OPERATIONS RESEARCH

O R

NETWORK MODELING/ANALYSIS:

APPLICATIONS OF NETWORK OPTIMIZATION

APPLICATIONS	PHYSICAL ANALOG OF NODES	PHYSICAL ANALOG OF ARCS	FLOW
Communication systems	phone exchanges, computers, transmission facilities, satellites	Cables, fiber optic links, microwave relay links	Voice messages, Data, Video transmissions
Hydraulic systems	Pumping stations Reservoirs, Lakes	Pipelines	Water, Gas, Oil, Hydraulic fluids
Integrated computer circuits	Gates, registers, processors	Wires	Electrical current
Mechanical systems	Joints	Rods, Beams, Springs	Heat, Energy
Transportation systems	Intersections, Airports, Rail yards	Highways, Airline routes Railbeds	Passengers, freight, vehicles, operators

NETWORKS ARE EVERYWHERE

- Physical Networks
  - Road Networks
  - Railway Networks
  - Airline traffic Networks
  - Electrical networks, e.g., the power grid
- Abstract networks
  - organizational charts
  - precedence relationships in projects
- Others?

NETWORKS ANALYSIS: DESCRIPTION

Many important optimization problems can be analyzed by means of graphical or network representation. The following network models will be discussed:

1. Minimum spanning tree problems
2. Shortest path problems
3. Maximum flow problems

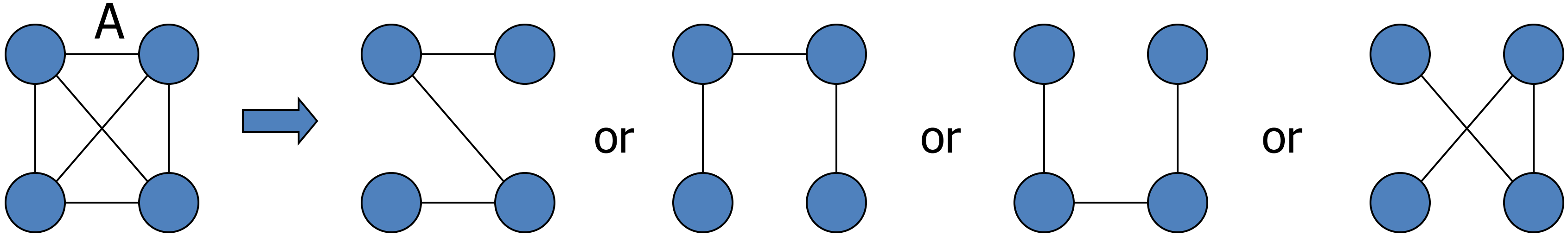
SPANNING TREES

A spanning tree of a graph is just a sub-graph that contains all the vertices and is a tree.

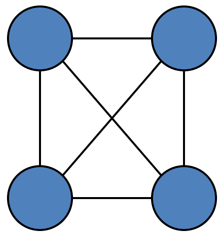
A graph may have many spanning trees.

Graph A

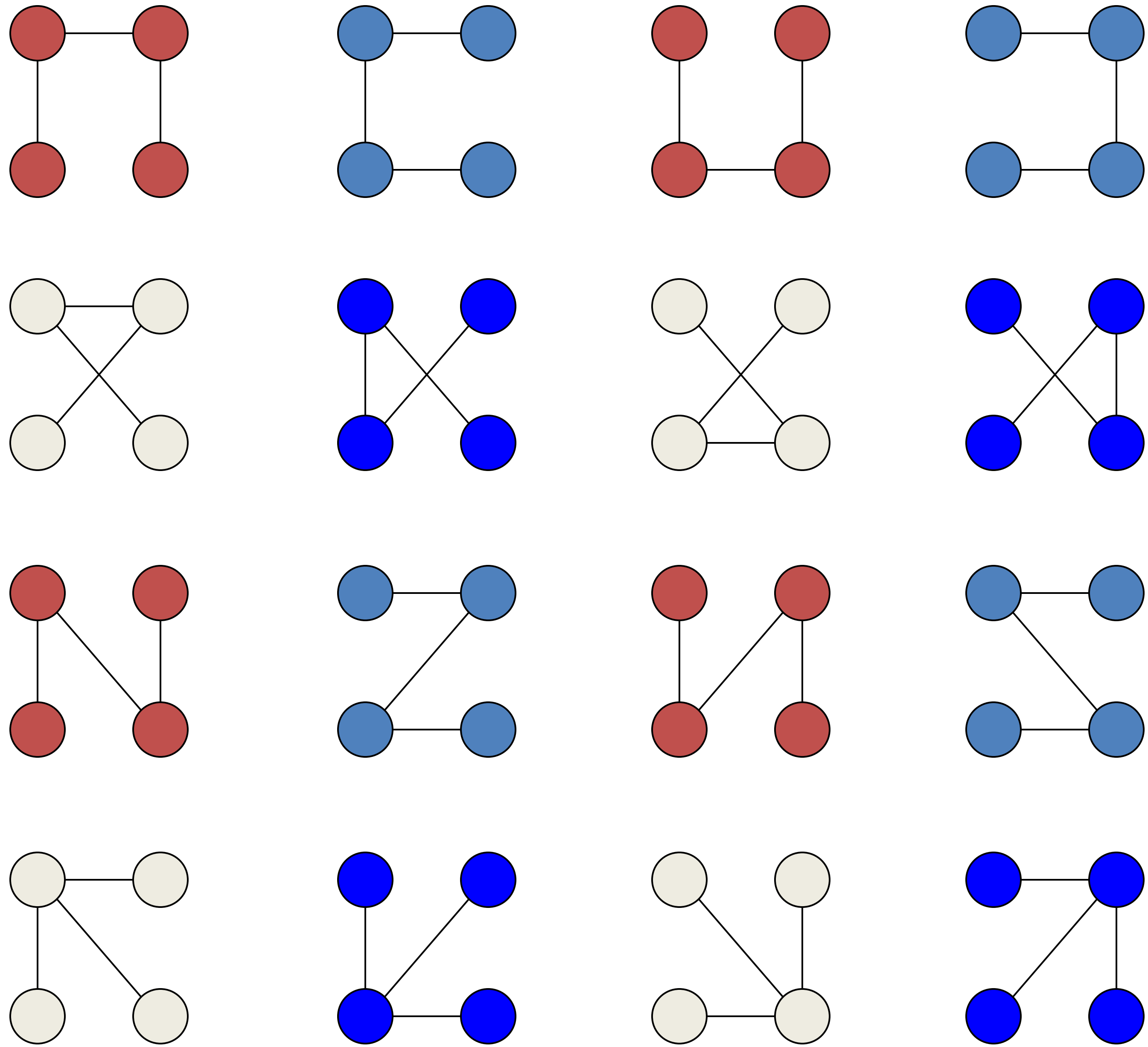
Some Spanning Trees from Graph A



Complete Graph



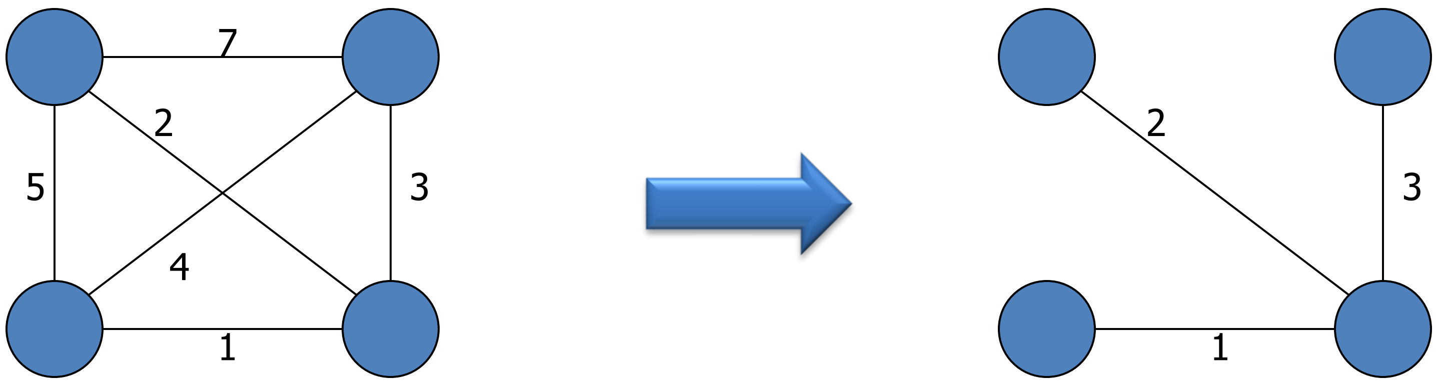
All 16 of its Spanning Trees



MINIMUM 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

- This algorithm creates a forest of trees. Initially the forest consists of ‘n’ single node trees (and no edges). At each step, we add one edge (the cheapest one) so that it joins two trees together.

- If it were to form a cycle, it would simply link two nodes that were already part of a single connected tree, so that this edge would not be needed.

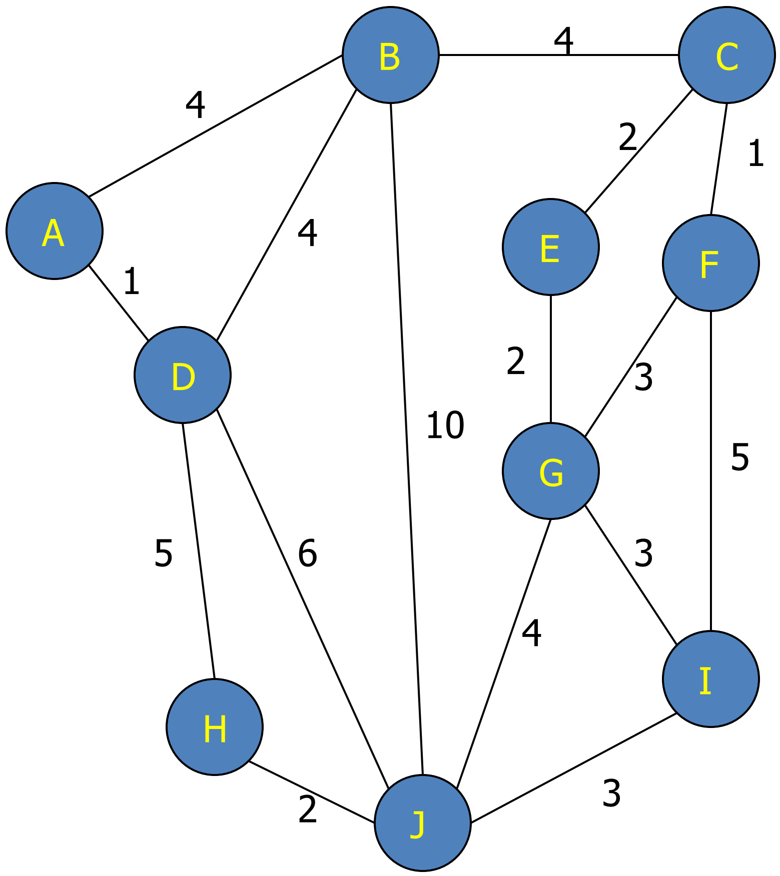
KRUSKAL'S ALGORITHM

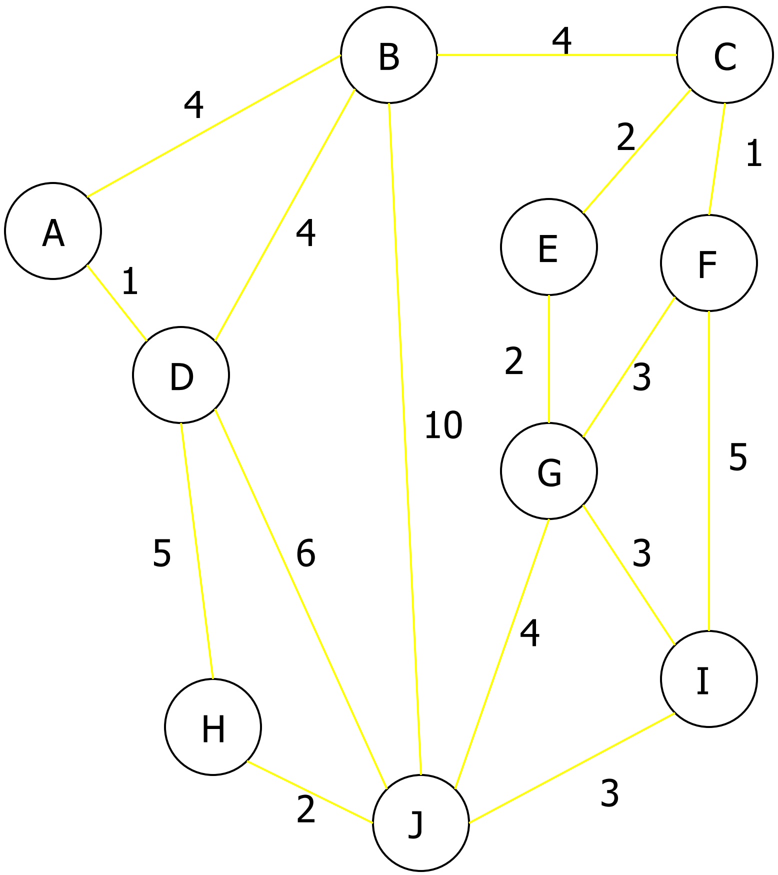
The steps are:

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,
    a. Extract the cheapest edge from the queue,
    b. If it forms a cycle, reject it,
    c. Else add it to the forest. Adding it to the forest will join two trees together.

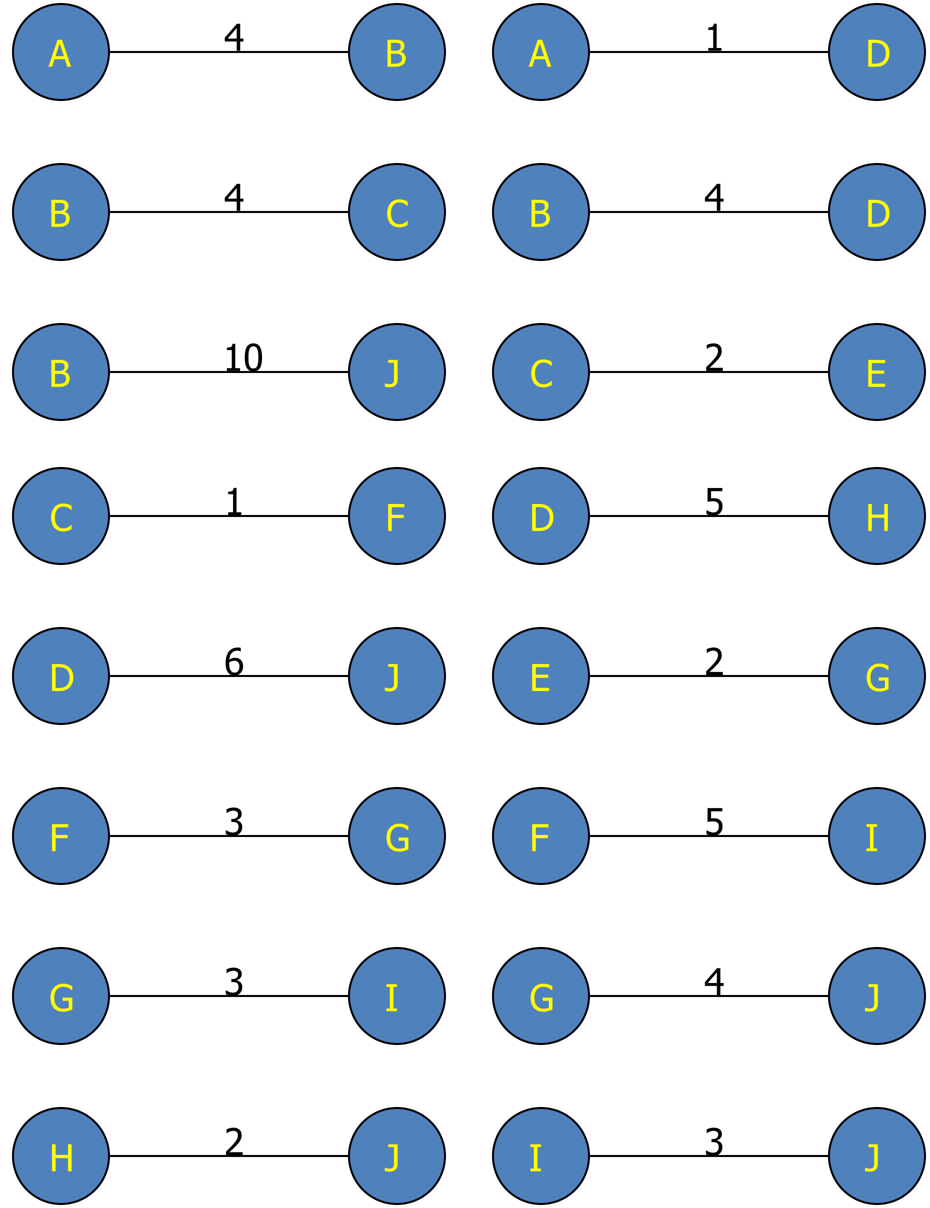
Every step will have joined two trees in the forest together, so that at the end, there will only be one tree in ‘T’.

