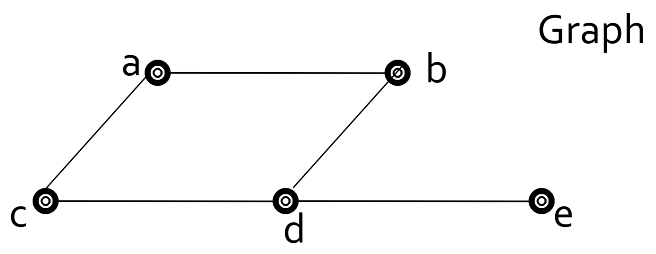
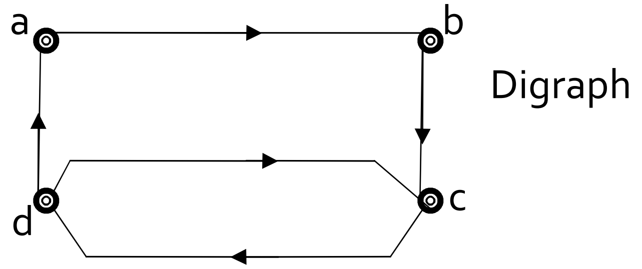
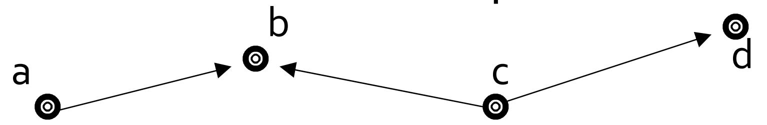
NETWORKS / GRAPHS

- **GRAPH**: A network of **NODES** (or **VERTICES**) and **ARCS** (or **EDGES**) joining the nodes with each other
- **DIGRAPH**: A graph where the arcs have an **ORIENTATION** (or **DIRECTION**).



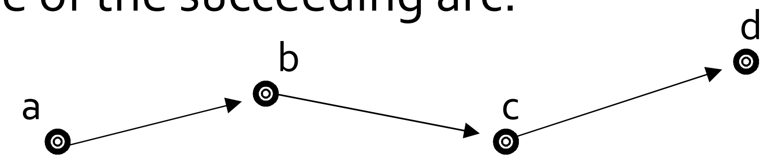


- A **CHAIN** between two nodes is a sequence of arcs where every arc has exactly one node in common with the previous arc



CHAIN: a-b-c-d; ARCS: a-b, c-b, c-d

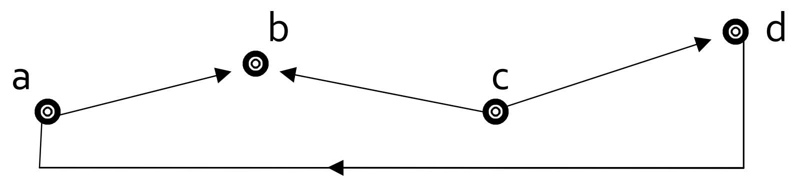
- A **PATH** is a chain of directed arcs, where the terminal node of each arc is the initial node of the succeeding arc:



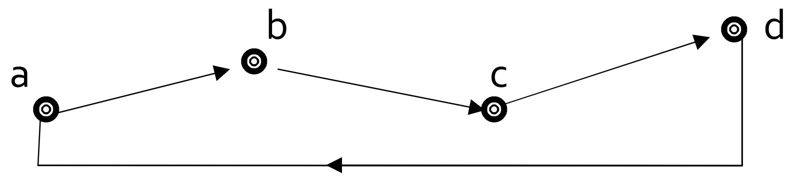
PATH: a-b-c-d; ARCS: a-b, b-c, c-d

A **CYCLE** is a closed chain:

CYCLE: a-b-c-d; ARCS: a-b, c-b, c-d, d-a



CIRCUIT: a-b-c-d; ARCS: a-b, b-c, c-d, d-a



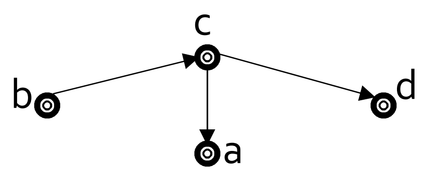
A **CIRCUIT** is a closed path:

A graph is said to be **CONNECTED** if there is a continuous <u>*chain*</u> of edges joining any two vertices.

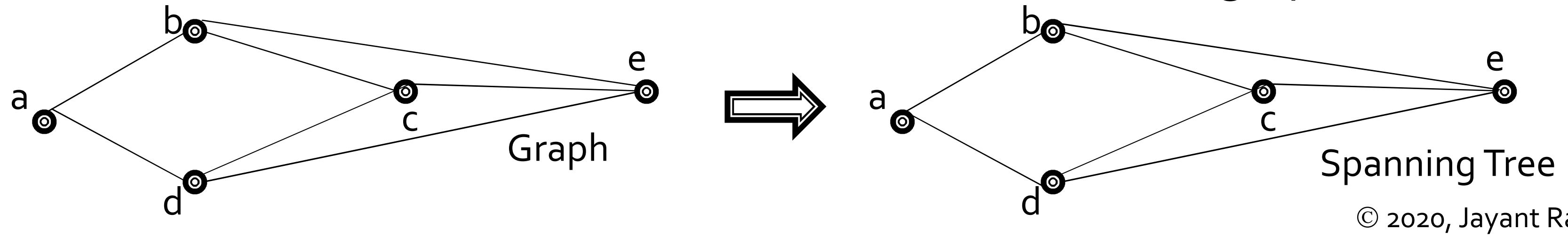
A graph is **STRONGLY CONNECTED** if for all ordered pairs of vertices (*i*,*j*) there is a <u>*path*</u> of arcs from *i* to *j*.

A graph is **COMPLETE** if every node is directly connected to every other node.

A **TREE** is a connected graph with no cycles.



A **SPANNING TREE** is a tree that contains <u>*all*</u> the nodes of a graph.



Minimum Cost Network Flow Problem (*MCNF*)

GIVEN: A digraph $\mathcal{G}$ with a set of *nodes* 1,2,…,*m*, and

- a set of *n* directed *arcs* *i-j* emanating from node *i* and ending in node *j*,
- with each node *i*, a number b_i that is the *supply* (if $b_i>0$) or the *demand* (if $b_i<0$); if $b_i=0$, then the node is called a *transshipment* node,
- with each arc *i-j*, a *flow* of x_{ij} and a unit transportation / movement *cost* of c_{ij}.

