21/01/2024

Telecommunication Software

Report 2

Jehanne Sarrazin

230AEM053

Task 1 - Python ADT stack and graphs:

1-4:

For this task I did exercise 1 to 4 included in a python notebook. You can find it at the end of this report and in the folder "TASK1" as well as the python notebook in a .ipynb file. I used Jupyter to run it. For every of my practical task in general, you can also access everything from my GitHub repository: https://github.com/Niennaaa/TelecommunicationSoftware

5 – Q-Learning example:

I downloaded the GitHub linked on ORTUS and started to work on it using VSCode. I started to analyse how it works:

First, we have a main file "main.py" It takes in a CVS file in the form three columns: sourceNode, destinationNode, and weight. It then uses a function get_dict to create a graph from the CSV.

The graph reads as follows:

A = All nodes read as sourceNode
Z = All nodes read as destinationNode
Weight = All weights read
A_Z_dict = {nth node: [list of all its neighbouring node]}

Next it calls the function InitialR and initialQ.

InitialR: Similar to A_Z_dict but with the information of weight: {nth node : [list of all its neighbouring node and their associated weight.
InitialQ: Takes InitialR and replace all weight for 100 (understood as the infinite)

Then in main, sets main parameters for the AI:

```
alpha = 0.7 # learning rate
epsilon = 0.1 #greedy policy
n_episodes = 1000
```

And finally calls the function getresult:

```
def get_result(R,Q,alpha,epsilon,n_episodes,start,end):
    Q = Q_routing(R,Q,alpha,epsilon,n_episodes,start,end)
    nodes = get_best_nodes(Q,start,end)
    graph = get_best_net(Q,nodes)
    route_len = len(get_route(Q,start,end))
    routes = get_all_best_routes(graph,start,end,route_len+1)
    result = count_routes(routes)
    ###
    ends_find = []
    for i in range(len(routes)):
        ends_find.append(routes[i][-1])
    ends_find = list(set(ends_find))
    ###
    cost = []
    for i in routes:
        cost.append(get_cost(R,i))
    Counter(cost)
    return {"nodes":nodes,
            "graph":graph,
            "ends_find":ends_find,
            "cost":dict(Counter(cost)),
            "routes_number":result['routes_number'],
            "all_routes":result['all_routes']}
```

Which itself calls the other following functions:

Q_routing: implements the Q-routing algorithm, which is a reinforcement learning method for finding the optimal path in a network. It runs a loop for n_episodes and updates the Q-table using the update_Q function, which is a matrix that stores the expected rewards for each state-action pair. It also uses the get_min_state functions which returns the state with the minimum value in a dictionary, and the get_key_of_min_value function which itself returns a list of keys that have the minimum value in a dictionary, to choose the next state based on exploration or exploitation. It finally returns the final Q-table as output.

get_best_nodes: Explores the graph represented by the Q dictionary to find the best path from the start node to any node in the end list using a breadth-first search and returns a list of nodes representing the best path from `start` to any node in the `end` list.

get_best_net: For each node in the input list, it finds the nodes in the graph that are both connected to the current node and present in the input list and returns a dictionary where keys are nodes, and values are lists of nodes representing the best network connections for each node in the input list.

get_route: It selects the next step at each stage by choosing the action with the maximum value in the Q-table until reaching a node in the end list or encountering a loop in the route and returns a list representing a single route from the start node to any node in the end list based on the Q-table.

get_all_best_routes: It performs a search to find all possible paths within the specified depth from the start node to any node in the end list. It returns a list of all possible paths from the start node to any node in the end list within the given depth.

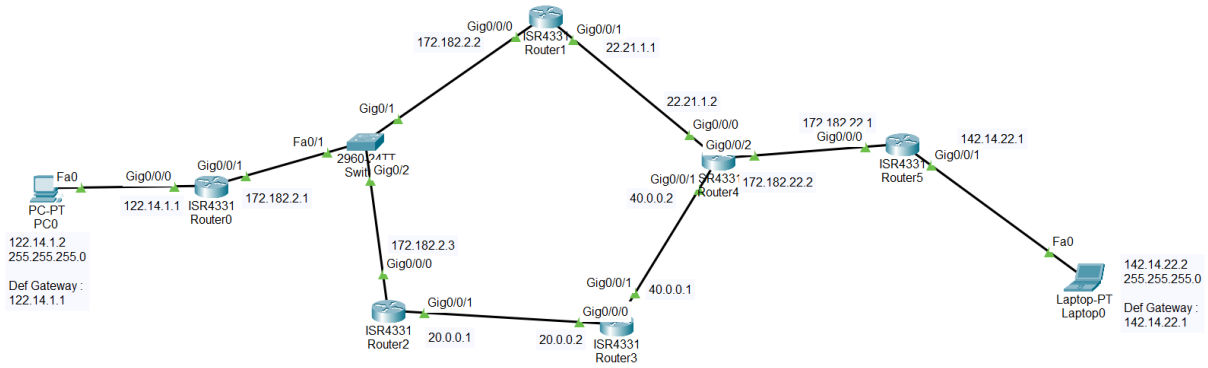
count_routes: Counts the occurrences of end nodes in the list of routes and organizes the routes based on their end nodes. Returns a dictionary with the count of each end node and a dictionary of all routes for each end node.

get_cost: Calculates and returns the total cost of a given route by summing up the costs between consecutive nodes using the provided cost dictionary `R`.

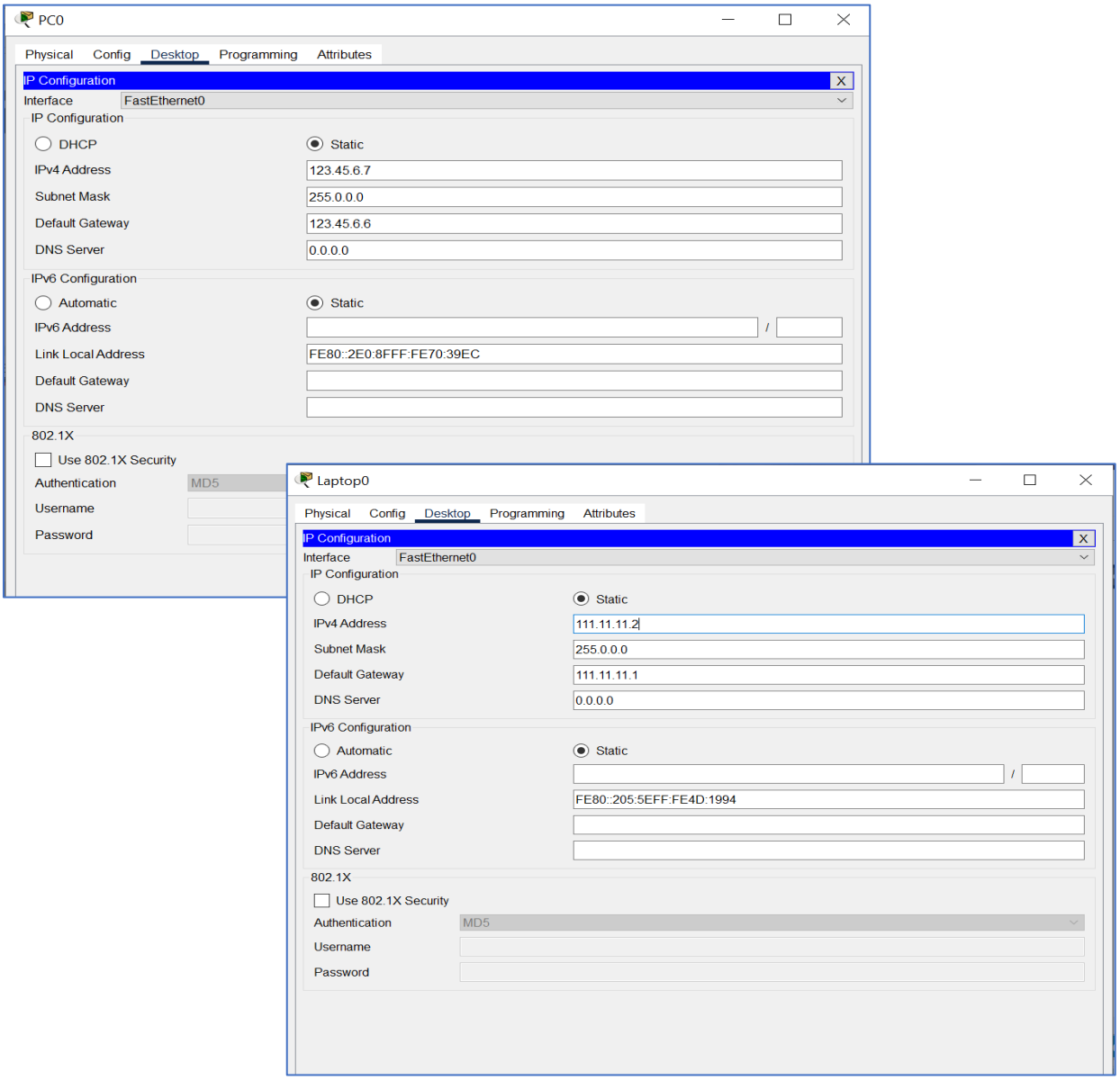
I then tried to apply my own graphs to the algorithm by simply changing the nodes inside the file graph1.csv, but it proved to be very long for graphs above 10 nodes. The creator mentions in the readme file that the program can run in minutes for graphs containing thousands of nodes, so it's possible that I am doing something wrong.

Task 2 – RIP experiment and requirements:

1) I re-created the following RIP configuration as seen in the lecture and changed all IP addresses:



I configured each port with both the Config interface and the CLI commands:



I also realized that the router with 3 ports (router 4) only allowed two ports to be connected at first. I found a way around it on the internet to turn on the third port: -> turn off the router from the physical interface -> connect the GLC-T module (see nav bar on the right) to bottom right port -> turn the router back on (and each ethernet port individually). And now the third port can be connected.