COMPLETE GRAPH





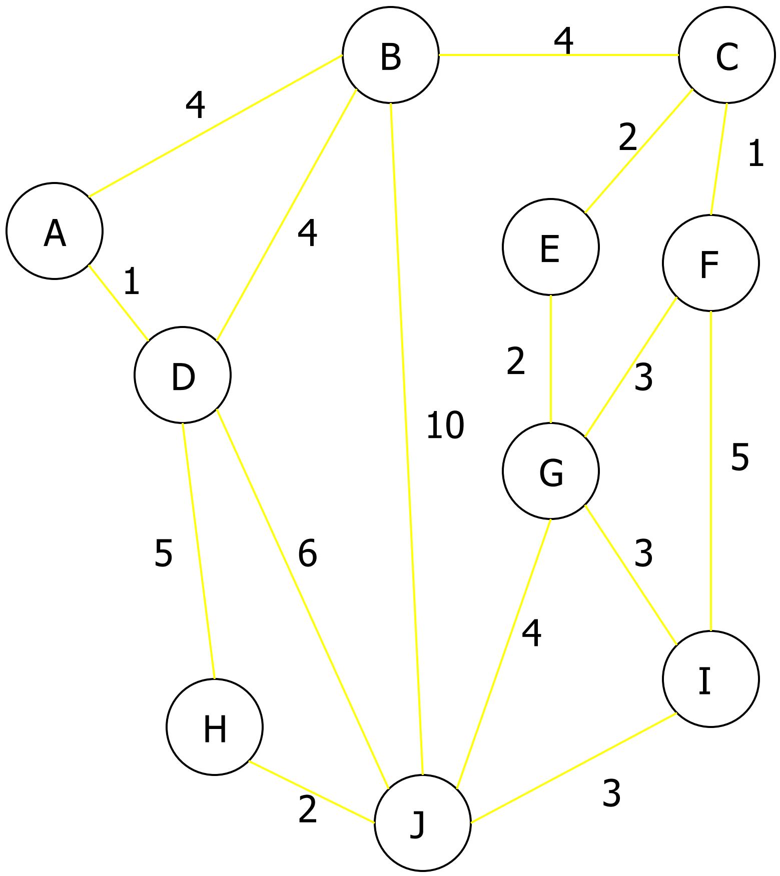
A
B
C
D
E
F
G
H
I
J
4
4
1
4
2
1
10
2
3
5
5
6
3
4
2
3

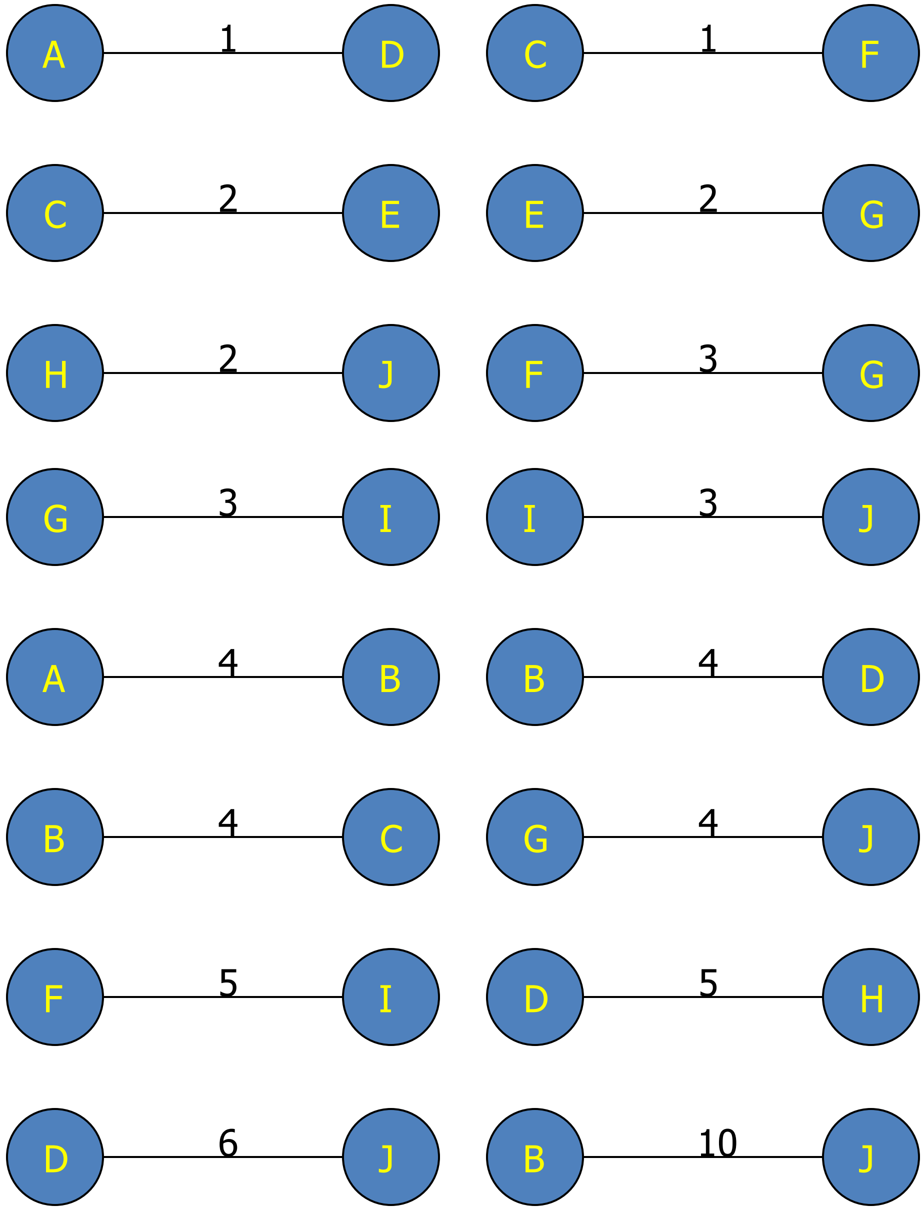


A 4 B
A 1 D
B 4 C
B 4 D
B 10 J
C 2 E
C 1 F
D 5 H
D 6 J
E 2 G
F 3 G
F 5 I
G 3 I
G 4 J
H 2 J
I 3 J

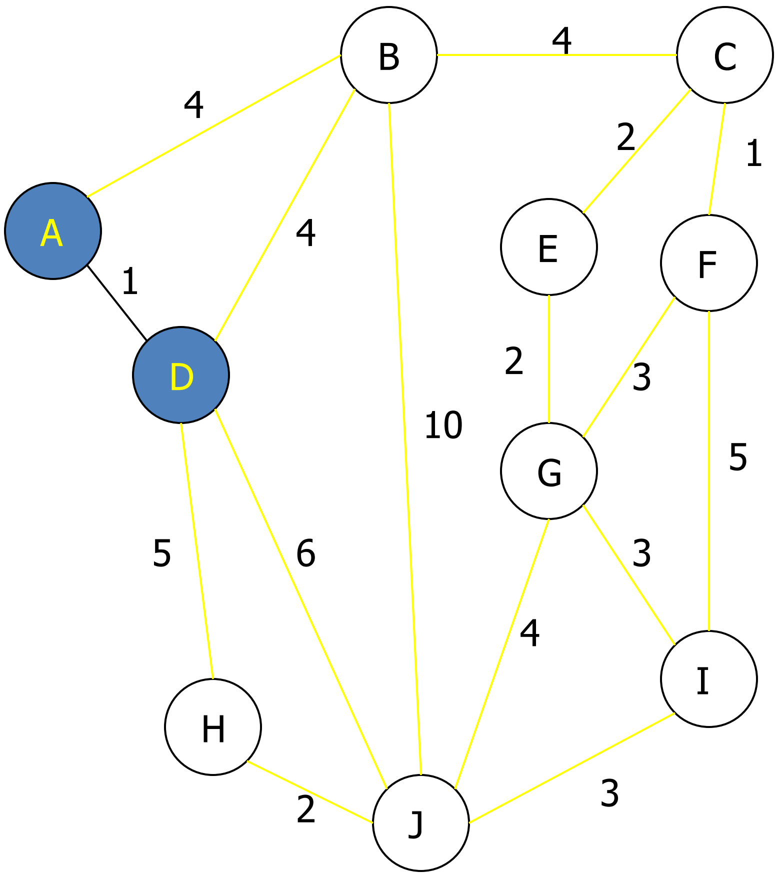
SORT EDGES

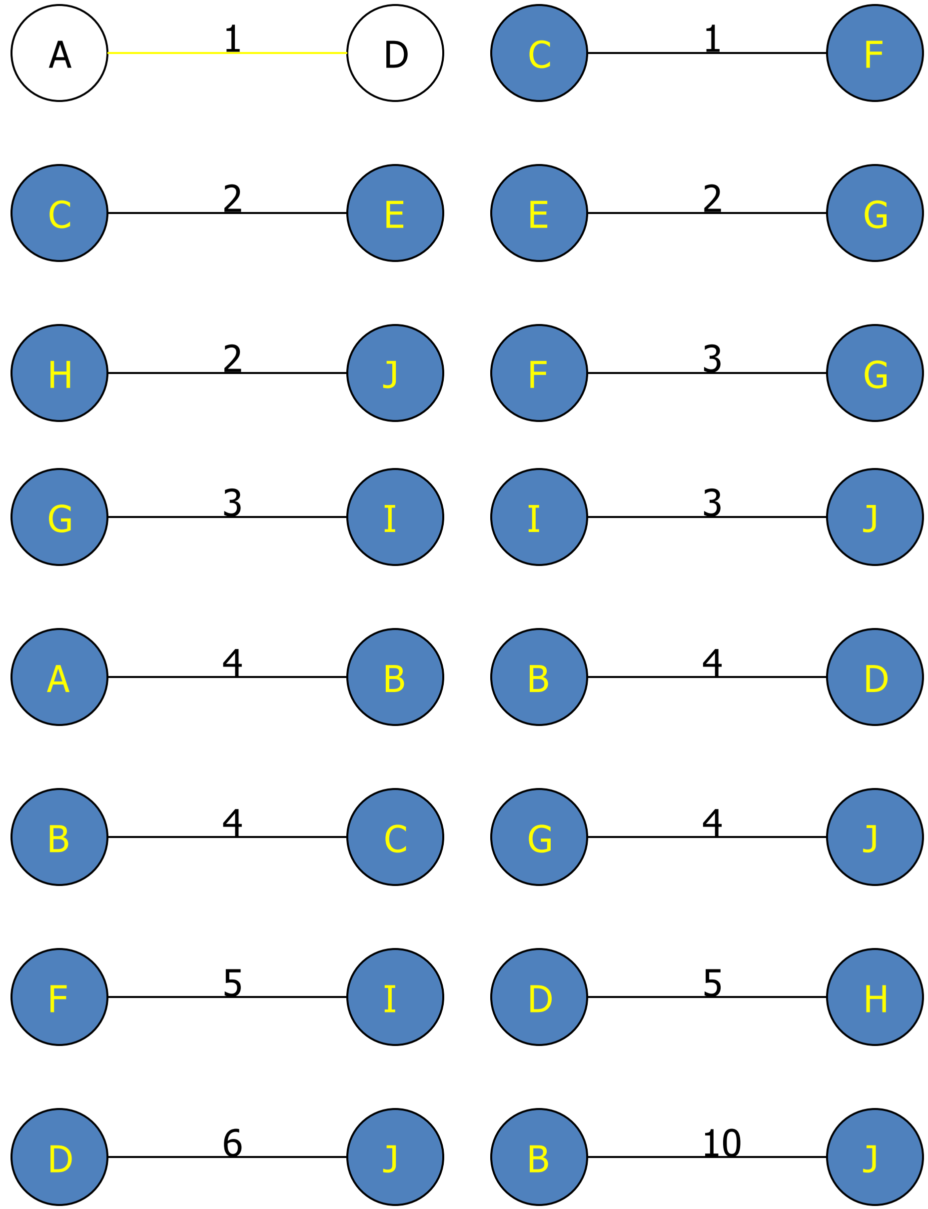
(in reality they are placed in a priority queue - not sorted - but sorting them makes the algorithm easier to visualize)



