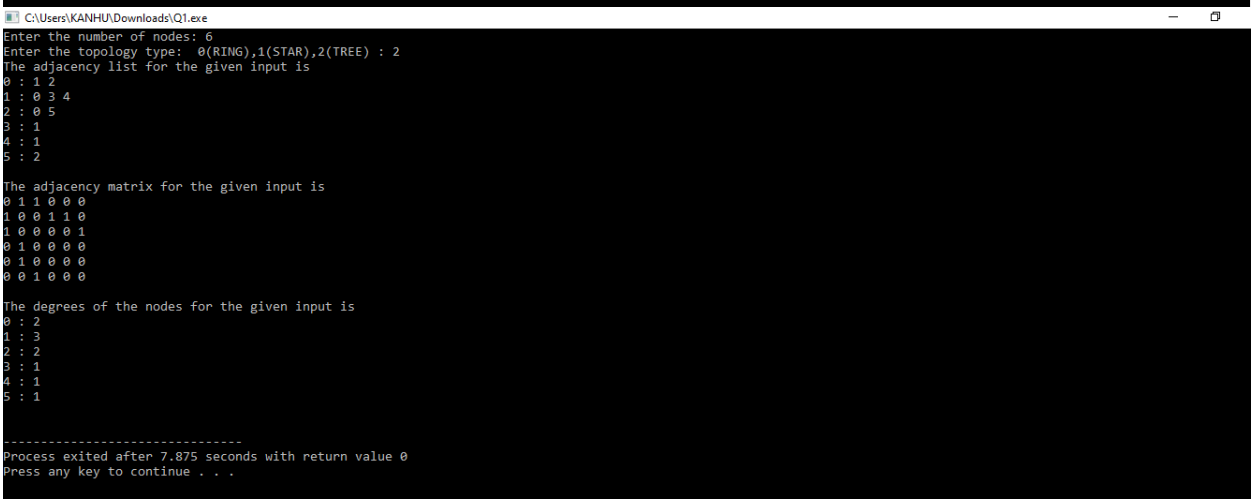
3. If the topologyType "RING" is selected,
    3.1. For each i from 0 to n-1
        i. Add an edge between node i and node ((i%n + 1)%n)
        ii. Add an edge between node i and node ((i%n-1)%n + n)%n)

4. If the topologyType "TREE" is selected,
    4.1. For i from 0 to n-1
        i. If i*2+1 < n, add an edge between node i and node i*2+1
        ii. If i*2+2 < n, add an edge between node i and node i*2+2
5. Output the Adjacency List, Adjacency Matrix and Degree for each node.
6. END

RESULTS: The screenshots of user inputs/outputs is presented below







CONCLUSION:

The Adjacency Lists, Adjacency Matrices and The Degrees of Nodes have been output for different network topologies (namely STAR, RING and TREE). The time complexity of computation is O(n) but the overall timecomplexity is O(n^2), due

to the formation of the adjacency matrix. The maximum n that can be input without exceeding the memory limit is ~10000.

LAB 2

AIM: Design a hypercube and hybrid topology (Combination of star, ring, tree topology). Determine the adjacency list, adjacency matrix, and the degree of a node.

PROCEDURE(ALGORITHM):

1. Take as input the topologyType (Enum).

2. If the topologyType == HYPERCUBE:

    a. Take the number of bits(n) to represent the topology(degree).

    b. The number of nodes in that topology is 2n

    c. For i from 0 to 2n-1,

        i. For j from 0 to n,

            1. Join the node i with node i XOR (2j).

3. If the topologyType == HYBRID:

    a. Take the number of nodes(n) and number of edges (e) as input.

    b. If the e > n(n-1)/2, throw error as a simple topology is not possible to be built

