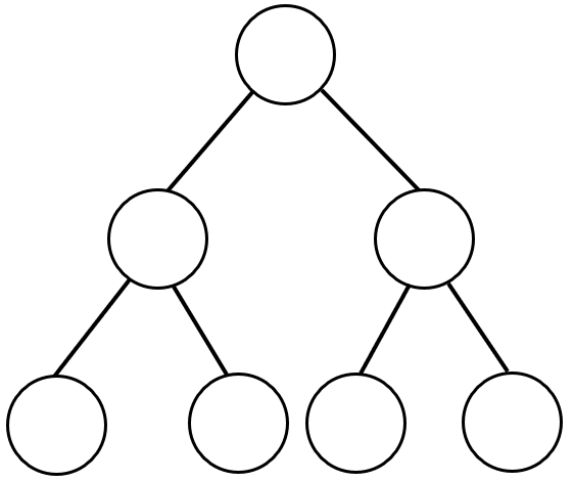
LECTURE 15: NETWORK SIMPLEX ALGORITHM

1. Trees

Definition:

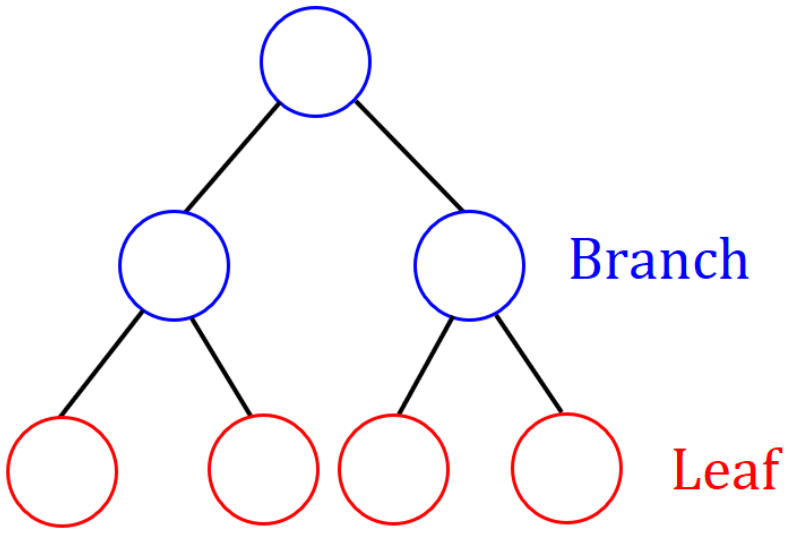
A **tree** is a connected graph that has no cycles.



Definition:

A **leaf** of a tree is a vertex of degree 1, else it is a **branch**

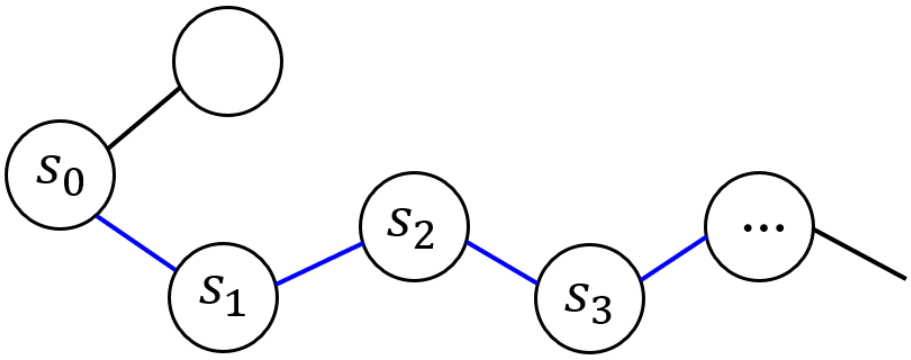
Date: Thursday, October 27, 2022.



Fact 1:

Every (finite) tree has a leaf

Why? Suppose not, that is each vertex has degree ≥ 2



Consider a path that starts at a vertex $s = s_0$. Since the graph is connected, s_0 has a neighbor s_1. Since s_1 is not a leaf, it has a neighbor $s_2 \neq s_0$. Since s_2 is not a leaf, it has a neighbor s_3 different from s_1 and s_2 Moreover $s_3 \neq s_0$ otherwise we have a cycle. Continuing this way we get an infinite path $s_0 \to s_1 \to s_2 \to \cdots$ which contradicts

that the tree is finite $\Rightarrow\Leftarrow$ $\square$

The tree above has 7 vertices and 6 edges. This is always true:

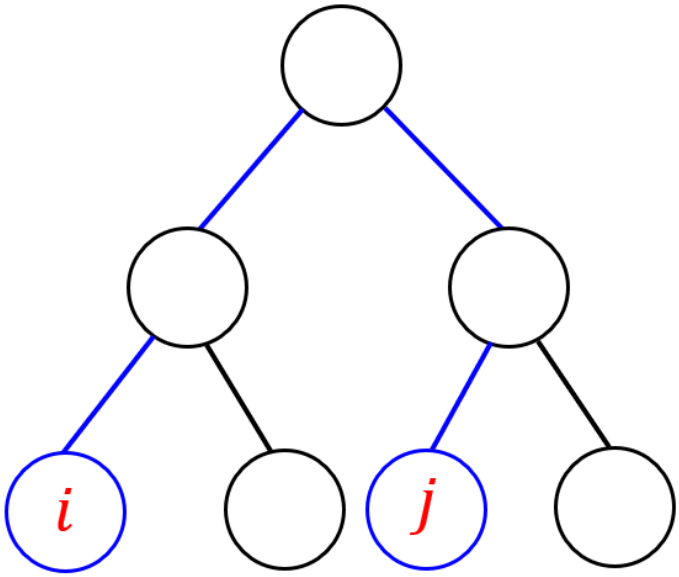
Fact 2:

A tree of n vertices has $n-1$ edges

(You'll prove this on the homework)

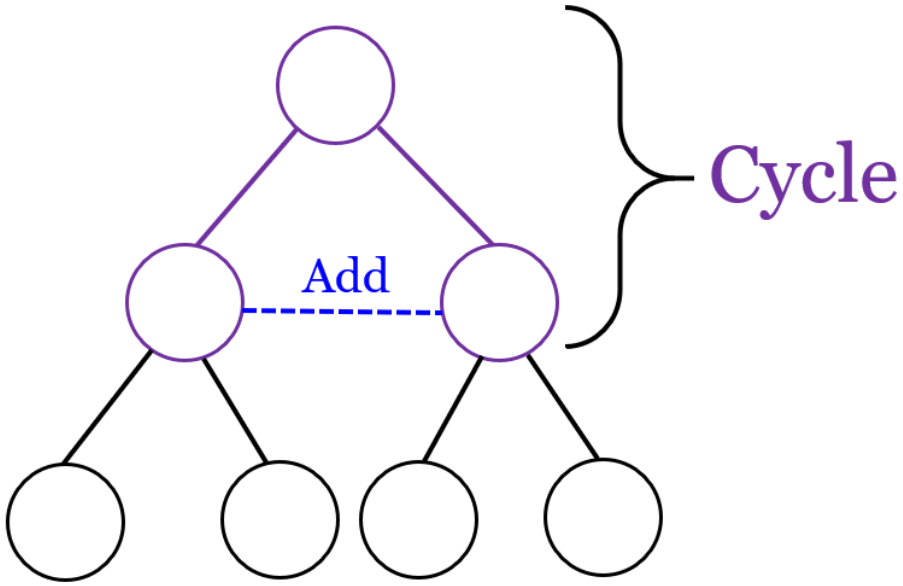
Fact 3:

For every two vertices i and j of a tree, there is a unique path going from i to j



Fact 4:

Adding any edge to a tree creates a unique cycle



2. Network Problems and Trees

Ultimate Goal: Find a simplex algorithm for network LP problems

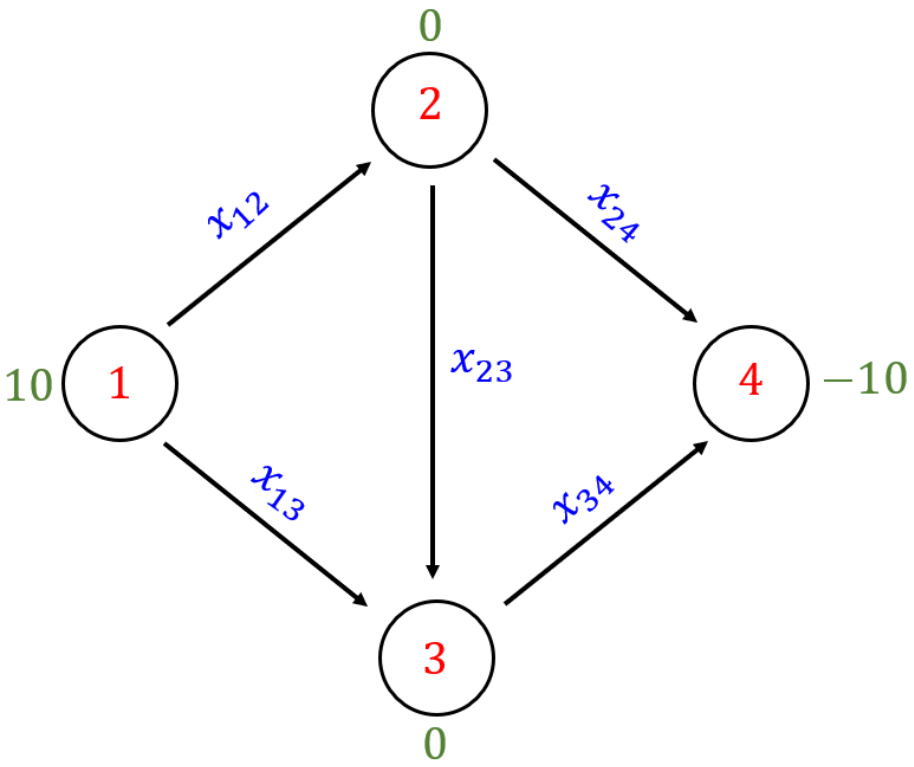
One way is to apply our usual simplex method, but this one is slow and doesn't use the special structure of the problem.

