۱.

۲.

الف) آدرس به صورت مقابل است:

10001111 . 00111000 . 00001111 . 00001010

Network Mask نیز به صورت زیر است:

11111111 . 11111111 . 11111000 . 00000000

پس از And کردن آدرس به صورت زیر خواهد بود:

1000111 . 00111000 . 00001000 . 00000000 → 143.56.8.0

ب)

143.56.8.1 – 143.56.8.254

اولی برای آدرس خود شبکه و آخری برای Broadcast است

۳.

چون ۱۴ تا کامپیوتر داریم پس به ۴ بیت در hostID نیاز داریم و به ۴ بیت در subnetID برای اینکه بتواند بیش از ۸ شبکه را آدرس دهی کند پس:

۸ بیت ۸ بیت ۴ بیت
---------- ---------- ----------
۴ بیت

۴.

بیت های متغییر

10010110 . 00001010. 00100011 . 00000000
10010110 . 00001010. 00100100 . 00000000
10010110 . 00001010. 00100101 . 00000000
10010110 . 00001010. 00100110 . 00000000

با توجه به تغییر بیت ها در ۱۱ بیت آخر پروتکل CIDR به شکل مقابل خواهد شد:

10010110 . 00001010. 00100---. -------- → 150.10.32.0/21

۵.

سرآیند IP به اندازه ۲۰ بایت را از میزان هر بسته کم می کنیم تا مشخص شود چقدر داده وجود دارد 380=20-400

این میزان داده ها با توجه به offset مربوط به IP که ۸ بایتی است چند offset می شود: $[380] = 47$

باید این تعدا چند بایتی باشند: 376 = 47*8

بنابراین داریم : 376+376+376+72

به هر کدام از این بسته ها نیز ۲۰ بایت هدر اضافه می شود: 396+396+396+92

برای محاسبه ی fragment offset داریم:

اولین ۳۷۶ بایت که fragment offset ندارد. دومی $\frac{376}{8} = 47$ و برای سومی $\frac{376+376}{8} = 94$ و برای چهارمی $\frac{376+376+376}{8} = 141$.

	Total Length	ID	DF	MF	Fragment Office
Original Packet	1220	X	0	0	0
Fragment1	396	X	0	1	0
Fragment2	396	X	0	1	47
Fragment3	396	X	0	1	94
Fragment4	92	X	0	0	141

۶.

۷.

دقت کنید که باید الگوریتم های bellman ford و dijkstra را بلد باشید. خود الگوریتم را از کتاب بخوانید و بلد باشید، نه اینکه سوال امتحان باشه ها !

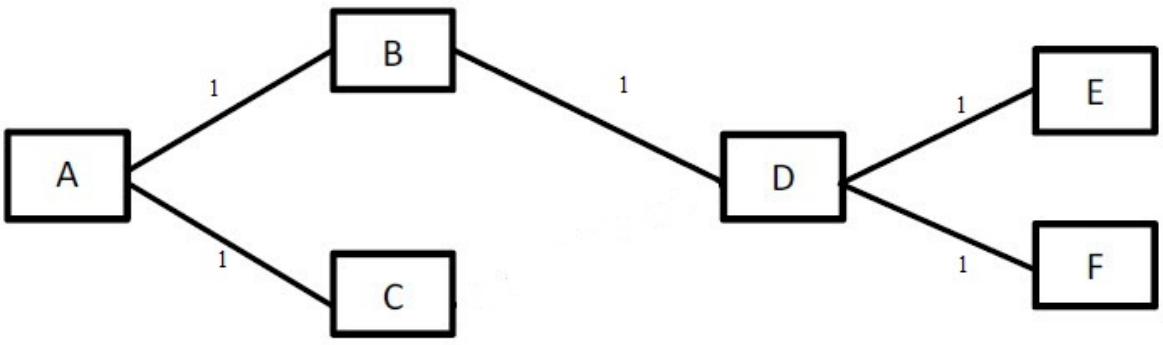
بعد از طی روند این الگوریتم جدول مطابق شکل زیر خواهد شد: (توجه داشته باشید که در امتحان طی مراحل حل بسیار مهم است و باید همه ی فرمول ها را بدست آورده و مشخص کنید از چه مسیری به جدول زیر رسیدید مثلا باید را حساب کنید برای همه ی نودها) $D(y) = \min\{$اندازه قبلی , از طریق مسیر جدید $\}$