ASSUMPTION: $\sum_{i=1..m} b_i = 0$

PROBLEM MCNF:

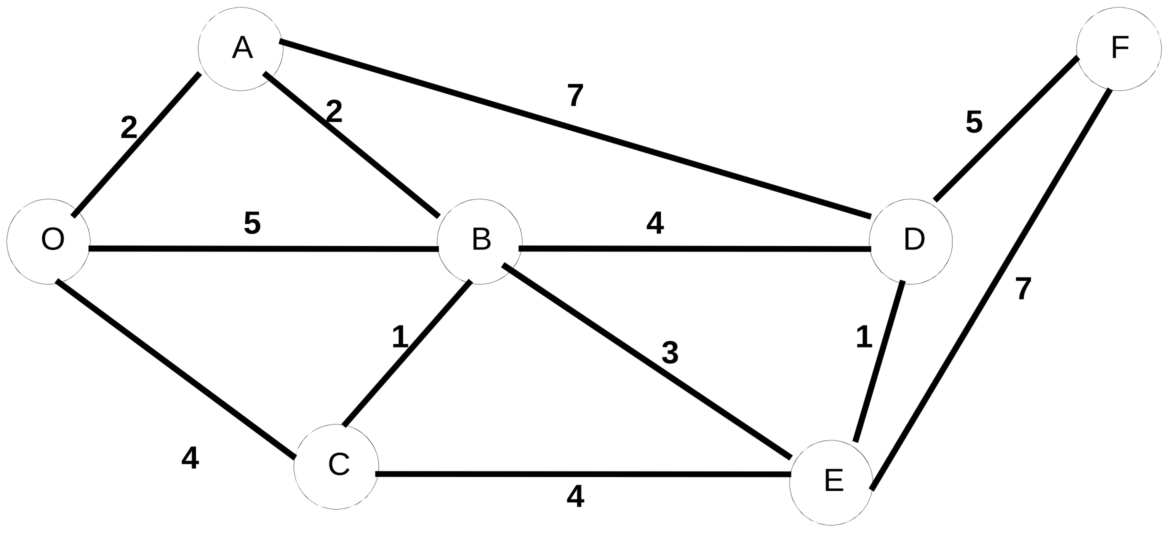
$$\boxed{\begin{array}{ll} \text{Minimize} & \sum_{\text{all defined } i\text{-}j} (c_{ij} x_{ij}) \\ \text{st} & \sum_j x_{ij} - \sum_k x_{ki} = b_i \quad \text{for } i=1,2,\ldots,m \\ & L_{ij} \le x_{ij} \le U_{ij} \quad \text{for all } i\text{-}j \end{array}}$$

The coefficient matrix for the constraints is called the *NODE-ARC INCIDENCE* matrix.

Network Flow Problems

A Prototypical Example

A new amusement park is being constructed by a large corporation. Cars are not allowed into the park, but there is a system of narrow, winding pathways for automated people movers and pedestrians. The road system is shown in the figure below, where O represents the entrance to the park and A-F represent sites of various popular amusements. The numbers above the arcs give the distances of the arcs.



The designers face three questions:

Network Flow Problems (*cont'd*)

1. Electrical lines must be laid under the pathways to establish communication between all nodes in the network. Since this is an expensive process, the question to be answered is how this can be accomplished with a *minimum* total distance ?

2. Location F is extremely popular with visitors, and it is planned that a small number of people movers will run directly from the entrance O to site F. The question is which path will provide the *smallest total distance* from O to F ?

3. During the peak season the number of people wanting to go from O to F is very large. Thus during this time various routes may be followed from O to F (irrespective of distance) so as to accommodate the increased demand through additional trips. However, due to the terrain and the quality of the construction, there are limits on the number of people mover trips that can run on any given path per day; these are different for different paths. The question is how to route various trips to *maximize* the number of trips that can be made per day without violating the limits on any individual path ?

- Question 1 may be answered by solving a ***minimum spanning tree problem.***
- Question 2 may be answered by solving a ***shortest path problem.***
- Question 1 may be answered by solving a ***maximum flow problem.***

DIJKSTRA'S SHORTEST PATH ALGORITHM

Let us represent the origin by **O**, and suppose that there are *n* additional nodes in the network:

Objective at Iteration i: To find the i^{th} closest node from **O**, along with the corresponding path and distance.

Input at Iteration i: The closest, the 2^{nd} closest,…,$(i\text{-}1)^{th}$ closest nodes to **O,** along with their paths and distances. These are designated as **permanent** nodes (***P***) while the remaining nodes are designated as **temporary** nodes (***T***). Initially, ***P***={**O**} and ***T***={all other nodes}.

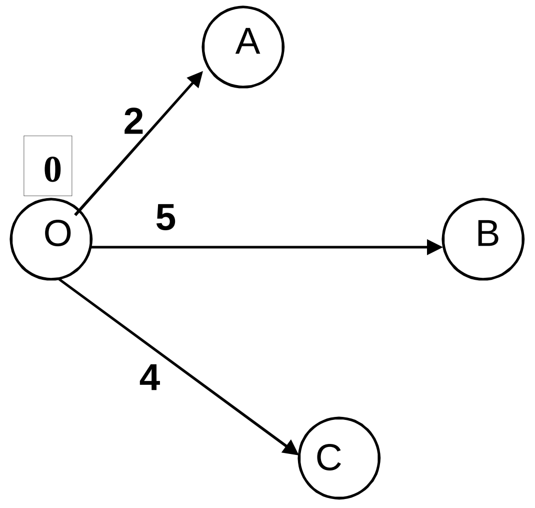
At Iteration i:

- Determine all nodes in ***T*** that are **directly** linked to at least one node in ***P***; call this subset of ***T*** as Ω.
- For each $j \in \Omega$, compute D_j= Minimum {shortest distance from **O** to a permanent node + distance of direct link from that permanent node to *j* }.
- Determine the $j \in \Omega$ that has the minimum value for D_j. Remove *j* from ***T*** and add it to ***P*** along with the shortest path and distance.

At the end of iteration *n* the shortest path to each node is available.

Dijkstra's Method: An Example

Iteration 1



$\mathbf{P}$={**O**}, D_O=0;

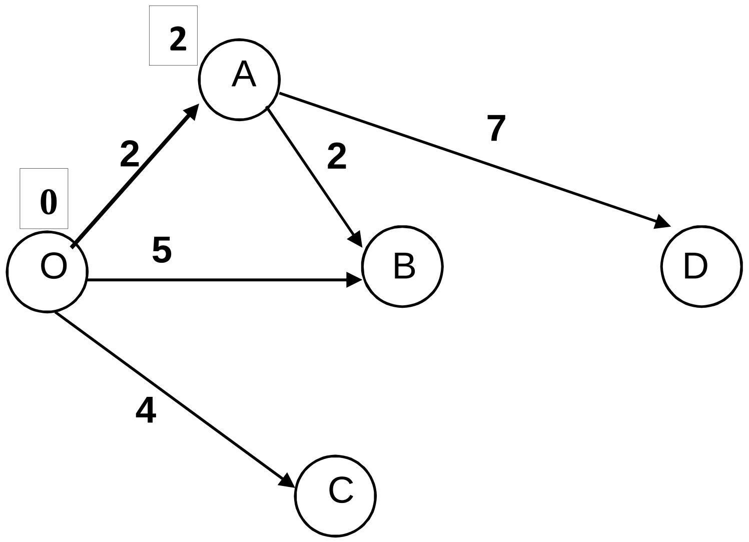
$\boldsymbol{T}$={A,B,C,D,E,F}

Ω= {A, B, C}

$D_A=\text{Min}\{D_O+L_{OA}\}=2 \leftarrow$
$D_B=\text{Min}\{D_O+L_{OB}\}=5$
$D_C=\text{Min}\{D_O+L_{OC}\}=4$

Closest node = {A}, D_A=2

Iteration 2



(breaking the tie arbitrarily…)

P={**O,A**}, $D_O=0$, $D_A=2$;