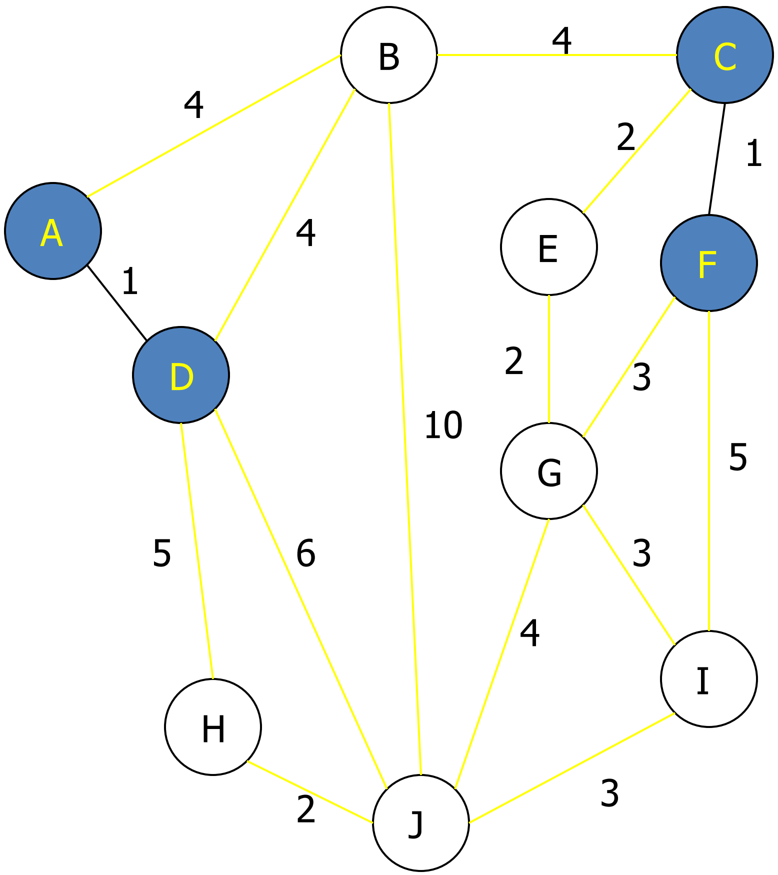


ADD EDGE

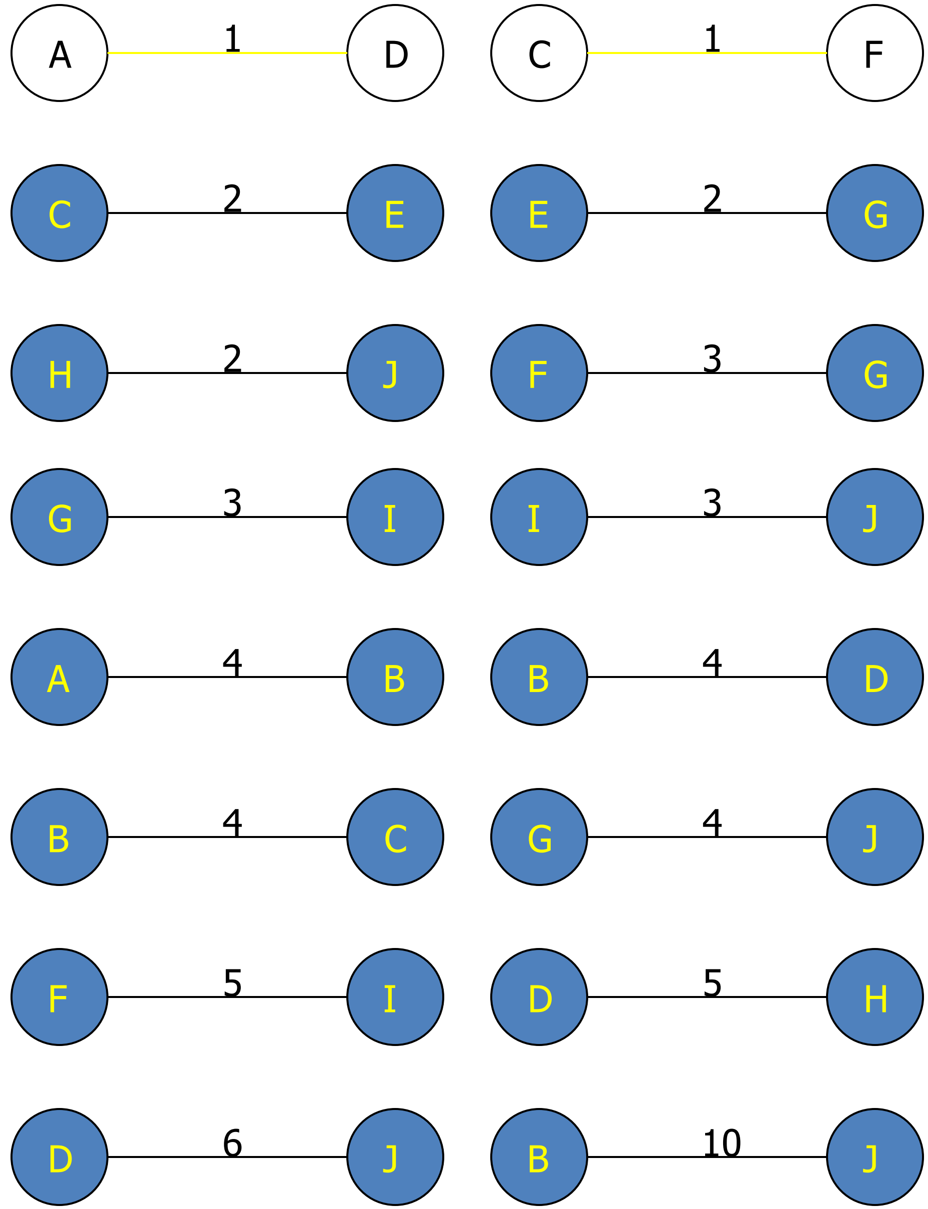

A
B
C
D
E
F
G
H
I
J
4
4
1
4
2
1
2
3
5
10
5
6
4
3
2
3


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

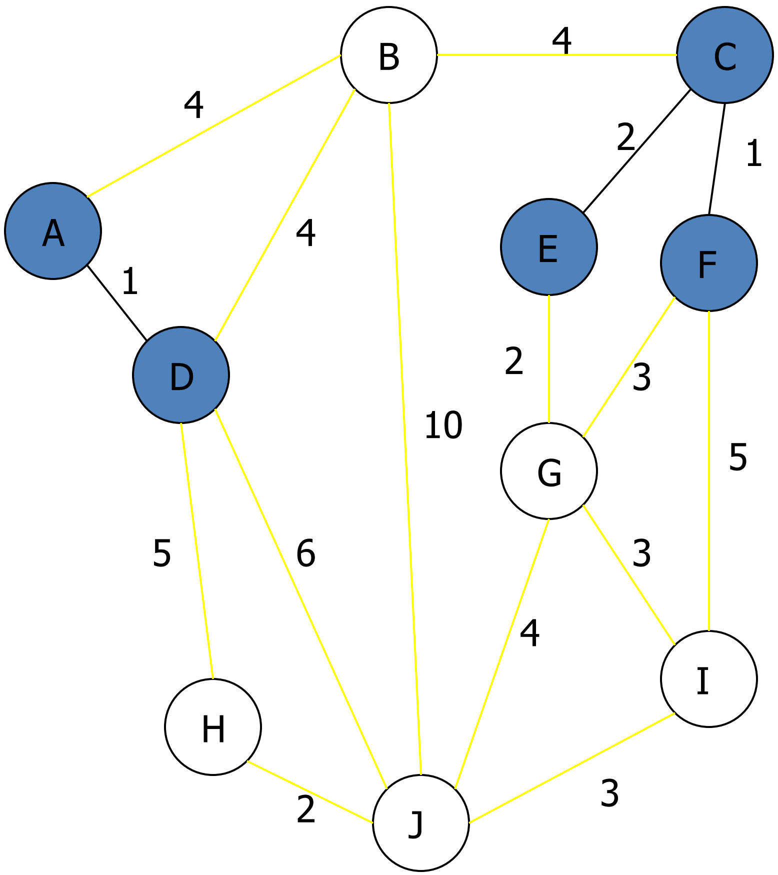


A
B
C
D
E
F
G
H
I
J
4
4
4
1
2
1
2
3
10
5
5
6
4
3
2
3

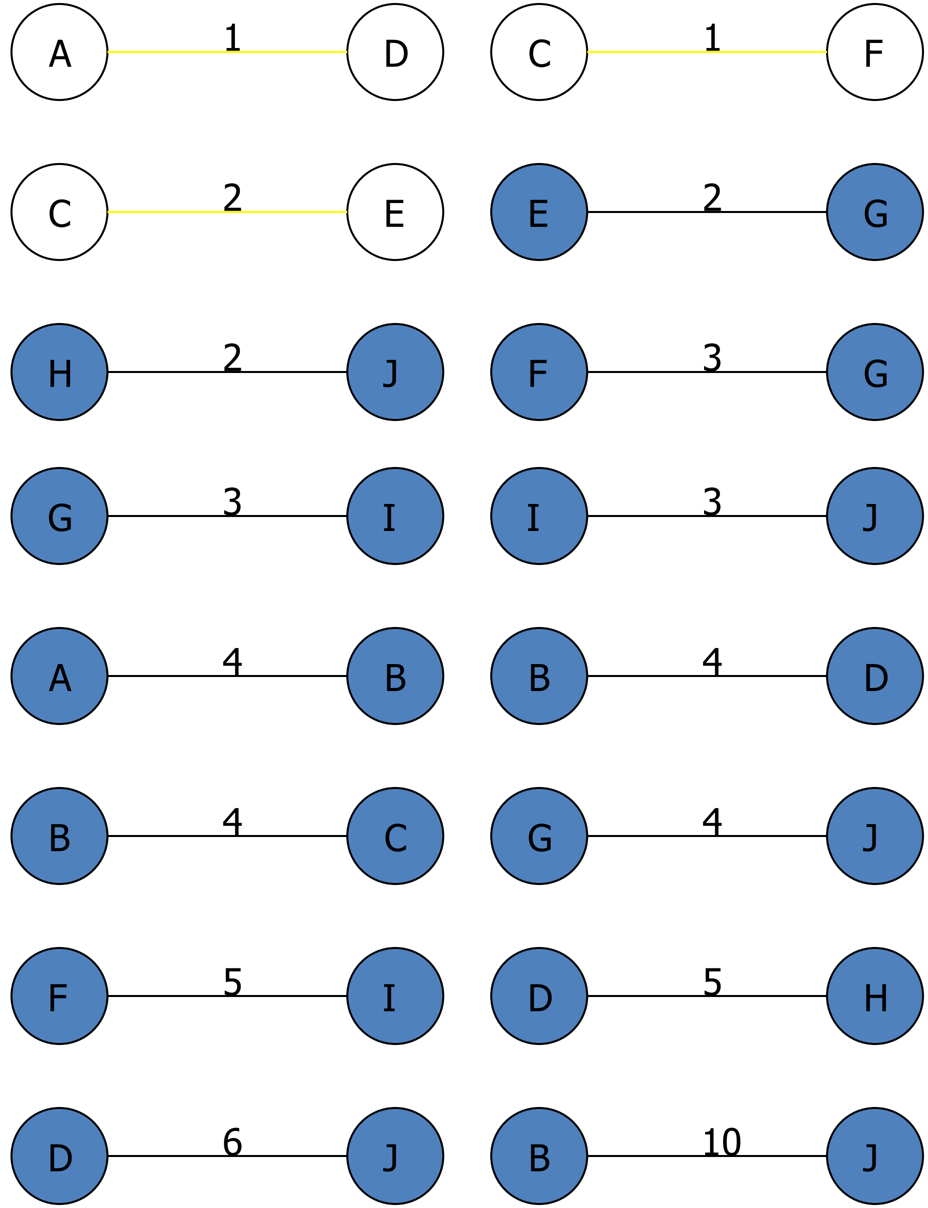


A 1 D
C 1 F
C 2 E
E 2 G
H 2 J
F 3 G
G 3 I
I 3 J
A 4 B
B 4 D
B 4 C
G 4 J
F 5 I
D 5 H
D 6 J
B 10 J

ADD EDGE

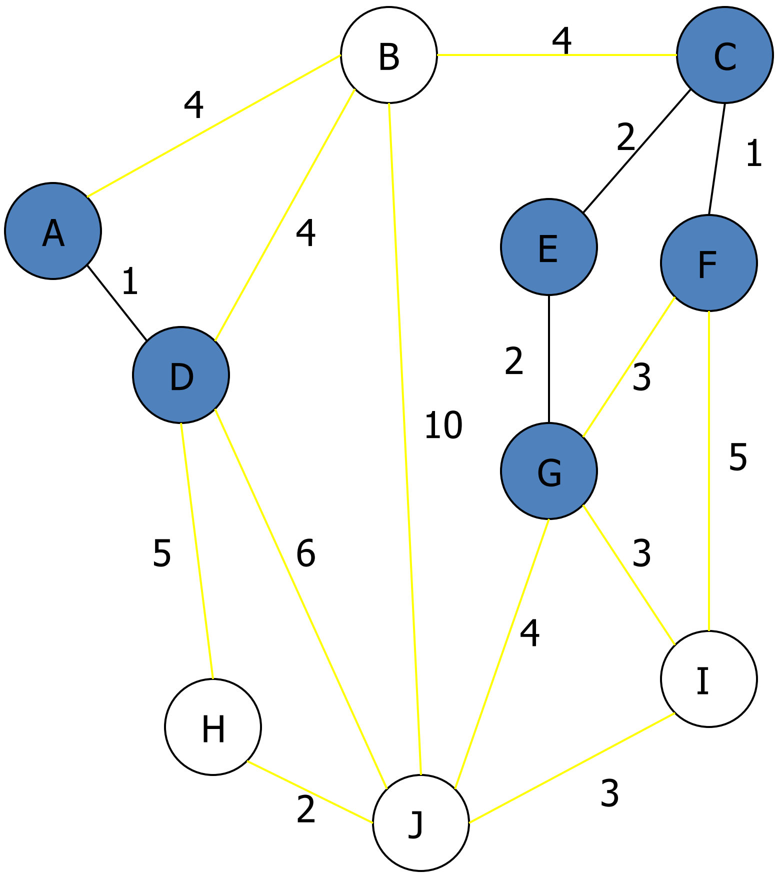


A
B
C
D
E
F
G
H
I
J
4
4
1
4
2
1
2
3
5
10
5
6
4
3
2
3

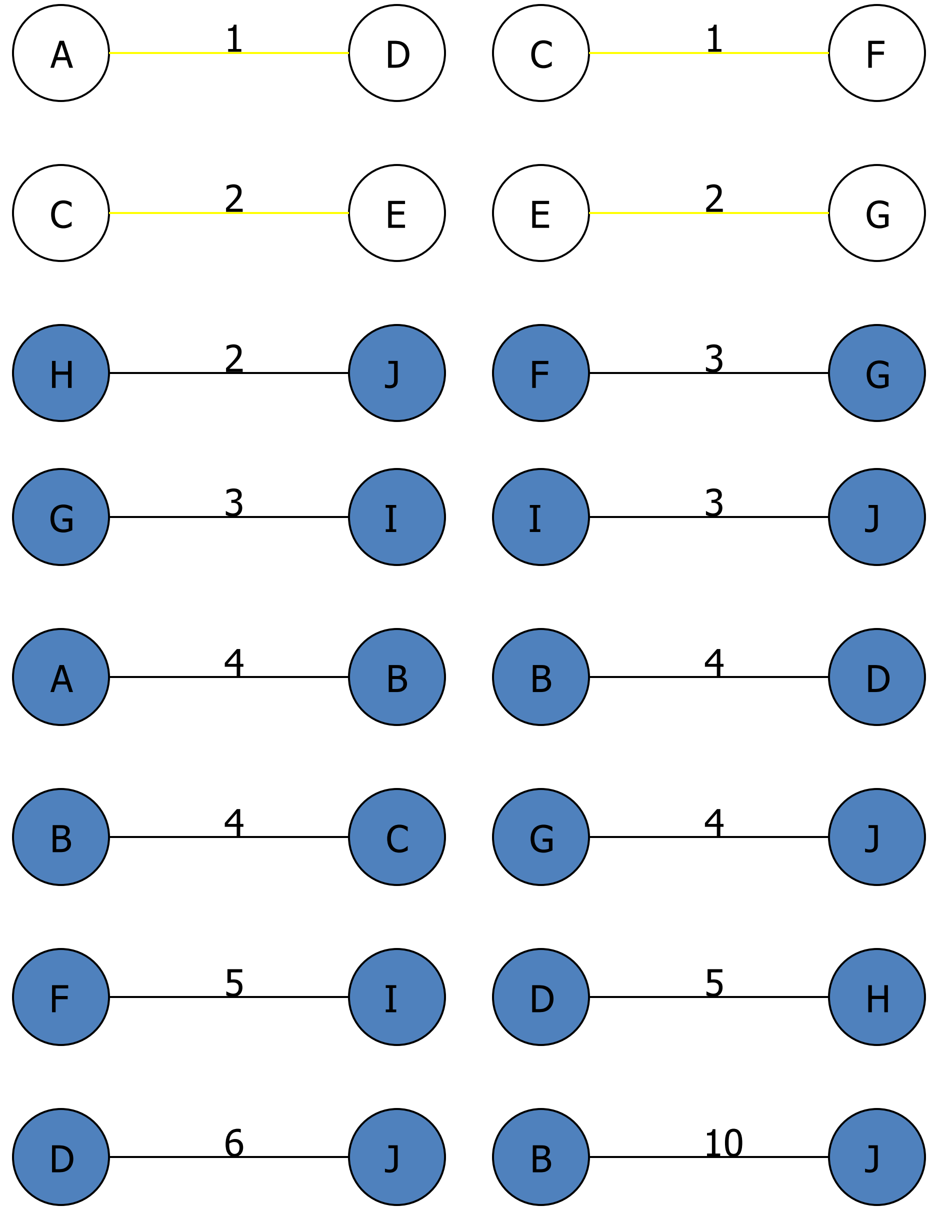


A 1 D
C 1 F
C 2 E
E 2 G
H 2 J
F 3 G
G 3 I
I 3 J
A 4 B
B 4 D
B 4 C
G 4 J
F 5 I
D 5 H
D 6 J
B 10 J

ADD EDGE

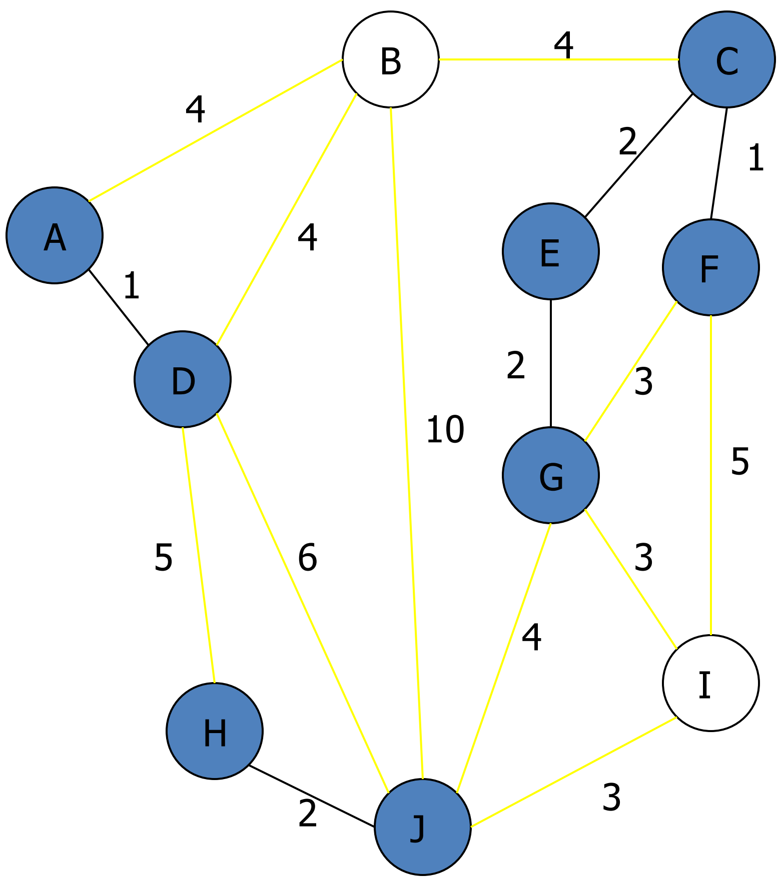


A
B
C
D
E
F
G
H
I
J
4
4
1
4
2
1
2
3
10
5
5
6
3
4
2
3

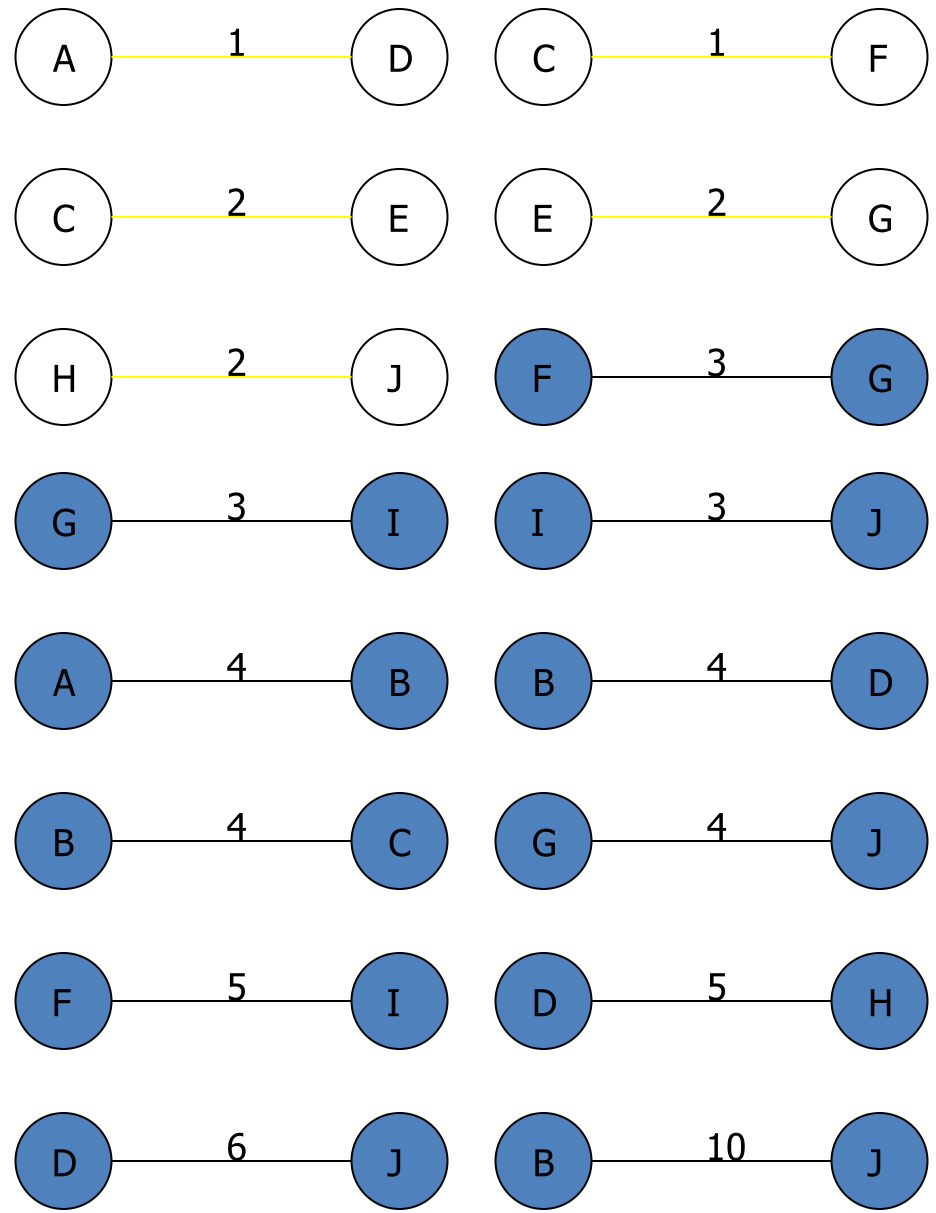


A 1 D
C 1 F
C 2 E
E 2 G
H 2 J
F 3 G
G 3 I
I 3 J
A 4 B
B 4 D
B 4 C
G 4 J
F 5 I
D 5 H
D 6 J
B 10 J

ADD EDGE



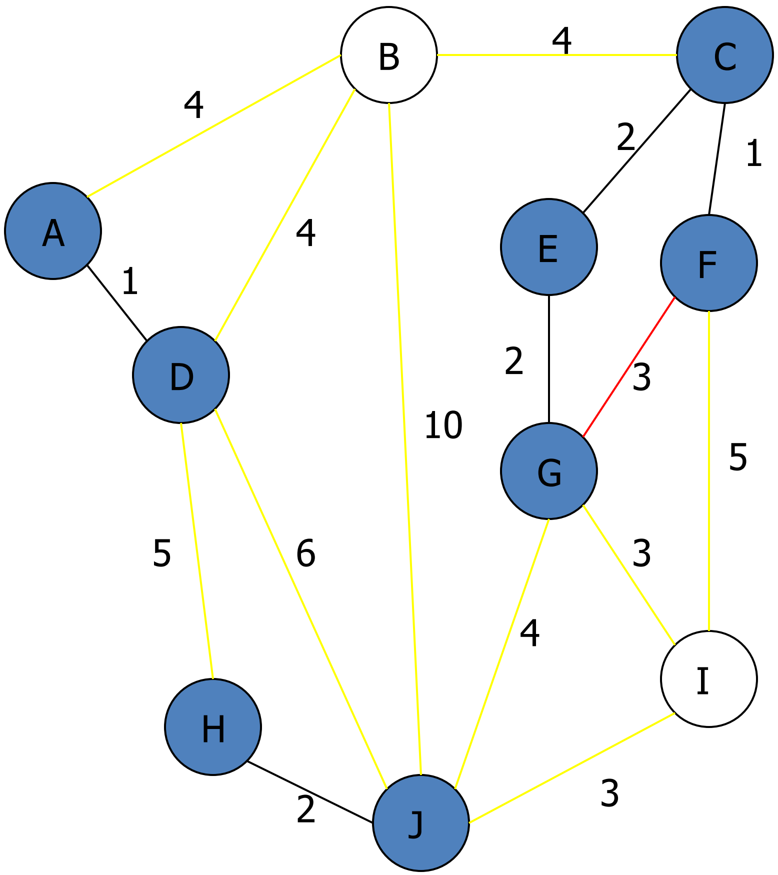
A
B
C
D
E
F
G
H
I
J
4
4
1
4
2
1
2
10
3
5
5
6
3
4
2
3

