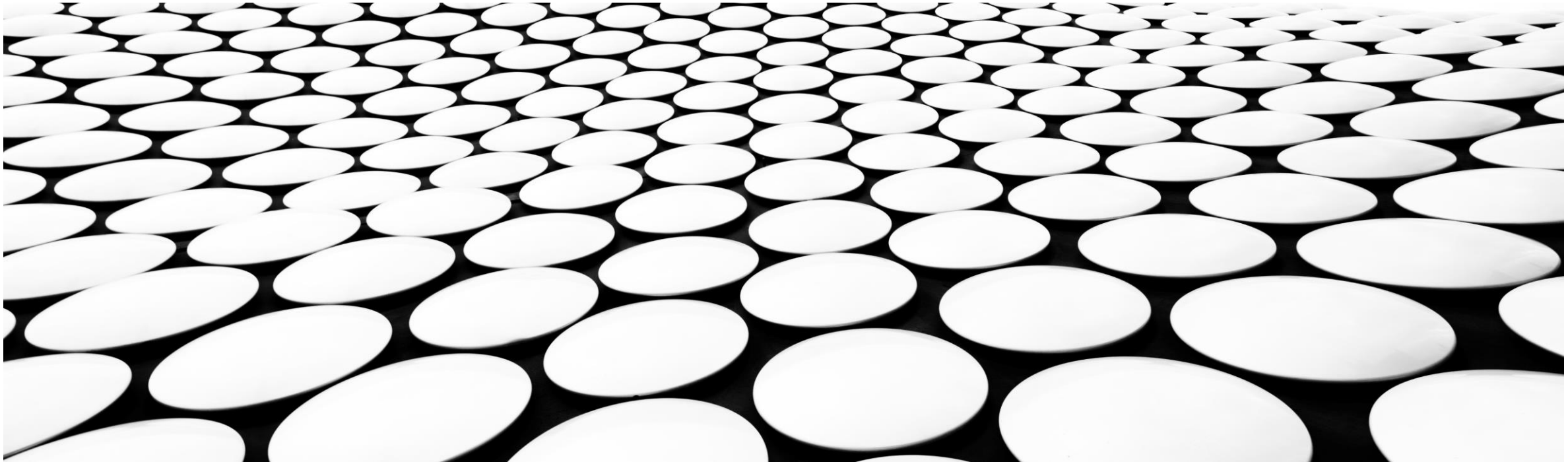
DYNAMIC ROUTING (CH. 17 AND 18)

HEMANT GHAYVAT

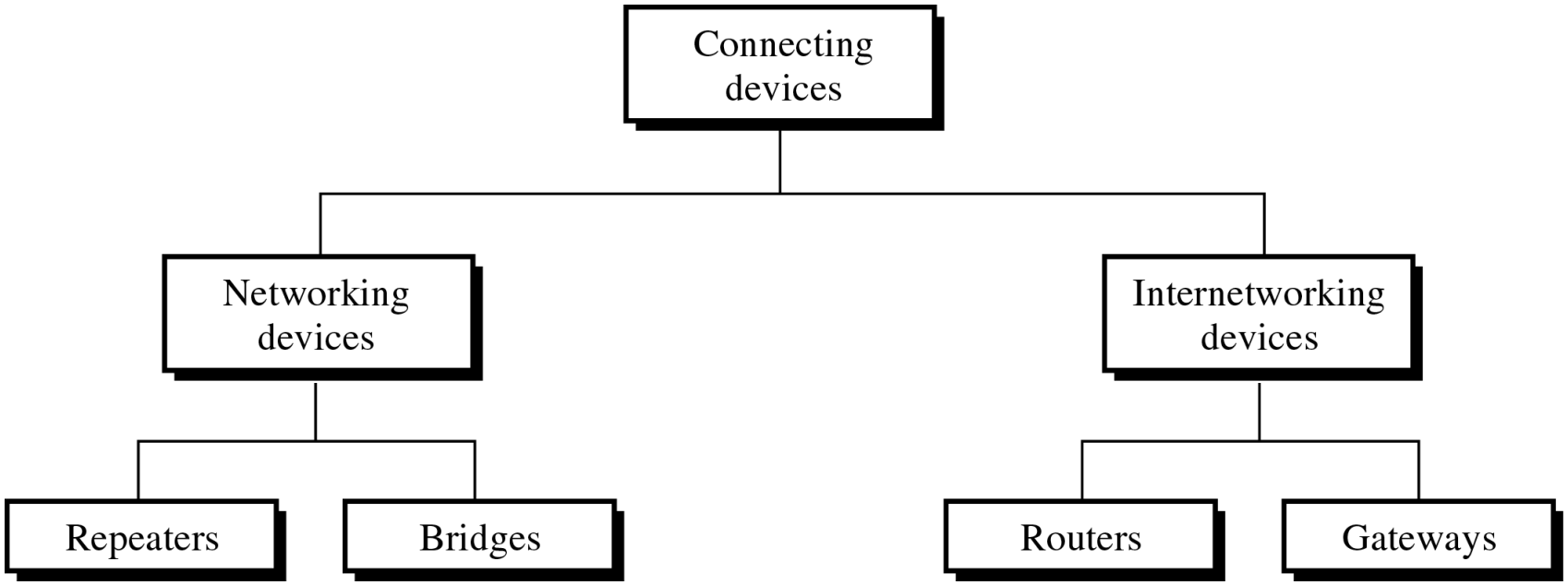



BRACE YOURSELVES
ANOTHER BORING PRESENTATION IS COMING
memegenerator.net

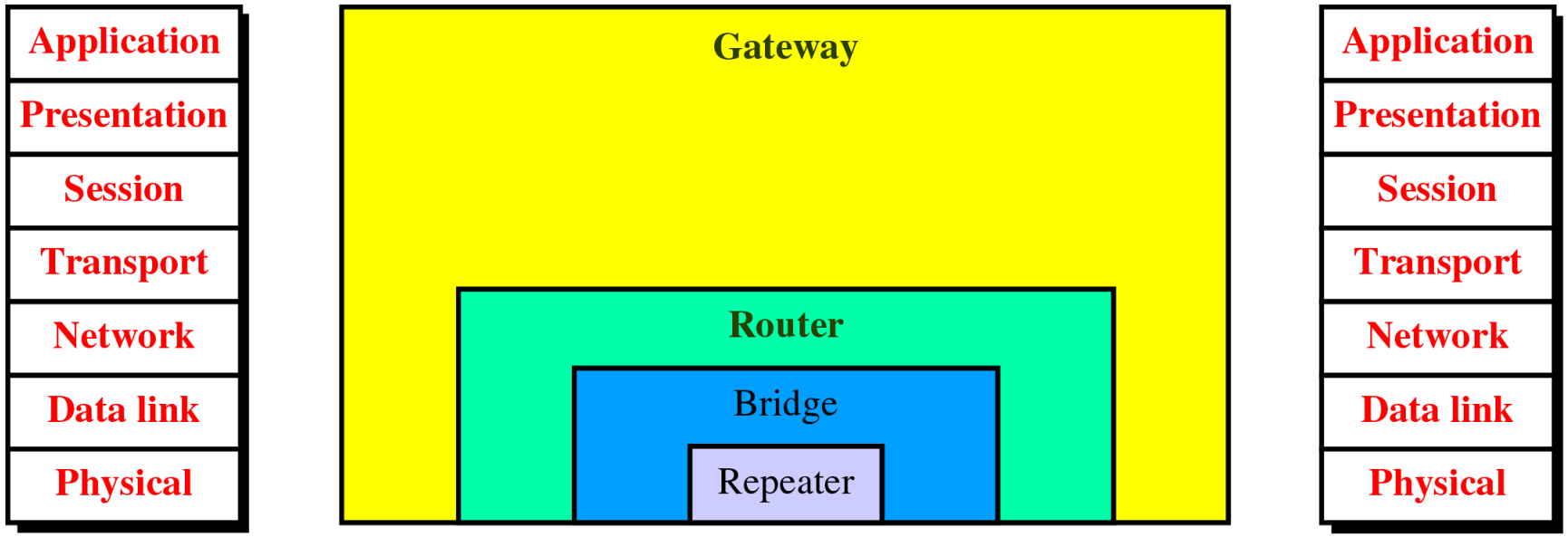
TODAY

- » Repeaters
- » Hubs/switches
- » Bridges
- » Routing
  - » Link state
  - » Distance vector

Connecting Devices



Connecting Devices and the OSI Model



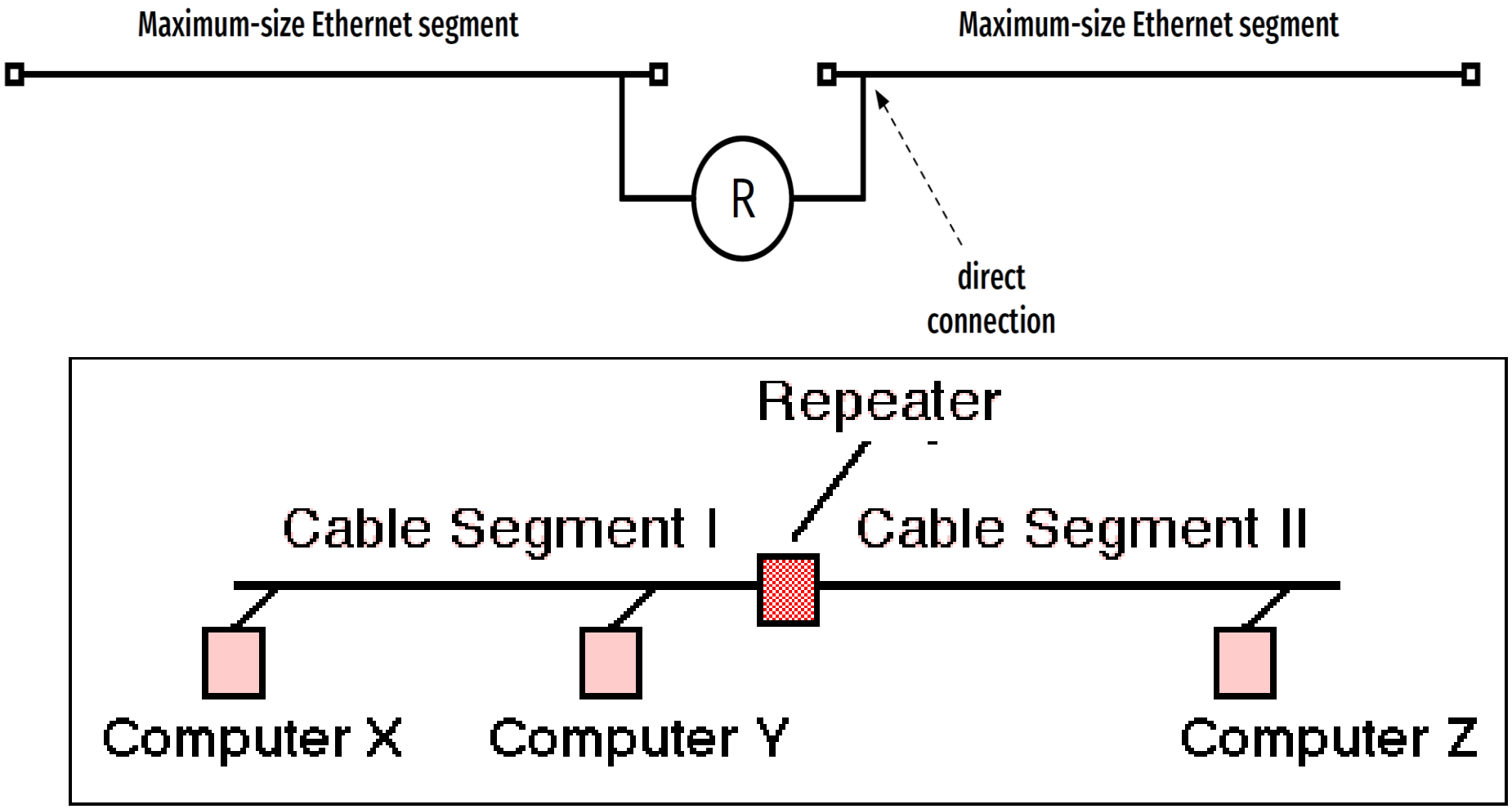
Remember!

Each LAN technology has a distance limitation, for example CSMA/CD cannot work across arbitrary distance.

However:

» Users desire arbitrary distance connections.

» Example: two computers across a corporate campus are part of one workgroup.

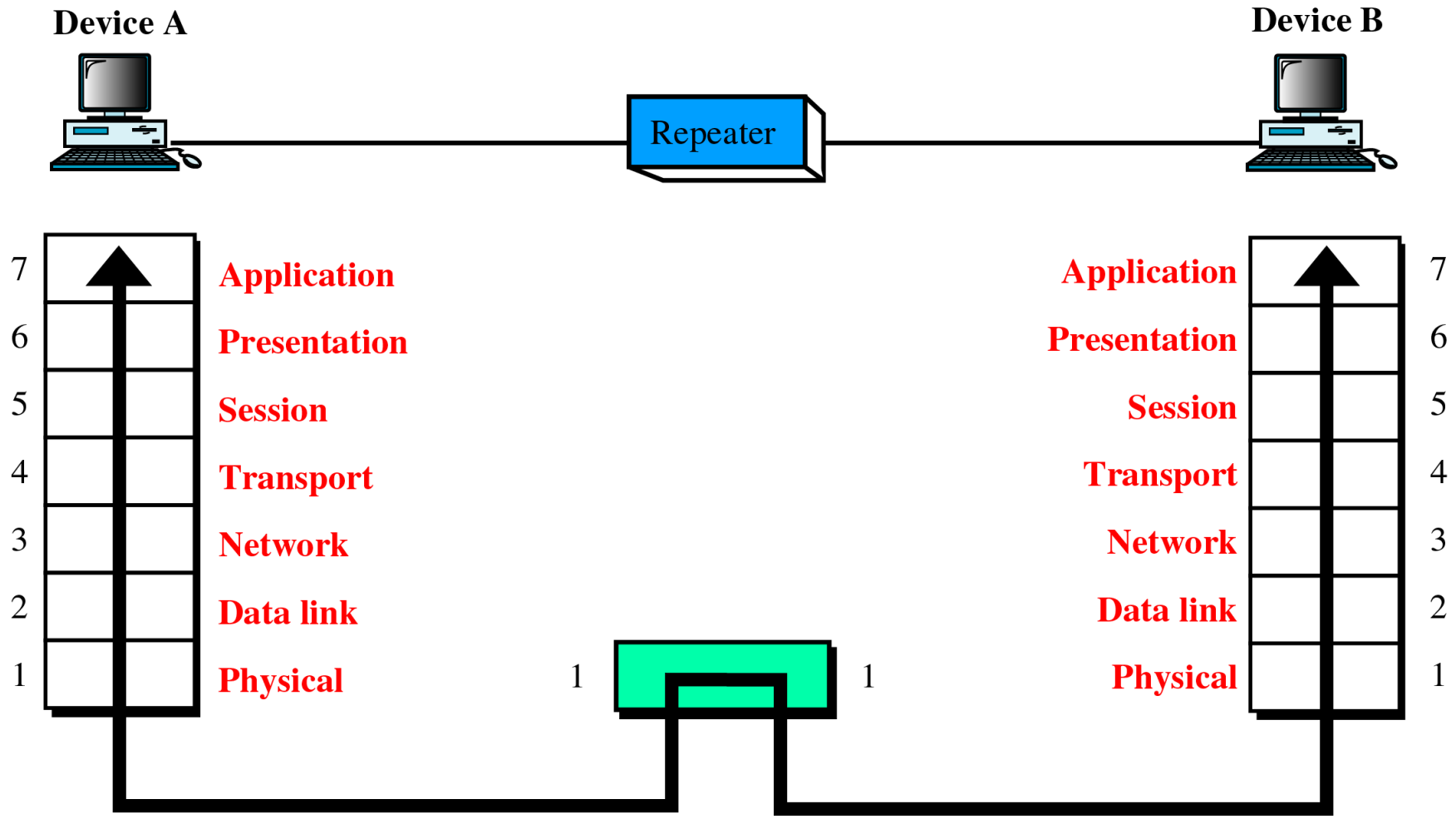


Hardware device that connects two LAN segments. Copies the signal from one segment to the other.

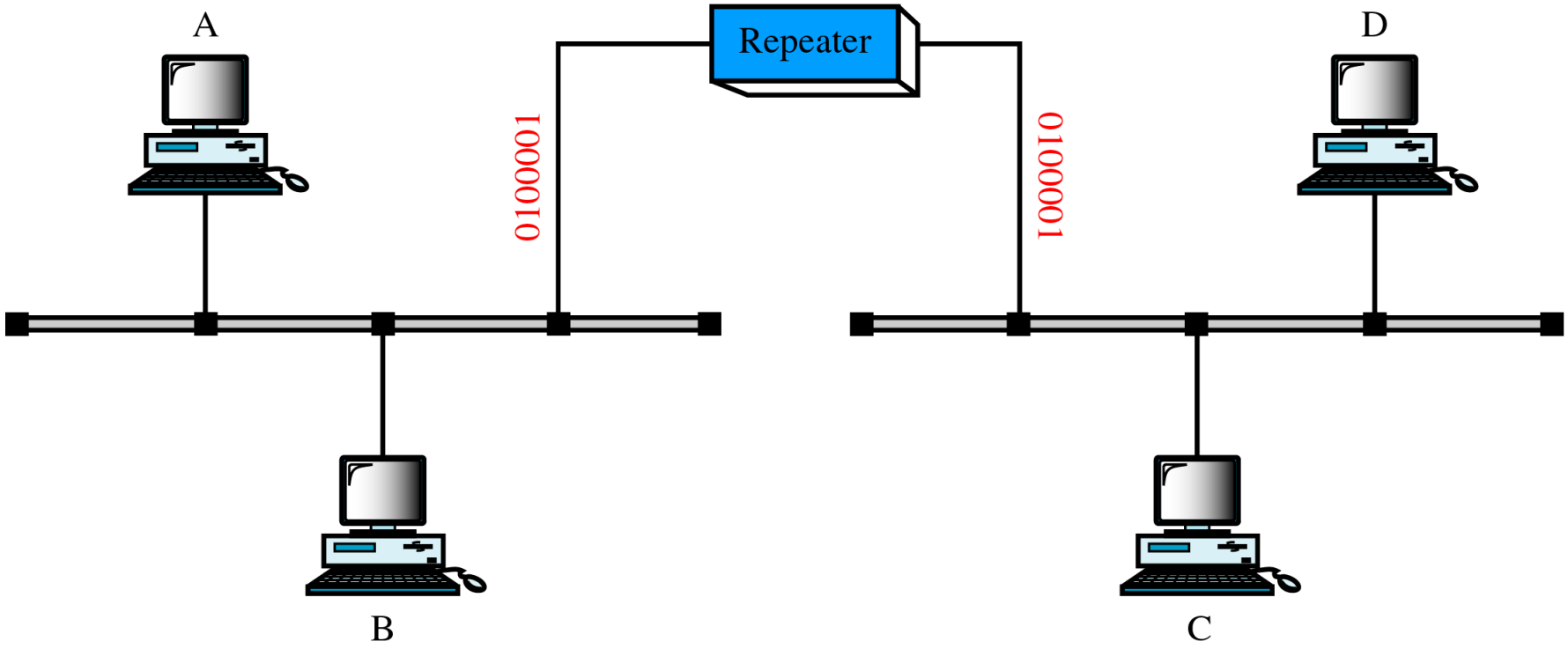
Repeater

» Amplifies signals from one segment and sends to the other.
» Operates in two directions simultaneously.
» Propagates noise and collisions.
» Layer-1 device.

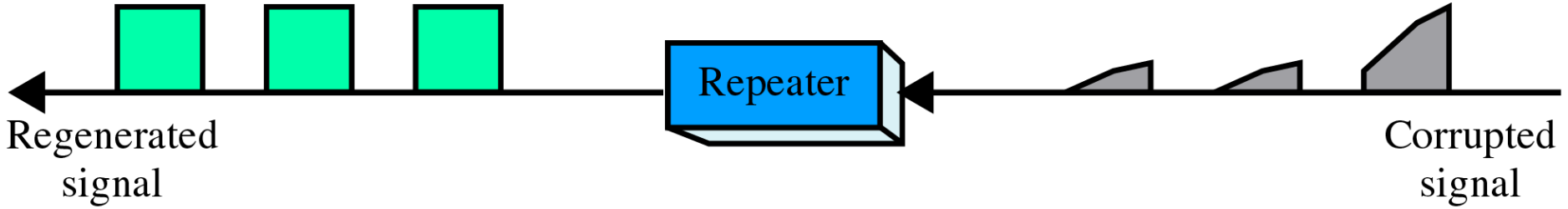
A Repeater in the OSI Model



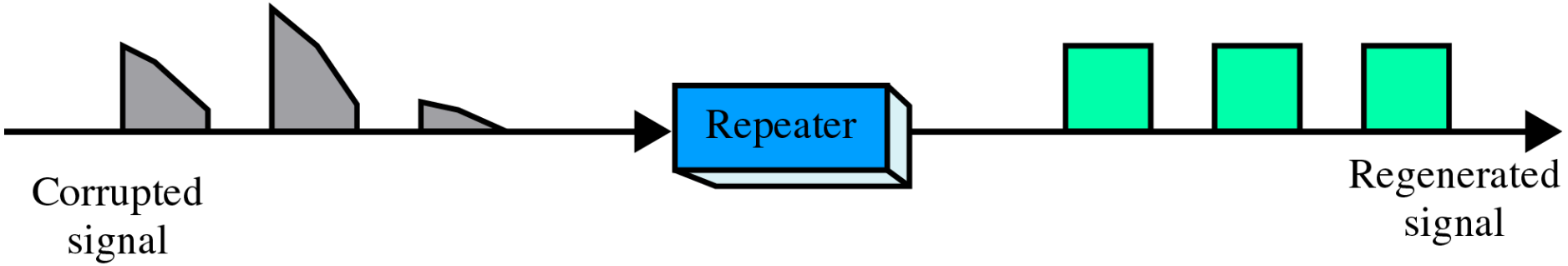
A Repeater



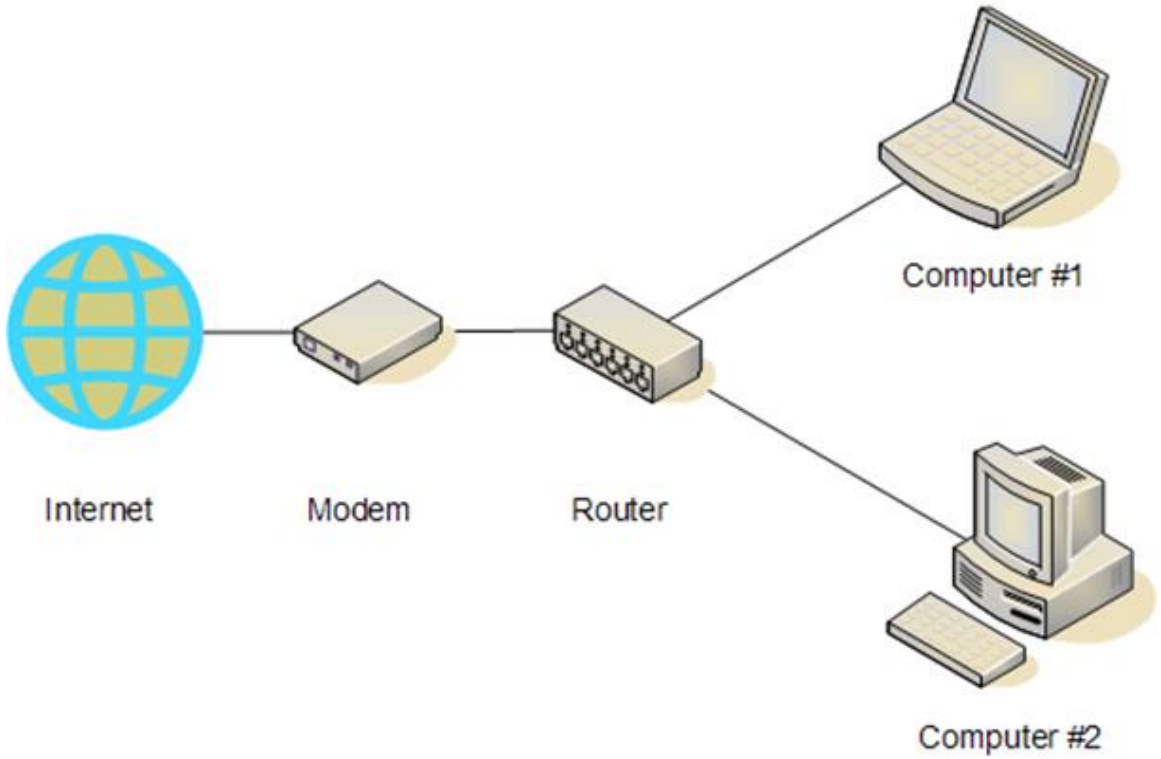
Function of a Repeater



(a) Right-to-left transmission.