Here is the configuration I used:

Name	Interface	IP Adress	Subnet Mask	Default gateway
PC0	-	**122.14.1.2**	255.255.255.0	**122.14.1.1**
Routeur0	G 0/0/0	**122.14.1.1**	255.255.255.0	N/A
	G 0/0/1	**172.182.2.1**	255.255.255.0	N/A
Routeur1	G 0/0/0	**172.182.2.2**	255.255.255.0	N/A
	G 0/0/1	**22.21.1.1**	255.255.255.0	N/A
Routeur2	G 0/0/0	**172.182.2.3**	255.255.255.0	N/A
	G 0/0/1	**20.0.0.1**	255.255.255.0	N/A
Routeur3	G 0/0/0	**20.0.0.2**	255.255.255.0	N/A
	G 0/0/1	**40.0.0.1**	255.255.255.0	N/A
Routeur4	G 0/0/0	**22.21.1.2**	255.255.255.0	N/A
	G 0/0/1	**40.0.0.2**	255.255.255.0	N/A
	G 0/0/2	**172.182.22.2**	255.255.255.0	N/A
Routeur5	G 0/0/0	**172.182.22.1**	255.255.255.0	N/A
	G 0/0/1	**142.14.22.1**	255.255.255.0	N/A
Laptop0	-	**142.14.22.2**	255.255.255.0	**142.14.22.1**

2) I used “show ip route” and “ip route source destination” to link all devices together such as follows:

```
Router>enable
Router#config terminal
Enter configuration commands, one per line.  End with CNTL/Z.
Router(config)#ip route 10.10.10.2 255.255.255.0 20.20.20.1
%Inconsistent address and mask
Router(config)#ip route 10.10.10.0 255.255.255.0 20.20.20.1
Router(config)#show ip route
                ^
% Invalid input detected at '^' marker.

Router(config)#end
Router#
%SYS-5-CONFIG_I: Configured from console by console
show ip route
Codes: L - local, C - connected, S - static, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2, E - EGP
       i - IS-IS, L1 - IS-IS level-1, L2 - IS-IS level-2, ia - IS-IS inter area
       * - candidate default, U - per-user static route, o - ODR
       P - periodic downloaded static route

Gateway of last resort is not set

     10.0.0.0/24 is subnetted, 1 subnets
S       10.10.10.0/24 [1/0] via 20.20.20.1
     20.0.0.0/8 is variably subnetted, 2 subnets, 2 masks
C       20.20.20.0/24 is directly connected, GigabitEthernet0/0/0
L       20.20.20.2/32 is directly connected, GigabitEthernet0/0/0
     30.0.0.0/8 is variably subnetted, 2 subnets, 2 masks
C       30.30.30.0/24 is directly connected, GigabitEthernet0/0/1
L       30.30.30.1/32 is directly connected, GigabitEthernet0/0/1

Router#
```

This way, all my devices can now ping each other correctly.

Task 3 – OSPF Experiment and requirement:

Here are some screenshots and comments following the steps explained in the OSPF_Basic_Lab PDF file.

1) Configuring the routers: changing hostname and passwords

```
Router>enable
Router#config terminal
Enter configuration commands, one per line.  End with CNTL/Z.
Router(config)#hostname R3
R3(config)#line console 0
R3(config-line)#password cisco
R3(config-line)#login
R3(config-line)#exit
R3(config)#line vty 0 4
R3(config-line)#password cisco
R3(config-line)#login
R3(config-line)#exit
R3(config)#enable secret class
R3(config)#exit
R3#
%SYS-5-CONFIG_I: Configured from console by console

R3#copy running-config startup-config
Destination filename [startup-config]?
Building configuration...
[OK]
R3#
```

2) Disabled DNS lookup by first doing “config terminal” then “no ip domain-lookup”.

3) Configured R1, R2 and R3 interfaces from the configuration view

4) This is the output:

```
R1#show ip interface brief
Interface              IP-Address      OK? Method Status                Protocol
FastEthernet0/0        172.16.1.17     YES manual up                    up
FastEthernet1/0        unassigned      YES unset  administratively down down
Serial2/0              192.168.10.1    YES manual up                    up
Serial3/0              192.168.10.5    YES manual up                    up
FastEthernet4/0        unassigned      YES unset  administratively down down
FastEthernet5/0        unassigned      YES unset  administratively down down
R1#
```

5) Configured ethernet interface of PC1, PC2 and PC3.

Task: Configure OSPF on the R1, R2 and R3 Routers:

Configuring the network statement for the LAN network:

```
R2(config)#
R2(config)#router ospf 1
R2(config-router)#network 10.10.10.0 0.0.0.255 area 0
R2(config-router)#network 192.168.10.0 0.0.0.3 area 0
R2(config-router)#
```

Configuring the router to advertise the other networks:

```
Enter configuration commands, one per line.  End with CN
R2(config)#
R2(config)#router ospf 1
R2(config-router)#network 10.10.10.0 0.0.0.255 area 0
R2(config-router)#network 192.168.10.0 0.0.0.3 area 0
R2(config-router)#network 192.168.10.8 0.0.0.3 area 0
R2(config-router)#end
R2#
%SYS-5-CONFIG_I: Configured from console by console

R2#
```

```
R3#configure terminal
Enter configuration commands, one per line.  End with CNTL/Z.
R3(config)#
R3(config)#router ospf 1
R3(config-router)#network 172.16.1.32 0.0.0.7 area 0
R3(config-router)#network 192.168.10.4 0.0.0.3 area 0
R3(config-router)#network 192.168.10.8 0.0.0.3 area 0
R3(config-router)#
00:25:10: %OSPF-5-ADJCHG: Process 1, Nbr 192.168.10.9 on Serial3/0 from LOADING to FULL, Loading
Done

R3(config-router)#end
R3#
%SYS-5-CONFIG_I: Configured from console by console

R3#
```

Task: Configure OSPF Router IDs:

What is the router ID for R1? 192.168.10.5
What is the router ID for R2? 192.168.10.9
What is the router ID for R3? 192.168.10.10

I used “show id ospf” and got the following output:

```
R3#show ip ospf
 Routing Process "ospf 1" with ID 192.168.10.10
 Supports only single TOS(TOS0) routes
 Supports opaque LSA
 SPF schedule delay 5 secs, Hold time between two SPFs 10 secs
 Minimum LSA interval 5 secs. Minimum LSA arrival 1 secs
 Number of external LSA 0. Checksum Sum 0x000000
 Number of opaque AS LSA 0. Checksum Sum 0x000000
 Number of DCbitless external and opaque AS LSA 0
 Number of DoNotAge external and opaque AS LSA 0
 Number of areas in this router is 1. 1 normal 0 stub 0 nssa
 External flood list length 0
    Area BACKBONE(0)
        Number of interfaces in this area is 3
        Area has no authentication
        SPF algorithm executed 3 times
        Area ranges are
        Number of LSA 2. Checksum Sum 0x0125eb
        Number of opaque link LSA 0. Checksum Sum 0x000000
        Number of DCbitless LSA 0
        Number of indication LSA 0
        Number of DoNotAge LSA 0
        Flood list length 0

R3#
```

2) Loopback interface:

```
R3(config)#interface loopback 0

R3(config-if)#
%LINK-5-CHANGED: Interface Loopback0, changed state to up

%LINEPROTO-5-UPDOWN: Line protocol on Interface Loopback0, changed state to up

R3(config-if)#ip address 10.3.3.3 255.255.255.255
R3(config-if)#
```

3) Reloaded routers:

```
R3#show ip ospf
 Routing Process "ospf 1" with ID 10.3.3.3
 Supports only single TOS(TOS0) routes
 Supports opaque LSA
 SPF schedule delay 5 secs, Hold time between two SPFs 10 secs
 Minimum LSA interval 5 secs. Minimum LSA arrival 1 secs
 Number of external LSA 0. Checksum Sum 0x000000
 Number of opaque AS LSA 0. Checksum Sum 0x000000
 Number of DCbitless external and opaque AS LSA 0
 Number of DoNotAge external and opaque AS LSA 0
 Number of areas in this router is 1. 1 normal 0 stub 0 nssa
 External flood list length 0
    Area BACKBONE(0)
        Number of interfaces in this area is 3
        Area has no authentication
        SPF algorithm executed 2 times
        Area ranges are
        Number of LSA 3. Checksum Sum 0x01599d
        Number of opaque link LSA 0. Checksum Sum 0x000000
        Number of DCbitless LSA 0
        Number of indication LSA 0
        Number of DoNotAge LSA 0
        Flood list length 0

R3#
```

When the router is reloaded, what is the router ID for R1? 10.1.1.1
When the router is reloaded, what is the router ID for R2? 10.2.2.2
When the router is reloaded, what is the router ID for R3? 10.3.3.3

4) “show ip ospf neighbors” gives the following outputs for each router:

R1 :

```
R1#show ip ospf neighbor


Neighbor ID     Pri   State           Dead Time   Address         Interface
10.2.2.2          0   FULL/  -        00:00:37    192.168.10.2    Serial2/0
10.3.3.3          0   FULL/  -        00:00:39    192.168.10.6    Serial3/0
```

R2 :

```
R2#show ip ospf neighbor


Neighbor ID     Pri   State           Dead Time   Address         Interface
10.3.3.3          0   FULL/  -        00:00:39    192.168.10.10   Serial3/0
10.1.1.1          0   FULL/  -        00:00:34    192.168.10.1    Serial2/0
R2#
```

R3 :

```
%SYS-5-CONFIG_I: Configured from console by console
show ip ospf neighbor


Neighbor ID     Pri   State           Dead Time   Address         Interface
10.1.1.1          0   FULL/  -        00:00:39    192.168.10.5    Serial2/0
10.2.2.2          0   FULL/  -        00:00:30    192.168.10.9    Serial3/0
R3#
```

