Network Flows

1.1 Introduction

¿Have you seen it?

Everywhere we look in our daily lives, ¡networks are there!

Examples:

Electrical and power networks,

Telephone networks,

Airline service networks,

Manufacturing and distribution networks,

Computer networks.

That means,

That all of these changed the way we conduct our business an personal lives.

¿Why do we learn networks?

because we want to move an entity from a point to another as efficiently as possible, and that means learn how to model application settings as mathematical objects know as network flow problems and study algorithms to solve these.

1.1 Introduction

Network flows

We want to address 3 basic questions

¿How can we send material at minimum possible cost?

1 **Minimum cost flow problem**

¿What is the best way to traverse a network to get from a point to another as cheaply as possible?

2 **Shortest path problem**

¿How can we send as much flow as possible between two points in the network while honoring the arc flow capacities?

3 **Maximum flow problem**

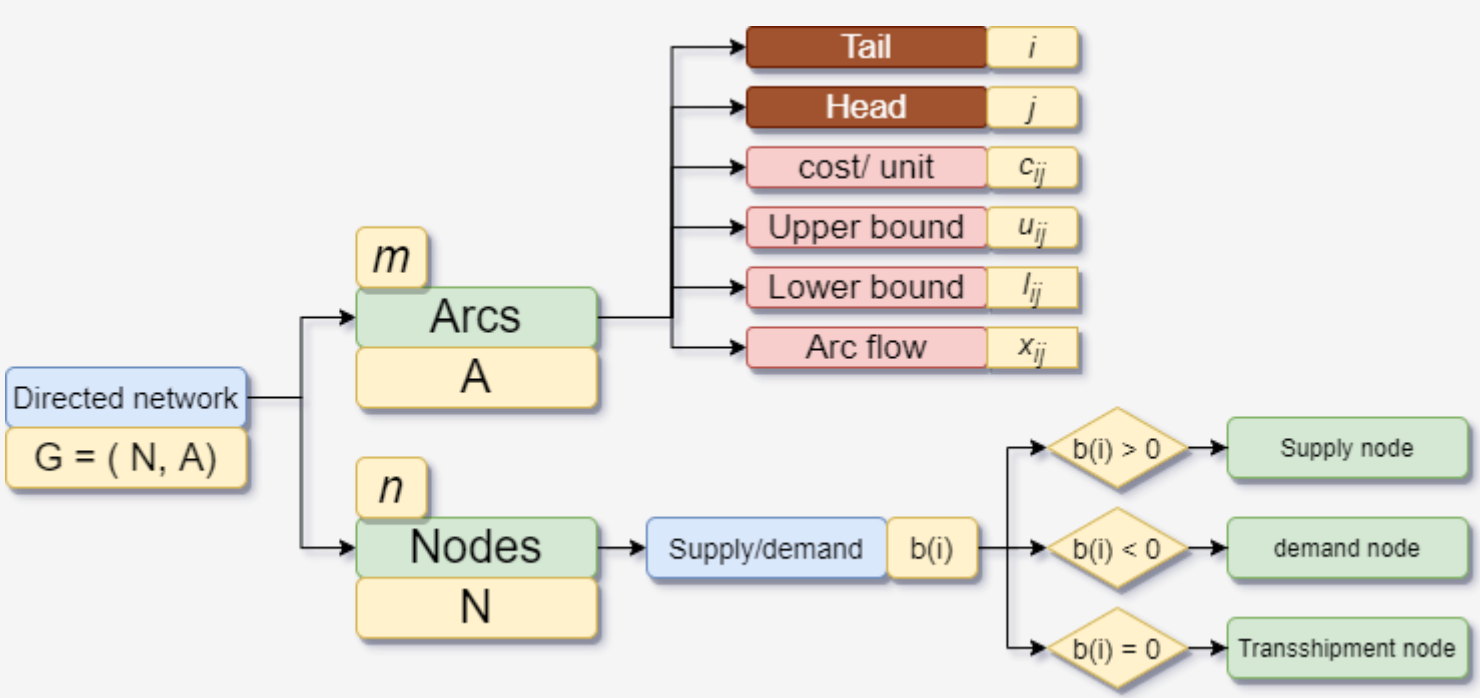
Minimum cost flow problem

Is the most funtamental of all networks flow problems

We want

to determinate a least cost shipment of an article trough a network in order to satisfy demands at certain nodes

Consists of



Variables

1	Cost/Unit	c_{ij}
2	Arc flow	x_{ij}

Formula

Minimize

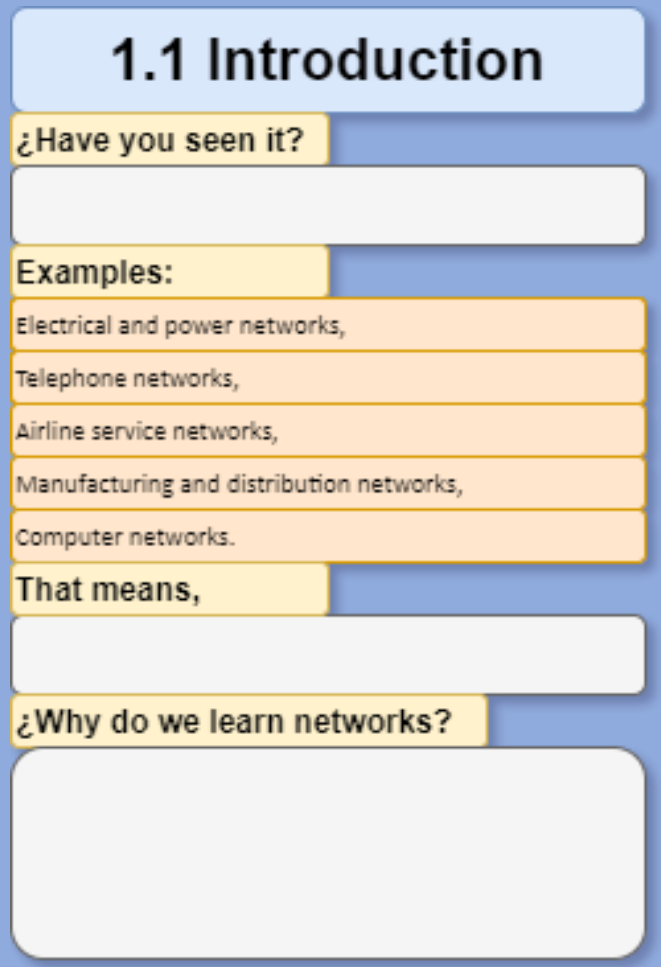
$$\sum_{(i,j)\in A} c_{ij}x_{ij}$$

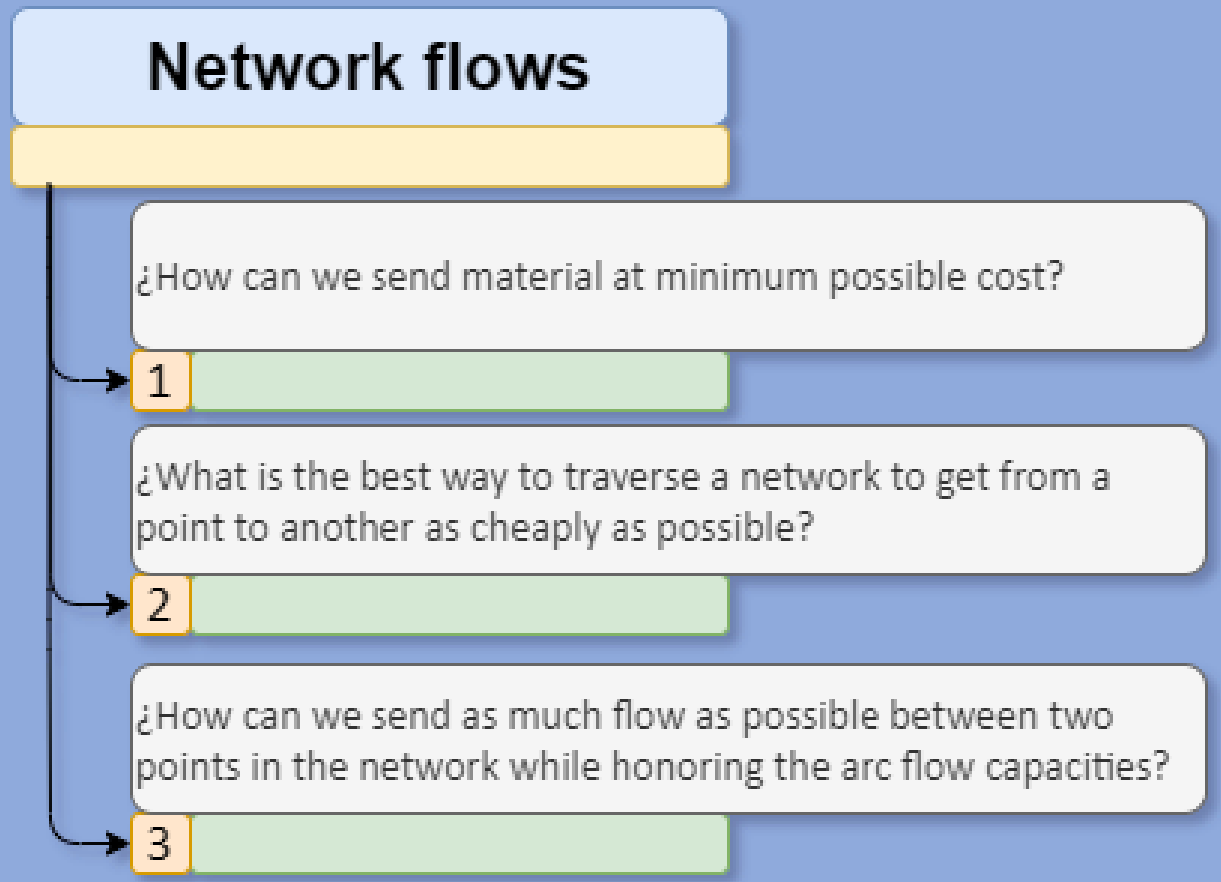
Subject to

1	$l_{ij} \le x_{ij} \le u_{ij}$	for all	$(i,j) \in A$
2	$b(i) = \sum x_{ij} - \sum x_{ji}$	for all	$i \in N$
3	$\sum_{i=1}^{n} b(i) = 0$	for all	

