Algorithm - TCP

Client

1. Create a socket
2. Connect the socket to the server
3. Read the string to be reversed from the standard input and send it to the server
Read the matrices from the standard input and send it to the server using the socket
4. Read the reversed string from the socket and display it on the standard output
Read the product matrix from the socket and display it on the standard output
5. Close the socket

Server

1. Create a listening socket
2. bind IP address and port number to the socket
3. listen for incoming requests on the listening socket
4. accept the incoming request
5. connection socket is created when accepting returns
6. Read the string using the connection socket from the client
7. Reverse the string
8. Send the string to the client using the connection socket
9. close the connection socket
10. close the listening socket

Algorithm- UDP

Client

1. Create a socket
2. Read the matrices from the standard input and send it to the server using the socket
3. Read the product matrix from the socket and display it on the standard output
4. Close the socket

Server

1. Create a socket
2. bind IP address and port number to the socket
3. Read the matrices socket from the client using the socket
4. Find the product of matrices
5. Send the product matrix to the client using socket
6. close the socket

Link-State Routing Algorithm:
(It is the Dijkstra algorithm. This algorithm finds the shortest (least cost paths) from the source u to every other node in the network.
D(v): cost of the least cost path from the source node to the destination node v as of this iteration of the
algorithm
p(v): previous node of v along the current least cost path from the source to v
N': a subset of nodes. v is in N'
if the least cost path from the source to v is definitely known

Initialization :
N' = {u}

for all nodes v
  if v is a neighbor of u
    then D(v) = c(u,v)
  else
    D(v) = ∞
do {
  find w, not in N'
  such that D(w) is a minimum
  add w to N'
  update D(v) for each neighbor v of w and not in N'.

  D(v) = min{D(v) , D(w) + c(w,v)}
} while (N'!= N)

Algorithm - Leaky Bucket

Step 1: Input the bucket size, outgoing rate, and no of inputs
Step 2: While n is not equal to 0,
Step 3: Input the incoming packet size
Step 4: Print the incoming packet size
Step 5: If the incoming packet size is less than or equal to the bucket size - store,
Step 6: Add the incoming packet size to the store
Step 7: Print the bucket buffer size and the store
Step 8: Subtract the outgoing rate from the store
Step 9: If the store is less than 0,
Step 10: Set the store to 0
Step 11: Print the after outgoing packets left out of the bucket buffer size and the store
Step 12: Subtract 1 from n
Step 13: End while
Step 14: End program