5) Using router-id :

```
R1#
R1#config terminal
Enter configuration commands, one per line.  End with CNTL/Z.
R1(config)#router ospf 1
R1(config-router)#router-id 10.4.4.4
R1(config-router)#Reload or use "clear ip ospf process" command, for this to take effect


R1(config-router)#end
R1#
%SYS-5-CONFIG_I: Configured from console by console

R1#clear ip ospf process
Reset ALL OSPF processes? [no]:
```

6)

```
R1#clear ip ospf process
Reset ALL OSPF processes? [no]: yes

R1#
00:07:58: %OSPF-5-ADJCHG: Process 1, Nbr 10.2.2.2 on Serial2/0 from FULL to DOWN, Neighbor Down: Adjacency forced to reset

00:07:58: %OSPF-5-ADJCHG: Process 1, Nbr 10.2.2.2 on Serial2/0 from FULL to DOWN, Neighbor Down: Interface down or detached

00:07:58: %OSPF-5-ADJCHG: Process 1, Nbr 10.3.3.3 on Serial3/0 from FULL to DOWN, Neighbor Down: Adjacency forced to reset

00:07:58: %OSPF-5-ADJCHG: Process 1, Nbr 10.3.3.3 on Serial3/0 from FULL to DOWN, Neighbor Down: Interface down or detached

00:08:13: %OSPF-5-ADJCHG: Process 1, Nbr 10.2.2.2 on Serial2/0 from LOADING to FULL, Loading Done

00:08:15: %OSPF-5-ADJCHG: Process 1, Nbr 10.3.3.3 on Serial3/0 from LOADING to FULL, Loading Done
```

And as expected we have:

```
R2#show ip ospf neighbor


Neighbor ID     Pri   State           Dead Time   Address         Interface
10.3.3.3          0   FULL/  -        00:00:39    192.168.10.10   Serial3/0
10.4.4.4          0   FULL/  -        00:00:34    192.168.10.1    Serial2/0
R2#
```

Task: Verify OSPF Operation:

1) Here are R1's neighbours:

```
R1#show ip ospf neighbor


Neighbor ID     Pri   State           Dead Time   Address         Interface
10.2.2.2          0   FULL/  -        00:00:37    192.168.10.2    Serial2/0
10.3.3.3          0   FULL/  -        00:00:39    192.168.10.6    Serial3/0
```

2) IP protocols:

```
R1#show ip protocols

Routing Protocol is "ospf 1"
  Outgoing update filter list for all interfaces is not set
  Incoming update filter list for all interfaces is not set
  Router ID 10.4.4.4
  Number of areas in this router is 1. 1 normal 0 stub 0 nssa
  Maximum path: 4
  Routing for Networks:
    172.16.1.16 0.0.0.15 area 0
    192.168.10.0 0.0.0.3 area 0
    192.168.10.4 0.0.0.3 area 0
  Routing Information Sources:
    Gateway         Distance      Last Update
    10.1.1.1             110      00:15:46
    10.2.2.2             110      00:06:31
    10.3.3.3             110      00:06:28
    10.4.4.4             110      00:06:28
  Distance: (default is 110)

R1#
```

Task: Examine OSPF Routes in the Routing Tables:

1)

```
R1#show ip route
Codes: C - connected, S - static, I - IGRP, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2, E - EGP
       i - IS-IS, L1 - IS-IS level-1, L2 - IS-IS level-2, ia - IS-IS inter area
       * - candidate default, U - per-user static route, o - ODR
       P - periodic downloaded static route

Gateway of last resort is not set

     10.0.0.0/8 is variably subnetted, 2 subnets, 2 masks
C       10.1.1.1/32 is directly connected, Loopback0
O       10.10.10.0/24 [110/65] via 192.168.10.2, 00:15:22, Serial2/0
     172.16.0.0/16 is variably subnetted, 2 subnets, 2 masks
C       172.16.1.16/28 is directly connected, FastEthernet0/0
O       172.16.1.32/29 [110/65] via 192.168.10.6, 00:15:22, Serial3/0
     192.168.10.0/30 is subnetted, 3 subnets
C       192.168.10.0 is directly connected, Serial2/0
C       192.168.10.4 is directly connected, Serial3/0
O       192.168.10.8 [110/128] via 192.168.10.2, 00:15:22, Serial2/0
                     [110/128] via 192.168.10.6, 00:15:22, Serial3/0

R1#
```

Task: Configure OSPF Cost:

1) This is done right above.
2) I'm actually using serial2/0 instead of serial0/0/0 because I only had serial2 and serial3 :

```
R1#show interface serial2/0
Serial2/0 is up, line protocol is up (connected)
  Hardware is HD64570
  Internet address is 192.168.10.1/30
  MTU 1500 bytes, BW 128 Kbit, DLY 20000 usec,
     reliability 255/255, txload 1/255, rxload 1/255
  Encapsulation HDLC, loopback not set, keepalive set (10 sec)
  Last input never, output never, output hang never
  Last clearing of "show interface" counters never
  Input queue: 0/75/0 (size/max/drops); Total output drops: 0
  Queueing strategy: weighted fair
  Output queue: 0/1000/64/0 (size/max total/threshold/drops)
     Conversations  0/0/256 (active/max active/max total)
     Reserved Conversations 0/0 (allocated/max allocated)
     Available Bandwidth 96 kilobits/sec
  5 minute input rate 259 bits/sec, 0 packets/sec
  5 minute output rate 217 bits/sec, 0 packets/sec
     252 packets input, 25288 bytes, 0 no buffer
     Received 0 broadcasts, 0 runts, 0 giants, 0 throttles
     0 input errors, 0 CRC, 0 frame, 0 overrun, 0 ignored, 0 abort
     368 packets output, 30552 bytes, 0 underruns
     0 output errors, 0 collisions, 1 interface resets
     0 output buffer failures, 0 output buffers swapped out
     0 carrier transitions
     DCD=up  DSR=up  DTR=up  RTS=up  CTS=up

R1#
R1#
```

3)Bandwidth command for R1 and R2:

R1 :

```
R1(config)#interface serial2/0
R1(config-if)#bandwith 64
                     ^
% Invalid input detected at '^' marker.

R1(config-if)#bandwidth 64
R1(config-if)#interface serial3/0
R1(config-if)#bandwidth 64
R1(config-if)#
```

R2 :

```
R2(config)#
R2(config)#interface serial2/0
R2(config-if)#bandwidth 64
R2(config-if)#interface serial3/0
R2(config-if)#bandwidth 64
R2(config-if)#
```

4) Output :

```
R1#show ip ospf interface

FastEthernet0/0 is up, line protocol is up
  Internet address is 172.16.1.17/28, Area 0
  Process ID 1, Router ID 10.4.4.4, Network Type BROADCAST, Cost: 1
  Transmit Delay is 1 sec, State DR, Priority 1
  Designated Router (ID) 10.4.4.4, Interface address 172.16.1.17
  No backup designated router on this network
  Timer intervals configured, Hello 10, Dead 40, Wait 40, Retransmit 5
    Hello due in 00:00:04
  Index 1/1, flood queue length 0
  Next 0x0(0)/0x0(0)
  Last flood scan length is 1, maximum is 1
  Last flood scan time is 0 msec, maximum is 0 msec
  Neighbor Count is 0, Adjacent neighbor count is 0
  Suppress hello for 0 neighbor(s)
Serial2/0 is up, line protocol is up
  Internet address is 192.168.10.1/30, Area 0
  Process ID 1, Router ID 10.4.4.4, Network Type POINT-TO-POINT, Cost: 1562
  Transmit Delay is 1 sec, State POINT-TO-POINT,
  Timer intervals configured, Hello 10, Dead 40, Wait 40, Retransmit 5
    Hello due in 00:00:00
  Index 2/2, flood queue length 0
  Next 0x0(0)/0x0(0)
  Last flood scan length is 1, maximum is 1
  Last flood scan time is 0 msec, maximum is 0 msec
  Neighbor Count is 1 , Adjacent neighbor count is 1
    Adjacent with neighbor 10.2.2.2
  Suppress hello for 0 neighbor(s)
Serial3/0 is up, line protocol is up
  Internet address is 192.168.10.5/30, Area 0
  Process ID 1, Router ID 10.4.4.4, Network Type POINT-TO-POINT, Cost: 1562
  Transmit Delay is 1 sec, State POINT-TO-POINT,
  Timer intervals configured, Hello 10, Dead 40, Wait 40, Retransmit 5
    Hello due in 00:00:01
  Index 3/3, flood queue length 0
  Next 0x0(0)/0x0(0)
  Last flood scan length is 1, maximum is 1
  Last flood scan time is 0 msec, maximum is 0 msec
  Neighbor Count is 1 , Adjacent neighbor count is 1
    Adjacent with neighbor 10.3.3.3
  Suppress hello for 0 neighbor(s)
R1#
```

5) Configuration of the cost

```
R3(config)#
R3(config)#interface serial2/0
R3(config-if)#ip ospf cost 1562
R3(config-if)#interface serial3/0
R3(config-if)#ip ospf cost 1562
R3(config-if)#end
R3#
```

