HW3

Question 1

Answer 1:

```
r0 = 1
a = 0.7
realrtt = 0.5
finalrtt = r0
for i in range(20):
    finalrtt = a * finalrtt + (1 - a) * realrtt
finalrtt
```

With this python simulation code, we can get the RTT-timeout(20) = 0.5004

Answer 2:

After changing $\alpha = 0.5$, RTT-timeout(20) = 0.5000

After changing $\alpha = 0.95$, RTT-timeout(20) = 0.679242961204271

We can find out when α is bigger, then the convergence is slower, when it is smaller, the convergence is faster, with the assumption that the real rtt never changes.

Question 2

1.

Time	**FIFO**		**Highest Priority**		**Round Robin**		**WFQ**	
	Packet	Delay	Packet	Delay	Packet	Delay	Packet	Delay
1	1	1	1	1	1	1	1	1
2	2	2	3	1	2	2	4	1
3	3	2	2	3	4	2	2	3
4	4	3	5	1	3	3	3	3
5	6	3	7	2	6	3	6	3
6	5	3	9	1	5	3	9	1
7	7	4	4	6	7	4	7	4
8	9	3	6	6	8	3	10	1
9	8	4	11	1	11	1	5	6

Time	FIFO		Highest Priority		Round Robin		WFQ	
10	10	3	8	5	9	5	12	2
11	11	3	10	4	12	3	8	6
12	12	4	12	4	10	5	11	4

2. Delay
   FIFO: 2.9167
   HP: 2.9167 (low priority: 1.167 high priority: 4.667) RR: 2.9167 (class 1: 3.289 class 2: 2.4) WFQ: 2.9167 (class 0: 2.25 class 1: 1.75 class 2: 4.75)
3. The average delays for them are the same. Class in HP with higher priority will have smaller delay and class in WFQ with higher weight tend to have smaller delay.

Question 3

Answer 1:

For subnet A, 128 addresses suffice;

For subnet B, 32 addresses suffice;

For subnet C, 32 addresses suffice;

So,

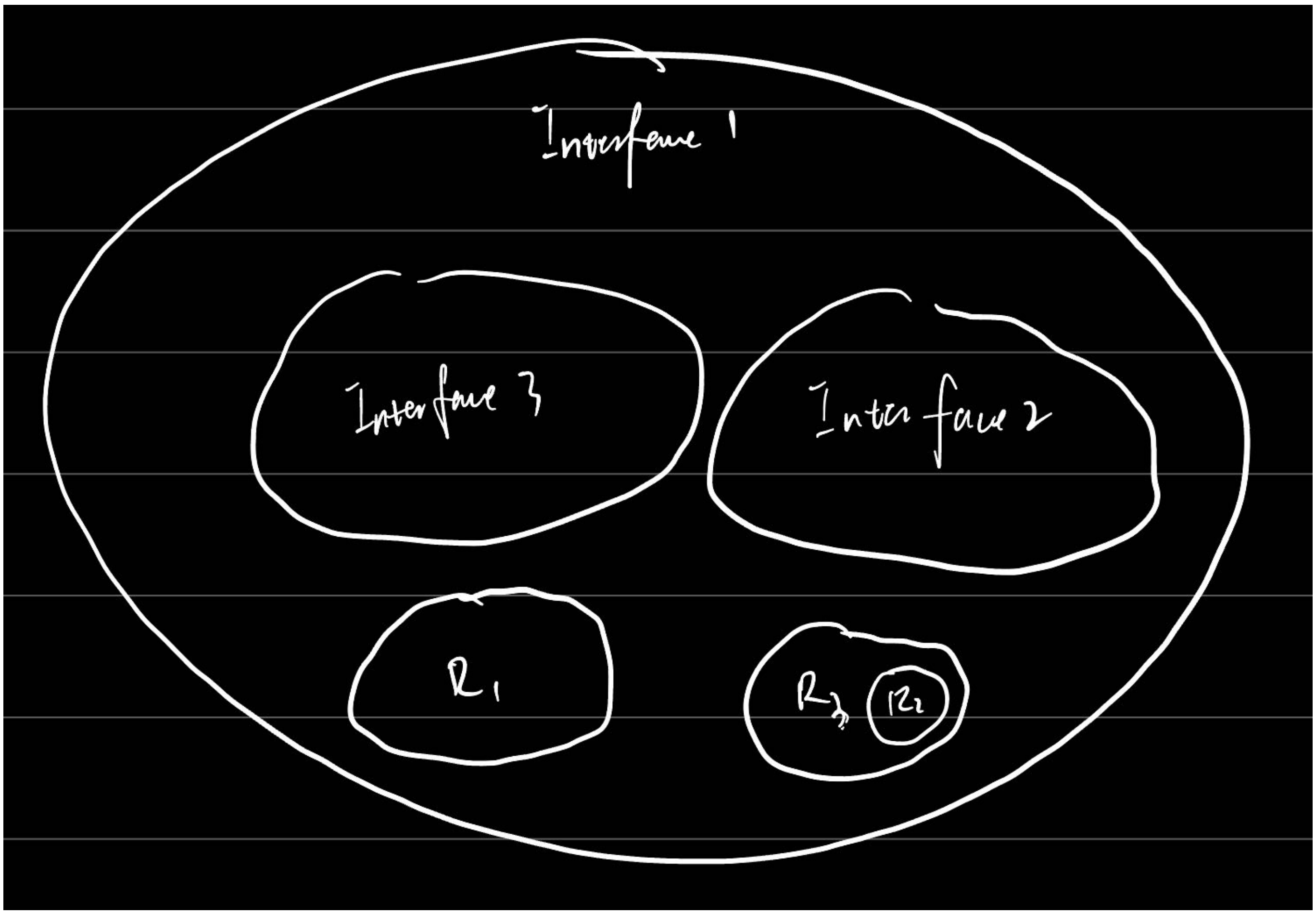
A: **200.20.15.0**0000000/25 = 200.20.15.0/25