    c. Else

        i. Take the nodes a and b as input

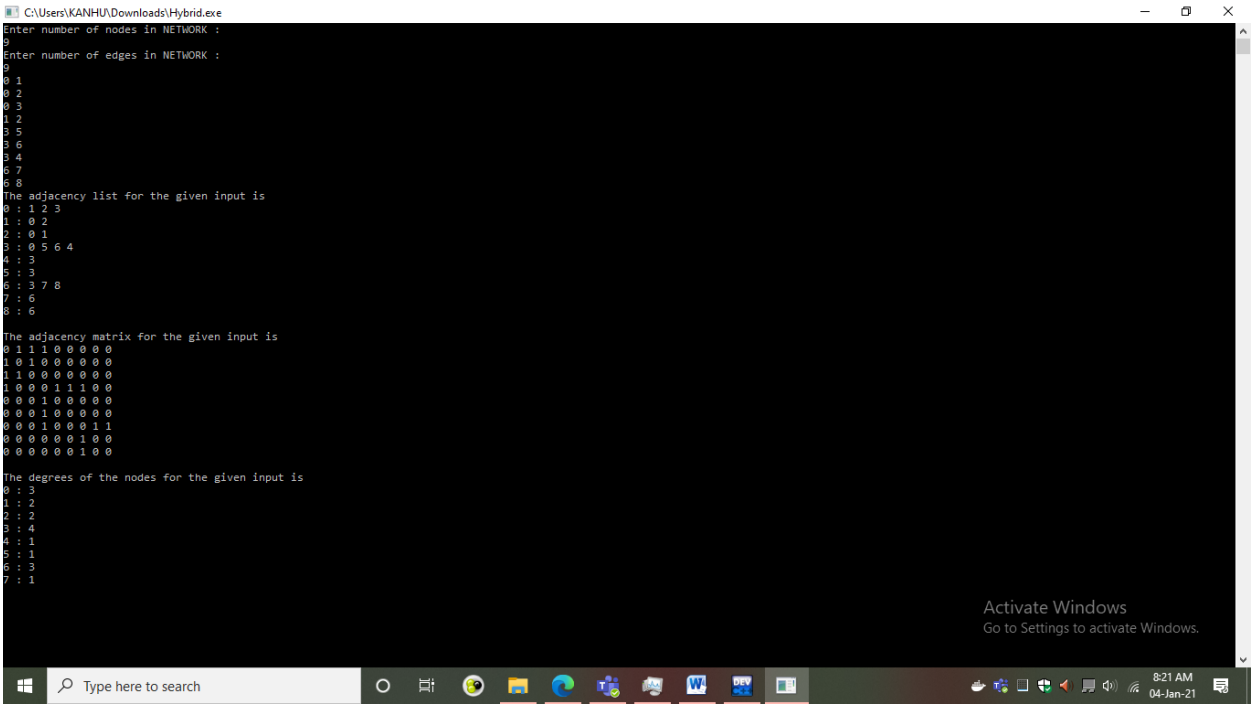
        ii. Add an edge between a and b.

4. Output the Adjacency List, Adjacency Matrix and Degree for each node.

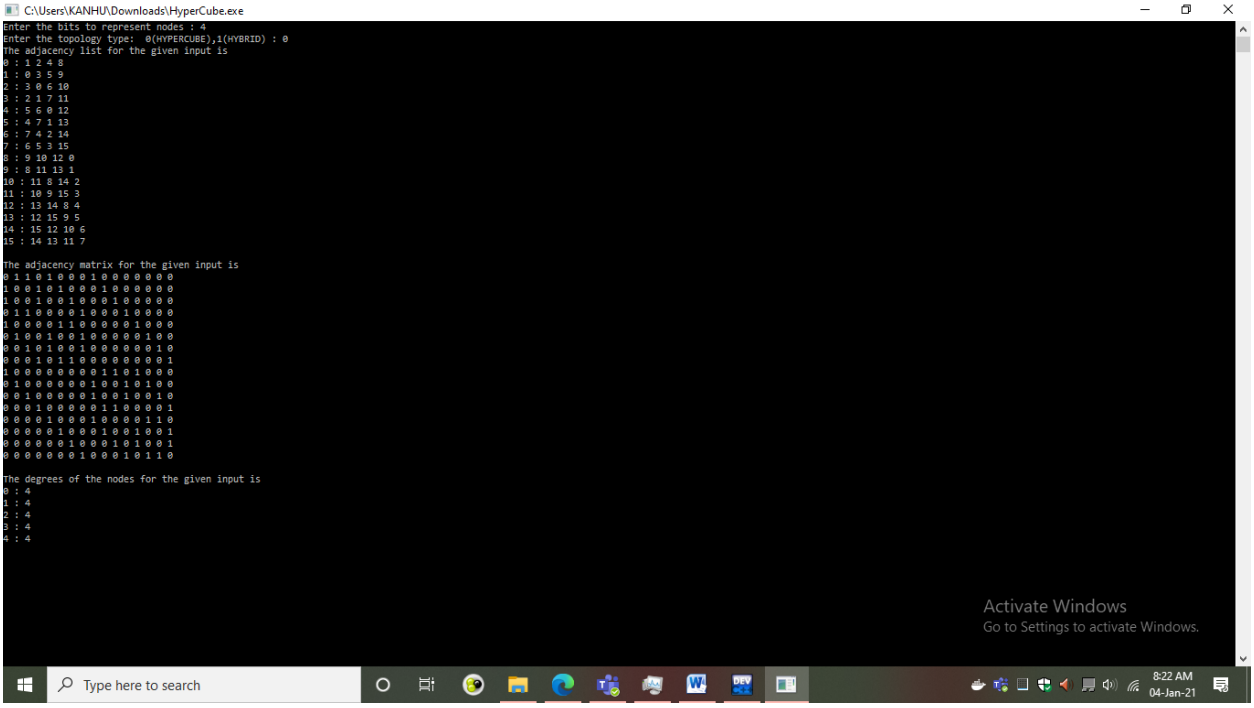
5. END

RESULT:

Hybrid Topology



Hypercube Topology:



CONCLUSION:

The Adjacency Lists, Adjacency Matrices and The Degrees of Nodes have been output for different network topologies (namely HYPERCUBE and HYBRID). The worst-case time complexity is O(2^n). The maximum n that can be input without exceeding the memory limit is ~30.

