Riga Technical University

Faculty of Telecommunications and Electronics

ASSIGNMENT TWO: ALGORITHMS and CISCO

Student: Rofaida Nezzar.

Professor: Tianhua Chen.

Riga 2024

Python ADT Stack and Graphs:

Task 1:

To convert a decimal number to its octal representation, we use a stack to reverse the order of remainders obtained by successive divisions of the decimal number by 8. The stack ensures that the least significant digit, obtained first, is processed last when constructing the octal number. The algorithm involves pushing the remainders onto the stack and then popping them off to form the final octal representation. For checking if brackets are properly matched, a stack is used to maintain the order of opening brackets as they appear in the expression. When a closing bracket is encountered, it is matched against the top of the stack to ensure it pairs with the correct opening bracket. If at any point the stack is empty or the brackets do not align, the expression is deemed invalid. The use of a stack provides a straightforward way to manage the nested and sequential nature of these operations, making it ideal for both tasks.

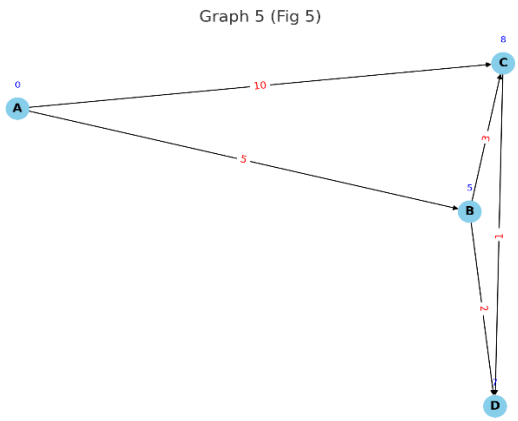
RESULTS:

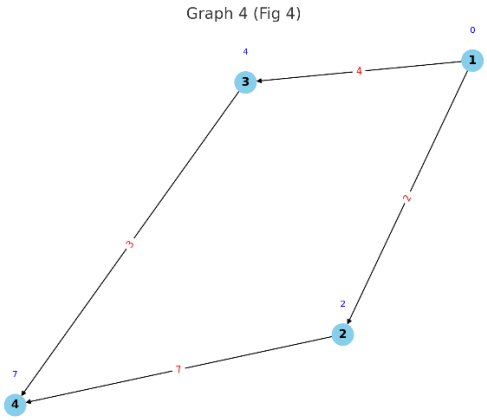
```
In [16]: runfile('C:/Users/nezza/.spyder-py3/
untitled2.py', wdir='C:/Users/nezza/.spyder-py3')
The octal representation of 12 is 14
Are the brackets in '{[(]}' properly matched?
False
```

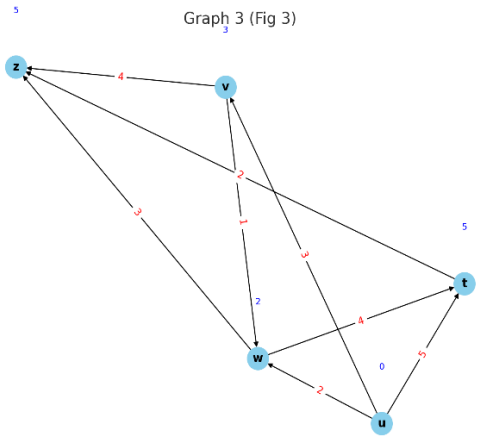
Task 2 Bellman-Ford Algorithm:

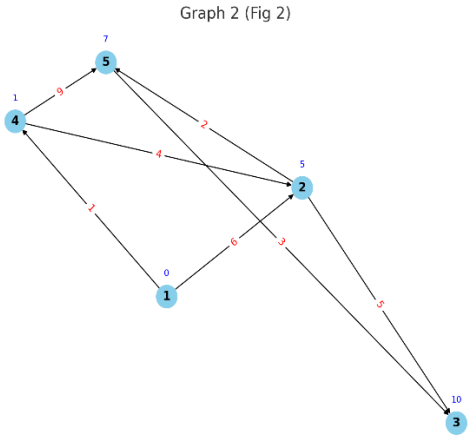
The Bellman-Ford algorithm was applied to all five graphs extracted from the document, with each visualization illustrating key aspects of the analysis. The nodes and edges represent the graph's structure, while the edge weights, annotated in red, highlight the cost of traversing each connection. Additionally, the shortest path costs, displayed near the nodes in blue, indicate the minimum distance from the source node, providing a clear depiction of the algorithm's output.