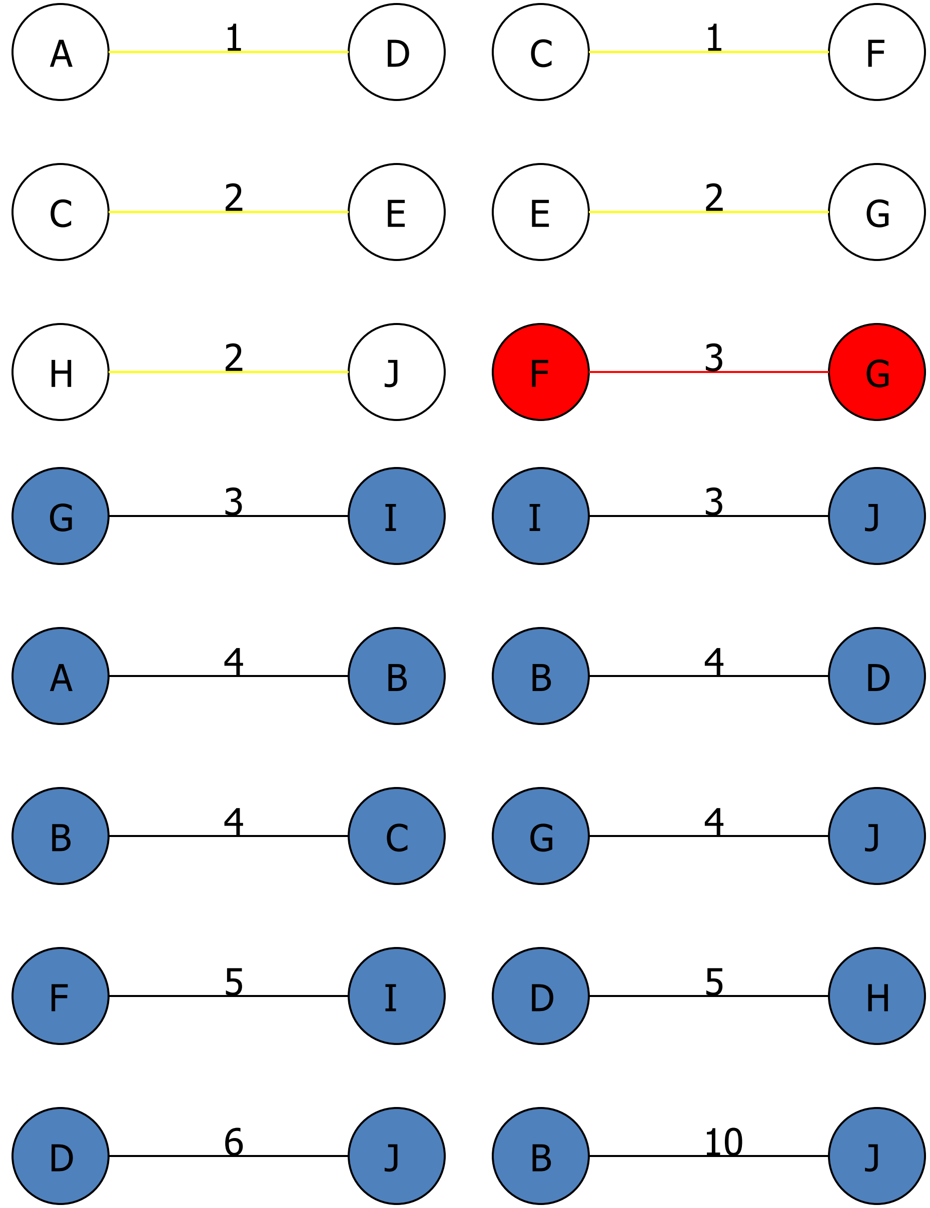


A 1 D
C 1 F
C 2 E
E 2 G
H 2 J
F 3 G
G 3 I
I 3 J
A 4 B
B 4 D
B 4 C
G 4 J
F 5 I
D 5 H
D 6 J
B 10 J

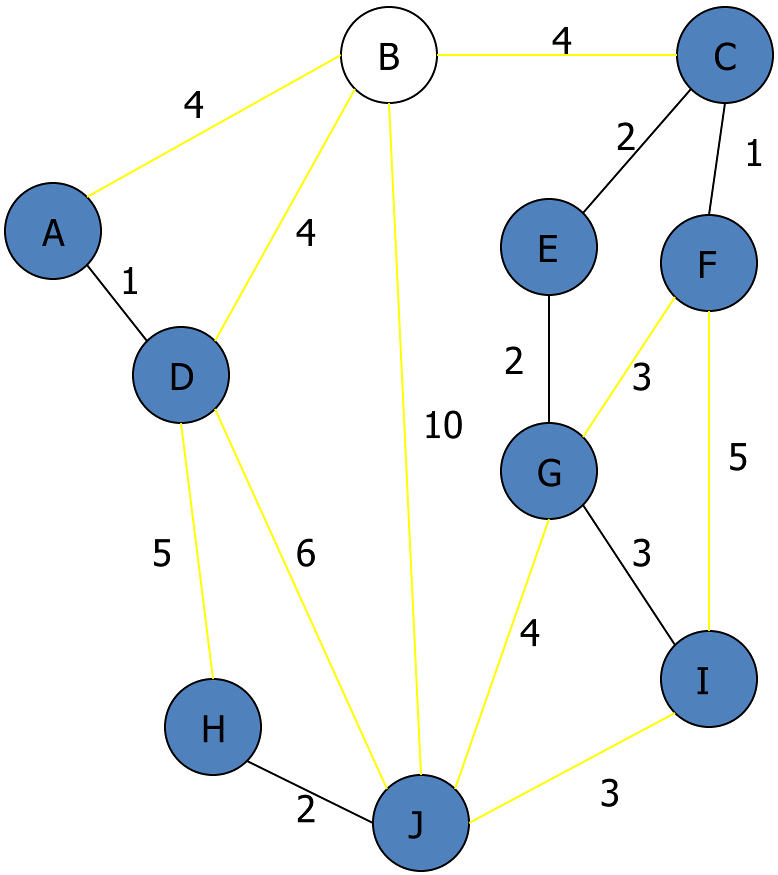
CYCLE

DON’T ADD EDGE

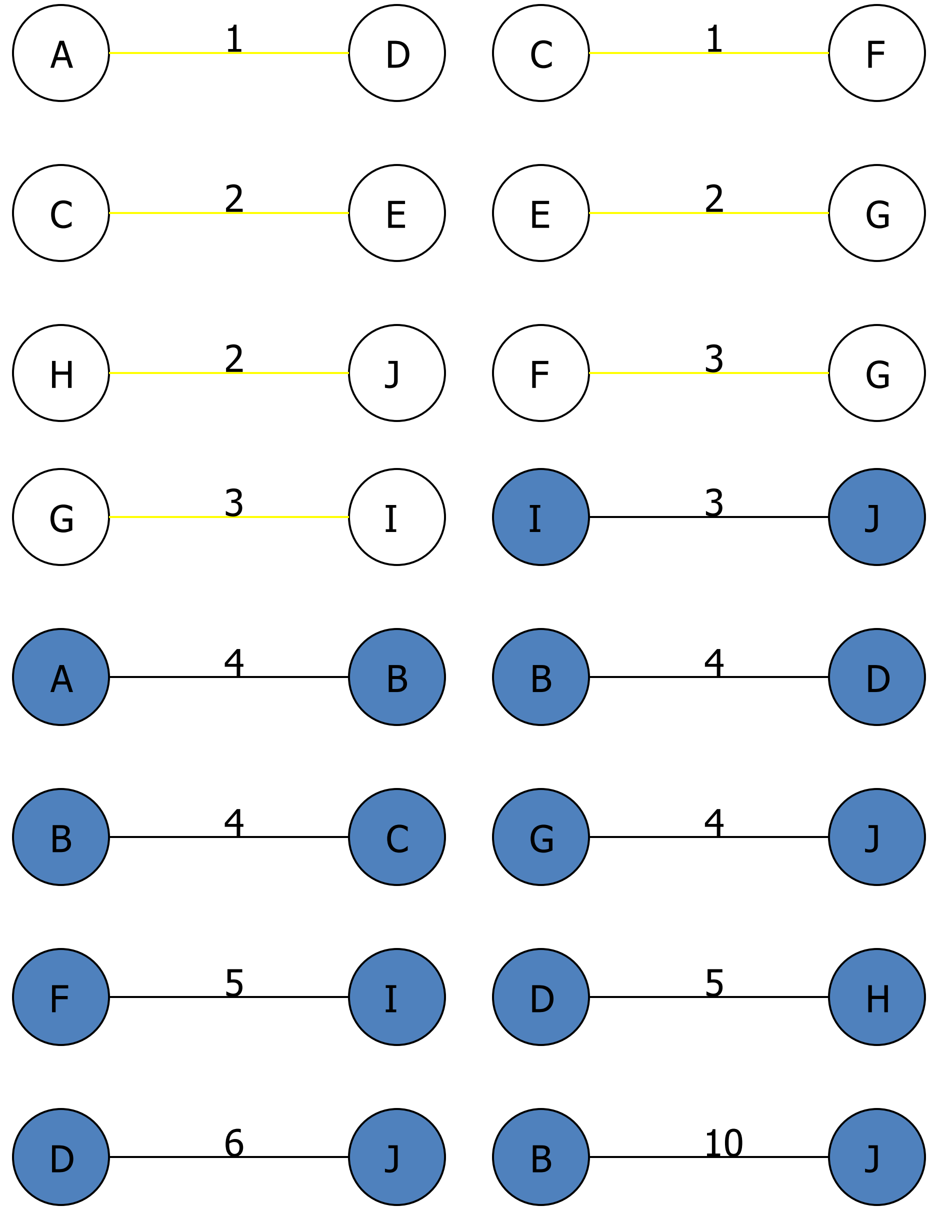




ADD EDGE

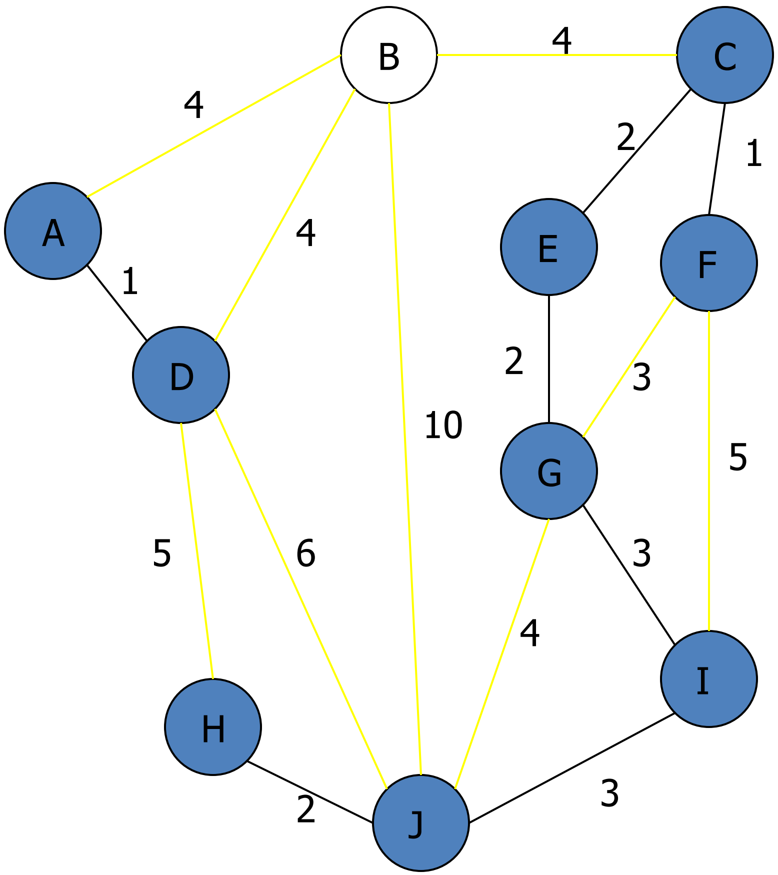


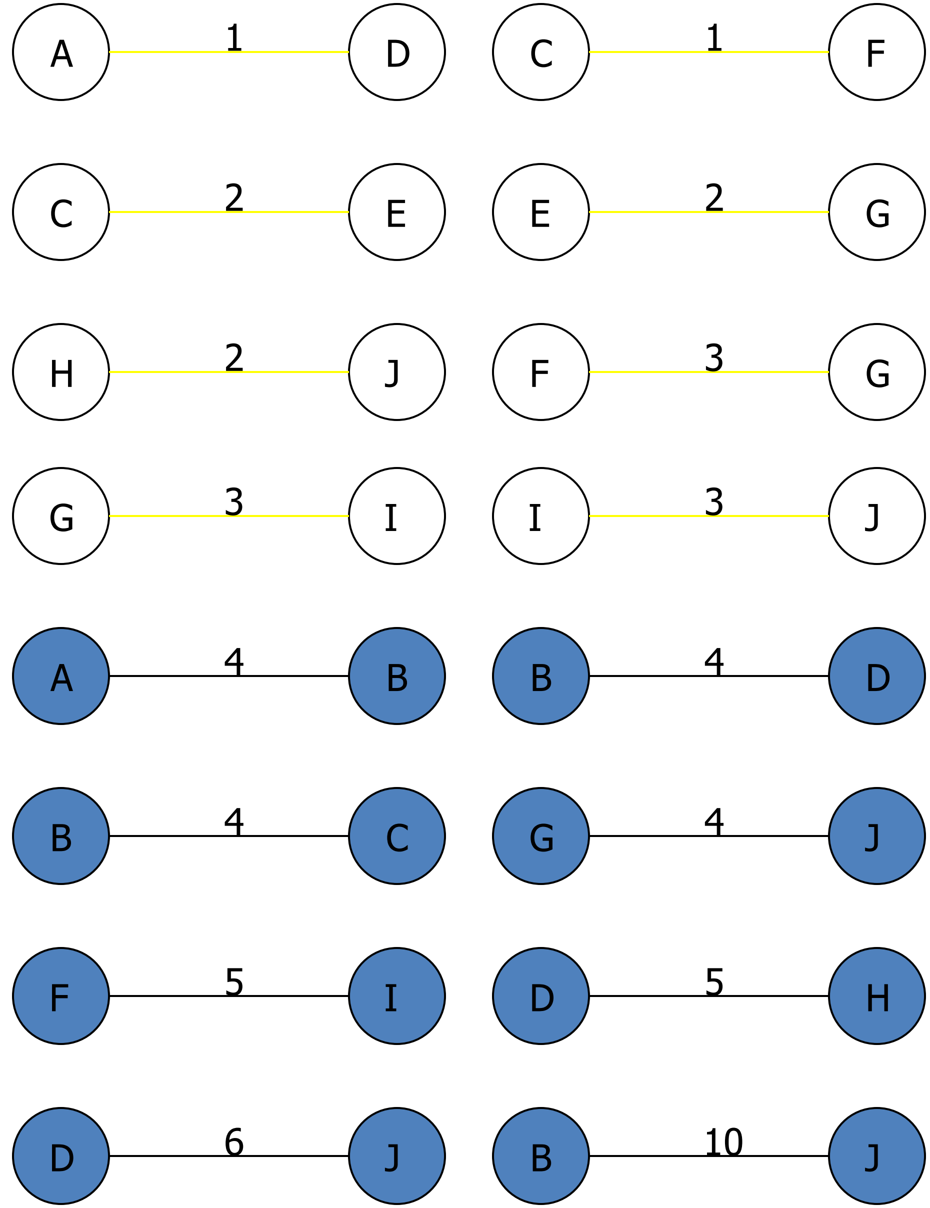
A
B
C
D
E
F
G
H
I
J
4
4
1
4
2
1
2
3
10
5
5
6
3
4
2
3



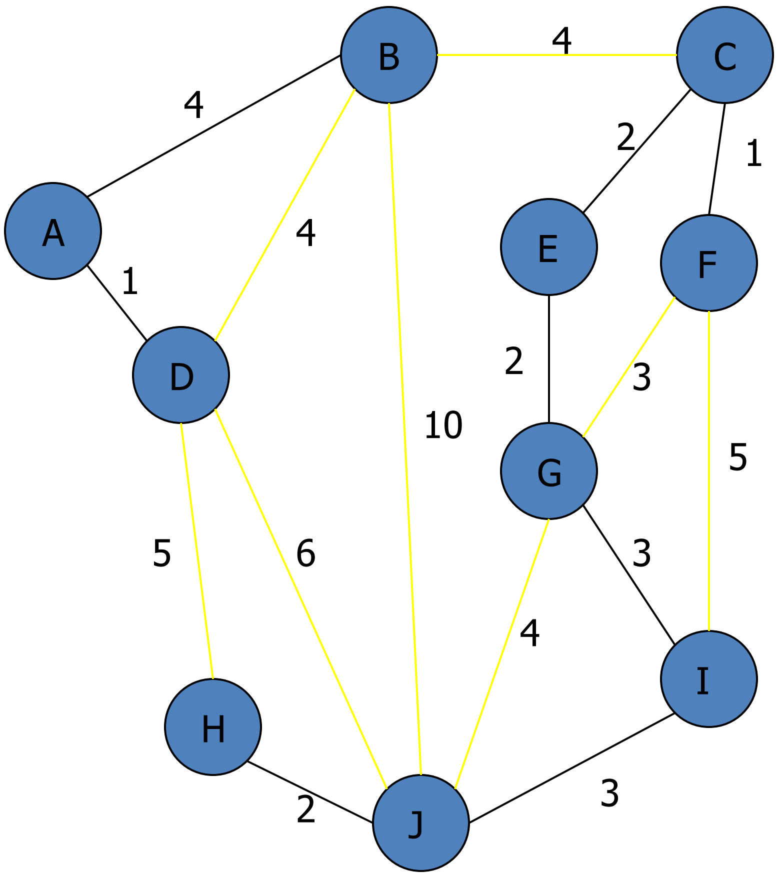
A 1 D
C 1 F
C 2 E
E 2 G
H 2 J
F 3 G
G 3 I
I 3 J
A 4 B
B 4 D
B 4 C
G 4 J
F 5 I
D 5 H
D 6 J
B 10 J