1.2 Introduction (problems 1)

- Complete



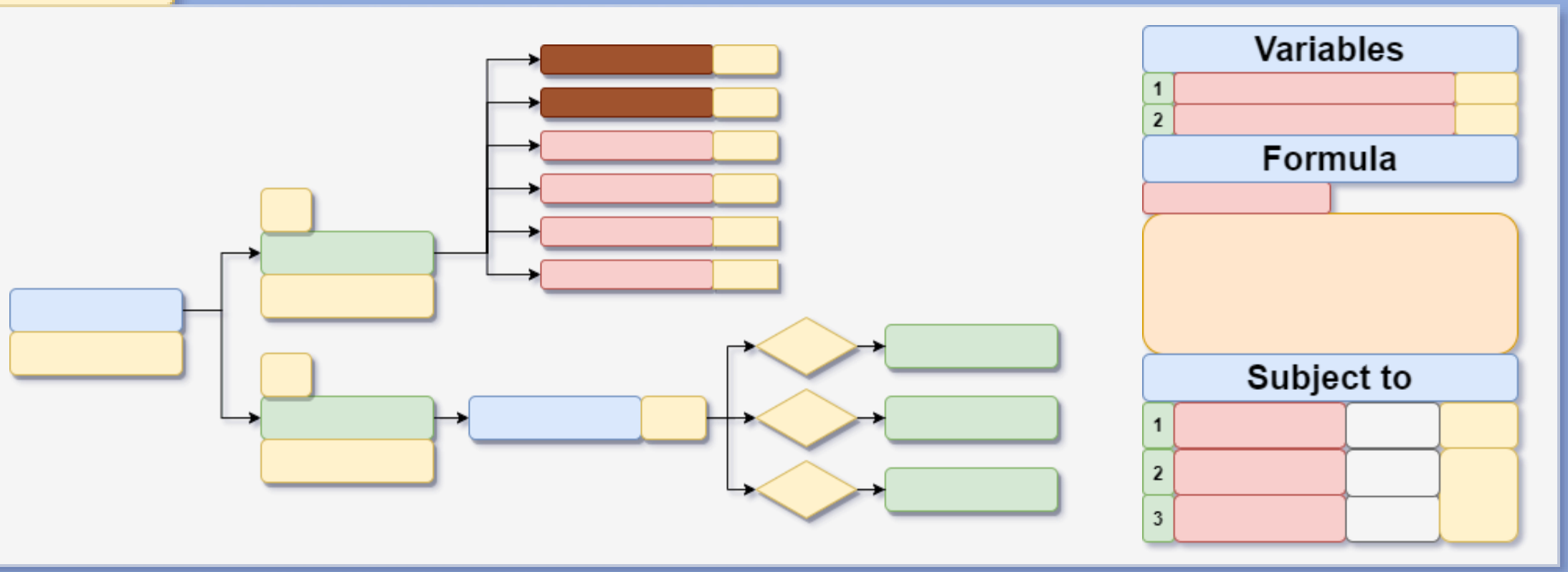


1.2 Introduction (problems 2)

- Complete

We want

Consists of



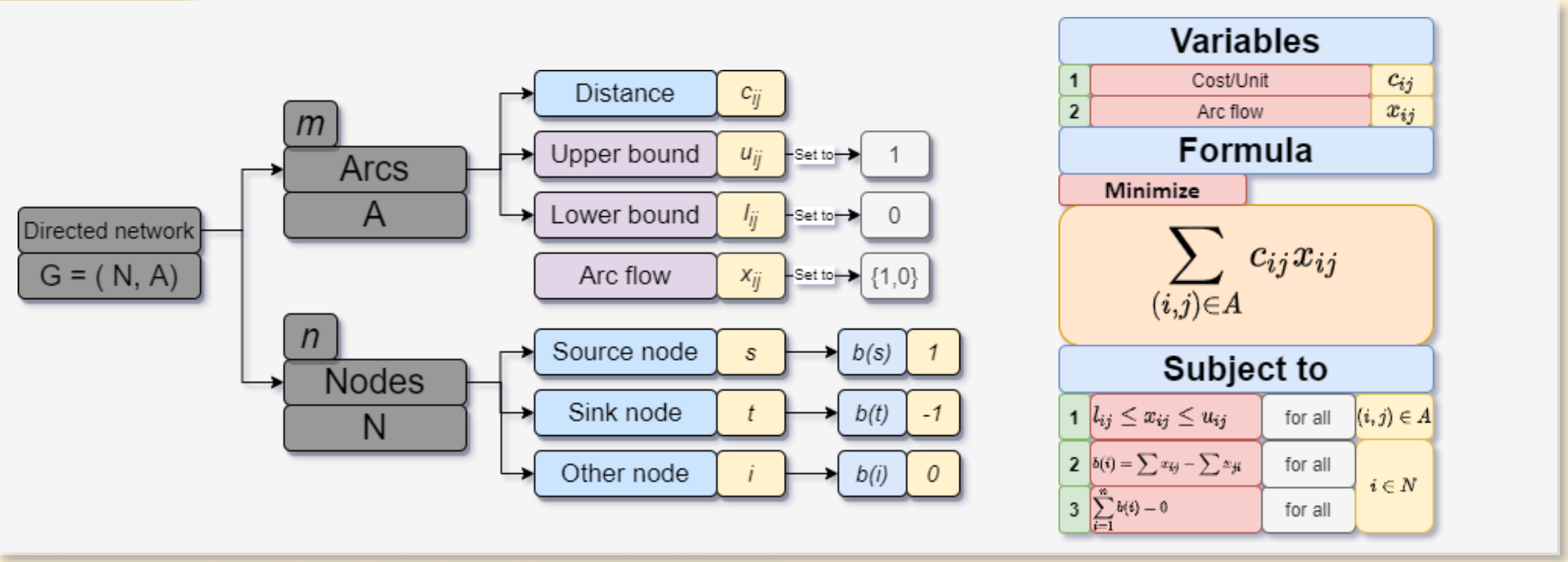
Shortest path problem

Perhaps the simplest of all networks problems

We want

to find the shortest path *s* to *t* in our network.

Consists of



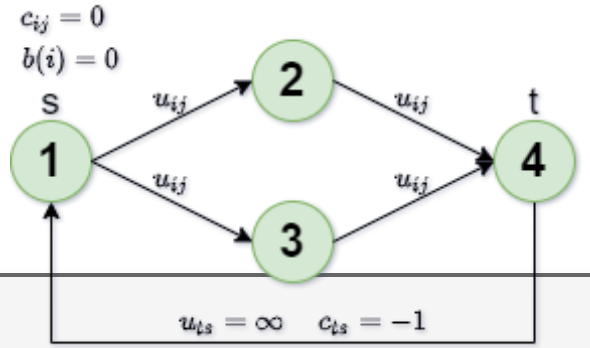
Maximum flow problem

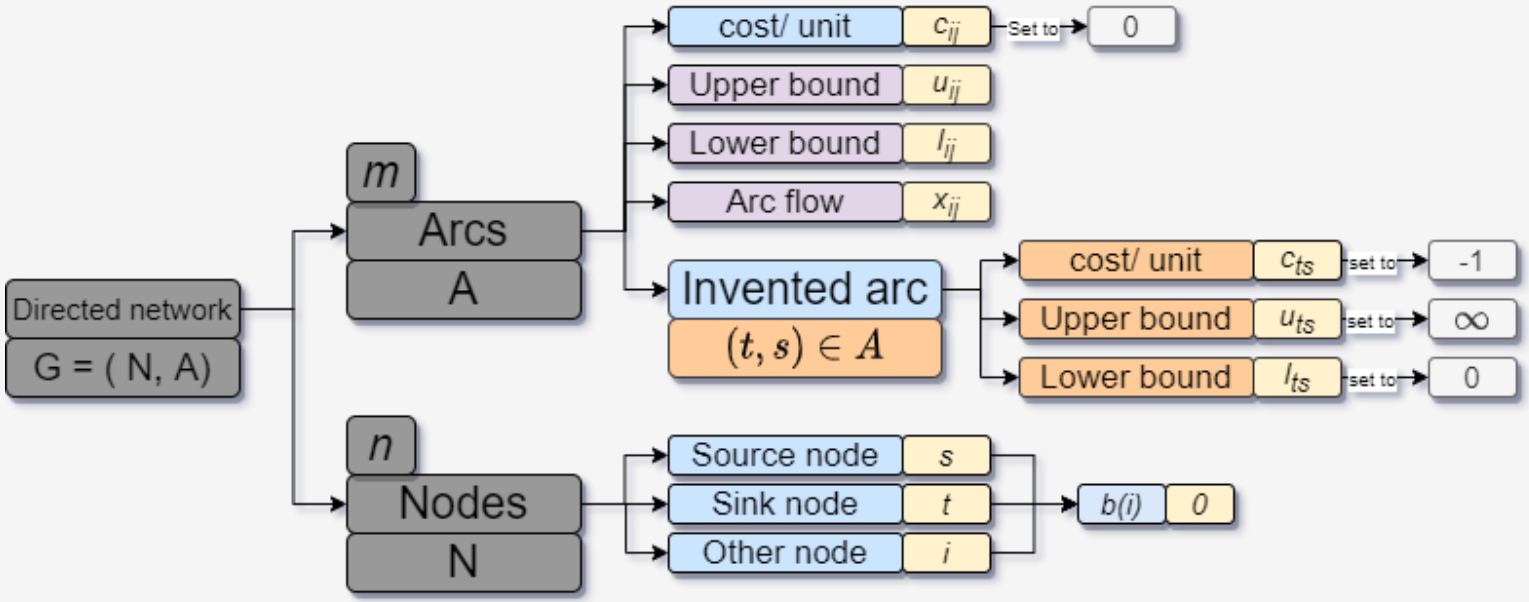
Is in a sense a complementary model to the shortest path problem

We want

to send as much flow from *s* to *t* in a network

Consists of





Variables

1	Cost/Unit	c_{ij}
2	Arc flow	x_{ij}

Formula

Minimize

$$\sum_{(i,j)\in A} c_{ij}x_{ij}$$

Subject to

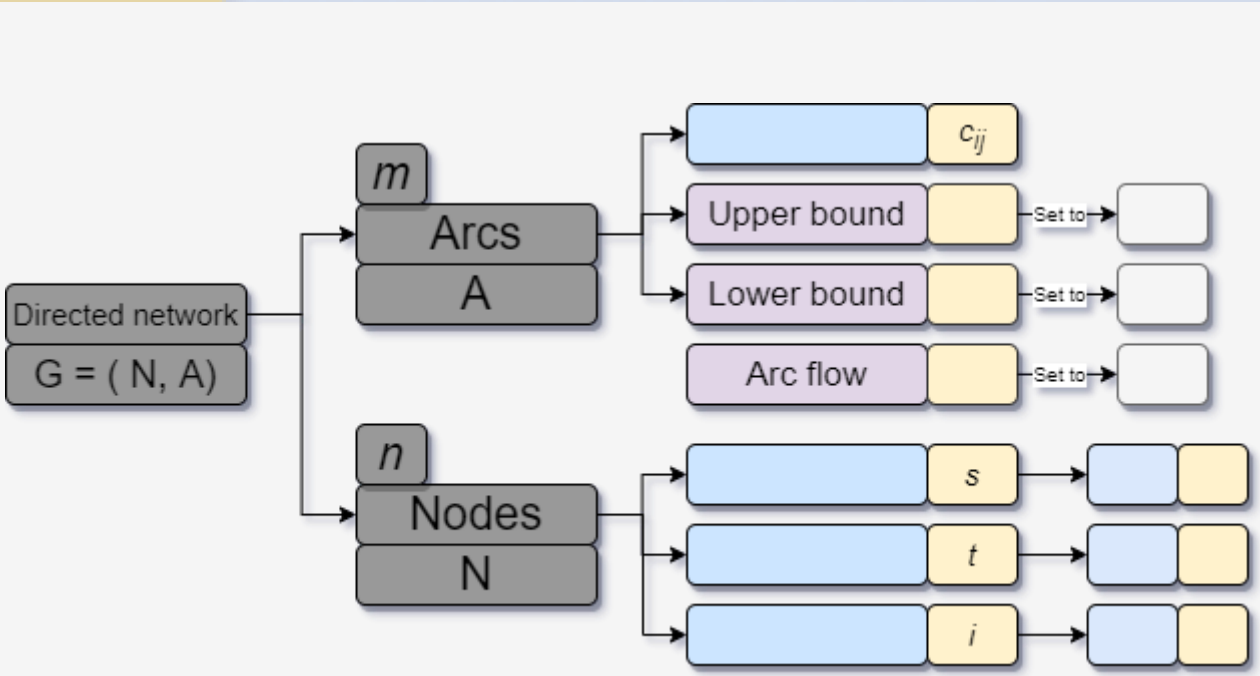
1	$l_{ij} \le x_{ij} \le u_{ij}$	for all	$(i,j) \in A$
2	$b(i) = \sum x_{ij} - \sum x_{ji}$	for all	$i \in N$
3	$\sum_{i=1}^{n} b(i) = 0$	for all	