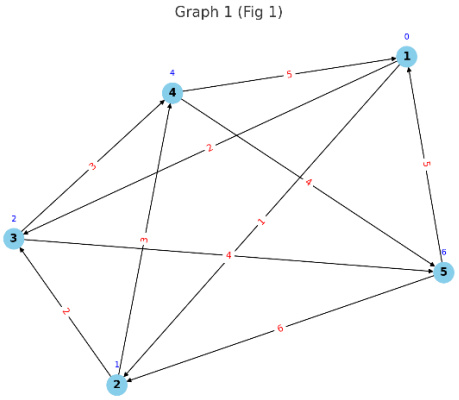
The graphs are below:





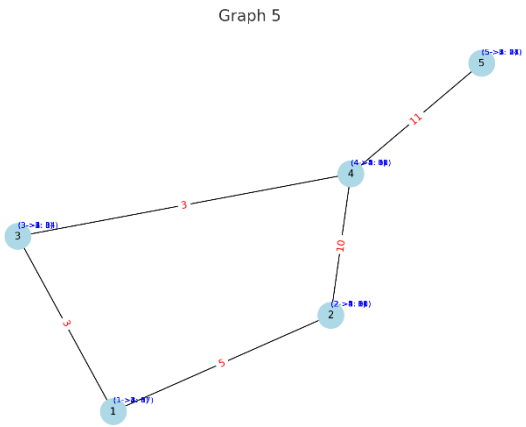


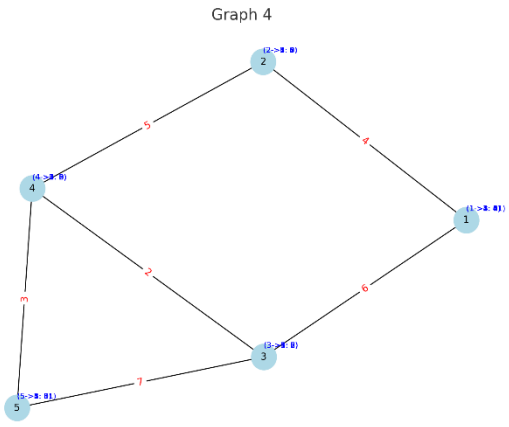


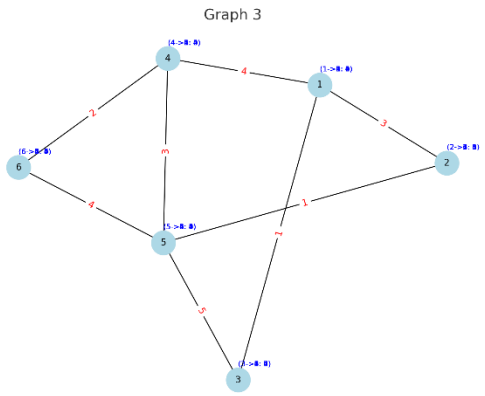


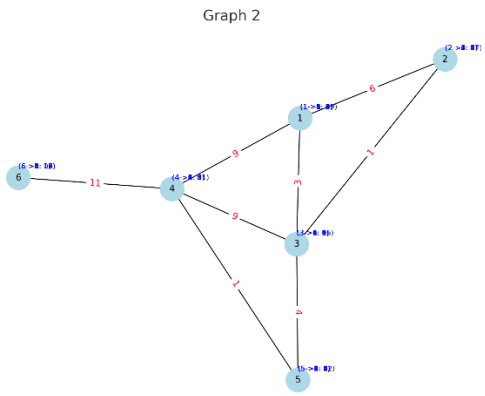
Task 3 Dijkstra's Algorithm:

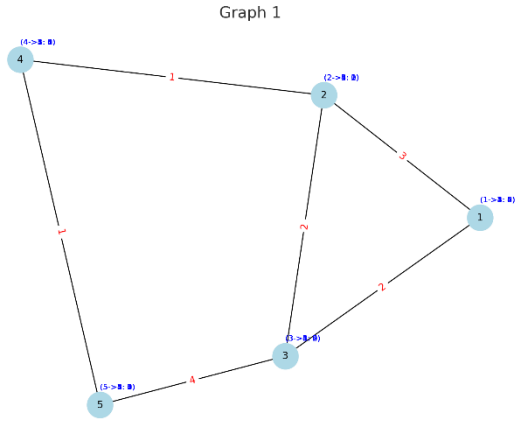
Dijkstra's algorithm is used to find the shortest paths from a **source node** to all other nodes in a graph. It ensures that the shortest path to a node is always computed before moving to other nodes. Using the same graphs to find the shortest path:











Task 4 Prim's Algorithm:

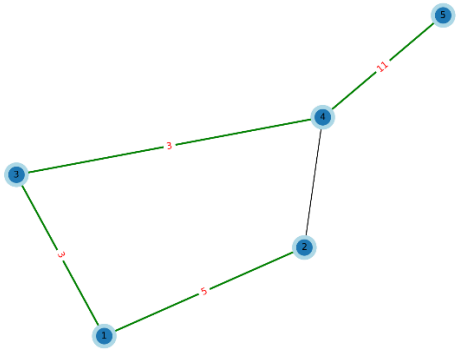
The visualizations show the Minimum Spanning Tree (MST) for each of the five graphs, highlighting:

- The edges included in the MST (in green).
- The total weight (cost) of the MST for each graph:
    - **Graph 1**: Cost = 6
    - **Graph 2**: Cost = 20
    - **Graph 3**: Cost = 10
    - **Graph 4**: Cost = 14
    - **Graph 5**: Cost = 22

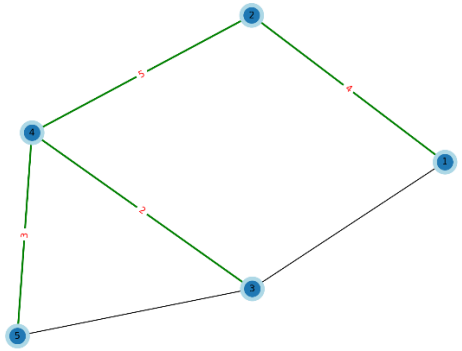
We can see that:

Dijkstra's algorithm finds the shortest path from a single source to all other nodes, making it suitable for routing problems (e.g., GPS navigation). On the other hand: Prim's algorithm finds an MST that connects all nodes with the minimum edge weight, which is ideal for network optimization problems.