B: **200.20.15.100**00000/27 = 200.20.15.128/27

C: **200.20.15.101**00000/27 = 200.20.15.160/27

Answer 2:

Here is the subnet relation:



Interface 1 128.174.240.0/20: **128.174.1111**0000.0/20, so $2^{12} - 2^{10} - 2^{10} - 2^{7} - 2^{3} = 1912$ address in total

R1 128.174.240.128/25: **128.174.240.1**0000000/25, so $2^{7} = 128$ addresses in total

R2 128.174.240.17: 1 address in total

Interface 3 128.174.252.0/22: **128.174.111111**00.0/22, so $2^{10} = 1024$ addresses in total

R3 128.174.240.16/29: **128.174.240.00010**000, so $2^3 - 1 = 7$ addresses in total

Interface 2 128.174.248.0/22: **128.174.111110**00.0/22, so $2^{10} = 1024$ addresses in total

Answer 3:

(a) R2

(b) Interface 1

(c) Interface 2

(d) Interface 3

(e) Interface 4

(f) R3

Question 4

1. (1)Process runing on the host computer join the network and find a server by sending DHCP discover packet.
   (2)The sever reponds with DHCP offer packet.
   (3)Then host sends DHCP request
   (4)Sever send DHCP ACK back.
   Host gets the IP address from sever.
2. Yes, it's possible. Their private address can be translated to the same IP using NAT.
3. IPv6 is 128 bits long and will provide larger IP address space. IPv6 can be global dedicated so it can get rid of NAT.
4. IPv6 can still be converted to IPv4 using tunneling and continue using NAT service. NAT provides security, devices inside local net not explicitly addressable, visible by outside world (a security plus).

