The *Max flow* problem formulated as a *Linear Program*

- **Using the *Simplex Method* to solve *Max Flow* problems**
    - **Fact:**
        - A **network flow problem** can be **easily** formulated as a **Linear Optimization problem (LP)**

      Therefore:
        - One **can** use the **Simpelx Method** to solve a **maximum network flow problem**

    - ***Network* Simplex Algorithm**:
        - The **Linear Program (LP)** that is **derived** from a **maximum network flow** problem has a ***large number* of constraints**

        - There is a **"*Network*" Simplex Method** developed just for **solving maximum network flow** problems

          We will **not** cover this algorithm

          If you want to learn more: **click here**

          Look for: "A polynomial time primal network simplex algorithm for minimum cost flows". Mathematical Programming 78: pages 109-129

- **Formulating a max flow problem as an *LP***
    - **Recall** the general form of a **Linear Program**:

$$
\begin{array}{ll}
\texttt{max:} & c_1x_1 + c_2x_2 + \ldots + + c_nx_n \\
 & \\
\texttt{s.t.:} & a_{11}x_1 + a_{12}x_2 + \ldots + + a_{1n}x_n \le b_1 \\
 & a_{21}x_1 + a_{22}x_2 + \ldots + + a_{2n}x_n \le b_2 \\
 & \ldots \\
 & a_{m1}x_1 + a_{m2}x_2 + \ldots + + a_{mn}x_n \le b_m
\end{array}
$$

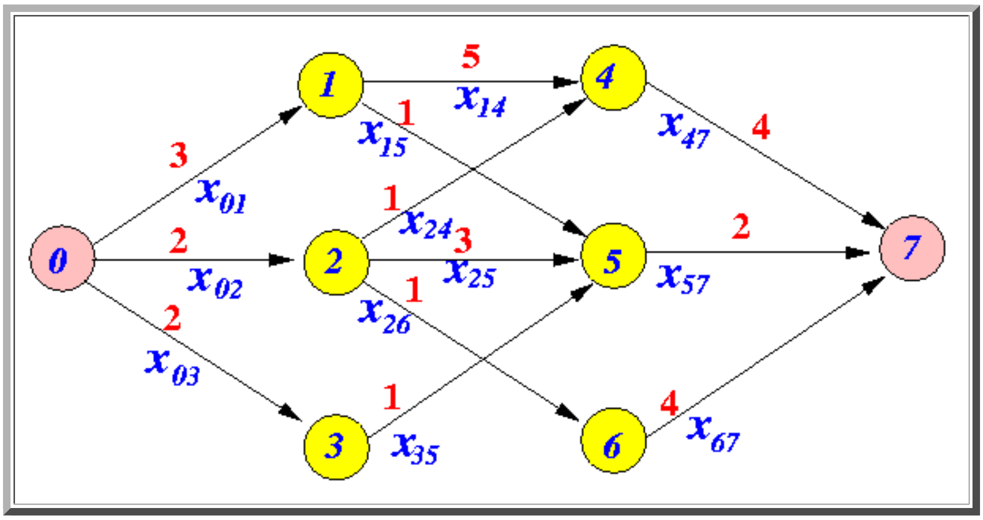
    - **How to** formulate a **max flow problem** as an **LP**:
        - Introduce **variables** to represent ***flow*** over **each edge** of the network
        - Formulate the ***capacity* constraints** and ***conservation* constraints**
        - ***Add*** an **artificial *feedback* link** from **sink → source** to represent the ***total*flow**
            - The **objective function** of the **LP** is the ***total* flow** (over the **artificial *feedback* link**)

- **Flow variables**
  - The **key** to convert a **max flow problem** into a **Linear Program** is the use of:
    - **Flow variables**

  - **Flow variable**: how much **flow** over a **link**:

```
xij = amount of flow from i → j
```

  - **Example:**
    - Consider the following **basic network**:



    - In order to **formulate** the **max flow** problem as an **LP**, we will need to **introduce** the following **flow variables**:

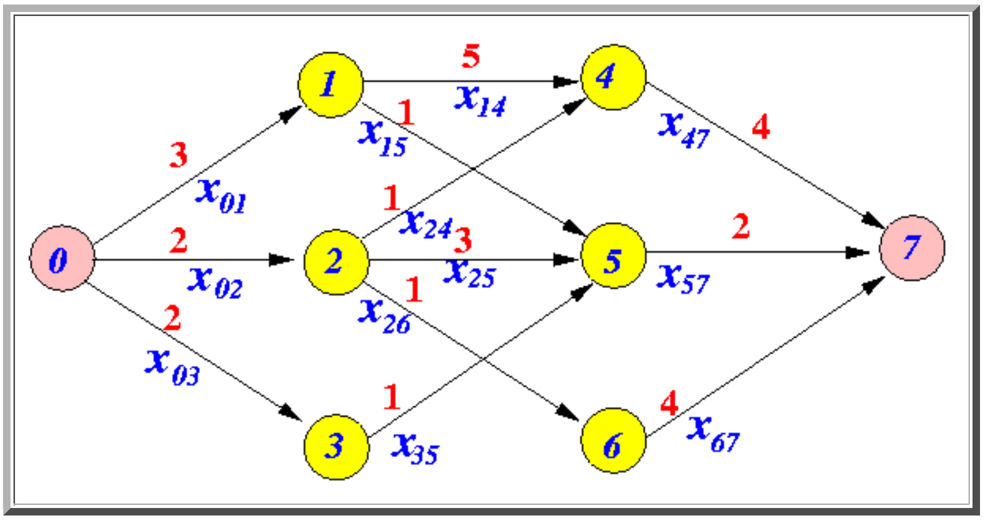
```
x01   x02   x03
x14   x15
x24   x25   x26
x35
x47
x57
x67
```

- **Constraints**
  - There are **2 types** of **constraints** in a **basic network**
    - **Capacity constraints**
    - **Flow conservation constraints**

- **Flow capacity constraints**
  - **Capacity constraints:**

- The **flow** over **any lonk** **cannot exceed** the **capacity** of that link
  - **∀ link *e*: *f(e)* ≤ *c(e)***

- **Example:** given **basic network**



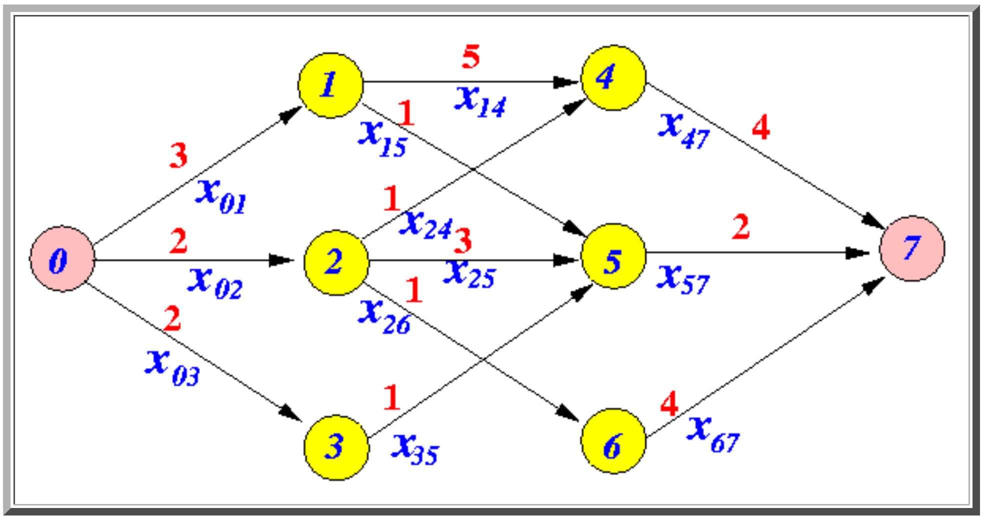
- **Flow capacity constraints:**

$$
\begin{array}{lll}
x_{01} & \le & 3 \\
x_{02} & \le & 2 \\
x_{03} & \le & 2 \\
x_{14} & \le & 5 \\
x_{15} & \le & 1 \\
x_{24} & \le & 1 \\
x_{25} & \le & 3 \\
x_{26} & \le & 1 \\
x_{35} & \le & 1 \\
x_{47} & \le & 4 \\
x_{57} & \le & 2 \\
x_{67} & \le & 4
\end{array}
$$

- **Flow conservation equations**
  - **Flow conservation constraints:**
    - The **flow conservation constraint** is **valid** for **nodes** ***other* than** the **source *S*** and **sink *T*** !!!

