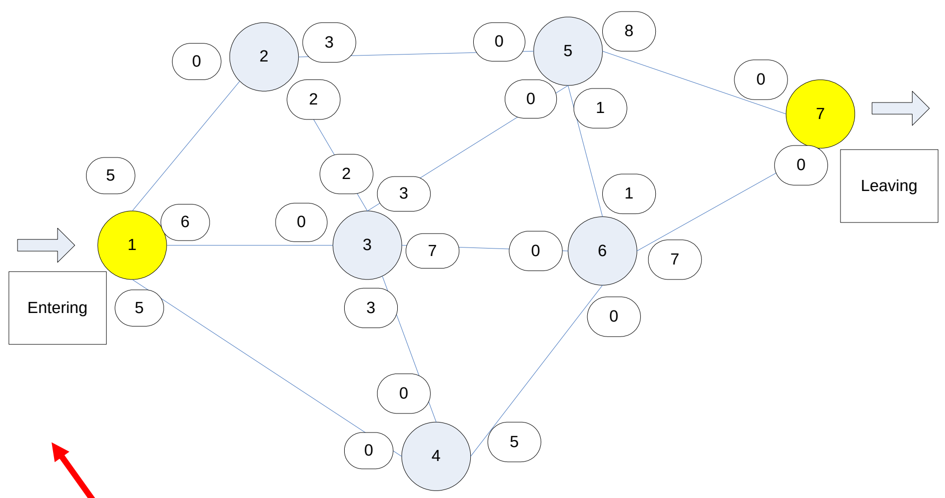
THE MAXIMAL FLOW PROBLEM

Consider a network with one input or source node and one output or sink node.
The maximal flow problem asks,
What is the maximum amount of flow (that is messages, packets, vehicles fluid, and so on) that can enter and exit from the network system in a *given period of time*?

In this problem we attempt to transmit flow through all branches (arcs) of the network as efficiently as possible. **The amount of flow is limited due to capacity restrictions on the various branches of the network**
(Highway types limit vehicle flow in a transportation system, pipe size limit oil flow in an oil distribution system.)
The maximum or upper limit on the flow in a branch is referred to as the *flow capacity of the branch*. While we do not specify capacities for the nodes, we do assume that the *flow out of a node is equal to the flow into the node*.

Example:



Flow Capacity 5000 packages (messages, vehicles) per hour from Node 1 to Node 4.

Since the maximum flow problem is a linear programming problem, it could be solved by the simplex algorithm for linear programming problems, However, using he simplex algorithm for this problem is like killing a mouse with a cannon.

Model:

Max Flow

s.t. input (source) node

node-1 $x_{12} + x_{13} + x_{14} - \text{Flow} = 0$

Intermediate nodes

node-2 $x_{12} + x_{32} - x_{23} - x_{25} = 0$

node-3 $x_{13} + x_{23} - x_{32} - x_{34} - x_{35} - x_{36} = 0$

node-4 $x_{14} - x_{34} - x_{46} = 0$

node-5 $x_{25} - x_{35} + x_{65} - x_{56} - x_{57} = 0$

node-6 $x_{36} + x_{46} + x_{56} - x_{65} - x_{67} = 0$

Output (sink) node

node-7 $x_{57} + x_{67} - \text{Flow} = 0$

Capacity Constraints

$x_{12} \le 5$, $x_{13} \le 6$, $x_{14} \le 5$, $x_{23} \le 2$, $x_{25} \le 3$, $x_{32} \le 2$,

$x_{35} \le 3$, $x_{34} \le 3$, $x_{36} \le 7$, $x_{46} \le 5$, $x_{56} \le 1$, $x_{58} \le 8$,

$x_{65} \le 1$, $x_{67} \le 7$,

$x_{ij} \ge 0$

The Maximal Flow Algorithm

Step 1.

Find any path from the source node to the sink node that has flow capacities in the direction of the flow greater than zero for all branches on the path. If no path is available, the optimal solution has been reached.

Step 2.