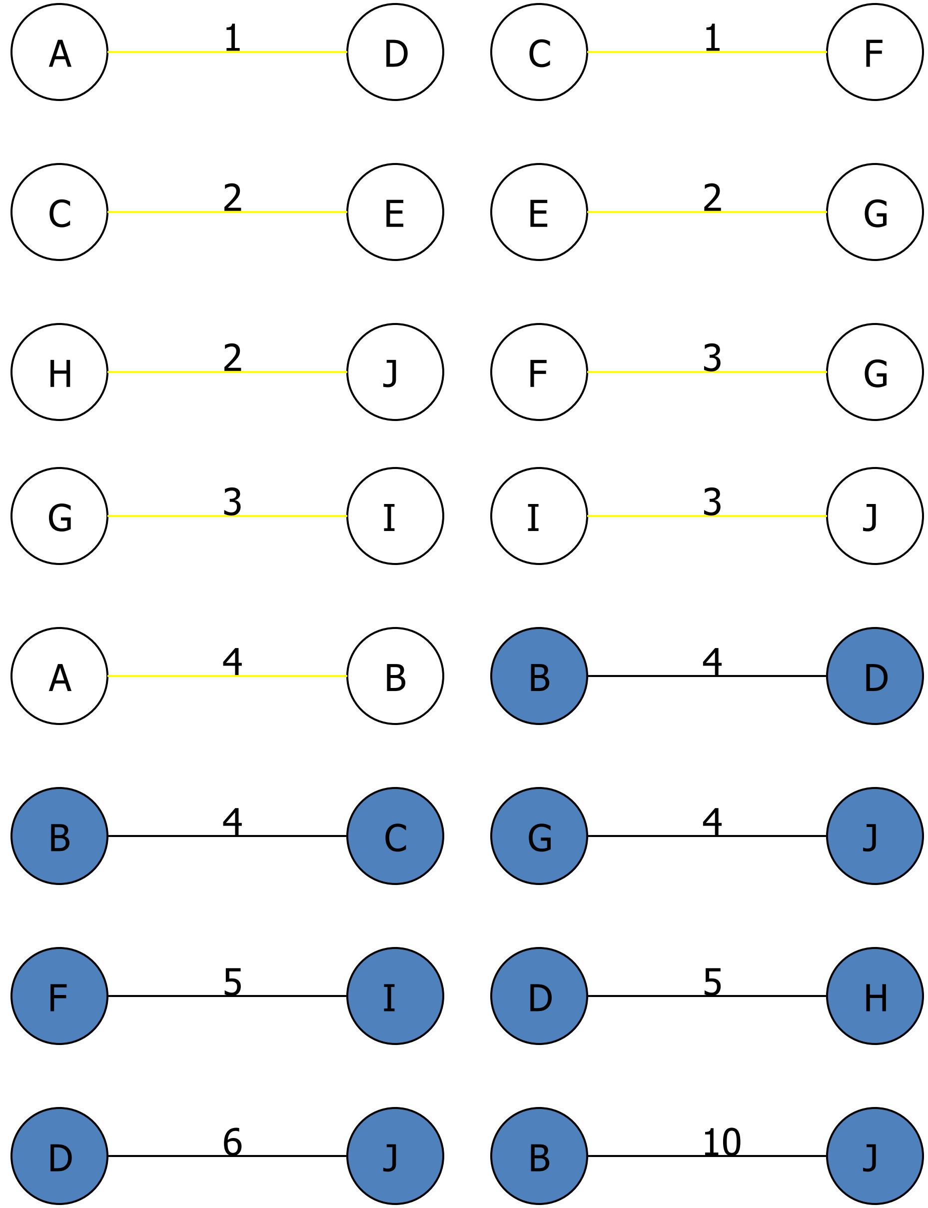
ADD EDGE


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

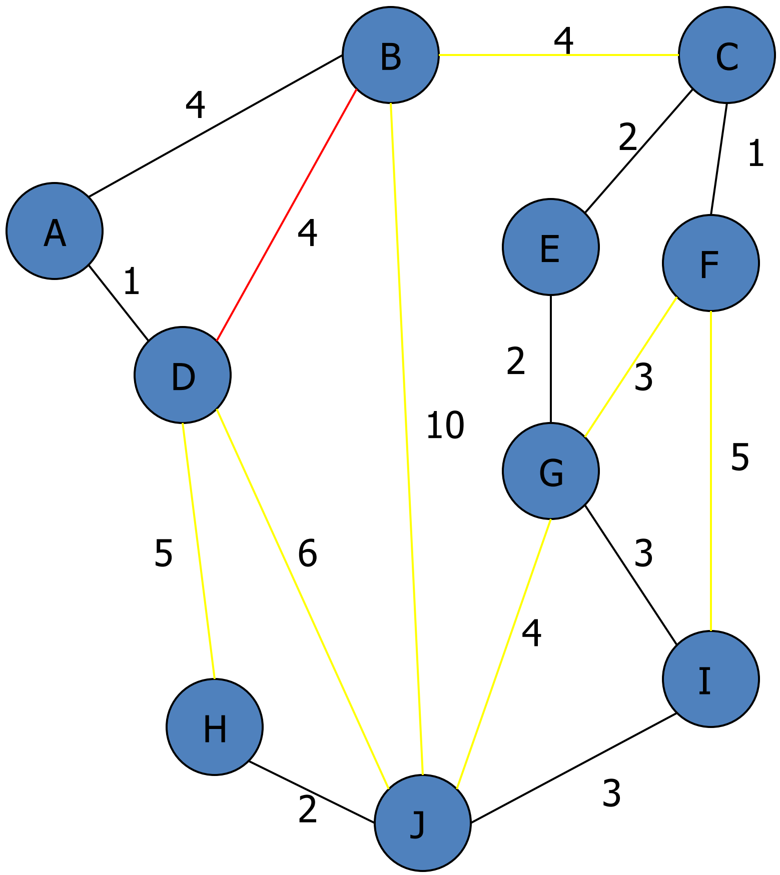
ADD EDGE

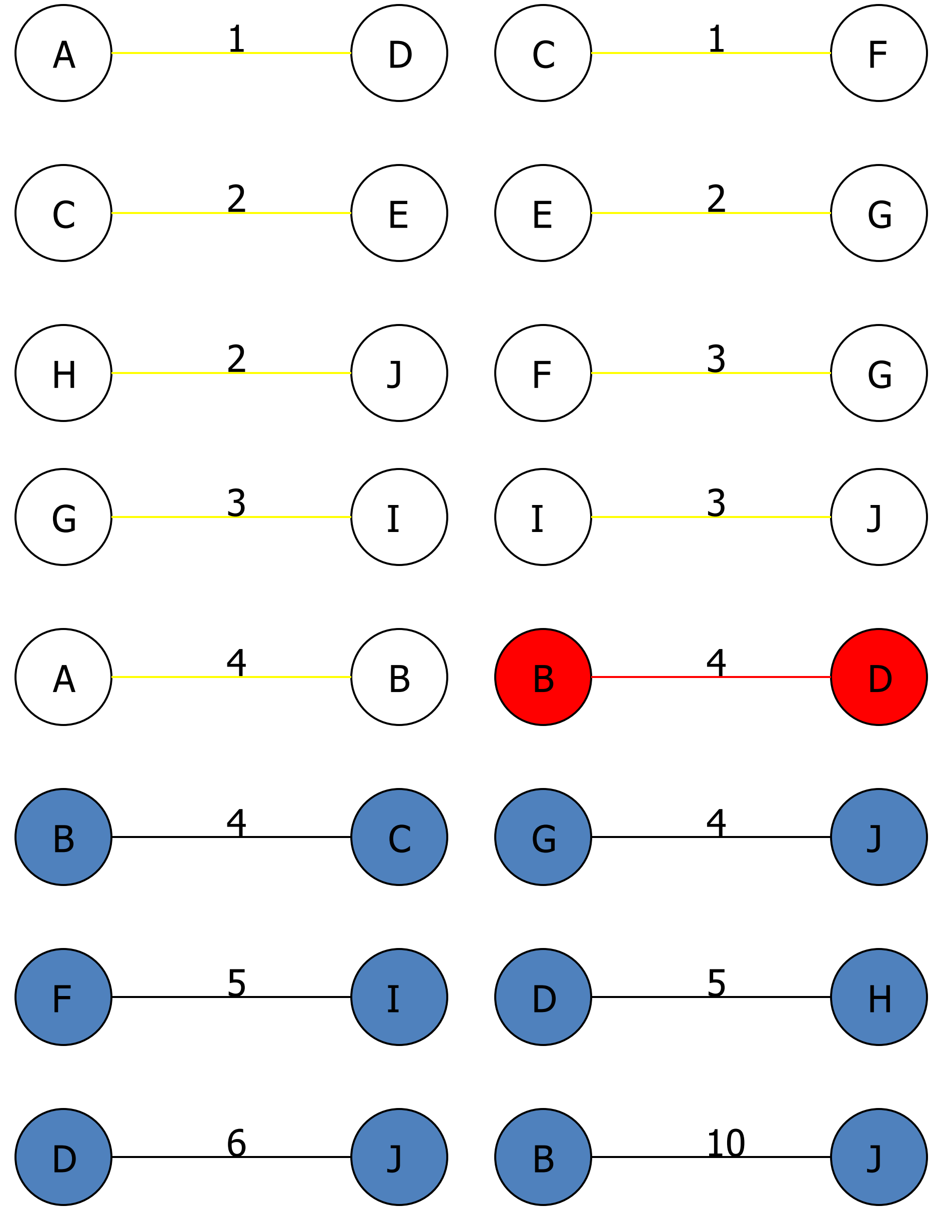

A
B
C
D
E
F
G
H
I
J
4
4
2
1
1
4
2
3
10
5
5
6
3
4
2
3


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

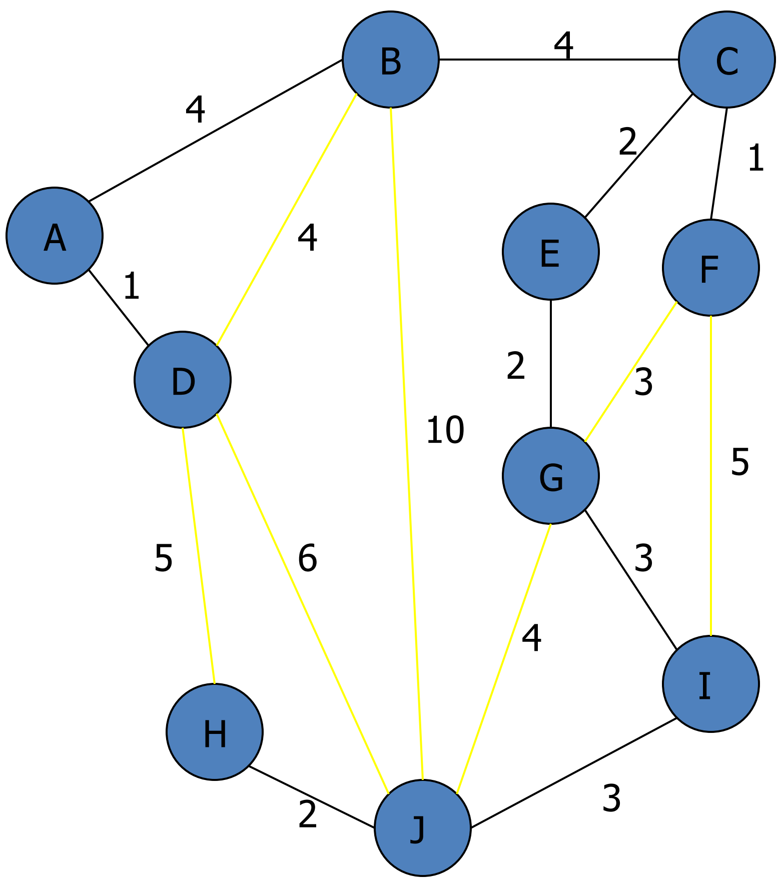
CYCLE

DON'T ADD EDGE

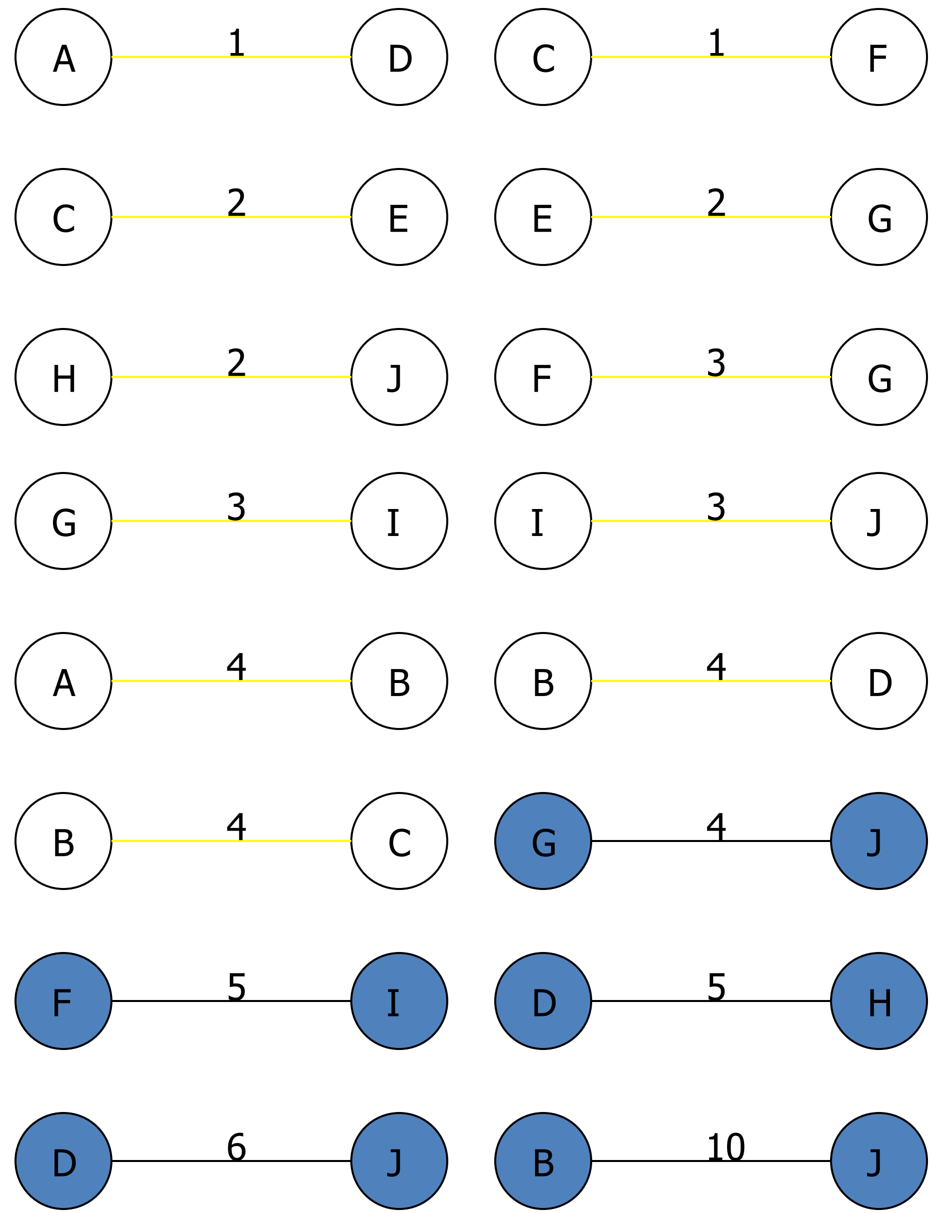




ADD EDGE

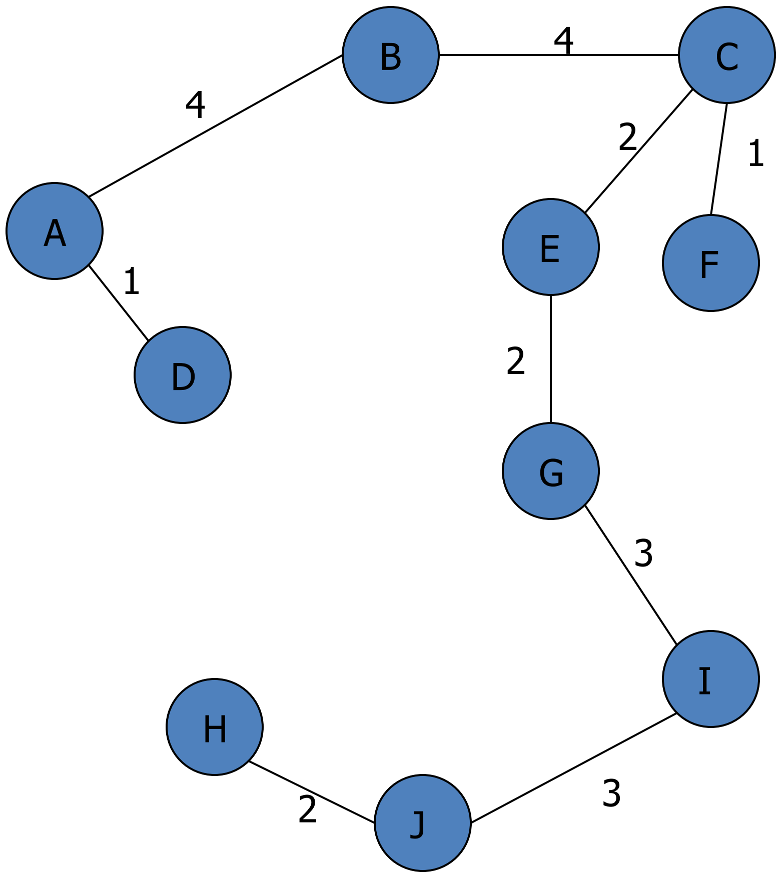


A
B
C
D
E
F
G
H
I
J
4
4
2
1
1
4
2
3
10
5
5
6
3
4
2
3

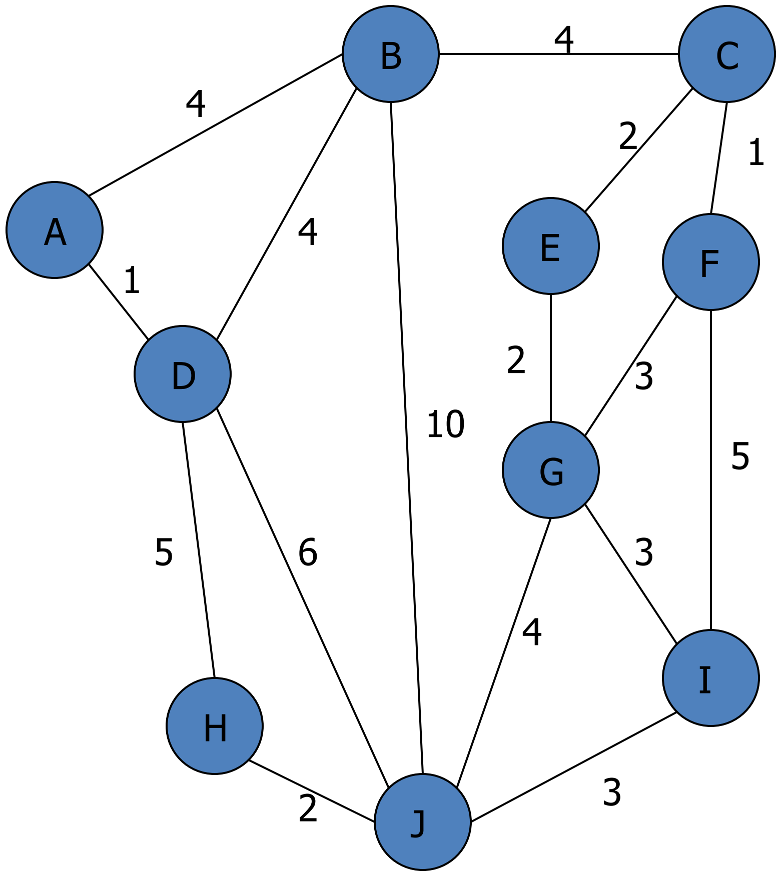


A 1 D
C 1 F
C 2 E
E 2 G
H 2 J
F 3 G
G 3 I
I 3 J
A 4 B
B 4 D
B 4 C
G 4 J
F 5 I
D 5 H
D 6 J
B 10 J

MINIMUM SPANNING TREE



COMPLETE GRAPH



SHORTEST PATH PROBLEM: FLOYD’S ALGORITHM

FLOYD'S ALGORITHM

Use to find the shortest path between any two nodes in the network.

- Floyd's algorithm represents an 'n' node network as a square matrix with 'n' rows and 'n' columns.

- Where:

  $D_n = [\ d_{ij}\ ]$ and S_n = Matrix of the node – Sequence

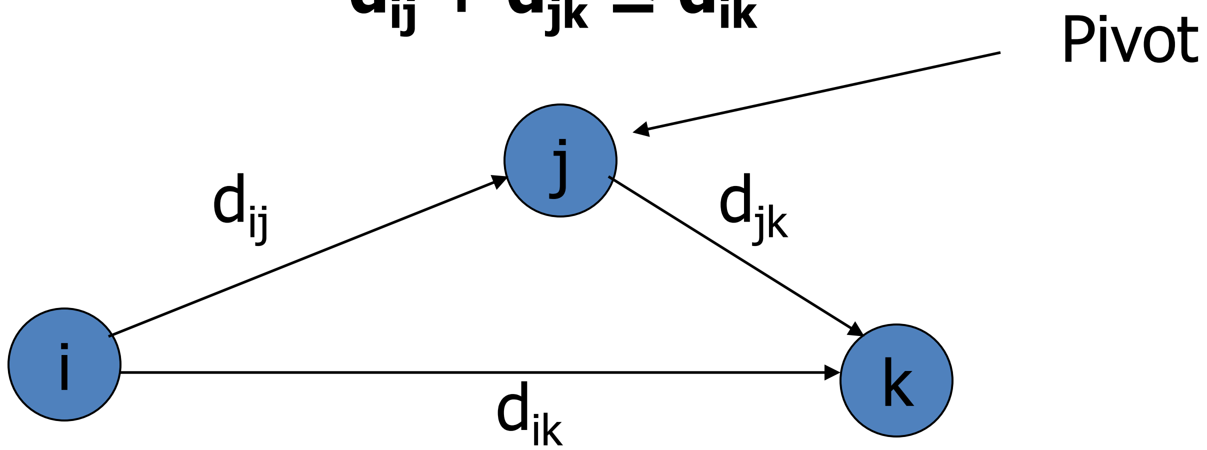
- **<u>For matrix D_n:</u>** Entry (i, j) of the matrix gives the distance d_{ij} from node – 'i' to node – 'j', which is finite if 'i' is linked directly to 'j', and infinite otherwise.