Here is where trees come in surprisingly handy!

Note: Assume there are no capacity constraints/max weights here.

Example 1:

Coffee Network, see graph below



Network LP Problem:

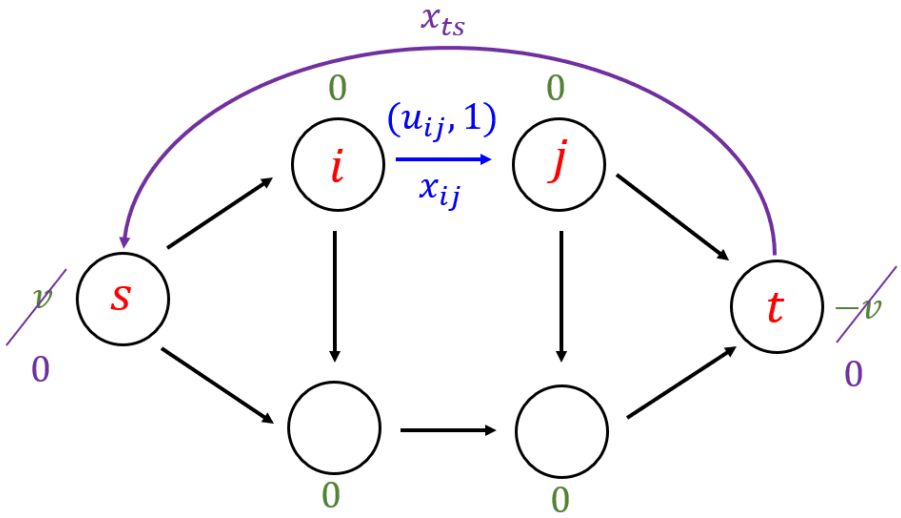
$$\begin{aligned} &\min c^T x \\ &\text{subject to } Ax = b \\ &\qquad\qquad x \geq 0 \end{aligned}$$

$$x = \begin{bmatrix} x_{12} \\ x_{13} \\ x_{23} \\ x_{24} \\ x_{34} \end{bmatrix}, \; c = \begin{bmatrix} 5 \\ 2 \\ 1 \\ 6 \\ 3 \end{bmatrix}, \; A = \begin{bmatrix} 1 & 1 & 0 & 0 & 0 \\ -1 & 0 & 1 & 1 & 0 \\ 0 & -1 & -1 & 0 & 1 \\ 0 & 0 & 0 & -1 & -1 \end{bmatrix}, \; b = \begin{bmatrix} 10 \\ 0 \\ 0 \\ -10 \end{bmatrix}$$

c is the cost vector (given by problem)
A is the oriented incidence matrix
b is the supply/demand vector

Notice that the entries of b sum to 0 (supply = demand)

Note: Sometimes people use f (flow) instead of x, so they write $Af = b$. Moreover, if $b = 0$, then the solution to $Ax = 0$ is called a **circulation**, because the flow circulates between all vertices, like this picture from last time:



Study of A

$$A = \begin{bmatrix} 1 & 1 & 0 & 0 & 0 \\ -1 & 0 & 1 & 1 & 0 \\ 0 & -1 & -1 & 0 & 1 \\ 0 & 0 & 0 & -1 & -1 \end{bmatrix}$$

Recall:

$$\begin{aligned}\text{rank}(A) &= \text{number of pivots of } A \\ &= \text{number of linearly independent rows/columns of } A\end{aligned}$$

In this particular example, we have $\text{rank}(A) = 3$

This means there is one row of A we can remove without changing the rank, here for example the last one:

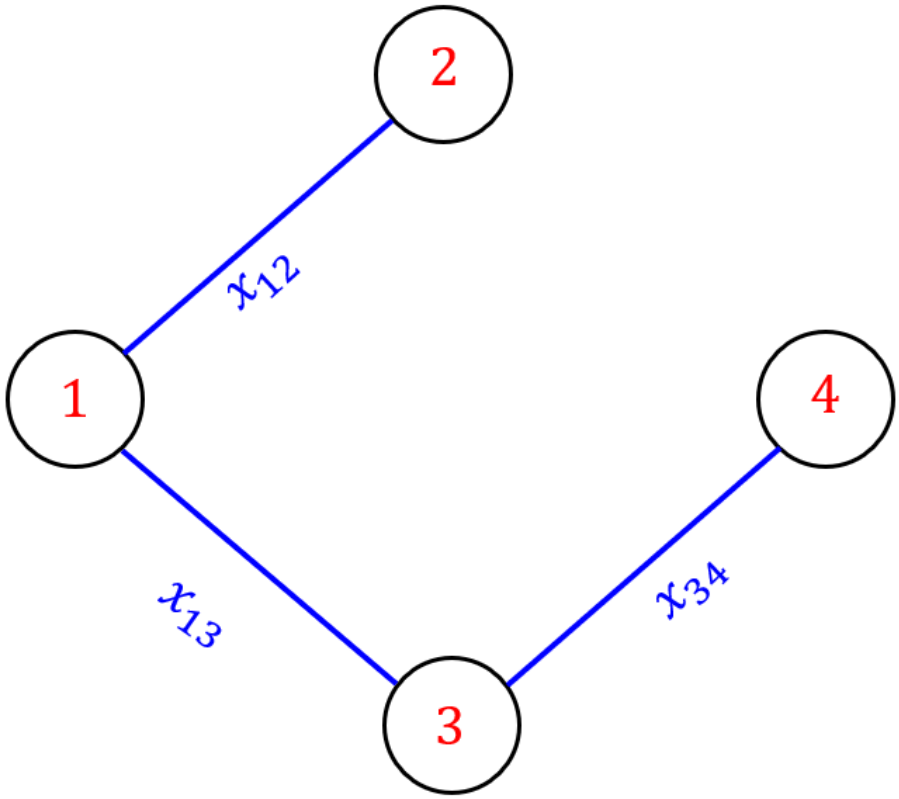
$$\begin{bmatrix} 1 & 1 & 0 & 0 & 0 \\ -1 & 0 & 1 & 1 & 0 \\ 0 & -1 & -1 & 0 & 1 \end{bmatrix}$$