Find the smallest branch capacity, F_j, on the path selected in step 1. Increase the flow through the network by sending an amount F_j over the path selected in step 1.

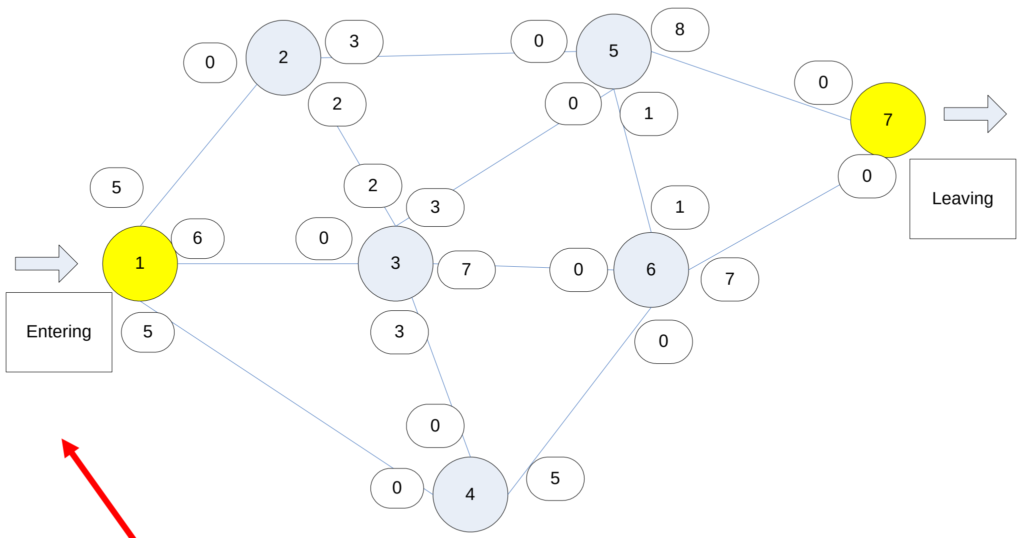
Step 3.

For the path selected in step 1, reduce all branch flow capacities in the direction of flow by F_j and increase all branch flow capacities in the reverse direction by F_j.

GO TO STEP 1.

While the procedure will vary depending on the analyst's choice of paths in step 1, the algorithm will eventually provide the maximum flow solution.

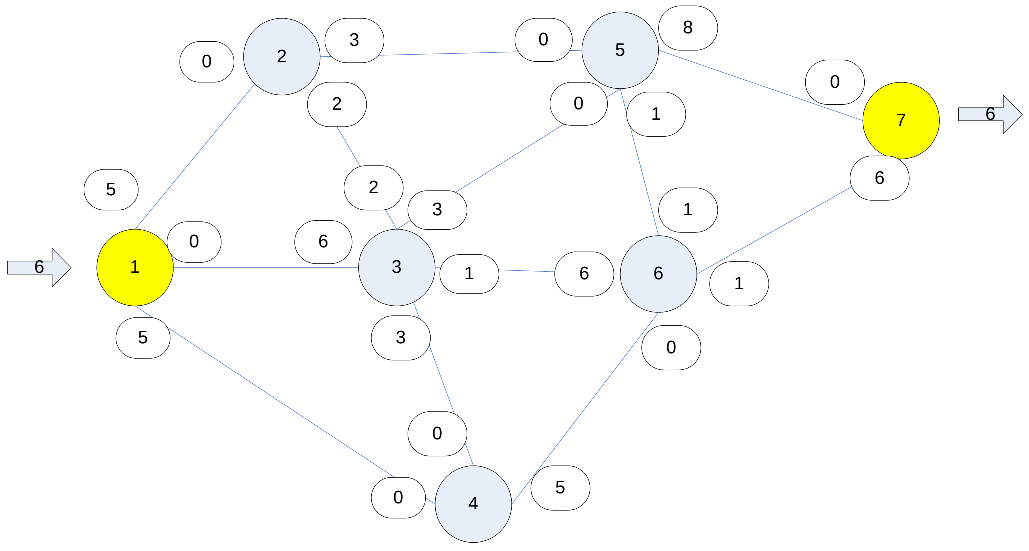
Example:



Flow Capacity 5000 packages (messages, vehicles) per hour from Node 1 to Node 4.