FLOYD'S ALGORITHM

IDEA OF FLOYD'S ALGORITHM:

Let three nodes i, j, and k shown in the below figure with the connecting distances shown on the three arcs, it is shorter to reach 'k' from 'i' passing through 'j' if:

$$\mathbf{d_{ij} + d_{jk} \leq d_{ik}}$$



Here, it is optimal to replace the direct route from i → k with the indirect route i → j → k.

(This is called TRIPLE OPERATION.)

FLOYD'S ALGORITHM

Step–0: Define the starting distance matrix D_0 and node sequence matrix S_0 as given subsequently. The diagonal elements are marked with (–) to indicates that they are blocked. Set k = 1.

General Step – k: Define row 'k' and column 'k' as pivot row and pivot column. Apply the triple operation to each element d_{ij} in $D_{(k-1)}$, for all i and j. if the condition

$$dij + djk \leq dik \quad , (i \neq k, j \neq k, \text{and } i \neq j)$$

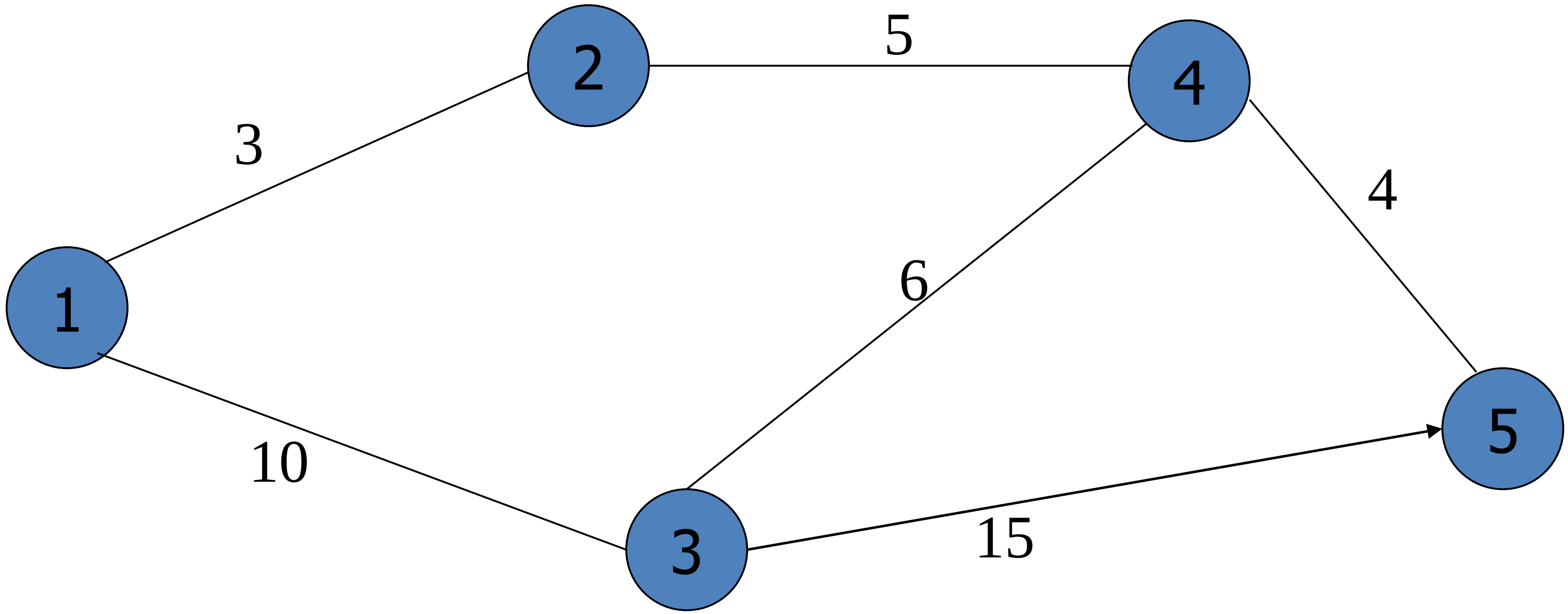
is satisfied, make the following changes:

a) Create D_k by replacing d_{ij} in $D_{(k-1)}$ with $d_{ik} + d_{kj}$.

b) Create S_k by replacing s_{ij} in $S_{(k-1)}$ with k. Set k = k+1, and repeat Step – k.

EXAMPLE:

Determine the shortest routes with their distances between node–1 & node–5. Also between node–2 & node–3 using Floyd's algorithm.



SOLUTION:

ITERATION – 0:

$$D_0 = \begin{array}{c|ccccc} & 1 & 2 & 3 & 4 & 5 \\ \hline 1 & - & 3 & 10 & \infty & \infty \\ \hline 2 & 3 & - & \infty & 5 & \infty \\ \hline 3 & 10 & \infty & - & 6 & 15 \\ \hline 4 & \infty & 5 & 6 & - & 4 \\ \hline 5 & \infty & \infty & \infty & 4 & - \end{array} \quad , \quad S_0 = \begin{array}{c|ccccc} & 1 & 2 & 3 & 4 & 5 \\ \hline 1 & - & 2 & 3 & 4 & 5 \\ \hline 2 & 1 & - & 3 & 4 & 5 \\ \hline 3 & 1 & 2 & - & 4 & 5 \\ \hline 4 & 1 & 2 & 3 & - & 5 \\ \hline 5 & 1 & 2 & 3 & 4 & - \end{array}$$

ITERATION – 1:

- Set k = 1, thus PIVOT column – 1 and row – 1.

- Improvements can be made for d_{23} and d_{32}.

    1. Replace d_{23} by $d_{21} + d_{13} = 3 + 10 = 13$ & Set $S_{23} = 1$.
    2. Replace d_{32} by $d_{31} + d_{12} = 10 + 3 = 13$ & Set $S_{32} = 1$.

$$D_1 = \begin{array}{c|ccccc} & 1 & 2 & 3 & 4 & 5 \\ \hline 1 & - & 3 & 10 & \infty & \infty \\ 2 & 3 & - & 13 & 5 & \infty \\ 3 & 10 & 13 & - & 6 & 15 \\ 4 & \infty & 5 & 6 & - & 4 \\ 5 & \infty & \infty & \infty & 4 & - \end{array} , \quad S_1 = \begin{array}{c|ccccc} & 1 & 2 & 3 & 4 & 5 \\ \hline 1 & - & 2 & 3 & 4 & 5 \\ 2 & 1 & - & 1 & 4 & 5 \\ 3 & 1 & 1 & - & 4 & 5 \\ 4 & 1 & 2 & 3 & - & 5 \\ 5 & 1 & 2 & 3 & 4 & - \end{array}$$

ITERATION – 2:

- Set k = 2, thus PIVOT column – 2 and row – 2.

- Improvements can be made for d_{14} and d_{41}.

    1. Replace d_{14} by $d_{12} + d_{24} = 3 + 5 = 8$ & Set $S_{14} = 2$.
    2. Replace d_{41} by $d_{42} + d_{21} = 5 + 3 = 8$ & Set $S_{41} = 2$.

$$
D_2 = \begin{array}{c|ccccc}
 & 1 & 2 & 3 & 4 & 5 \\
\hline
1 & - & 3 & 10 & 8 & \infty \\
\hline
2 & 3 & - & 13 & 5 & \infty \\
\hline
3 & 10 & 13 & - & 6 & 15 \\
\hline
4 & 8 & 5 & 6 & - & 4 \\
\hline
5 & \infty & \infty & \infty & 4 & - \\
\hline
\end{array}
, \quad S_2 = \begin{array}{c|ccccc}
 & 1 & 2 & 3 & 4 & 5 \\
\hline
1 & - & 2 & 3 & 2 & 5 \\
\hline
2 & 1 & - & 1 & 4 & 5 \\
\hline
3 & 1 & 1 & - & 4 & 5 \\
\hline
4 & 2 & 2 & 3 & - & 5 \\
\hline
5 & 1 & 2 & 3 & 4 & - \\
\hline
\end{array}
$$

ITERATION – 3:

- Set k = 3, thus PIVOT column – 3 and row – 3.
- Improvements can be made for d_{15} and d_{25}.

1. Replace d_{15} by $d_{13} + d_{35} = 10 + 15 = 25$ & Set $S_{15} = 3$.
2. Replace d_{25} by $d_{23} + d_{35} = 13 + 15 = 28$ & Set $S_{25} = 3$.

$$D_3 = \begin{array}{c|ccccc} & 1 & 2 & 3 & 4 & 5 \\ \hline 1 & - & 3 & 10 & 8 & 25 \\ \hline 2 & 3 & - & 13 & 5 & 28 \\ \hline 3 & 10 & 13 & - & 6 & 15 \\ \hline 4 & 8 & 5 & 6 & - & 4 \\ \hline 5 & \infty & \infty & \infty & 4 & - \\ \hline \end{array}, \quad S_3 = \begin{array}{c|ccccc} & 1 & 2 & 3 & 4 & 5 \\ \hline 1 & - & 2 & 3 & 2 & 3 \\ \hline 2 & 1 & - & 1 & 4 & 3 \\ \hline 3 & 1 & 1 & - & 4 & 5 \\ \hline 4 & 2 & 2 & 3 & - & 5 \\ \hline 5 & 1 & 2 & 3 & 4 & - \\ \hline \end{array}$$

ITERATION – 4:

- Set k = 4, thus PIVOT column – 4 and row – 4.
- Improvements can be made for d_{25}, d_{52}, d_{23}, d_{32}, d_{35}, d_{53}, d_{15} and d_{51}.

1. Replace d_{25} by $d_{24} + d_{45} = 5 + 4 = 9$ & Set $S_{25} = 4$.
2. Replace d_{52} by $d_{54} + d_{42} = 4 + 5 = 9$ & Set $S_{52} = 4$.
3. Replace d_{23} by $d_{24} + d_{43} = 5 + 6 = 11$ & Set $S_{23} = 4$.
4. Replace d_{32} by $d_{34} + d_{42} = 6 + 5 = 11$ & Set $S_{32} = 4$.
5. Replace d_{35} by $d_{34} + d_{45} = 6 + 4 = 10$ & Set $S_{35} = 4$.
6. Replace d_{53} by $d_{54} + d_{43} = 4 + 6 = 10$ & Set $S_{53} = 4$.
7. Replace d_{15} by $d_{14} + d_{45} = 8 + 4 = 12$ & Set $S_{15} = 4$.
8. Replace d_{51} by $d_{54} + d_{41} = 4 + 8 = 12$ & Set $S_{51} = 4$.

$$
D_4 = \begin{array}{c|ccccc}
 & 1 & 2 & 3 & 4 & 5 \\
\hline
1 & - & 3 & 10 & 8 & 12 \\
\hline
2 & 3 & - & 11 & 5 & 9 \\
\hline
3 & 10 & 11 & - & 6 & 10 \\
\hline
4 & 8 & 5 & 6 & - & 4 \\
\hline
5 & 12 & 9 & 10 & 4 & - \\
\hline
\end{array}
\quad , \quad
S_4 = \begin{array}{c|ccccc}
 & 1 & 2 & 3 & 4 & 5 \\
\hline
1 & - & 2 & 3 & 2 & 4 \\
\hline
2 & 1 & - & 4 & 4 & 4 \\
\hline
3 & 1 & 4 & - & 4 & 4 \\
\hline
4 & 2 & 2 & 3 & - & 5 \\
\hline
5 & 4 & 4 & 4 & 4 & - \\
\hline
\end{array}
$$

ITERATION – 5:

- Set k = 5, thus PIVOT column–5 and row–5.
- No further Improvements are possible thus:

1. d_{15} = 12 ROUTE = 1 → 2 → 4 → 5

   " Route is 1 → 5 if S_{15} = 5 but S_{15} = 4. So, Route is 1 → 4 → 5 if S_{14} = 4 but = 2. So, Route is 1 → 2 → 4 → 5 if S_{12} = 2."

2. Route Node – 2 to Node – 3 is: 2 → 4 → 3

   " Route is 2 → 4 if S_{24} = 4. Route is 4 → 3 if S_{43} = 3. So, Route from node – 2 to node – 3 is: 2 → 4 → 3

MAXIMUM FLOW ALGORITHM

MAXIMUM FLOW ALGORITHM

NOTATIONS:

$\bar{C}_{ij}$ = Initial capacity of arc from node ‘ i ’ to node ‘ j ’.

$\bar{C}_{ji}$ = Initial capacity of arc from node ‘ j ’ to node ‘ i ’.

C_{ij} = Residual capacity of arc from node ‘ i ’ to node ‘ j ’.

C_{ji} = Residual capacity of arc from node ‘ j ’ to node ‘ i ’.

[a_j , i] = Denotes that a_j amount of flow is received at node ‘ j’ from node ‘ i ’.

MAXIMUM FLOW ALGORITHM

<u>Step 1</u>: Set $a_1 = \infty$, then label node_1 as [∞, -].

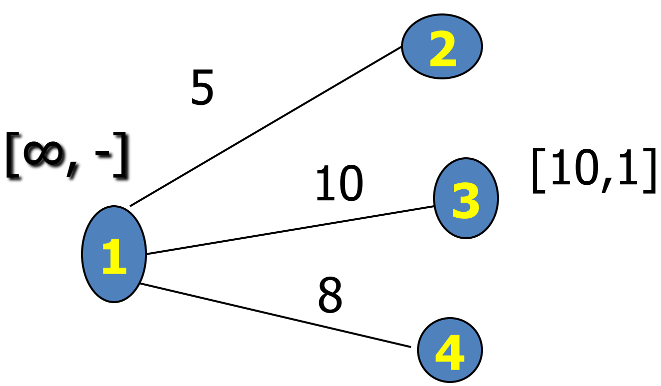
<u>Step 2</u>: S_i=Set of unlabelled nodes ‘ j ’ directly connected to node ‘ i ’, with positive capacity i.e.: $C_{ij}>0$.

<u>Step 3</u>: Determine $C_{ik} = \underset{j\,\varepsilon\,S_i}{\text{Max}} \left\{ C_{ij} \right\}$

Thus, $a_k = C_{ik}$, label node_k $\{a_k, i\}$

Set i = k, go to step_2.

<u>For Example</u>:



$S_1 = \{ 2,3,4\}$

$\text{Max} = \{ C_{12}, C_{13}, C_{14}\}$

$= C_{13}$

MAXIMUM FLOW ALGORITHM

<u>Step 4</u>: (BACKTRACKING)

If $S_i=\Phi$ in step_2 and ‛ i ′ is not the source node then backtrack to node ‛ r ′ that cause lead the node ‛ i ′ in that iteration.

<u>Step 5</u>: (Determination of Residual network)

N_p = Set of nodes involved in determined path in p^{th} iteration.

$f_p = \text{Min}\ \{ a_1, a_{k1}, a_{k2}, - - -, a_n\}$

f_p = Amount of flow in p^{th} iteration then we revise the network.

a) $(C_{ij} – f_p\ , C_{ji} + f_p)$ if flow is from node ‛ i ′ to node ‛ j ′.

b) $(C_{jI} + f_p\ , C_{ji} - f_p)$ if flow is from node ‛ j ′ to node ‛ i ′.

Reinstate any node which is removed in step_4 and make the next iteration.

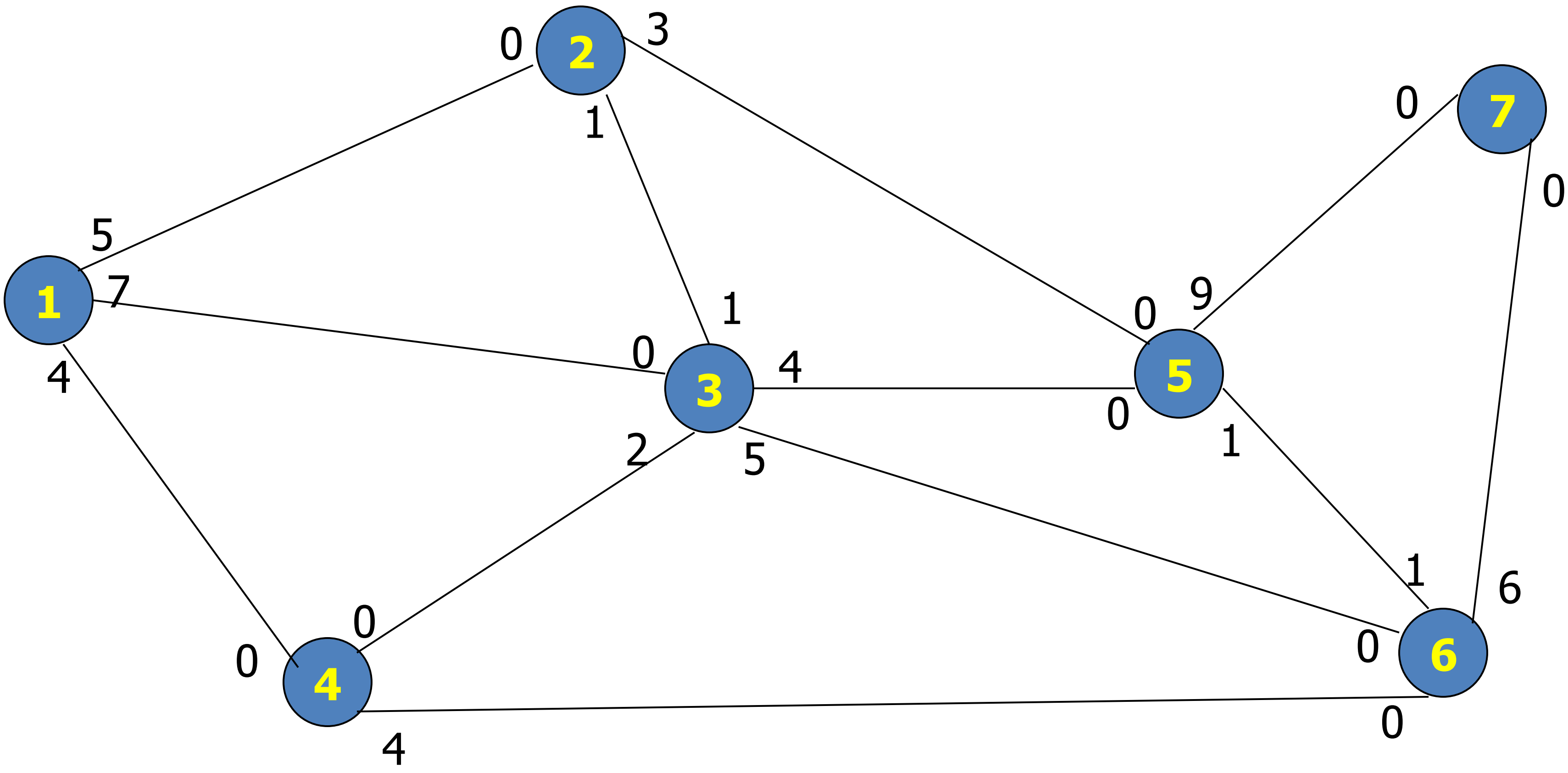
<u>Step 6</u>: (Solution) :