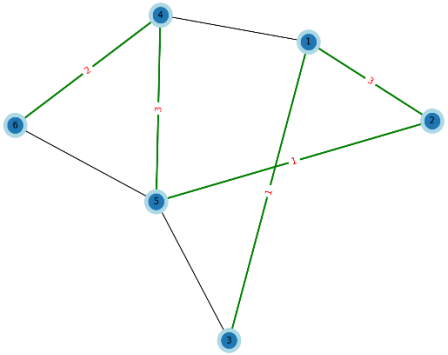
Graph 5 - Prim's MST (Cost: 22)



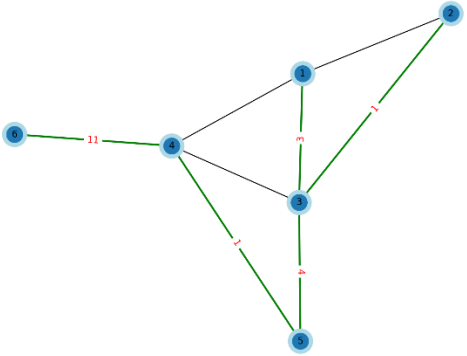
Graph 4 - Prim's MST (Cost: 14)



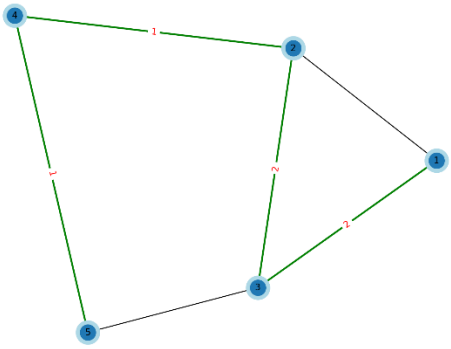
Graph 3 - Prim's MST (Cost: 10)



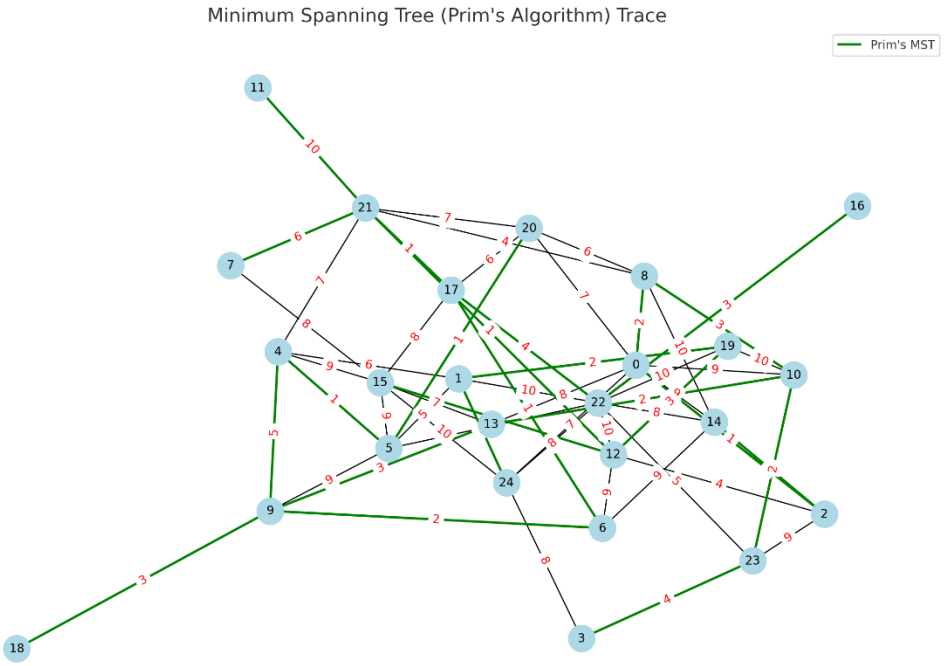
Graph 2 - Prim's MST (Cost: 20)