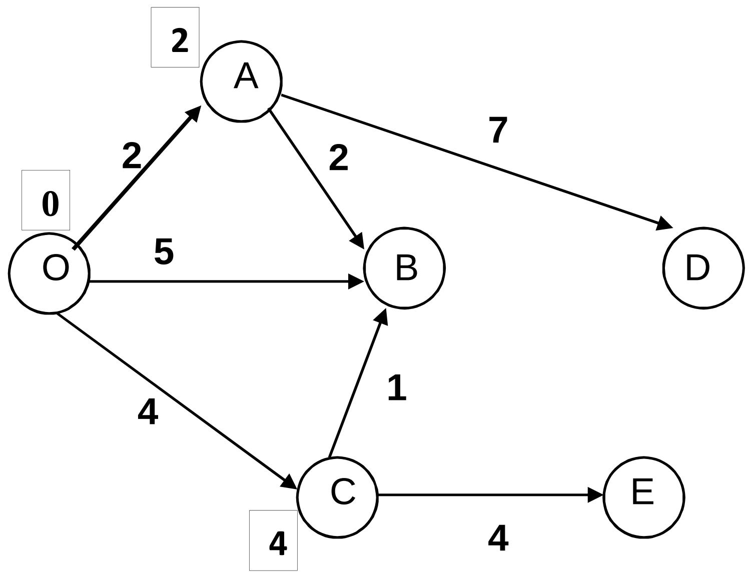
T={B,C,D,E,F}

Ω= {B, C, D}

$D_B = \text{Min}\{D_O+L_{OB}, D_A+L_{AB}\}$
$= \text{Min}\{0+5,\ 2+2\}=4 \leftarrow$
$D_C = \text{Min}\{D_O+L_{OC}\}=0+4=4 \leftarrow$
$D_D = \text{Min}\{D_A+L_{AD}\}=2+7=9$

2nd closest node = {C}, $D_C=4$

Iteration 3



P={**O,A,C**}, D_O=0, D_A=2, D_C=4;

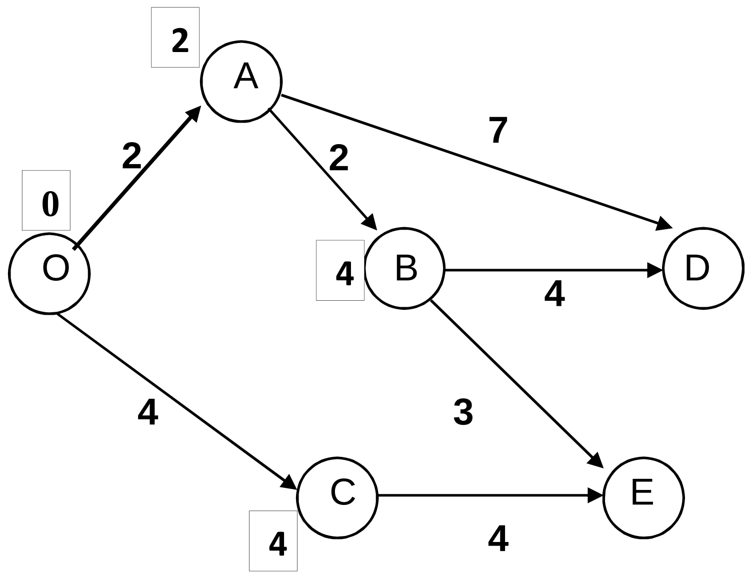
T={B,D,E,F}

Ω= {B, D, E}

D_B =Min{D_O+L_{OB}, D_A+L_{AB}, D_C+L_{CB}}
=Min{0+5, 2+2, 4+1}=4 $\leftarrow$
D_D =Min{D_A+L_{AD}}=2+7=9
D_E =Min{D_C+L_{CE}}=4+4=8

3rd closest node = {B}, D_B=4

Iteration 4



P={O,A,C,B}, D_O=0, D_A=2, D_C=4, D_B=4;

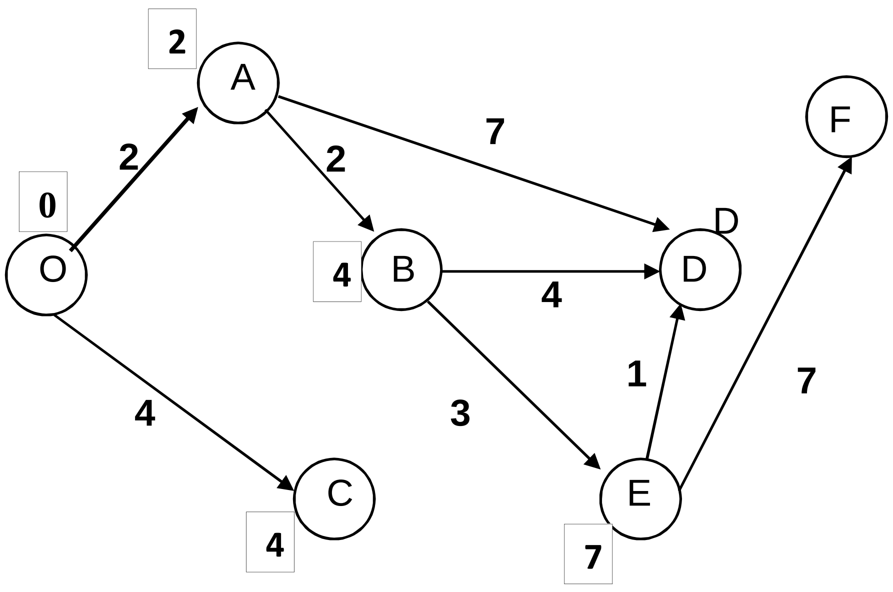
T={D,E,F}

Ω = {D, E}

D_D=Min{D_A+L_{AD}, D_B+L_{BD}}
=Min{2+7, 4+4}=8
D_E =Min{D_C+L_{CE}, D_B+L_{BE}}
=Min{4+4, 4+3}=7 ←

4th closest node = {E}, D_E=7

Iteration 5



P={O,A,C,B,E}, D_O=0, D_A=2, D_C=4, D_B=4, D_E=7;

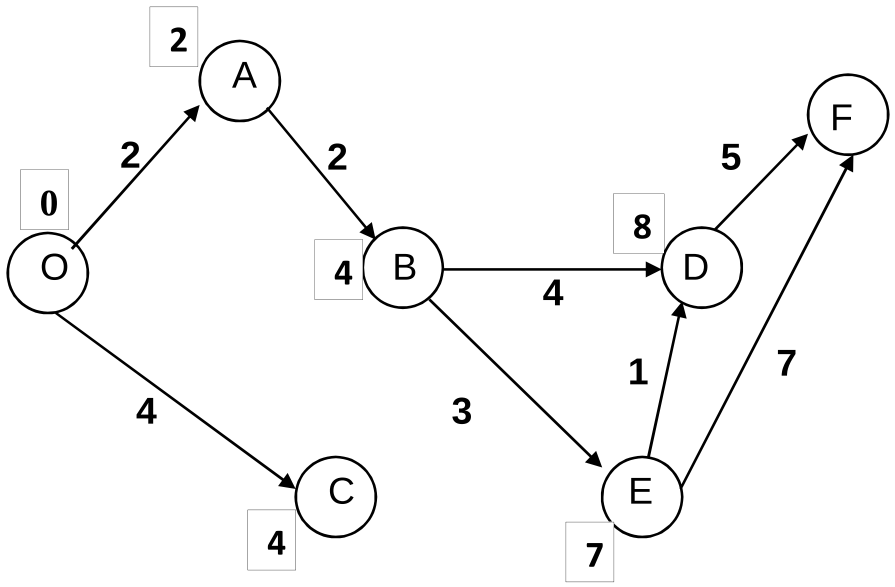
T={D,F}

Ω = {D, F}

D_D=Min{D_A+L_{AD}, D_B+L_{BD}, D_E+L_{ED}}
=Min{2+7, 4+4, 7+1}=8 ←
D_F =Min{D_E+L_{EF}}=7+7=14

5th closest node = {D}, D_D=8

Iteration 6



P={O,A,C,B,E,D}, D_O=0, D_A=2, D_C=4, D_B=4, D_E=7, D_D=8;

