

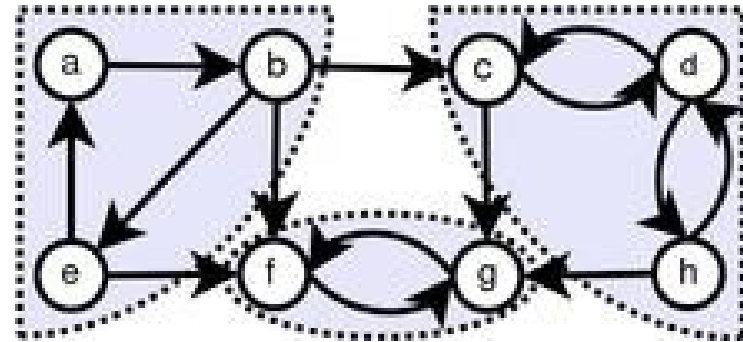
Graph Theory

– Course 4 –

Chapter 4 : Problems of Flows (1/1)



Summary



Goals of course

- Define the concepts of cycle /Co cycle, elementary (Cycle/ Co cycle) , Cocircuit, number cyclomatic/cocyclomatic, Co-TREE and Co-forest.
- Define the concept of flow In a graph.
- Apply an algorithm of research of a maximum flow.



Section 1 :Definitions

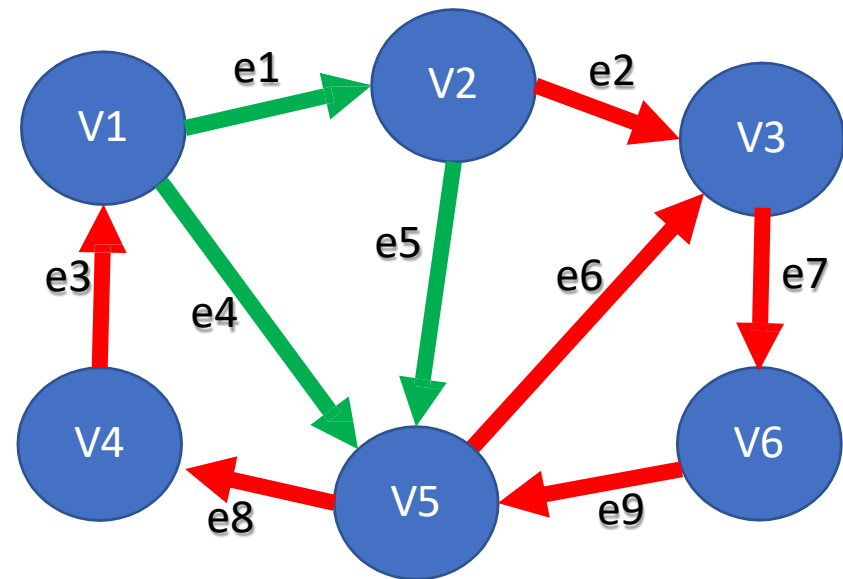
Cycle

$G1 = (\{V1, V2, V3, V4, V5, V6\}, \{e1, e2, e3, e4, e5, e6, e7, e8, e9\})$

- Cycle

A cycle of length h is a succession of h edges, all different, each intermediate edge having an end in common with the following edge.

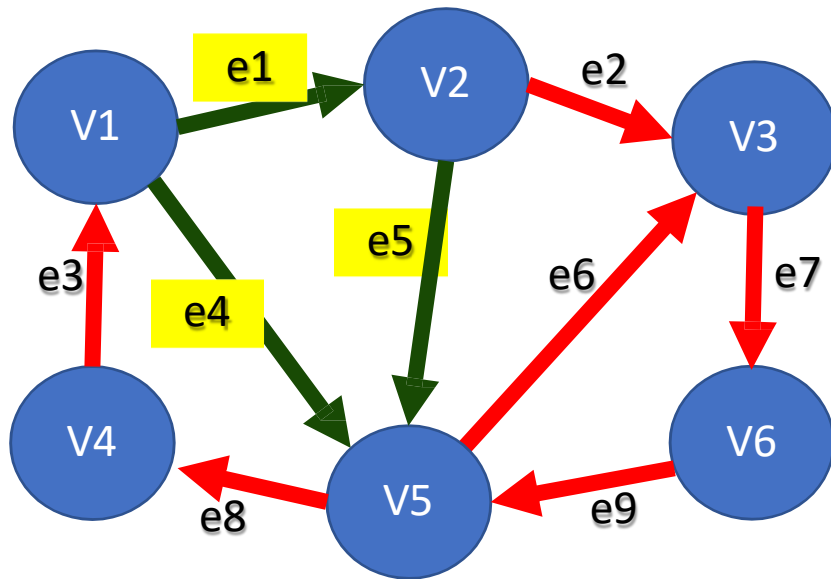
The initial and the final edges also having their free end in common.



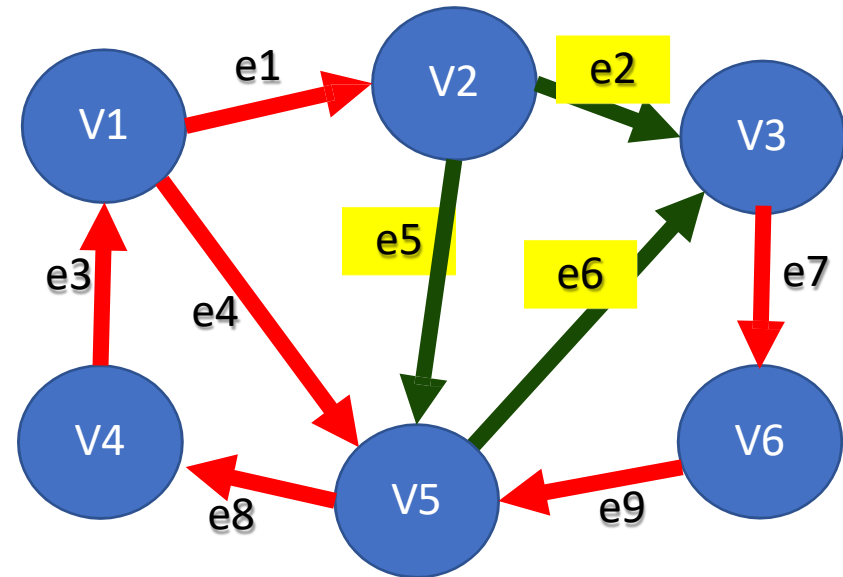
$C1 = (e1, e5, e4)$ is a cycle

Cycle

$G1 = (\{V1, V2, V3, V4, V5, V6\}, \{e1, e2, e3, e4, e5, e6, e7, e8, e9\})$



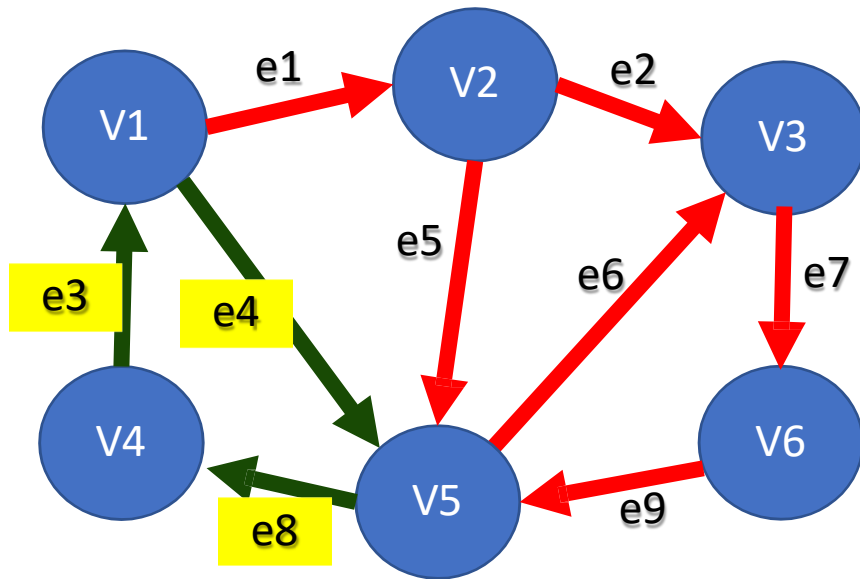
$C1 = (e1, e5, e4)$ is a cycle of length 3



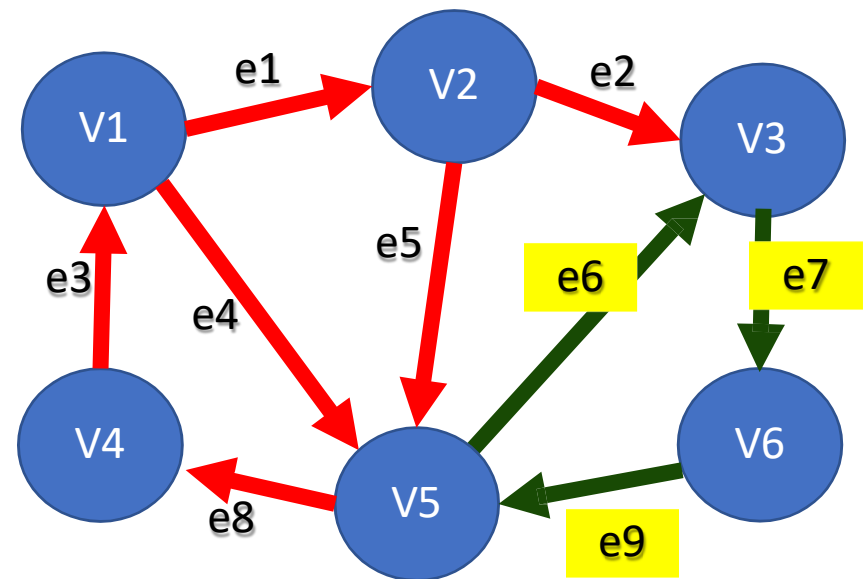
$C2 = (e2, e6, e5)$ is a cycle of length 3