Iteration	N'	D(t),P(t)	D(u),P(u)	D(v),P(v)	D(w),P(w)	D(x),P(x)	D(y),P(y)	D(z),P(z)
0	{z}	∞	∞	∞	∞	7,z	13,z	0,z
1	{z,x}	∞	∞	10,x	13,x	7,z	13,z	0,z
2	{z,x,v}	14,v	12,v	10,x	13,x	7,z	13,z	0,z
3	{z,x,v,u}	14,v	12,v	10,x	13,x	7,z	13,z	0,z
4	{z,x,v,u,y}	14,v	12,v	10,x	13,x	7,z	13,z	0,z
5	{z,x,v,u,y,w}	14,v	12,v	10,x	13,x	7,z	13,z	0,z
6	{z,x,v,u,y,w}	14,v	12,v	10,x	13,x	7,z	13,z	0,z

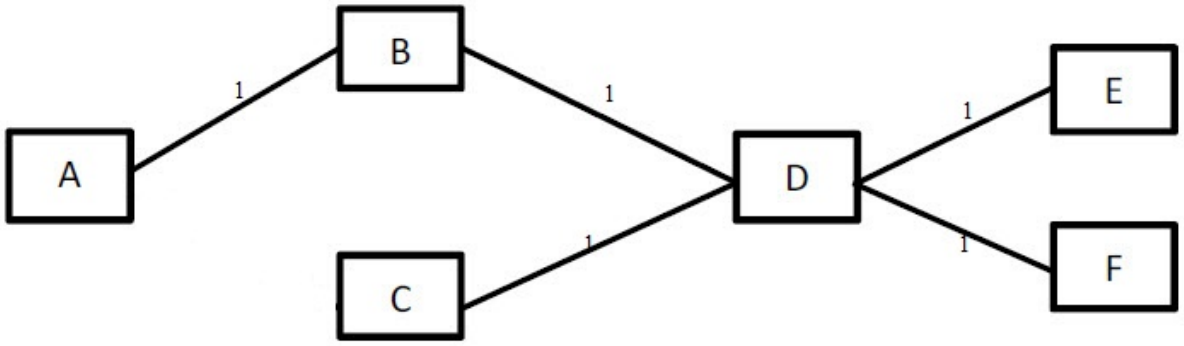
۸.

a. برای همه ی گره ها الگوریتم dijkstra را با اندازه ی هر یال برابر ۱ بدست می آوریم

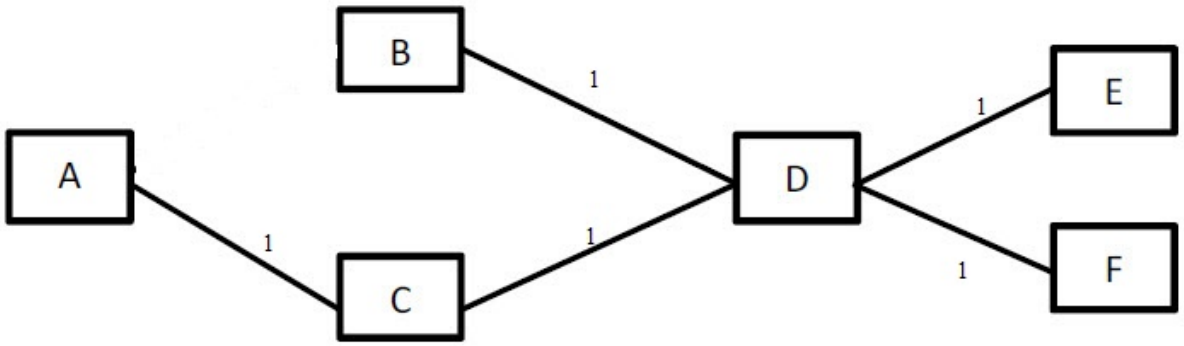
For Node A	حرکت بعدی	Cost
B	B	**1**
C	C	**1**
D	B	**2**
E	B	**3**
F	B	**3**