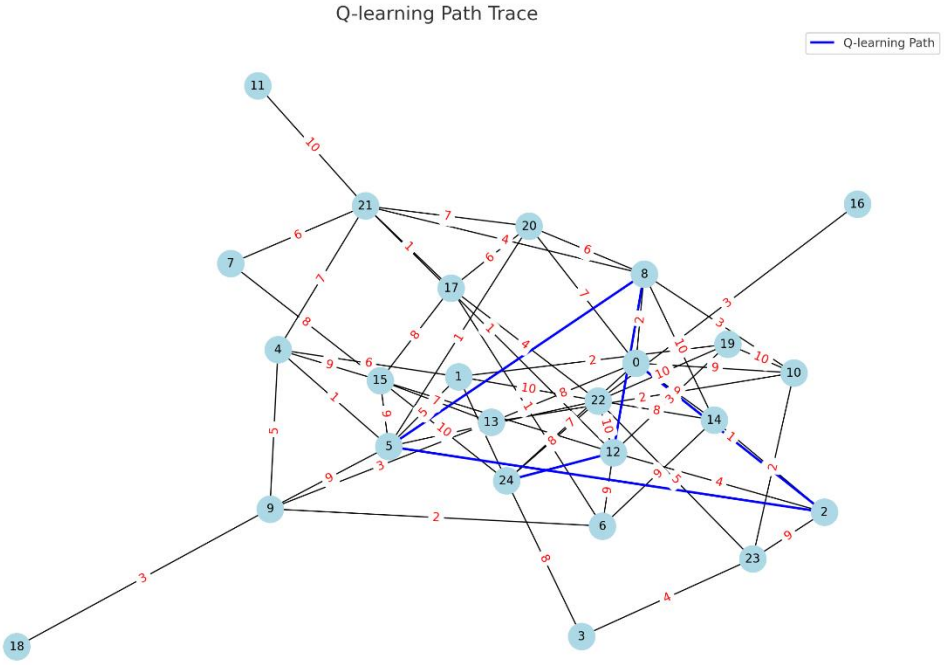


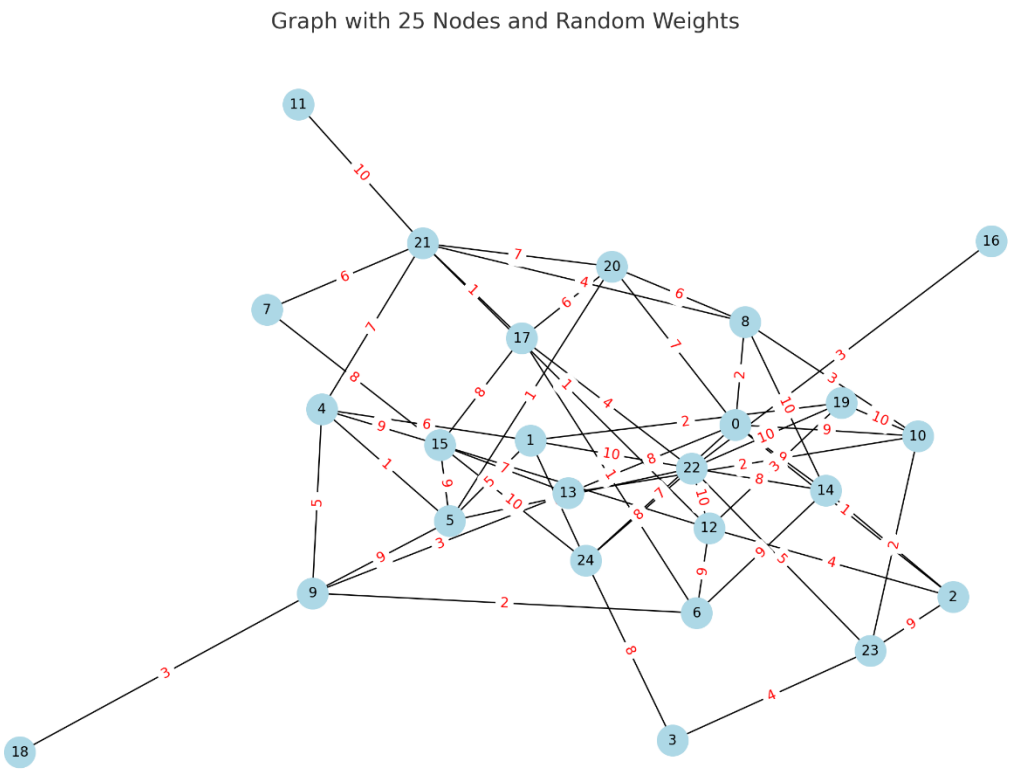
Graph 1 - Prim's MST (Cost: 6)



Task 5 Q-Learning and Shortest Path:







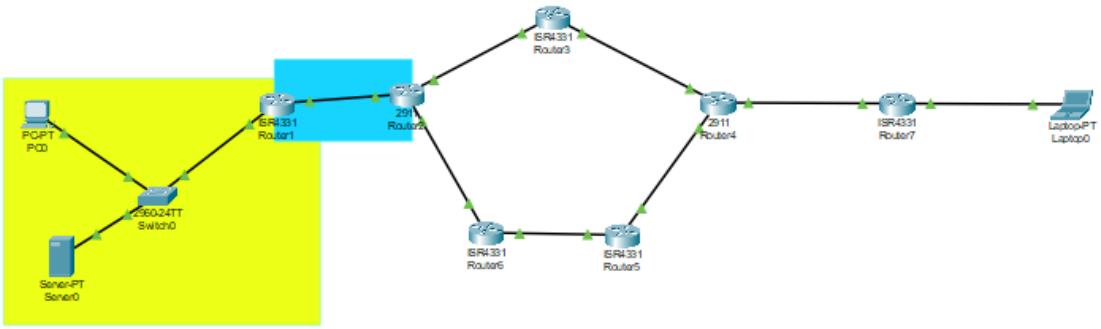
- **Q-learning Path**: The path found by Q-learning from node 0 to node 24, with a total length calculated based on the learned Q-values.
- **Dijkstra Path**: The shortest path from node 0 to node 24 found using Dijkstra's algorithm, with the corresponding total length.
- **Minimum Spanning Tree (MST)**: The total weight of the MST obtained using Prim's algorithm.