Cycle

$G1 = (\{V1, V2, V3, V4, V5, V6\}, \{e1, e2, e3, e4, e5, e6, e7, e8, e9\})$



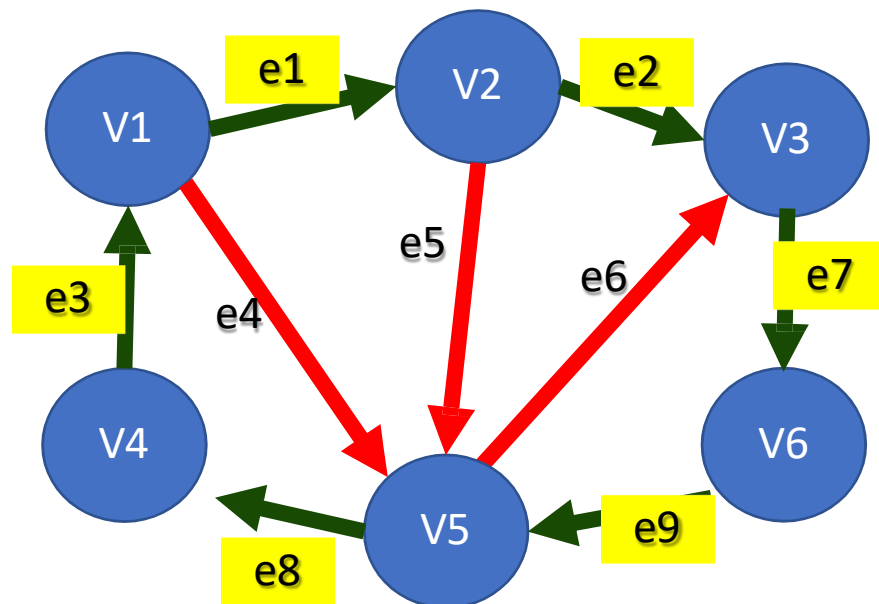
$C3 = (e3, e4, e8)$ is a cycle of length 3



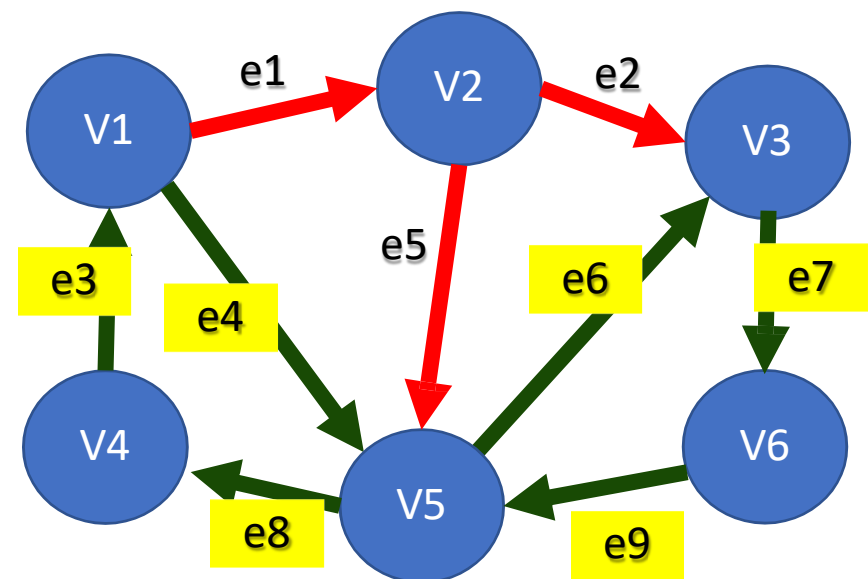
$C4 = (e7, e9, e6)$ is a cycle of length 3

Cycle

$G1 = (\{V1, V2, V3, V4, V5, V6\}, \{e1, e2, e3, e4, e5, e6, e7, e8, e9\})$



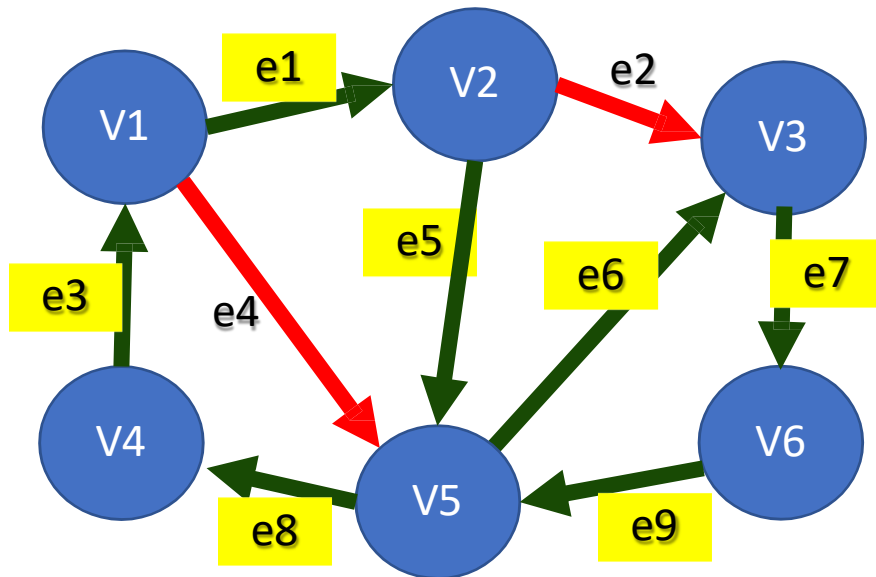
$C5 = (e1, e2, e7, e9, e8, e3)$
is a cycle of length 6



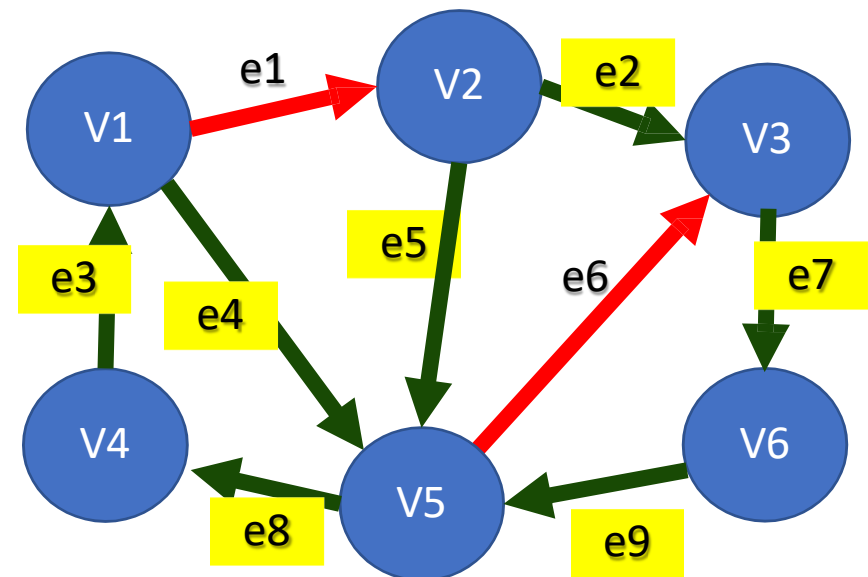
$C6 = (e3, e4, e9, e7, e6, e8)$
is a cycle of length 6

Cycle

$G1 = (\{V1, V2, V3, V4, V5, V6\}, \{e1, e2, e3, e4, e5, e6, e7, e8, e9\})$



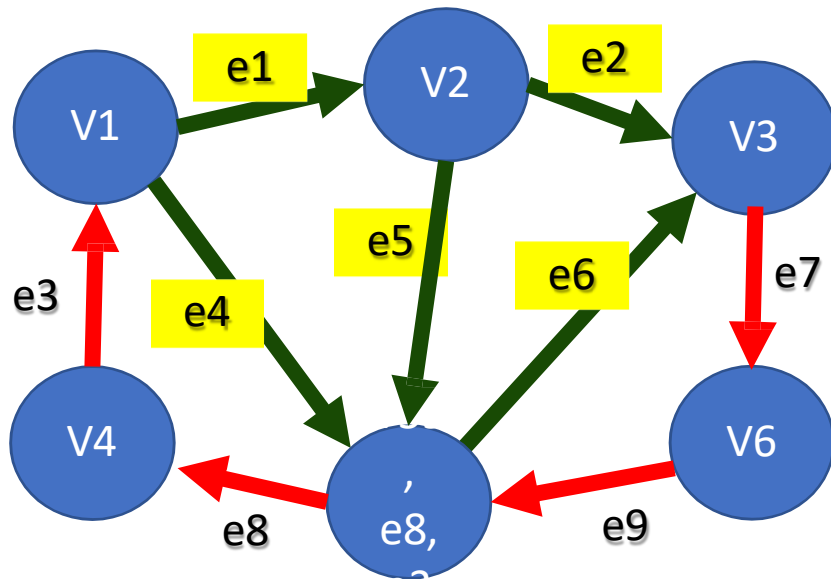
$C7 = (e3, e1, e5, e9, e7, e6, e8)$
is a cycle of length 7



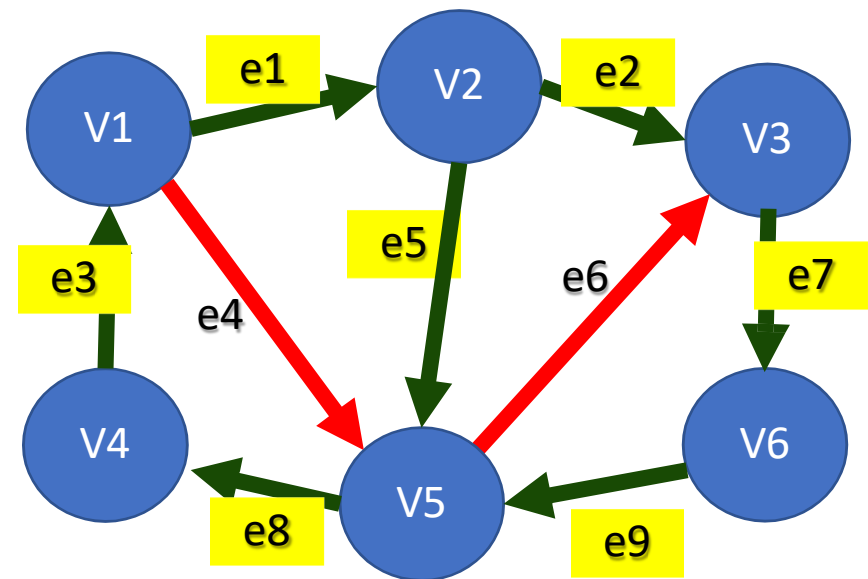
$C8 = (e3, e4, e9, e7, e2, e5, e8)$
is a cycle of length 7