    - **Total flow** flowing ***into* a node** = **Total flow** flowing ***out of* a node**
      - **∀ node *n*** (***n* ≠ *S*** and ***n* ≠ *T***): **∑(flow into *n*) = ∑(flow out of *n*)**

  - **Example:**

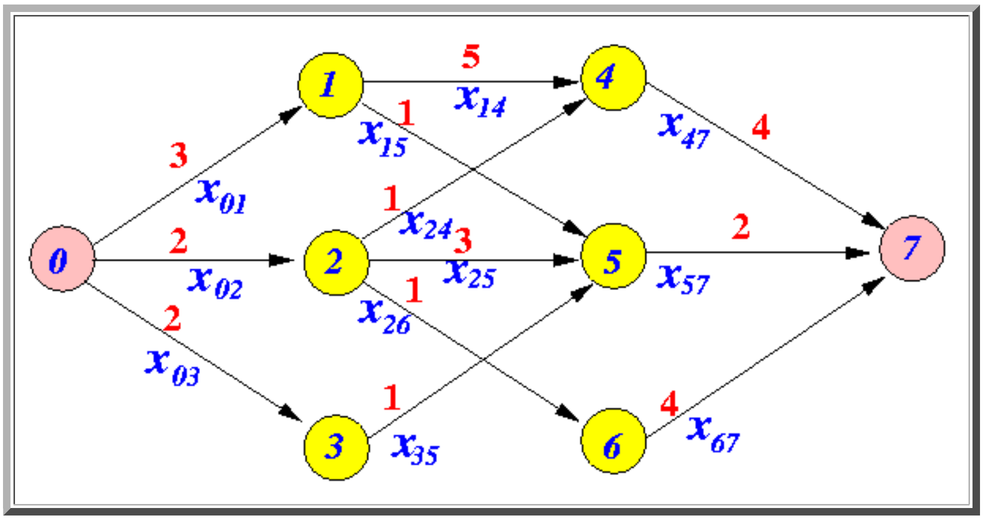


- **Flow conservation constraints:**

```
node 1:  x01             = x14 + x15
node 2:  x02             = x24 + x25 + x26
node 3:  x03             = x35
node 4:  x14 + x24        = x47
node 5:  x15 + x25 + x35 = x57
node 6:  x26             = x67
```

- **Objective function**
  - The ***objective*** is to **maximize the flow** from **source node 0** to **sink node 7**
  - **Direct Method:**
    - **Objective function** = **sum** of ***all*** **flows** eminating from the **source *S***

Example:



Objective function:

```
max:  x01 + x02 + x03
```

- **Example Program:** (Demo the **lp_solve** code) *Example*

  - **lp_solve** input file: click here

How to run the program:

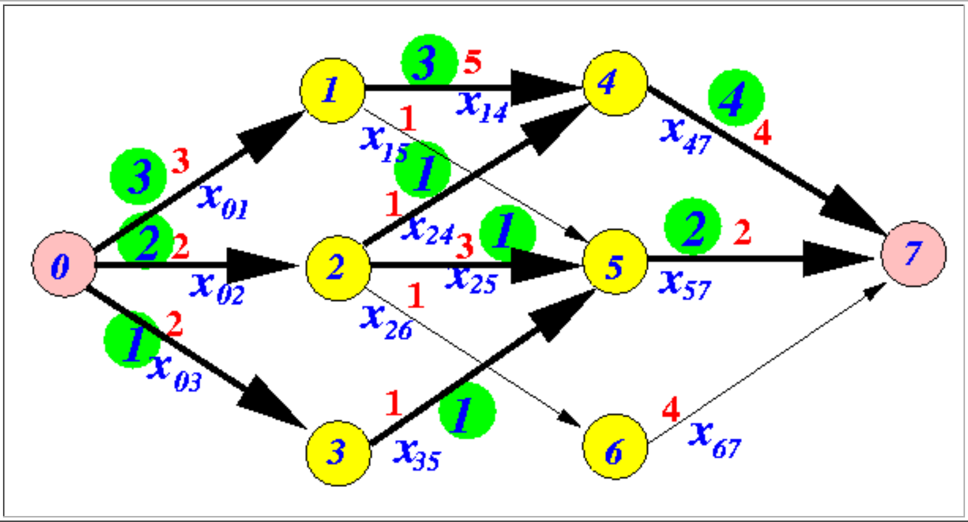
- **Right click** on link(s) and **save** in a scratch directory
- To run: `lp_solve lp2`

Output:

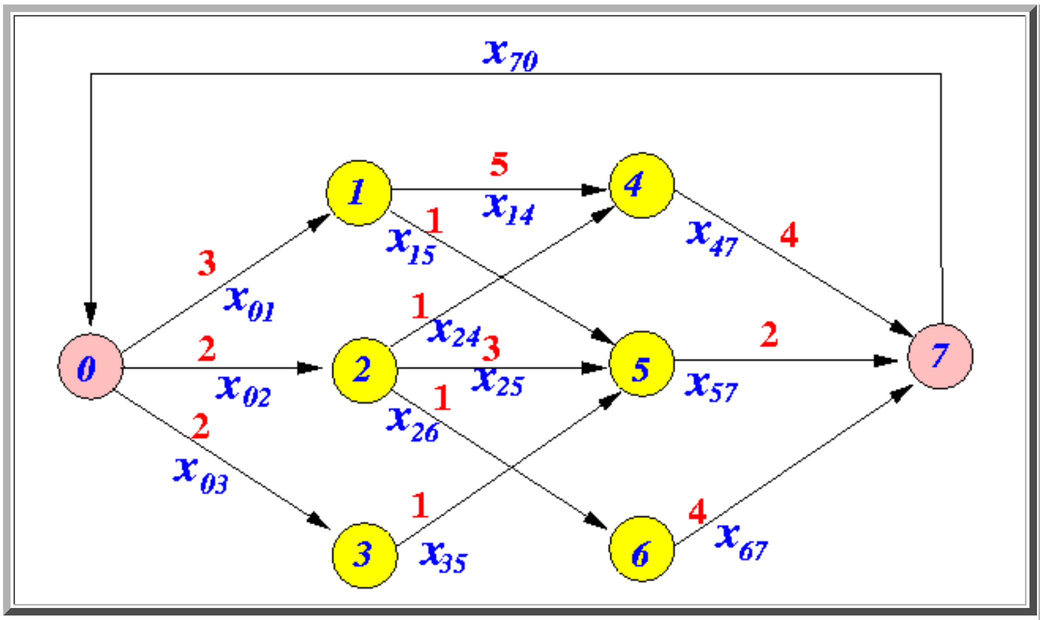
```
Value of objective function: 6.00000000

Actual values of the variables:
x01                          3
x02                          2
x03                          1
x14                          3
x15                          0
x24                          1
x25                          1
x26                          0
x35                          1
x47                          4
x57                          2
x67                          0
```

Corresponding max flow:



- **An alternative way to introduce the objective function**
  - The following is a **popular way** to **introduce** the ***objective function***:
    - We **introduce** an **artificial flow** from **sink *T*** back to **source *S***:



I.e.:

- We make a ***closed* network**

- We can **now add** the following **flow *conservation* constraints** to the **source *S*** and the **sink *T***:

```
flow into source = flow out of source:

          x70 = x01 + x02 + x03

flow into sink = flow out of sink:

          x47 + x57 + x67 = x70
```

- The **objective function** is ***simply***:

```
max:  x70
```

(**Note:** $x_{70} = x_{01} + x_{02} + x_{03}$ --- so we are not doing anything different....)

- **Resulting Linear Program:**

```
max:  x70

s.t.:
        x01               - x14 - x15        = 0
        x02               - x24 - x25 - x26  = 0
        x03               - x35              = 0
        x14 + x24          - x47             = 0
        x15 + x25 + x35 - x57                = 0
        x26               - x67              = 0

        x70               - x01 - x02 - x03  = 0
        x47 + x57 + x67 - x70                = 0

        x01  ≤  3
        x02  ≤  2
        x03  ≤  2
        x14  ≤  5
        x15  ≤  1
```

$$\begin{aligned} x_{24} &\le 1 \\ x_{25} &\le 3 \\ x_{26} &\le 1 \\ x_{35} &\le 1 \\ x_{47} &\le 4 \\ x_{57} &\le 2 \\ x_{67} &\le 4 \end{aligned}$$

- **Example Program:** (Demo above code)

  Example

  - **lp_solve** input file: click here

How to run the program:

- **Right click** on link(s) and **save** in a scratch directory
- To run: `lp_solve lp1`

- **Solution:**

```
>> lp_solve lp1

        Value of objective function: 6.00000000

        Actual values of the variables:
        x70                         6
        x01                         3
        x14                         3
        x15                         0
        x02                         2
        x24                         1
        x25                         1
        x26                         0
        x03                         1
        x35                         1
        x47                         4
        x57                         2
        x67                         0
```