Iteration 1

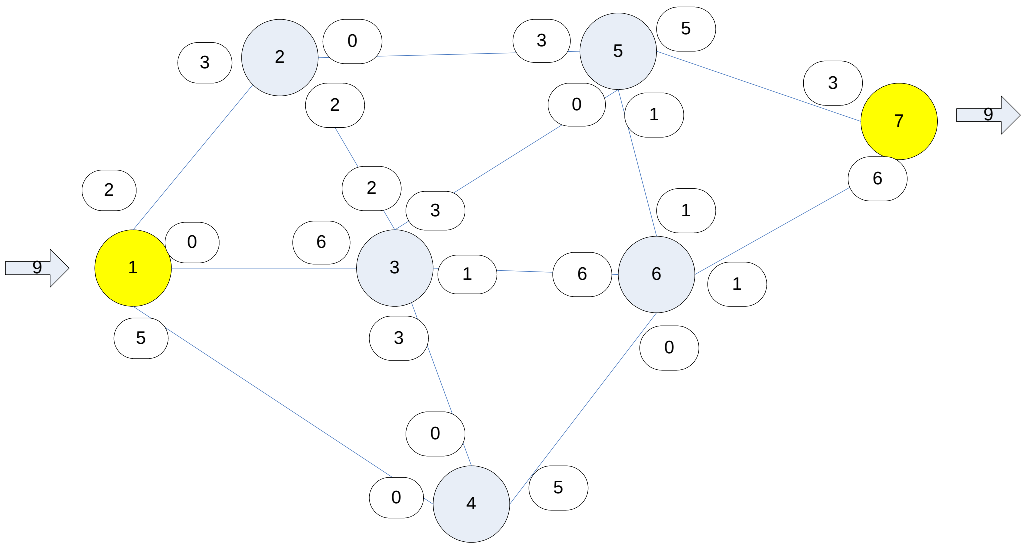
The path selected is 1 – 3 – 6 – 7; F_j determined by branch 1 – 3, is 6. The revised network is as follows:



Total flow after iteration 1 is 6.

Iteration 2

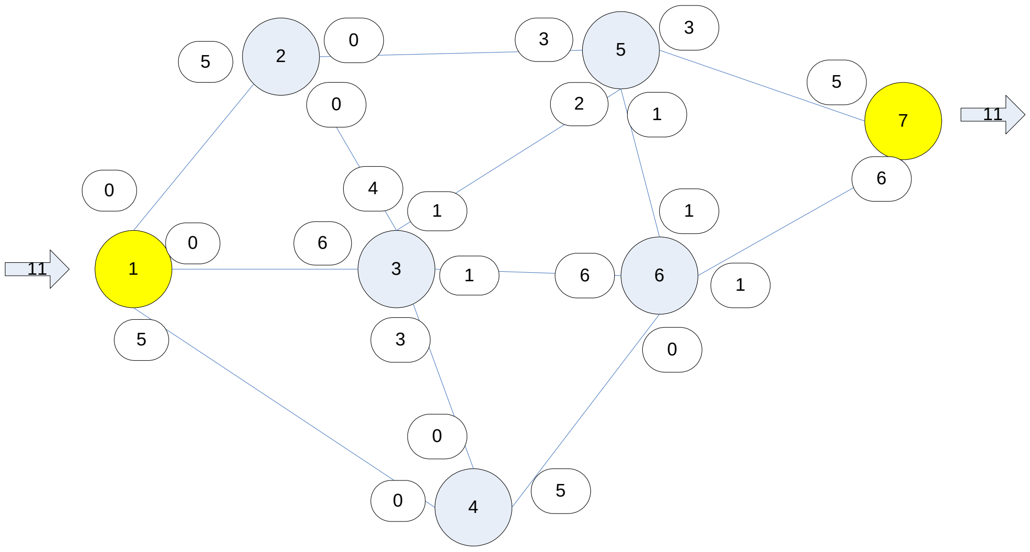
The path selected is 1 – 2 – 5 – 7; F_j determined by branch 2 – 5 , is 3. The revised network is as follows:



Total flow after iteration 2 is 9.