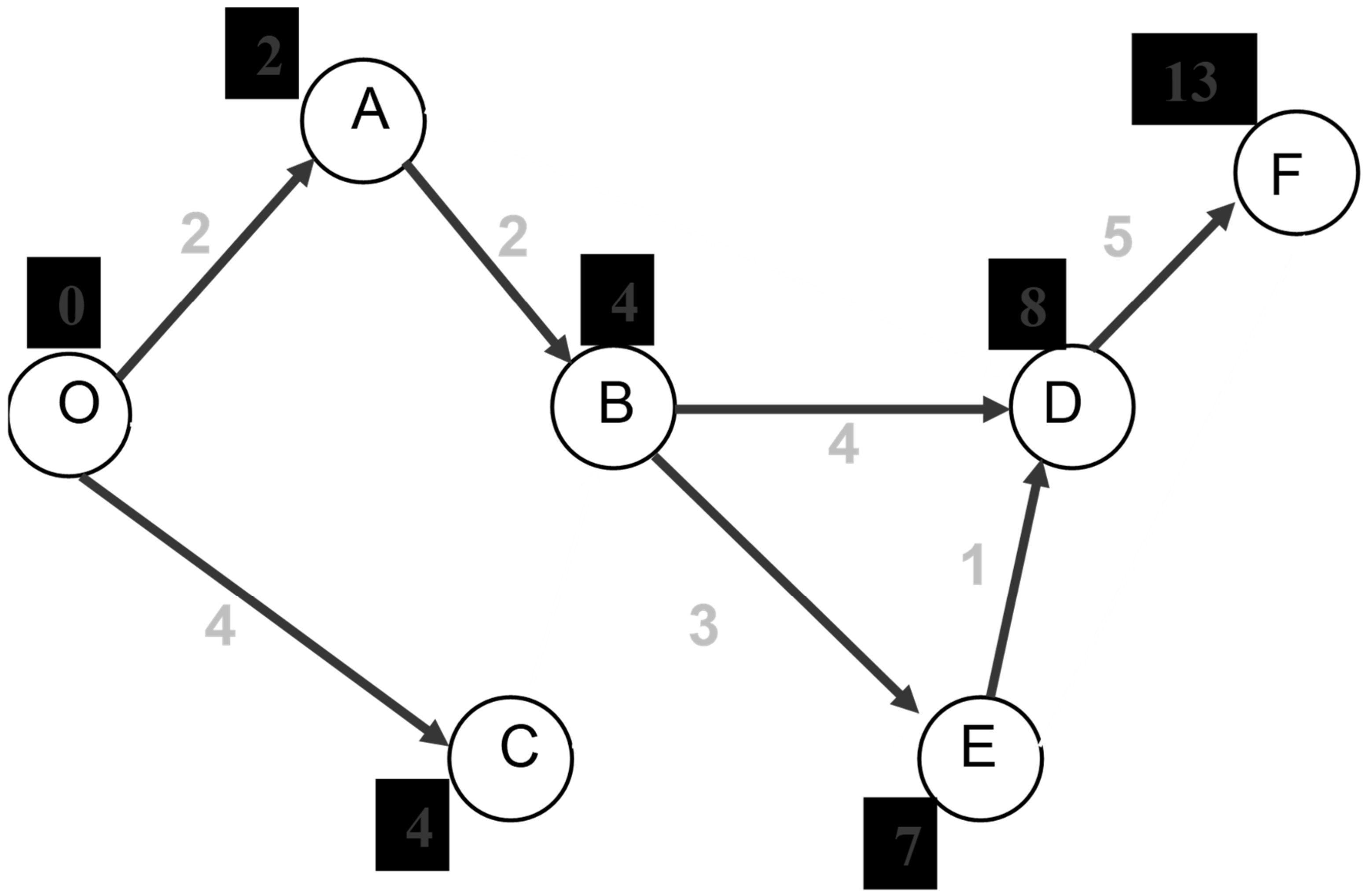
T={F}

Ω = {F}

D_F=Min{D_D+L_{DF}, D_E+L_{EF}}
=Min{8+5, 7+7}=13 ←

6th closest node = {F}, D_F=13

Final set of shortest paths



THE MAXIMUM FLOW PROBLEM : FORD-FULKERSON ALGORITHM

STEP 0: Start with some feasible flow (if one isn't obvious let flow in each arc be equal to zero) from the source node to the sink node.

STEP 1:
If flow in arc (*i-j*) is **less than** capacity of arc (*i-j*) assign it to set **I** (set of arcs along which amount of flow can be **I**ncreased).
If flow in arc (*i-j*) is **greater than zero** assign it to set **R** (set of arcs along which amount of flow can be **R**educed).

NOTE: An arc can belong to both **I** and **R**!

STEP 2: LABELING PROCEDURE

- Label the source node.
- If the flow along an arc (*i-j*) is **from a labeled** node **to an unlabeled** node and the arc belongs to **I**, then label the unlabeled node, call the arc a **forward** arc and let *I*(*i-j*) = Capacity(*i-j*) - Flow(*i-j*)
- If the flow along an arc is **from an unlabeled** node **to a labeled** node and the arc belongs to **R**, then label the unlabeled node and call the arc a **backward** arc and let *R*(*i-j*) = Flow(*i-j*)

FORD-FULKERSON ALGORITHM

Continue the labeling procedure until (a) the sink has been labeled, or (b) no more nodes can be labeled.