6)

```
R3#show ip ospf interface

FastEthernet0/0 is up, line protocol is up
  Internet address is 172.16.1.33/29, Area 0
  Process ID 1, Router ID 10.3.3.3, Network Type BROADCAST, Cost: 1
  Transmit Delay is 1 sec, State DR, Priority 1
  Designated Router (ID) 10.3.3.3, Interface address 172.16.1.33
  No backup designated router on this network
  Timer intervals configured, Hello 10, Dead 40, Wait 40, Retransmit 5
    Hello due in 00:00:00
  Index 1/1, flood queue length 0
  Next 0x0(0)/0x0(0)
  Last flood scan length is 1, maximum is 1
  Last flood scan time is 0 msec, maximum is 0 msec
  Neighbor Count is 0, Adjacent neighbor count is 0
  Suppress hello for 0 neighbor(s)
Serial3/0 is up, line protocol is up
  Internet address is 192.168.10.10/30, Area 0
  Process ID 1, Router ID 10.3.3.3, Network Type POINT-TO-POINT, Cost: 1562
  Transmit Delay is 1 sec, State POINT-TO-POINT,
  Timer intervals configured, Hello 10, Dead 40, Wait 40, Retransmit 5
    Hello due in 00:00:05
  Index 2/2, flood queue length 0
  Next 0x0(0)/0x0(0)
  Last flood scan length is 1, maximum is 1
  Last flood scan time is 0 msec, maximum is 0 msec
  Neighbor Count is 1 , Adjacent neighbor count is 1
    Adjacent with neighbor 10.2.2.2
  Suppress hello for 0 neighbor(s)
Serial2/0 is up, line protocol is up
  Internet address is 192.168.10.6/30, Area 0
  Process ID 1, Router ID 10.3.3.3, Network Type POINT-TO-POINT, Cost: 1562
  Transmit Delay is 1 sec, State POINT-TO-POINT,
  Timer intervals configured, Hello 10, Dead 40, Wait 40, Retransmit 5
    Hello due in 00:00:05
  Index 3/3, flood queue length 0
  Next 0x0(0)/0x0(0)
  Last flood scan length is 1, maximum is 1
  Last flood scan time is 0 msec, maximum is 0 msec
  Neighbor Count is 1 , Adjacent neighbor count is 1
    Adjacent with neighbor 10.4.4.4
  Suppress hello for 0 neighbor(s)
R3#
```

Task: Redistribute an OSPF Default Route

1,2,3) Configuring loopback address and static default route :

```
R1#config terminal
Enter configuration commands, one per line.  End with CNTL/Z.
R1(config)#interface loopback1

R1(config-if)#
%LINK-5-CHANGED: Interface Loopback1, changed state to up

%LINEPROTO-5-UPDOWN: Line protocol on Interface Loopback1, changed state to up

R1(config-if)#ip address 172.30.1.1 255.255.255.252
R1(config-if)#exit
R1(config)#ip route 0.0.0.0 0.0.0.0 loopback1
R1(config)#router ospf 1
R1(config-router)#default-information originate
R1(config-router)#
```

4)

```
R2#show ip route
Codes: C - connected, S - static, I - IGRP, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2, E - EGP
       i - IS-IS, L1 - IS-IS level-1, L2 - IS-IS level-2, ia - IS-IS inter area
       * - candidate default, U - per-user static route, o - ODR
       P - periodic downloaded static route

Gateway of last resort is 192.168.10.1 to network 0.0.0.0

     10.0.0.0/8 is variably subnetted, 2 subnets, 2 masks
C       10.2.2.2/32 is directly connected, Loopback0
C       10.10.10.0/24 is directly connected, FastEthernet0/0
     172.16.0.0/16 is variably subnetted, 2 subnets, 2 masks
O       172.16.1.16/28 [110/1563] via 192.168.10.1, 00:10:24, Serial2/0
O       172.16.1.32/29 [110/1563] via 192.168.10.10, 00:10:24, Serial3/0
     192.168.10.0/30 is subnetted, 3 subnets
C       192.168.10.0 is directly connected, Serial2/0
O       192.168.10.4 [110/3124] via 192.168.10.10, 00:05:33, Serial3/0
                     [110/3124] via 192.168.10.1, 00:05:33, Serial2/0
C       192.168.10.8 is directly connected, Serial3/0
O*E2 0.0.0.0/0 [110/1] via 192.168.10.1, 00:01:05, Serial2/0
```

Task: Configure Additional OSPF Features:

1) Auto-cost bandwidth:

```
R1(config-router)#auto-cost reference-bandwidth 10000
% OSPF: Reference bandwidth is changed.
        Please ensure reference bandwidth is consistent across all routers.
R1(config-router)#
```

2,3)

```
%SYS-5-CONFIG_I: Configured from console by console
show ip route
Codes: C - connected, S - static, I - IGRP, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2, E - EGP
       i - IS-IS, L1 - IS-IS level-1, L2 - IS-IS level-2, ia - IS-IS inter area
       * - candidate default, U - per-user static route, o - ODR
       P - periodic downloaded static route

Gateway of last resort is 0.0.0.0 to network 0.0.0.0

     10.0.0.0/8 is variably subnetted, 2 subnets, 2 masks
C       10.1.1.1/32 is directly connected, Loopback0
O       10.10.10.0/24 [110/25278] via 192.168.10.2, 00:01:06, Serial2/0
     172.16.0.0/16 is variably subnetted, 2 subnets, 2 masks
C       172.16.1.16/28 is directly connected, FastEthernet0/0
O       172.16.1.32/29 [110/25278] via 192.168.10.6, 00:00:17, Serial3/0
     172.30.0.0/30 is subnetted, 1 subnets
C       172.30.1.0 is directly connected, Loopback1
     192.168.10.0/30 is subnetted, 3 subnets
C       192.168.10.0 is directly connected, Serial2/0
C       192.168.10.4 is directly connected, Serial3/0
O       192.168.10.8 [110/50356] via 192.168.10.2, 00:00:56, Serial2/0
S*   0.0.0.0/0 is directly connected, Loopback1

R1#
R1#
R1#show ip ospf neighbor


Neighbor ID     Pri   State           Dead Time   Address         Interface
10.2.2.2          0   FULL/  -        00:00:33    192.168.10.2    Serial2/0
10.3.3.3          0   FULL/  -        00:00:35    192.168.10.6    Serial3/0
R1#
```

4) Configuring OSPF Hello and dead intervals:

```
R1(config)#interface serial2/0
R1(config-if)#ip ospf hello-interval 5
R1(config-if)#ip ospf dead-interval 20
R1(config-if)#
R1(config-if)#
00:40:49: %OSPF-5-ADJCHG: Process 1, Nbr 10.2.2.2 on Serial2/0 from FULL to DOWN, Neighbor Down: Dead timer expired

00:40:49: %OSPF-5-ADJCHG: Process 1, Nbr 10.2.2.2 on Serial2/0 from FULL to DOWN, Neighbor Down: Interface down or detached
```

5) Modifying the timer:

```
Enter configuration commands, one per line.  End with CNTL/Z.
R2(config)#interface serial2/0
R2(config-if)#ip ospf hello-interval 5
R2(config-if)#ip ospf dead-interval 20
R2(config-if)#
01:48:31: %OSPF-5-ADJCHG: Process 1, Nbr 10.4.4.4 on Serial2/0 from LOADING to FULL, Loading
Done

R2(config-if)#end
R2#
%SYS-5-CONFIG_I: Configured from console by console

R2#show ip ospf interface serial2/0

Serial2/0 is up, line protocol is up
  Internet address is 192.168.10.2/30, Area 0
  Process ID 1, Router ID 10.2.2.2, Network Type POINT-TO-POINT, Cost: 25178
  Transmit Delay is 1 sec, State POINT-TO-POINT,
  Timer intervals configured, Hello 5, Dead 20, Wait 20, Retransmit 5
    Hello due in 00:00:03
  Index 3/3, flood queue length 0
  Next 0x0(0)/0x0(0)
  Last flood scan length is 1, maximum is 1
  Last flood scan time is 0 msec, maximum is 0 msec
  Neighbor Count is 1 , Adjacent neighbor count is 1
    Adjacent with neighbor 10.4.4.4
  Suppress hello for 0 neighbor(s)
R2#
```

6)

```
R1#show ip ospf neighbor


Neighbor ID     Pri   State           Dead Time   Address         Interface
10.2.2.2          0   FULL/  -        00:00:19    192.168.10.2    Serial2/0
10.3.3.3          0   FULL/  -        00:00:36    192.168.10.6    Serial3/0
R1#
```

I left the final configuration in the folder “TASK 3”.

After this PDF file, I verified that all Pings were successful:

```
C:\>ping 172.16.1.17

Pinging 172.16.1.17 with 32 bytes of data:

Reply from 172.16.1.17: bytes=32 time<1ms TTL=255
Reply from 172.16.1.17: bytes=32 time<1ms TTL=255
Reply from 172.16.1.17: bytes=32 time<1ms TTL=255
Reply from 172.16.1.17: bytes=32 time<1ms TTL=255

Ping statistics for 172.16.1.17:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 0ms, Maximum = 0ms, Average = 0ms

C:\>ping 10.10.10.1

Pinging 10.10.10.1 with 32 bytes of data:

Reply from 10.10.10.1: bytes=32 time=18ms TTL=254
Reply from 10.10.10.1: bytes=32 time=1ms TTL=254
Reply from 10.10.10.1: bytes=32 time=15ms TTL=254
Reply from 10.10.10.1: bytes=32 time=2ms TTL=254

Ping statistics for 10.10.10.1:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 1ms, Maximum = 18ms, Average = 9ms

C:\>ping 172.16.1.33

Pinging 172.16.1.33 with 32 bytes of data:

Reply from 172.16.1.33: bytes=32 time=12ms TTL=254
Reply from 172.16.1.33: bytes=32 time=9ms TTL=254
Reply from 172.16.1.33: bytes=32 time=22ms TTL=254
Reply from 172.16.1.33: bytes=32 time=1ms TTL=254

Ping statistics for 172.16.1.33:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 1ms, Maximum = 22ms, Average = 11ms
```