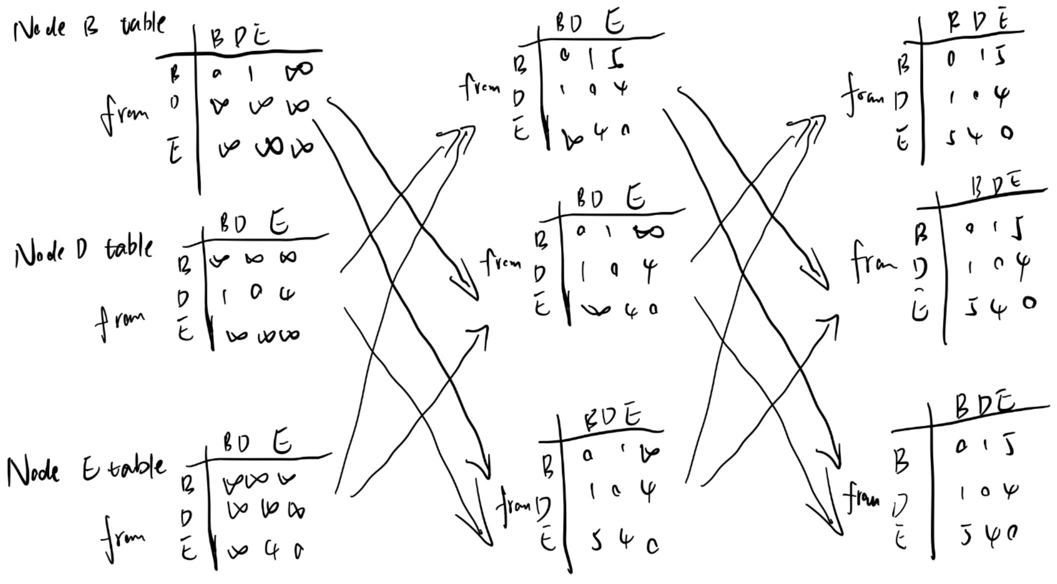
Question 5

Answer 1

Step	**N**	**D(A)**	**D(B)**	**D(C)**	**D(D)**	**D(F)**	**D(G)**
0	E	INF	INF	6, E	4, E	3, E	INF
1	EF	INF	INF	6, E	4, E		5, F
2	EFD	INF	5, D	6, E			5, F
3	EFDB	17, B		6, E			5, F
4	EFDBG	17, B		6, E			
6	EFDBGC	14, C					
7	EFDBGCA						

Answer 2

I only consider the route among B, D, E.



∴ Final Result

	B	D	E
B	0 (None)	1 (D)	5 (D)
D	1 (B)	0 (None)	4 (E)
E	5 (D)	4 (D)	0 (None)

Answer 3

In each iteration, each row in the table means the distance between the destination node and that node in each of the node routing table; the first table is the distance table and the second table is the next hop table. I don't print all the entries in the routing table for the reason of making the homework clean, other wise I have to print 7 * 7×7 tables in one iteration.

Before Change

_	A	B	C	D	E	F	G
A	0.0	12.0	8.0	11.0	14.0	14.0	15.0
B	12.0	0.0	4.0	1.0	5.0	4.0	5.0
C	8.0	4.0	0.0	3.0	6.0	6.0	7.0
D	11.0	1.0	3.0	0.0	4.0	3.0	4.0
E	14.0	5.0	6.0	4.0	0.0	3.0	5.0
F	14.0	4.0	6.0	3.0	3.0	0.0	2.0
G	15.0	5.0	7.0	4.0	5.0	2.0	0.0

_	A	B	C	D	E	F	G
A	None	C	C	C	C	C	C
B	A	None	D	D	D	D	D
C	A	D	None	D	E	D	D

_	A	B	C	D	E	F	G
D	C	B	C	None	E	F	G
E	C	D	C	D	None	F	F
F	D	D	D	D	E	None	G
G	D	D	D	D	F	F	None

After Change (C to D increase to 30)

iteration 1

_	A	B	C	D	E	F	G
A	0.0	12.0	8.0	11.0	14.0	14.0	15.0
B	12.0	0.0	4.0	1.0	5.0	4.0	5.0
C	8.0	9.0	0.0	10.0	6.0	9.0	11.0
D	13.0	1.0	5.0	0.0	4.0	3.0	4.0
E	14.0	5.0	6.0	4.0	0.0	3.0	5.0
F	14.0	4.0	6.0	3.0	3.0	0.0	2.0
G	15.0	5.0	7.0	4.0	5.0	2.0	0.0

_	A	B	C	D	E	F	G
A	NA	C	C	C	C	C	C
B	A	NA	D	D	D	D	D
C	A	B	NA	B	E	E	E
D	B	B	B	NA	E	F	G
E	C	D	C	D	NA	F	F
F	D	D	D	D	E	NA	G
G	D	D	D	D	F	F	NA

iteration 2

_	A	B	C	D	E	F	G
A	0.0	12.0	8.0	13.0	14.0	16.0	17.0
B	12.0	0.0	6.0	1.0	5.0	4.0	5.0
C	8.0	9.0	0.0	10.0	6.0	9.0	11.0
D	13.0	1.0	5.0	0.0	4.0	3.0	4.0
E	14.0	5.0	6.0	4.0	0.0	3.0	5.0

_	A	B	C	D	E	F	G
F	16.0	4.0	8.0	3.0	3.0	0.0	2.0
G	16.0	5.0	8.0	4.0	5.0	2.0	0.0

_	A	B	C	D	E	F	G
A	NA	B	C	B	C	B	B
B	A	NA	D	D	D	D	D
C	A	B	NA	B	E	E	E
D	B	B	B	NA	E	F	G
E	C	D	C	D	NA	F	F
F	D	D	D	D	E	NA	G
G	F	D	F	D	F	F	NA

iteration 3

_	A	B	C	D	E	F	G
A	0.0	12.0	8.0	13.0	14.0	16.0	17.0
B	12.0	0.0	6.0	1.0	5.0	4.0	5.0
C	8.0	9.0	0.0	10.0	6.0	9.0	11.0
D	13.0	1.0	7.0	0.0	4.0	3.0	4.0
E	14.0	5.0	6.0	4.0	0.0	3.0	5.0
F	16.0	4.0	8.0	3.0	3.0	0.0	2.0
G	17.0	5.0	9.0	4.0	5.0	2.0	0.0