(b) Left-to-right transmission.

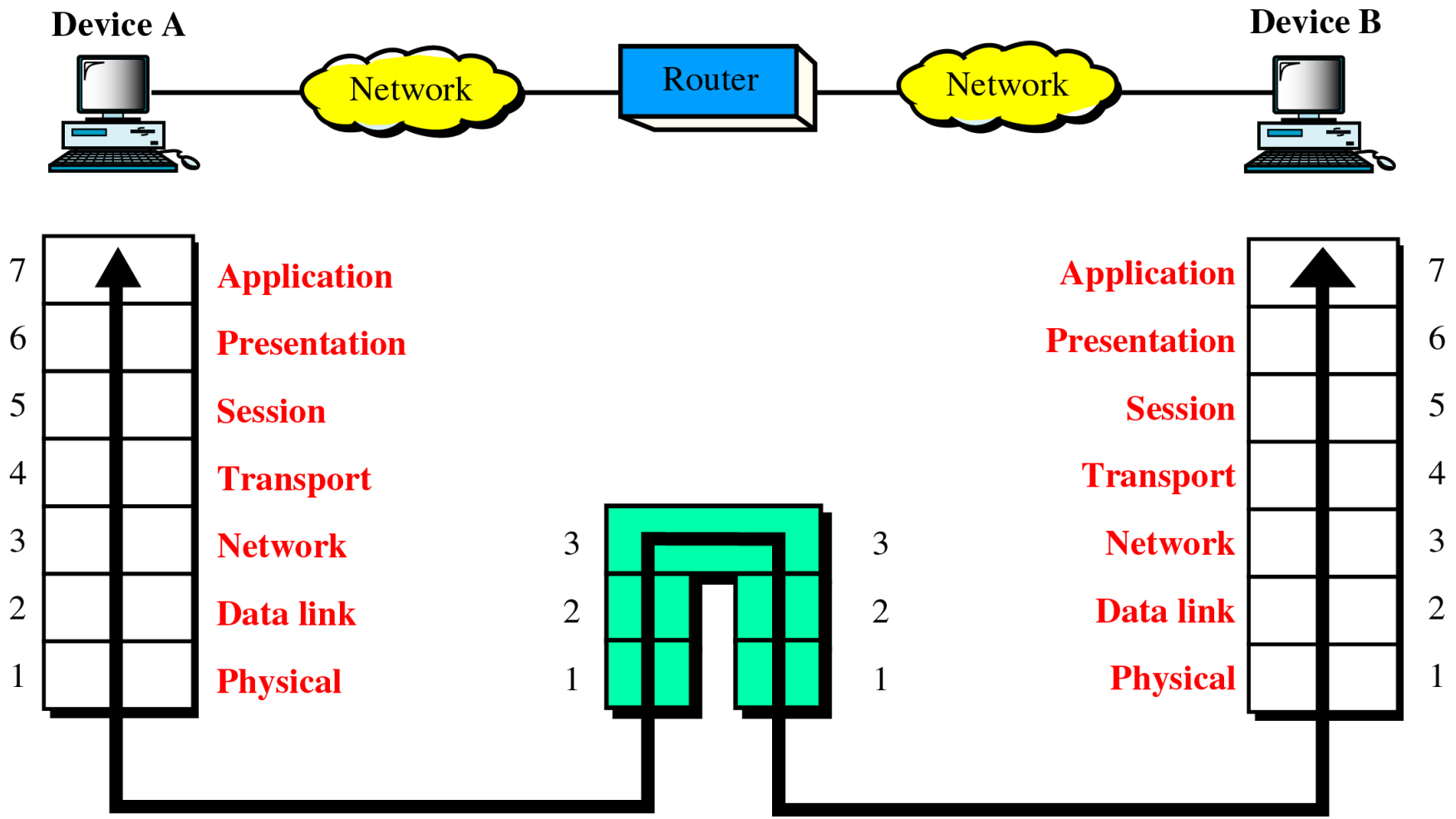


ROUTER

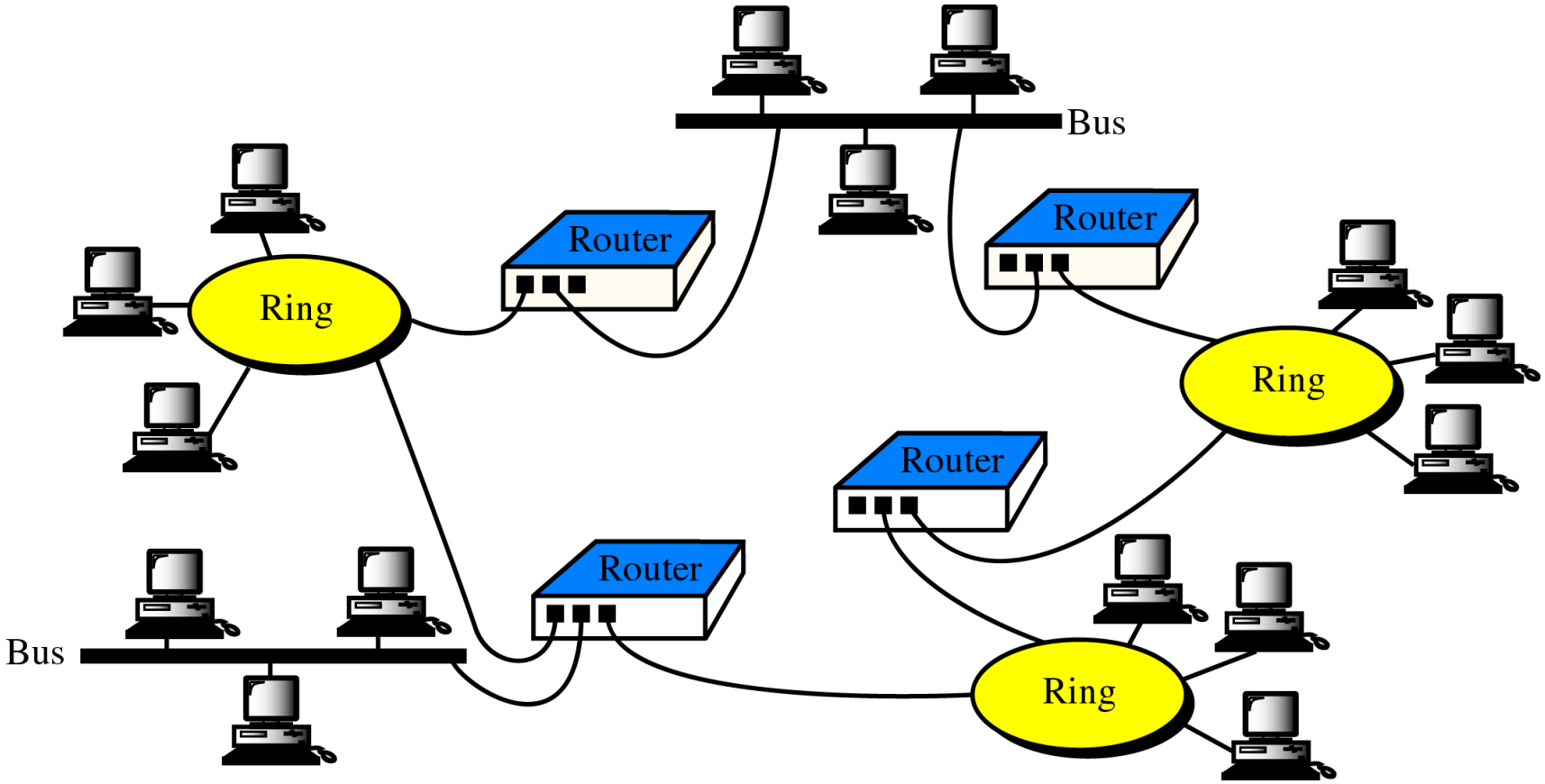
- Routers pass the information provided by the modem and routes it to the devices in the network such as the home computers. The information transferred by a router can be directed to a specific device by its unique number or rather its IP address. As noted before each device in that network is labelled by an IP address that allows other devices to communicate with it.
- There are two ways to connect to a router:

- Wired: the device is connected by a wire directly to the router (or an attached switch)
- Wirelessly: the device is not connected with a wire but rather through WiFi.

A Router in the OSI Model



Routers in an Internet



NETWORK HUB

- A hub is a device that allows several network devices to connect together to exchange data on a single network however, they have no management component. Network hubs are also known as repeaters. They are less ‘intelligent’ than switches. Unlike switches, which forward data to the intended devices, hubs merely send the data packets to all its ports. So as the name repeaters suggests, it only repeats the data from an incoming port to all the other devices; this leads to frequent collisions between packets.

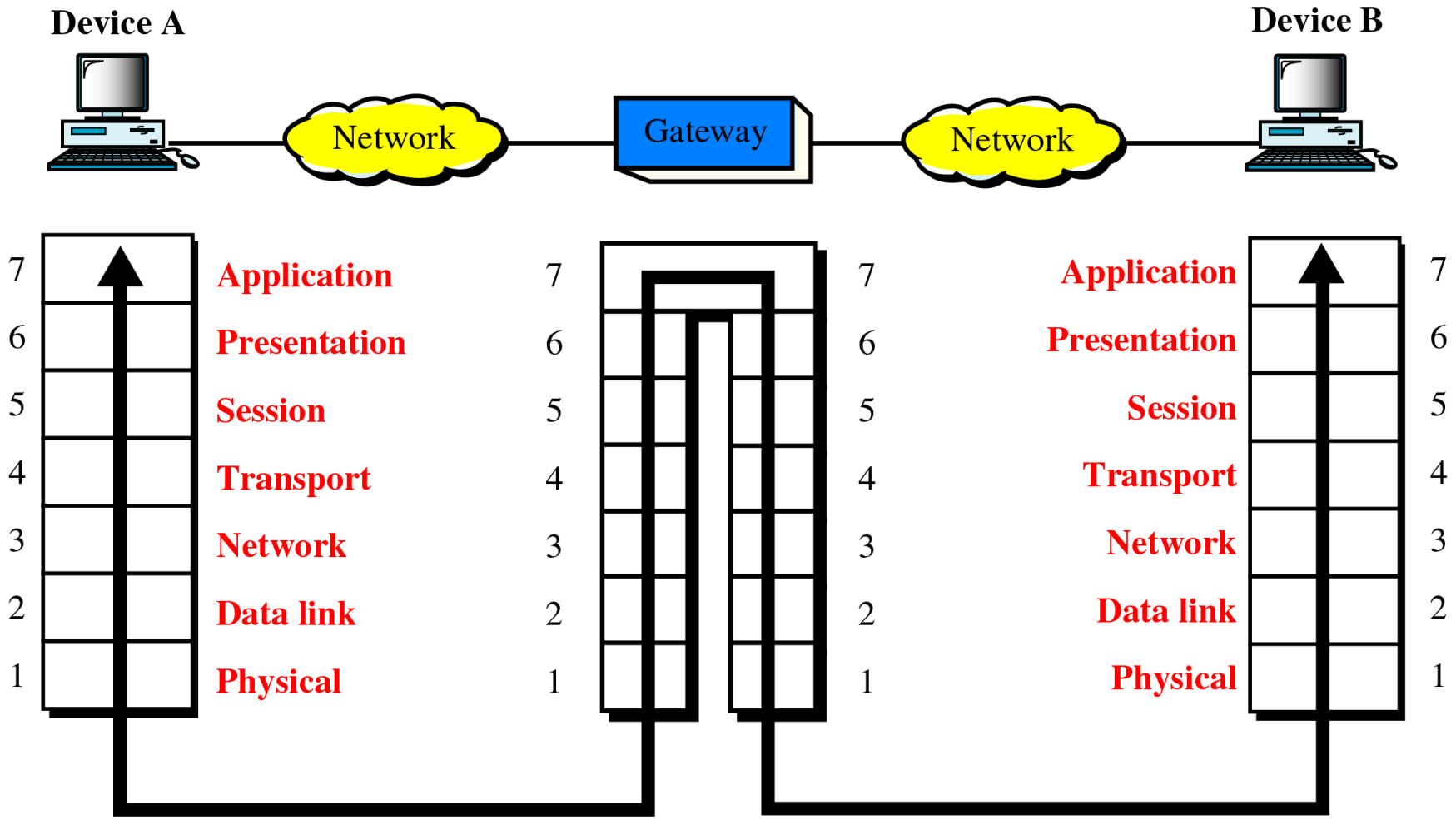
HUB

- » Small electronic device, has connections from several computers (e.g., 4 or 20).
- » Operates on signals, propagates each incoming signal to all connections. Similar to connecting segments with repeaters.
- » Does not understand packets.
- » Extremely low cost.

Gateway

- A node on a network that serves as an entrance to another network.
- In enterprises, the gateway is the computer that routes the traffic from a workstation to the outside network that is serving the Web pages. In homes, the gateway is the ISP that connects the user to the internet.
- Gateways, also called protocol converters, can operate at any network layer. The activities of a gateway are more complex than that of the router or switch as it communicates using more than one protocol.

A Gateway in the OSI Model



Switch

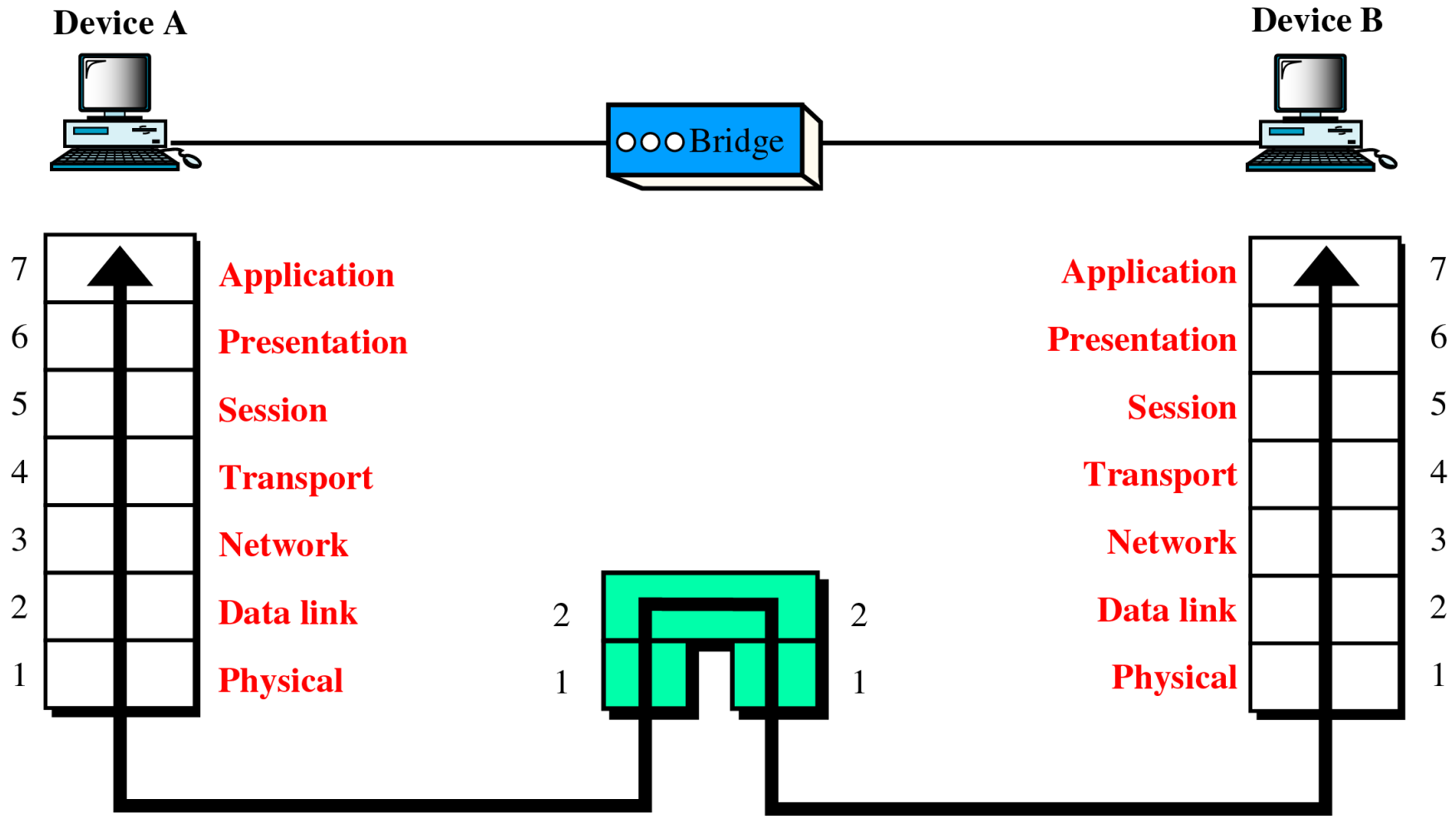
- A switch connects two or more nodes in the same or different network. Unlike the router which labels through IP address, switches use MAC addresses to direct the data to its correct destination.
- A switch can be used connect multiple Network devices (such as a computer, laptop, printer etc.) to the Home LAN.

A typical router

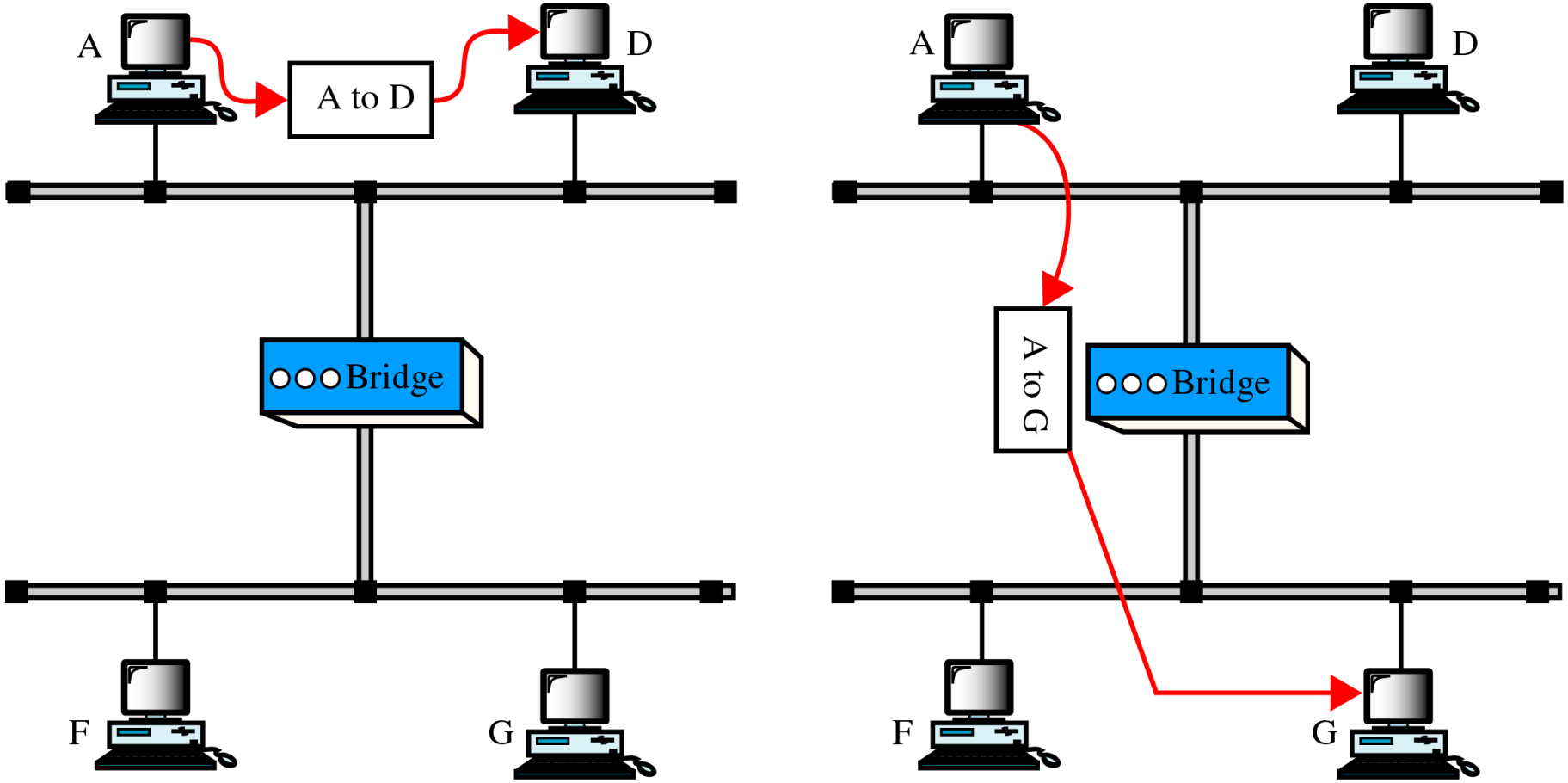
MOTIVATION OF USAGE OF BRIDGES

- » Overcomes the distance limitations of LANs.
- » Connects LANs that use different technologies (Token Ring, Fast Ethernet, Gigabit Ethernet, 10 Gigabit Ethernet, Fiber Distributed Data Interface (FDDI)).
- » Connects LANs built by different organizations.

A Bridge in the OSI Model



Function of a Bridge



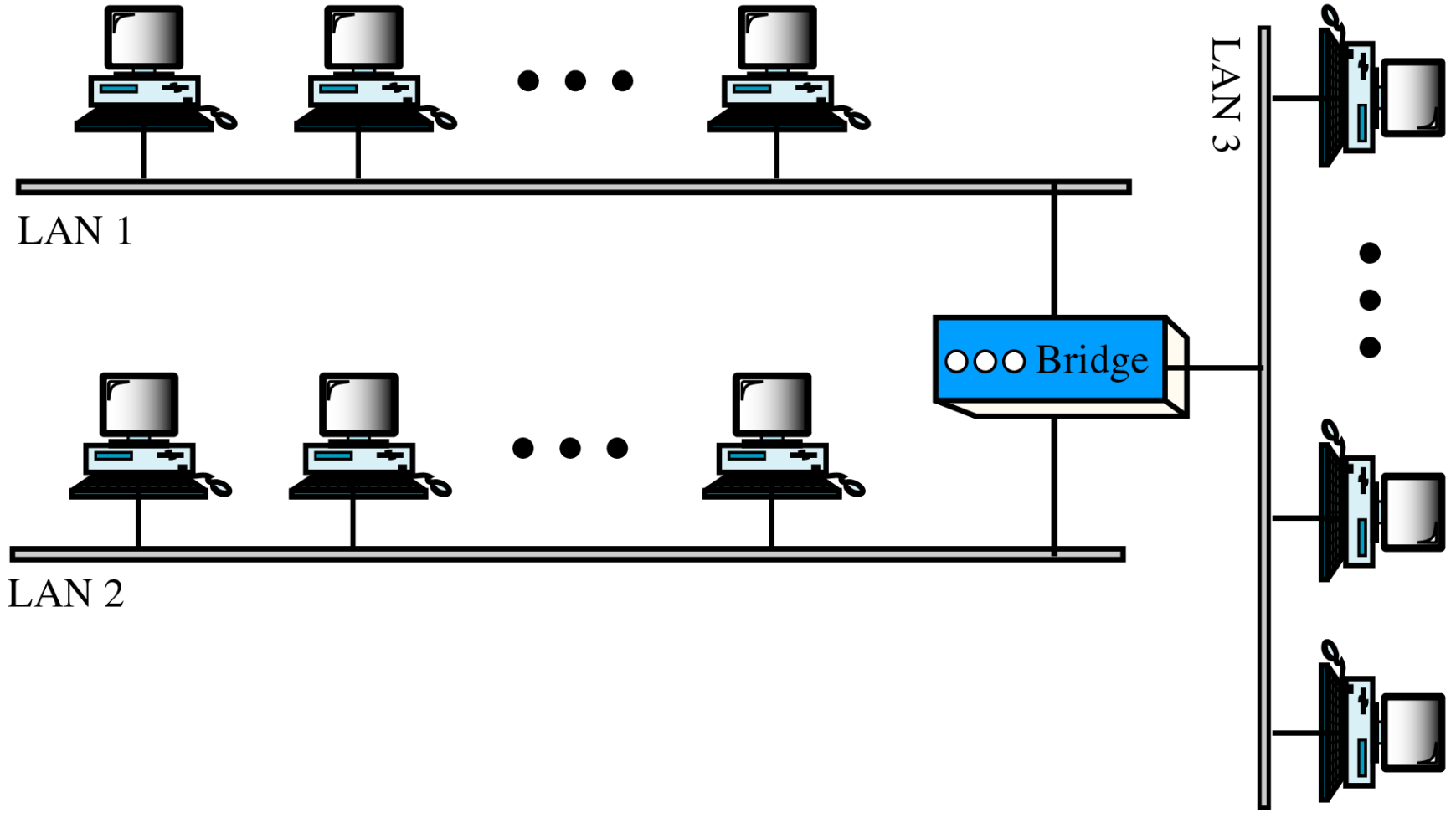
a. A packet from A to D

b. A packet from A to G

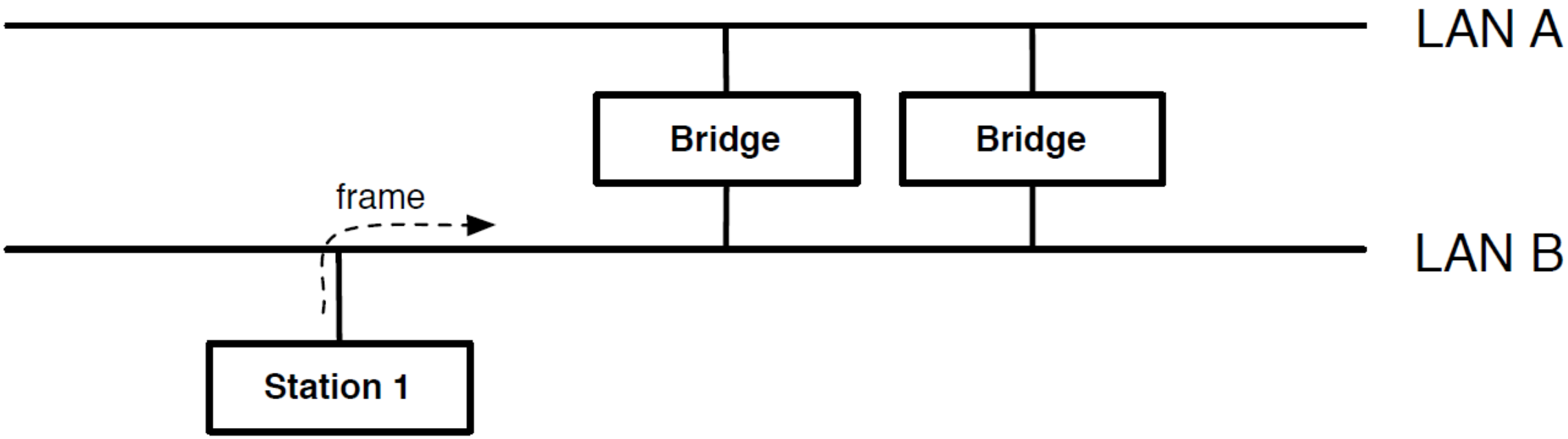
BRIDGE FUNCTIONS

- » Broadcast on one LAN everything it receives from another LAN.
- » As a result, the network appears to all machines as a single LAN

Multiport Bridge



PARALLELL BRIDGES?