We can play the same game with the columns! Since the rank is 3, this means we can remove two columns and still get a matrix with rank $= 3$. For example, let's remove columns $1, 2, 5$

$$\widetilde{A} = \begin{bmatrix} 1 & 1 & 0 \\ -1 & 0 & 0 \\ 0 & -1 & 1 \end{bmatrix}$$

This is a 3×3 matrix with rank 3, so it's invertible.

Graphical Interpretation: More interestingly, let's see what this looks like on the graph. Here the columns correspond to x_{12}, x_{13} and x_{34}



The graph here is in fact a **tree** between all the vertices, called **spanning tree** (WOW)

Cool Fact:

There is a one-to-one correspondence between

$$\{ \text{ Spanning Trees } \} \leftrightarrow \{ \text{ Invertible } 3 \times 3 \text{ sub-matrices } \}$$

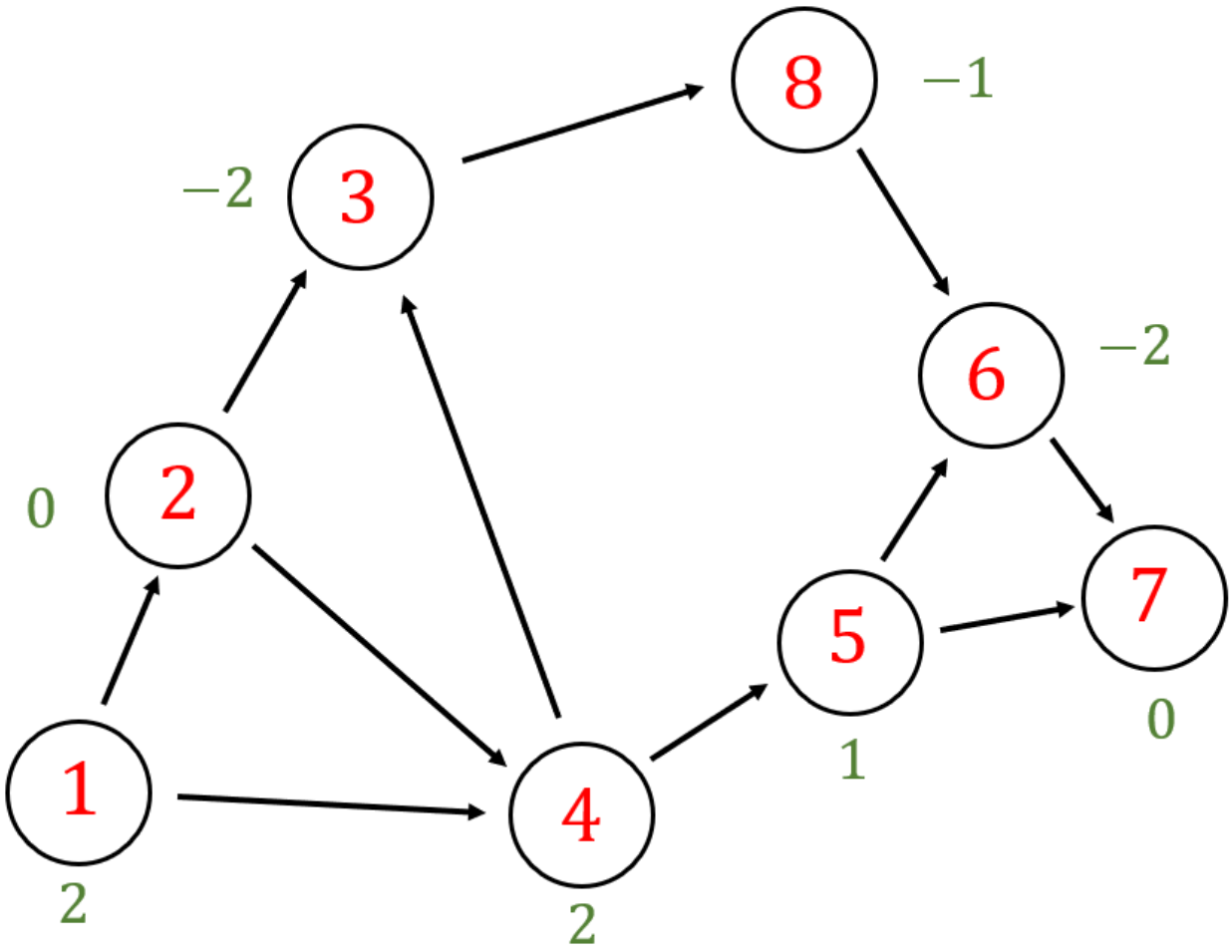
Note: So it's not really a coincidence that $\text{rank}(A) = 3$ here because a tree with 4 vertices has $4 - 1 = 3$ edges!