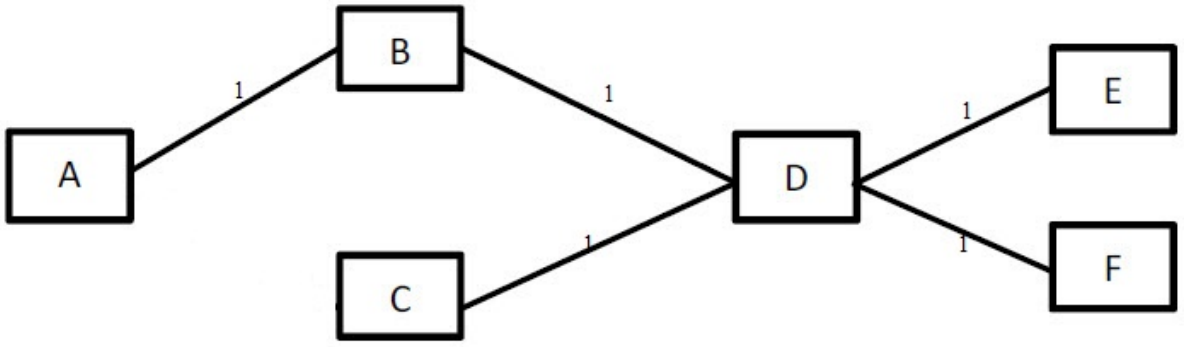
For Node B	حرکت بعدی	Cost
A	A	**1**
C	D	**2**
D	D	**1**
E	D	**2**
F	D	**2**



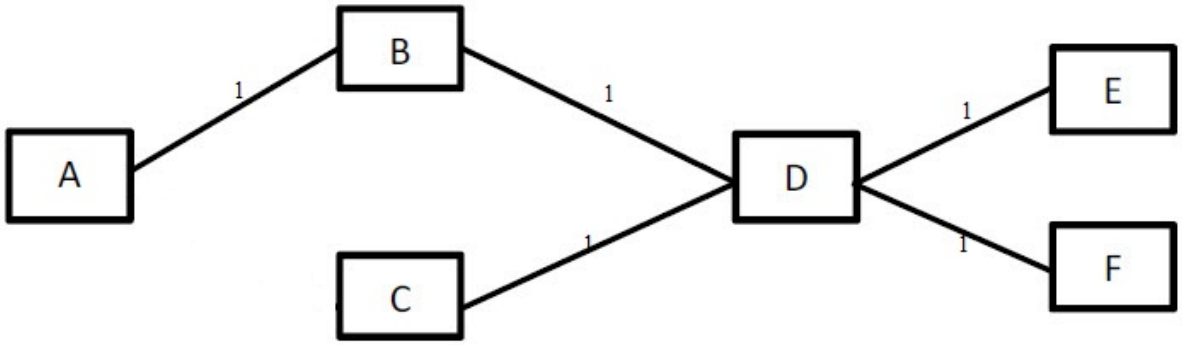
For Node C	حرکت بعدی	Cost
A	A	1
B	D	2
D	D	1
E	D	2
F	D	2



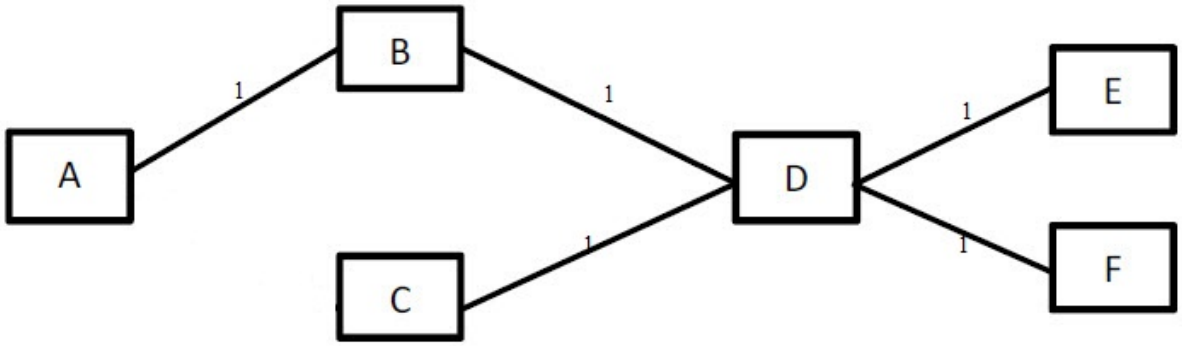
For Node D	حرکت بعدی	Cost
A	B	2
B	B	1
C	C	1
E	E	1
F	F	1