Iteration 3

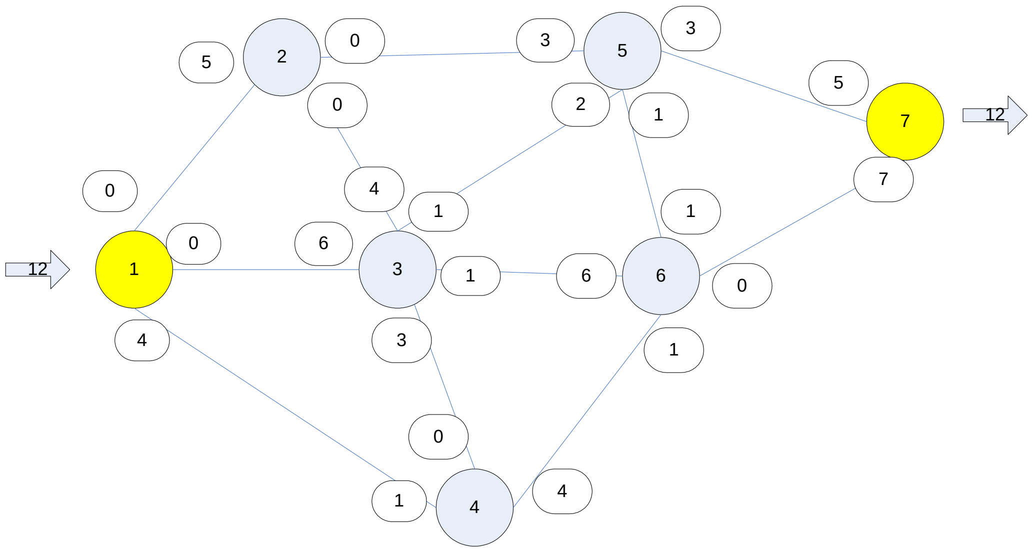
The path selected is 1 - 2 - 3 - 5 - 7; F_j determined by branch 1-2 or (2- 3), is 2. The revised network is as follows:



Total flow after iteration 3 is 11.

Iteration 4

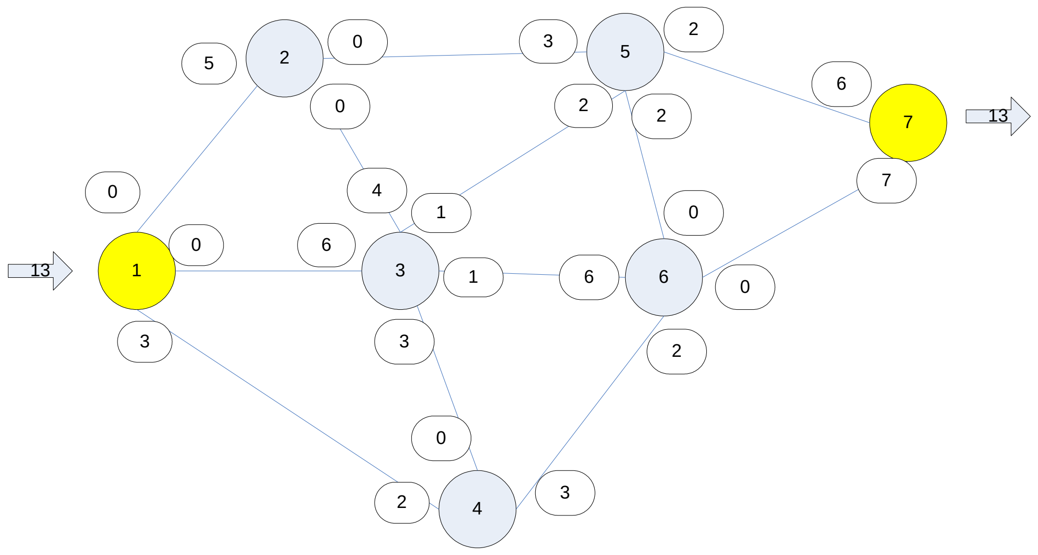
The path selected is 1 - 4 - 6 - 7; F_j determined by branch 6 - 7, is 1. The revised network is as follows:



Total flow after iteration 4 is 12.

Iteration 5

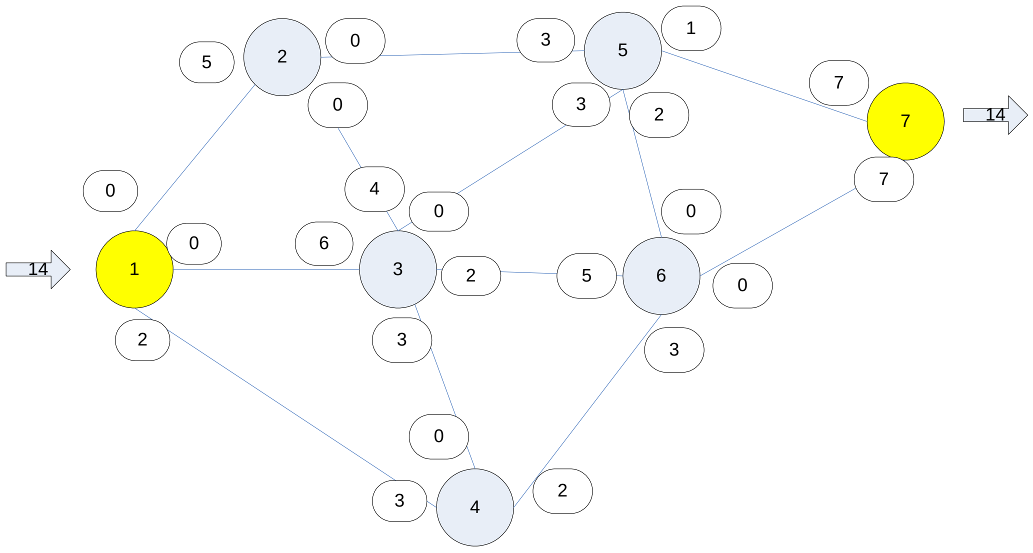
The path selected is 1 - 4 - 6 - 5 - 7; F_j determined by branch 6 - 5, is 1. The revised network is as follows:



Total flow after iteration 5 is 13.

Iteration 6

The path selected is 1 - 4 - 6 - 3 - 5 - 7; F_j determined by branch 3 - 5, is 1. The revised network is as follows:



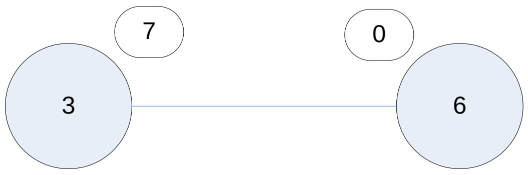
Total flow after iteration 6 is 14.

Let us now determine the amount and direction of flow in each branch so that the total flow of 14000 packages per hour can be attained.

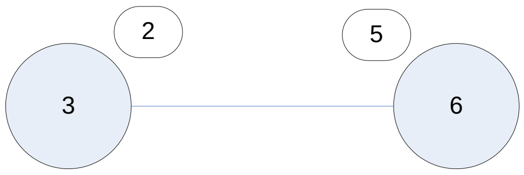
Branch flows for the maximal flows solution can be found by comparing the final branch flow capacities with the initial branch flow capacities. If the final flow capacity is *less* than the initial flow capacity, flow is occurring in the branch with an amount equal to the *difference* between the initial and final flow capacities.