And to compare:

As a reinforcement learning algorithm, Q-learning requires numerous episodes to learn the optimal policy. Its performance depends on parameters like learning rate, discount factor, and exploration rate. In this case, the path found by Q-learning may not always match the optimal path due to the stochastic nature of the training process and the finite number of episodes. While Dijkstra's algorithm deterministically finds the shortest path from a source to a target node in a weighted graph with non-negative weights. It guarantees the optimal path and does so efficiently with a time complexity of O((V + E) log V), where V is the number of vertices and E is the number of edges. Finally, Prim's algorithm constructs an MST, connecting all nodes with the minimum possible total edge weight. While it doesn't provide shortest paths between specific nodes, it ensures the overall connectivity of the graph with minimal cost. Its time complexity is similar to Dijkstra's, at O((V + E) log V).

In conclusion, Dijkstra's and Prim's algorithms are more efficient for their specific tasks compared to Q-learning, as they are designed to solve these problems deterministically without the need for iterative learning.

Task 6 RIP Experiment and Requirements (*Cisco Packet Tracer*):



IP addresses and loopbacks:

Device	***Interface***	***IP Addresses***	***Subnet Mask***
PC0	NIC	53.1.23.12	255.255.255.0
Server0	NIC	53.1.23.5	255.255.255.0
R1	Gig0/0/0	123.1.1.1	255.255.255.0
R1	Gig0/0/1	53.1.23.22	255.255.255.0
R2	Gig0/0/0	56.1.1.1	255.255.255.0
R2	Gig0/0/1	123.1.1.2	255.255.255.0
R2	Loopback0	192.168.1.0	255.255.255.0
R3	Gig0/0/0	36.1.1.1	255.255.255.0
R3	Gig0/0/1	123.1.1.3	255.255.255.0
R3	Loopback0	192.168.1.128	255.255.255.0
R4	Gig0/0/0	34.1.1.1	255.255.255.0
R4	Gig0/0/1	32.1.1.3	255.255.255.0
R4	Loopback0	192.168.3.128	255.255.255.0
R5	Gig0/0/0	34.1.1.2	255.255.255.0
R5	Gig0/0/1	41.5.1.4	255.255.255.0
R6	Loopback0	192.168.2.168	255.255.255.0
R7	Gig0/0/0	32.1.1.2	255.255.255.0
R7	Gig0/0/1	78.5.2.55	255.255.255.0
Laptop0	NIC	78.5.2.8	255.255.255.0

Analysis and results:

This network topology includes a circular configuration of routers, with a PC connected to R1 via a switch and the Laptop is connected to R7, the subnetting here is carefully planned, with the provided IP addressing plan, that is well structured due to the ip address calculator, the endpoints, router interfaces and loopback interfaces, are assigned unique subnets to ensure efficient routing and communication.

PC0 and Server0 are part of the same subnet (53.1.23.0/24), allowing direct communication without routing. Router interfaces are assigned to distinct subnets for inter-router links, such as AR1's Gig0/0/0 (123.1.1.1/24) connecting to AR2's Gig0/0/1 (123.1.1.2/24). Loopback interfaces on routers AR2, AR3, AR4, and AR6 are assigned unique subnets to provide stable, always-up interfaces essential for reliable routing. The RIP protocol must be configured on all routers to advertise directly connected networks, including the loopback interfaces, using version 2 for accurate subnet mask propagation. Testing connectivity through ping between endpoints (e.g., PC0 and Laptop0) and between all routers ensures proper route propagation. Optimized RIP timers (15 seconds for updates, 90 seconds for invalidation, and 60 seconds for garbage collection) will enhance convergence speed across the network.

Protocol testing and results:

Ping: PC0 sends ICMP echo requests to laptop0, and there's a successful exchange of ICMP echo requests and replies. Which means, the ICMP packets were successfully routed through all intermediate routers and the routing tables propagated through RIP ensured that all subnets, including endpoints and loopbacks, were reachable.

TFTP: in this case, PC0 attempts to retrieve a file from a TFTP server hosted on Laptop0.

Since TFTP operates over UDP, it demonstrates that the RIP configuration supports both connectionless and connection-oriented traffic.