Cycle

$G1 = (\{V1, V2, V3, V4, V5, V6\}, \{e1, e2, e3, e4, e5, e6, e7, e8, e9\})$



$C7 = (e1, e5, e6, e2, e5, e4)$
is not not a cycle



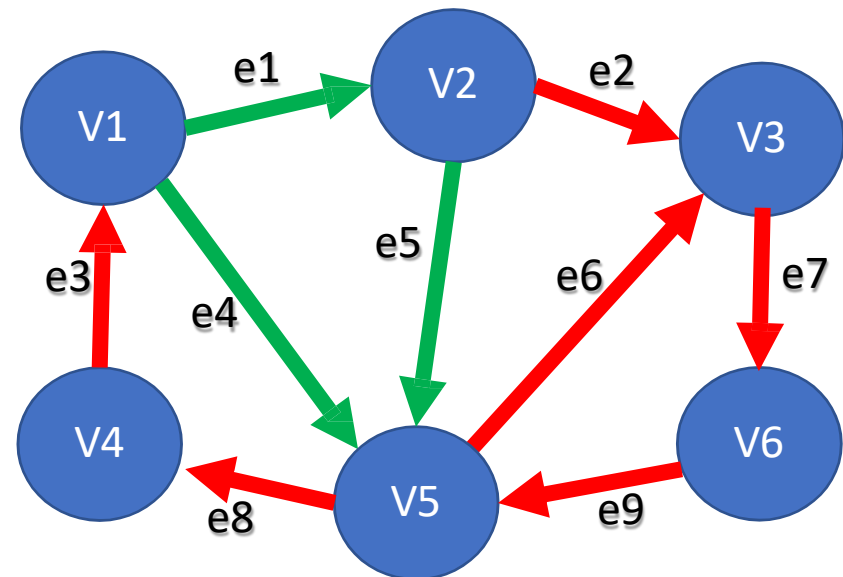
$C8 = (e3, e1, e5, e9, e7, e2, e5, e8)$
is not not a cycle

Elementary cycle

$$G1 = (\{V1, V2, V3, V4, V5, V6\}, \{e1, e2, e3, e4, e5, e6, e7, e8, e9\})$$

- Elementary cycle

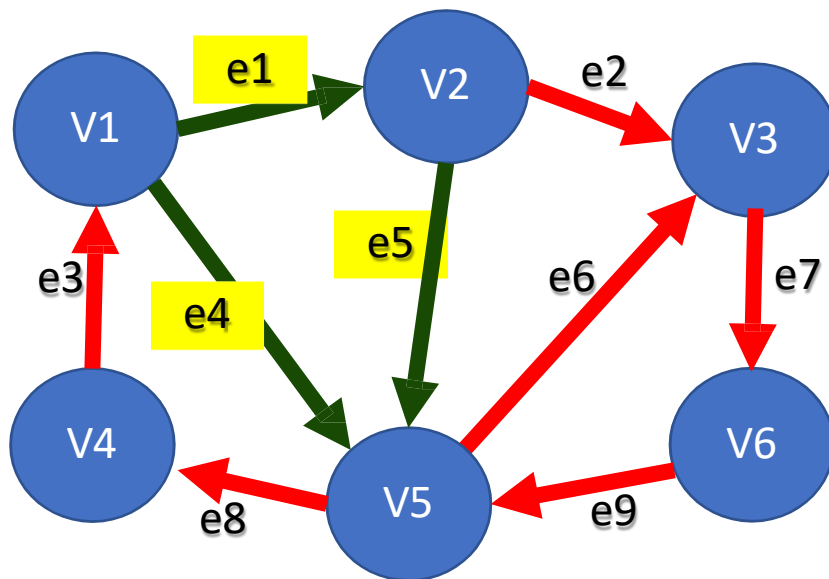
A cycle is said to be "**elementary**" if, while going through it, we do not encounter the same vertex several times.



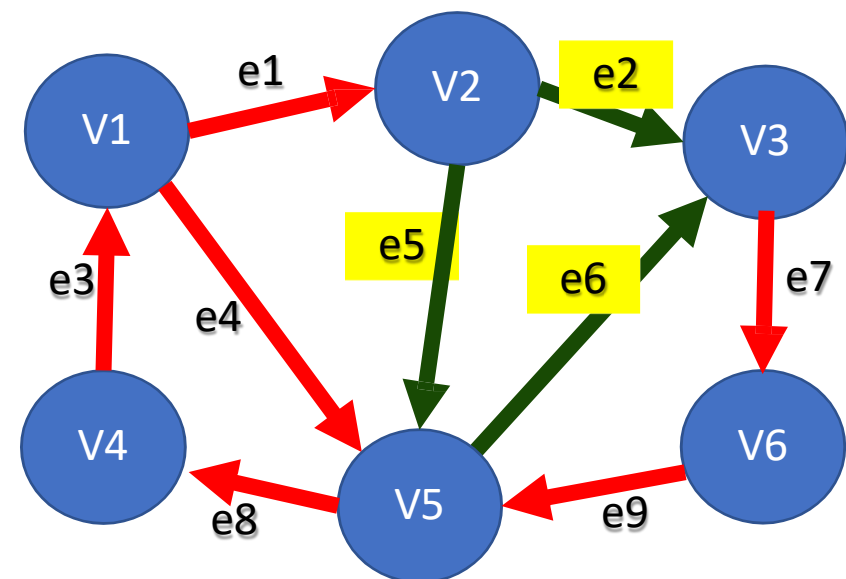
$C1 = (e1, e5, e4)$ is an elementary cycle

Elementary cycle

$G1 = (\{V1, V2, V3, V4, V5, V6\}, \{e1, e2, e3, e4, e5, e6, e7, e8, e9\})$



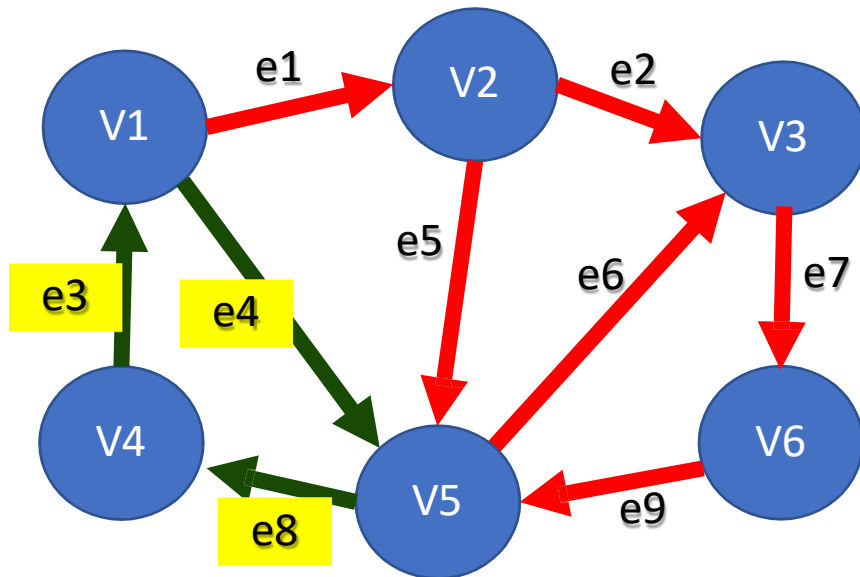
$C1 = (e1, e5, e4)$ is **elementary**



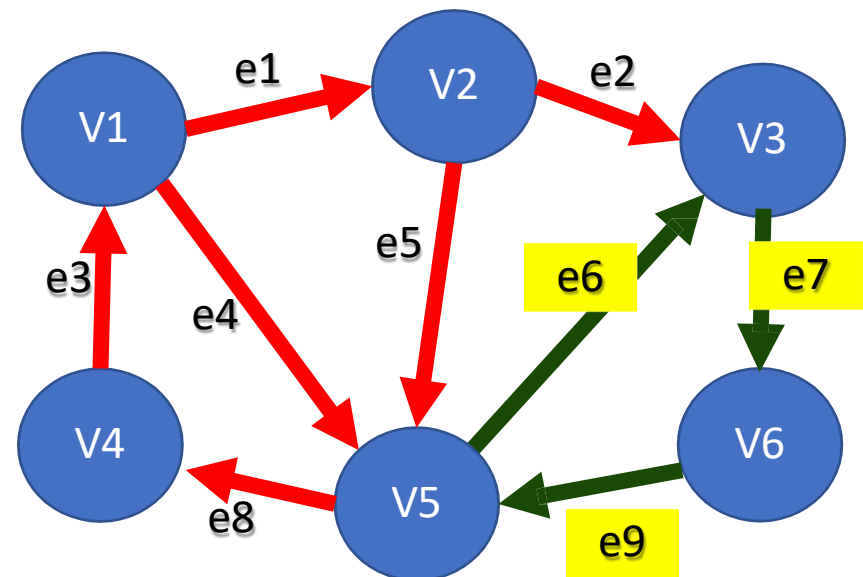
$C2 = (e2, e6, e5)$ is an **elementary** cycle