In general, this problem arises with any topology containing loops.
Solution? Avoid loops by Bridge Data Unit Protocol.

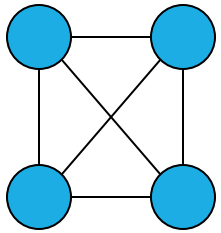
Solution? Avoid loops by Bridge Data Unit Protocol

Spanning Trees

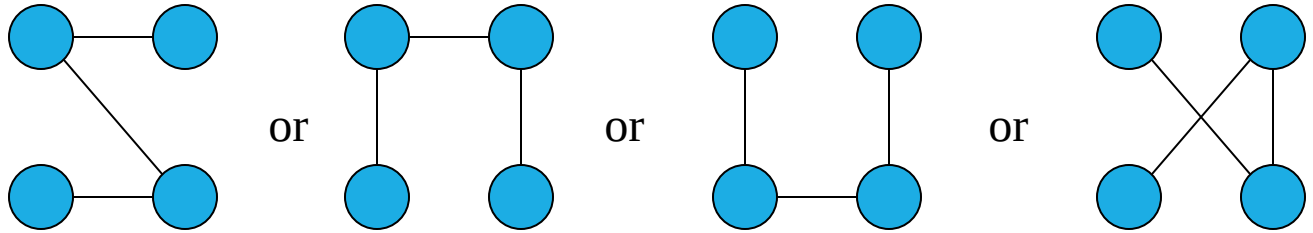
A spanning tree of a graph is just a subgraph that contains all the vertices (nodes) and is a tree.

A graph may have many spanning trees.

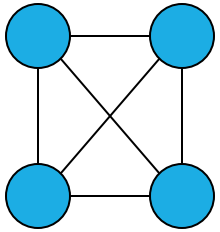
Graph A



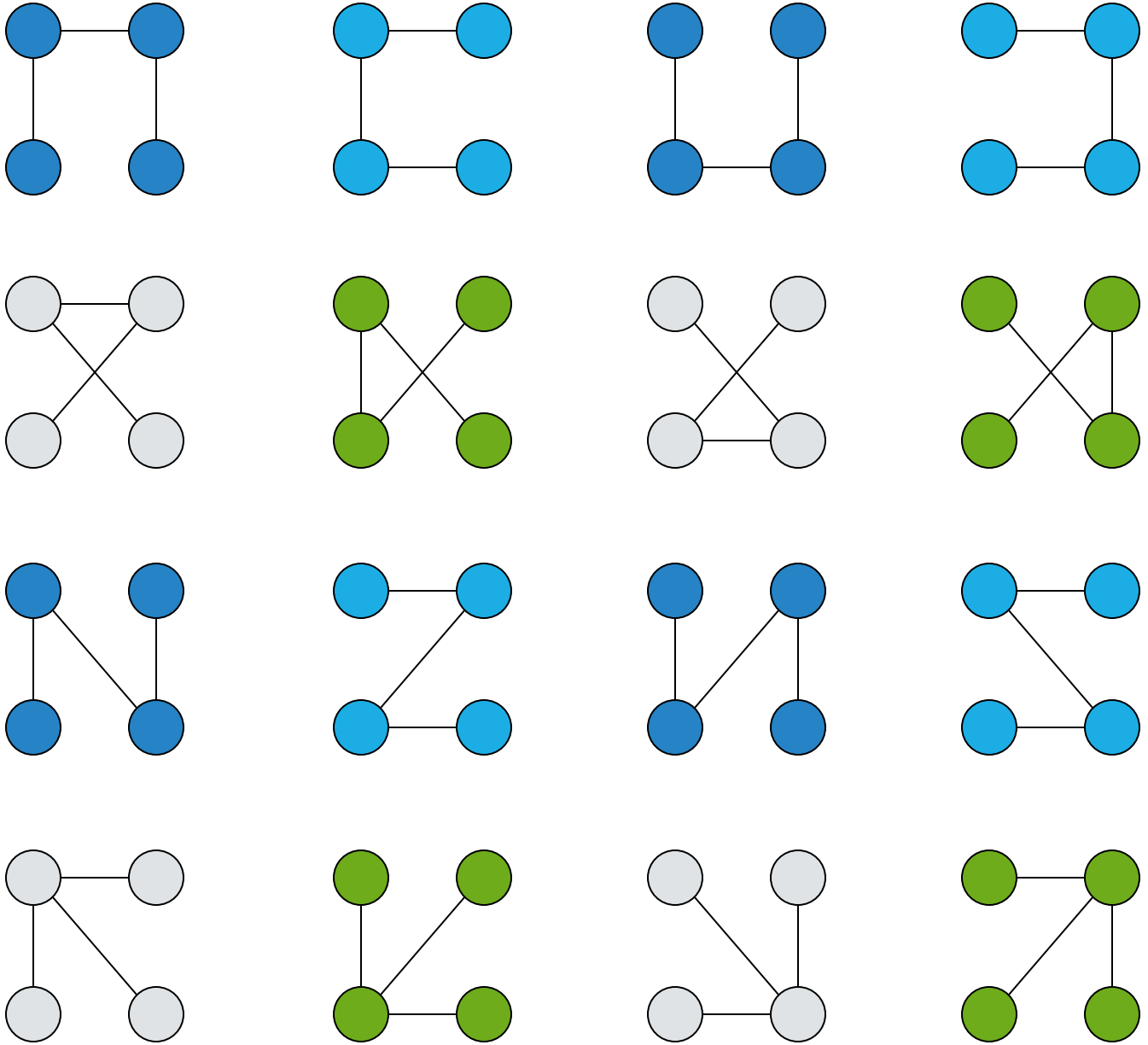
Some Spanning Trees from Graph A



Complete Graph

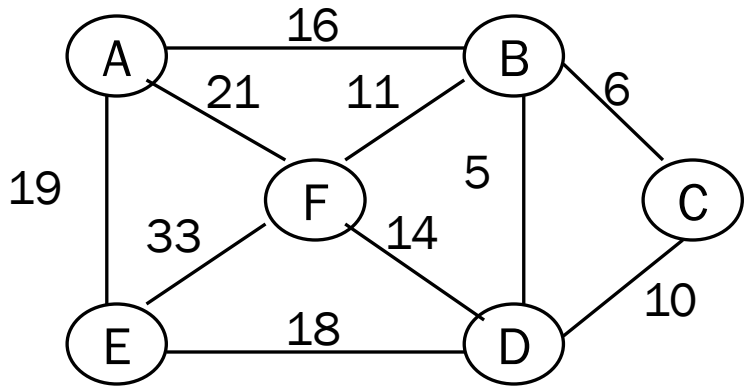


All 16 of its Spanning Trees

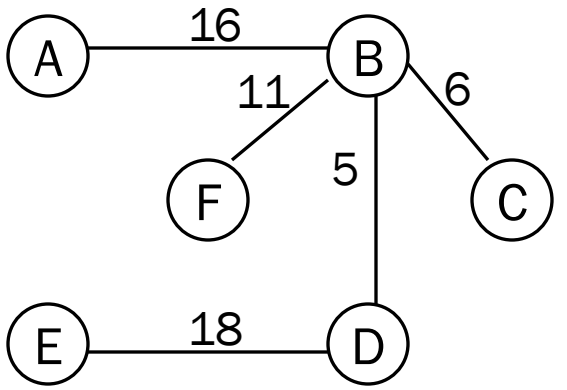


MINIMUM-COST SPANNING TREES

- Suppose you have a connected undirected graph with a weight (or cost) associated with each edge
- The cost of a spanning tree would be the sum of the costs of its edges
- A minimum-cost spanning tree is a spanning tree that has the lowest cost



A connected, undirected graph

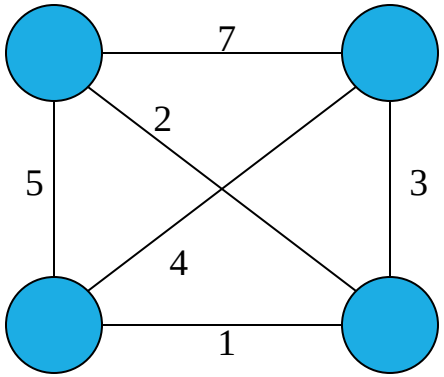


A minimum-cost spanning tree

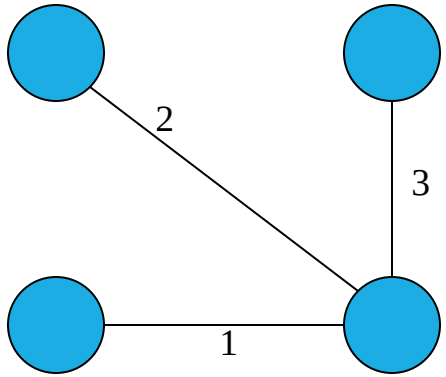
Minimum Spanning Trees

The Minimum Spanning Tree for a given graph is the Spanning Tree of minimum cost for that graph.

Complete Graph



Minimum Spanning Tree



Algorithms for Obtaining the Minimum Spanning Tree

- Kruskal's Algorithm
- Prim's Algorithm

FINDING SPANNING TREES

- There are two basic algorithms for finding minimum-cost spanning trees, and both are greedy algorithms

- **Kruskal's algorithm:** Start with *no* nodes or edges in the spanning tree, and repeatedly add the cheapest edge that does not create a cycle
  - Here, we consider the spanning tree to consist of edges only

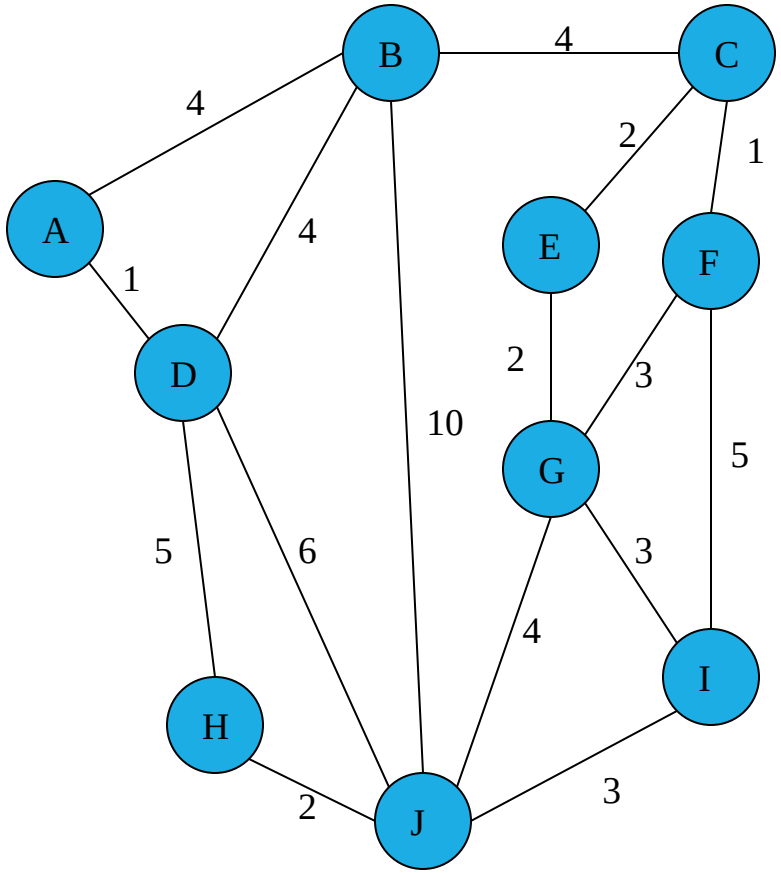
- **Prim's algorithm:** Start with any *one node* in the spanning tree, and repeatedly add the cheapest edge, and the node it leads to, for which the node is not already in the spanning tree.
  - Here, we consider the spanning tree to consist of both nodes and edges

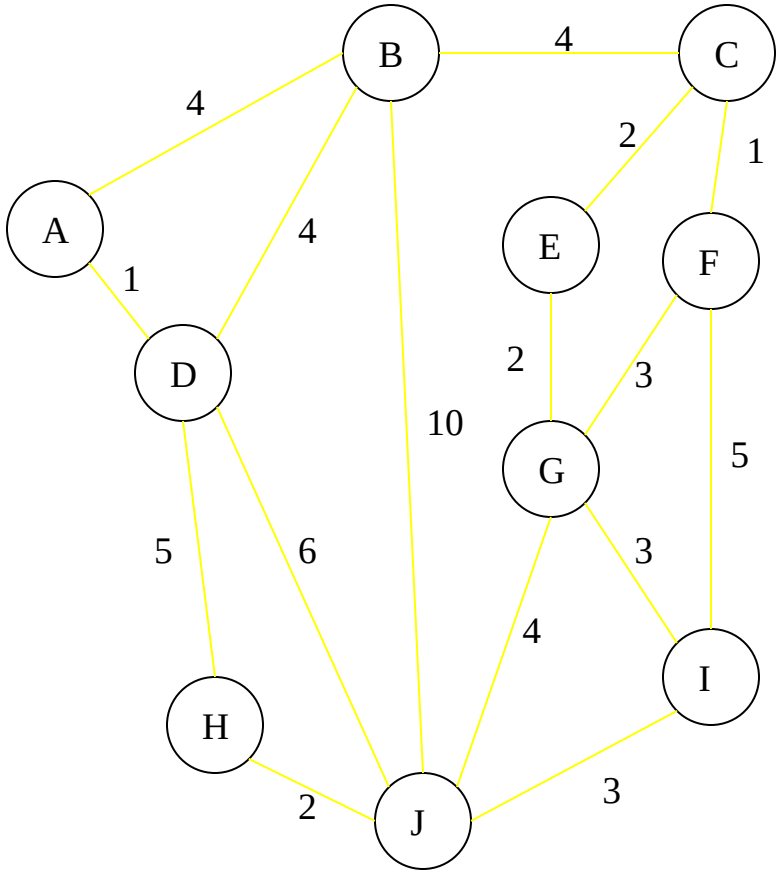
The steps are:

1. The forest is constructed - with each node in a separate tree.
2. The edges are placed in a priority queue.
3. Until we've added n-1 edges,
    1. Extract the cheapest edge from the queue,
    2. If it forms a cycle, reject it,
    3. Else add it to the forest. Adding it to the forest will join two trees together.

Every step will have joined two trees in the forest together, so that at the end, there will only be one tree in T.

Complete Graph

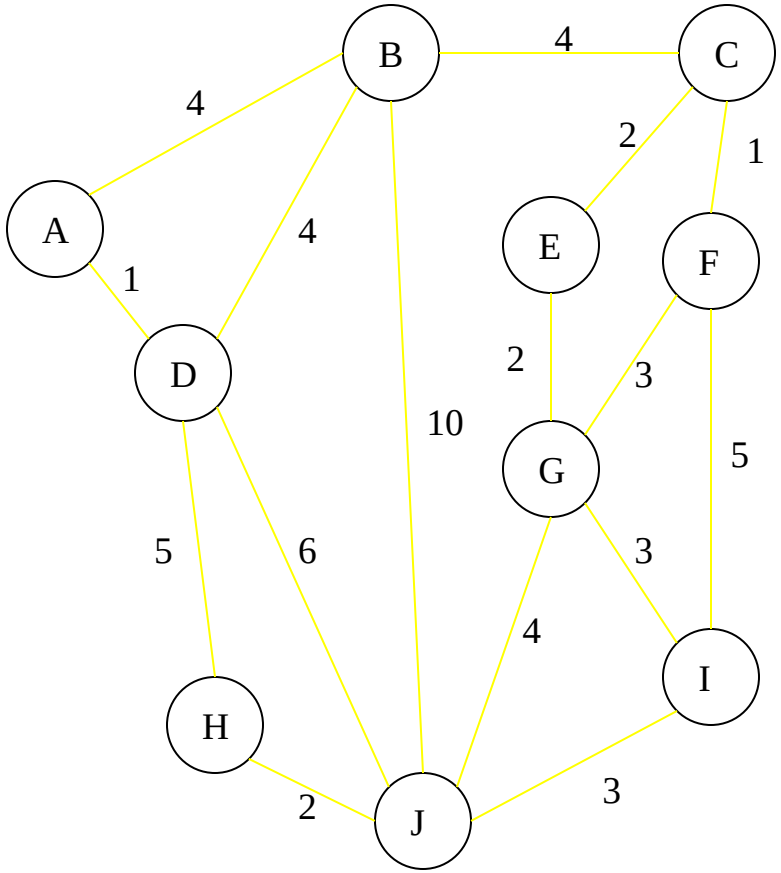


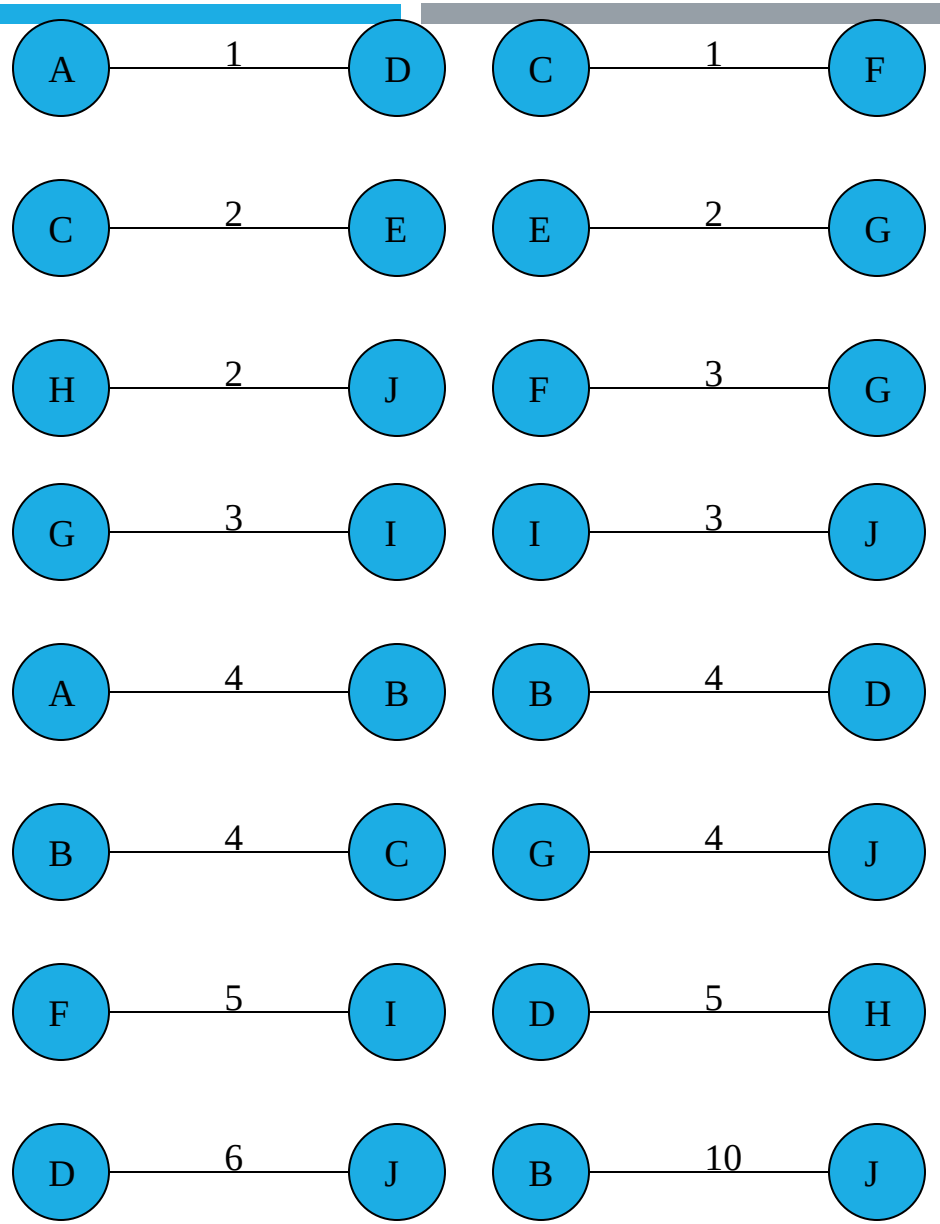


A	4	B	A	1	D
B	4	C	B	4	D
B	10	J	C	2	E
C	1	F	D	5	H
D	6	J	E	2	G
F	3	G	F	5	I
G	3	I	G	4	J
H	2	J	I	3	J

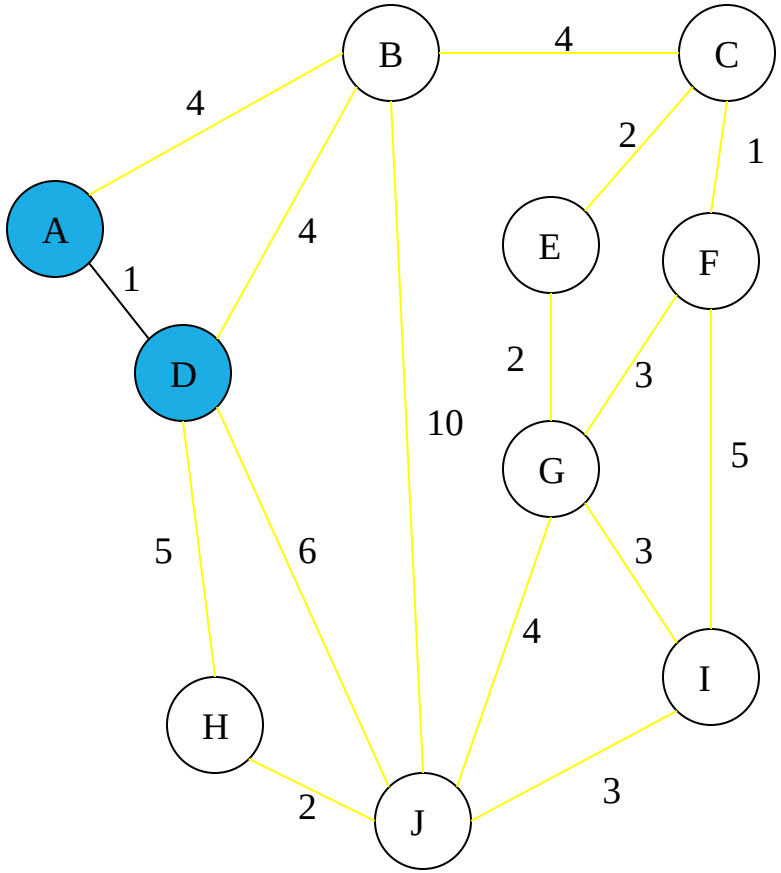
Sort Edges

(in reality they are placed in a priority queue - not sorted - but sorting them makes the algorithm easier to visualize)