1.2 Network flow problems (problems 3)

Shortest path problem

We want

Consists of



Variables

1

2 Arc flow x_{ij}

Formula

Minimize

Subject to

1 for all $(i,j) \in A$

2 $b(i) = \sum x_{ij} - \sum x_{ji}$ for all

3 $\sum_{i=1}^{n} b(i) = 0$ for all

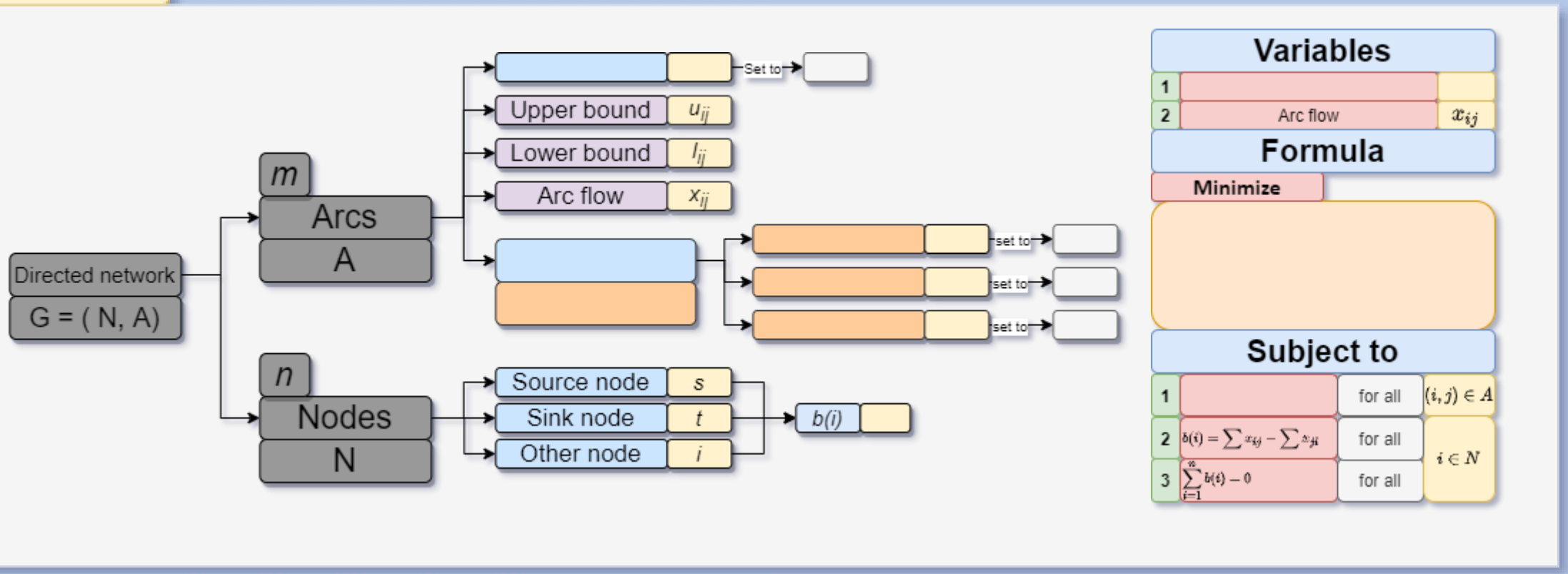
$i \in N$

1.3 Network flow problems (problems 4)

Maximum flow problem

We want

Consists of



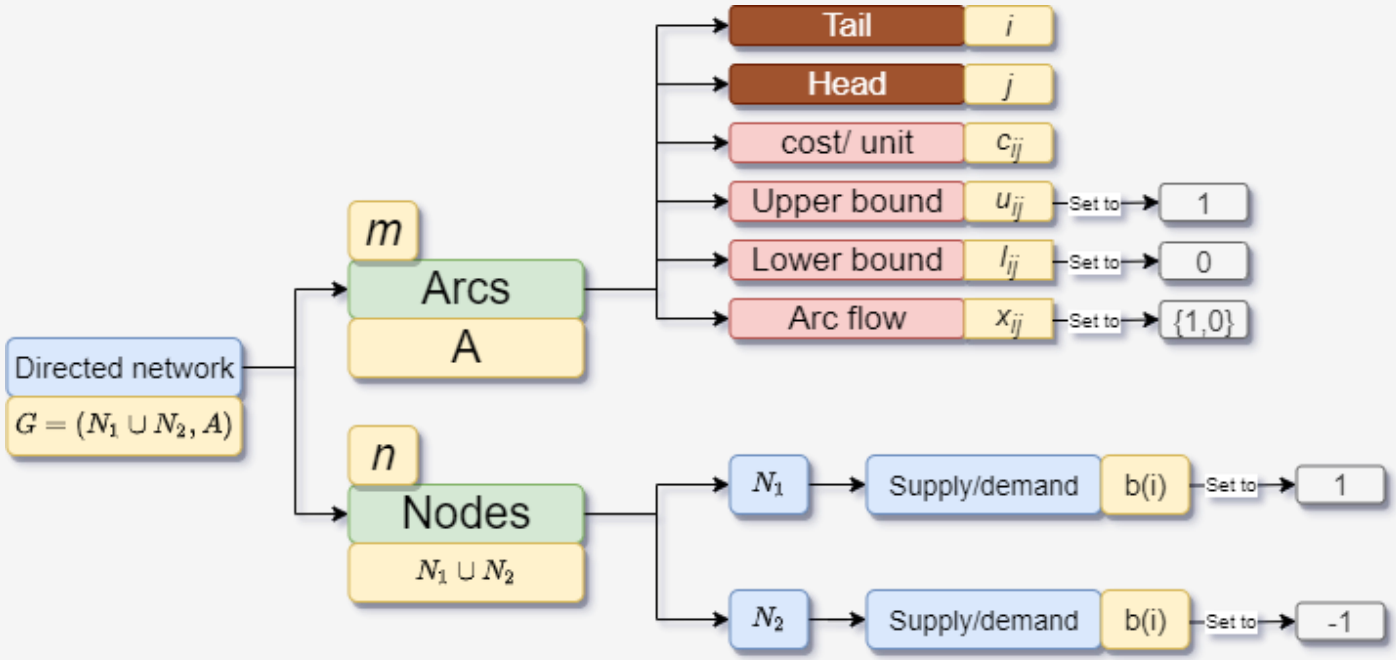
Assignment problem

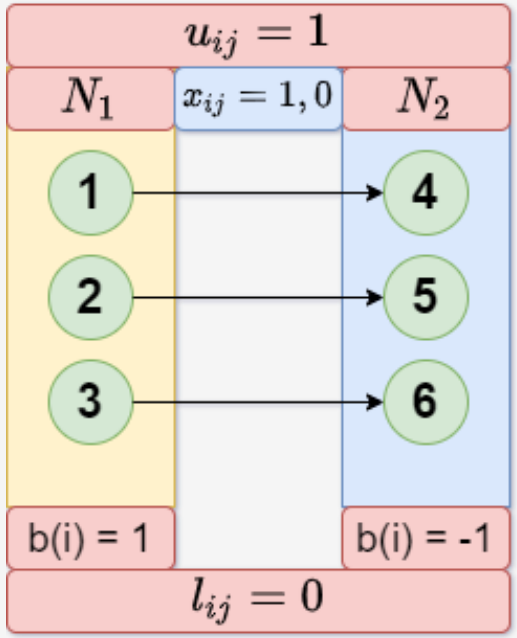
Is a minimum cost flow problem in a network

We want

to pair, each object in N_1 to exactly one of N_2

Consists of





We should note that

1	$l_{ij} = 0$ and $u_{ij} = 1$, because we want to assign one employee to one task
2	$x_{ij} = \{1,0\}$, because we want to know if it was assigned or not
3	$b(i) = 1$ for all i in N_1 because this are nodes that only gives
4	$b(i) = -1$ for all i in N_2 because this is a node that only receive

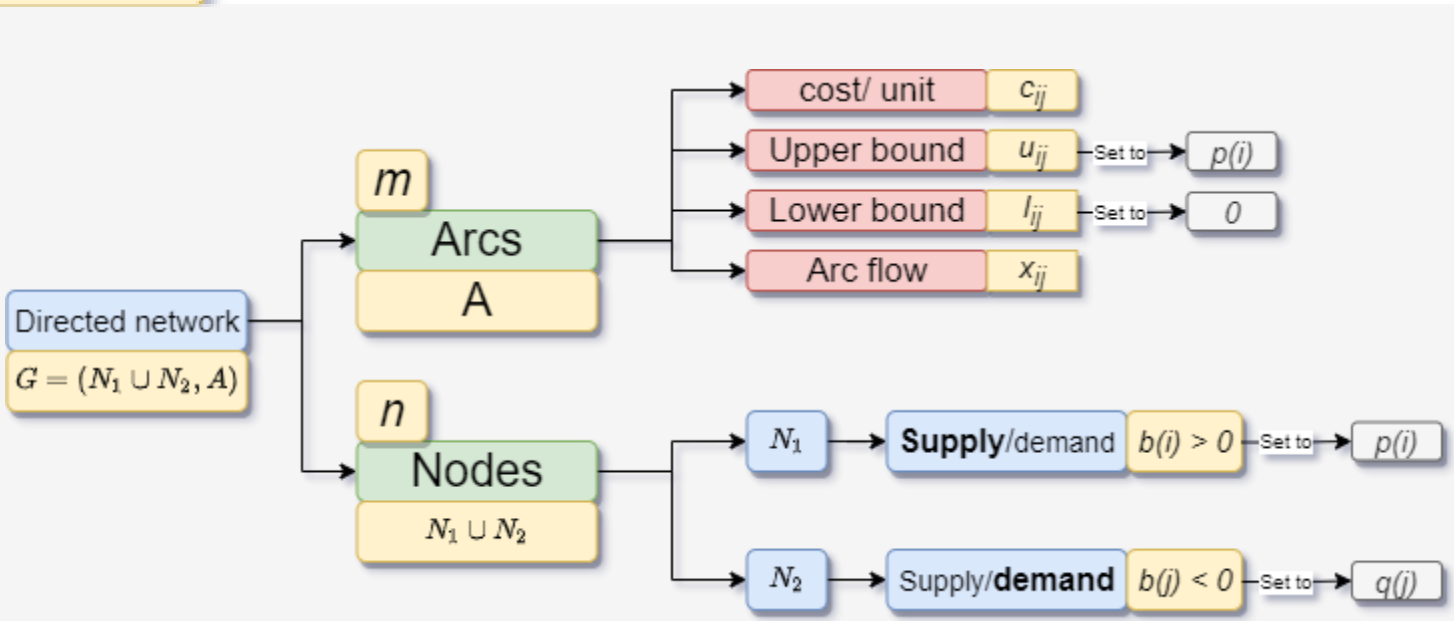
Transportation problem

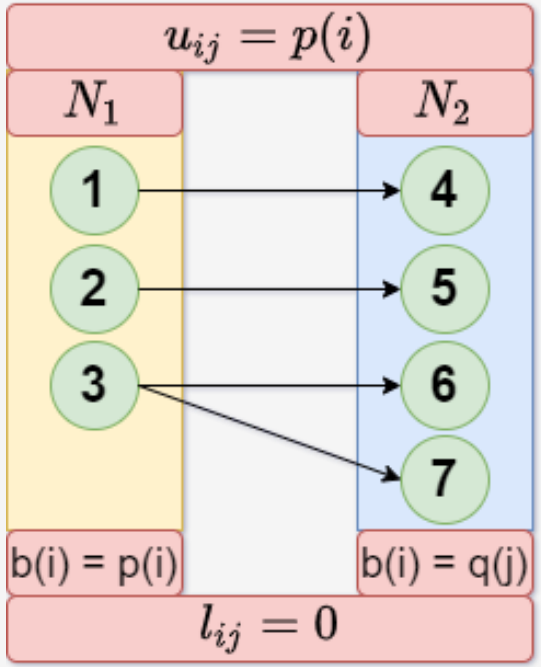
Is a special case of minimum cost flow problem

We want

To send supply from a subset N_1 to a subset N_2, of possibly unequal cardinality

Consists of





We should note that

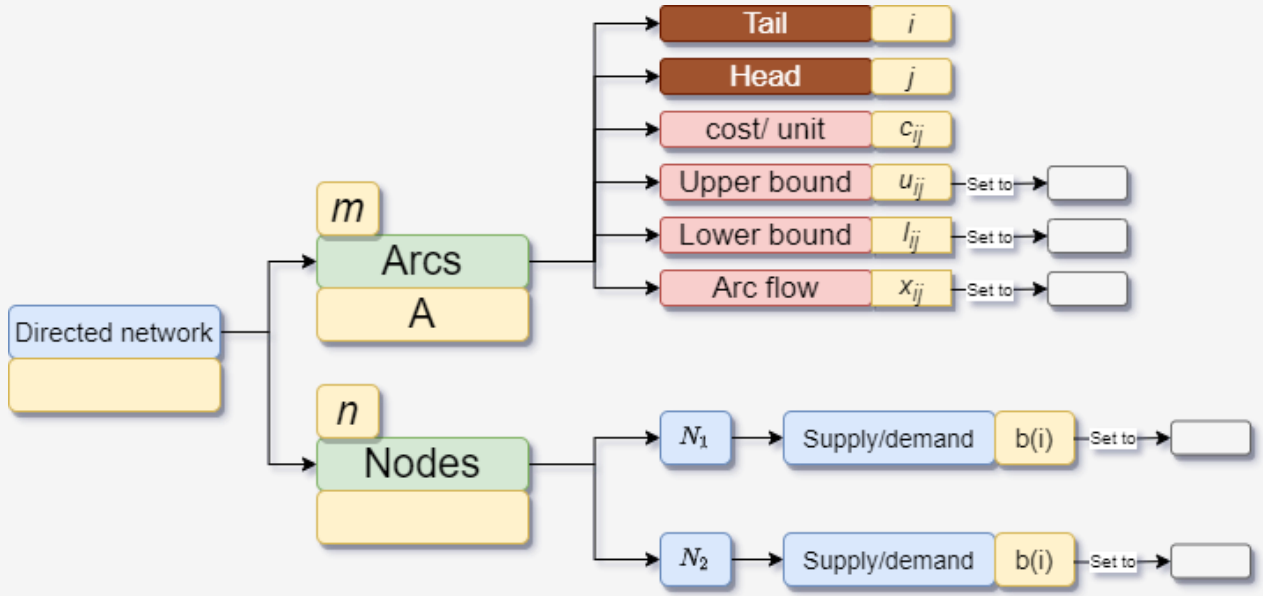
1	*Each node in* N_1 *is a supply node, because it only gives, and his production capacity is* **b(i) = p(i)**
2	*Each node in* N_2 *is a demand node, because it only receive*
3	Each arc *(i,j)* in *A*, has always *i* in N_1 and *J* in N_2
4	the capacity of each *arc (i, j)* in *A*, is all what the supply node can give u_{ij} = *p(i)*

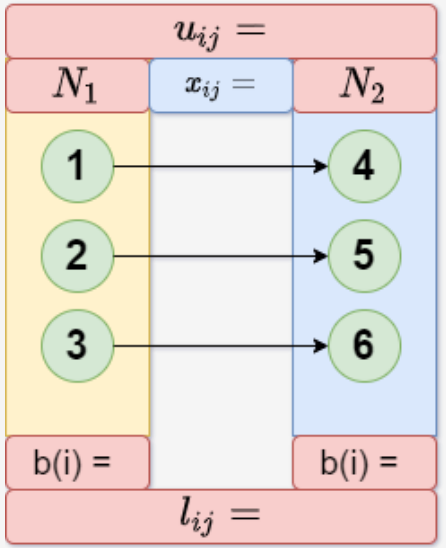
1.2 Network flow problems (problems 5)

Assignment problem

We want

Consists of





We should note that

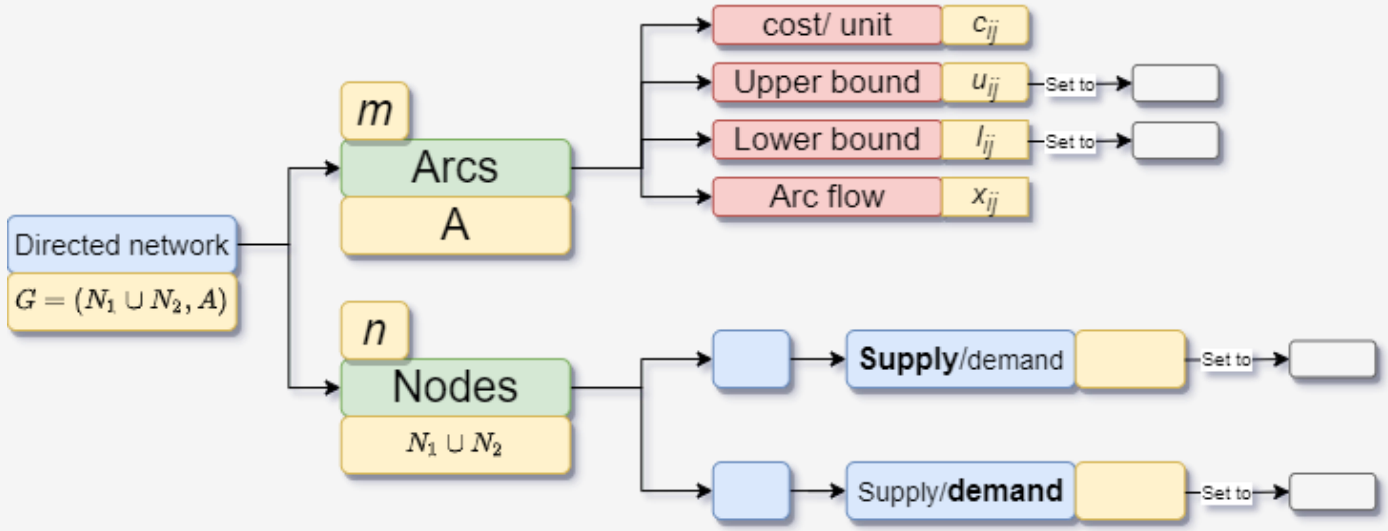
1	$l_{ij} = 0$ and $u_{ij} = 1$, because we want to assign one employee to one task
2	$x_{ij} = \{1,0\}$, because we want to know if it was assigned or not
3	$b(i) = 1$ for all i in N_1 because this are nodes that only gives
4	$b(i) = -1$ for all i in N_2 because this is a node that only receive