Elementary cycle

$G1 = (\{V1, V2, V3, V4, V5, V6\}, \{e1, e2, e3, e4, e5, e6, e7, e8, e9\})$



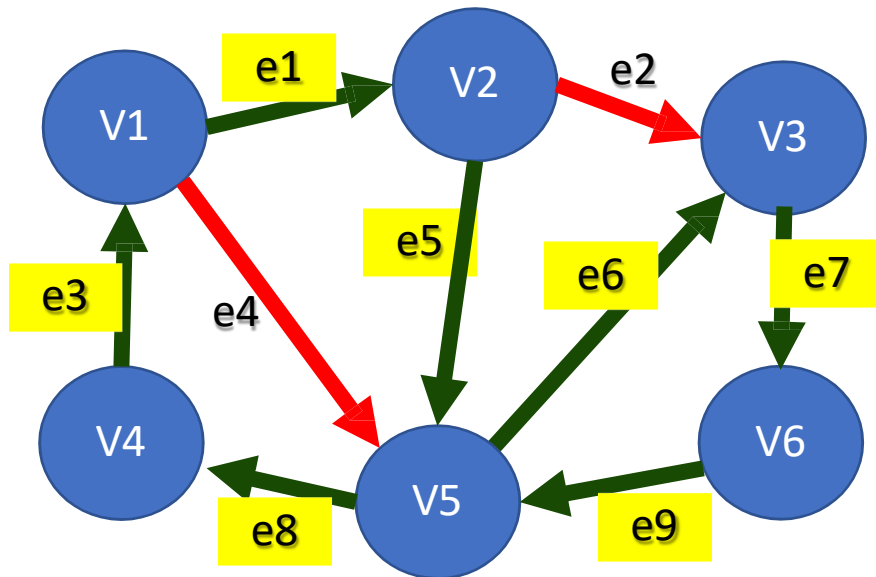
$C3 = (e3, e4, e8)$ is **elementary**



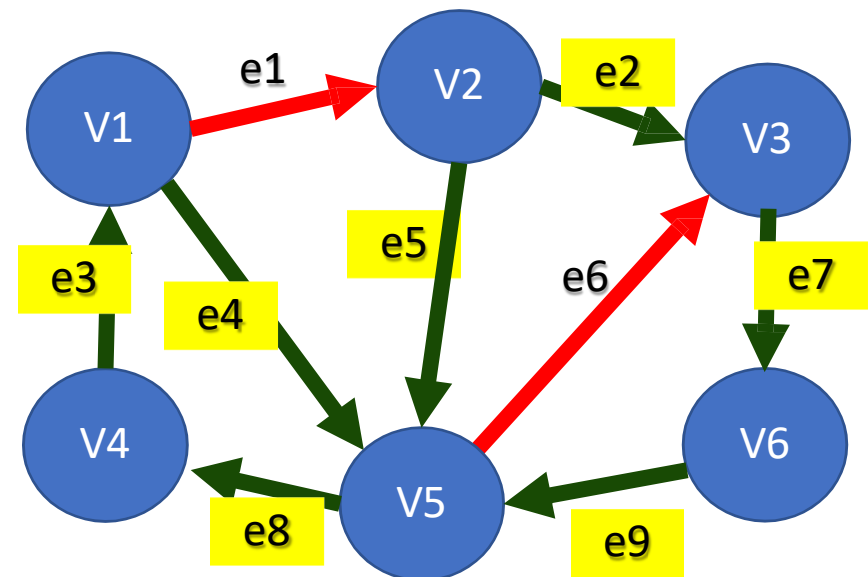
$C4 = (e7, e9, e6)$ is **elementary**

Elementary cycle

$G1 = (\{V1, V2, V3, V4, V5, V6\}, \{e1, e2, e3, e4, e5, e6, e7, e8, e9\})$



$C7 = (e3, e1, e5, e9, e7, e6, e8)$
is not an elementary cycle

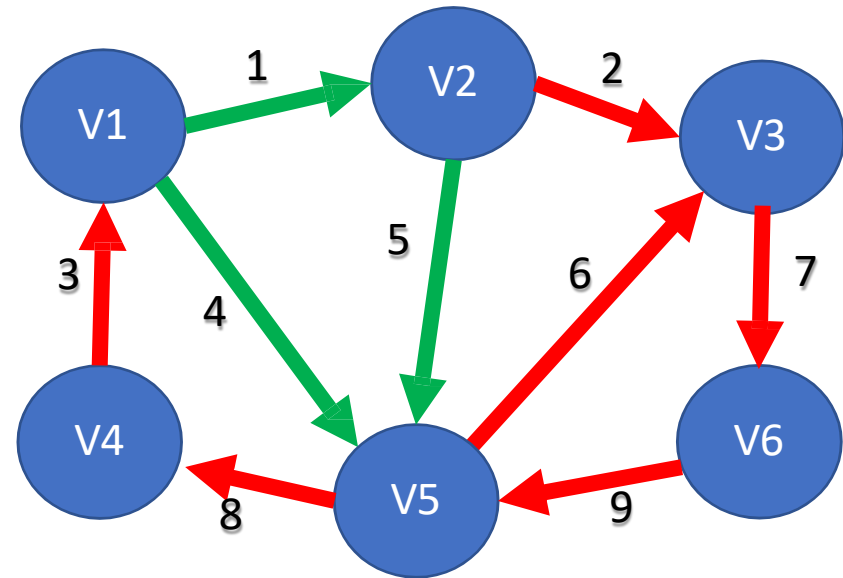


$C8 = (e3, e4, e9, e7, e2, e5, e8)$
is not an elementary cycle

Representative vector of a cycle (Vector notation)

- Vector notation

- The edges are numbered from **1** to **m**
- A cycle is represented by an **m-tuple** vector composed of **-1**, **1** and **0** in the following way :
- We consider all the edges in their order
- ✓ If the edge does not belong to the cycle, we put a "**0**"
- ✓ If the edge belongs to the cycle and is traveled in the right direction (we then call it "**direct**"), we put a **+1**
- ✓ If the edge belongs to the cycle but traveled in the wrong direction (we then call it "**reverse**"), we put a **-1**



The vector notation of the cycle $C1=(1, 5, 4)$ is $(+1, 0, 0, -1, +1, 0, 0, 0, 0)$

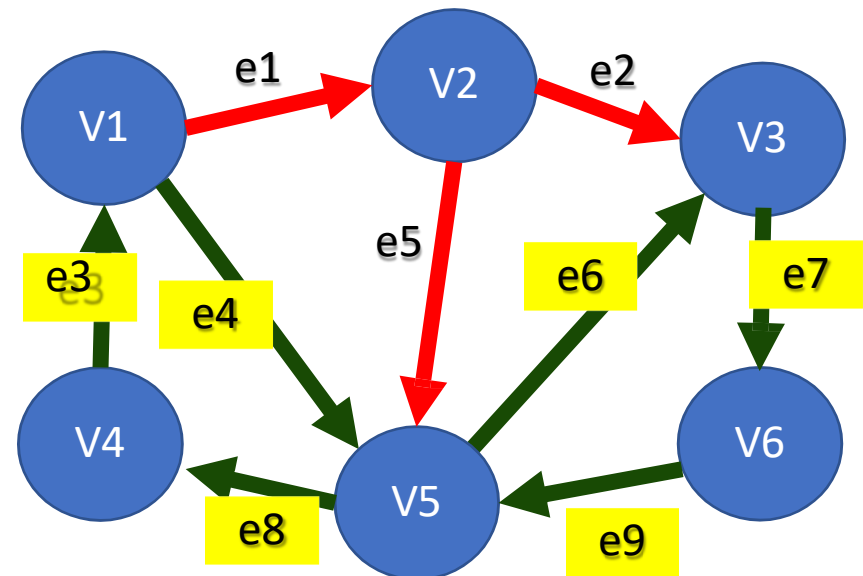
Cyclomatic Number

$G1 = (\{V1, V2, V3, V4, V5, V6\}, \{e1, e2, e3, e4, e5, e6, e7, e8, e9\})$

The **Cyclomatic Number** is denoted μ

Let G be a graph of order n (n vertices) with m edges and p connected components

$$\mu(G) = m - n + p.$$



$$\mu(G) = 9 - 6 + 1 = 4$$

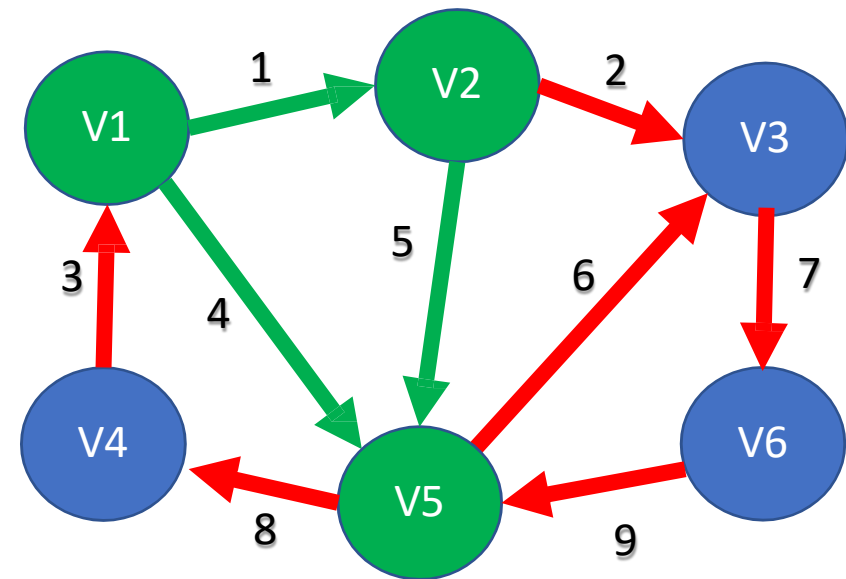
CoCycle

$G1 = (\{V1, V2, V3, V4, V5, V6\}, \{e1, e2, e3, e4, e5, e6, e7, e8, e9\})$

CoCycle

Let A be a set of vertices. The CoCycle associated to A denoted $\omega(A)$ is the set of edges leaving A denoted Ω^+ and the set of edges entering A noted Ω^-

$$\omega(A) = \Omega^+(A) \cup \Omega^-(A)$$



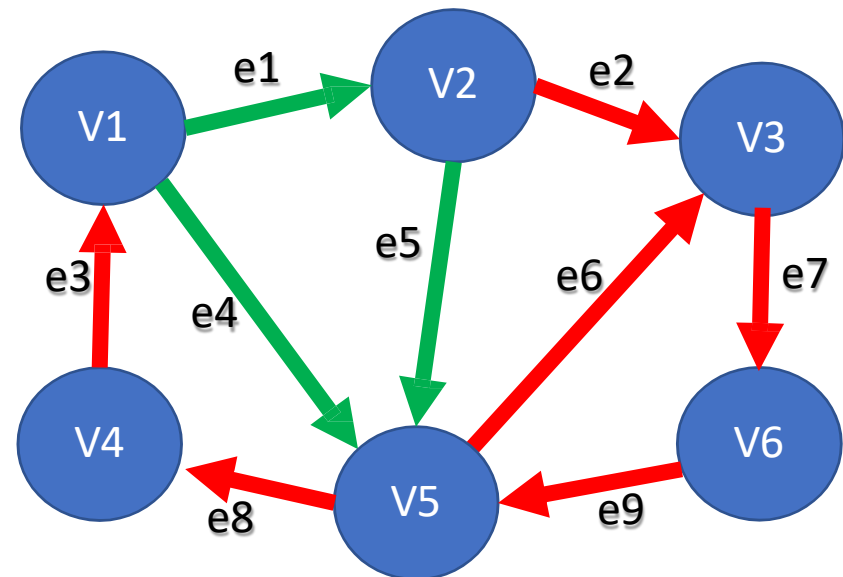
Let A be the set of vertices $A = \{V1, V2, V5\}$
 The CoCycle associated to A is $\omega(A) = (\Omega^+, \Omega^-)$
 $(+2, +6, +8, -3, -9)$
 Its vector notation is $(0, +1, -1, 0, 0, +1, 0, +1, -1)$