Add Edge



A 1 D

C 1 F

C 2 E

E 2 G

H 2 J

F 3 G

G 3 I

I 3 J

A 4 B

B 4 D

B 4 C

G 4 J

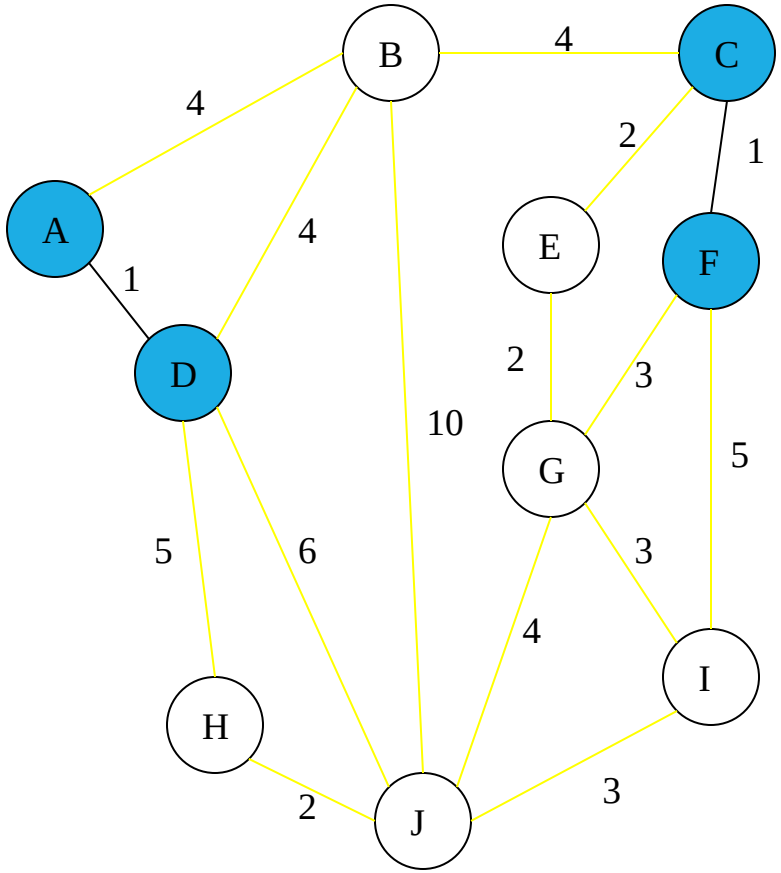
F 5 I

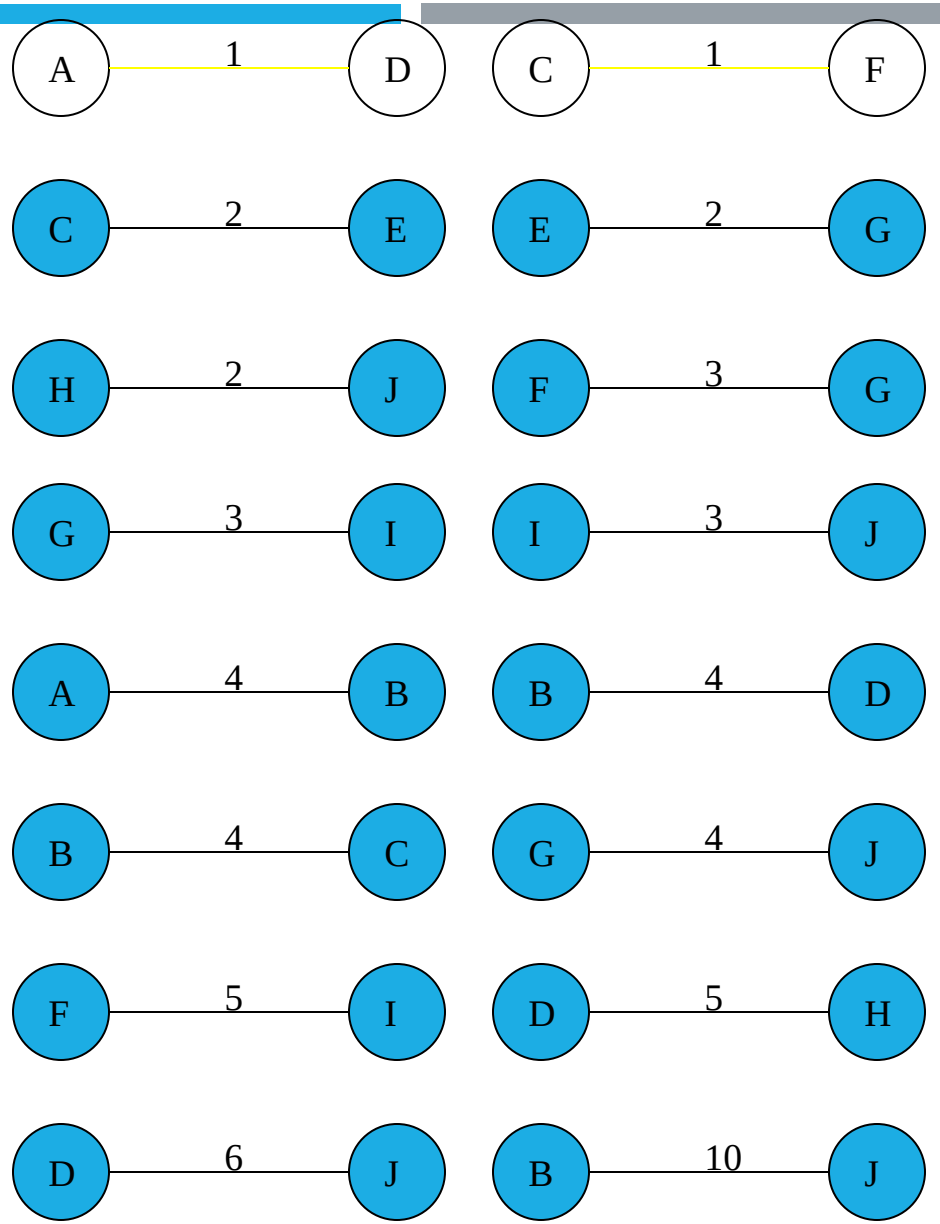
D 5 H

D 6 J

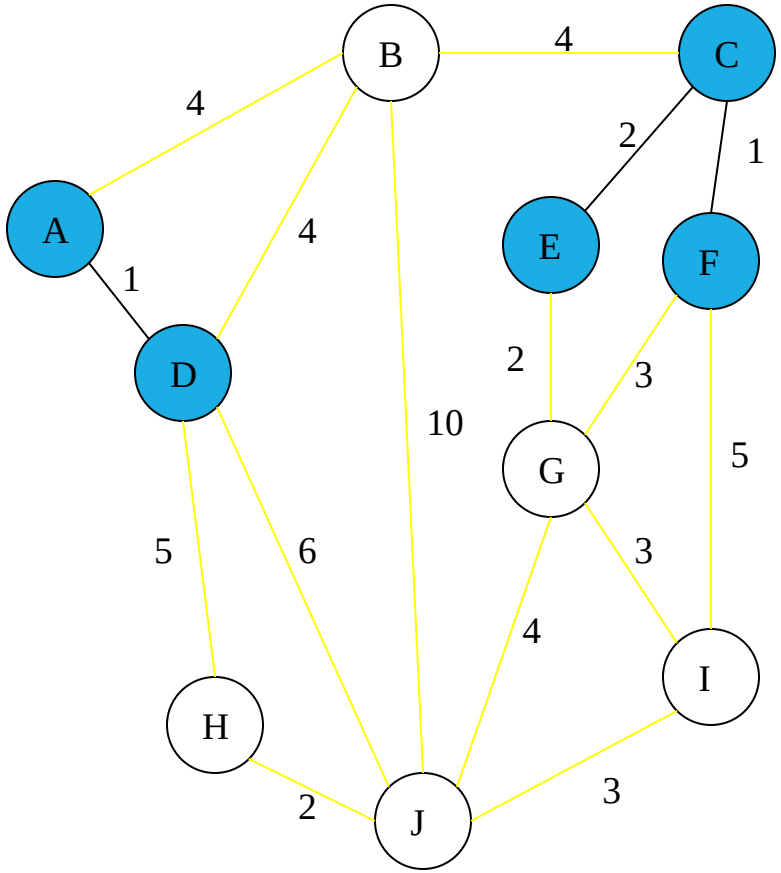
B 10 J

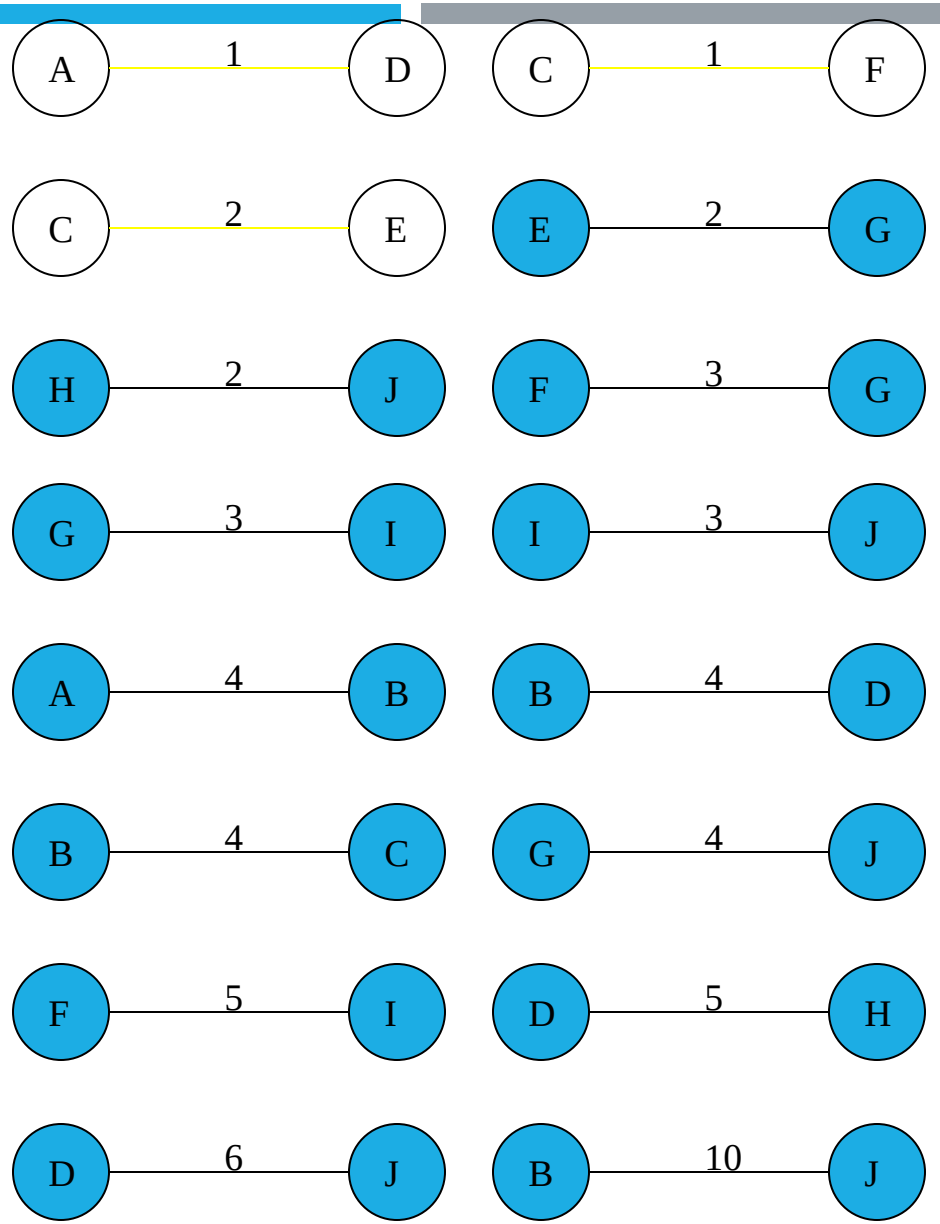
Add Edge


A
B
C
D
E
F
G
H
I
J
4
4
4
1
2
1
10
2
3
5
5
6
3
4
2
3

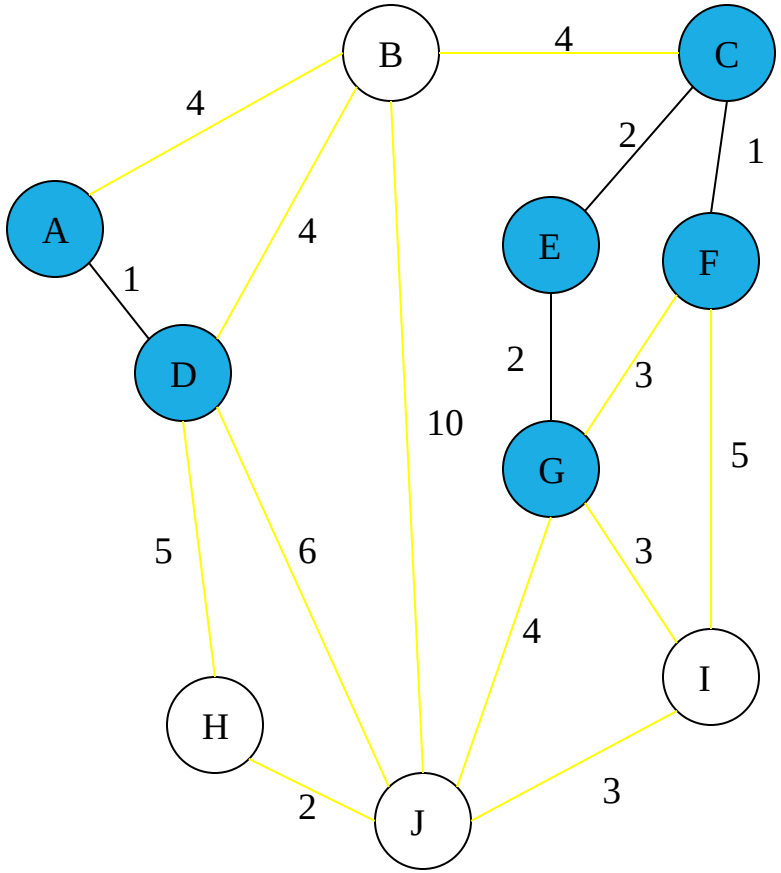

A 1 D
C 1 F
C 2 E
E 2 G
H 2 J
F 3 G
G 3 I
I 3 J
A 4 B
B 4 D
B 4 C
G 4 J
F 5 I
D 5 H
D 6 J
B 10 J

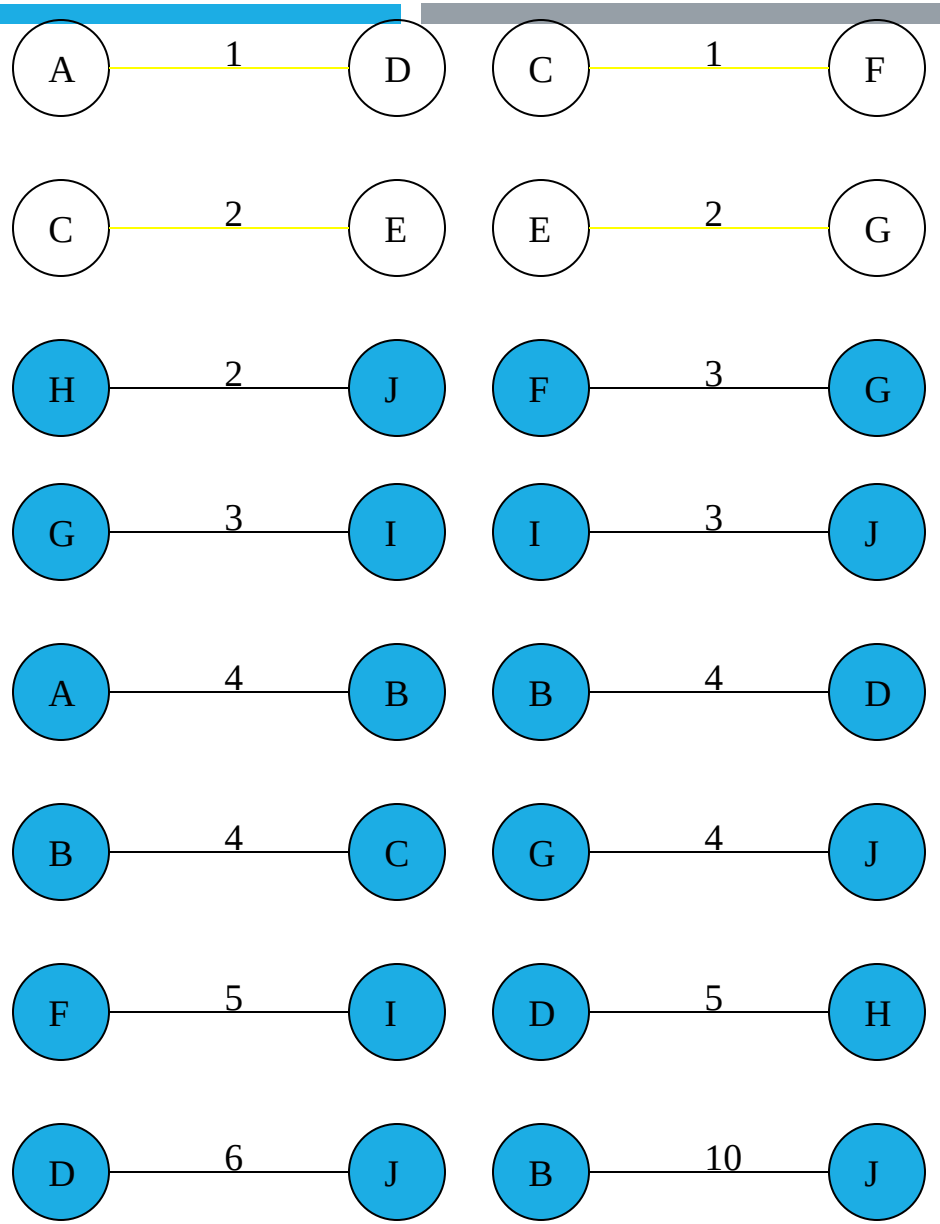
Add Edge


A
B
C
D
E
F
G
H
I
J
4
4
4
1
2
1
10
2
3
5
5
6
3
4
2
3

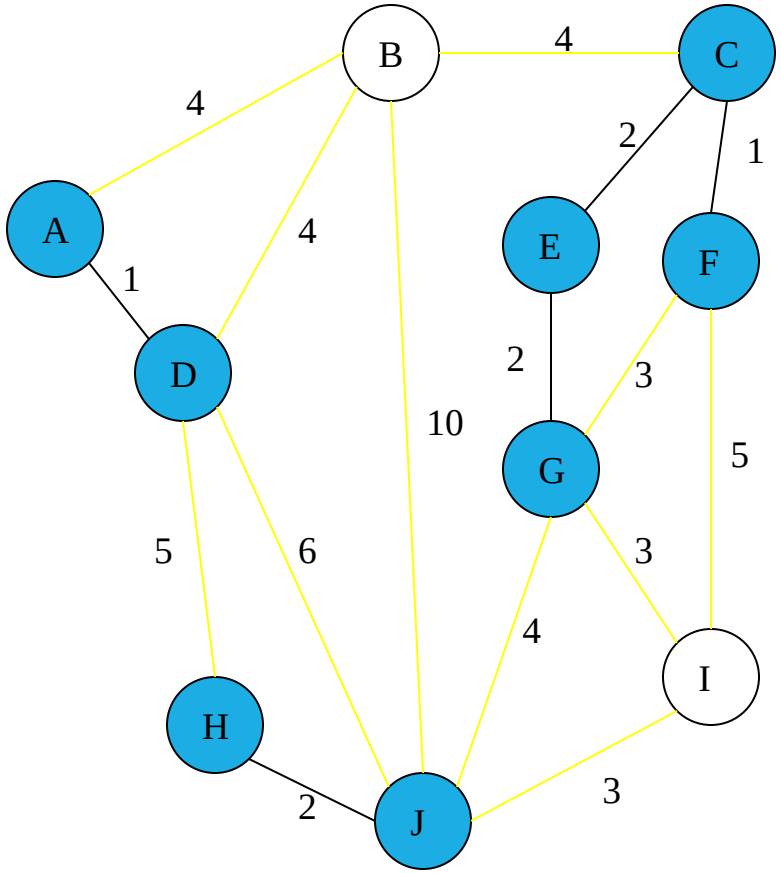

A 1 D
C 1 F
C 2 E
E 2 G
H 2 J
F 3 G
G 3 I
I 3 J
A 4 B
B 4 D
B 4 C
G 4 J
F 5 I
D 5 H
D 6 J
B 10 J

Add Edge


A
B
C
D
E
F
G
H
I
J
4
4
4
1
2
1
10
2
3
5
5
6
3
4
2
3


A 1 D
C 1 F
C 2 E
E 2 G
H 2 J
F 3 G
G 3 I
I 3 J
A 4 B
B 4 D
B 4 C
G 4 J
F 5 I
D 5 H
D 6 J
B 10 J

Add Edge



A 1 D

C 1 F

C 2 E

E 2 G

H 2 J

F 3 G

G 3 I

I 3 J

A 4 B

B 4 D

B 4 C

G 4 J

F 5 I

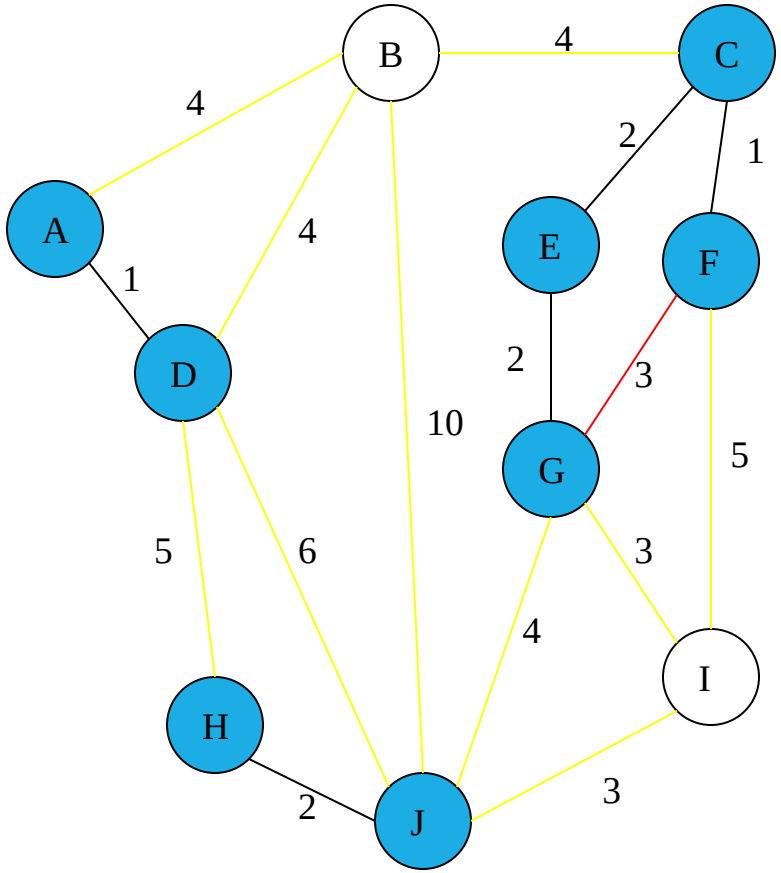
D 5 H

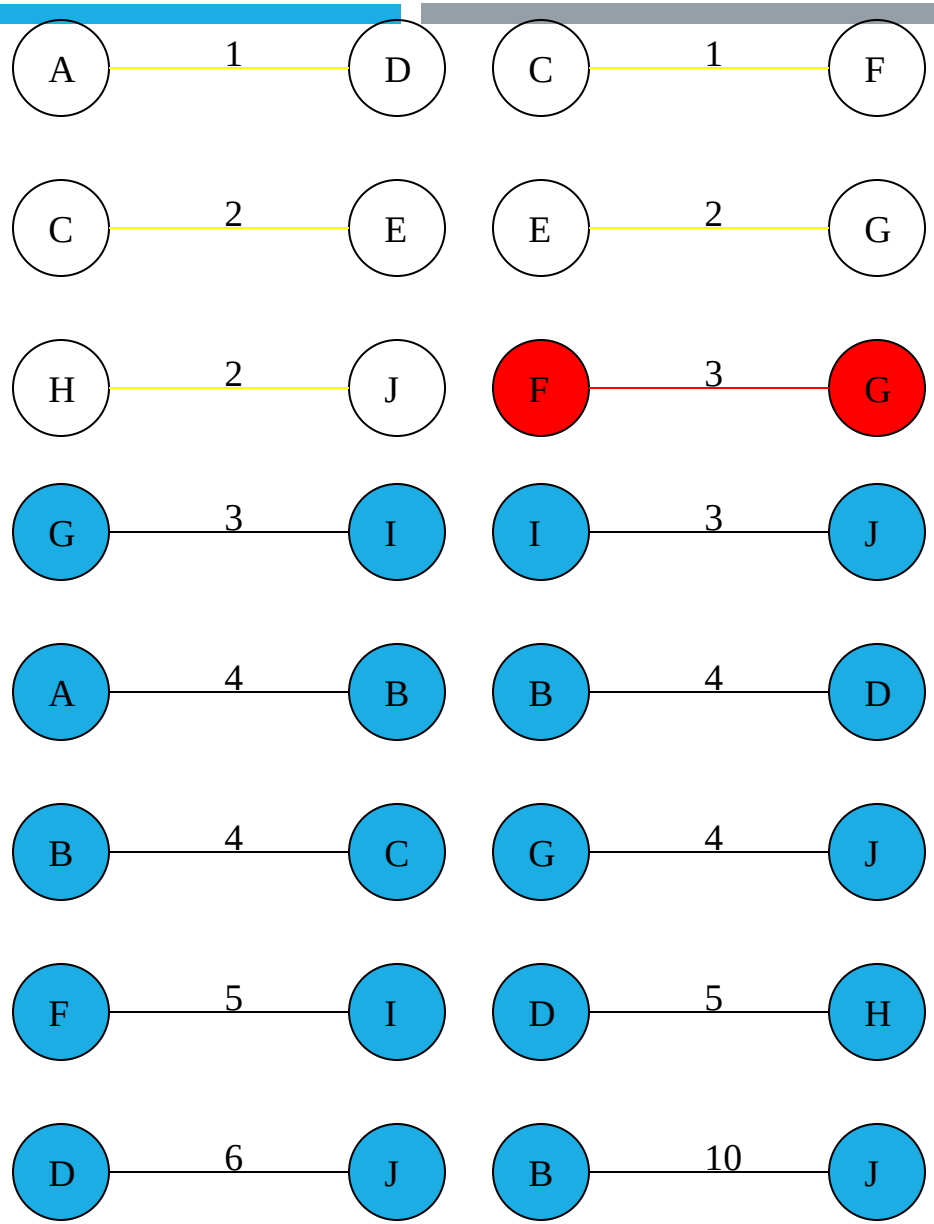
D 6 J

B 10 J

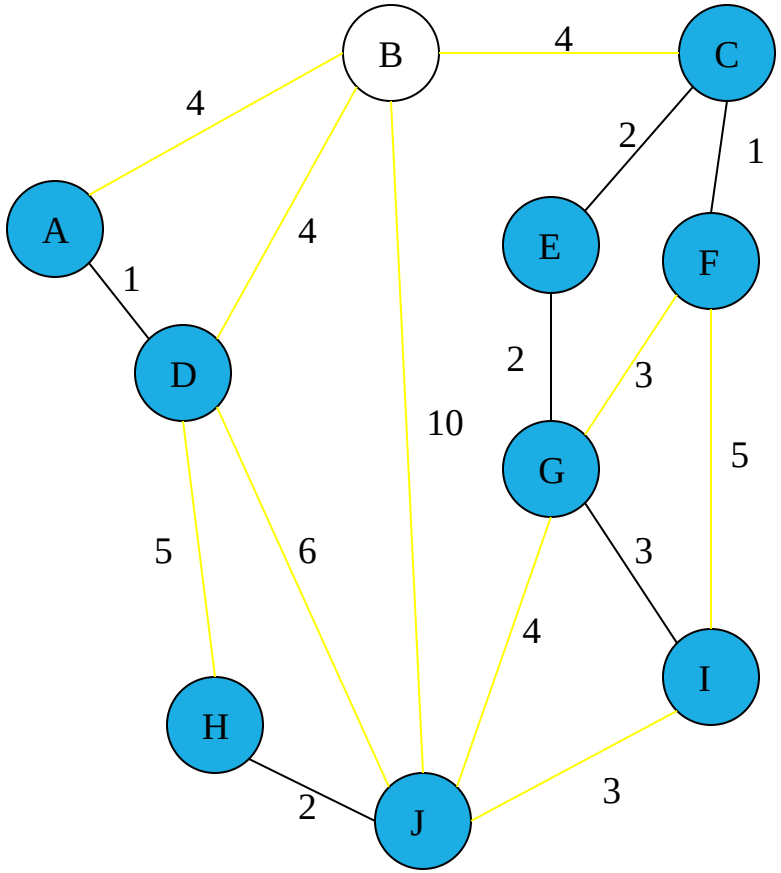
Cycle

Don’t Add Edge





Add Edge



A 1 D

C 1 F