For Node E	حرکت بعدی	Cost
A	D	3
B	D	2
C	D	2
D	D	1
F	D	2

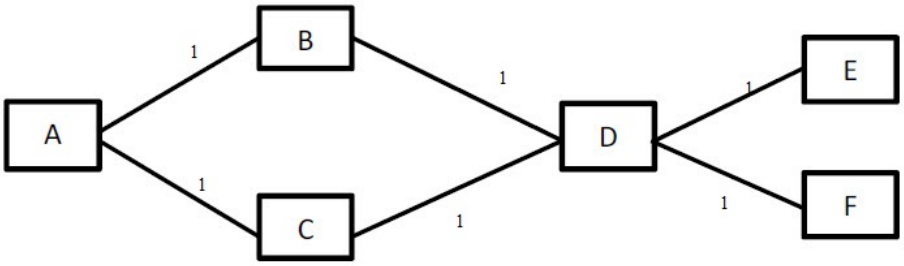


For Node F	حرکت بعدی	Cost
A	D	3
B	D	2
C	D	2
D	D	1
E	D	2



b. همانطور که می بینیم این مسیریابی load balancing را ندارد، زیرا مثلا در شکل اول میبینیم که برای رسیدن به نود D,E,F همواره از طریق B حرکت می کند در صورتی که می توانست مثلا برای رفتن به E از B و برای رفتن به F از C استفاده کند. در واقع بار شبکه و load شبکه را balance نمی کند و در همه جا پخش نمی کند و اگر ازدحام در B اتفاق بیفتند دیگر نمی توان بسته ها را به D رساند درصورتی که ممکن است نود C خالی باشد. باید از الگوریتمی مانند الگوریتم OSPF استفاده کرد در واقع نود هایی که اندازه هایی برابر دارند مانند B,C را هردو را در شبکه نگه داشت و هرکدام که کمتر برای رسیدن به نودهایی دیگر استفاده شده را به کار برد. مثلا برای رسیدن به E از B استفاده کرد و برای رسیدن به F از C استفاده کرد که به شکل زیر در مسیریابی خواهیم رسید:

.c



.۹

الف) چون آدرس با ۱۶۵ شروع شده است پس متعلق به اولی در جدول نیست. ما آدرس IP را با آدرس Destination ها Mask می کنیم مثلا در اولی چون Mask آنها ---.---.----.255 (یعنی ابتدای آن ها ۲۵۵ قرار داده شده است) پس باید با ابتدای destination مقایسه شود و چون ۱۹۲ مخالف ۱۶۵ است پس به اولی نمی رود. و همچنین چون مخالف ۱۲۷ است به سومی نمی رود. و 192.168.1.10 را با دومی یعنی 192.168.1.0 در mask آن یعنی 255.255.255.0 مقایسه می کنیم (یعنی سه تای اولی را فقط مقایسه می کنیم) و چون برابرند پس آدرس مورد نظر ما به Interface دومی eth1 خواهد رفت.

ب) برای اولی داریم:

165.230.198.64 – 165.230.198.127

که از این ۶۳ تا، دو تا کم می شود. (۶۱)

برای دو می داریم:

192.168.1.0 – 192.168.1.255

که از این 256 تا، دو تا کم می شود. (۲۵۴)

.۱۰

1. Initialization

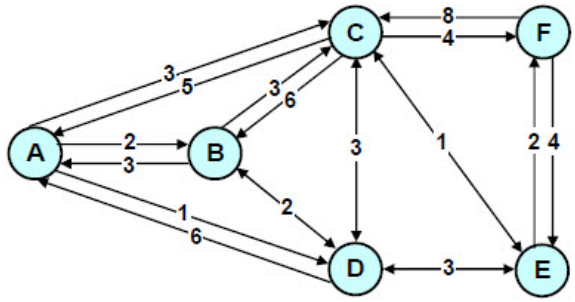
$$D_i = \infty, \forall i \neq d$$
$$D_d = 0$$

2. Updating: For each $i \neq d$,

$$D_i = \min_j \{C_{ij} + D_j\}, \forall j \neq i$$

Repeat step 2 until no more changes occur in the iteration.