Elementary CoCycle

- Elementary CoCycle

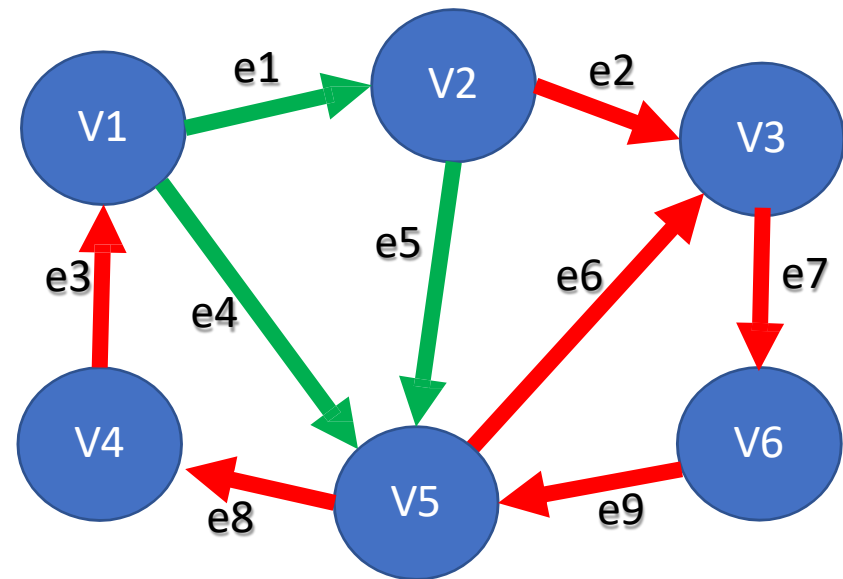
A CoCycle associated with a set of vertices Δ is **elementary** if the deletion of Δ generates a connected component

A CoCycle is said to be **elementary** if it is minimal.



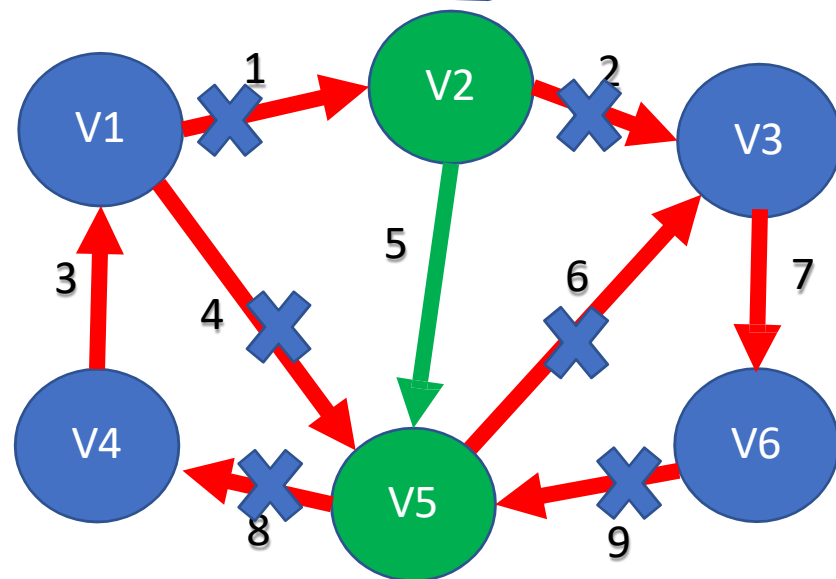
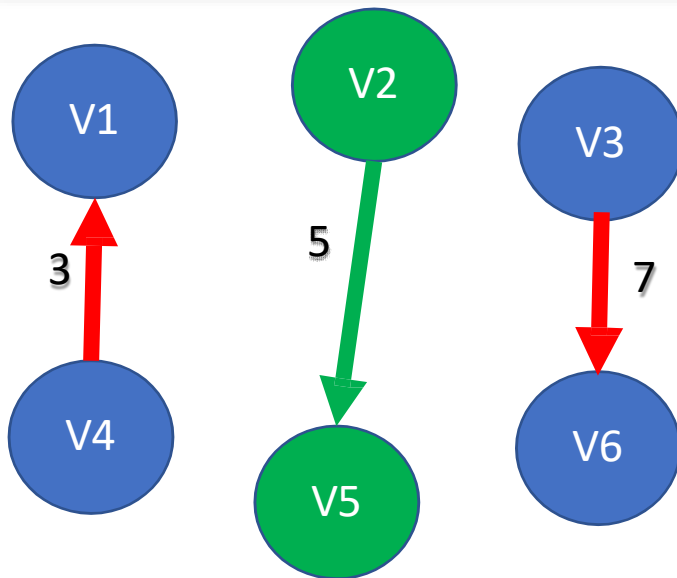
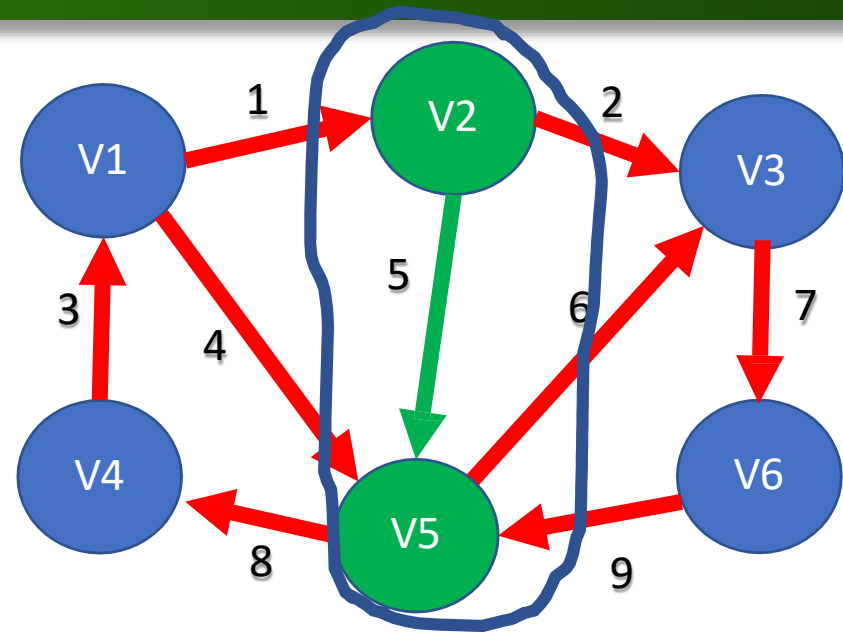
Elementary CoCycle

A cocycle is **elementary** when it is composed of edges connecting two subsets of connected vertices that partition a connected component of the graph



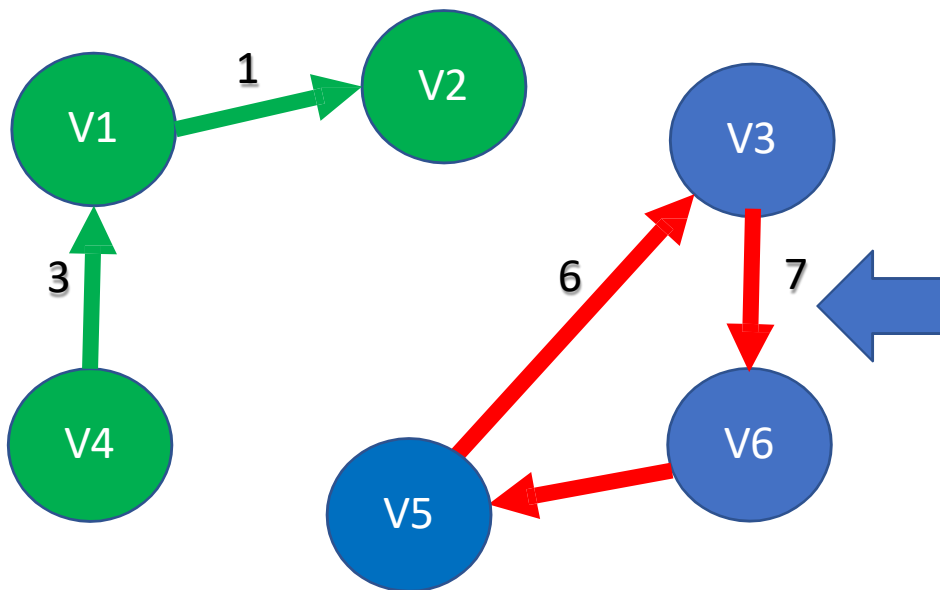
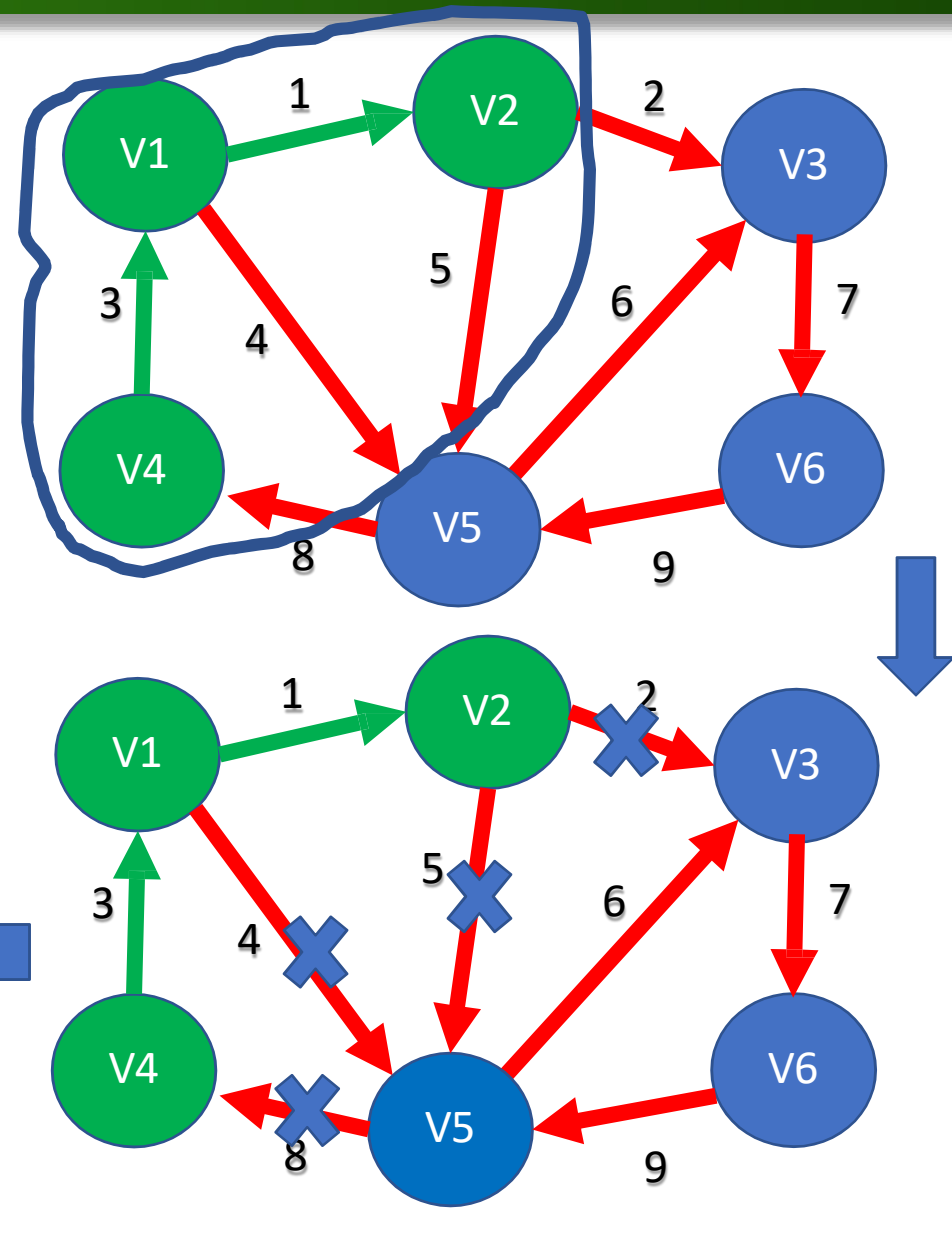
Elementary CoCycle (example 1)