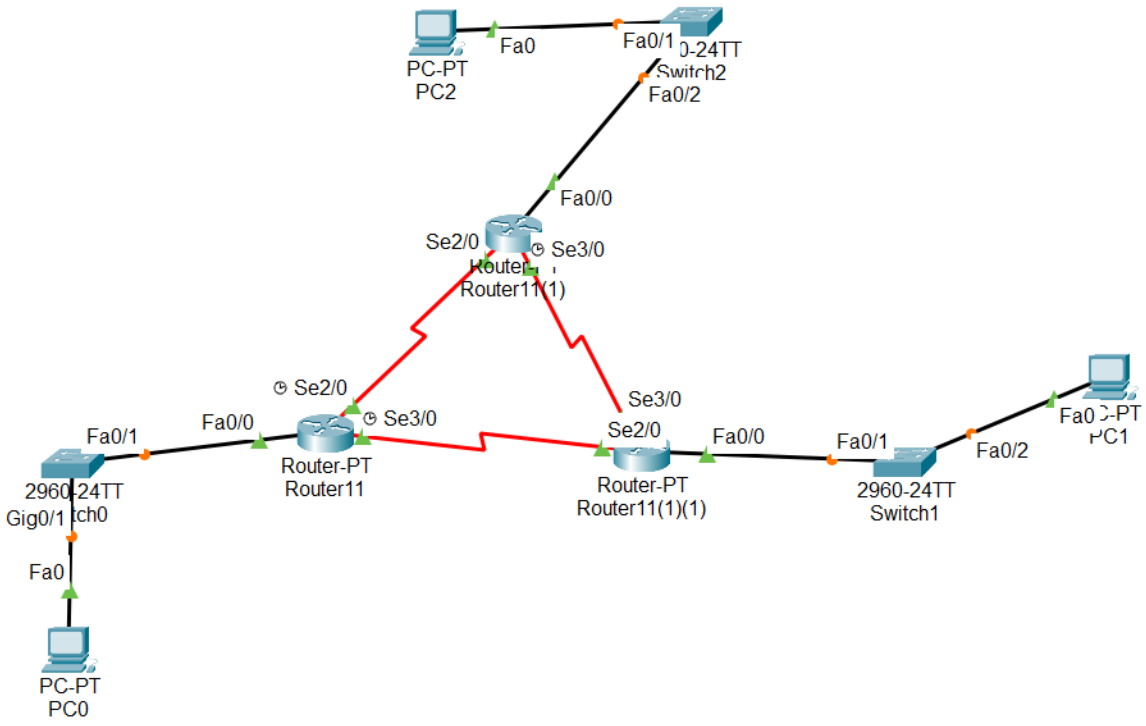
```
C:\>ping 10.10.10.10

Pinging 10.10.10.10 with 32 bytes of data:

Request timed out.
Reply from 10.10.10.10: bytes=32 time=1ms TTL=126
Reply from 10.10.10.10: bytes=32 time=22ms TTL=126
Reply from 10.10.10.10: bytes=32 time=7ms TTL=126

Ping statistics for 10.10.10.10:
    Packets: Sent = 4, Received = 3, Lost = 1 (25% loss),
Approximate round trip times in milli-seconds:
    Minimum = 1ms, Maximum = 22ms, Average = 10ms

C:\>ping 172.16.1.35

Pinging 172.16.1.35 with 32 bytes of data:

Request timed out.
Reply from 172.16.1.35: bytes=32 time=2ms TTL=126
Reply from 172.16.1.35: bytes=32 time=1ms TTL=126
Reply from 172.16.1.35: bytes=32 time=1ms TTL=126

Ping statistics for 172.16.1.35:
    Packets: Sent = 4, Received = 3, Lost = 1 (25% loss),
Approximate round trip times in milli-seconds:
    Minimum = 1ms, Maximum = 2ms, Average = 1ms

C:\>
```

```
C:\>ping 192.168.10.1

Pinging 192.168.10.1 with 32 bytes of data:

Reply from 192.168.10.1: bytes=32 time<1ms TTL=255
Reply from 192.168.10.1: bytes=32 time=15ms TTL=255
Reply from 192.168.10.1: bytes=32 time<1ms TTL=255
Reply from 192.168.10.1: bytes=32 time<1ms TTL=255

Ping statistics for 192.168.10.1:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 0ms, Maximum = 15ms, Average = 3ms

C:\>ping 192.168.10.5

Pinging 192.168.10.5 with 32 bytes of data:

Reply from 192.168.10.5: bytes=32 time<1ms TTL=255
Reply from 192.168.10.5: bytes=32 time<1ms TTL=255
Reply from 192.168.10.5: bytes=32 time<1ms TTL=255
Reply from 192.168.10.5: bytes=32 time<1ms TTL=255

Ping statistics for 192.168.10.5:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 0ms, Maximum = 0ms, Average = 0ms

C:\>ping 192.168.10.2

Pinging 192.168.10.2 with 32 bytes of data:

Reply from 192.168.10.2: bytes=32 time=1ms TTL=254
Reply from 192.168.10.2: bytes=32 time=1ms TTL=254
Reply from 192.168.10.2: bytes=32 time=18ms TTL=254
Reply from 192.168.10.2: bytes=32 time=1ms TTL=254

Ping statistics for 192.168.10.2:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 1ms, Maximum = 18ms, Average = 5ms
```

```
C:\>ping 192.168.10.9

Pinging 192.168.10.9 with 32 bytes of data:

Reply from 192.168.10.9: bytes=32 time=29ms TTL=254
Reply from 192.168.10.9: bytes=32 time=20ms TTL=254
Reply from 192.168.10.9: bytes=32 time=1ms TTL=254
Reply from 192.168.10.9: bytes=32 time=15ms TTL=254

Ping statistics for 192.168.10.9:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 1ms, Maximum = 29ms, Average = 16ms

C:\>ping 192.168.10.6

Pinging 192.168.10.6 with 32 bytes of data:

Reply from 192.168.10.6: bytes=32 time=28ms TTL=254
Reply from 192.168.10.6: bytes=32 time=21ms TTL=254
Reply from 192.168.10.6: bytes=32 time=2ms TTL=254
Reply from 192.168.10.6: bytes=32 time=1ms TTL=254

Ping statistics for 192.168.10.6:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 1ms, Maximum = 28ms, Average = 13ms

C:\>ping 192.168.10.10

Pinging 192.168.10.10 with 32 bytes of data:

Reply from 192.168.10.10: bytes=32 time=40ms TTL=254
Reply from 192.168.10.10: bytes=32 time=22ms TTL=254
Reply from 192.168.10.10: bytes=32 time=26ms TTL=254
Reply from 192.168.10.10: bytes=32 time=3ms TTL=254

Ping statistics for 192.168.10.10:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 3ms, Maximum = 40ms, Average = 22ms
```

And changed the IP as instructed, to have the following configuration:

Device	Interface	IP Address	Subnet Mask	Default Gateway
R1	Fa0/1	139.0.1.3	255.255.128.0	N/A
	S2/0	223.0.0.1	255.255.240.0	N/A
	S3/0	223.0.0.2	255.255.240.0	N/A
R2	Fa0/1	192.0.2.3	255.255.240.0	N/A
	S2/0	223.0.0.3	255.255.240.0	N/A
	S3/0	223.0.0.4	255.255.240.0	N/A
R3	Fa0/1	203.0.113.3	255.255.224.0	N/A
	S2/0	223.0.0.5	255.255.240.0	N/A
	S3/0	223.0.0.6	255.255.240.0	N/A
PC1	NIC	139.0.1.2	255.255.128.0	139.0.1.1
PC2	NIC	192.0.2.2	255.255.192.0	192.0.2.1
PC3	NIC	203.0.113.2	255.255.224.0	203.0.113.1



Annex:

Practical work 2

In [1]:
```
#Requirements
!pip install pythonds
```

```
Requirement already satisfied: pythonds in c:\users\jehanne\anaconda3\lib\site-packa
ges (1.2.1)
WARNING: Ignoring invalid distribution -umpy (c:\users\jehanne\anaconda3\lib\site-pa
ckages)
WARNING: Ignoring invalid distribution -umpy (c:\users\jehanne\anaconda3\lib\site-pa
ckages)
```

In [2]:
```
from pythonds import *
```

Task 1 : Decimal to hexadecimal with Stack

In [3]:
```
#First function
s=Stack()

def fun(decimal): #Converts decimal values to hexadecimal ones using stacks
  if decimal==0 : return "0"

  L = ["A", "B", "C", "D", "E", "F"]
  result = Stack()
  remainder = 0
  newdec = 0

  while True:
    newdec = decimal//16
    remainder = decimal%16
    if remainder ==0 and decimal<16: #we've reached the end of the loop
      break

    if remainder>9 : #above 9, we use number
      remainder = L[remainder-10]

    result.push(remainder)
    decimal = newdec

  strres = ""
  while not result.isEmpty(): #at the end of the loop, we pop out the results
    strres = strres + str(result.pop())
  return strres
```

In [4]:
```
#Test of the function
for k in range(0,64):
  print(str(k) +" : " +fun(k))
```

0 : 0
1 : 1
2 : 2
3 : 3
4 : 4
5 : 5
6 : 6
7 : 7
8 : 8
9 : 9
10 : A
11 : B
12 : C
13 : D
14 : E
15 : F
16 : 10
17 : 11
18 : 12
19 : 13
20 : 14
21 : 15
22 : 16
23 : 17
24 : 18
25 : 19
26 : 1A
27 : 1B
28 : 1C
29 : 1D
30 : 1E
31 : 1F
32 : 20
33 : 21
34 : 22
35 : 23
36 : 24
37 : 25
38 : 26
39 : 27
40 : 28
41 : 29
42 : 2A
43 : 2B
44 : 2C
45 : 2D
46 : 2E
47 : 2F
48 : 30
49 : 31
50 : 32
51 : 33
52 : 34
53 : 35
54 : 36
55 : 37

```
56 : 38
57 : 39
58 : 3A
59 : 3B
60 : 3C
61 : 3D
62 : 3E
63 : 3F
```

Task 2 : Brackets match with Stacks

In [5]:
```
def bracketsCheck(input): #Functions that takes a string in input and uses Stacks
    #to verifies if all brackets are complete
  firstBrackets = Stack()
  brackets = "<({[>)}]"

  for k in input:
    if k in brackets :

      if not firstBrackets.isEmpty():
        last = firstBrackets.peek() #verifies the last brackets seen before
        if match1(last, k):
          firstBrackets.pop() #if it's a match, we move on
        else:
          firstBrackets.push(k) #if it's not, we had it to the stack
      else :
        firstBrackets.push(k)
  return firstBrackets.isEmpty()



def match1(one, two): #For two brackets in input, match1 verifies whether they
    #match or not
  brackets1 = "<({["
  brackets2 = ">)}]"
  for k in range(len(brackets1)):
    if (brackets1[k]==one and brackets2[k]==two) or (brackets2[k]==one and \
      brackets1[k]==two):
      return True
  return False
```