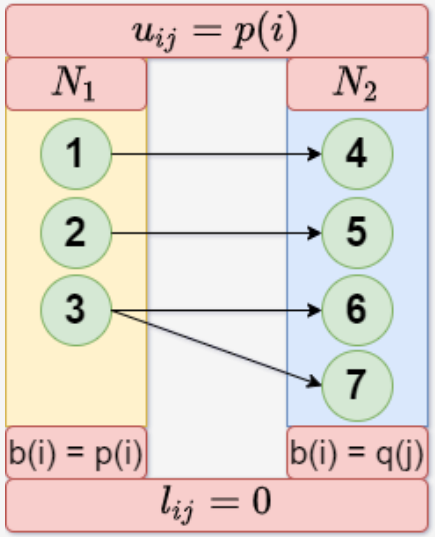
1.3 Network flow problems (problems 6)

Transportation problem

We want

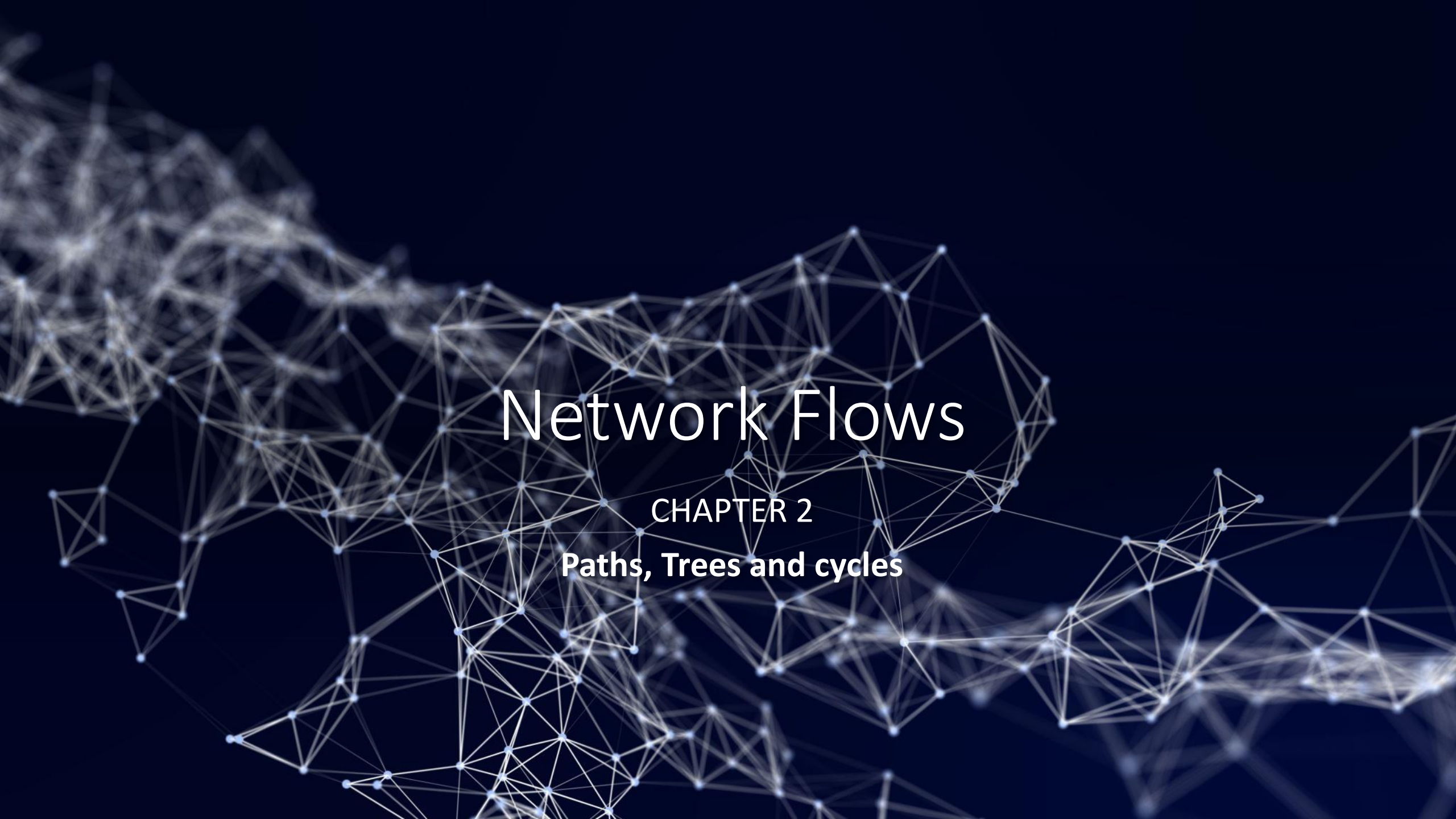
Consists of





We should note that

1	*Each node in* N_1 *is a supply node, because it only gives, and his production capacity is* **b(i) = p(i)**
2	*Each node in* N_2 *is a demand node, because it only receive*
3	Each arc *(i,j)* in *A*, has always *i* in N_1 and *J* in N_2
4	the capacity of each *arc (i, j)* in *A*, is all what the supply node can give u_{ij} = *p(i)*



Network Flows

CHAPTER 2

Paths, Trees and cycles

2) Paths, trees and cycles

Let me tell you

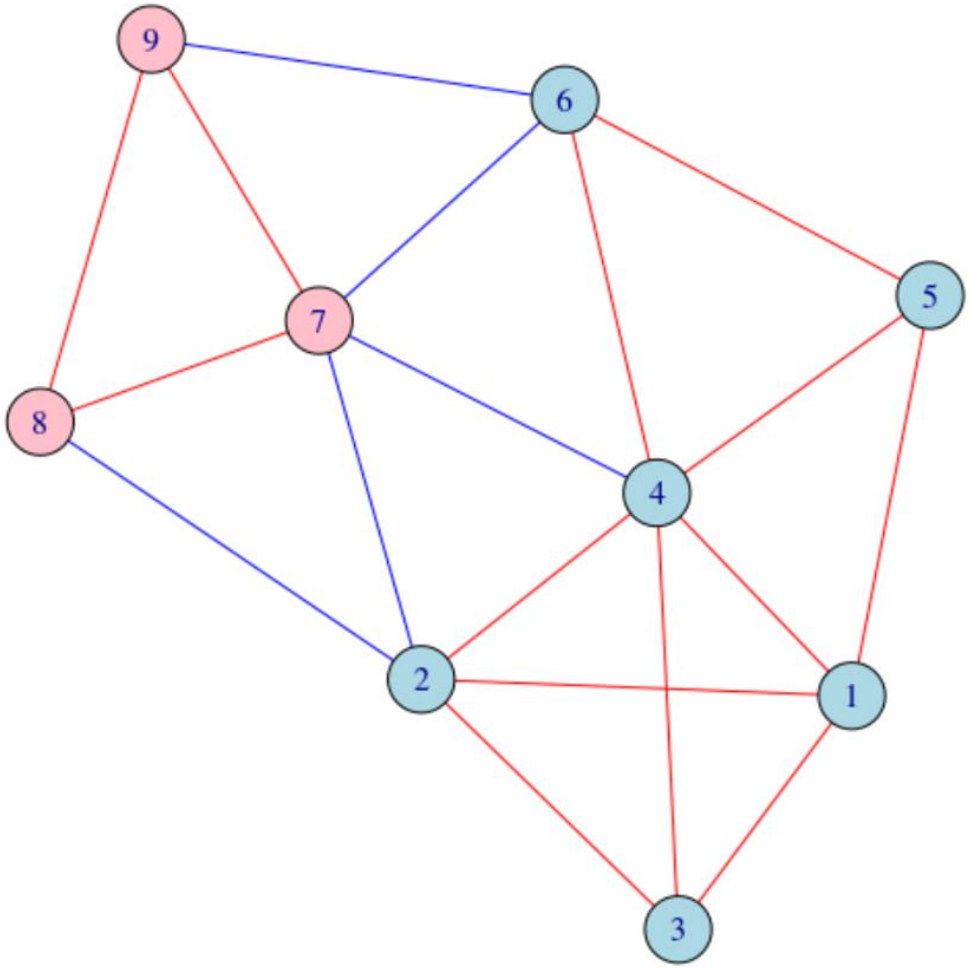
Graphs and networks arise everywhere and in a variety of alternative form, so thats why many have contributed to the evolution of networks flow

However

The theory lacks unity, because many authors have adopted a wide variety of conventions and notations

but we

Have chosen to adopt arcs and nodes instead of edges and vertices.
We make no distinction between graphs and networks

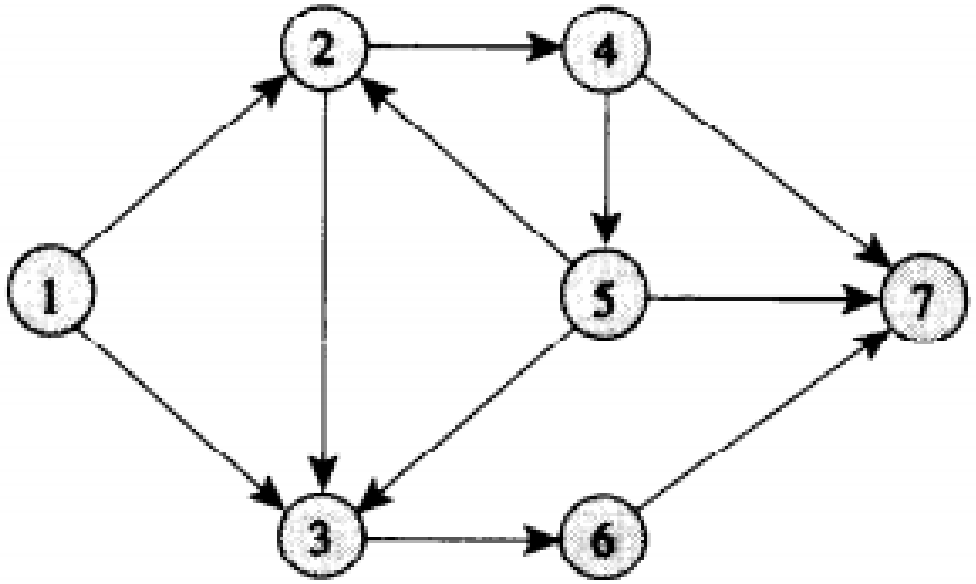


2.2 Notation and definitions

- **Directed Graph and networks**

1. A directed graph *G = (N, A)*

consists of a set *N* of ***nodes*** and a set *A* of **Arcs** whose elements are ordered in pairs of distinct nodes.



N = { 1, 2, 3, 4, 5, 6, 7}

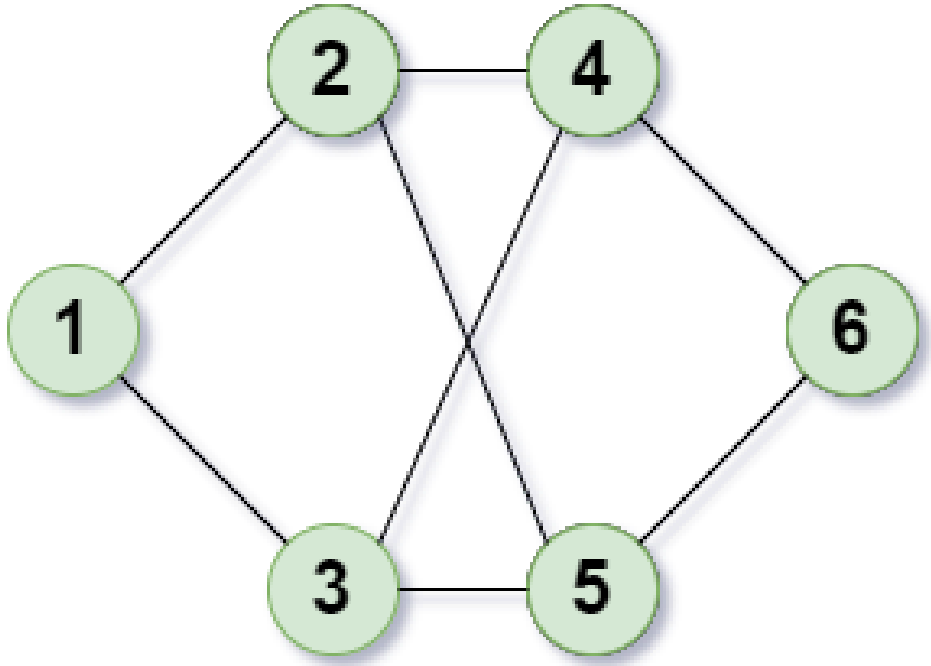
A = { (1, 2),(1, 3),(2, 3),(2, 4),(3, 6),(4, 5),(4, 7),(5 , 2),(5, 3),(5, 7),(6, 7)}

A directed network is a graph whose nodes have associated numerical values

We denote ***n*** as the number of **nodes** and ***m*** as the numbers of **arcs** in G

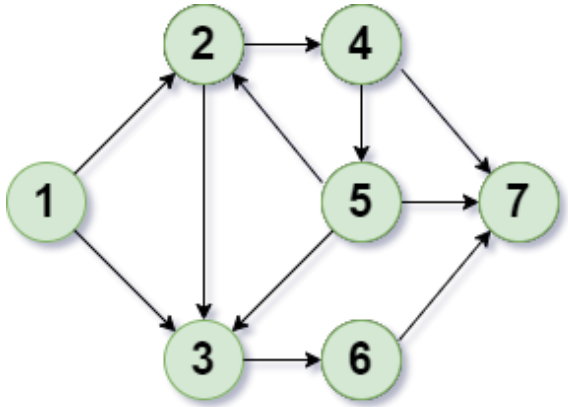
2.2 Notation and definitions

- **undirected Graph and networks**
- We define an undirected graph in the same manner as we define a direct one.
- The only difference is that the arcs are unordered pairs of distinct nodes
- We can refer to an ***arc*** joining the node pair as ***i*** and ***j*** as either ***(i , j)*** or ***(j, i).***
- And **arc** can be regarded as a **two-way street** with flow in both directions.