C 2 E

E 2 G

H 2 J

F 3 G

G 3 I

I 3 J

A 4 B

B 4 D

B 4 C

G 4 J

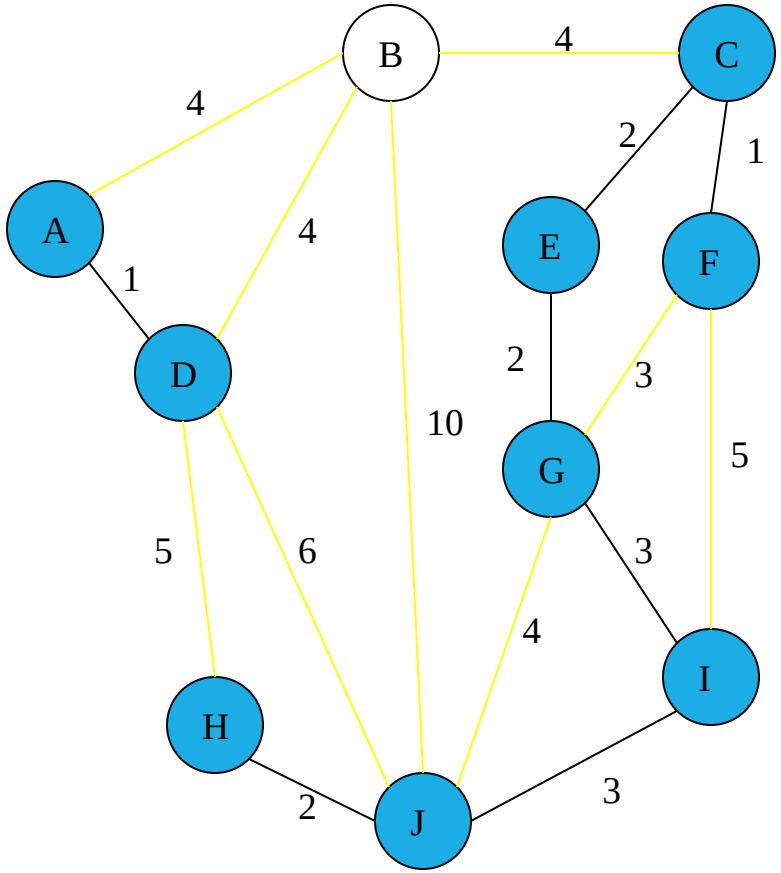
F 5 I

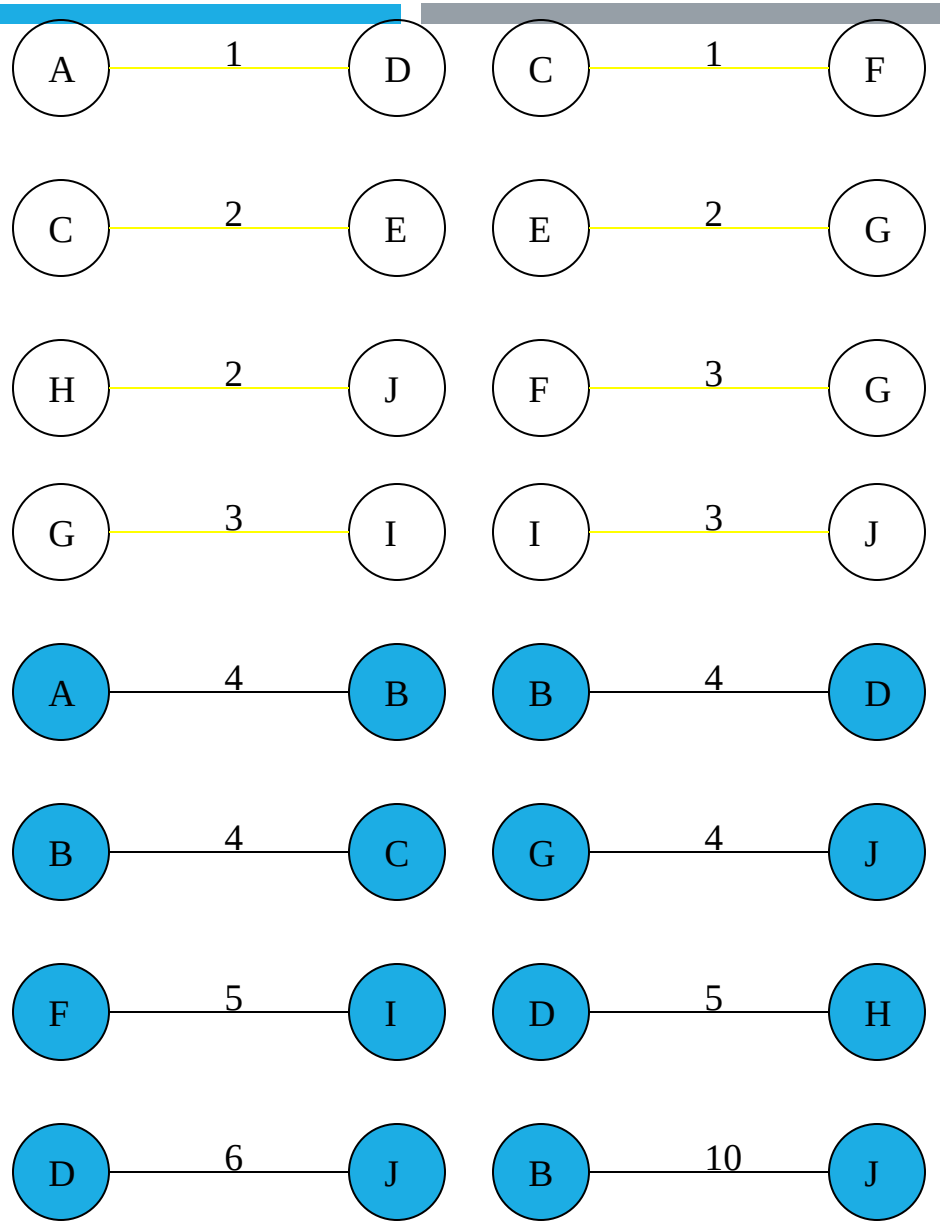
D 5 H

D 6 J

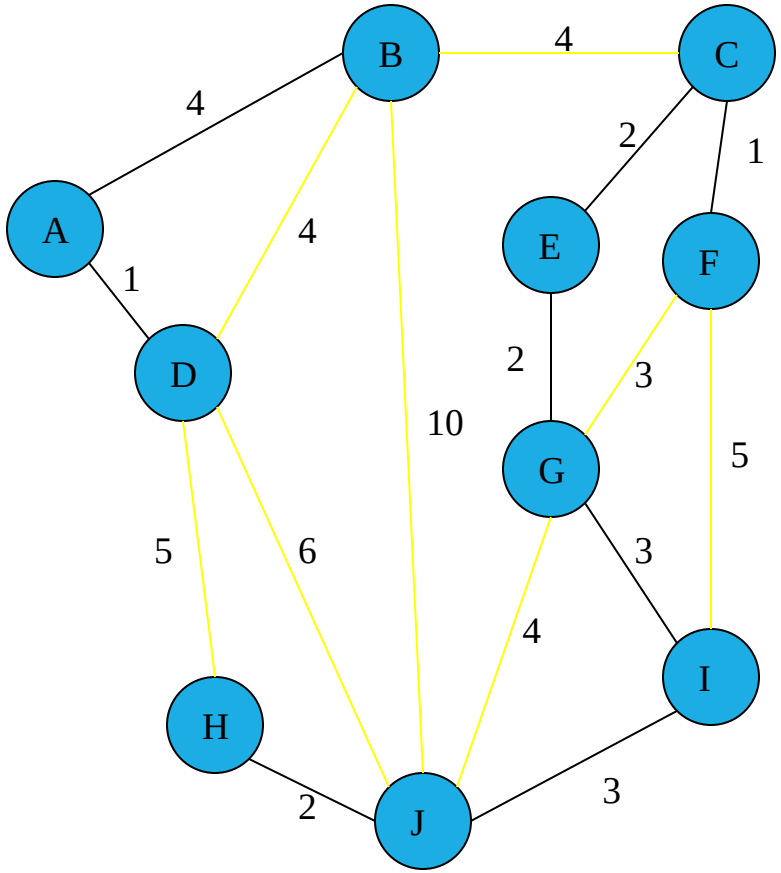
B 10 J

Add Edge


A
B
C
D
E
F
G
H
I
J
4
4
4
1
2
1
2
3
10
5
5
6
3
4
3
2


A 1 D
C 1 F
C 2 E
E 2 G
H 2 J
F 3 G
G 3 I
I 3 J
A 4 B
B 4 D
B 4 C
G 4 J
F 5 I
D 5 H
D 6 J
B 10 J

Add Edge



A 1 D

C 1 F

C 2 E

E 2 G

H 2 J

F 3 G

G 3 I

I 3 J

A 4 B

B 4 D

B 4 C

G 4 J

F 5 I

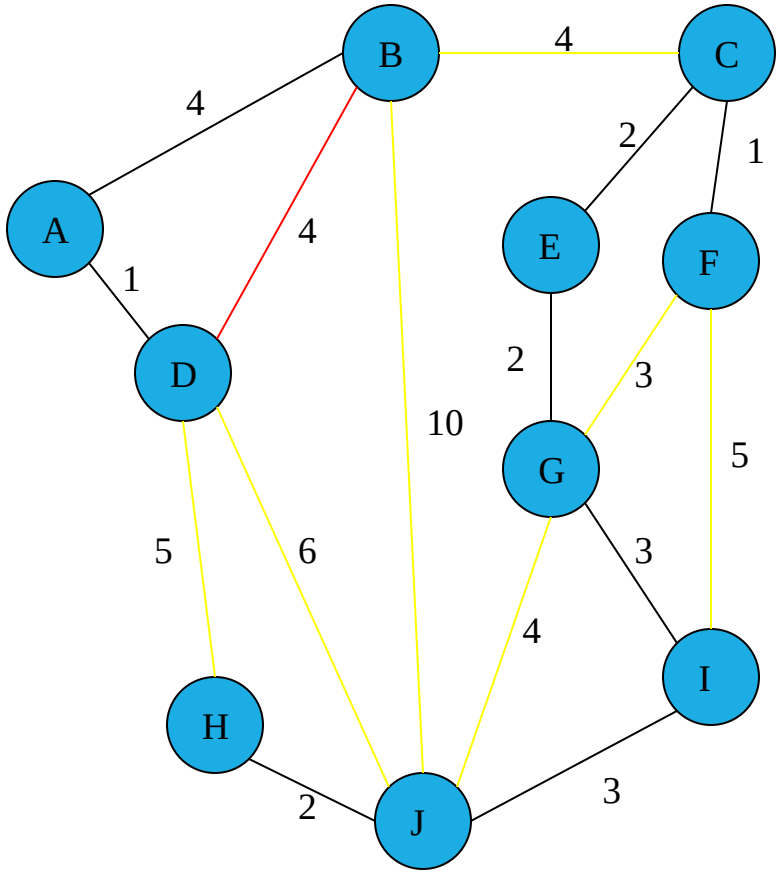
D 5 H

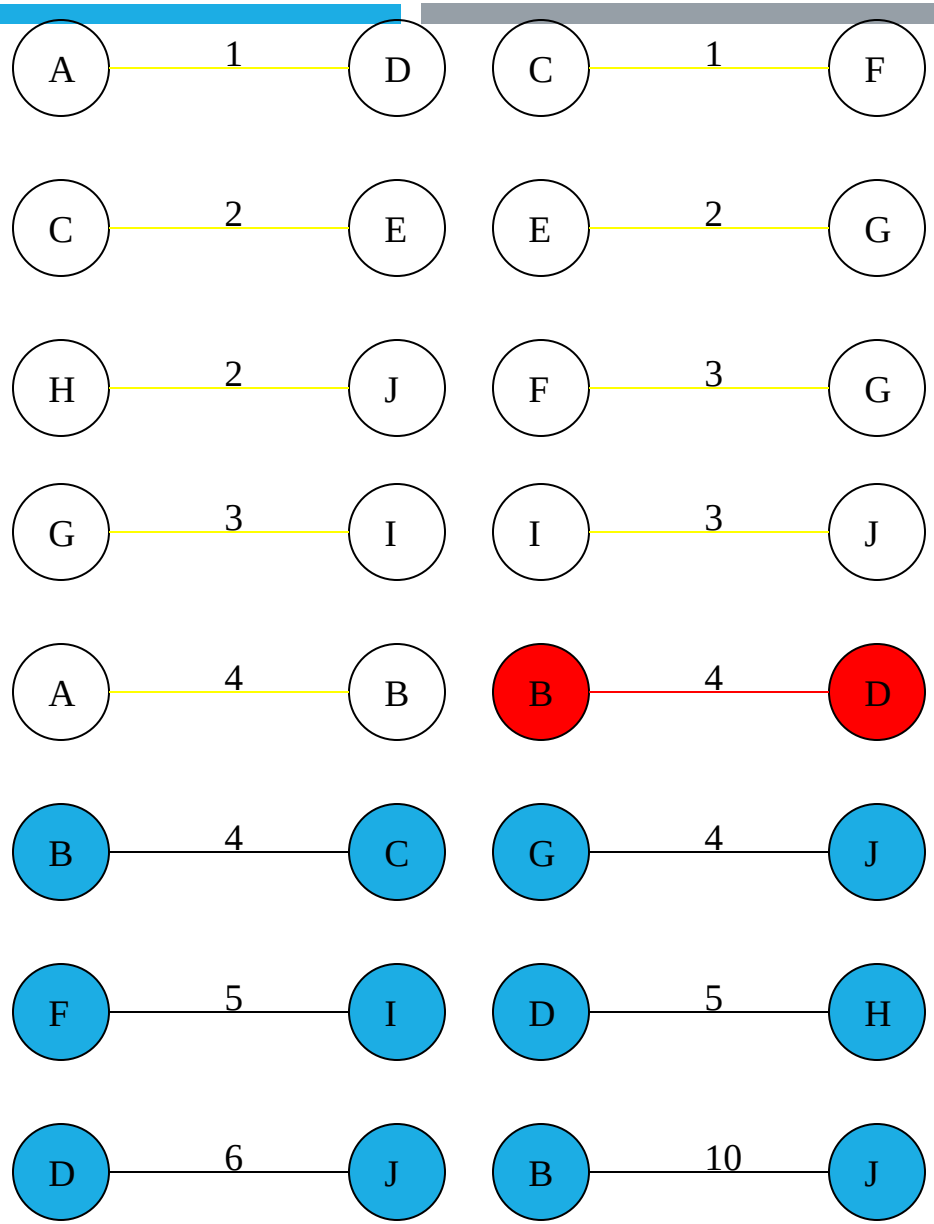
D 6 J

B 10 J

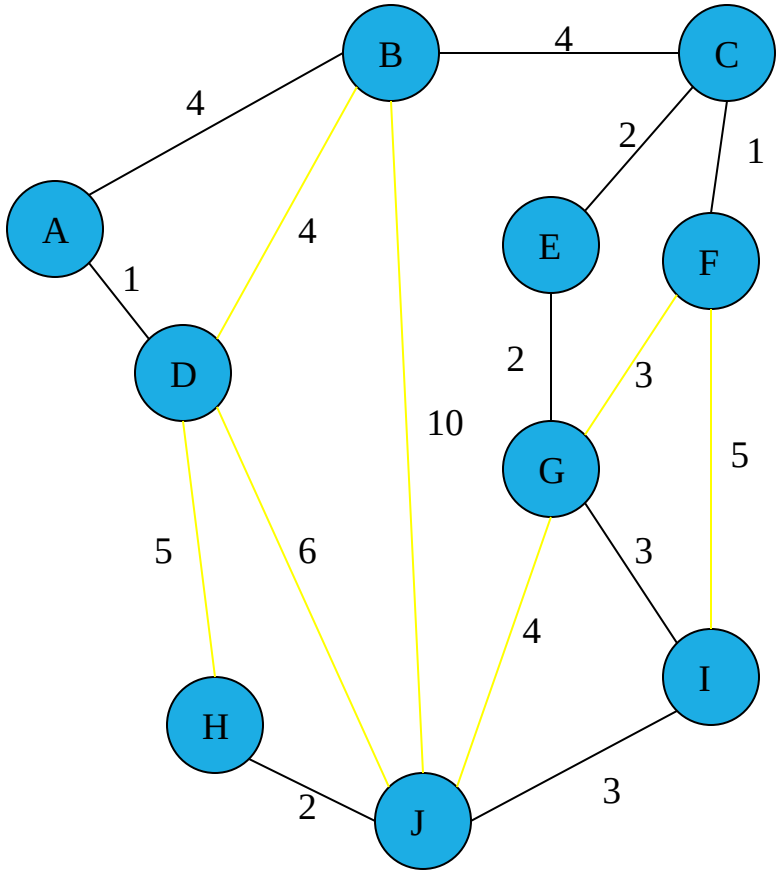
Cycle

Don’t Add Edge



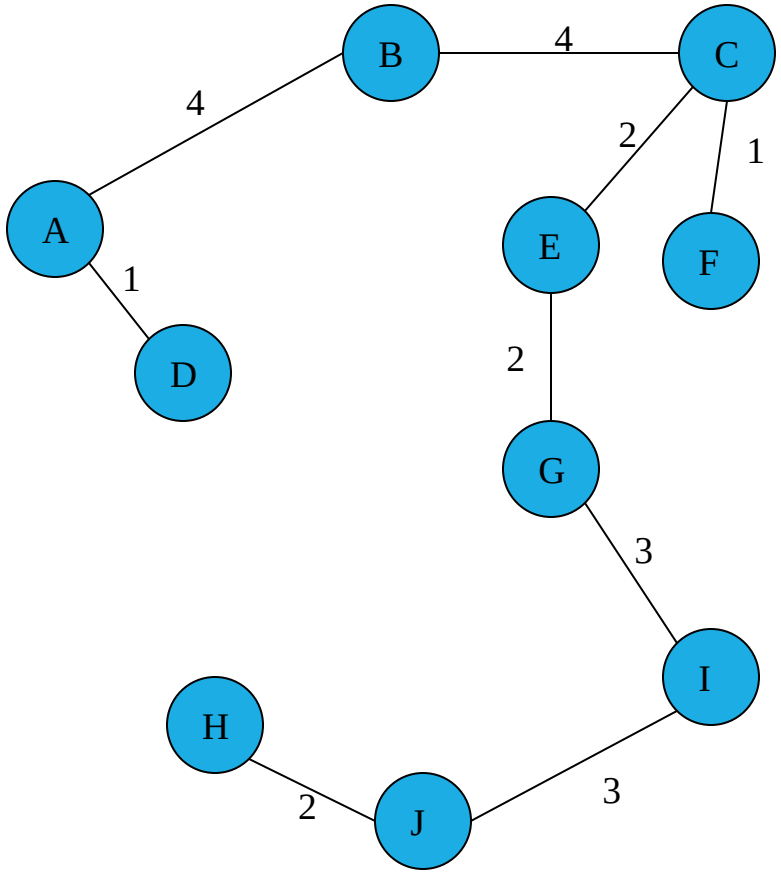


Add Edge

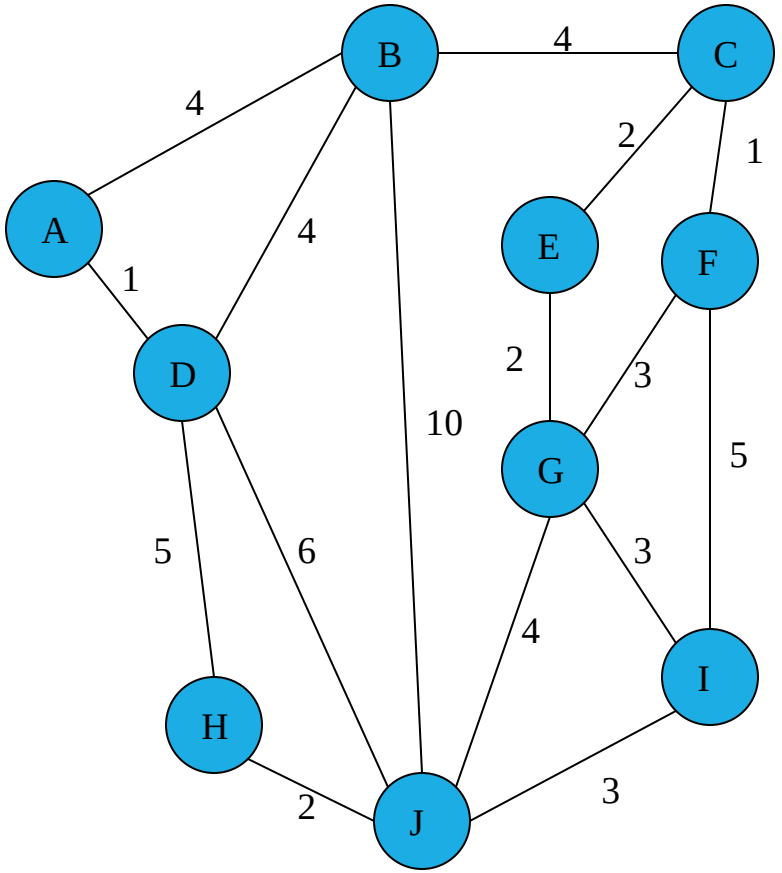


A	1	D	C	1	F
C	2	E	E	2	G
H	2	J	F	3	G
G	3	I	I	3	J
A	4	B	B	4	D
B	4	C	G	4	J
F	5	I	D	5	H
D	6	J	B	10	J

Minimum Spanning Tree



Complete Graph



Prim's Algorithm

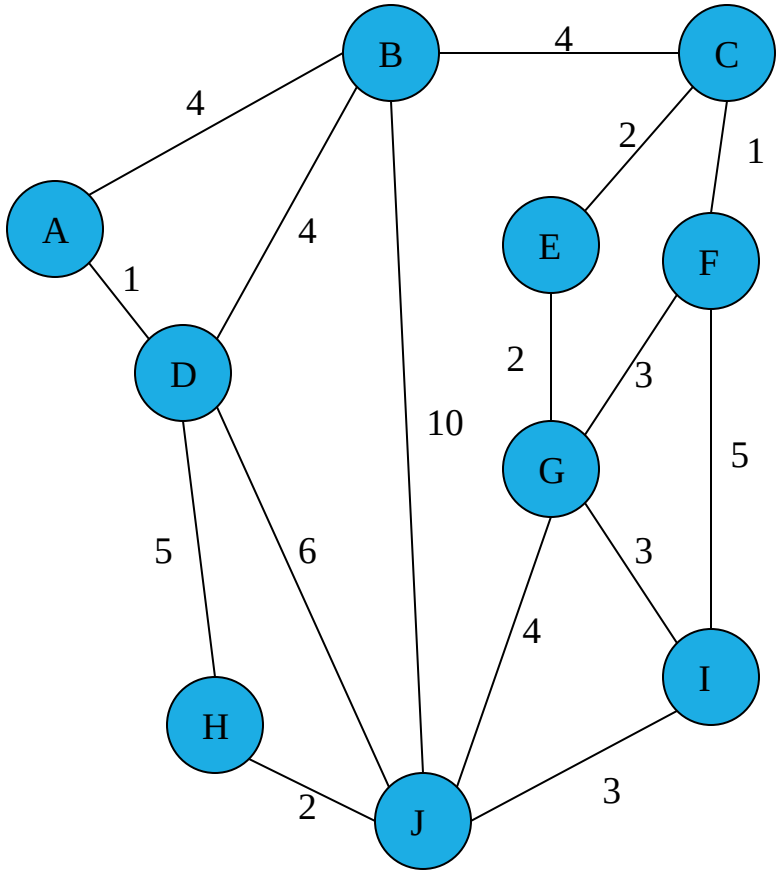
This algorithm starts with one node. It then, one by one, adds a node that is unconnected to the new graph to the new graph, each time selecting the node whose connecting edge has the smallest weight out of the available nodes' connecting edges.