Maximum flow $= f_1+f_2+ - - - +f_m$

(Where ‘m’ is the number of iterations)

EXAMPLE:

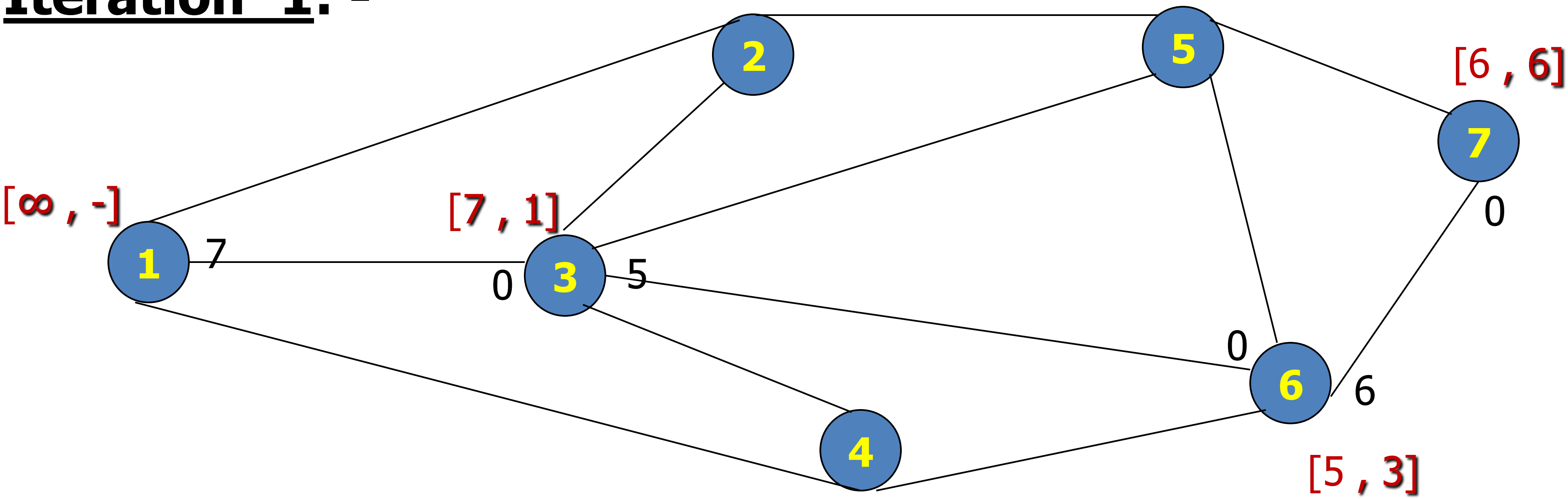
Consider the following bidirected network:



Determine the maximal flow from source node to sink node.

SOLUTION:

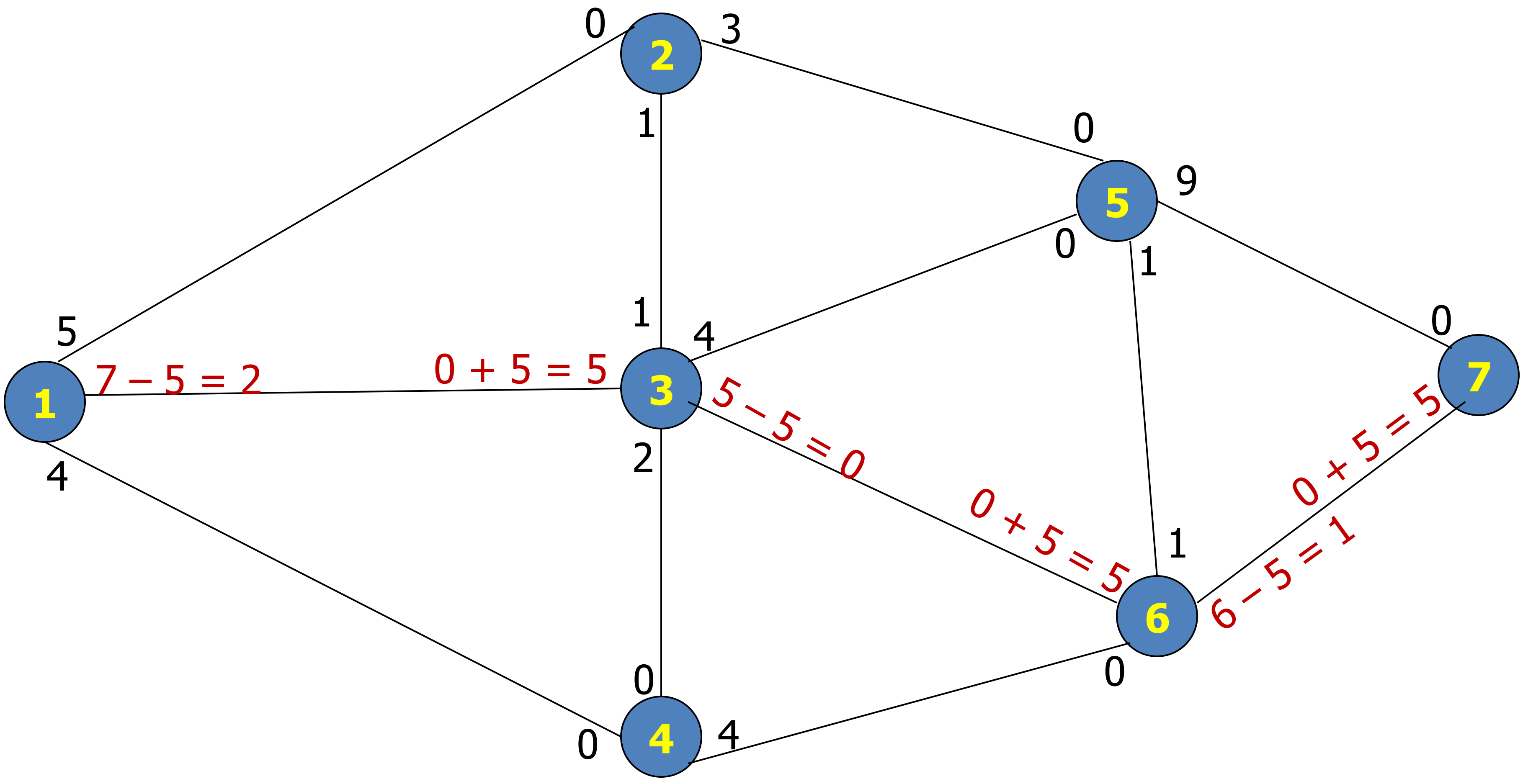
Iteration 1: -



$f_1 = 5$

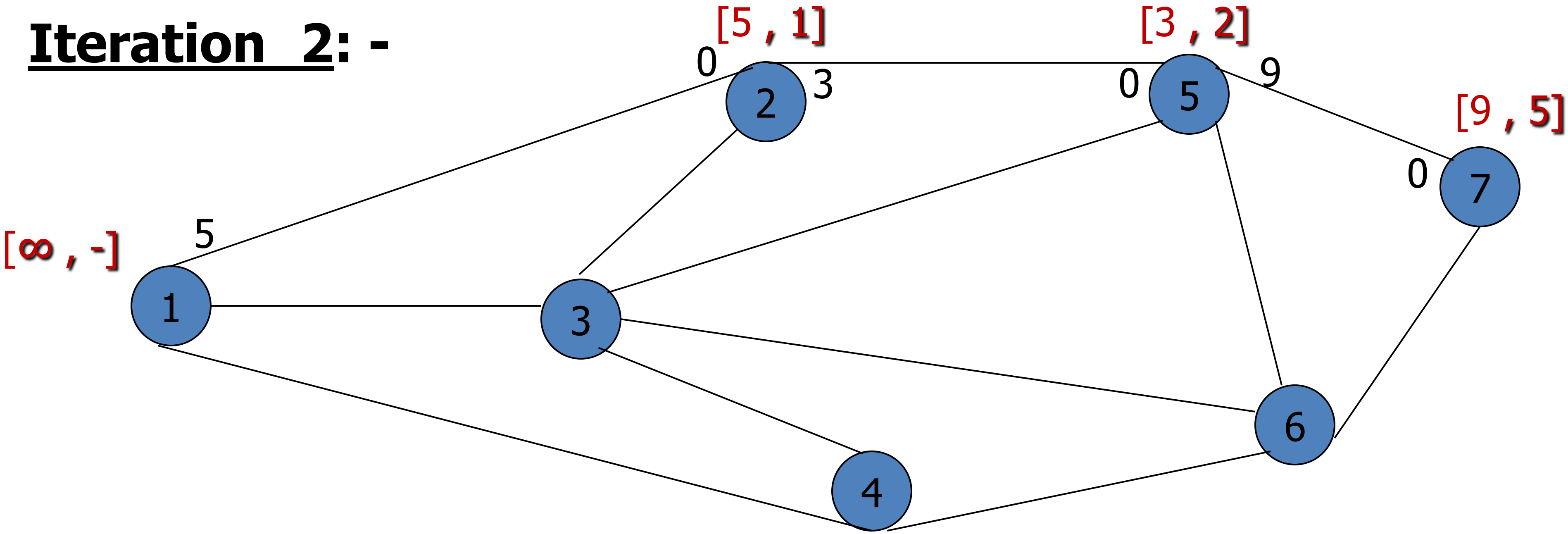
Now, We develop the Residual network from the above network ---

RESIDUAL NETWORK:



SOLUTION:

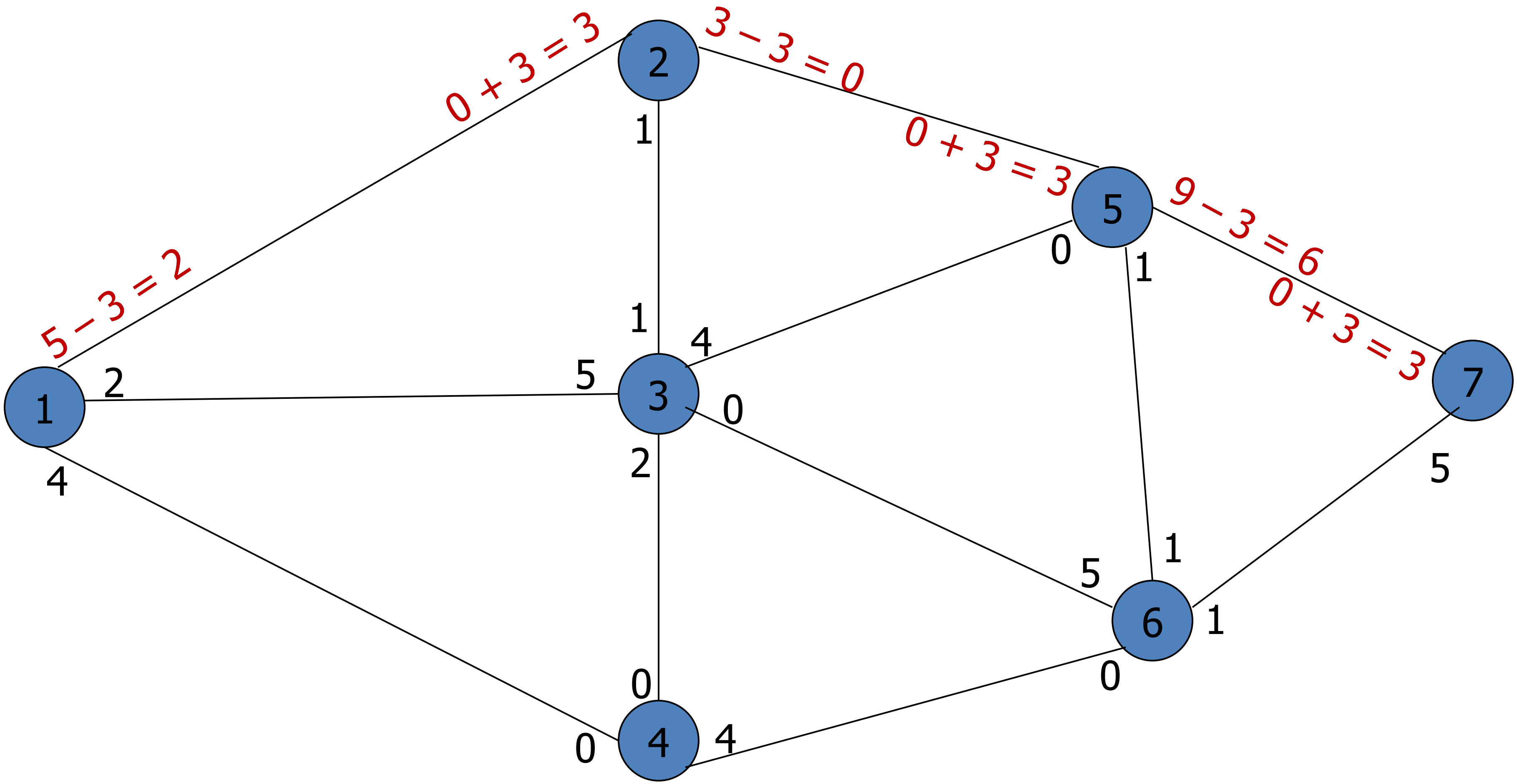
Iteration 2: -



$f_2 = 3$

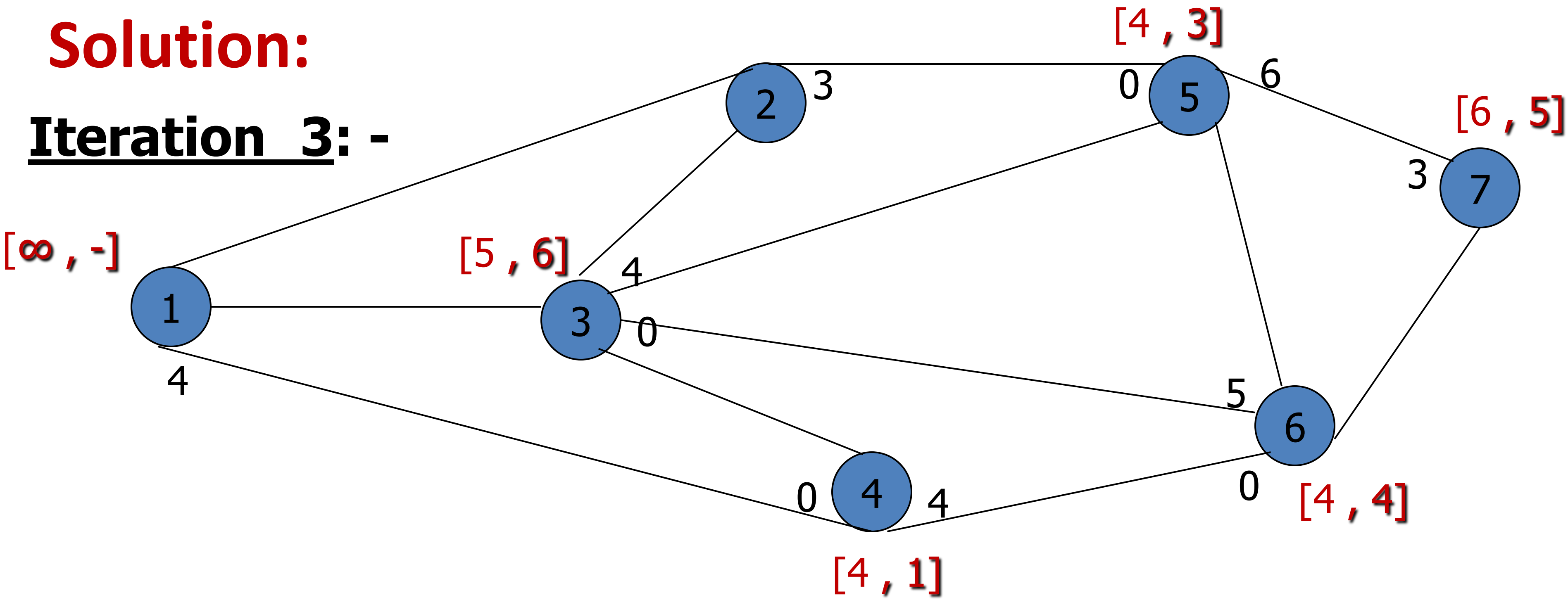
Now again, We develop the Residual network from the above network ---

RESIDUAL NETWORK:



Solution:

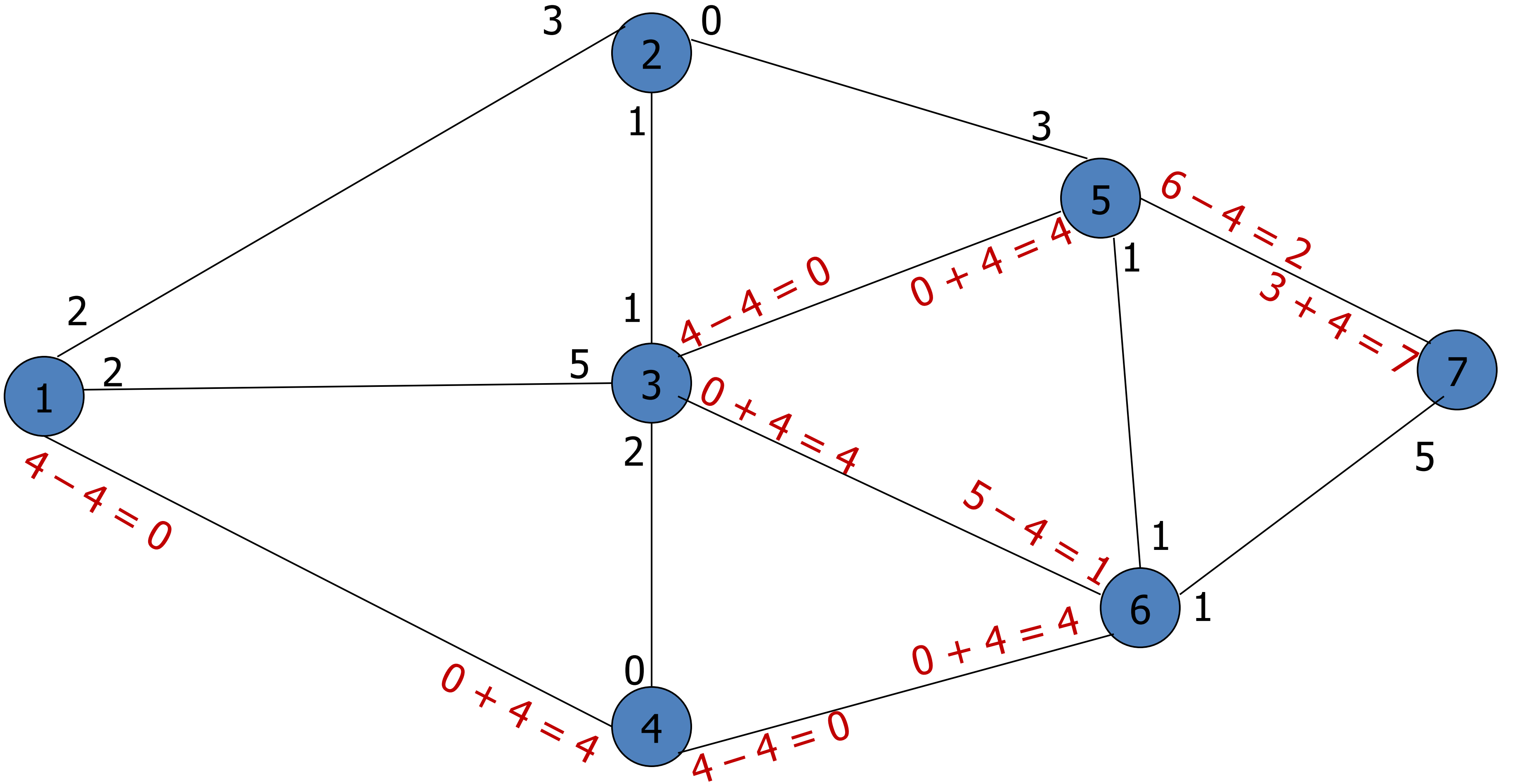
Iteration 3: -



$f_3 = 4$

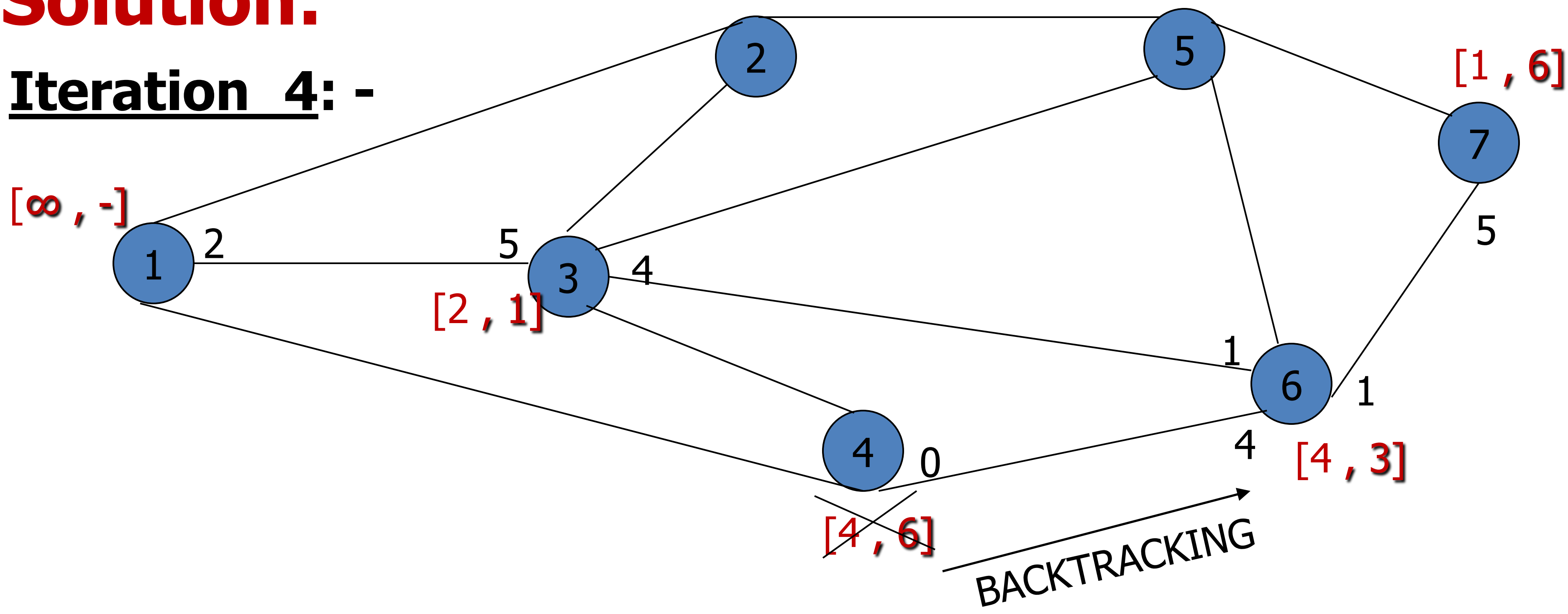
Now again, We develop the Residual network from the above network ---

RESIDUAL NETWORK:



Solution:

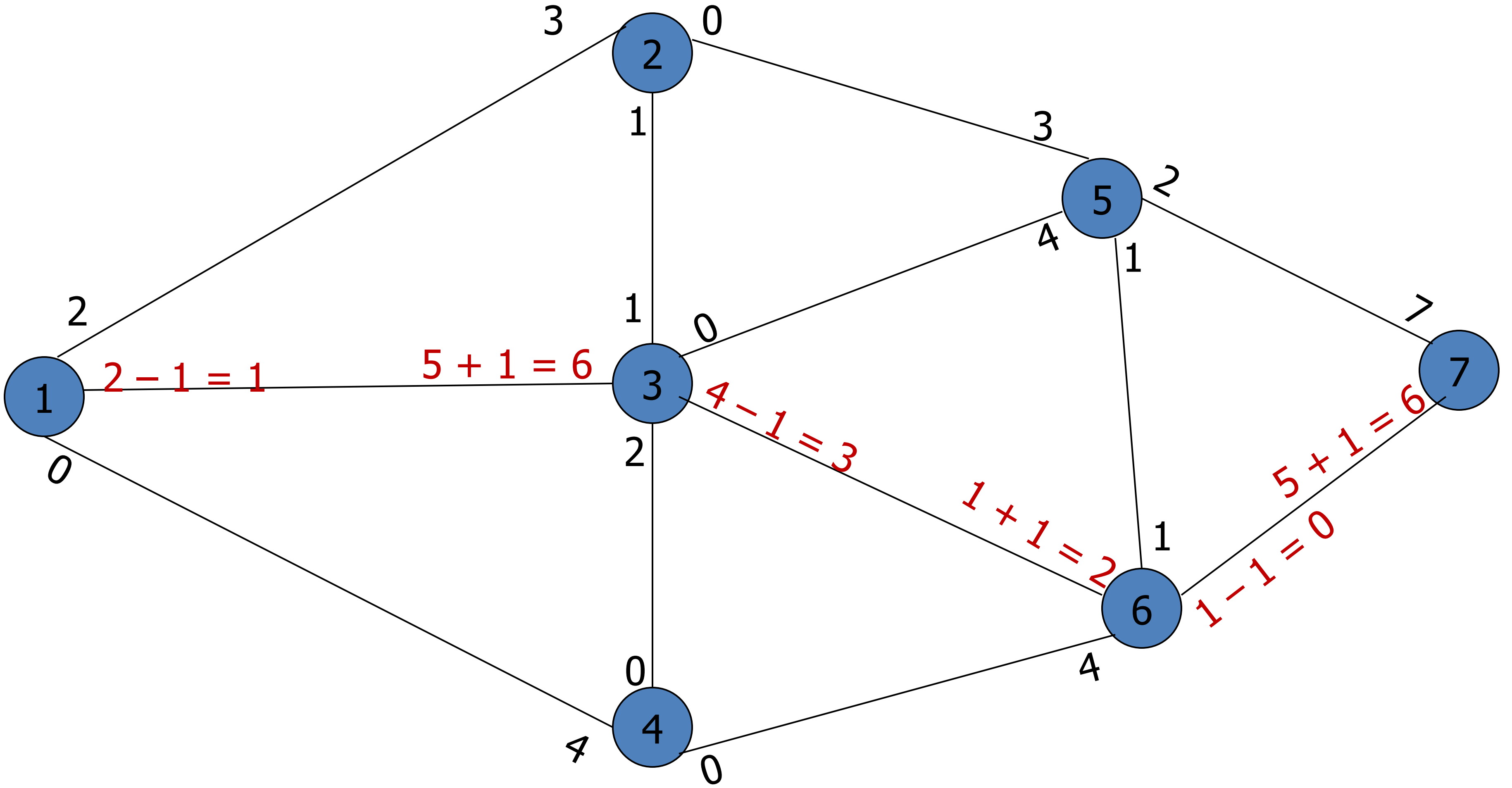
Iteration 4: -



$\underline{f_4 = 1}$

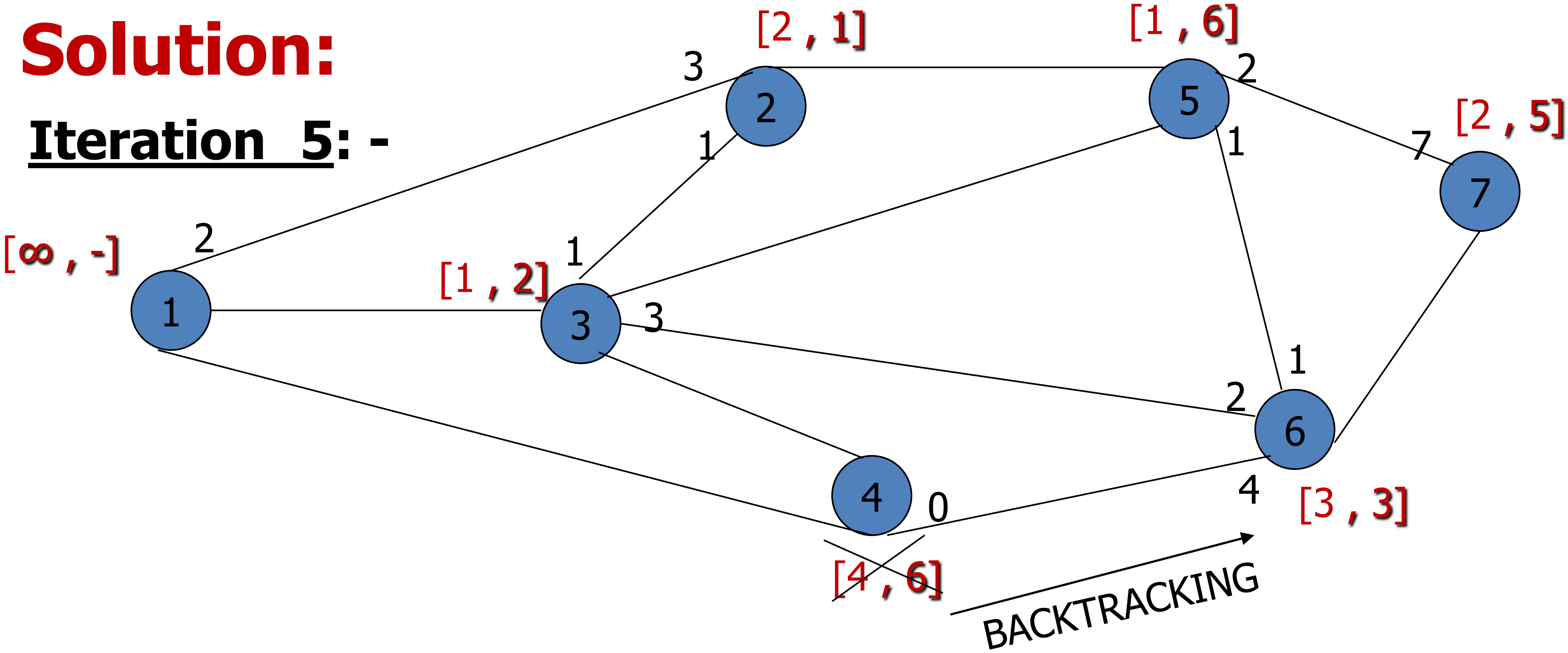
Now again, We develop the Residual network from the above network ---

RESIDUAL NETWORK:



Solution:

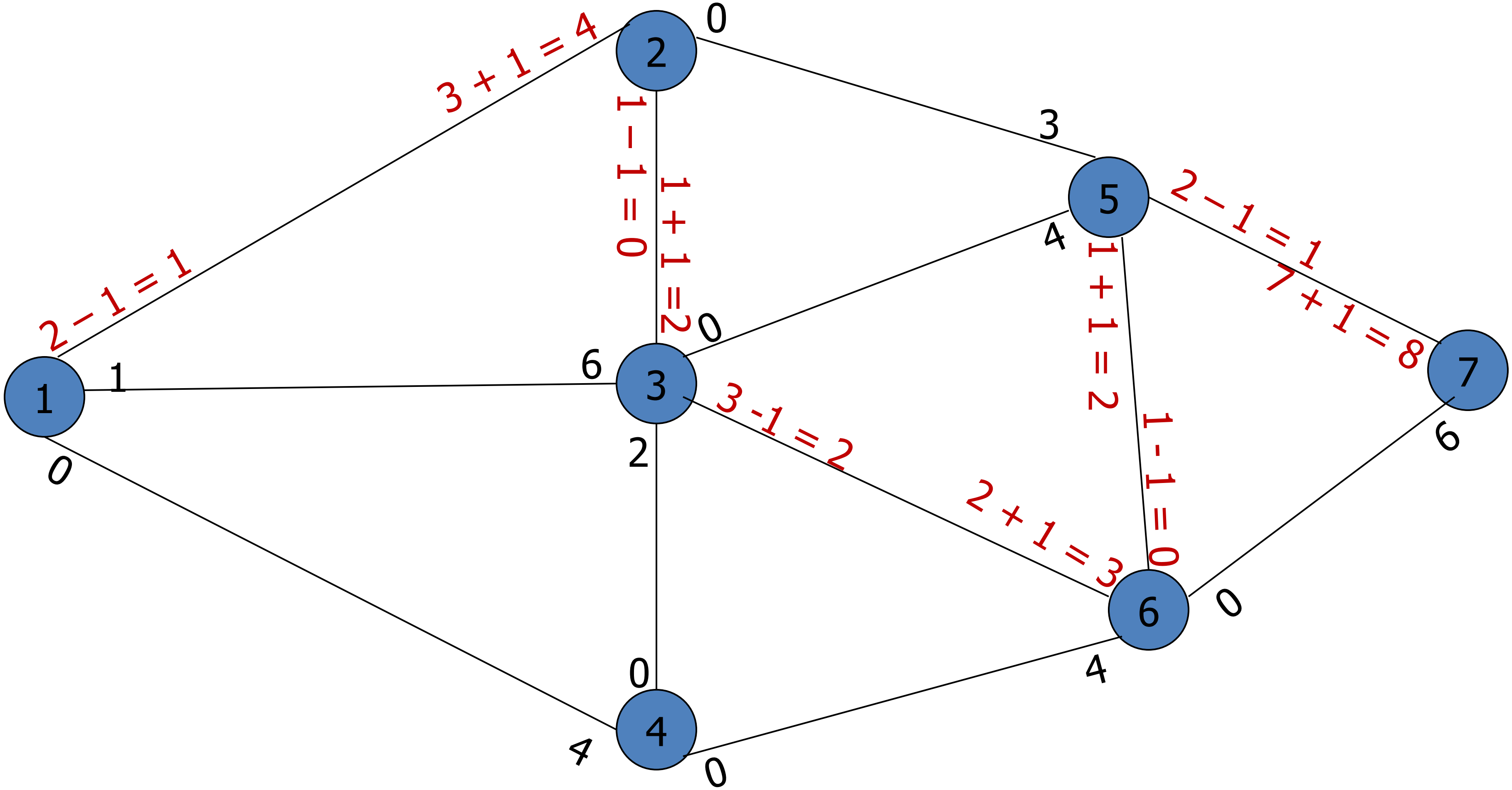
Iteration 5: -



$\underline{f_5 = 1}$

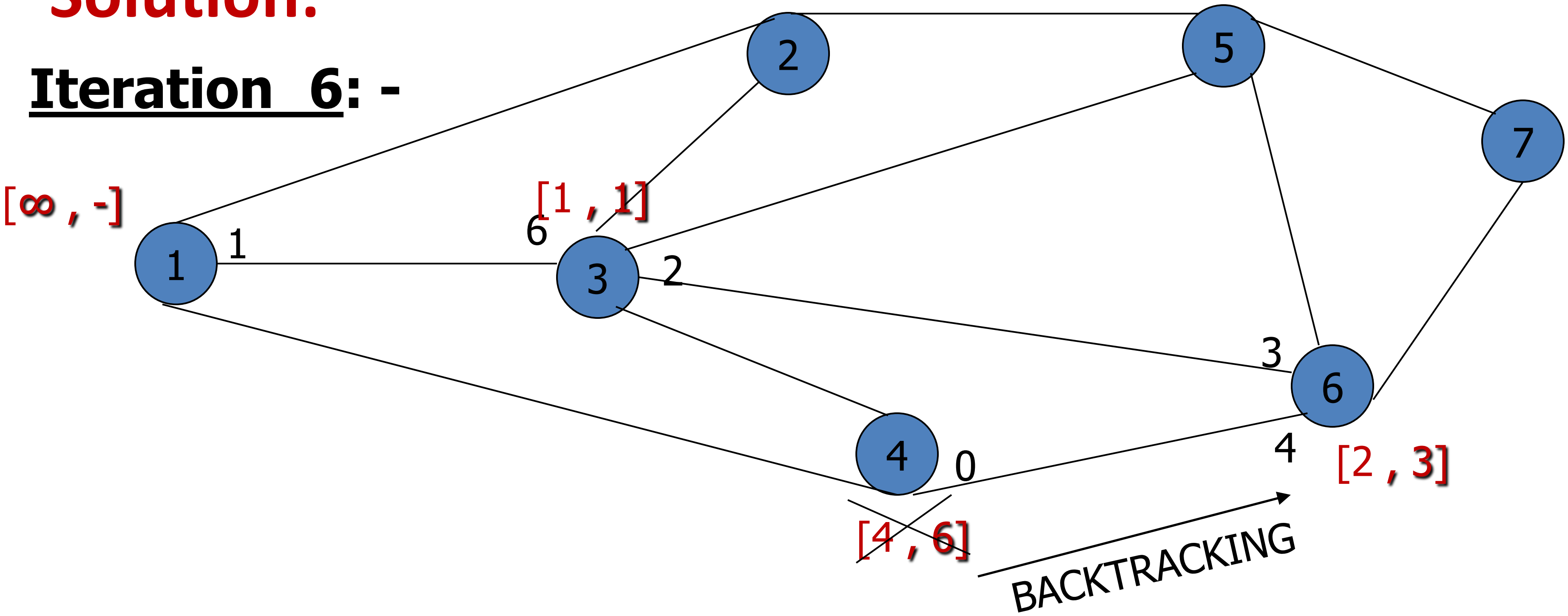
Now again, We develop the Residual network from the above network ---

RESIDUAL NETWORK:



Solution:

Iteration 6: -



Now, There is no way to move towards sink node. So,

SOLUTION:

Maximal Flow = $f_1 + f_2 + f_3 + f_4 + f_5$

$= 5 + 3 + 4 + 1 + 1$

$=$ **14**

Maximal Flow = (Initial capacities of source node) – (Ending capacities of source node)

$= 16 - 2$

$=$ **14**

Maximal Flow = Sum of ending capacities of sink node

$= 8 + 6$

$=$ **14**

PRACTICE QUESTION

- Consider the following details of piping network which is used to transfer oil.

	FLOW			FLOW	
Arc (i – j)	**f_{ij}**	**f_{ji}**	**Arc (i – j)**	**f_{ij}**	**f_{ji}**
1–2	20	–	3–4	13	–
1–3	25	–	3–5	10	8
2–3	5	10	4–5	15	–
2–4	9	4	4–6	30	–
2–5	15	–	5–6	25	–

1. Draw the flow network.
2. Determine the maximum flow from the Node–1 to Node–6.

QUESTIONS