The cocycle associated to $(V2, V5)$ is **not elementary** because it is composed of edges connecting $\{V2, V5\}$ to $\{V1, V4, V3, V6\}$ and if $\{V2, V5\}$ is deleted, this is not the case for $\{V1, V4, V3, V6\}$.



Elementary CoCycle (example 2)

The cocycle associated to $\{V1, V2, V4\}$ of which its notation vector is $(0,1,0,1,1,0,0,-1,0)$ is elementary.



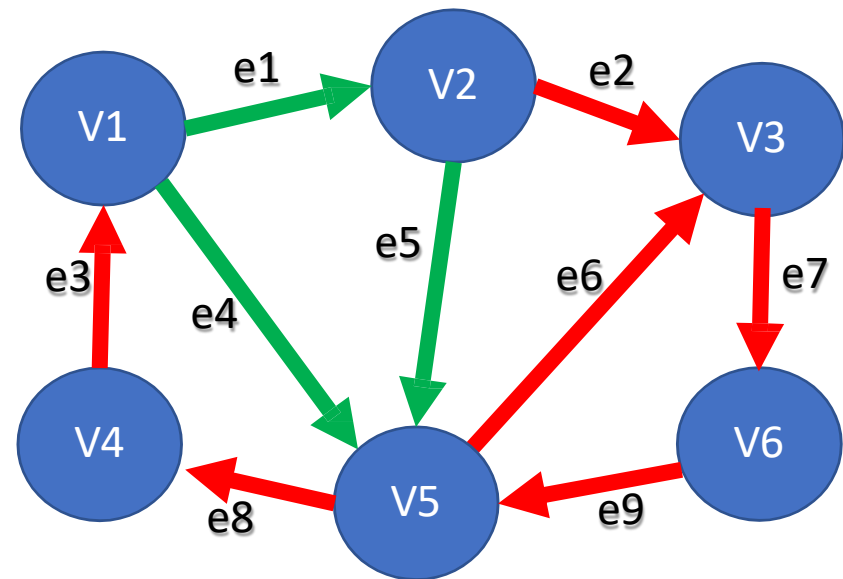
CoCyclomatic Number

- CoCyclomatic Number

Let G be a graph of order n (n vertices) with p connected components.

The cocyclomatic number is denoted λ :

$$\lambda(G) = n - p.$$



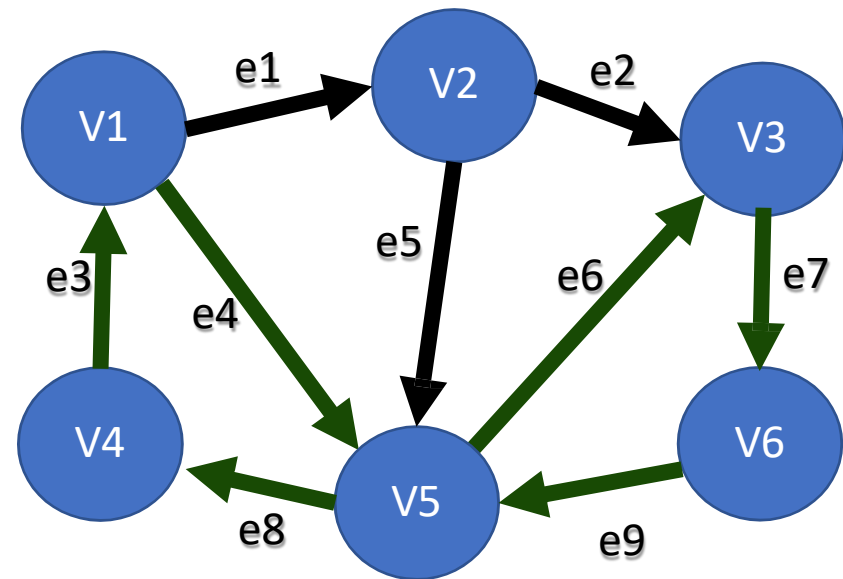
$$\lambda(G) = 6 - 1 = 5$$

CoCircuit

- CoCircuit

In a cocycle associated with a set of vertices A ($\Omega^+(A), \Omega^-(A)$), if one of the sets $\Omega^+(A)$ or $\Omega^-(A)$ is empty, the cocycle is called **cocircuit**