3. Tree Solutions

In order to start the simplex method, we need a starting vertex. Here it's much easier to find.

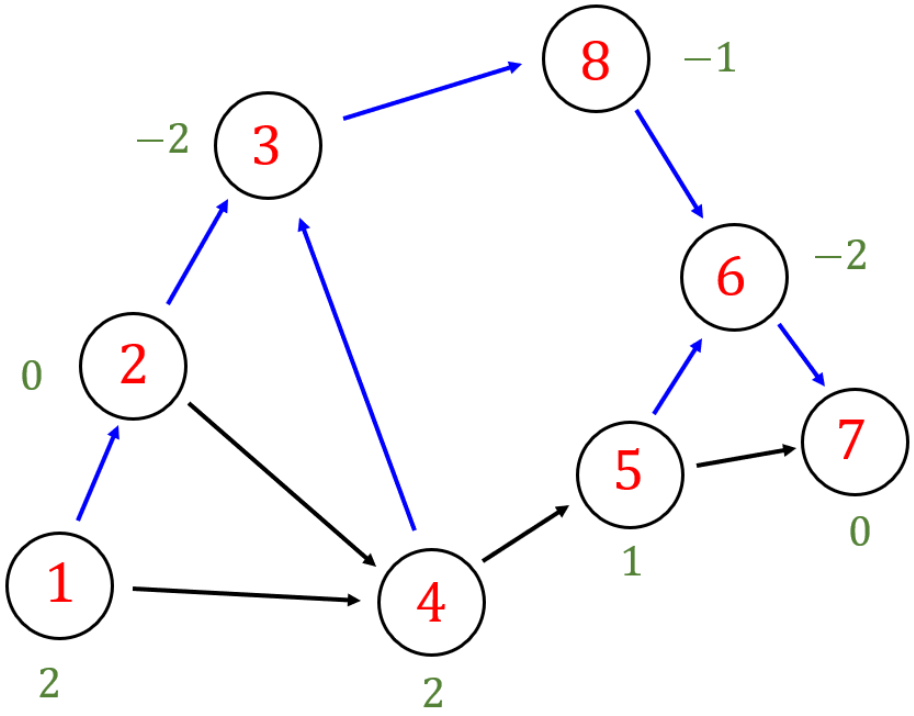
Example 2:

Find a "tree solution" (vertex) of the following graph

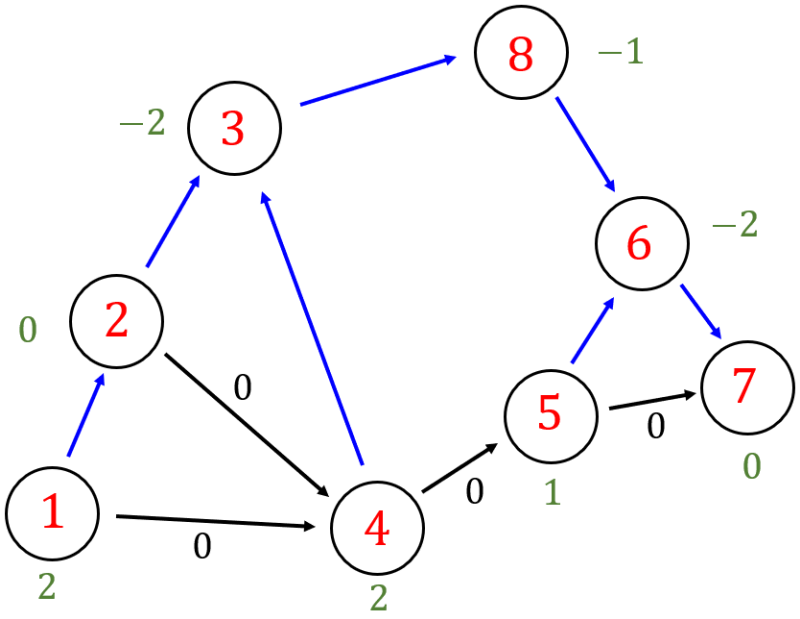


STEP 1: Find a spanning tree in this graph

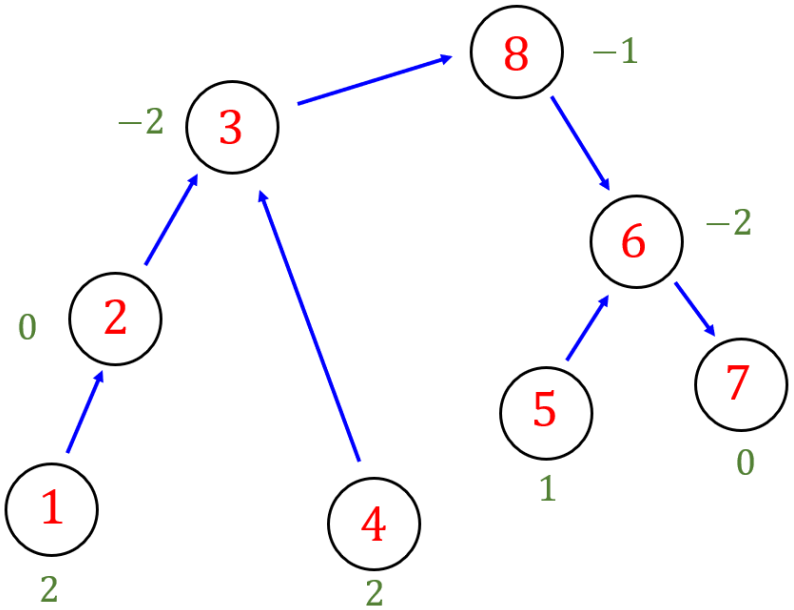
Here find a tree with 8 vertices. This is not particularly hard to do in practice: Connect all the vertices and make sure there are no cycles.



STEP 2: Assign value 0 to each edge that you didn't pick



STEP 3: Let's focus on the tree part



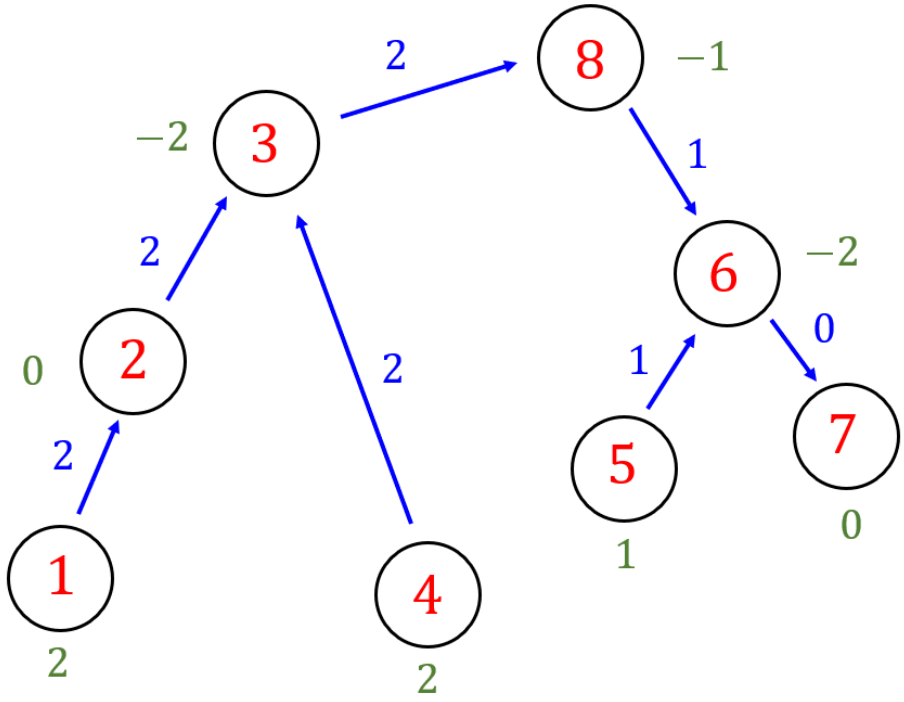
Start with a leaf, say① and find x_{12} so that the demand constraint is satisfied. This gives $x_{12} = 2$

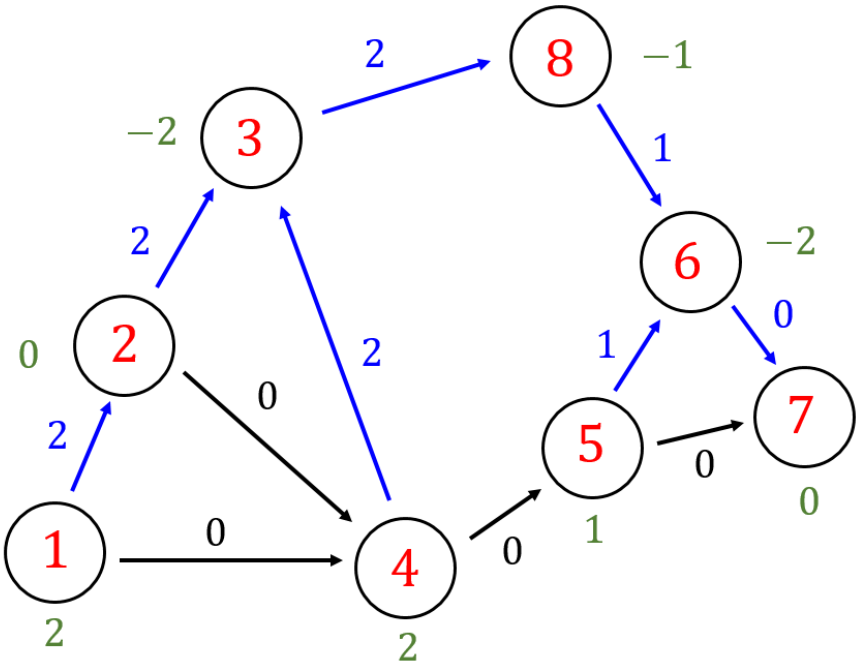
Now ignore/chop off the edge $1 \to 2$. Then ② is a new leaf, so find x_{23} so that the constraint is satisfied. This gives $x_{23} = 2$.