The steps are:

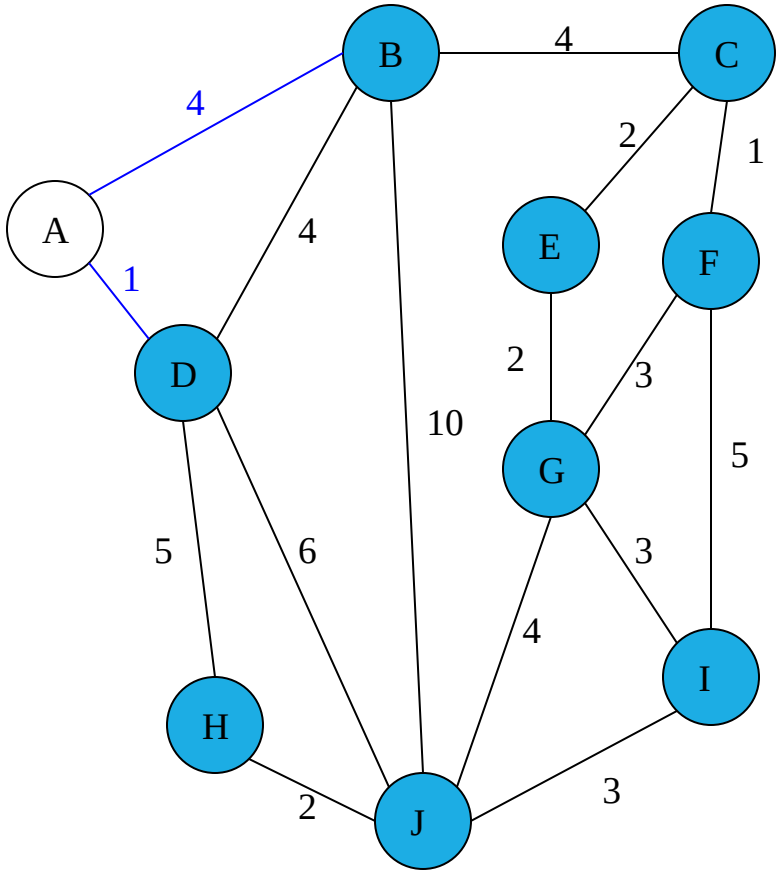
1. The new graph is constructed - with one node from the old graph.
2. While new graph has fewer than n nodes,
    1. Find the node from the old graph with the smallest connecting edge to the new graph,
    2. Add it to the new graph

Every step will have joined one node, so that at the end we will have one graph with all the nodes and it will be a minimum spanning tree of the original graph.