LAB-3

AIM: Design a weighted hybrid topology and determine the shortest path between random source and destination using Djikstra's algorithm.

PROCEDURE(ALGORITHM):

1. Take the number of edges (e) and the number of nodes(n) as input

2. For i from 0 to e-1:

a. Take as input two nodes a, b and the weight of the edge between them (d).

b. Add an edge between a and b, with weight d.

3. Take as input the source node(src) and the destination node(dest)

4. Create a min priority queue initially empty(pq)

5. Maintain a map of the distance(distanceRecord) of each node. Initially, distanceRecord(src) =0 and for all other nodes it is +infinity.

6. Push the src node along with distanceRecord(src) to pq.

7. While pq is not empty:

a. Pop the top node from pq. Let it be currentNode.

b. For all nodes adjacent to currentNode, (adjacentNode):

i. If distanceRecord(currentNode) + weight (edge between currentNode and adjacentNode) < distanceRecord(adjacentNode)

1. Remove the adjacentNode from pq, if already present

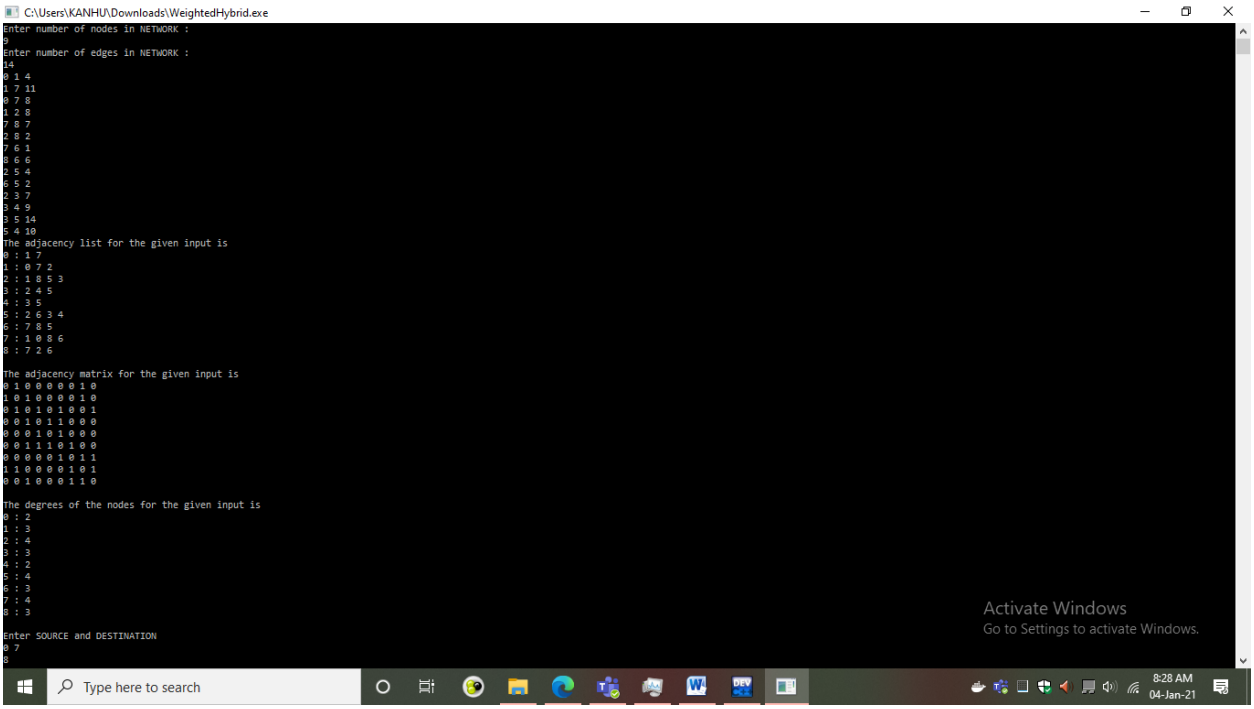
2. Update the distanceRecord(adjacentNode) = distanceRecord(currentNode) + weight (edge between currentNode and adjacentNode).

3. Push the adjacentNode along with distanceRecord(adjacentNode) to pq.

8. Return the distanceRecord(dest) as the output

9. END

RESULT:



CONCLUSION: The shortest path between any source and destination in a network having Hybrid Topology was calculated using the Dijkstra's Shortest Path algorithm, assuming that there is no Negative edges in the graph. The time complexity of the algorithm is O(V+Elog(V)).