For example, consider the 3 - 6 branch with initial flow capacities shown below.

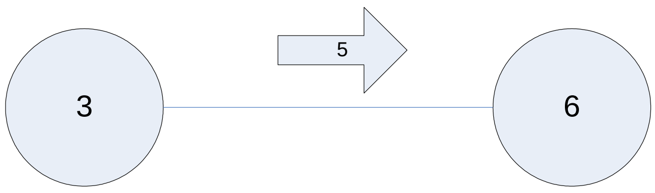
Initial capacities



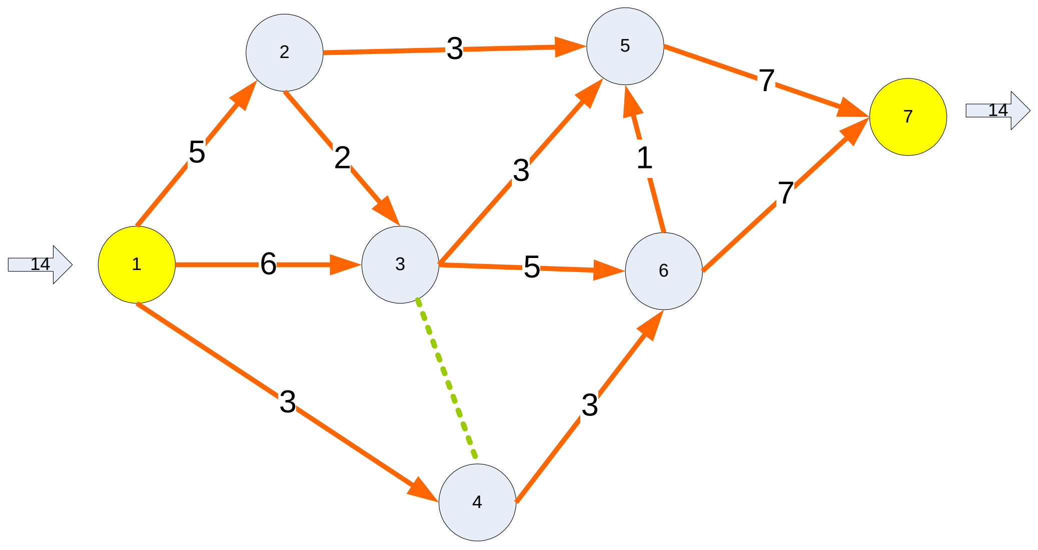
Final capacities



Since the final capacity in the 3 - 6 direction is less than the initial flow capacity, the branch flow is summarized as follows:



Comparing final and initial branch flow capacities for all branches in the network enables us to determine the final flow pattern as shown in the following figure.



If the network is extended or modified, another maximal flow analysis will determine the extent of any improved flow.

Related Problems:

- **Minimum Cost Network Flow Algorithm**
- **Dynamic Flow Algorithm**
- **Flows with Gains**

Minimum Cost Network Flow Algorithm

The transportation, assignment, transshipment, shortest path, maximum flow problems are all special cases of the minimum cost network flow problem (MCNFP). Any minimum cost network flow problem can be solved by a generalization of the transportation simplex called the network simplex.

To define an MCNFP, let

x_{ij} = number of units of flow sent from node i to node j through arc (i , j)
b_i = net supply (outflow – inflow) at node i.

c_{ij} = cost of transporting 1 unit of flow from node i to node j via arc (i , j)
L_{ij} = Lower bound on flow through arc (i , j)

(If there is no lower bound, let L_{ij} = 0)
U_{ij} = Upper bound on flow through arc (i,j)
(If there is no upper bound, let U_{ij} = 0)

The MCNFP may be written as

$$\min \sum_{allarcs} c_{ij}x_{ij}$$

$$s.t \sum_j x_{ij} - \sum_k x_{ki} = b_i \quad \text{(for each node i in the network)}$$

$$L_{ij} \le x_{ij} \le U_{ij} \quad \text{(for each arc in the network)}$$