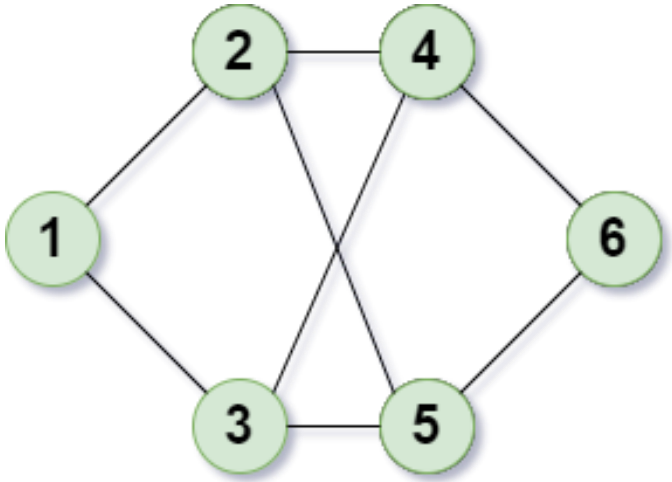
2.2 Notation and definitions (problems 2)

- **What is a directed graph and what is the notation.**



- **Complete the sets below**
- **N =**
- **A =**
- **n =**
- **m =**

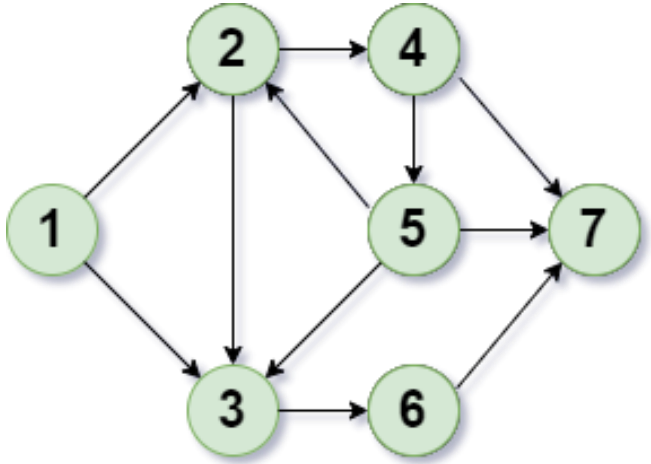
- What is the difference between directed graph and undirected graph.
- ¿How can we refer to an arc of an undirected graph?
- ¿How can be regarded?



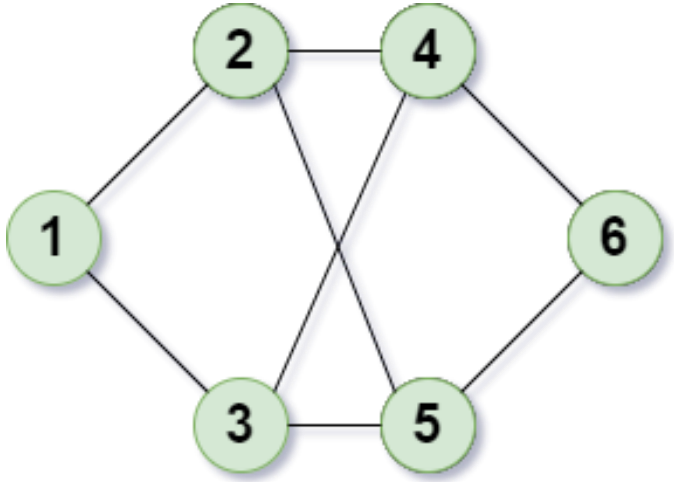
2.2 Notation and definitions (results 1)

- **What is a directed graph and what is the notation.**

- **Complete the sets below**
- *N = { 1, 2, 3, 4, 5, 6, 7}*
- A *= { (1, 2),(1, 3),(2, 3),(2, 4),(3, 6),(4, 5),(4, 7),(5 , 2),(5, 3),(5, 7),(6, 7)}*
- n = 7
- m = 11

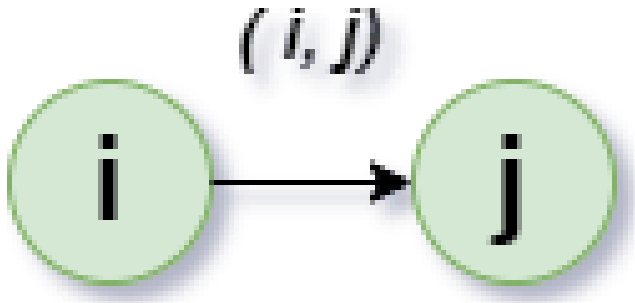


- What is the difference between directed graph and undirected graph.
  - The only difference is that the arcs are unordered pairs of distinct nodes
- ¿How can we refer to an arc of an undirected graph?
  - We can refer to an ***arc*** joining the node pair as ***i*** and ***j*** as either ***(i , j)*** or ***(j, i).***
- ¿How can be regarded?
  - And **arc** can be regarded as a **two-way street** with flow in both directions

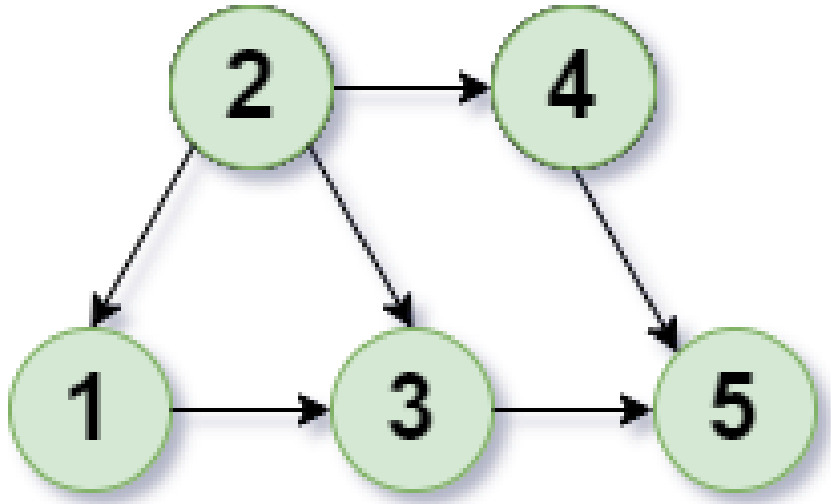


2.2 Notation and definitions

- **Tails and Heads:**
  - A directed ***arc (i, j)*** has two endpoints ***i*** and ***j***.
  - We refer to ***i*** as the ***tail***.
  - And node ***j*** as the ***head***.
  - Whenever an arc ***(i, j)*** ∈ ***A***, we say that node ***j*** is adjacent to node ***i.***



- **Degress:**
  - The **indegree** of a node is the number of incoming arcs of that node.
  - The **outdegree** is the number of its outgoing arcs.
  - The **degree** of a node is the sum of its degree and indegree
  - Example:
    - In node **3** his **indegree** is **2**, **outdegree** is **1**, and **degree** is **3.**

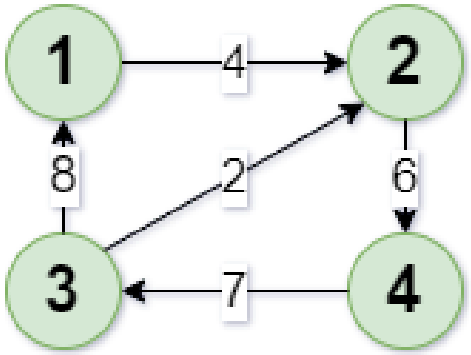


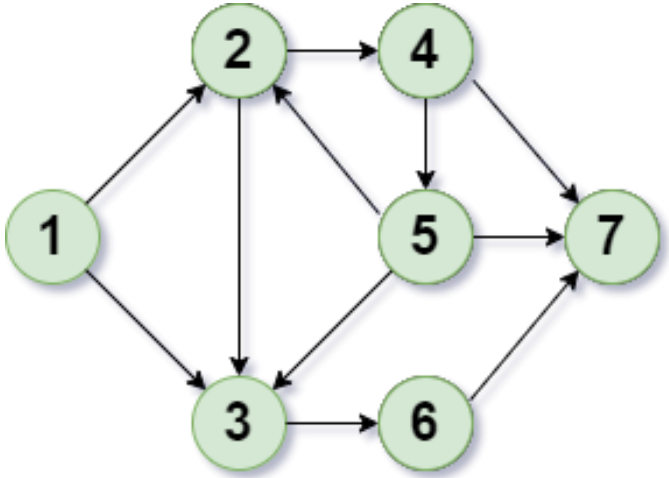
2.2 Notation and definitions (problems 3)

Arc	I , TAIL	J , Head	(i, j) ∈ A , we say
(1, 2)			That node _ is adjacent to _
(1, 3)			That node _ is adjacent to _
(2, 3)			That node _ is adjacent to _
(2, 4)			That node _ is adjacent to _
(3, 6)			That node _ is adjacent to _
(4, 5)			That node _ is adjacent to _
(4, 7)			That node _ is adjacent to _
(5, 2)			That node _ is adjacent to _
(5, 3)			That node _ is adjacent to _
(5, 7)			That node _ is adjacent to _
(6, 7)			That node _ is adjacent to _

Node	Indegrees	Outdegrees	Degrees
1			
2			
3			
4			
5			
6			
7			

Node	B(i)	Type
1		
2		
3		
4		



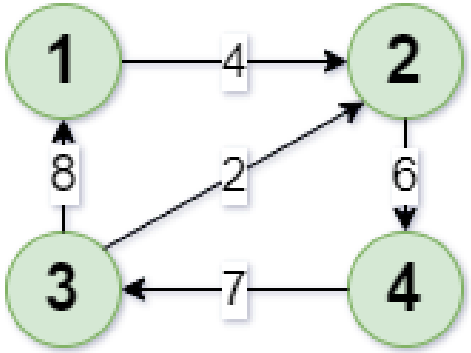


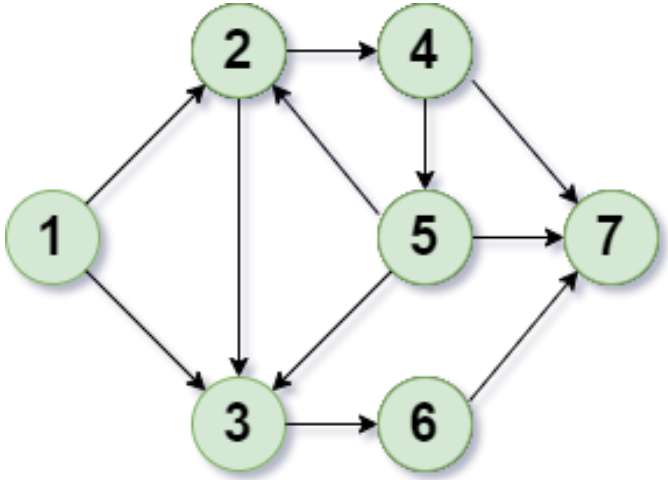
2.2 Notation and definitions (results 3)

Arc	I , TAIL	J , Head	(i, j) ∈ A , we say
(1, 2)	1	2	That node 2 is adjacent to 1
(1, 3)	1	3	That node 3 is adjacent to 1
(2, 3)	2	3	That node 3 is adjacent to 2
(2, 4)	2	4	That node 4 is adjacent to 2
(3, 6)	3	6	That node 6 is adjacent to 3
(4, 5)	4	5	That node 5 is adjacent to 4
(4, 7)	4	7	That node 7 is adjacent to 4
(5, 2)	5	2	That node 2 is adjacent to 5
(5, 3)	5	3	That node 3 is adjacent to 5
(5, 7)	5	7	That node 7 is adjacent to 5
(6, 7)	6	7	That node 7 is adjacent to 6

Node	Indegrees	Outdegrees	Degrees
1	0	2	2
2	2	2	4
3	3	1	4
4	1	2	3
5	1	3	4
6	1	1	2
7	3	0	3