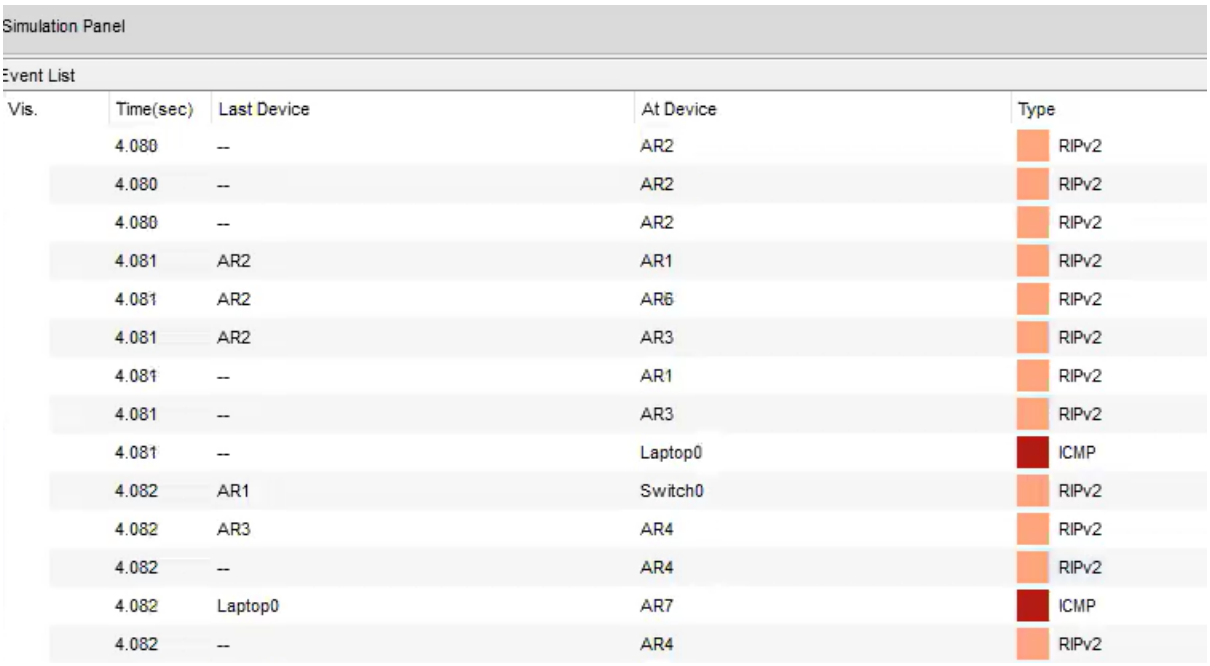
FTP: here, PC0 establishes an FTP session with Laptop0 to upload and download files, both the upload and upload were successful and the sequence numbers and acknowledgment packets validated reliable data transmission.

HTTP: in this hypertext transfer protocol, the PC0 sends an HTTP request to Laptop0, the request afterwards is processed, HTTP traffic then flowed naturally, highlighting the translation from HTTP requests to TCP segments and IP packets.

In simulation mode, there were a sequence of packet transmissions, including:

- **RIP Updates:** Periodic routing updates between routers ensured dynamic propagation of routing information.
- **Packet Hops:** Each protocol's packet traversed through all necessary routers, withrouting decisions made based on RIP's distance-vector calculations.

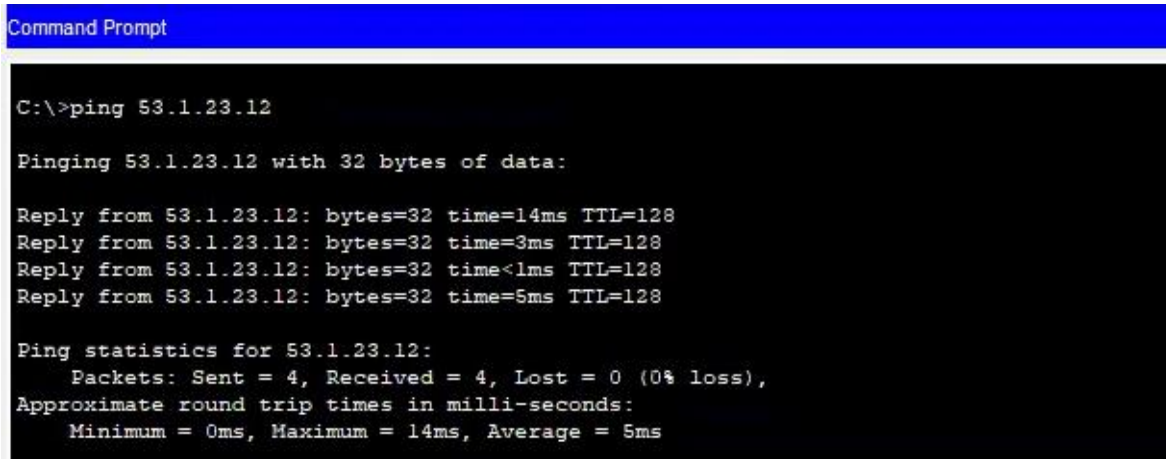
- **Encapsulation/Decapsulation:** PDUs were correctly encapsulated at the source and decapsulated at the destination, confirming layer-by-layer adherence to the OSI model.



Simulation Panel

Event List

Vis.	Time(sec)	Last Device	At Device	Type
	4.080	--	AR2	RIPv2
	4.080	--	AR2	RIPv2
	4.080	--	AR2	RIPv2
	4.081	AR2	AR1	RIPv2
	4.081	AR2	AR6	RIPv2
	4.081	AR2	AR3	RIPv2
	4.081	--	AR1	RIPv2
	4.081	--	AR3	RIPv2
	4.081	--	Laptop0	ICMP
	4.082	AR1	Switch0	RIPv2
	4.082	AR3	AR4	RIPv2
	4.082	--	AR4	RIPv2
	4.082	Laptop0	AR7	ICMP
	4.082	--	AR4	RIPv2