In [6]:
```
test_cases = [
    "(i(nput)[<>])",
    "(((())))",
    "{[()]}",
    "[{(<})]",
    "{[<>]}",
    "(([]))",
    "((()))]",
    "{<[()]}>",
    "([)]",
    "abc",
]
```

```
for test_case in test_cases:
    result = bracketsCheck(test_case)
    print(f"bracketsCheck('{test_case}') = {result}")
```

```
bracketsCheck('(i(nput)[<>])') = True
bracketsCheck('((()))') = True
bracketsCheck('{[()]}') = True
bracketsCheck('[{(<})]') = False
bracketsCheck('{[<>]}') = True
bracketsCheck('(([]))') = True
bracketsCheck('((()))]') = False
bracketsCheck('{<[()]}>') = False
bracketsCheck('([)]') = False
bracketsCheck('abc') = True
```

Task 3 : Dijkstra's algorithm

In [7]:
```
#We're going to read it like :
#Graph :
#[[distance from node 1 to node 1 = 0, from 1 to 2 = 3, from 1 to
#3 = 0 (=no links between 1 and 3), from 1 to 4 : 7],
#[from 2 to 1 = 3, from 2 to 2 = 0, from 2 to 3 = 1, from 2 to 4 =0],
#[3 to 1 =0, 3 to 2 = 1, 3 to 3 = 0, 3 to 4 = 2],
#[4 to 1 = 7, 4 to 2 = 0, 4 to 3=2,, 4 to 4 = 0]]
# so :
# [[0, 3, 0, 7],
#  [3, 0, 1, 0],
#  [0, 1, 0, 2],
#  [7, 0, 2, 0]]


#Requirements
import networkx as nx #to draw the graphs
import numpy as np

class Graph: #As defined in the comment aboce
  def __init__(self, vertex):
    self.nbVertex = vertex
    self.vertex = [[0 for column in range(vertex)] for row in range(vertex)]
    self.distanceFromSource = [np.inf for column in  range(vertex)]
      #calculated distance from the source
    self.visited = [False for column in range(vertex)]
      #whether a node has been visited yet or not


  def listNeightbour(self, node): #list all nodes neighbour to the one of entry
    res = []
    n=0
    for others in self.vertex[node]:
      if others!=0:
        res.append(n)
      n=n+1
    return res

  def resetVisited(self):#self-explanatory
```

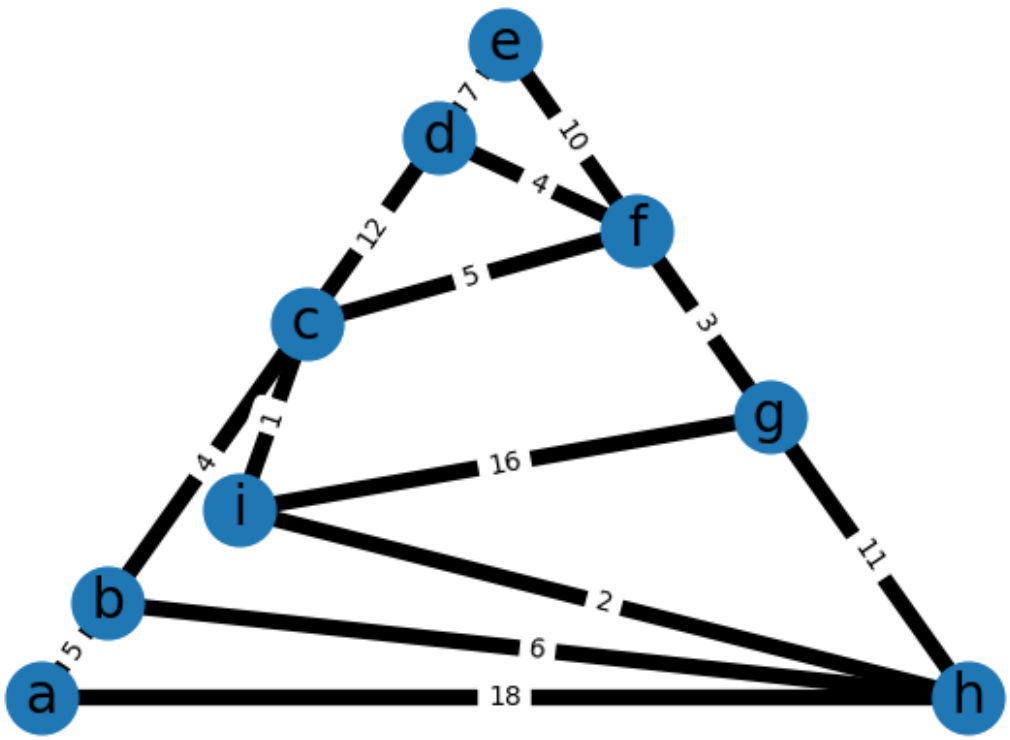
```
    for u in range(self.nbVertex):
      self.visited[u]=False

  def updateDistanceTable(self, node): #update the distance from the source
      #of all nodes neighbour to the one of entry
    for otherNodes in self.listNeightbour(node):
      if self.distanceFromSource[otherNodes]>self.distanceFromSource[node]\
        +self.vertex[node][otherNodes]:
        self.distanceFromSource[otherNodes]=self.distanceFromSource[node]\
          +self.vertex[node][otherNodes]

  def nextNodeToVisit(self): #looks for unvisited node with the smallest
      #distance from the source. Also marks it as visited.
    minDist = np.inf
    res = None
    for nodes in range(self.nbVertex):
      if self.visited[nodes]==False and self.distanceFromSource[nodes]<minDist:
        minDist = self.distanceFromSource[nodes]
        res = nodes
    if res!=None :
      self.visited[res]=True
      return res
    return False



  def dijkstra(self, source): #the algorithm

    self.distanceFromSource[source]=0 #we start at the source
    self.visited[source]=True
    self.updateDistanceTable(source) #input all distances from the source
    nextNode = self.nextNodeToVisit() #find the next node to use
      #(smallest distance from the source)
    while nextNode!=False: #while there are other nodes to visit
      self.updateDistanceTable(nextNode) #we update the distance
        #from the source if we find better
      nextNode = self.nextNodeToVisit() #then look for another node to visit
    return self.distanceFromSource #returns all the distances from the source
      #to the n^th node




#Let's test it quickly with an example :
G = Graph(9)
G.vertex = [
    [0, 5, 0, 0, 0, 0, 0, 18, 0],
    [5, 0, 4, 0, 0, 0, 0, 6, 0],
    [0, 4, 0, 12, 0, 5, 0, 0, 1],
    [0, 0, 12, 0, 7, 4, 0, 0, 0],
    [0, 0, 0, 7, 0, 10, 0, 0, 0],
    [0, 0, 5, 4, 10, 0, 3, 0, 0],
    [0, 0, 0, 0, 0, 3, 0, 11, 16],
    [18, 6, 0, 0, 0, 0, 11, 0, 2],
    [0, 0, 1, 0, 0, 0, 16, 2, 0]]
```

```
print(G.dijkstra(0))
```

```
[0, 5, 9, 18, 24, 14, 17, 11, 10]
```

In [8]:

```
#plotting :
import matplotlib.pyplot as plt
import networkx as nx

def generateNodeName(nbNodes, node): #generate the name of a node
    #based on the total nb of nodes and which node is being named
  alphabet="abcdefghijklmnopqrstuvwxyz" #it goes a,b,c,d etc if the nb
    #of node is <=26, it goes a1, b1...a2, b2 etc if not.
  if nbNodes<=26:
    return alphabet[node]
  return alphabet[node%26]+str(node//26)

def generateGraph(graph, title):#generate a visual graph based
    #on the format of my class Graph
  G1 = nx.Graph() #building the graph
  for i in range(graph.nbVertex):
    for j in range(graph.nbVertex):
      if graph.vertex[i][j]!=0:
        G1.add_edge(generateNodeName(graph.nbVertex, i),\
            generateNodeName(graph.nbVertex, j), weight=graph.vertex[i][j])


  pos = nx.planar_layout(G1)  # positions for all nodes, seed for reproducibility

  # nodes
  nx.draw_networkx_nodes(G1, pos, node_size=700)

  # edges
  nx.draw_networkx_edges(G1, pos, width=6)

  # node labels
  nx.draw_networkx_labels(G1, pos, font_size=20, font_family="sans-serif")
  # edge weight labels
  edge_labels = nx.get_edge_attributes(G1, "weight")
  nx.draw_networkx_edge_labels(G1, pos, edge_labels)


  ax = plt.gca()
  ax.margins(0.08)
  plt.axis("off")
  plt.title(title)
  plt.tight_layout()
  plt.show()

#using the previous example :
generateGraph(G, "Example")
```