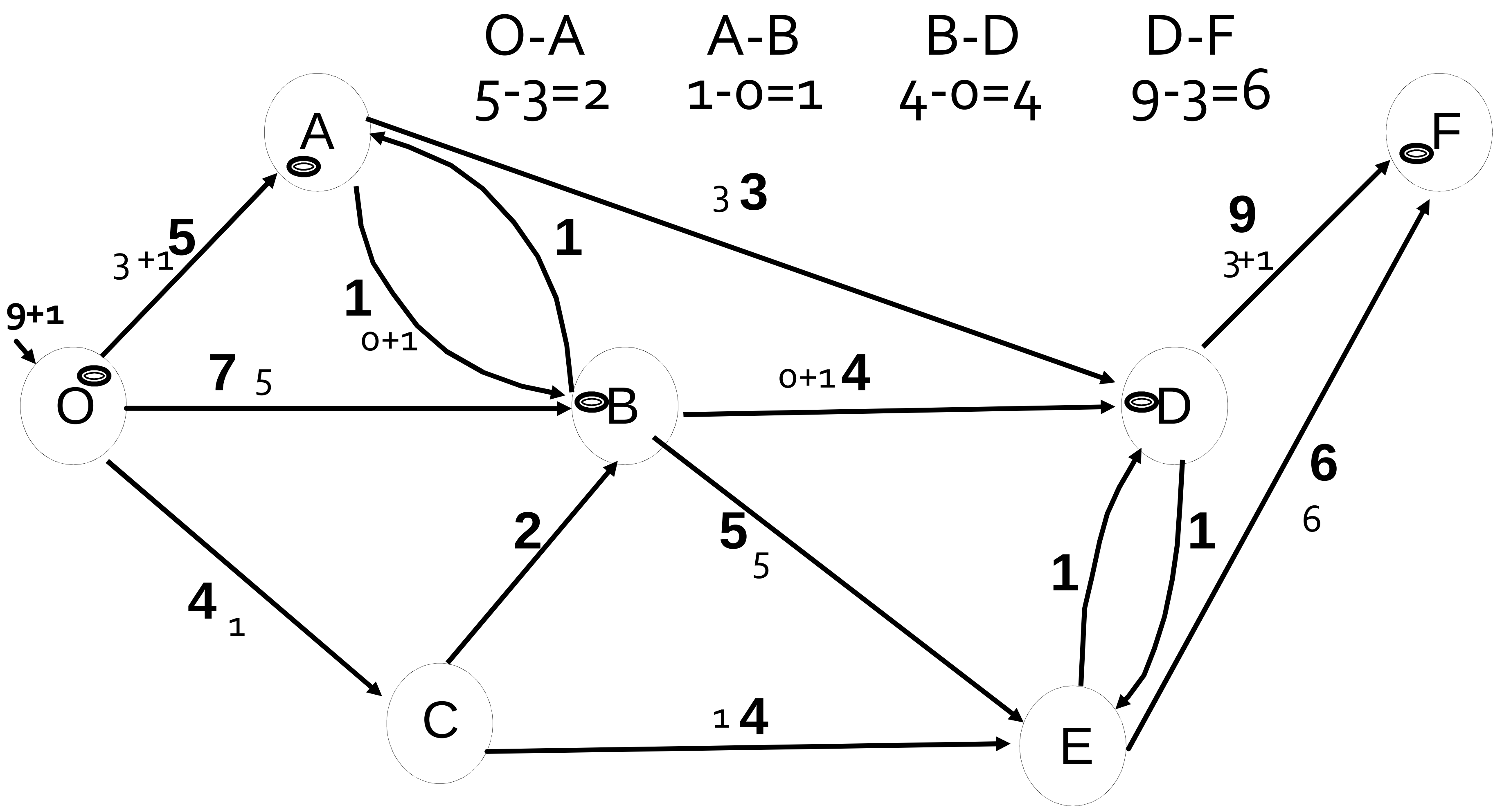
STEP 3: If the sink has not been labeled, STOP; this is the optimum flow. If the sink has been labeled go to Step 4.

STEP 4: There is a chain C from source to sink.

- If C has only forward arcs, increase the flow in each arc by an amount $\Delta f = Min_{(i\text{-}j)\in C} I(i\text{-}j)$
- If C has forward and backward arcs, increase the flow in each forward and decrease the flow in each backward arc by an amount $\Delta f = \text{MIN}\ \{k_1 = Min_{(i\text{-}j)\in I\cap C} I(i\text{-}j),\ k_2 = Min_{(i\text{-}j)\in R\cap C}\{R(i\text{-}j)\}$