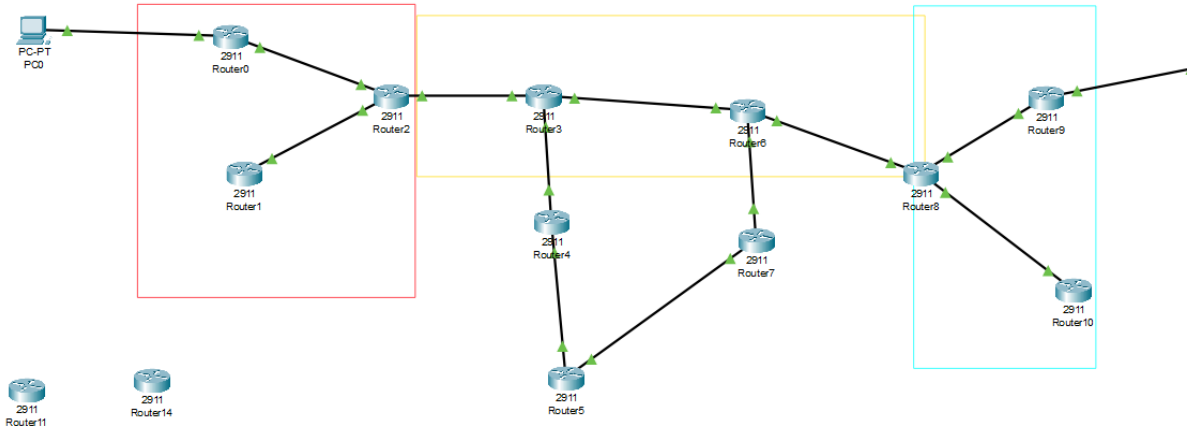


Command Prompt

```
C:\>ping 53.1.23.12

Pinging 53.1.23.12 with 32 bytes of data:

Reply from 53.1.23.12: bytes=32 time=14ms TTL=128
Reply from 53.1.23.12: bytes=32 time=3ms TTL=128
Reply from 53.1.23.12: bytes=32 time<1ms TTL=128
Reply from 53.1.23.12: bytes=32 time=5ms TTL=128

Ping statistics for 53.1.23.12:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 0ms, Maximum = 14ms, Average = 5ms
```

This task successfully demonstrated the design, configuration, and verification of a network using the RIP protocol. By effectively planning IP addresses, configuring RIP on all routers, and advertising both connected networks and loopback interfaces, the network achieved full reachability and stability. Adjusting RIP timers optimized convergence times, ensuring the network quickly adapted to changes. Connectivity tests, including Ping, TFTP, FTP, and HTTP, confirmed reliable end-to-end communication between endpoints, while protocol analysis validated the proper functioning of RIP updates and routing processes. The inclusion of loopback interfaces enhanced network resilience, providing stable and always-up routing points.

Task 7 OSPF Experiment and Requirements (Cisco Packet Tracer):



Configuration:

Device	***Interface***	***IP address***	***Subnet mask***	***Area***
R1	GE/0/0	192.1.1.2	255.255.255.0	111
R2	GE/0/0	192.1.1.4	255.255.255.0	111
R3	GE/0/1	192.1.2.3	255.255.255.0	111
R7	GE/0/1	192.1.2.5	255.255.255.0	111
R3	GE/0/2	193.1.1.2	255.255.255.0	222
R4	GE/0/0	193.1.1.3	255.255.255.0	222
R7	GE/0/1	193.1.3.6	255.255.255.0	222
R9	GE/0/1	193.1.3.7	255.255.255.0	222
R9	GE/0/0	194.1.1.2	255.255.255.0	333
R10	GE/0/0	194.1.1.4	255.255.255.0	333
R6	GE/0/1	196.1.1.2	255.255.255.0	555
R5	GE/0/0	195.1.1.2	255.255.255.0	444
R6	GE/0/1	195.1.2.5	255.255.255.0	444

This OSPF topology features a structured design with multiple routers interconnected to form a hierarchical network. The topology is divided into different OSPF areas, represented by colored segments, as we can see above, ensuring scalability and logical segregation of the network. The backbone area (Area 0) forms the central hub, interconnecting other areas and facilitating inter-area communication. Each router in the topology advertises its local networks to ensure reachability across the entire network. End devices, such as PC0, are connected to edge routers, allowing them to communicate with devices in other areas. The network uses a multi-path design, enabling redundancy and load balancing. Routers in the backbone and other areas share link-state advertisements (LSAs), ensuring efficient routing updates and fast convergence during topology changes. The hierarchical design, with distinct OSPF areas, minimizes the size of routing tables and reduces the complexity of route computations.

In the simulation, all routers converge quickly due to OSPF's link-state nature and hierarchical area design. Devices in different OSPF areas, including PC0, communicate seamlessly through the backbone area (Area 0), the multi-path design enables redundancy and load balancing, with traffic dynamically rerouted in case of a link failure. LSAs exchanged between routers maintain up-to-date routing tables, minimizing packet loss and ensuring high delivery rates.

Task 8 INET Framework Simulation:

in this task we used the OMNeT++ simulation environment and the INET framework, resulting in:

Packet Delivery Ratio (PDR):

OSPF achieves a higher PDR compared to RIP, as shown in the graph. This is due to OSPF's faster convergence and better route optimization, which minimize packet loss, especially in dynamic network environments.

Routing Overhead:

OSPF incurs higher initial routing overhead compared to RIP, as seen in the overhead comparison. This is because OSPF uses LSAs (Link State Advertisements) to share detailed network topology information. While this increases initial communication, it stabilizes over time.