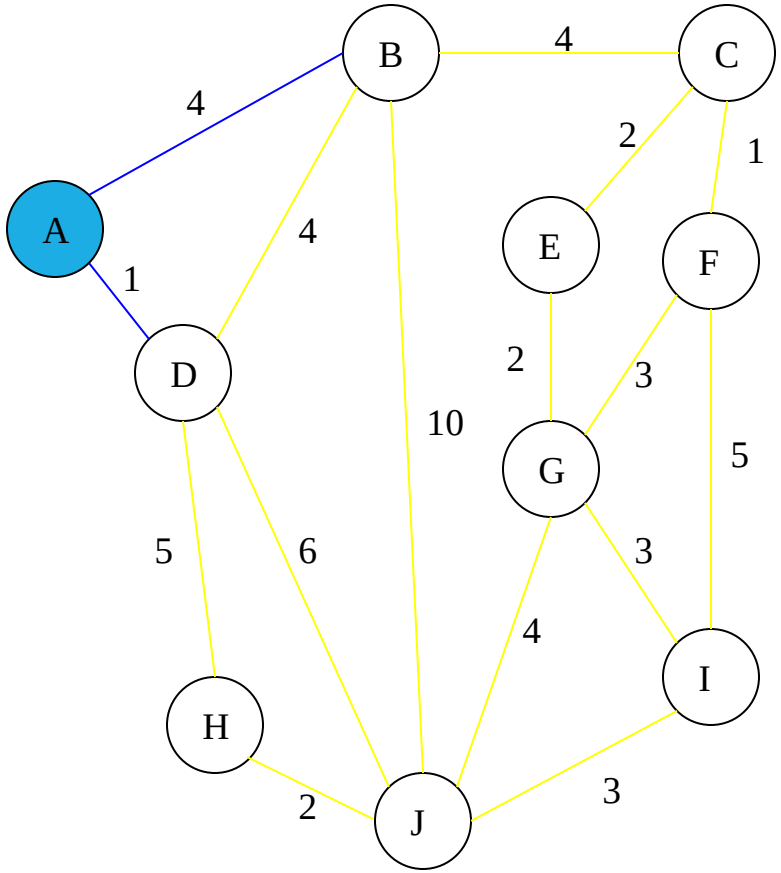
Complete Graph



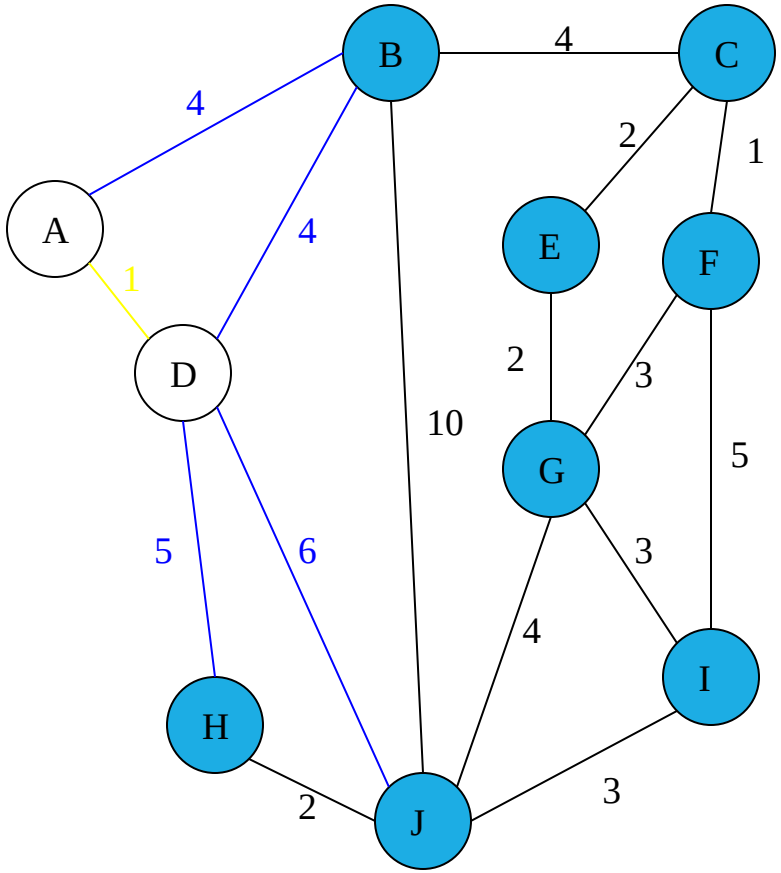
Old Graph



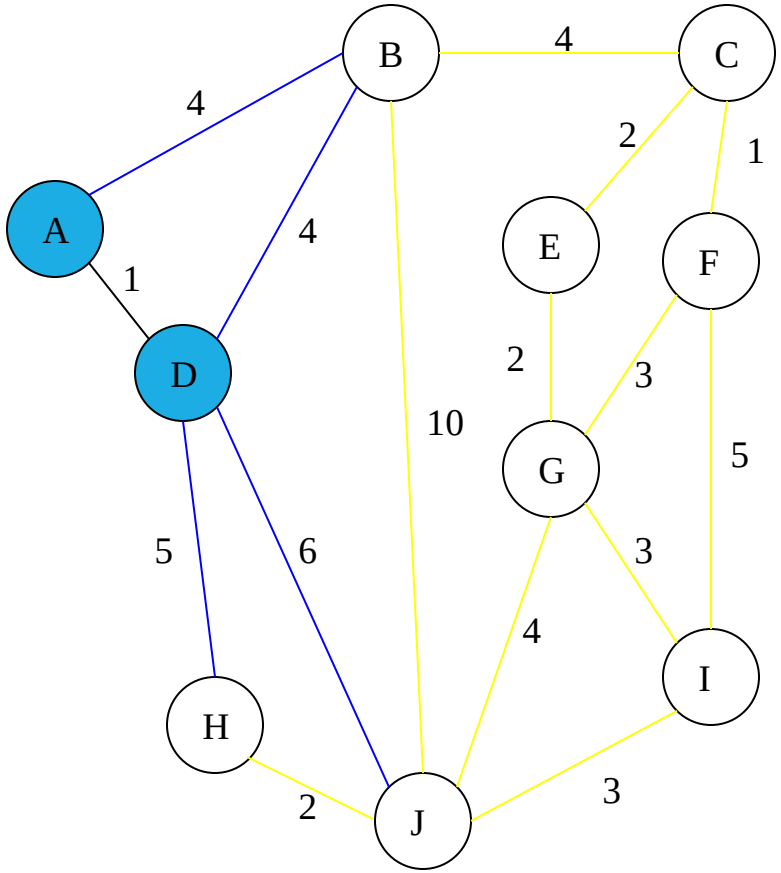
New Graph



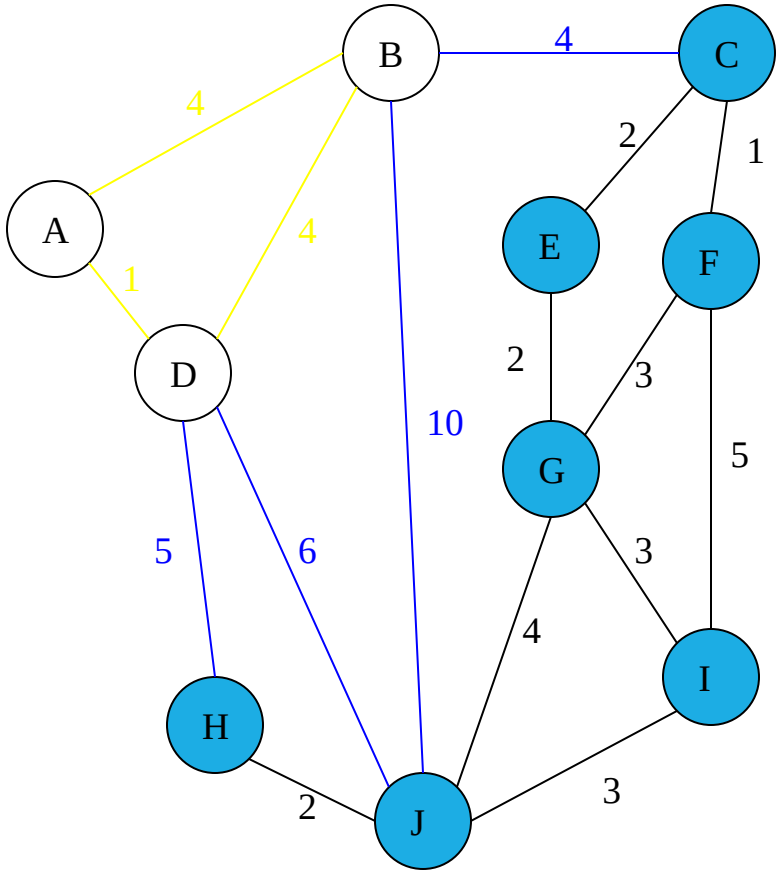
Old Graph



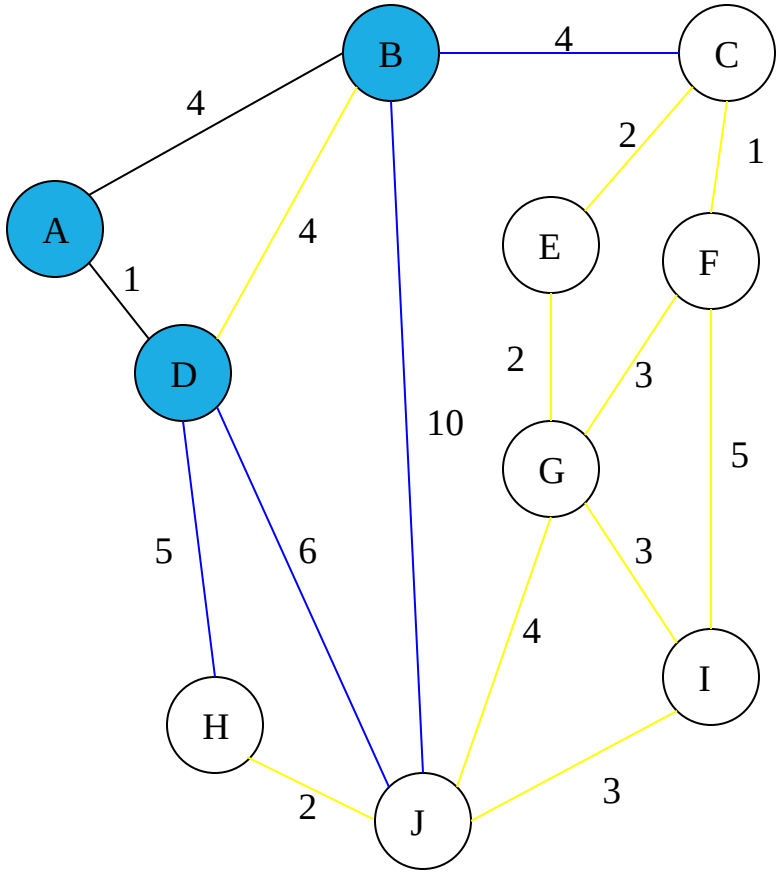
New Graph



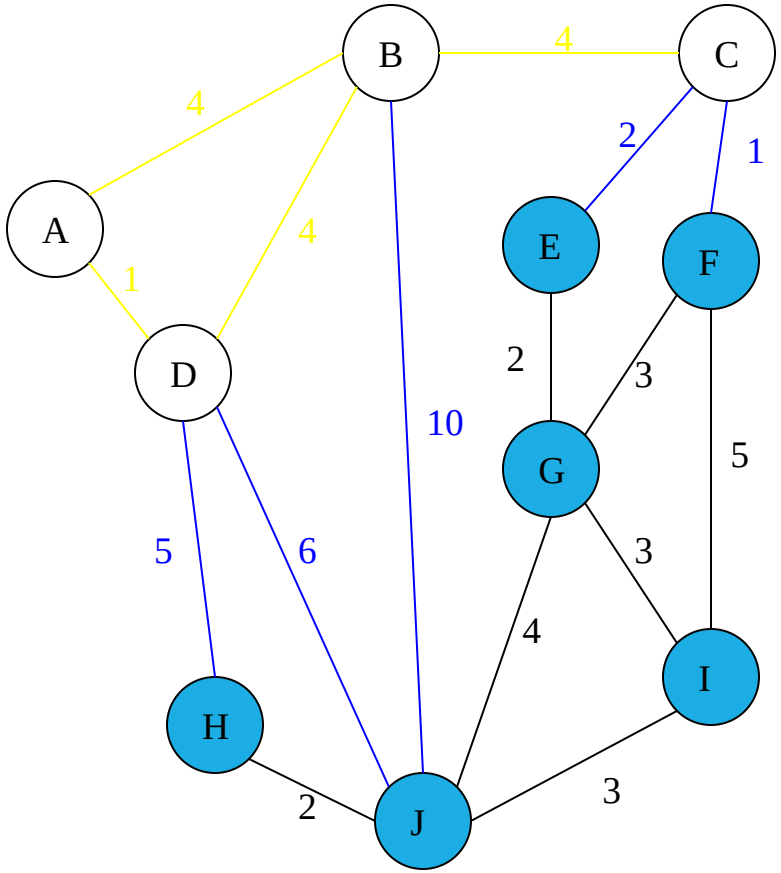
Old Graph



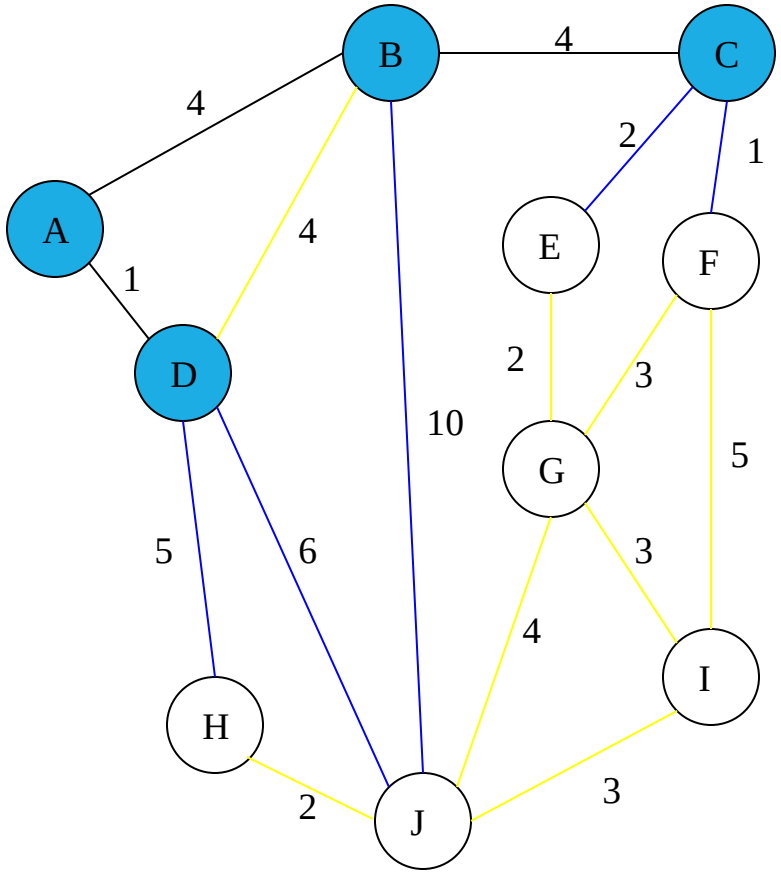
New Graph



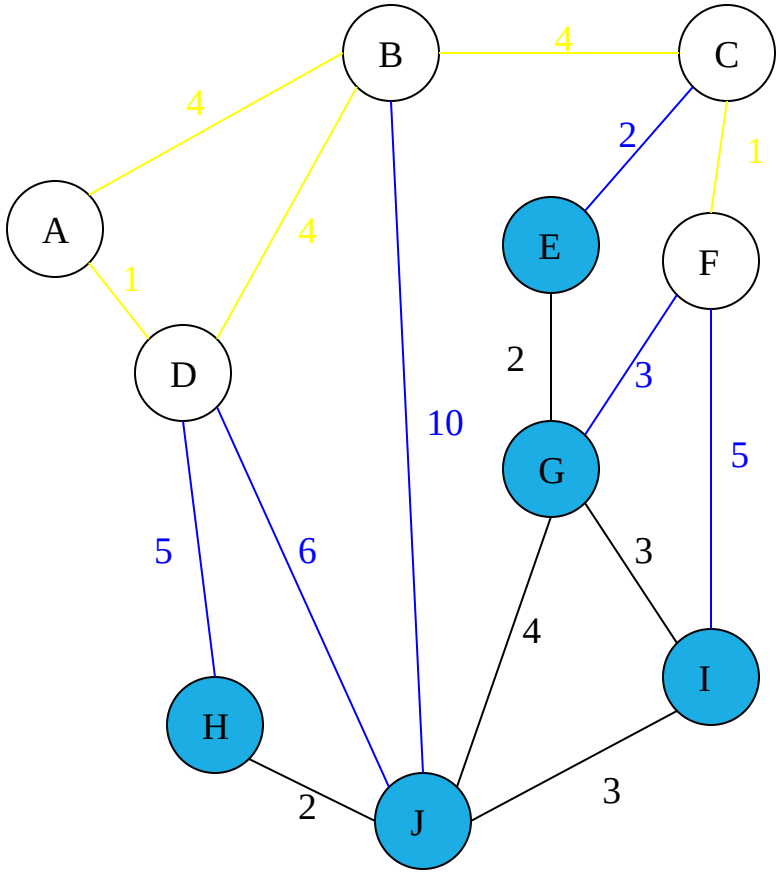
Old Graph



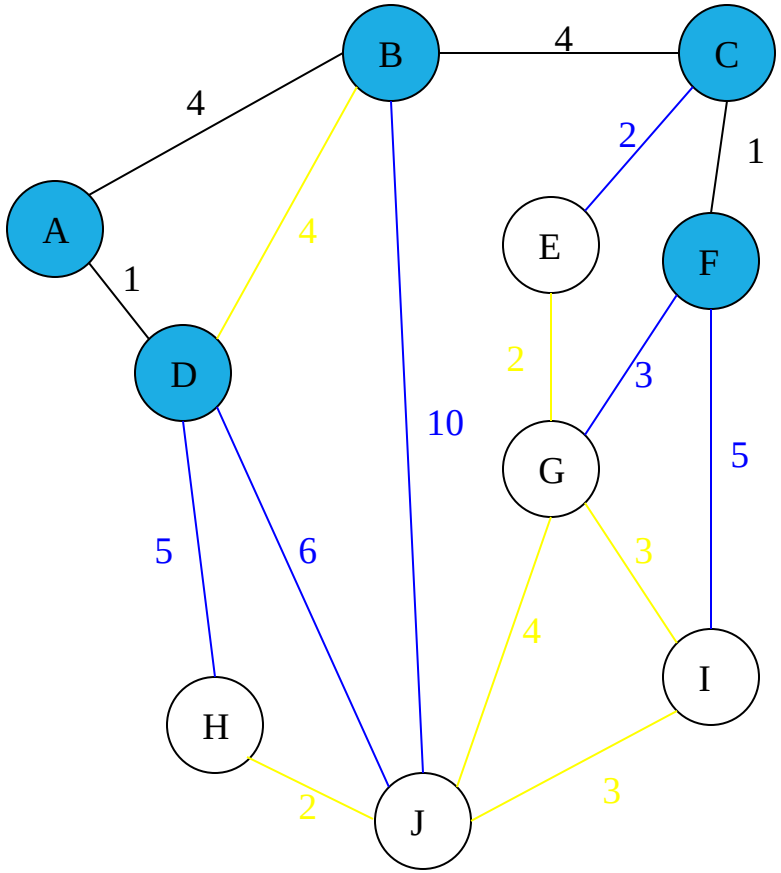
New Graph



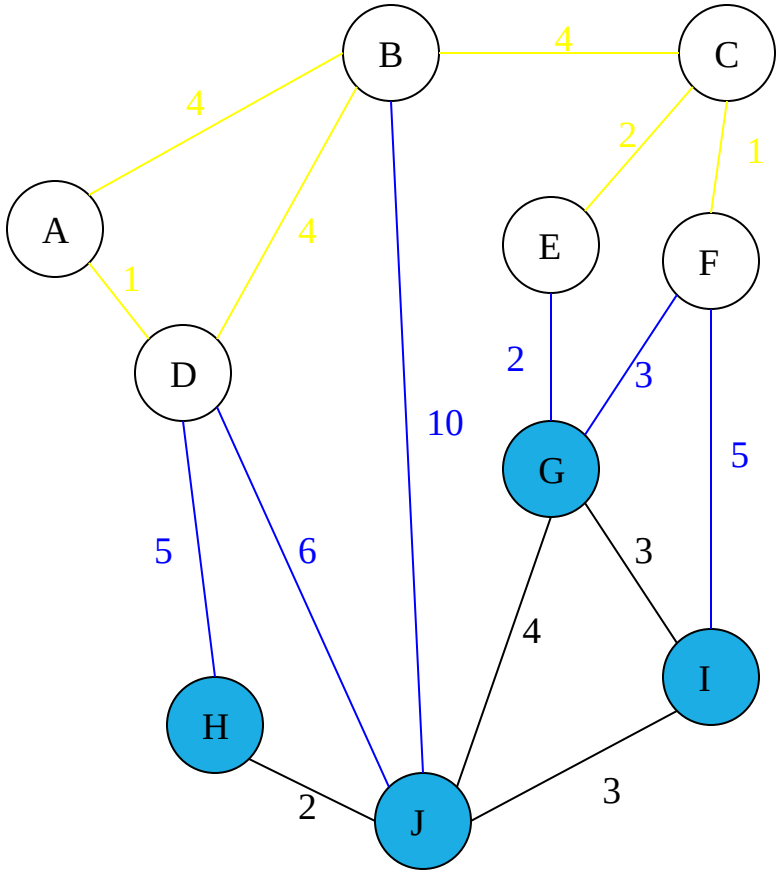
Old Graph



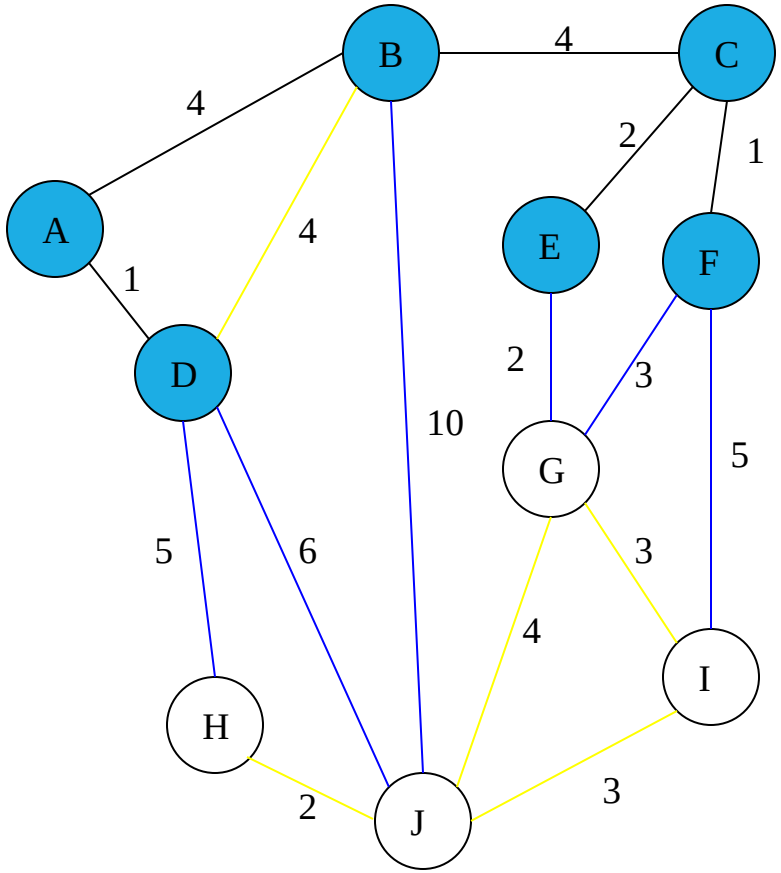
New Graph



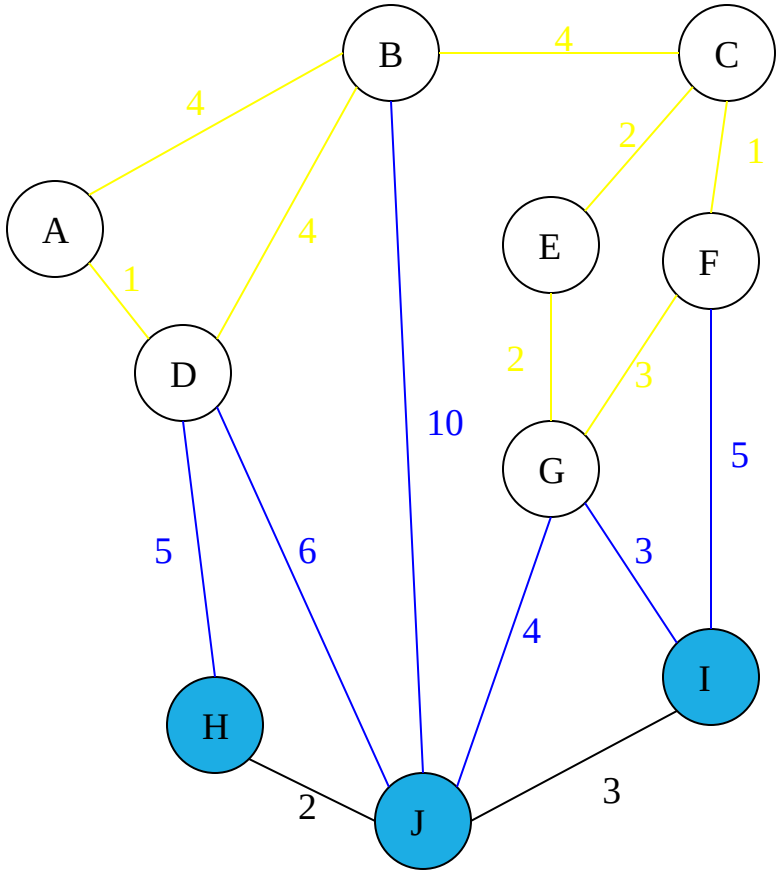
Old Graph



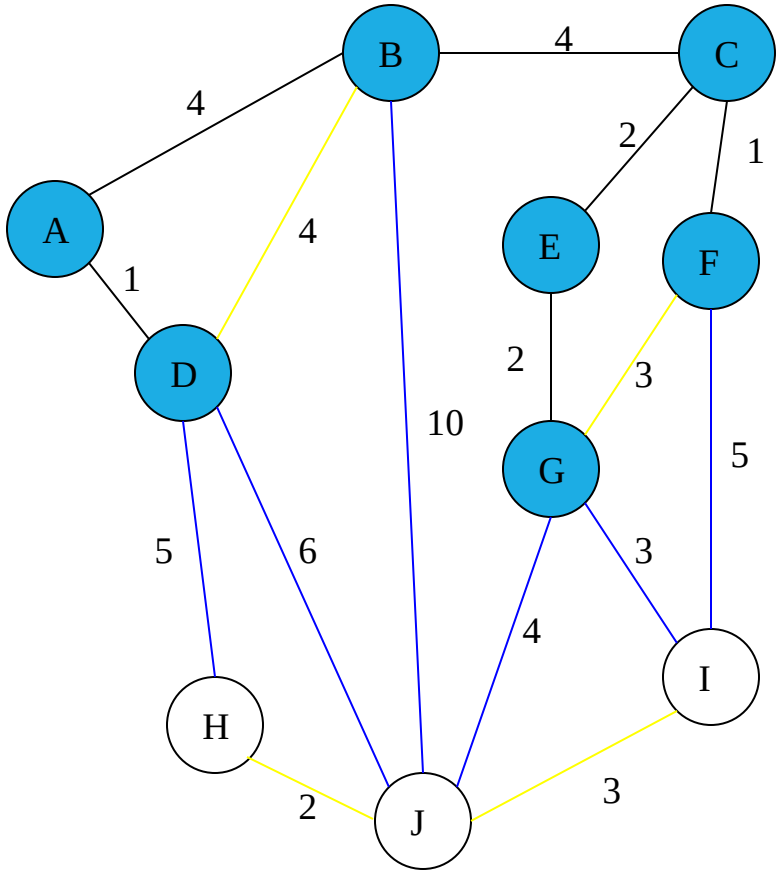
New Graph



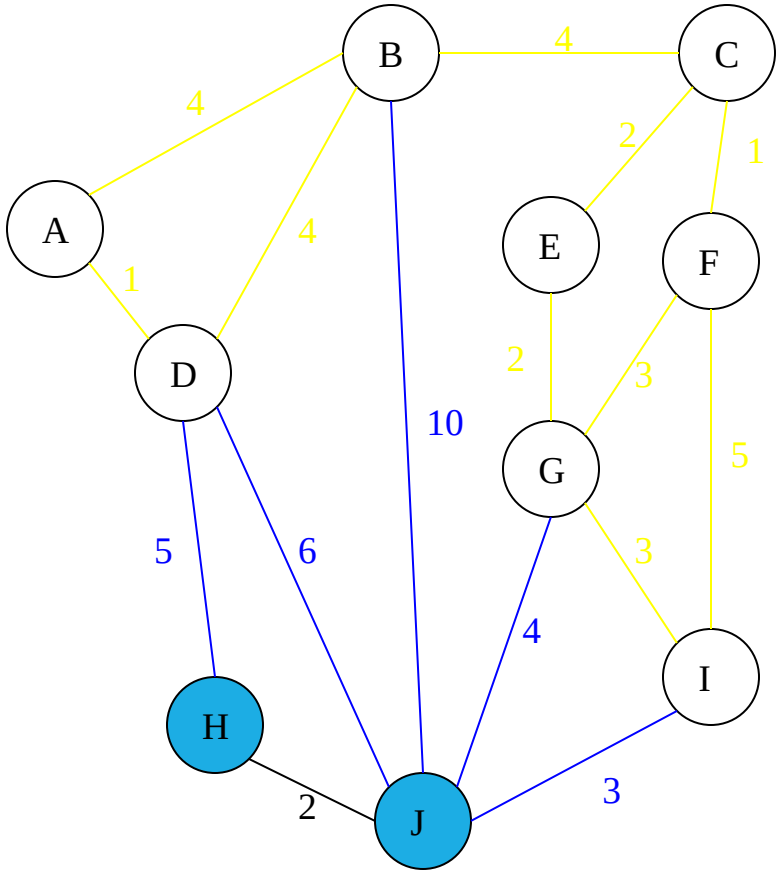
Old Graph



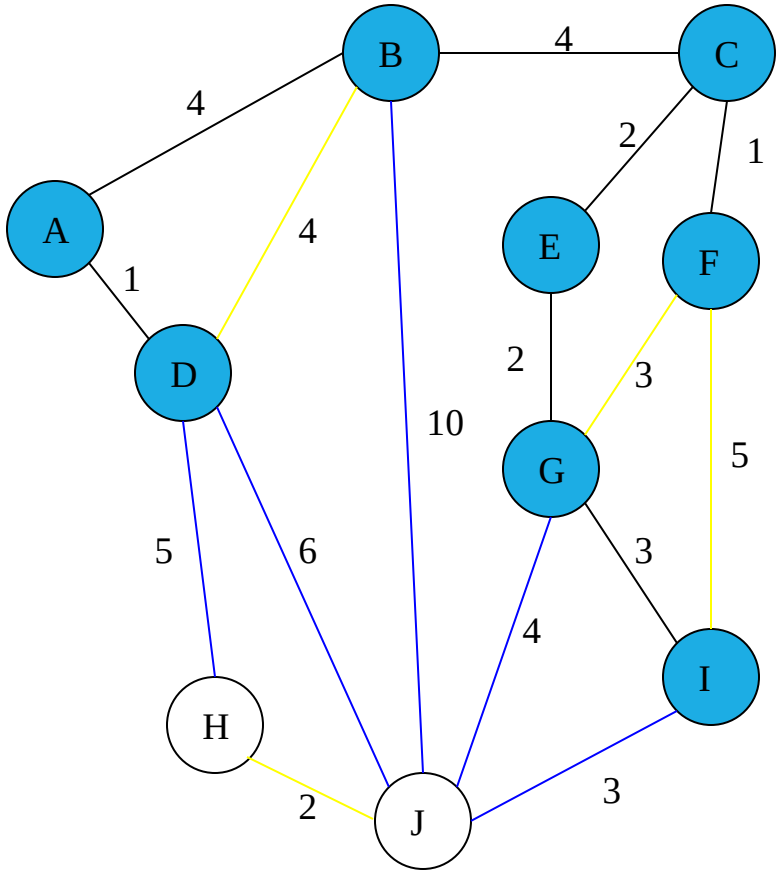
New Graph



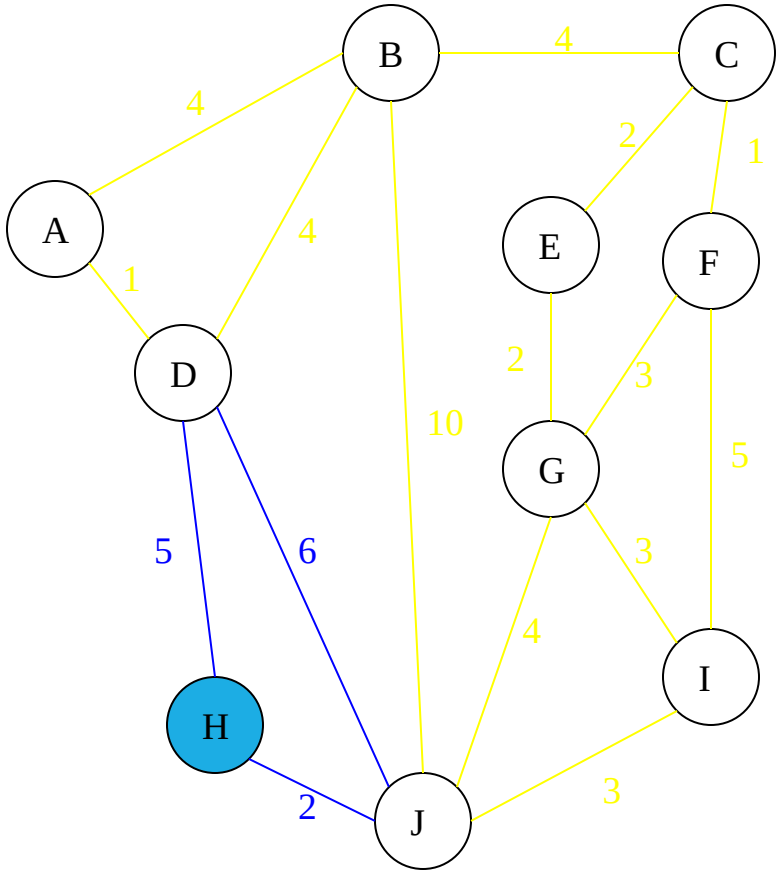
Old Graph



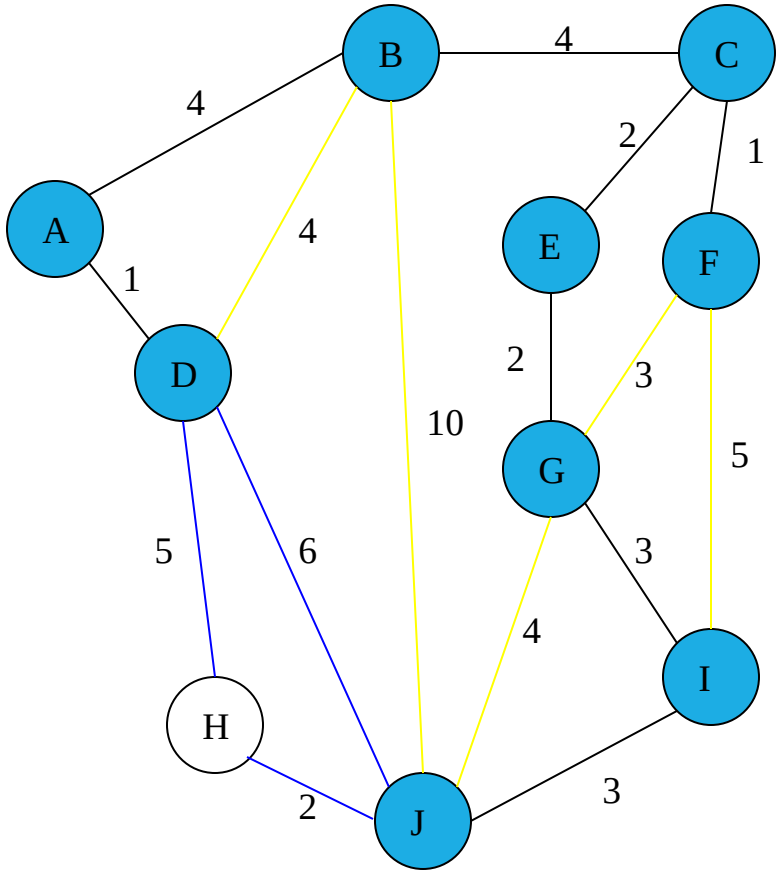
New Graph



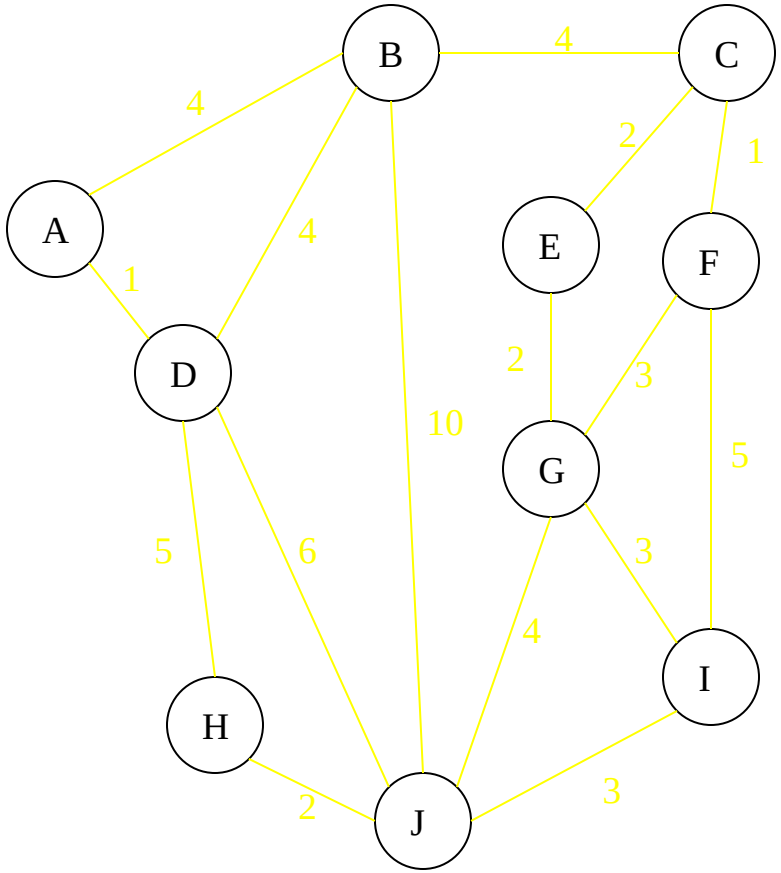
Old Graph



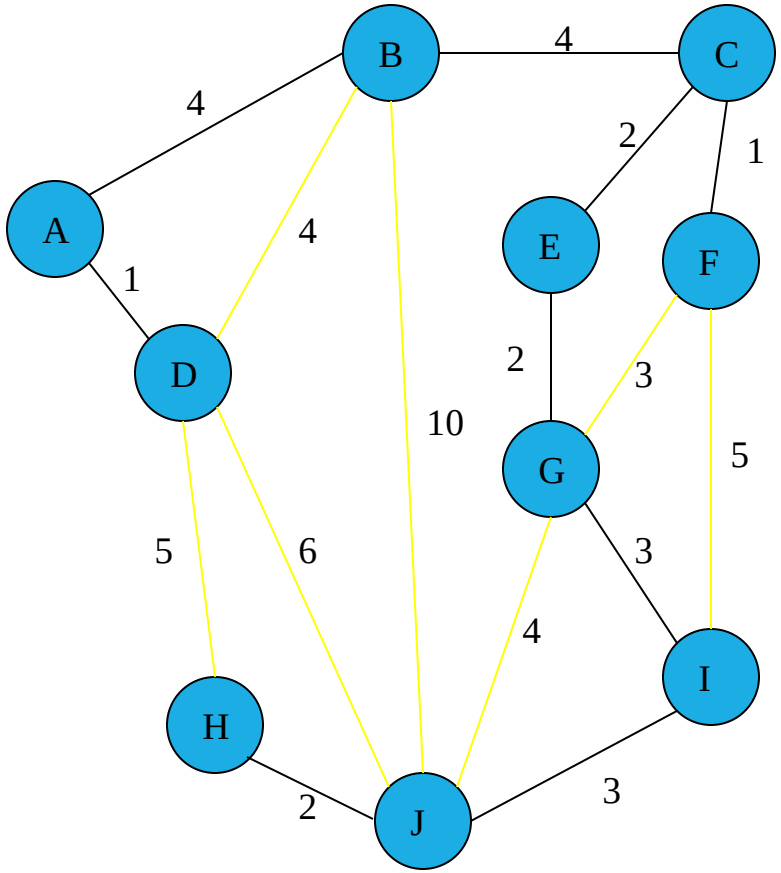
New Graph



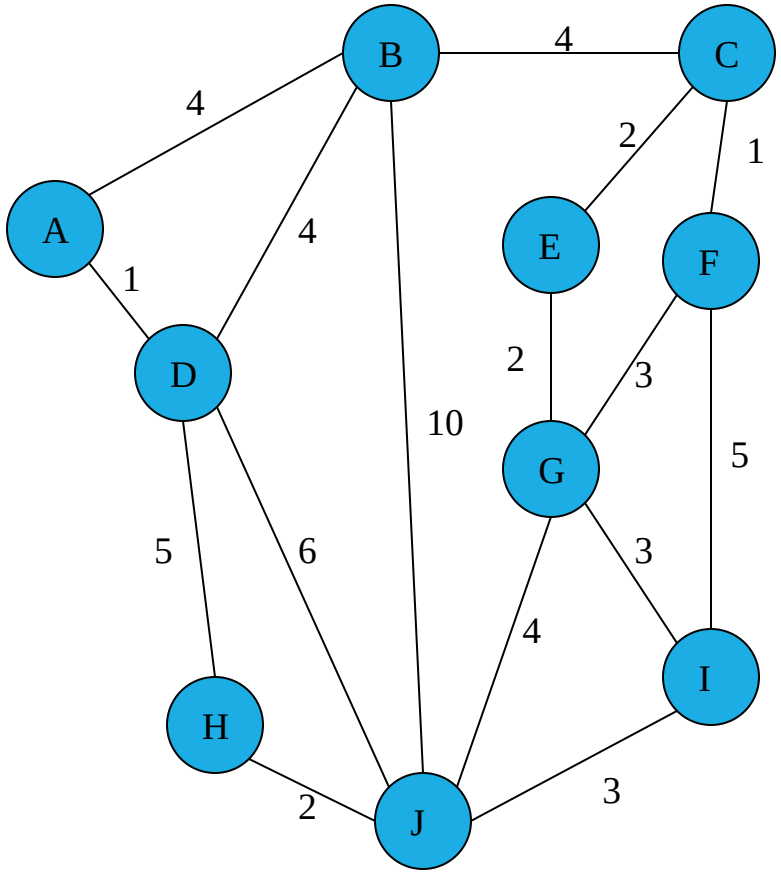
Old Graph



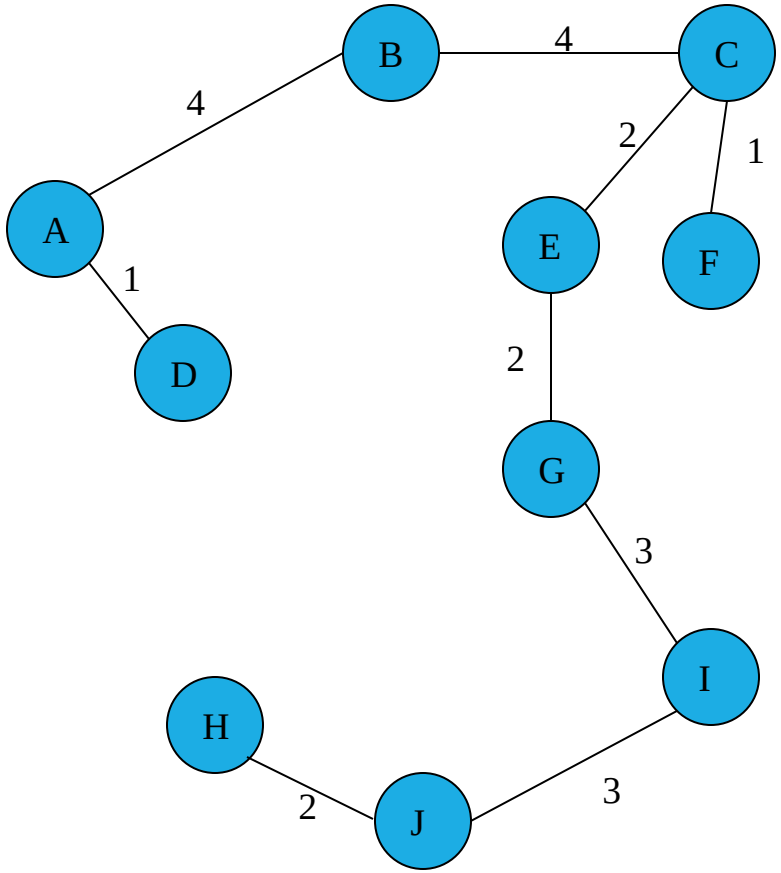
New Graph



Complete Graph



Minimum Spanning Tree

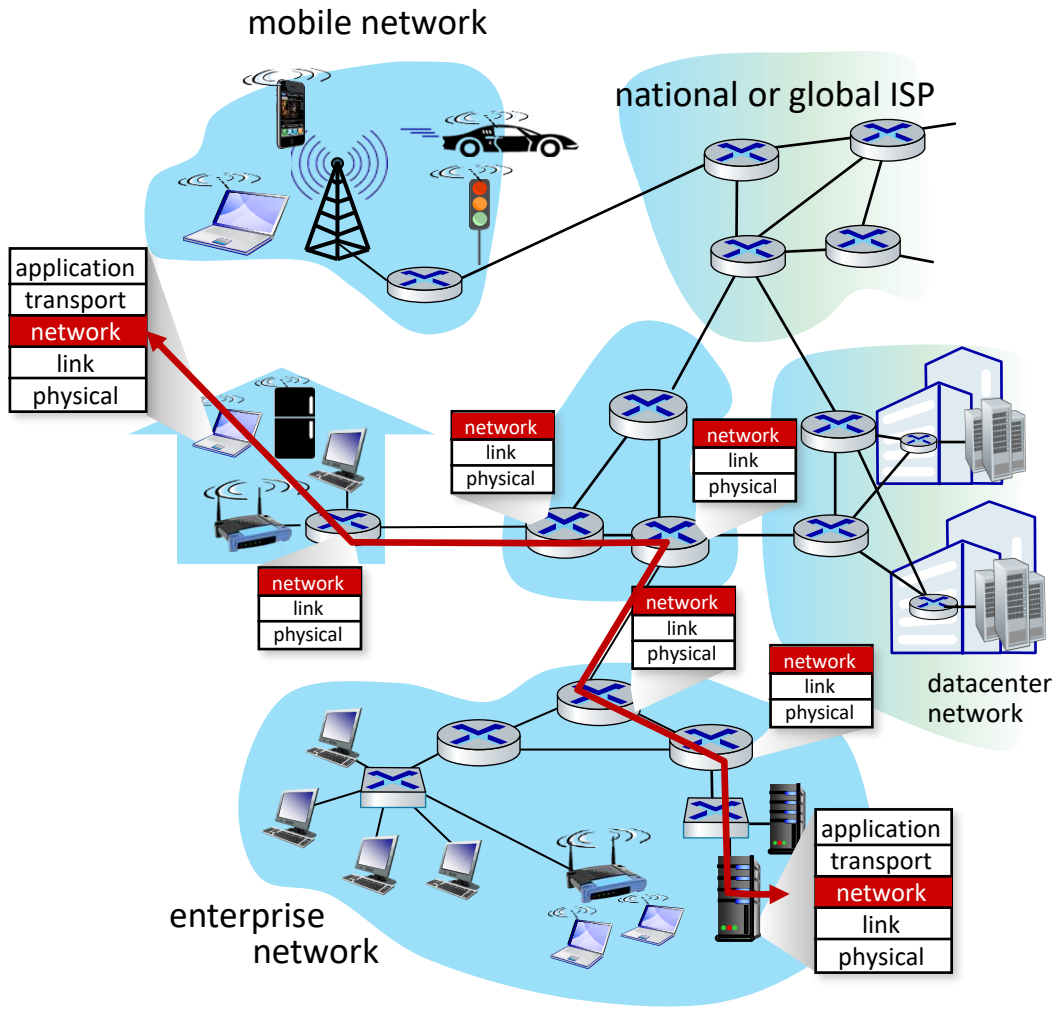



I KNOW
ROUTING
memegenerator.net

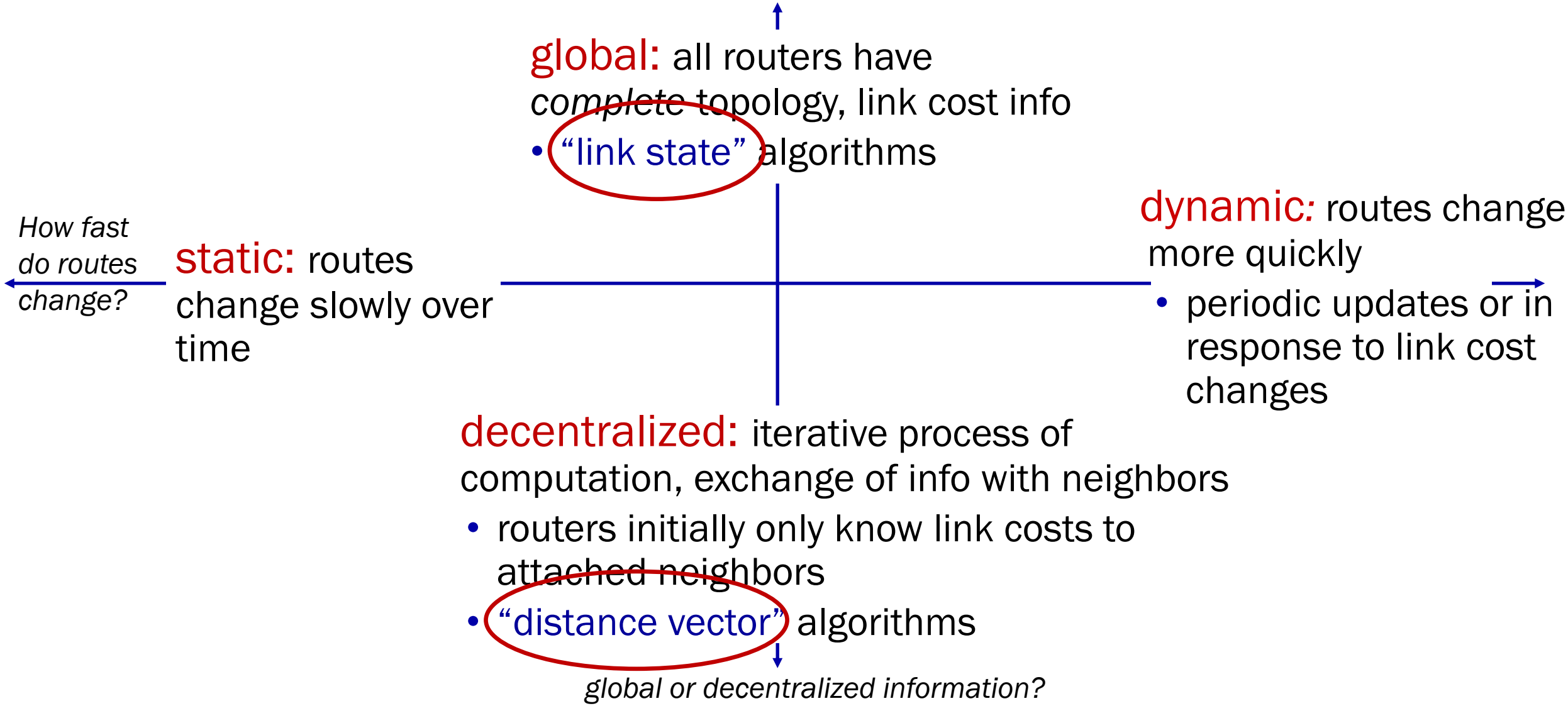
ROUTING PROTOCOLS

Routing protocol goal: determine "good" paths (equivalently, routes), from sending hosts to receiving host, through network of routers

- path: sequence of routers packets traverse from given initial source host to final destination host
- "good": least "cost", "fastest", "least congested"
- routing: a "top-10" networking challenge!



ROUTING ALGORITHM CLASSIFICATION



NETWORK LAYER: “CONTROL PLANE” ROADMAP

- routing protocols
  - link state
  - distance vector

DIJKSTRA'S LINK-STATE ROUTING ALGORITHM

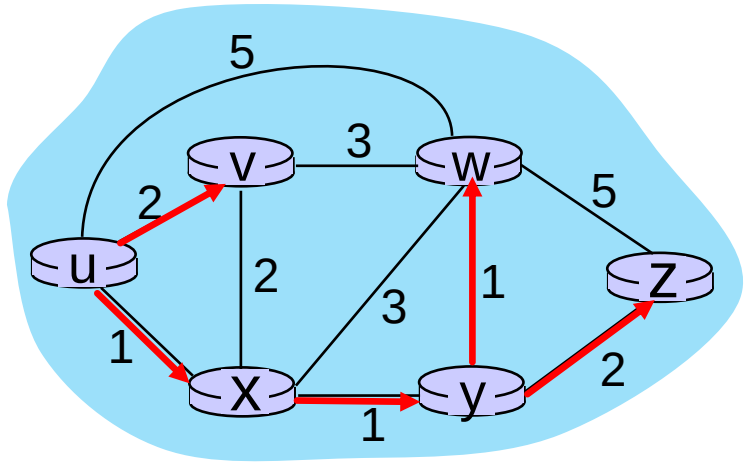
- centralized: network topology, link costs known to *all* nodes
  - accomplished via "link state broadcast"
  - all nodes have same info
- computes least cost paths from one node ("source") to all other nodes
  - gives *forwarding table* for that node
- iterative: after *k* iterations, know least cost path to *k* destinations

notation

- $c_{x,y}$: direct link cost from node *x* to *y*; = ∞ if not direct neighbors
- *D(v)*: *current* estimate of cost of least-cost-path from source to destination *v*
- *p(v)*: predecessor node along path from source to *v*
- *N'*: set of nodes whose least-cost-path *definitively* known

DIJKSTRA'S ALGORITHM: AN EXAMPLE

Step	N'	D(v),p(v)	D(w),p(w)	D(x),p(x)	D(y),p(y)	D(z),p(z)
0	u	2,u	5,u	1,u	∞	∞
1	ux	2,u	4,x		2,x	∞
2	uxy	2,u	3,y			4,y
3	uxyv		3,y			4,y
4	uxyvw					4,y
5	uxyvwz					

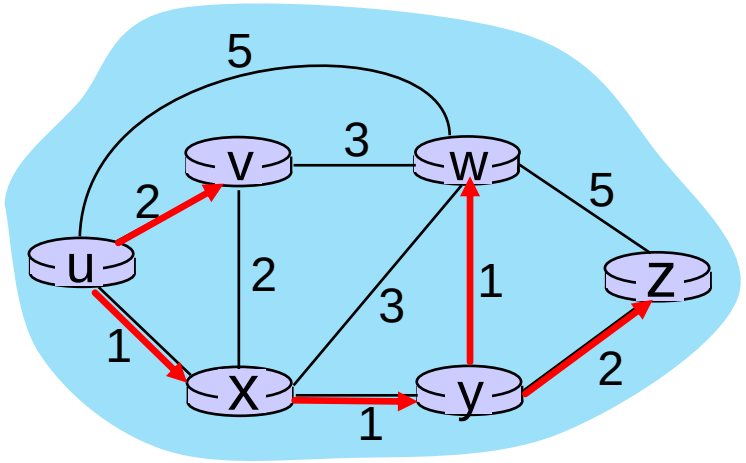


Initialization (step 0): For all a: if *a* adjacent to then $D(a) = c_{u,a}$

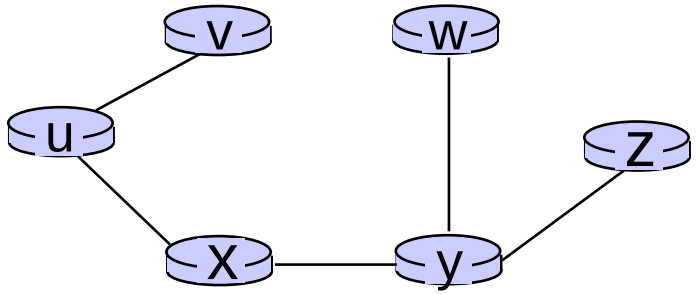
find *a* not in *N'* such that *D(a)* is a minimum
add *a* to *N'*