ماتریس همجواری به شکل مقابل است:



$$\Rightarrow c_{ij} = \begin{array}{c} \\ A \\ B \\ C \\ D \\ E \\ F \end{array} \begin{array}{c} \begin{array}{cccccc} A & B & C & D & E & F \end{array} \\ \begin{bmatrix} 0 & 2 & 3 & 1 & \infty & \infty \\ 3 & 0 & 3 & 2 & \infty & \infty \\ 5 & 6 & 0 & 3 & 1 & 4 \\ 6 & 2 & 3 & 0 & 3 & \infty \\ \infty & \infty & 1 & 3 & 0 & 2 \\ \infty & \infty & 8 & \infty & 4 & 0 \end{bmatrix} \end{array}$$

سپس الگوریتم را اعمال می کنیم و جواب به صورت زیر خواهد شد: (توجه خود الگوریتم را حفظ کنید و محاسبات را هم کامل بنویسید)

.a

Iteration	A	C	D	E	F
Initial	**(-1, ∞)**	**(-1, ∞)**	**(-1, ∞)**	**(-1, ∞)**	**(-1, ∞)**
	(B, 2) (C, ∞) (D, ∞)	(A, ∞) (B, 6) (D, ∞) (E, ∞) (F, ∞)	(A, ∞) (B, 2) (C, ∞) (E, ∞)	(C, ∞) (D, ∞) (F, ∞)	(C, ∞) (E, ∞)
1	**(B, 2)**	**(B, 6)**	**(B, 2)**	**(-1, ∞)**	**(-1, ∞)**
	(B, 2) (C, 9) (D, 3)	(A, 7) (B, 6) (D, 5) (E, ∞) (F, ∞)	(A, 8) (B, 2) (C, 9) (E, ∞)	(C, 7) (D, 5) (F, ∞)	(C, 14) (E, ∞)
2	**(B, 2)**	**(D, 5)**	**(B, 2)**	**(D, 5)**	**(C, 14)**
	(B, 2) (C, 8) (D, 3)	(A, 7) (B, 6) (D, 5) (E, 6) (F, 18)	(A, 8) (B, 2) (C, 8) (E, 8)	(C, 6) (D, 5) (F, 16)	(C, 13) (E, 9)
3	**(B, 2)**	**(D, 5)**	**(B, 2)**	**(D, 5)**	**(E, 9)**
	(B, 2) (C, 8) (D, 3)	(A, 7) (B, 6) (D, 5) (E, 6) (F, 13)	(A, 8) (B, 2) (C, 8) (E, 8)	(C, 6) (D, 5) (F, 11)	(C, 13) (E, 9)
4	**(B, 2)**	**(D, 5)**	**(B, 2)**	**(D, 5)**	**(E, 9)**

.b

Iteration	A	C	D	E	F
Before Break	**(B, 2)**	**(D, 5)**	**(B, 2)**	**(D, 5)**	**(E, 9)**
	(B, 2) (C, 8) (D, 3)	(A, 7) (B, 6) (D, 5) (E, 6) (F, 13)	(A, 8) (C, 8) (E, 8)	(C, 6) (D, 5) (F, 11)	(C, 13) (E, 9)
1	**(B, 2)**	**(D, 5)**	**(A, 8)**	**(D, 5)**	**(E, 9)**
	(B, 2) (C, 8) (D, 9)	(A, 7) (B, 6) (D, 11) (E, 6) (F, 13)	(A, 8) (C, 8) (E, 8)	(C, 6) (D, 11) (F, 11)	(C, 13) (E, 9)
2	**(B, 2)**	**(B, 6)**	**(A, 8)**	**(C, 6)**	**(E, 9)**
	(B, 2) (C, 9) (D, 9)	(A, 7) (B, 6) (D, 11) (E, 7) (F, 13)	(A, 8) (C, 9) (E, 9)	(C, 7) (D, 11) (F, 11)	(C, 14) (E, 10)
3	**(B, 2)**	**(B, 6)**	**(A, 8)**	**(C, 7)**	**(E, 10)**
	(B, 2) (C, 9) (D, 9)	(A, 7) (B, 6) (D, 11) (E, 8) (F, 14)	(A, 8) (C, 9) (E, 10)	(C, 7) (D, 11) (F, 12)	(C, 17) (E, 11)
4	**(B, 2)**	**(B, 6)**	**(A, 8)**	**(C, 7)**	**(E, 11)**
	(B, 2) (C, 9) (D, 9)	(A, 7) (B, 6) (D, 11) (E, 8) (F, 15)	(A, 8) (C, 9) (E, 10)	(C, 7) (D, 11) (F, 13)	(C, 17) (E, 11)
5	**(B, 2)**	**(B, 6)**	**(A, 8)**	**(C, 7)**	**(E, 11)**

۱۱.

۱۲.

برای distance-vector برای m*N خواهد شد و برای link-state برابر $N+m*N$ خواهد شد.