_	A	B	C	D	E	F	G
A	NA	B	C	B	C	B	B
B	A	NA	D	D	D	D	D
C	A	B	NA	B	E	E	E
D	B	B	B	NA	E	F	G
E	C	D	C	D	NA	F	F
F	D	D	D	D	E	NA	G
G	D	D	D	D	F	F	NA

iteration 4

_	A	B	C	D	E	F	G

_	A	B	C	D	E	F	G
A	0.0	12.0	8.0	13.0	14.0	16.0	17.0
B	12.0	0.0	8.0	1.0	5.0	4.0	5.0
C	8.0	9.0	0.0	10.0	6.0	9.0	11.0
D	13.0	1.0	7.0	0.0	4.0	3.0	4.0
E	14.0	5.0	6.0	4.0	0.0	3.0	5.0
F	16.0	4.0	9.0	3.0	3.0	0.0	2.0
G	17.0	5.0	10.0	4.0	5.0	2.0	0.0

_	A	B	C	D	E	F	G
A	NA	B	C	B	C	B	B
B	A	NA	D	D	D	D	D
C	A	B	NA	B	E	E	E
D	B	B	B	NA	E	F	G
E	C	D	C	D	NA	F	F
F	D	D	E	D	E	NA	G
G	D	D	F	D	F	F	NA

iteration 5

_	A	B	C	D	E	F	G
A	0.0	12.0	8.0	13.0	14.0	16.0	17.0
B	12.0	0.0	8.0	1.0	5.0	4.0	5.0
C	8.0	9.0	0.0	10.0	6.0	9.0	11.0
D	13.0	1.0	9.0	0.0	4.0	3.0	4.0
E	14.0	5.0	6.0	4.0	0.0	3.0	5.0
F	16.0	4.0	9.0	3.0	3.0	0.0	2.0
G	17.0	5.0	11.0	4.0	5.0	2.0	0.0

_	A	B	C	D	E	F	G
A	NA	B	C	B	C	B	B
B	A	NA	D	D	D	D	D
C	A	B	NA	B	E	E	E
D	B	B	B	NA	E	F	G

_	A	B	C	D	E	F	G
E	C	D	C	D	NA	F	F
F	D	D	E	D	E	NA	G
G	D	D	D	D	F	F	NA

iteration 6

_	A	B	C	D	E	F	G
A	0.0	12.0	8.0	13.0	14.0	16.0	17.0
B	12.0	0.0	9.0	1.0	5.0	4.0	5.0
C	8.0	9.0	0.0	10.0	6.0	9.0	11.0
D	13.0	1.0	9.0	0.0	4.0	3.0	4.0
E	14.0	5.0	6.0	4.0	0.0	3.0	5.0
F	16.0	4.0	9.0	3.0	3.0	0.0	2.0
G	17.0	5.0	11.0	4.0	5.0	2.0	0.0

_	A	B	C	D	E	F	G
A	NA	B	C	B	C	B	B
B	A	NA	C	D	D	D	D
C	A	B	NA	B	E	E	E
D	B	B	B	NA	E	F	G
E	C	D	C	D	NA	F	F
F	D	D	E	D	E	NA	G
G	D	D	F	D	F	F	NA

iteration 7

_	A	B	C	D	E	F	G
A	0.0	12.0	8.0	13.0	14.0	16.0	17.0
B	12.0	0.0	9.0	1.0	5.0	4.0	5.0
C	8.0	9.0	0.0	10.0	6.0	9.0	11.0
D	13.0	1.0	10.0	0.0	4.0	3.0	4.0
E	14.0	5.0	6.0	4.0	0.0	3.0	5.0
F	16.0	4.0	9.0	3.0	3.0	0.0	2.0
G	17.0	5.0	11.0	4.0	5.0	2.0	0.0

_	A	B	C	D	E	F	G
A	NA	B	C	B	C	B	B
B	A	NA	C	D	D	D	D
C	A	B	NA	B	E	E	E
D	B	B	B	NA	E	F	G
E	C	D	C	D	NA	F	F
F	D	D	E	D	E	NA	G
G	D	D	F	D	F	F	NA

Question 6

1. (a)
   d->a->c->e
   d->c->e
   d->b->e
   d->e
   (b)
   d->a
   d->c->a
   d->e->c->a
   d->b->e->c->a

2.

Link State: Each node stores a detailed map to entire network topology

Distance Vector: Each node stores distance tp destinationa and neighbors distance to destination.

Path Vector: The alorigthm advertise paths to different destination network prefixes. So each node contains path vector to destination. The entire path information is in the node.

stored states: Distance Vector < Path Vector < Link State