Node	B(i)	Type
1	-4	Demand
2	0	Transshipment
3	3	Supply
4	1	Supply





2.2 Notation and definitions

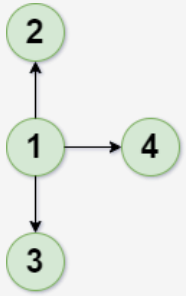
Adjacency list

can be two concepts denoted by A(i)

1 **the arc adjacency list**

is the set of arcs emanating from that node

$$A(i) = \{(i, j) \in A : j \in N\}$$

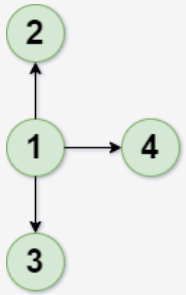


$$A(i) = \{2, 3, 4\}$$

2 **the node adjacency list**

is the set of nodes adjacent from that node

$$A(i) = \{j \in N : (i, j) \in A\}$$



$$A(i) = \{(1,2), (1,3), (1,4)\}$$

Is worth noticing that ***|A(i)|*** = ***outdegree*** of a node ***i***, *which means:*

$$\sum_{i \in N} |A(i)| = m$$

Multiarc and loops

1 **Multiarc**

Are two or more arc with the same tail and head nodes

2 **Loop**

Is and arc whose tail node is the same as its head node

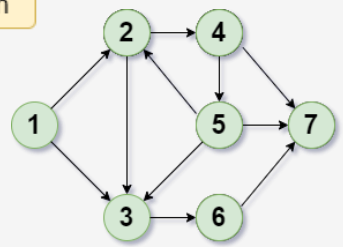
$$i = 1, j = 1$$

2.2 Notation and definitions

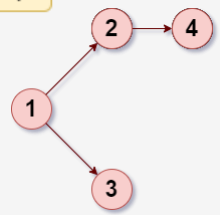
Subgraph

Denoted by G', it means that it have some of the nodes and some of the arcs $N' \subseteq N, A' \subseteq A$

Graph



Subgraph

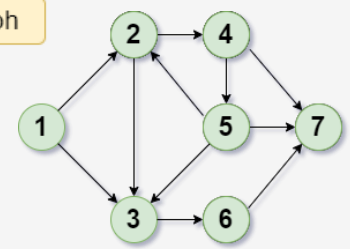


There are some types of subgraphs

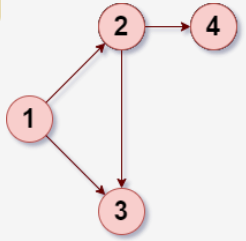
1 **Induced subgraph**

G' is the subgraph of G induced by N', if all A' contains all arcs of A with both endpoints in N' this is denoted as A'(N').

Graph



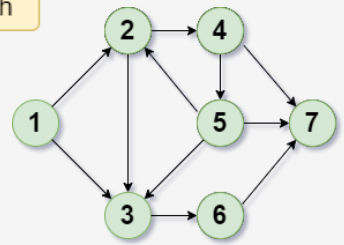
Subgraph



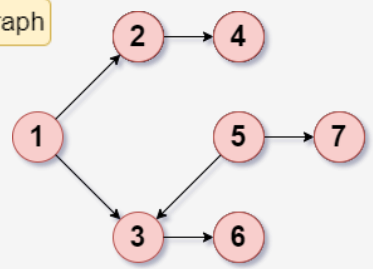
2 **Spanning subgraph**

G' is a spanning subgraph of G if N' = N, and it have some of the arcs, that means all nodes but not all arcs.

Graph



Subgraph

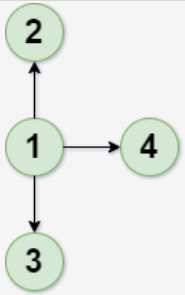


2.2 Notation and definitions (problems 7)

Adjacency list

1 **the arc adjacency list**

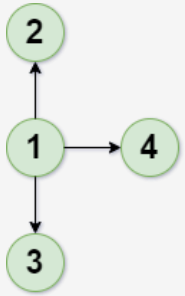
$$A(i) = \{(i,j) \in A : j \in N\}$$



$$A(i) = \{\}$$

2 **the node adjacency list**

$$A(i) = \{j \in N : (i,j) \in A\}$$



$$A(i) = \{\}$$

Is worth noticing that ***|A(i)|*** = ***outdegree*** of a node ***i***, *which means:*

Multiarc and loops

1 **Multiarc**

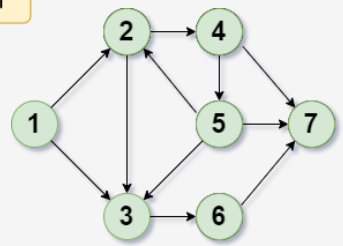
2 **Loop**

$$i =, j =$$

2.2 Notation and definitions (problems 8)

Denoted by G', it means that it have some of the nodes and some of the arcs

Graph



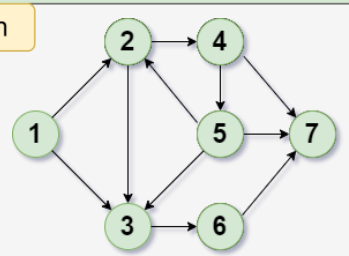
Subgraph

There are some types of subgraphs

1

G' is the subgraph of G induced by N', if all A' contains all arcs of A with both endpoints in N' this is denoted as A'(N').

Graph

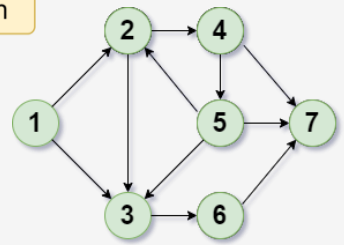


Subgraph

2

G' is a spanning subgraph of G if N' = N, and it have some of the arcs, that means all nodes but not all arcs.

Graph



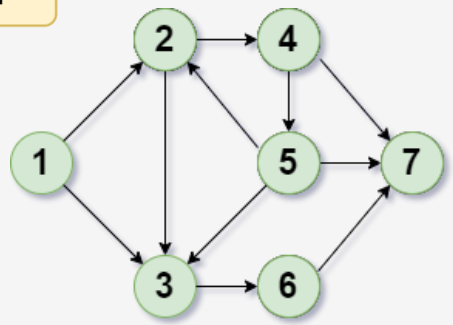
Subgraph

2.2 Notation and definitions

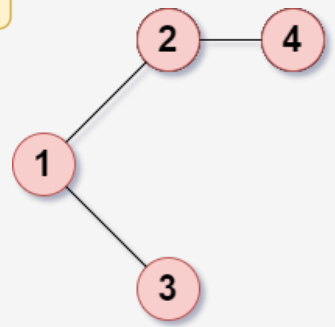
Walks

Is a sequence of nodes, that can be repeated, satisfying the property that $W = (i_k, i_{k+1}) \in A, (i_{k+1}, i_k) \in A$

Graph



Walk

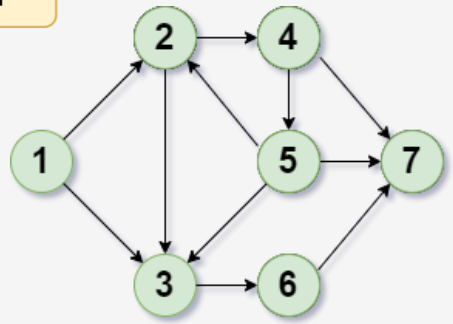


$W = 1 - 3 - 1 - 2 - 4 - 2$

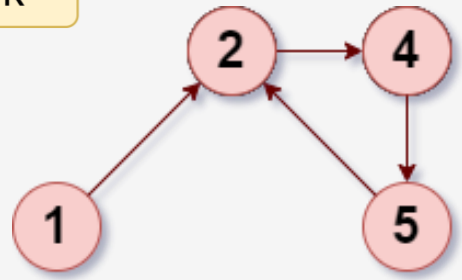
Directed walks

Is a sequence of nodes, that can be repeated, taking the direction in consideration

Graph



Walk



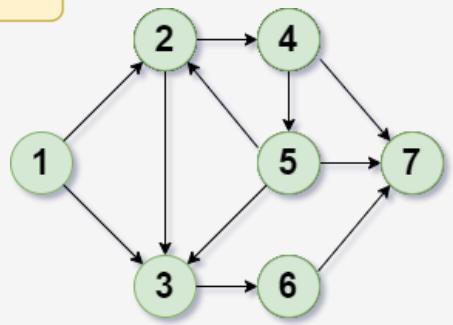
$W = 1 - 2 - 4 - 5 - 2$

2.2 Notation and definitions

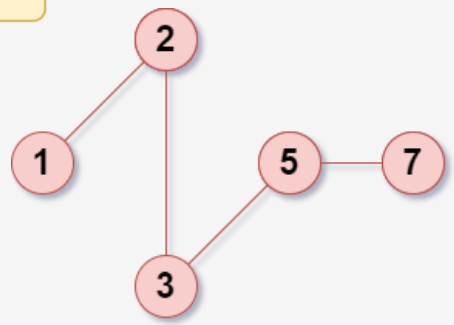
Path

Is a walk ,without any repetition on nodes

Graph



Path

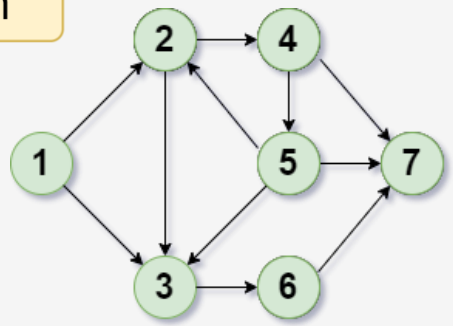


$W = 1 - 2 - 3 - 5 - 7$

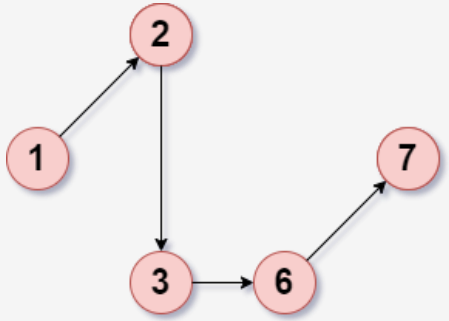
Directed Path

Is a directed walk, without any repetition on nodes

Graph



Directed path



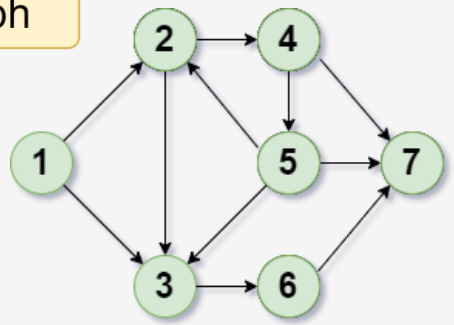
$W = 1 - 2 - 3 - 6 - 7$

2.2 Notation and definitions (problems 9)

Is a sequence of nodes, that can be repeated, satisfying the property that

$W = (i_k, i_{k+1}) \in A, (i_{k+1}, i_k) \in A$

Graph

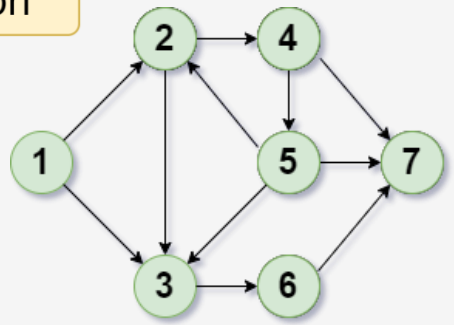


Walk

$W = 1 - 3 - 1 - 2 - 4 - 2$

Is a sequence of nodes, that can be repeated, taking the direction in consideration

Graph



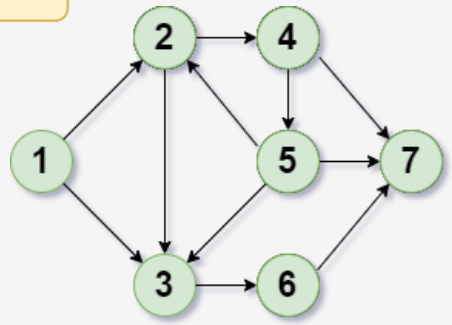
directed Walk

$W = 1 - 2 - 4 - 5 - 2$

2.2 Notation and definitions (problems 10)

Is a walk ,without any repetition on nodes

Graph

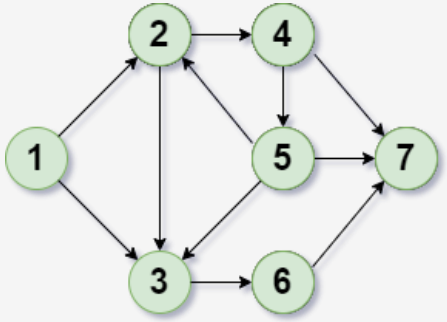


Path

$W = 1 - 2 - 3 - 5 - 7$

Is a directed walk, without any repetition on nodes

Graph



Directed path

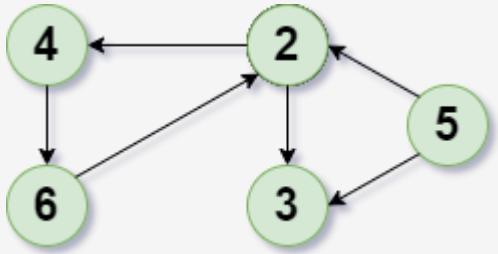
$W = 1 - 2 - 3 - 6 - 7$

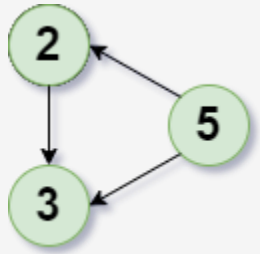
2.2 Notation and definitions

1	Cycle	$i_1 - i_2 - \ldots - i_r - i_1$	2 - 5 - 3 - 2

Its a sequence of nodes with no repetition that ends, where its starts, without direction