update *D(b)* for all *b* adjacent to *a* and not in *N'* :

$$D(b) = \min (D(b), D(a) + c_{a,b})$$

DIJKSTRA'S ALGORITHM: AN EXAMPLE



resulting least-cost-path tree from u:



resulting forwarding table in u:

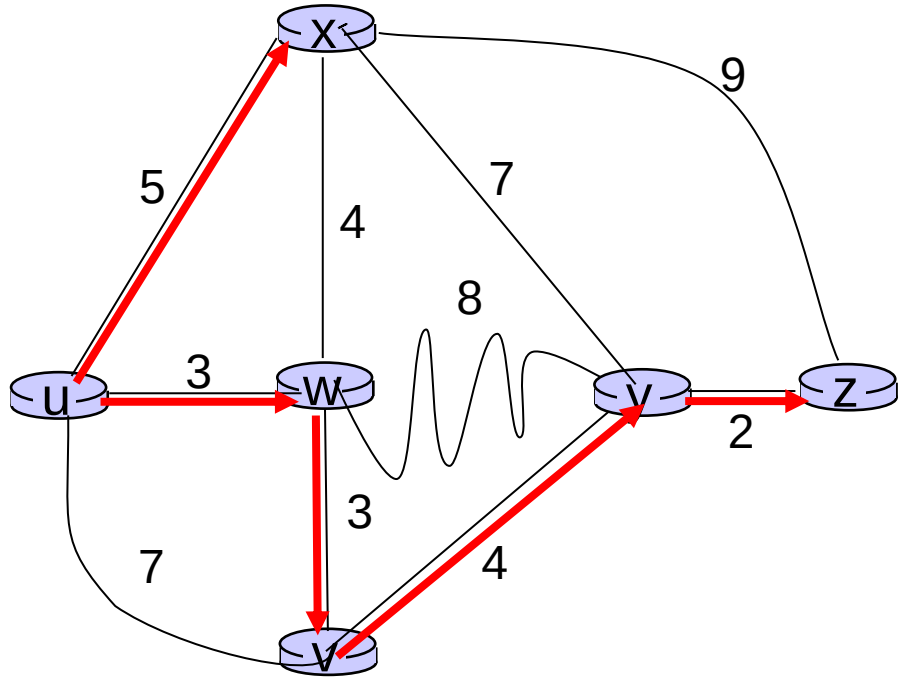
destination	outgoing link
v	(u,v)
x	(u,x)
y	(u,x)
w	(u,x)
x	(u,x)

route from *u* to *v* directly

route from u to all other destinations via *x*

DIJKSTRA'S ALGORITHM: ANOTHER EXAMPLE

Step	N'	v D(v), p(v)	w D(w), p(w)	x D(x), p(x)	y D(y), p(y)	z D(z), p(z)
0	u	7,u	3,u	5,u	∞	∞
1	uw	6,w		5,u	11,w	∞
2	uwx	6,w			11,w	14,x
3	uwxv				10,v	14,x
4	uwxvy					12,y
5	uwxvyz					



notes:

- construct least-cost-path tree by tracing predecessor nodes
- ties can exist (can be broken arbitrarily)

NETWORK LAYER: "CONTROL PLANE" ROADMAP

- routing protocols
  - link state
  - distance vector

DISTANCE VECTOR ALGORITHM

Based on *Bellman-Ford* (BF) equation (dynamic programming):

Bellman-Ford equation

Let $D_x(y)$: cost of least-cost path from x to y.
Then:

$$D_x(y) = \min_v \{ c_{x,v} + D_v(y) \}$$

v's estimated least-cost-path cost to y

min taken over all neighbors v of x

direct cost of link from x to v

Optimum 1-hop paths

Table for A		
Dst	Cst	Hop
A	0	A
B	4	B
C	∞	–
D	∞	–
E	2	E
F	6	F

Table for B		
Dst	Cst	Hop
A	4	A
B	0	B
C	∞	–
D	3	D
E	∞	–
F	1	F

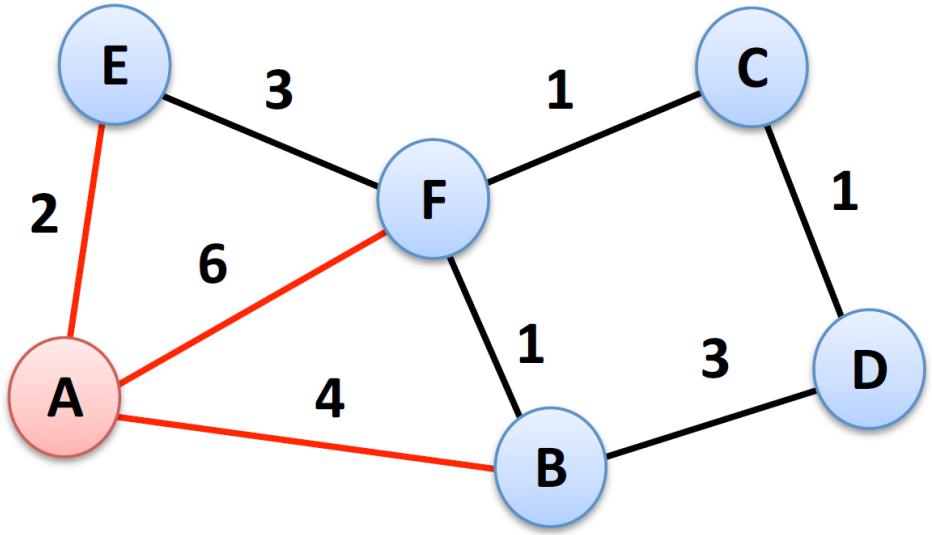


Table for C		
Dst	Cst	Hop
A	∞	–
B	∞	–
C	0	C
D	1	D
E	∞	–
F	1	F

Table for D		
Dst	Cst	Hop
A	∞	–
B	3	B
C	1	C
D	0	D
E	∞	–
F	∞	–

Table for E		
Dst	Cst	Hop
A	2	A
B	∞	–
C	∞	–
D	∞	–
E	0	E
F	3	F

Table for F		
Dst	Cst	Hop
A	6	A
B	1	B
C	1	C
D	∞	–
E	3	E
F	0	F

Optimum 2-hop paths

Table for A		
Dst	Cst	Hop
A	0	A
B	4	B
C	7	F
D	7	B
E	2	E
F	5	E

Table for B		
Dst	Cst	Hop
A	4	A
B	0	B
C	2	F
D	3	D
E	4	F
F	1	F

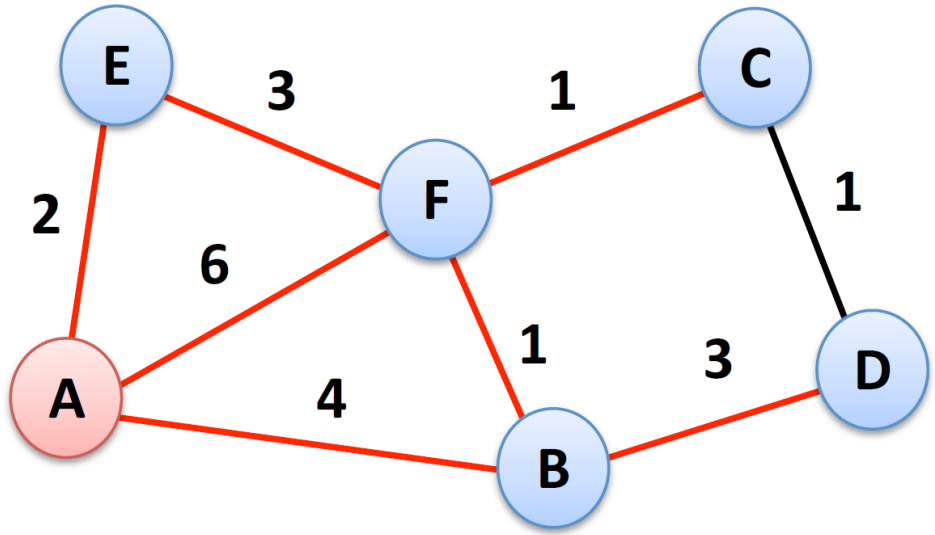


Table for C		
Dst	Cst	Hop
A	7	F
B	2	F
C	0	C
D	1	D
E	4	F
F	1	F

Table for D		
Dst	Cst	Hop
A	7	B
B	3	B
C	1	C
D	0	D
E	∞	–
F	2	C

Table for E		
Dst	Cst	Hop
A	2	A
B	4	F
C	4	F
D	∞	–
E	0	E
F	3	F

Table for F		
Dst	Cst	Hop
A	5	B
B	1	B
C	1	C
D	2	C
E	3	E
F	0	F

Optimum 3-hop paths

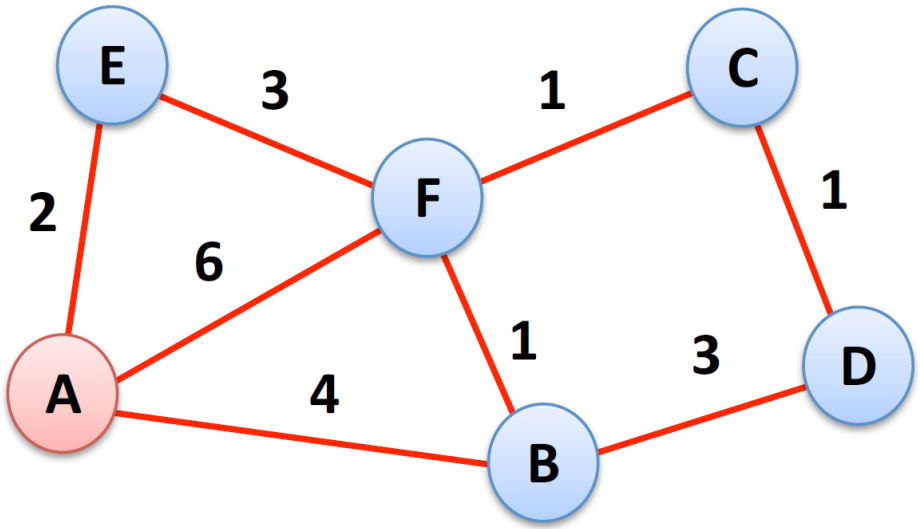


Table for A			Table for B		
Dst	Cst	Hop	Dst	Cst	Hop
A	0	A	A	4	A
B	4	B	B	0	B
C	6	E	C	2	F
D	7	B	D	3	D
E	2	E	E	4	F
F	5	E	F	1	F

Table for C			Table for D			Table for E			Table for F		
Dst	Cst	Hop	Dst	Cst	Hop	Dst	Cst	Hop	Dst	Cst	Hop
A	6	F	A	7	B	A	2	A	A	5	B
B	2	F	B	3	B	B	4	F	B	1	B
C	0	C	C	1	C	C	4	F	C	1	C
D	1	D	D	0	D	D	5	F	D	2	C
E	4	F	E	5	C	E	0	E	E	3	E
F	1	F	F	2	C	F	3	F	F	0	F


ANY QUESTIONS???

REFERENCES

- https://www.net.t-labs.tu-berlin.de/teaching/computer_networking
- https://techterms.in/
- https://github.com/HanochShi/Supplements-ComputerNetworking-ATopDownApproach-7th-ed
- https://www.youtube.com/channel/UCJQJ4GjTiq5lmn8czf8ooOQ
- http://www.whatis.com
- http://www.webopedia.com
- Understanding Data Communications & Networks, Shay (1999)
- http://www.daemon.org/ip.html

READING INSTRUCTIONS

» CH. 17: ALL » CH. 18: ALL