Now even after ignoring the edge $2 \to 3$, ③ is still not a leaf, and you're stuck! So find a new leaf, for example ④.

Then repeat this for ④ and complete the graph, starting with a new leaf whenever you're stuck.

This gives you the following graph at the end:





This gives a vector x with $x_{12} = 2$, $x_{14} = 0$ etc. called a **tree solution**

Definition:

A solution x found with this algorithm is called a **tree solution**

Note: The technical definition of a tree solution is that you

(1) Find a spanning tree T

(2) Set the non-tree variables to 0

(3) Solve $\widetilde{A}x = b$ where $\widetilde{A}$ is the invertible matrix corresponding to T discussed above, which gives you the remaining variables.

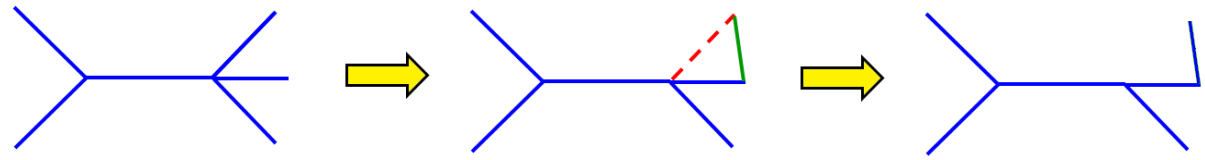
(Here x must be ≥ 0, otherwise we don't count it as a tree solution)

Fact:

x is a tree solution $\Leftrightarrow$ x is a basic feasible solution (= vertex)

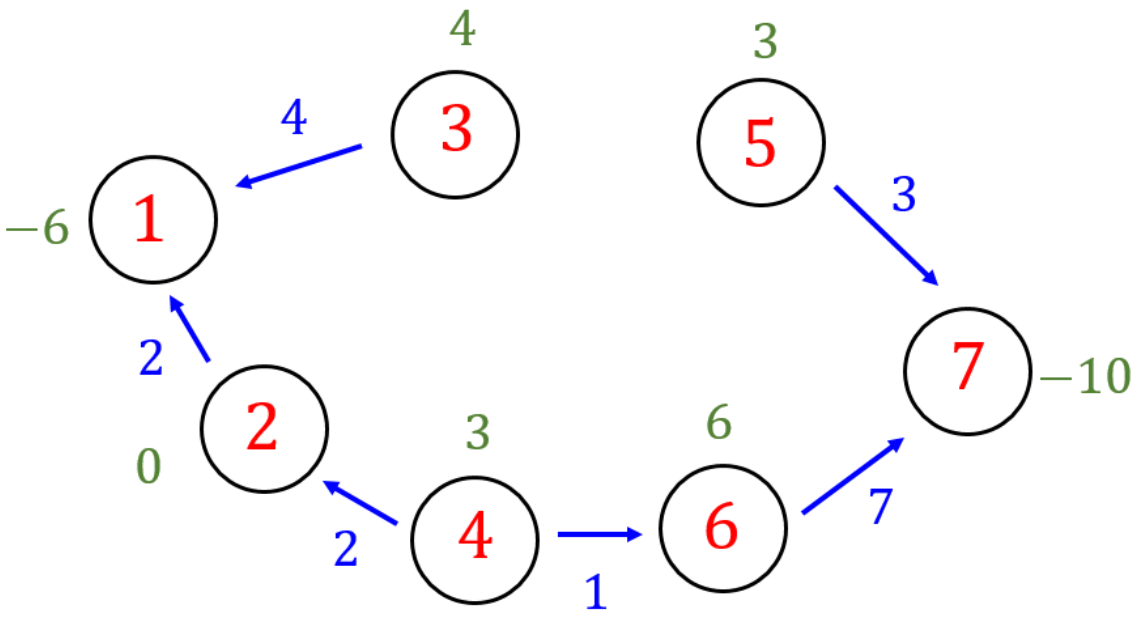
4. Network Simplex Algorithm

Main Idea: Start with a tree, add/remove an edge to get a tree with smaller $z-$value, until we can't find a better tree.