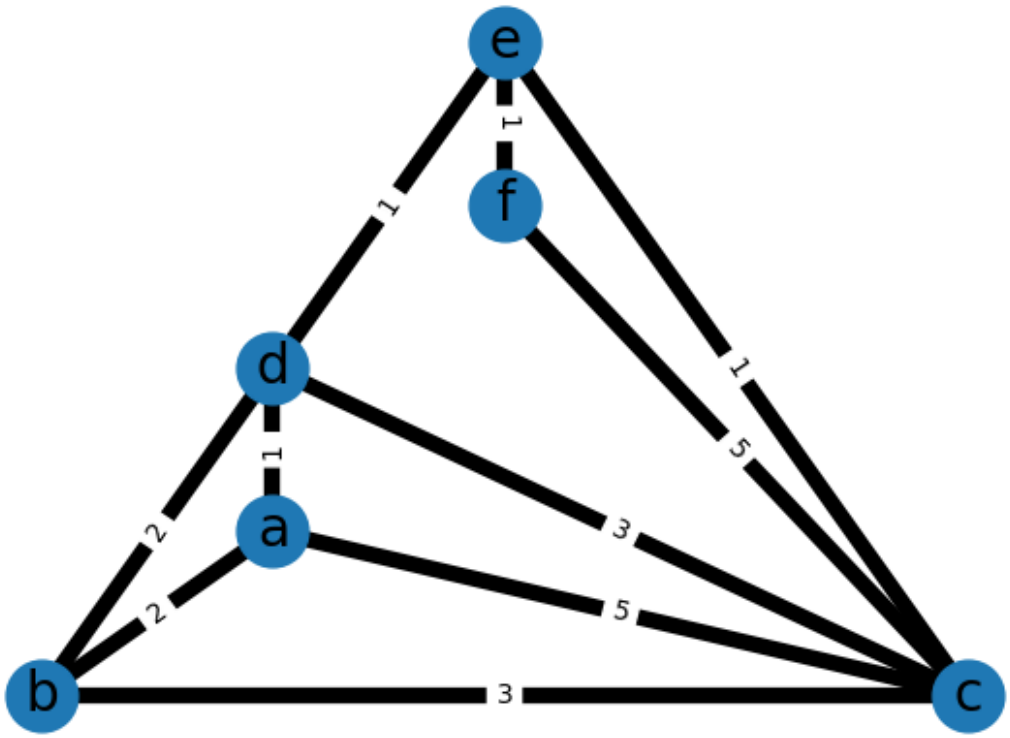


In [9]:
```
#Now let's do it for the figure seen in class :

#Implementation :
fig1 = [[0, 2, 5, 1, 0, 0],
        [2, 0, 3, 2, 0, 0],
        [5, 3, 0, 3, 1, 5],
        [1, 2, 3, 0, 1, 0],
        [0, 0, 1, 1, 0, 1],
        [0, 0, 5, 0, 1, 0]]
G1 = Graph(len(fig1))
G1.vertex = fig1

fig2 = [[0, 3, 0, 0, 1, 6],
        [3, 0, 4, 0, 1, 0],
        [0, 4, 0, 9, 0, 0],
        [0, 0, 9, 0, 1, 0],
        [1, 1, 0, 1, 0, 2],
        [6, 0, 0, 0, 2, 0]]
G2 = Graph(len(fig2))
G2.vertex = fig2

fig3 = [[0, 3, 0, 0, 0, 0, 0, 0],
        [3, 0, 0, 0, 0, 0, 0, 0],
        [0, 0, 0, 3, 2, 0, 0, 0],
        [0, 0, 0, 0, 0, 1, 2, 0],
        [4, 0, 0, 0, 0, 1, 0, 0],
        [0, 5, 0, 1, 0, 0, 0, 4],
        [0, 0, 0, 0, 0, 0, 0, 1],
```

```
        [0, 0, 0, 0, 0, 4, 0, 0]]
G3 = Graph(len(fig3))
G3.vertex = fig3

fig4 = [[0, 2, 3, 0, 0, 0, 0],
        [2, 0, 1, 1, 4, 0, 0],
        [3, 1, 0, 0, 0, 5, 0],
        [0, 1, 0, 0, 1, 0, 0],
        [0, 4, 0, 1, 0, 1, 0],
        [0, 0, 5, 0, 1, 0, 1],
        [0, 0, 0, 0, 0, 1, 0]]
G4 = Graph(len(fig4))
G4.vertex = fig4

fig5 = [[0, 5, 10, 0],
        [5, 0, 3, 11],
        [10, 3, 0, 2],
        [0, 11, 2, 0]]
G5 = Graph(len(fig5))
G5.vertex = fig5

def representation(graph, source, titre):
  print("\n"+ titre + " : ")
  generateGraph(graph, titre)
  print("\nSolution : ")
  print(graph.dijkstra(source))


representation(G1, 0, "Fig1")
representation(G2, 0, "Fig2")
representation(G3, 0, "Fig3")
representation(G4, 0, "Fig4")
representation(G5, 0, "Fig5")
```

```
Fig1 :
```

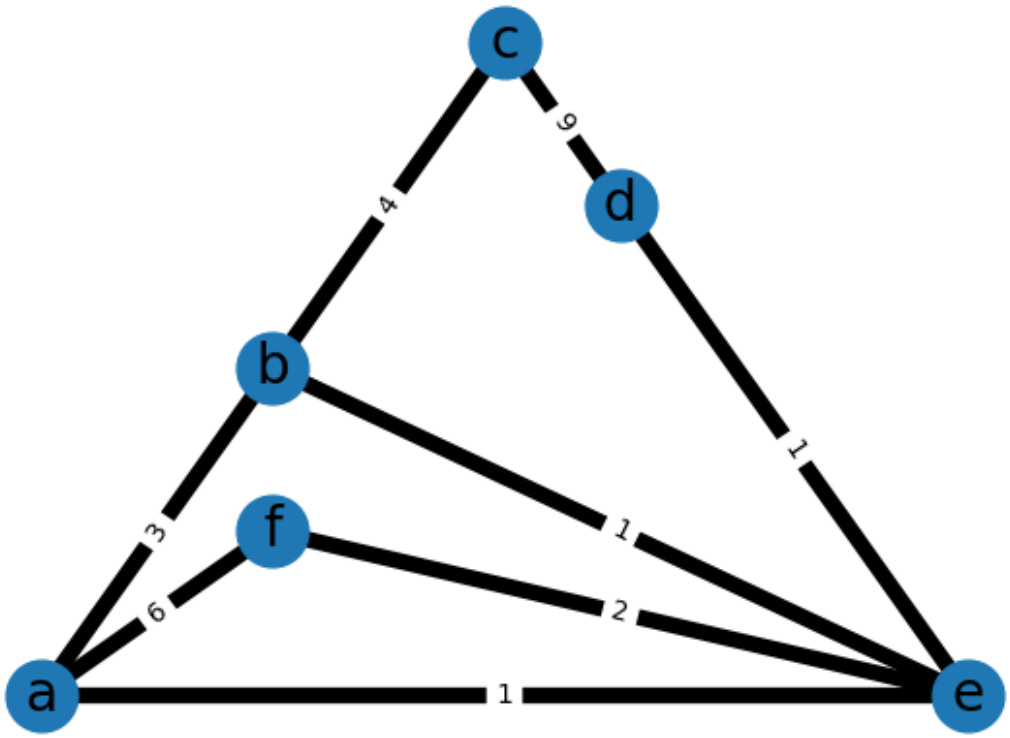
Fig1



Solution :
[0, 2, 3, 1, 2, 3]

Fig2 :

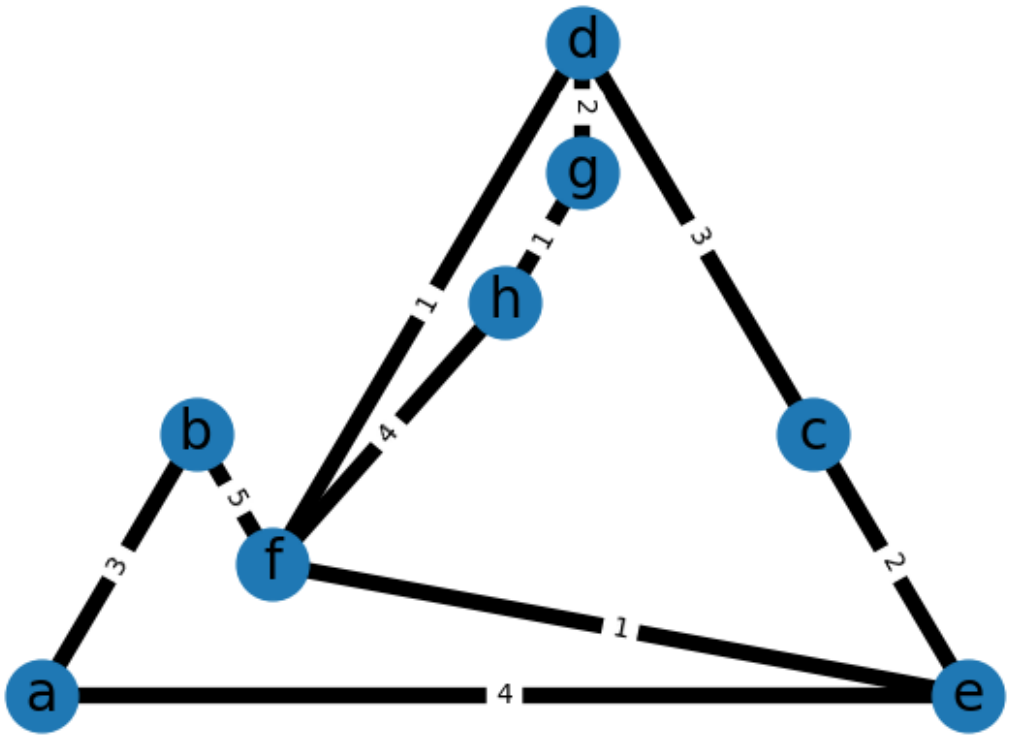
Fig2



Solution :
[0, 2, 6, 2, 1, 3]

Fig3 :