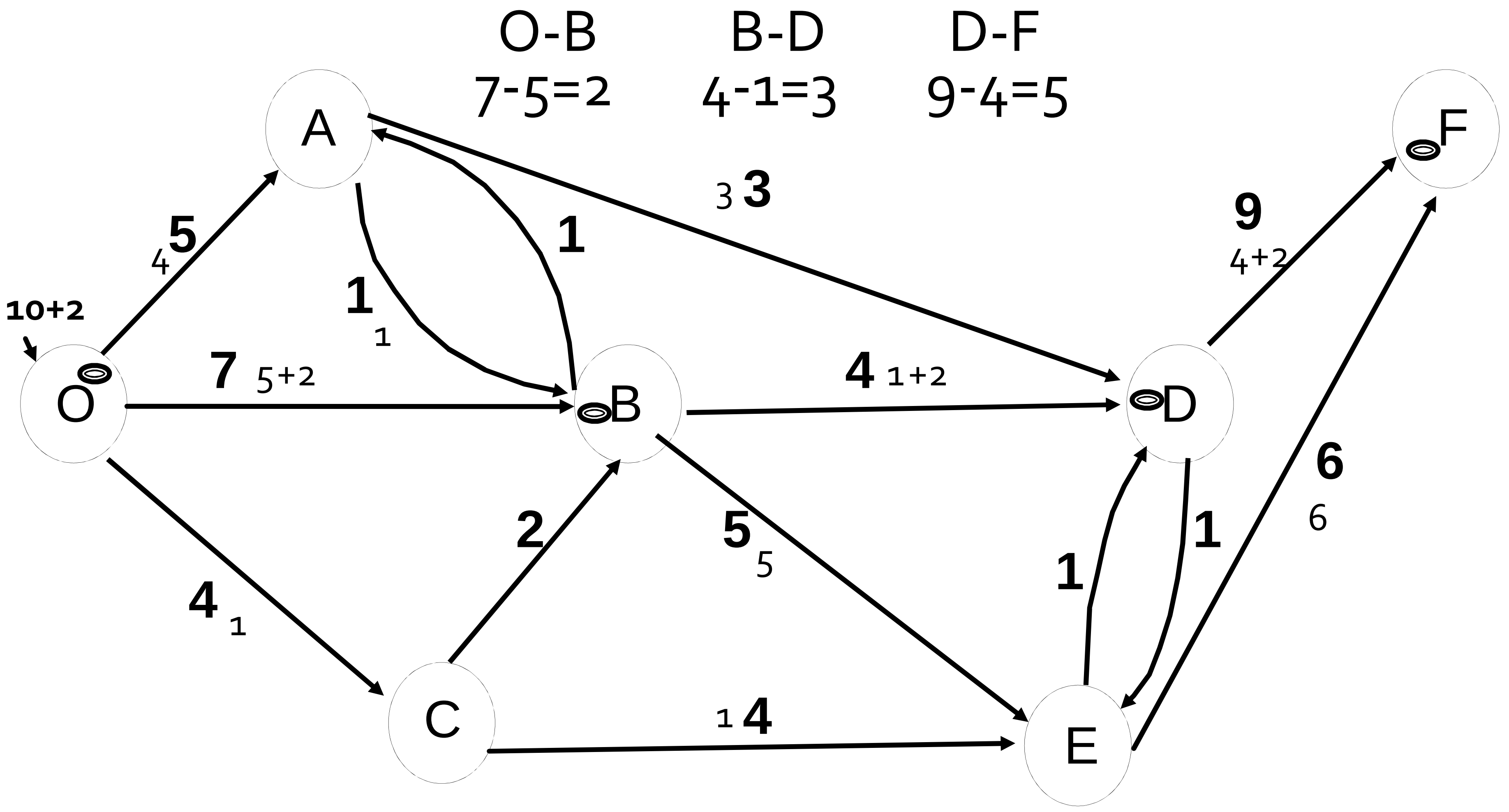
Return to Step 1

Example

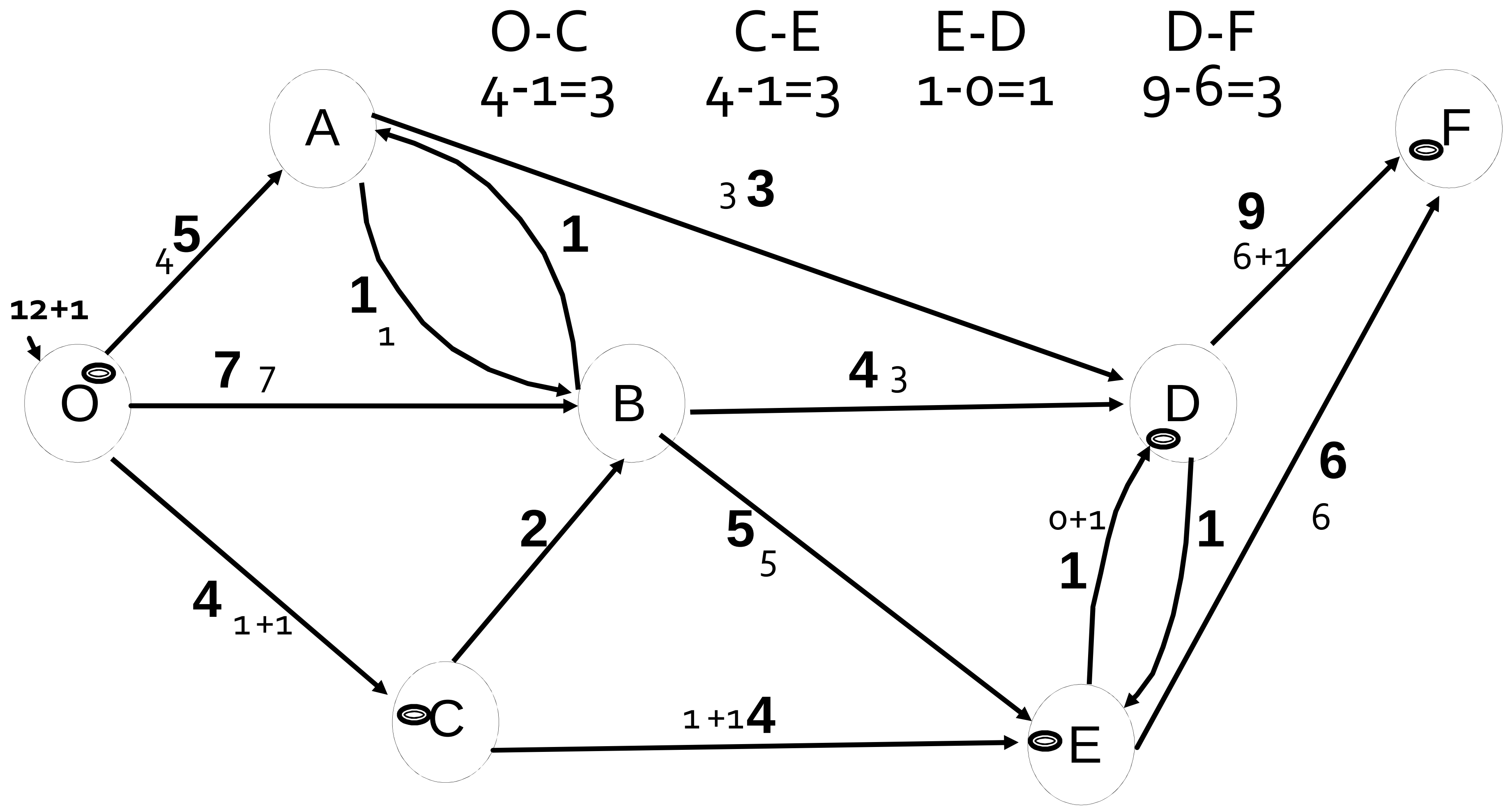


© 2020, Jayant Rajgopal

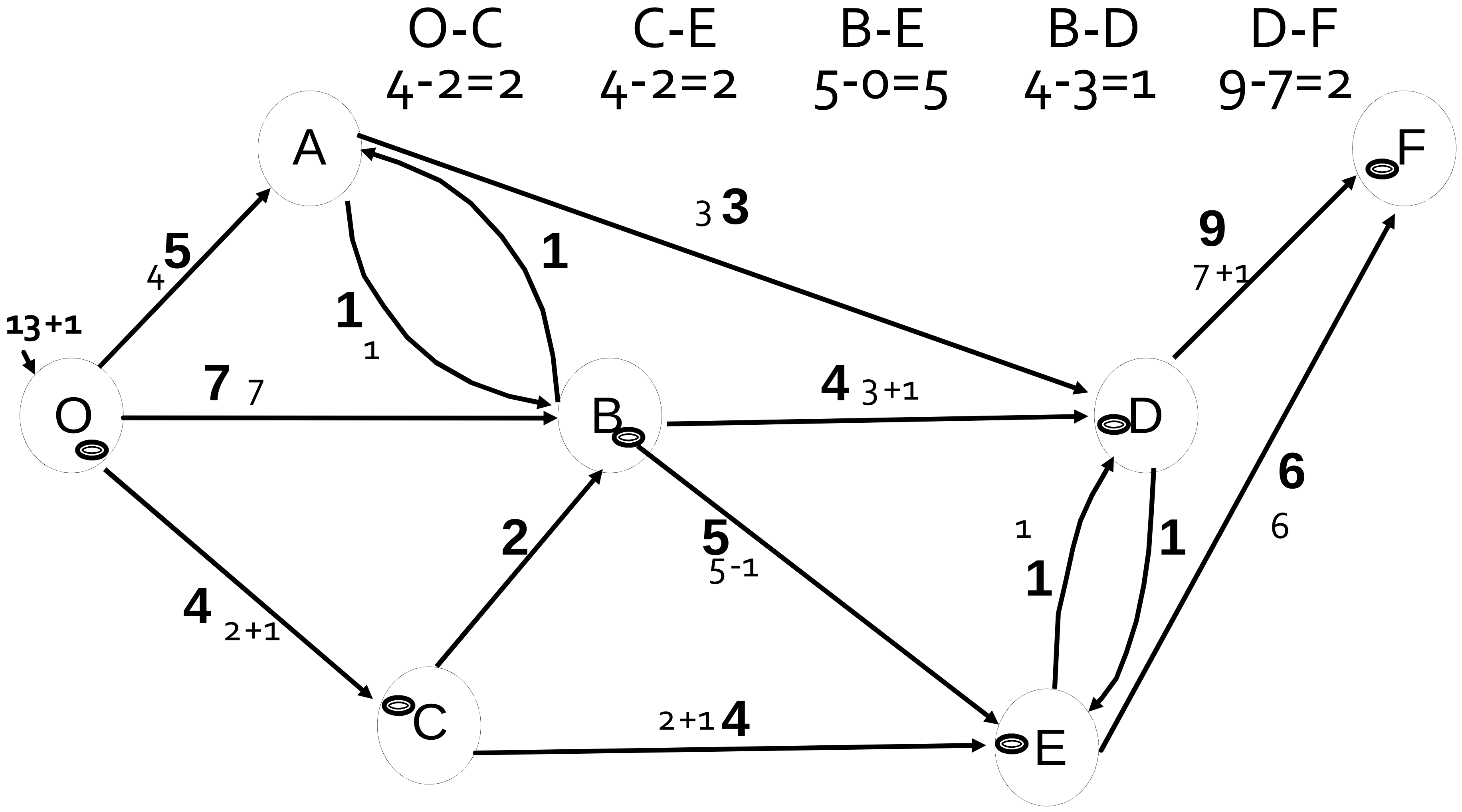
Example (cont'd)