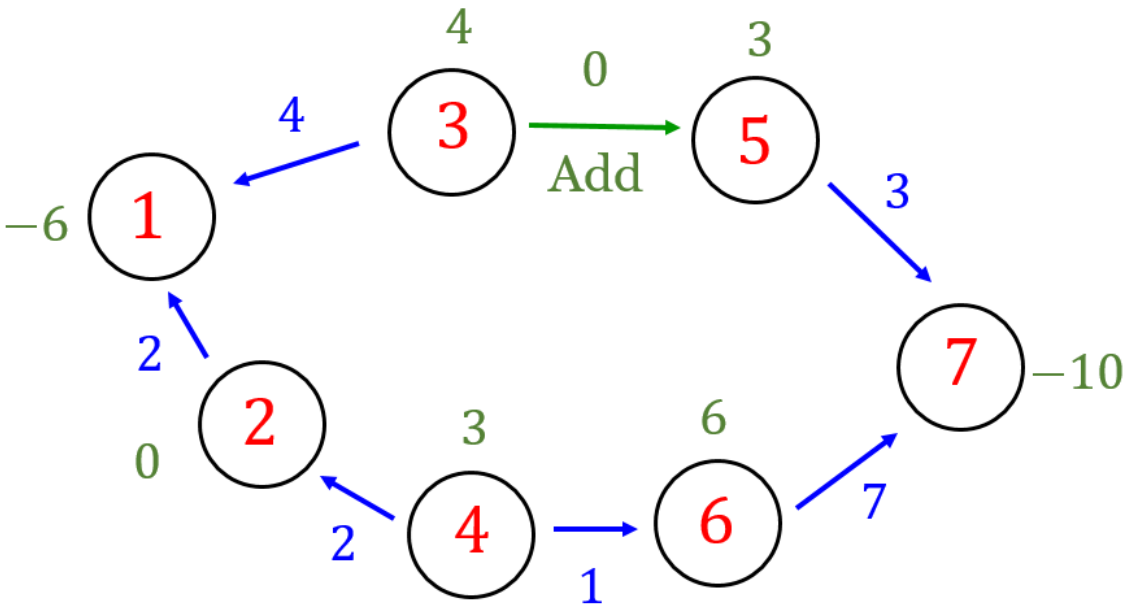
Example 3:

Suppose for example we have a graph where one tree solution is



STEP 1: Check for optimality (see later). Assume it's not optimal

STEP 2: Add an edge, say $3 \to 5$ with value 0 (assume it's is part of the original graph)



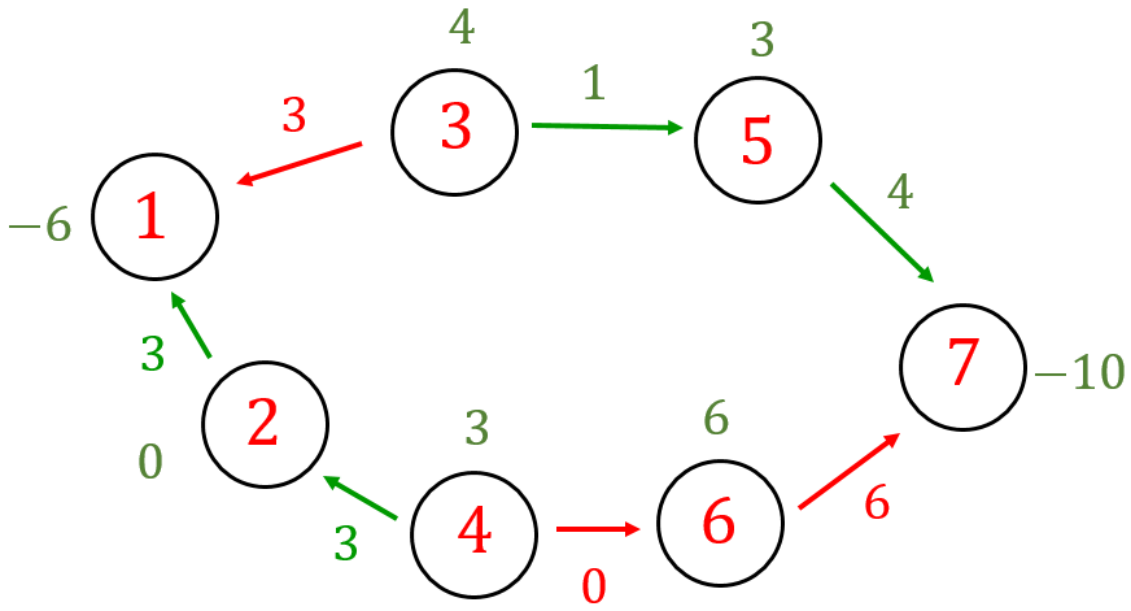
We need to remove an edge, otherwise we have a cycle! Which one to remove?

Suppose you increase the value on $3 \to 5$ from 0 to 1 (imagine shipping a new product in an already saturated marked)

Effect of that increase:

(1) All the values on the edges in the same direction as $3 \to 5$ (green arrows) get increased by 1

(2) All the values in the opposite direction as $3 \to 5$ (red arrows) get decreased by 1



Continue increasing until a red arrow becomes 0, here $4 \to 6$, and this is the edge that you remove, to get a new tree