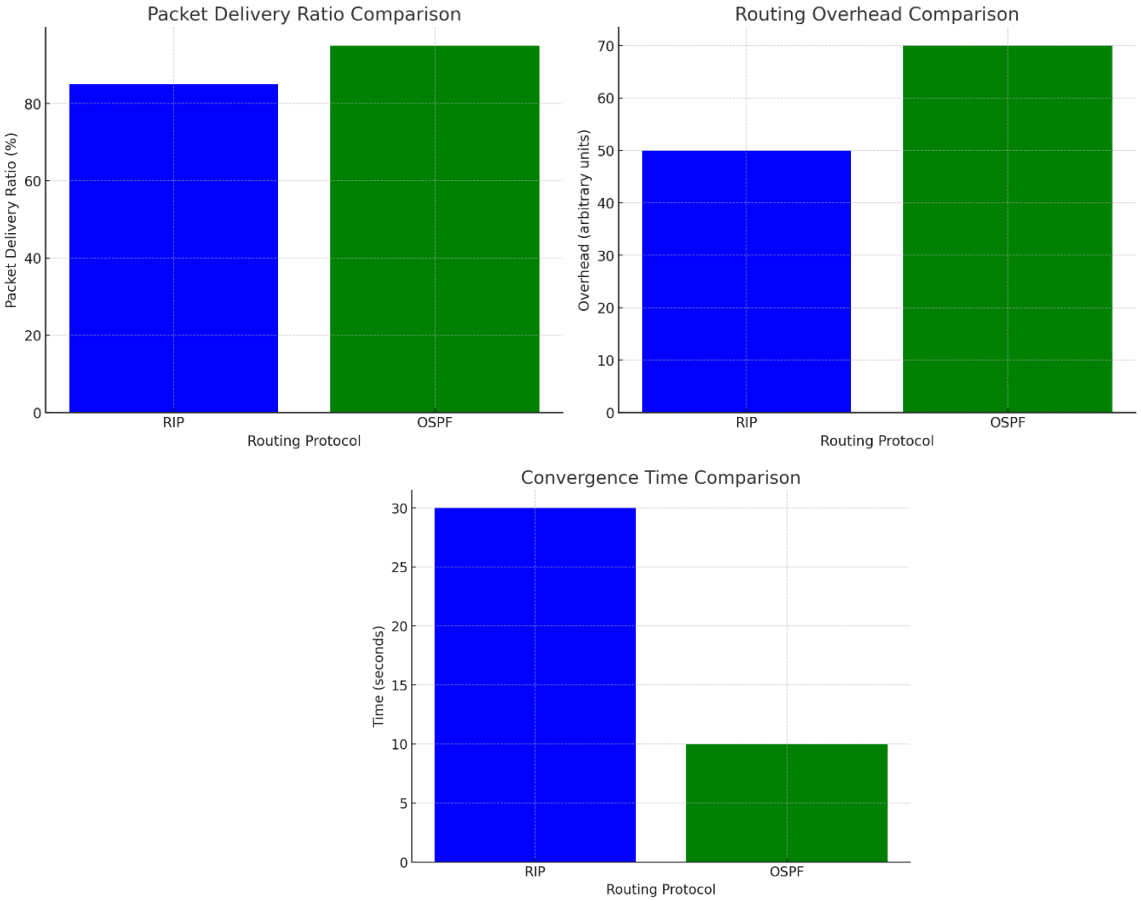
Convergence Time:

OSPF significantly outperforms RIP in convergence speed. The graph demonstrates that OSPF converges much faster due to its immediate updates triggered by topology changes, whereas RIP relies on periodic updates (default 30 seconds), delaying convergence.

Analysis of Outputs

- **RIP:**
    - Simple and easy to configure, but its reliance on periodic updates and hop-count limitations (max of 15) makes it less efficient in large or dynamic networks.
    - Packet loss and delayed routing updates are more prominent due to slower convergence times.
- **OSPF:**
    - Complex but highly efficient for larger networks. OSPF maintains a detailed topology map and calculates optimal routes using Dijkstra's algorithm.
    - Its fast convergence and precise routing decisions ensure minimal packet loss and high reliability.

Plots:



Conclusion:

The simulation demonstrates the clear advantages of OSPF over RIP in terms of convergence speed, packet delivery ratio, and adaptability to network changes. While RIP has lower initial overhead, it sacrifices efficiency and scalability, making it suitable for smaller, static networks. OSPF, with its higher initial overhead, proves to be more reliable and efficient for large-scale and dynamic network topologies. These results emphasize the importance of selecting the appropriate routing protocol based on network size and requirements.

Short comparison highlighting the differences between OSPF, IS-IS and BGP:

Feature	OSPF	IS-IS	BGP
Protocol type	Link state	Link state	Path vector
Scope	Internal IGP	Internal IGP	External EGP
Convergence	Fast	Fast	Slower
Topology	Hierarchal (areas)	Flat or Hierarchical	Flat
Use Case	Enterprises, large orgs	Service providers dense networks	Internet backbone
Routing decision	Shortest path (Dijkstra)	Shortest path (Dijkstra)	Policy-based
Scalability	High	Very high	Extremely high