Graph

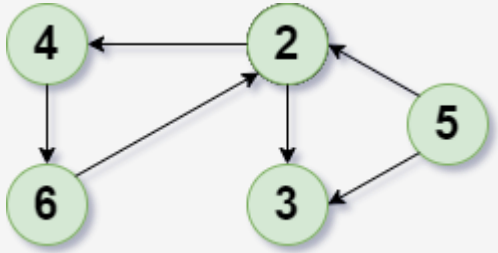


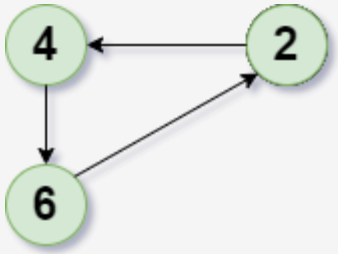


2	Directed cycle	$i_1 - i_2 - \ldots - i_r - i_1$	2 - 4 - 6 - 2

Its a sequence of nodes with no repetition that ends, where its starts, with direction

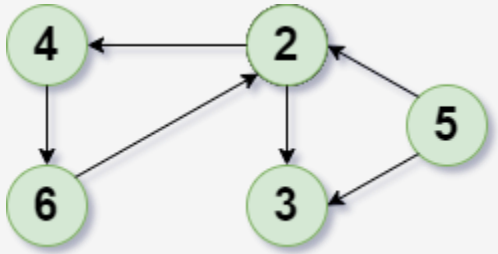
Graph



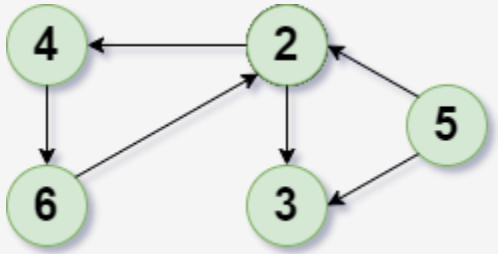


2.2 Notation and definitions

1			2 - 5 - 3 - 2
	Its a sequence of nodes with no repetition that ends, where its starts, without direction		
	Graph		



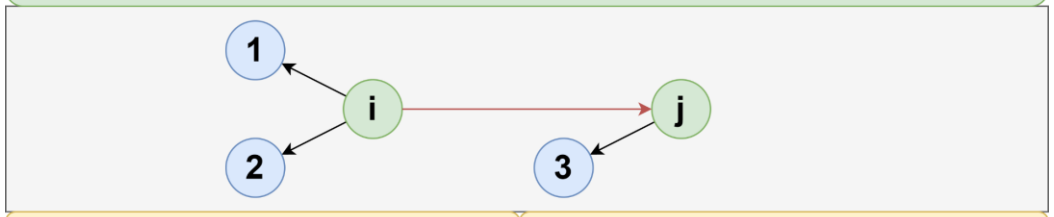
2			2 - 4 - 6 - 2
	Its a sequence of nodes with no repetition that ends, where its starts, with direction		
	Graph		



Connectivity

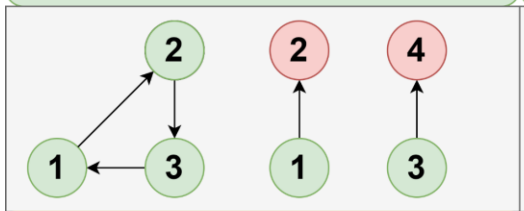
Nodes are connected

i and **j** are connected if there exists at least one path from **i** to **j**



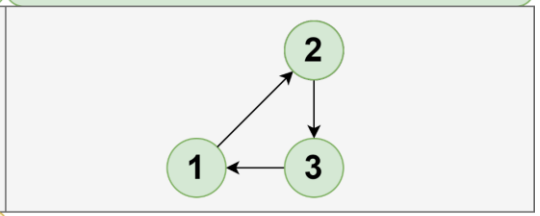
graph is connected

if every **pair of nodes** in **G** is **connected**, otherwise is disconnected



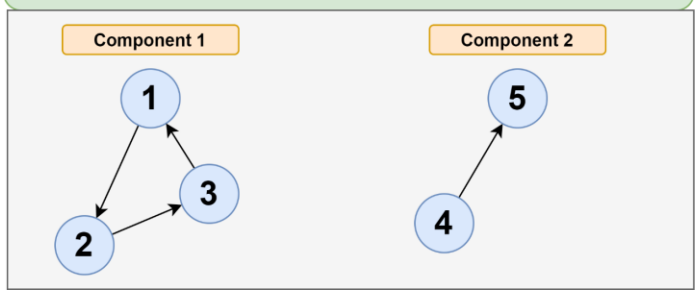
graph is strongly connected

if there exists a **directed path** from every node to every other node



¿What is a component?

is a connected subgraph in a disconnected subgraph



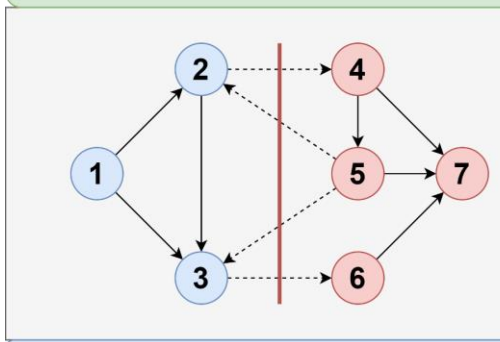
Cut

denotes as $[S, \overline{S}]$

is a partition of the node set N into two parts, S and $\overline{S} = N - S$

consisting

in a set of arcs that have one endpoint in S and another endpoint in $\overline{S}$



Example

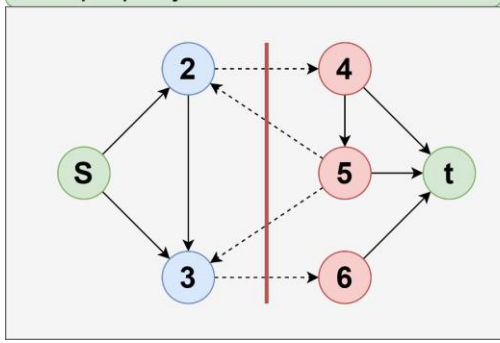
$[S, \overline{S}] = (2,4)(5,4)(5,3)(3,6)$

s-t Cut

is a partition of the node set N into two parts, S and $\overline{S} = N - S$

satisfying

the property that $s \in S$ and $t \in \overline{S}$

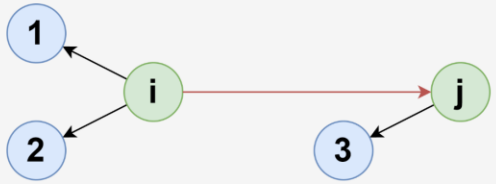


Example

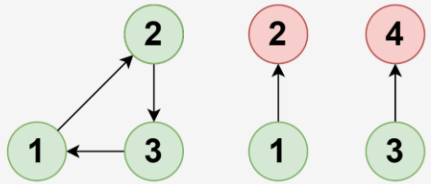
$[S, \overline{S}] = (2,4)(5,4)(5,3)(3,6)$

Connectivity

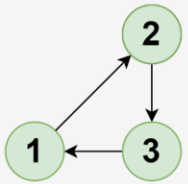
Nodes are connected



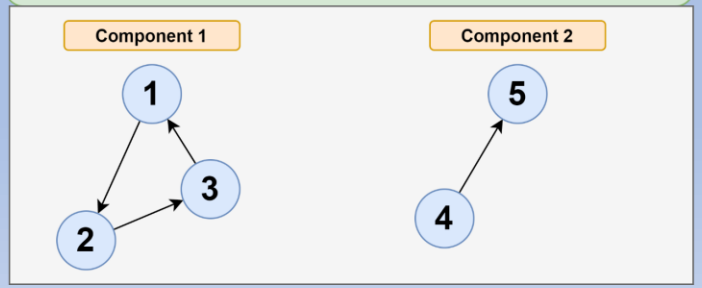
graph is connected



graph is strongly connected



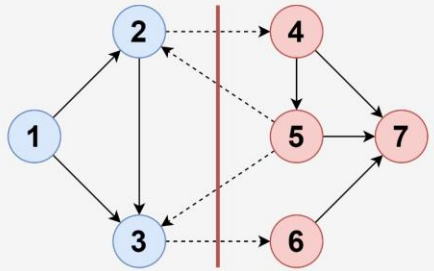
¿What is a component?



Cut

denotes as

consisting



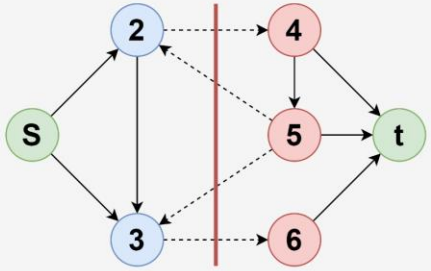
Example

$[S, \overline{S}] =$

s-t Cut

is a partition of the node set N into two parts, S and $\overline{S} = N - S$

satisfying



Example

$[S, \overline{S}] = (2,4)(5,4)(5,3)(3,6)$

Network transformations

Frequently, we require network transformations to simplify a network or to a network problem in a standard form required by a computer code.

We assume

that the network problem is a minimum cost flow problem

1. **Undirected arcs to directed arcs**

   Sometimes a minimum cost flow problem contain undirected arcs

2. **Removing nonzero lower bounds**

   we use this method to to remove lower bounds different of 0

3. **Arc reversal**

   we use this method to remove arcs with negative costs

4. **Removing Arc capacities**

   If and arc has a positive capacity we can make them uncapacitated or infinite

5. **Node Splitting**

   we use this method if our nodes have capacities

6. **Reduced costs**

   Method to work with a vector $\pi(i)$ in each node

7. **Residual networks**

   used to measure flow in terms of incremental flow about some given feasible solution

2.4 Network transformations

Red Residual

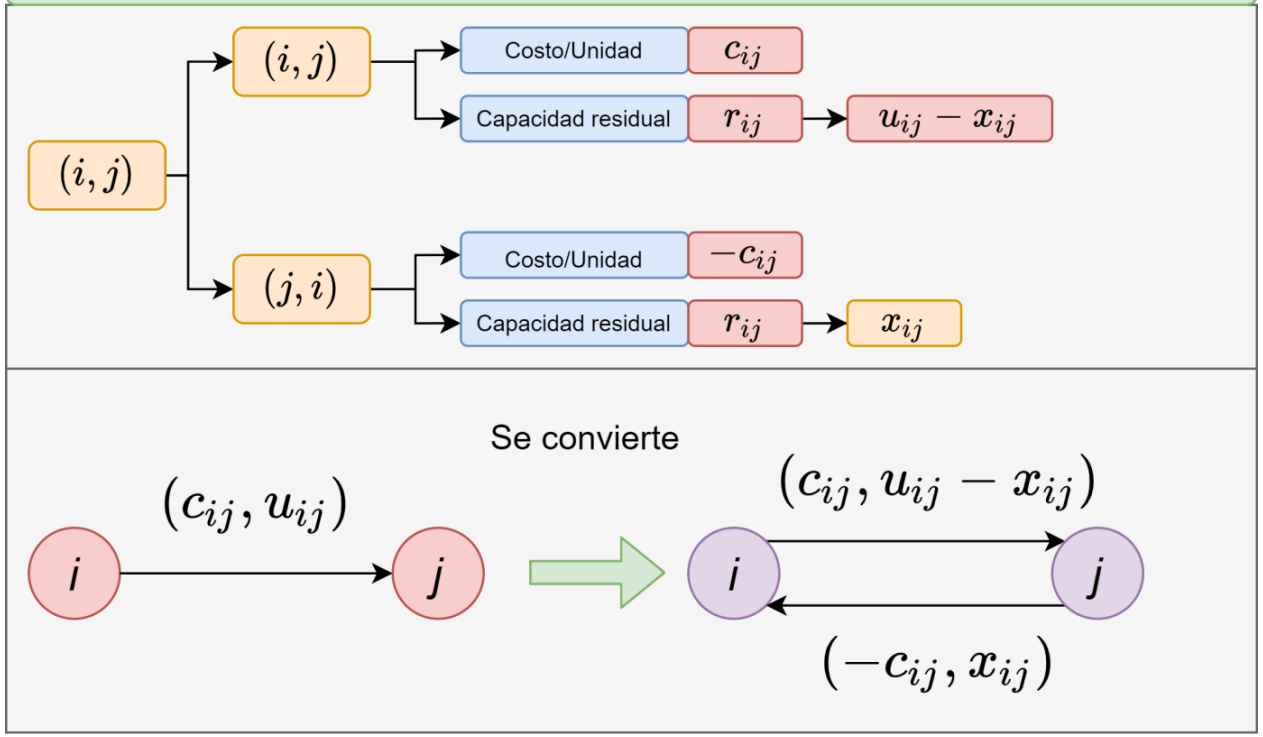
Es una red que se compone de arcos con un **flujo residual** los cuales se obtienen de un arco una vez que halla pasado flujo por este