CoCircuit



Let A be the set of vertices $A = (V1, V2, V3, V4, V5, V6)$

The CoCycle associated to A is $\omega(A) = (\Omega^+, \Omega^-) = ()$

Its vector notation is $(0, 0, 0, 0, 0, 0)$

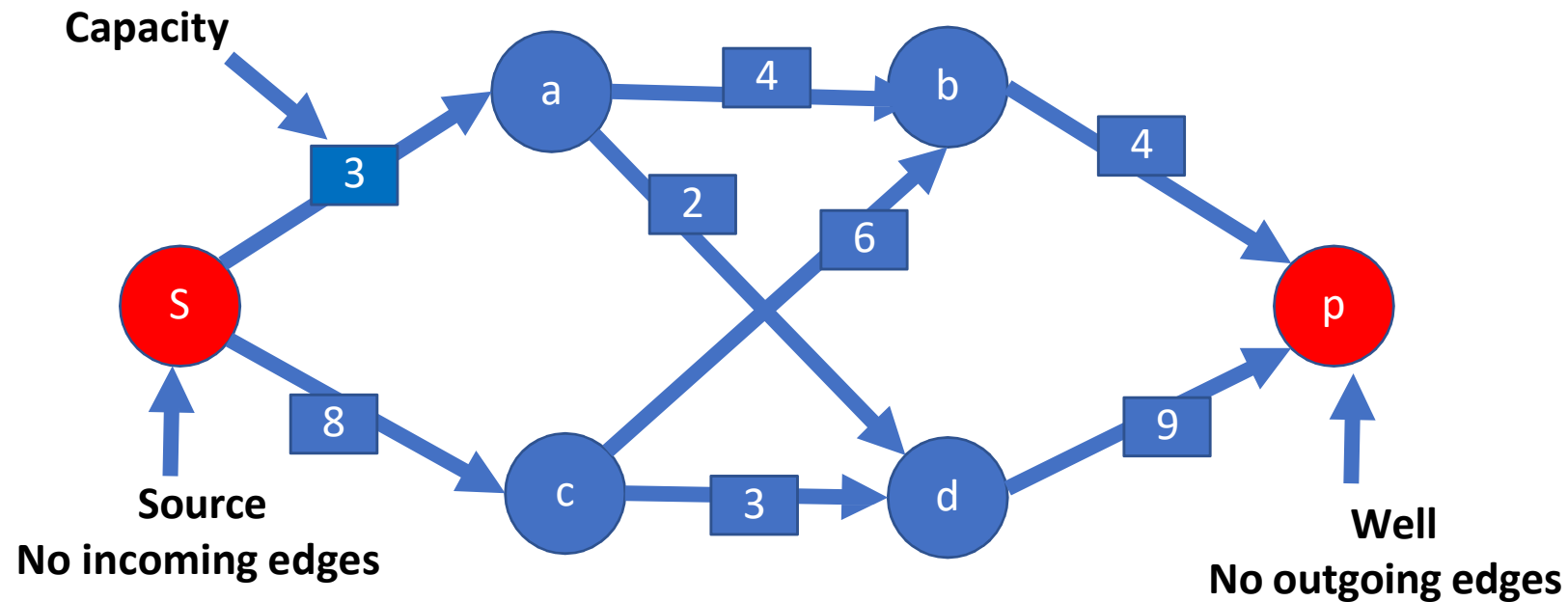


Section 2:Flows

Flows and tensions

Basic Concepts

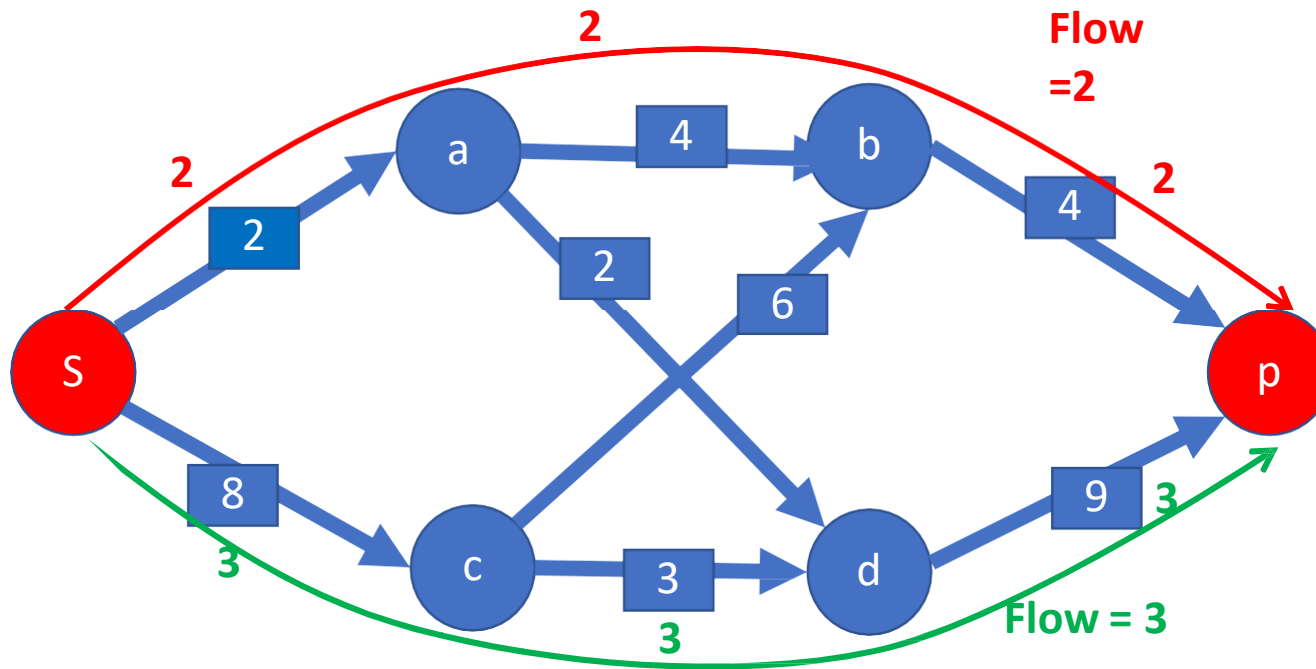
Flow Networks (Networks of transportation)



Flows and tensions

Basic Concepts

Flow Networks (Networks of transportation)



Flow of the nextwork = \sum flows

The flow (flux)= is positive and less than or equal to the edge capacity

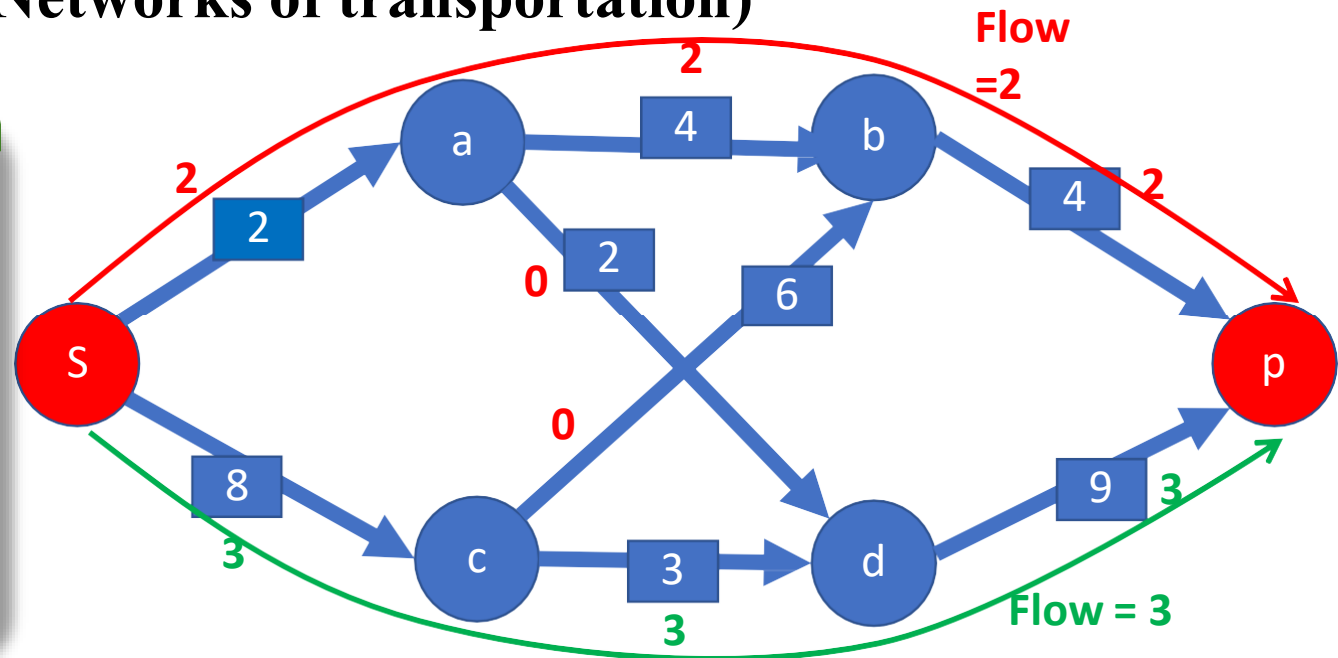
Flows and tensions

Basic Concepts

Flow Networks (Networks of transportation)

The edge (s a) is saturated

The edge (c d) is saturated

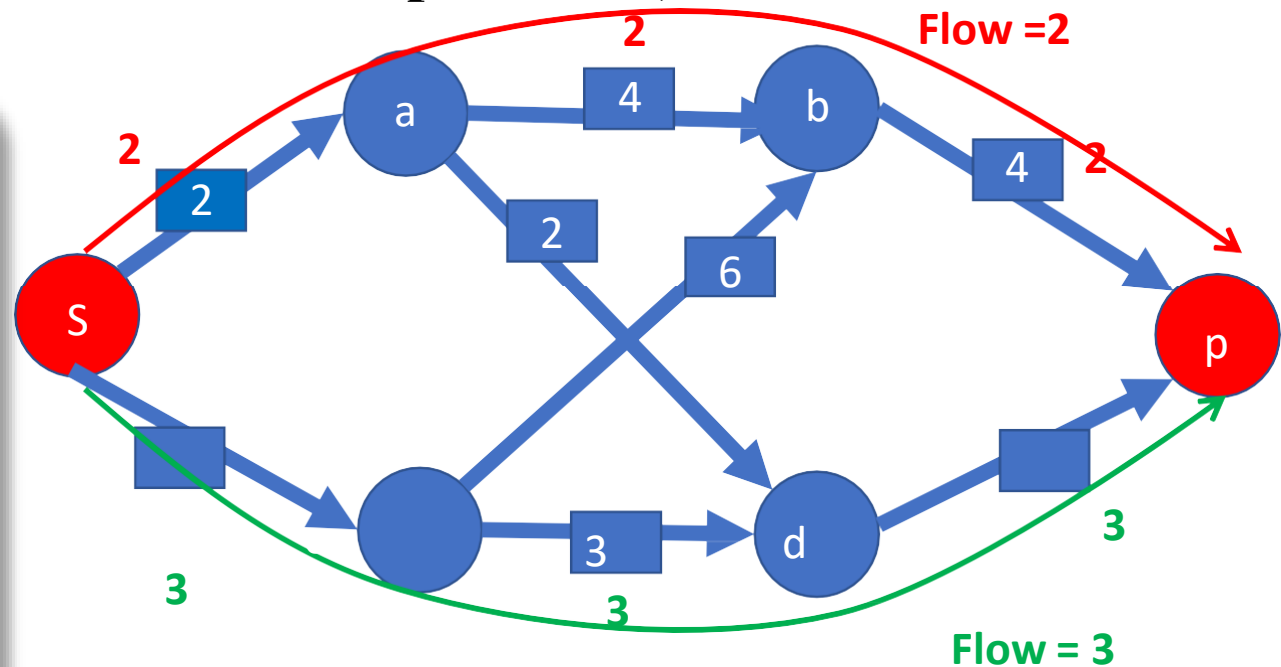



- **Saturated direct edge:** if the flow circulating there = the capacity of the edge
- **Saturated reverse edge :** if the flow circulating there = 0
- **Saturated Path/Chain :** if it contains at least one saturated edge

Basic Concepts

Flow Networks (Networks of transportation)

Law of conservation of the flow (fluxes):
The sum of the flows arriving at a vertex is equal to the sum of the flows leaving it





Section 3:

Problematic of Maximum Flow

Ford-Fulkerson Algorithm

Problematic of Maximum Flow

The problem we need to solve is that of finding a channeled flow ϕ_m of maximum value $\phi_m(u_0)$, in a network with capacities:

It is the issue of the Maximum Flow

Problematic of Maximum Flow

- This problem finds various practical applications in relation to network problems.
- The applications are multiple: computer problems, roads, railways, etc. It also applies to all other transfer problems such as imports/exports, migratory and demographic flows, but also to more abstract flows such as financial transfers.

Ford-Fulkerson Algorithm

- The Ford-Fulkerson algorithm is an algorithm for the maximum flow problem, a classic optimization problem in operations research.
- This optimization problem can be represented by a graph with an input (left) and an output (right). The flow represents the flow from the input to the output

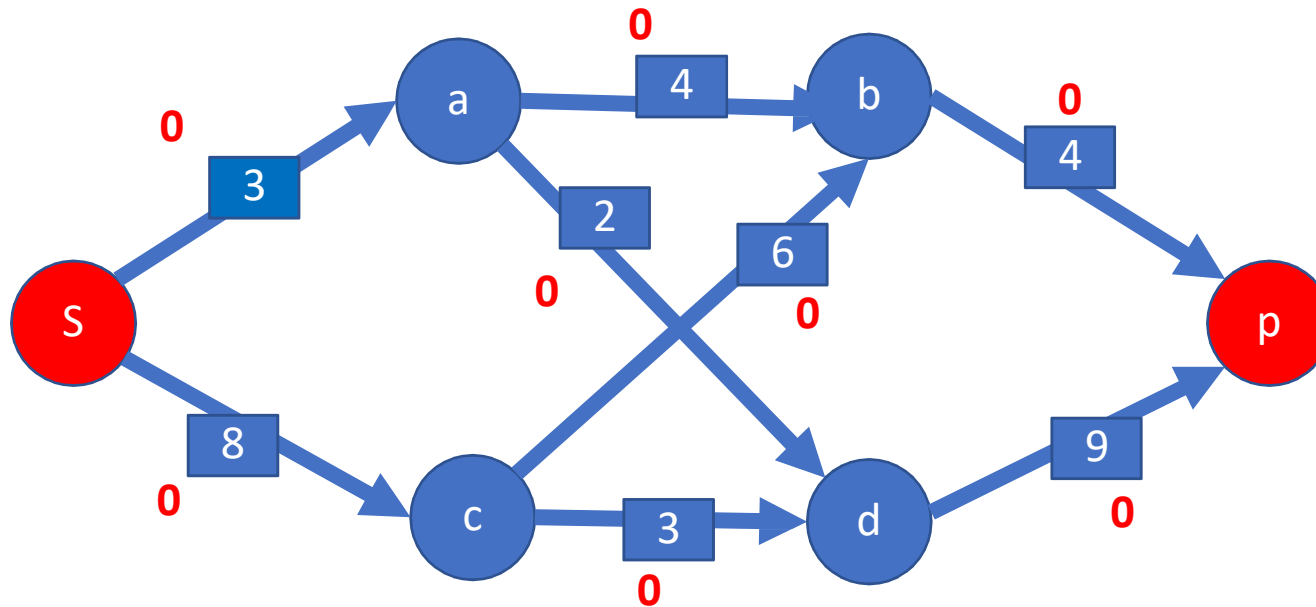
Ford-Fulkerson Algorithm

- In pseudo-computer code, the algorithm can be presented as follows:
- Initialization (Start from an initial flow compatible with the capacities);
 - End := False
 - While (End = False)
 - Perform the marking procedure from the current flow
 - If (P is unmarked) Then
 - set end:= True {the flow is maximal}
 - Else
 - Modify the flow from an improving chain
 - Endwhile
- End

From an initial flow compatible with the capacities, the algo improves the flow as long as the marking procedure applied to the current flow allows to mark **P**.