Example (cont'd)

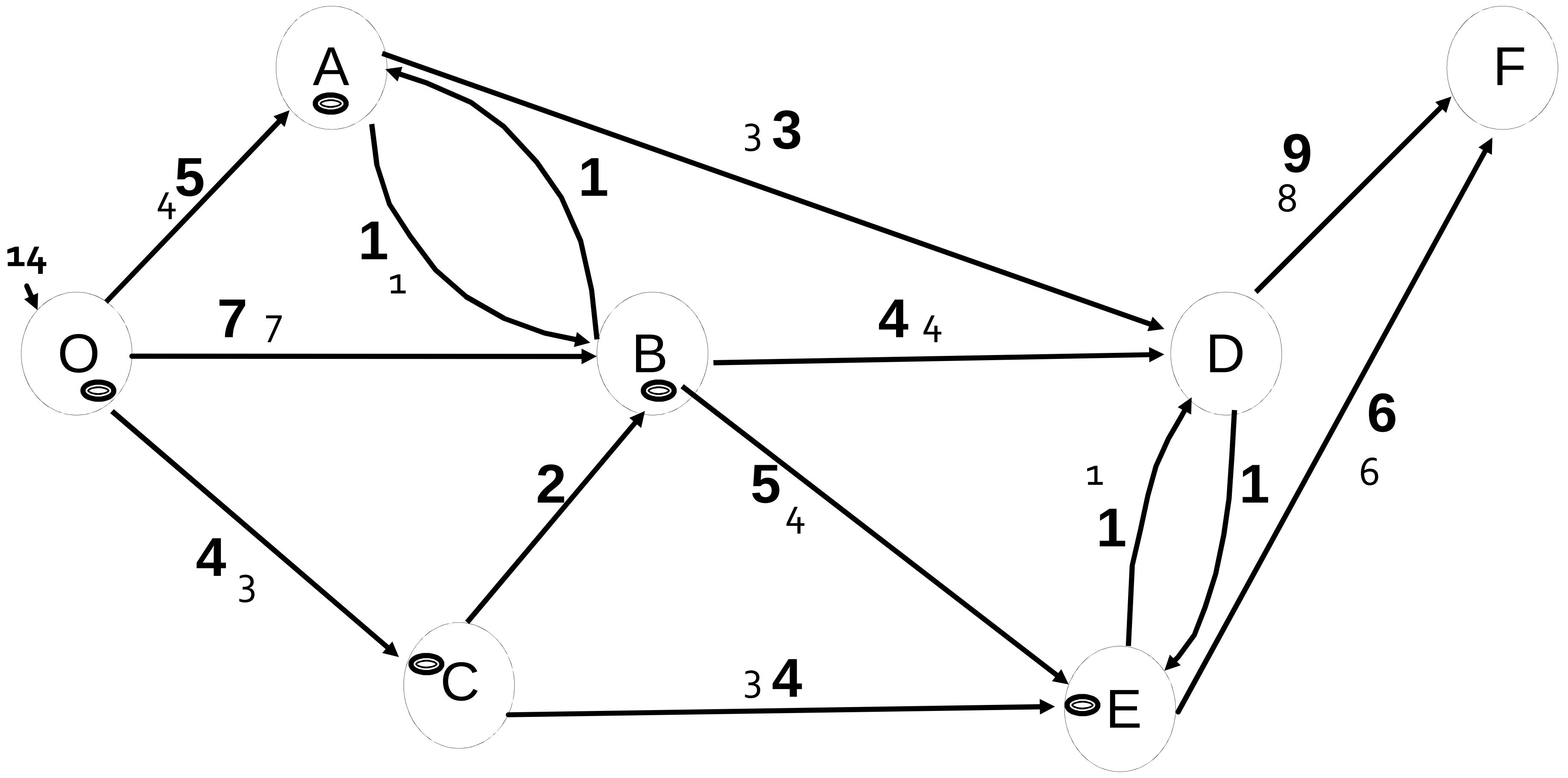


Example (cont'd)



Example (cont'd)

can't go further---OPTIMAL FLOW!

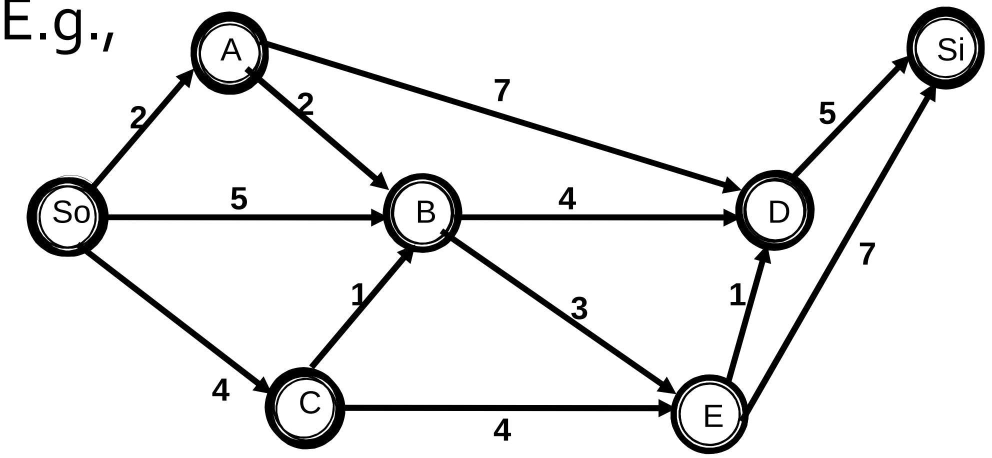


VERIFYING AN OPTIMAL FLOW

<u>Cut Set</u>: Suppose V_1 is any subset of all the nodes that <u>contains the sink but not the source</u>, and V_2 is the set of remaining nodes. Then the corresponding cut set is the set of all arcs (*i*-*j*) such that $i \in V_2$ and $j \in V_1$.

The capacity of the cut is the sum of the capacities of all arcs in the cut set.

E.g.,



V1={B, D, Si}
V2={So, A, C, E}
Cut Set={So-B,A-B,A-D,C-B,E-D,E-Si}
Capacity=5+2+7+1+1+7 =23

V1={C, D, E, Si}
V2={So, A, B}
Cut Set={So-C,A-D,B-D,B-E}
Capacity=4+7+4+3=18

VERIFYING OPTIMALITY (cont'd)

Lemma 1: The total flow from source to sink in any feasible flow is no higher than the capacity of ***any*** cut set.