Se denota como

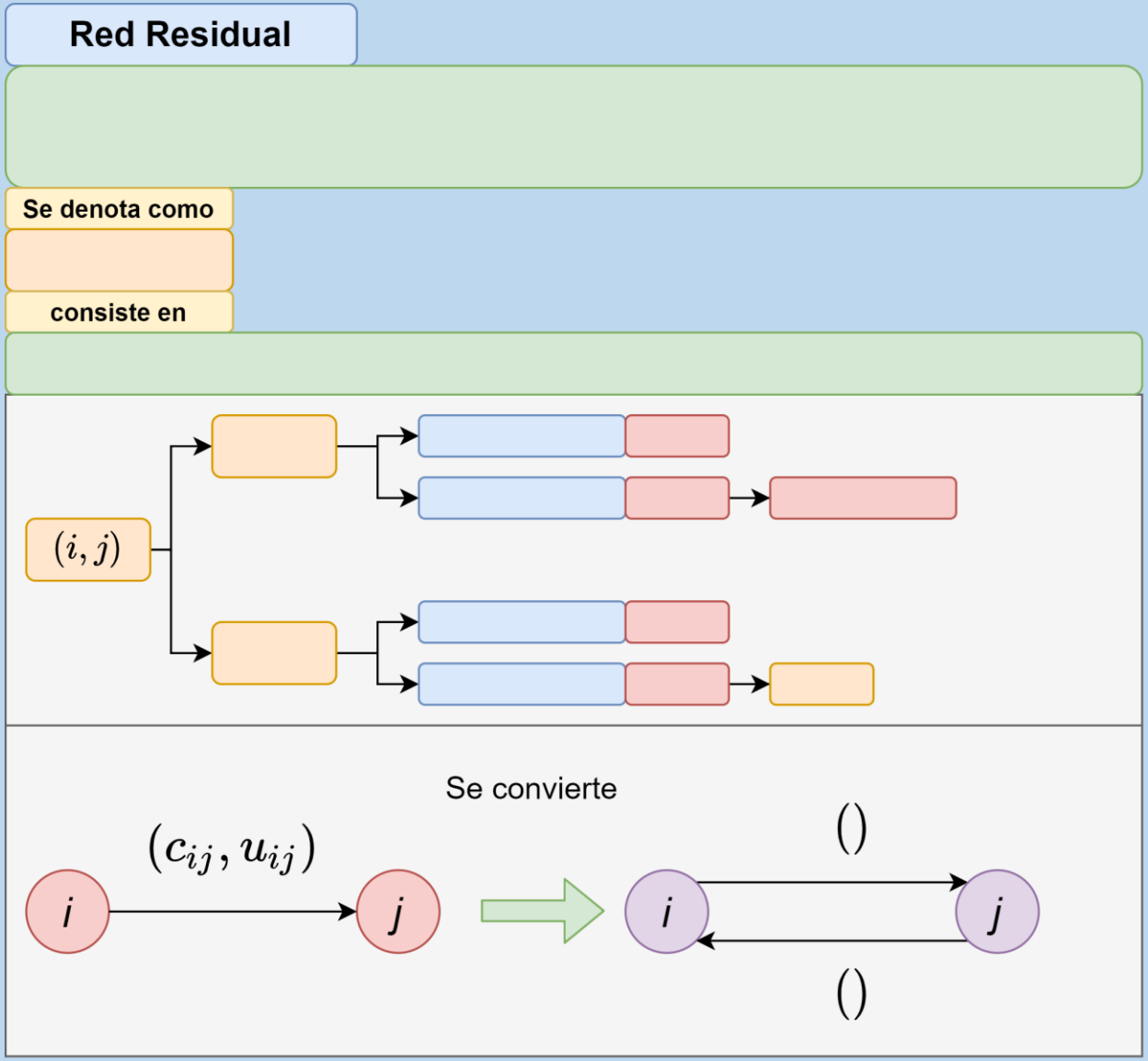
$G = (x^*)$

consiste en

reemplazar cada arco ***(i,j)*** en ***A*** por 2 arcos ***(i,j)*** e ***(j,i)***



2.4 Network transformations



Reduced costs

We associate with each node $i \in N$ a number $\pi(i)$

called

potential of that node

we define the reduced cost of an arc *(i,j)*

$$C_{ij}^{\pi} = C_{ij} - \pi(i) + \pi(j)$$

It is important

to understand the relationship between $z(\pi)$ and $z(0)$

denoted as

$$z(\pi) = \sum_{(i,j)\in A} c_{ij}^{\pi} x_{ij}$$

$$z(0) = \sum_{(i,j)\in A} c_{ij} x_{ij}$$

we have

$$z(0) - z(\pi) = \sum_{i \in N} \pi(i) b(i) = \pi b$$

Property 2.4

Minimum cost flow problems with arc cost c_{ij} or c_{ij}^{π} have the same optimal solutions, moreover, $z(\pi) = z(0) - \pi b$

Network Flows

CHAPTER 3

Network Flows

CHAPTER 4-5

Shortest path algorithms

1 Label-setting algorithm

Designate one label as permanent (optimal) each iteration

Applicable

problems with acyclic networks

problems with nonnegative arc lengths

is better at

efficient in worst-case scenarios

Examples

Dijkstra's algorithm	$O(n^2)$
Dial's implementation	$O(m + nC)$
d-Heap implementation	$O(m \log_d n)$
Fibonacci heap implementation	$O(m + n \log n)$
Radix heap implementation	$O(m + n \log(nC))$

2 Label-correcting algorithm

Consider all labels as temporary until final step, when all became permanent

Applicable

All classes of problems including with negative arc length

is better at

giving more algorithm flexibility

Examples

Generic label-correcting algorithm	$O(\min\{n^2mC, m2^n\})$
Modified label-correcting algorithm	$O(\min\{nmC,m2^2\})$
FIFO implementation	$O(nm)$
Dequeue implementation	$O(\min\{nmC,m2^n\})$

All pair label-correcting algorithm

Repeated shortest path algorithm	$O(nS(n,m,C))$
Floyd-Warshall algorithm	$O(n^3)$

Network Flows

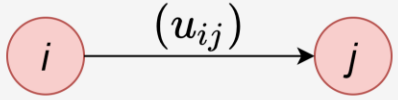
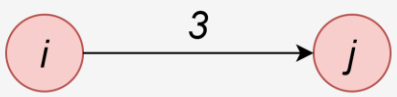
CHAPTER 6

Maximum flow problems

INTRODUCTION

Maximum flow problem

In a network with capacities, we wish to send as much flow as possible between two nodes

Denoted as	Example
$MaxF = u_{ij}$ 	$MaxF = 3$ 

Focus

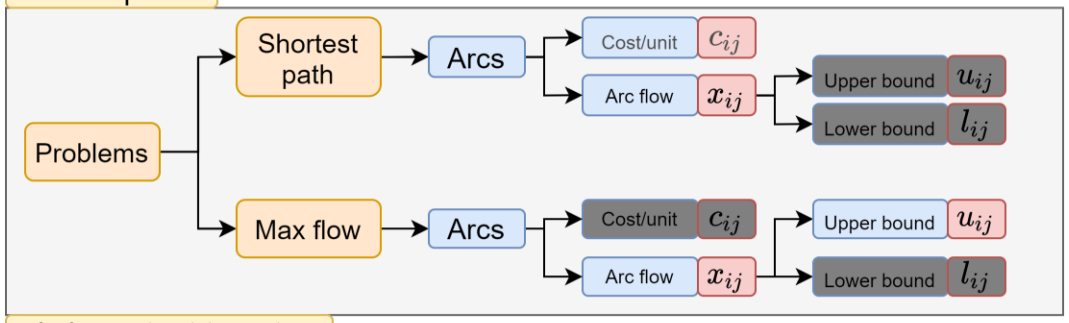
Shortest fast problem

the focus on the shortest path problem is **model arc cost**, not arc capacities.

Maximum flow problem

the focus on the maximum flow problem is **model arc capacities**, not arc costs.

Map



is important to note

that if we resolve the Maximum flow problem, we resolve to the **minimum cut problem**

which is

a special case of the strong duality theorem of linear programming.

Algorithms

Augmenting path algorithm

they maintain the flow of restrictions for each node in the networks except for s and t

these algorithms increase flow through the possible paths from **s** to **t**

Example

label Algorithm

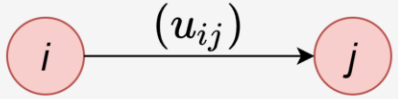
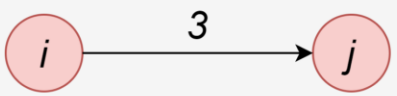
Preflow-push algorithm

these flood the network so that some nodes have excesses

these gradually ease the flow of the nodes by sending flow to **s** or **t**

INTRODUCTION

Maximum flow problem

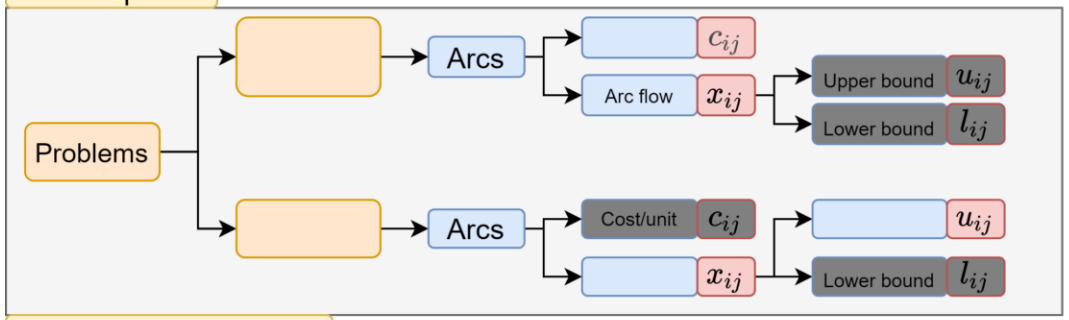
Denoted as	Example
$MaxF = u_{ij}$ 	$MaxF =$ 

Focus

Shortest fast problem

Maximum flow problem

Map



is important to note

which is

Algorithms

these flood the network so that some nodes have excesses

these algorithms increase flow through the possible paths from **s** to **t**

Example

Network Flows

CHAPTER 9

Minimum cost flows: basic algorithms

Minimum cost flow problem review

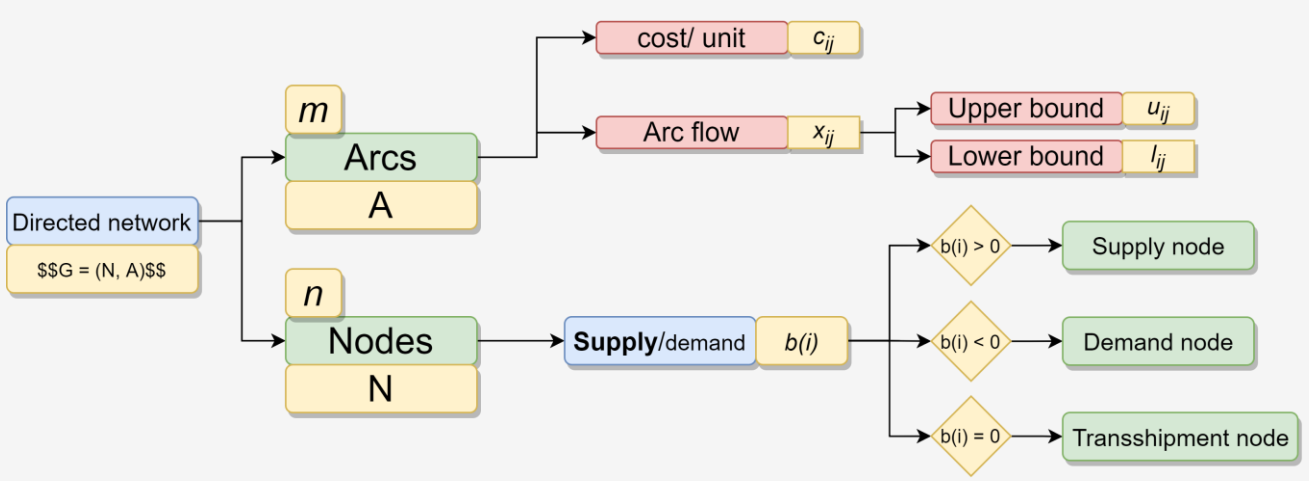
The shortest path problem and the maximum flow problem are special cases of this problem.

The algorithms

are not as efficient as those for the shortest path problem and the maximum flow problem.

However

They are still quite efficient and indeed are among the most efficient algorithms know for solving large-scale optimization problems.



Variables

1	Cost/Unit	c_{ij}
2	Arc flow	x_{ij}

Formula

Minimize

$$Min \quad z(x) = \sum_{(i,j)\in A} c_{ij}x_{ij}$$

Subject to

1	$l_{ij} \le x_{ij} \le u_{ij}$	for all	$(i,j) \in A$
2	$b(i) = \sum x_{ij} - \sum x_{ji}$	for all	$i \in N$
3	$\sum_{i=1}^{n} b(i) = 0$	for all	

Assumpsions for the minimum cost flow problem

Assumpsion 9.1

All data (cost, supply/demand, and capacity) are integers.

Assumpsion 9.2

The network is directed

if is not we can transform it to a directed one

Assumpsion 9.3

the supply/demand satisfy the condition $\sum_{i \in N} b(i) = 0$ which means the solution is feasible

Assumpsion 9.4

we assume that network G contain an uncapacitated directed path (which means each arc have infinite capacity) between every pair of nodes.

Assumpsion 9.5

all cost are nonnegative

if they are we can use an arc reversal transformation, however it needs to have finite capacity

Red Residual

Es una red que se compone de arcos con un **flujo residual** los cuales se obtienen de un arco una vez que halla pasado flujo por este

Se denota como

$G = (x^*)$

consiste en

reemplazar cada arco ***(i,j)*** en ***A*** por 2 arcos ***(i,j)*** e ***(j,i)***