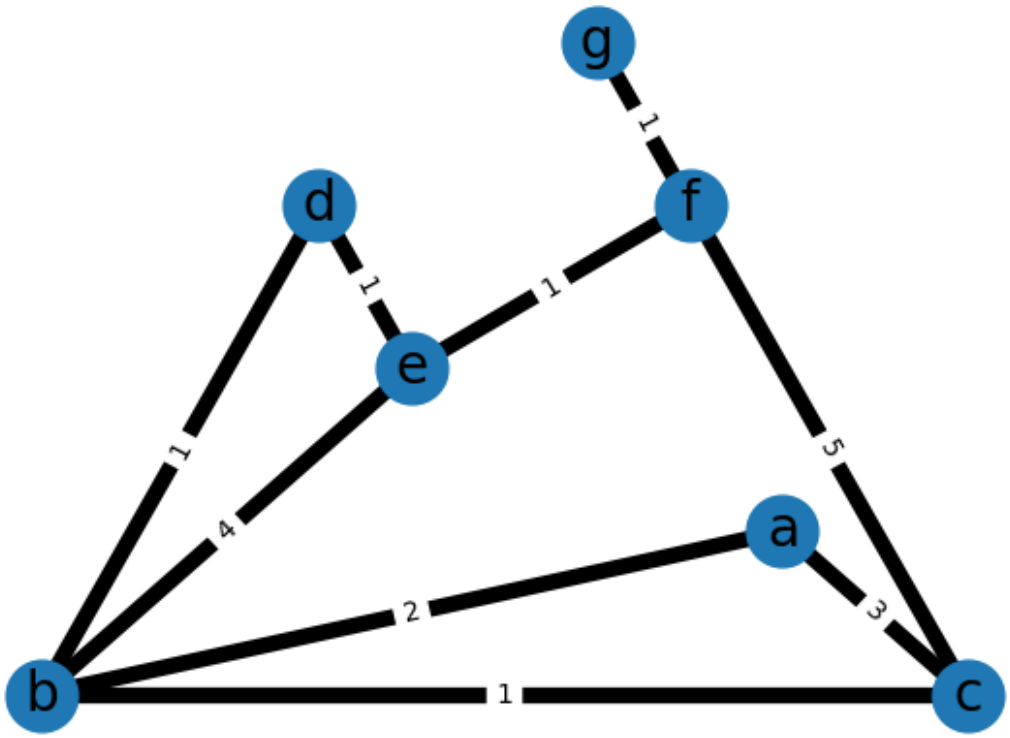
Fig3



Solution :
[0, 3, inf, inf, inf, inf, inf, inf]

Fig4 :

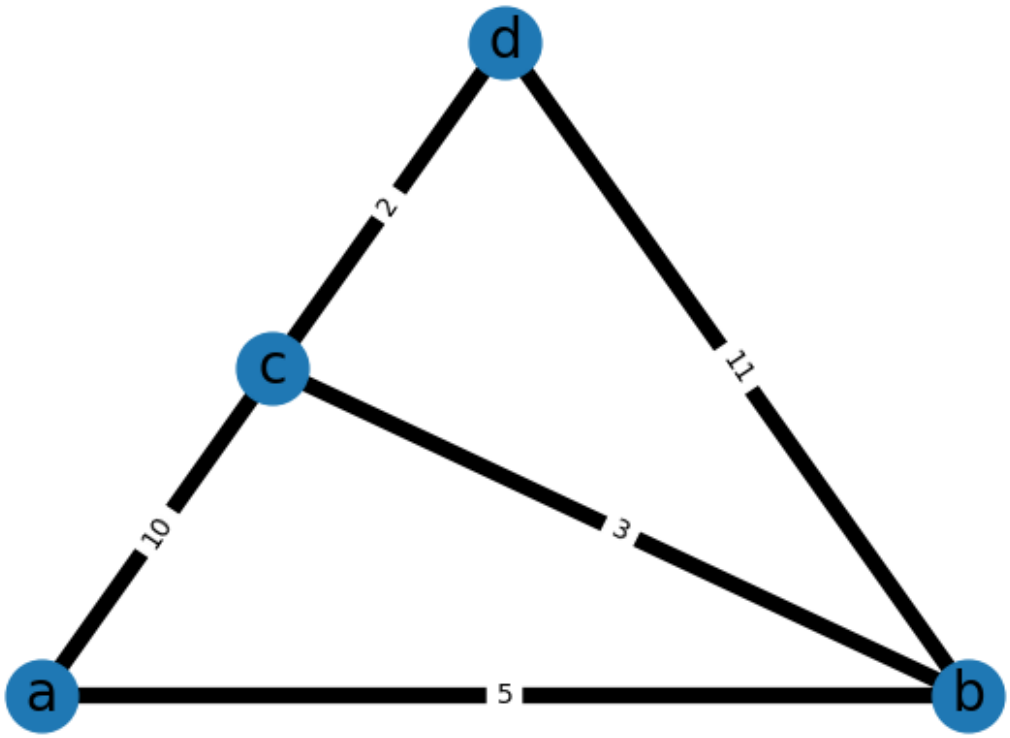
Fig4



Solution :
[0, 2, 3, 3, 4, 5, 6]

Fig5 :

Fig5




```
Solution :
[0, 5, 8, 10]
```

Task 4 : Prim's Algorithm

In [10]:

```
#NOW for Prim algorithm
import random as rd

#we'll just use the same class Graph as input
def prim(graph, source=None):

  if source==None : #in the algorithm, the start can be chosen randomly.
      #I let the user choose as we are comparing it to Dijkstra.
    source = rd.randint(0, graph.nbVertex)
  graph.visited[source]=True
  origin = source #im not the brightest so i changed the source later

  #initialiation of variables
  table2 = [[np.inf for column in range(graph.nbVertex)] for \
            row in range(graph.nbVertex)]
  seen = []
  mini = np.inf
  n = None

  #finiding the next node
  for node in range(graph.nbVertex):
    table2[source][node] = graph.vertex[source][node]
    if graph.vertex[source][node]<mini and  graph.vertex[source][node]!=0:
```

```
    mini = graph.vertex[source][node]
    n = node

#first path taken
seen.append([source, n, mini])
nextNode = n
graph.visited[n] = True

#filling in the table of reference of what's left to explore
for u in range(graph.nbVertex):
      table2[nextNode][u] = graph.vertex[nextNode][u]

while nextNode !=None:

  mini = np.inf
  nextNode = None

  for i in range(graph.nbVertex):
    for j in range(graph.nbVertex):

      if graph.visited[i] and graph.visited[j] and table2[i][j]!=np.inf:
        table2[i][j] = np.inf
      if table2[i][j]!=np.inf and table2[i][j]!=0 and table2[i][j]<mini:

        if not graph.visited[j] :
          source = i
          nextNode = j
          mini = table2[i][j]
          break
  if nextNode!=None :
    seen.append([source, nextNode, mini])
    graph.visited[nextNode] = True
    for u in range(graph.nbVertex):
      table2[nextNode][u] = graph.vertex[nextNode][u]

#print(seen) #seen is the output : it's a list of the path taken
  #in the form of [from node1, to node2, distance node1<->node2]

#now that we have the "shortest" tree we want to calculate
  #all distance from the source to the n^th node
#so as to format the answer in the same way as Dijkstra's algorithm :

res = [0 for i in range (graph.nbVertex)] #0 the distance, false whether
  #the node as been fully linked to the origin or not yet
node = 0
for step in seen:
  res[step[1]]=res[step[1]]+step[2]+res[step[0]]
  node = node +1

for node in range(graph.nbVertex):
  if res[node]==0 and node!=origin:
    res[node]=np.inf #removing the values that can't be reached
#print(res)
return res
```

```
#first test
G = Graph(9)
G.vertex = [
    [0, 5, 0, 0, 0, 0, 0, 18, 0],
    [5, 0, 4, 0, 0, 0, 0, 6, 0],
    [0, 4, 0, 12, 0, 5, 0, 0, 1],
    [0, 0, 12, 0, 7, 4, 0, 0, 0],
    [0, 0, 0, 7, 0, 10, 0, 0, 0],
    [0, 0, 5, 4, 10, 0, 3, 0, 0],
    [0, 0, 0, 0, 0, 3, 0, 11, 16],
    [18, 6, 0, 0, 0, 0, 11, 0, 2],
    [0, 0, 1, 0, 0, 0, 16, 2, 0]]
prim(G,0)
```

Out[10]: [0, 5, 9, 18, 25, 14, 17, 12, 10]

In [11]:
```
#Now with the 5 graph we've seen :

G1.resetVisited()
print(prim(G1,0))

G2.resetVisited()
print(prim(G2,0))

G3.resetVisited()
print(prim(G3,0))

G4.resetVisited()
print(prim(G4,0))

G5.resetVisited()
print(prim(G5,0))
```

```
[0, 2, 3, 1, 2, 3]
[0, 2, 6, 2, 1, 3]
[0, 3, inf, inf, inf, inf, inf, inf]
[0, 2, 3, 3, 4, 5, 6]
[0, 5, 8, 10]
```

Comparison of Prim's and Djikstra's algorithm

In [17]:
```
#We will now compare Dijkstra's algorithm and Prim alorithm :
import timeit

#Based on their results :
def bestDistance(graphList):
  resDji = 0
  resPri = 0

  for graph in graphList :
    graph.resetVisited()
    resDji = resDji+sum(graph.dijkstra(0))
    graph.resetVisited()
    resPri = resPri+sum(prim(graph,0))
```

```
  return "Dijkstra total weight : "+str(resDji)\
    +" - Prim total weight: "+str(resPri)

#Based on their time :
def bestTime(graphList):
    for graph in graphList:
        graph.resetVisited()

    dij_time = timeit.timeit(lambda: [graph.dijkstra(0) for graph in graphList], nu

    for graph in graphList:
        graph.resetVisited()

    pri_time = timeit.timeit(lambda: [prim(graph, 0) for graph in graphList], numbe

    return f"Dijkstra total time: {dij_time:.6f} seconds - Prim total time: {pri_ti



graphList = [G1, G2, G4, G5] #removed G3 bc starting at 0 raise
#infinity and I dont want to make longer functions to handle it
print(bestDistance(graphList))
print(bestTime(graphList))

#When we run, we can see that both Djikstra and Prim have the
#same weight in distance, but Djikstra is slightly faster than Prim
```

```
Dijkstra total weight : 71 - Prim total weight: 71
Dijkstra total time: 0.000077 seconds - Prim total time: 0.000372 seconds
```

When we run, we can see that both Djikstra and Prim have the same weight in distance, but Djikstra is slightly faster than Prim