In particular, the ***optimal*** flow (which is obviously also a feasible flow) can thus never be higher than the capacity of ***any*** cut set.

So, if we find ***some*** feasible flow and ***some*** cut set for which the flow is **equal** to the capacity of the cut set, then this **must** be the optimal flow!

Now, suppose we are doing our labeling and we get to a point where we're unable to label the sink. Let V_1 here correspond to the nodes (including the sink…) that are *not* labeled and V_2 to the nodes that *have* been labeled, and let us denote the corresponding cut set via *C.*

Lemma 2: If the sink cannot be labeled, then the capacity of the cut set *C* must be equal to the current flow from source to sink.

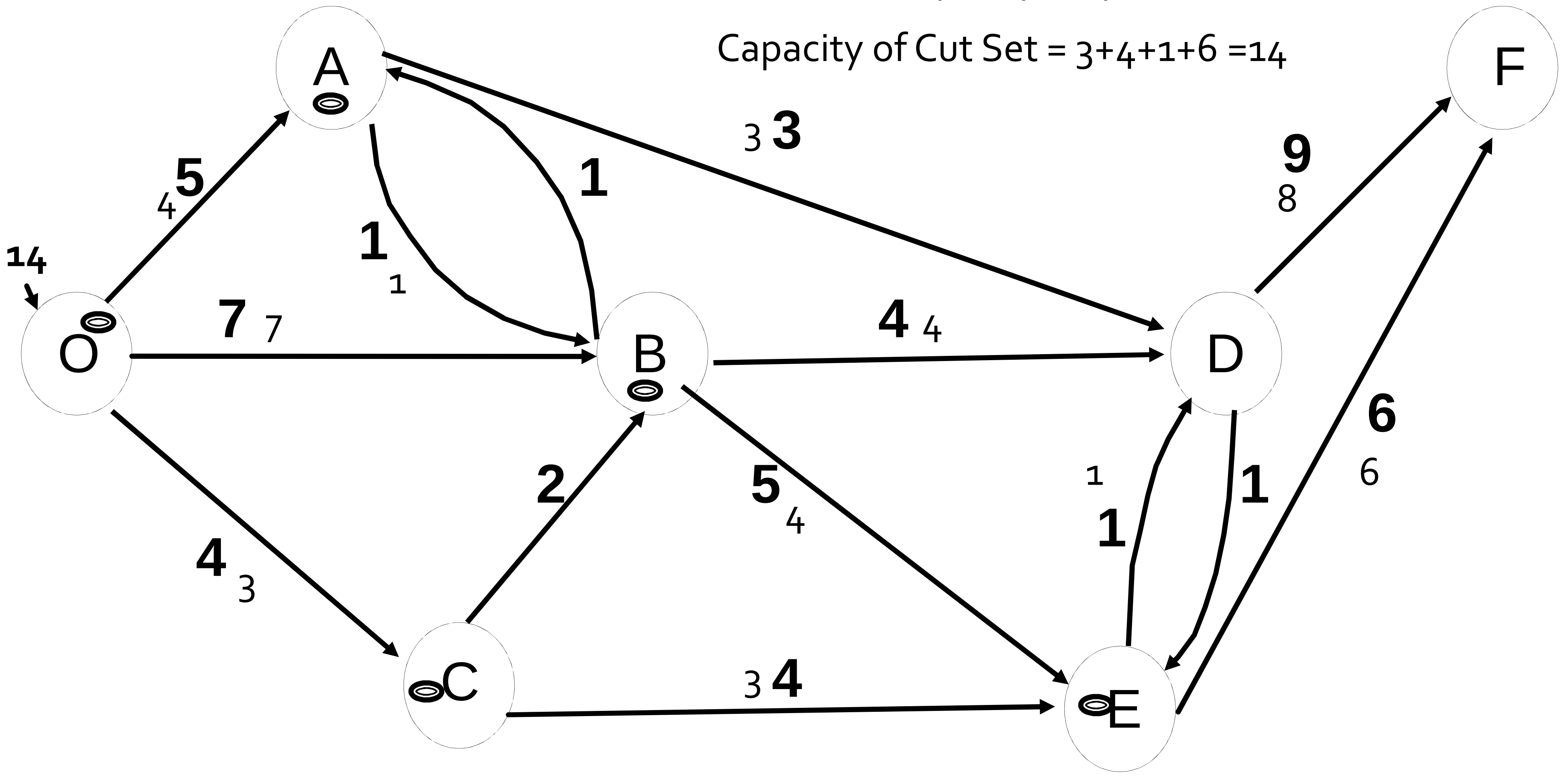
So, if the sink cannot be labeled, the current flow must be optimal –**we just use Lemma 2 to verify that the sink cannot be labeled.**

Verifying Optimality

V_2 = {O, A, B, C, E}; V_1 = {D,F}

Cut Set: {A-D, B-D, E-D, E-F}

Capacity of Cut Set = 3+4+1+6 =14



Network Flow Problems as Special Cases of the MCNFP

- Recall the MCNFP
- Assuming that $\sum_{i=1..m} b_i = 0$

PROBLEM MCNF:

Minimize $\sum_{\text{all defined } i\text{-}j} (c_{ij} x_{ij})$

st $\sum_j x_{ij} - \sum_k x_{ki} = b_i$ for $i=1,2,\ldots,m$

$L_{ij} \leq x_{ij} \leq U_{ij}$ for all i-j

Transshipment Problem(Balanced)

Minimize $\sum_i \sum_j (c_{ij} x_{ij})$

st $\sum_j x_{ij} = S_i$ if $i \equiv$ pure supply nodes

$-\sum_j x_{ij} = -D_j$ if $i \equiv$ pure demand nodes

$\sum_j x_{ij} - \sum_k x_{ki} = 0$ or S_i or D_i if $i \equiv$ transshipment node

(*as the case may be…*)

$0 \leq x_{ij}$ for all i-j

Shortest Path Problem

- Define the starting node as a source node with a supply of 1 unit and the ending node (say *m*) as a sink node with a demand of 1 unit.
- All other nodes are transshipment nodes
- The cost associated with a node is its distance

Then we have

Minimize $\sum_{\text{all defined } i\text{-}j} (c_{ij} x_{ij})$

st $\sum_j x_{1j} = 1$

$\sum_j x_{ij} - \sum_k x_{ki} = 0$ for $i=2,\ldots,m-1$

$- \sum_k x_{km} = -1$

$0 \leq x_{ij}$ for all $i\text{-}j$

Max Flow Problem

- Let the source node have a very large supply (*M*) and the sink node have a very large demand equal to the same value *M*
- All other nodes are transshipment nodes
- The cost associated with all arcs are equal to zero, and the upper bound U_{ij} for the flow along arc *i-j* is set to the capacity of the arc (C_{ij})
- Finally, add a fictitious arc from the source to the sink node with $U_{so\text{-}si}=M$ and a cost of 1

Then we have (assume that source node is node no. 1, and the sink node is node no. *m*)

Minimize $x_{so\text{-}si}$

st $\sum_j x_{so\text{-}j} = M$

$\sum_j x_{ij} - \sum_k x_{ki} = 0$ for $i=2,\ldots,m-1$

$- \sum_k x_{k\text{-}si} = -M$

$0 \le x_{ij} \le U_{ij}$ for all $i\text{-}j$