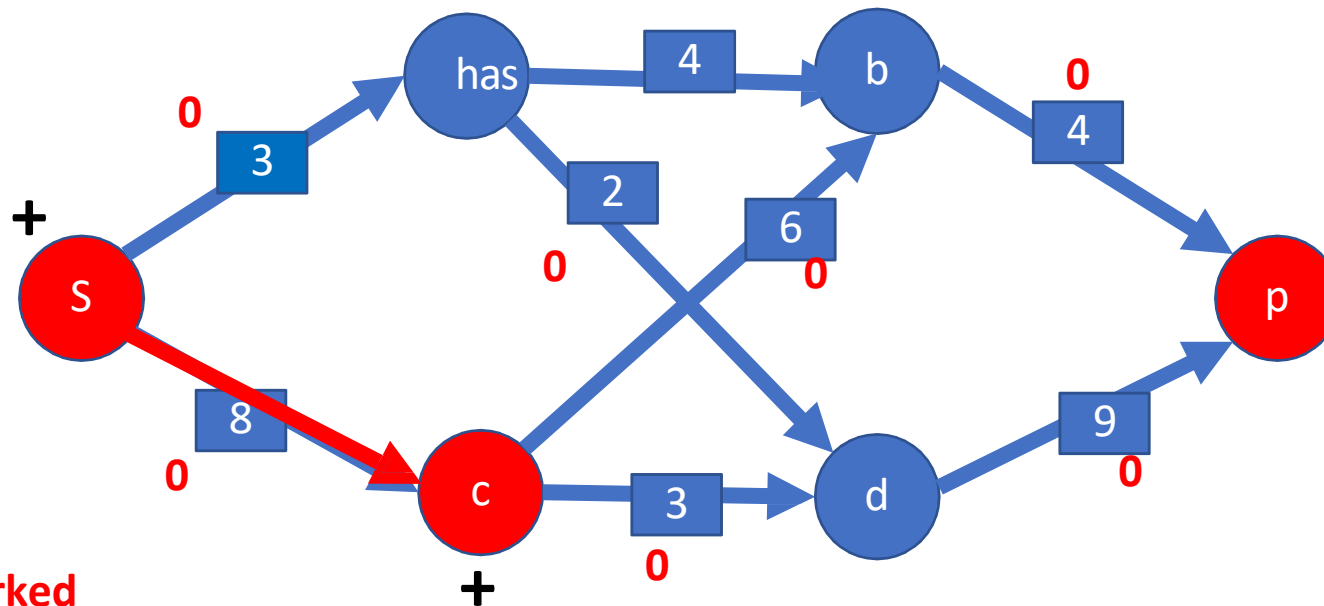
Ford-Fulkerson Algorithm

- Flow + (Maximum Flow) = 0



Ford-Fulkerson Algorithm

- Flow + (Maximum Flow) = 0
- The research of an improving chain
- **Mark S, mark the adjacent vertex if (it is not marked and the flux is strictly less than the capacity), choose the largest flux 0**



S is marked

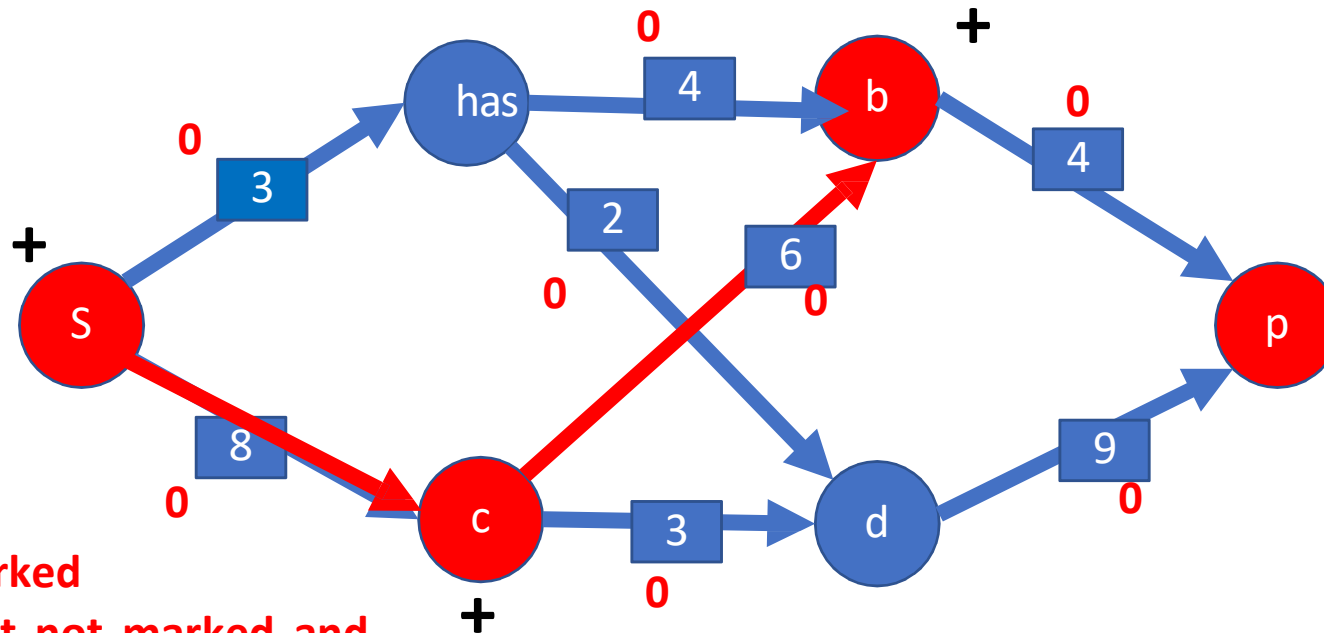
a is not marked and $0 < 3$

c is not marked and $0 < 8$

**We choose to mark C
because $8 - 0 > 3 - 0$**

Ford-Fulkerson Algorithm

- Flow + (Maximum Flow) = 0
- The research of an improving chain
- Mark S, mark the adjacent vertex if (it is not marked and the flux is strictly less than the capacity), choose the largest flux



C is marked

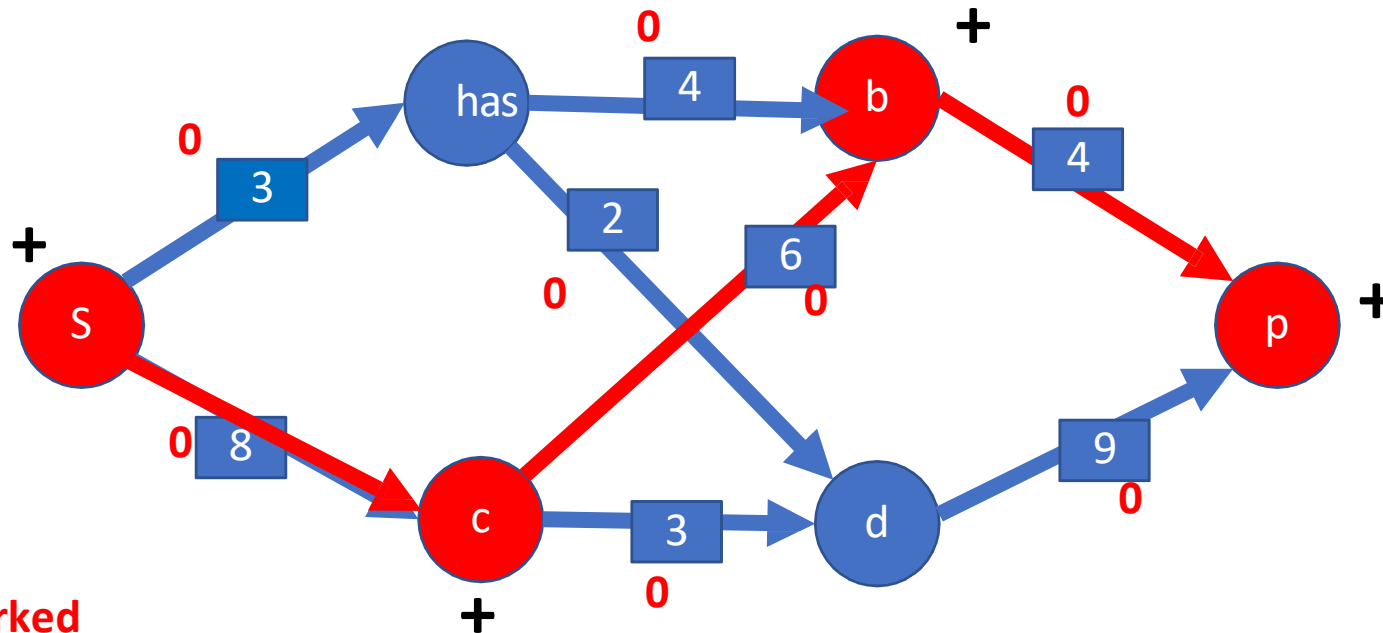
b is not not marked and
 $0 < 6$

d is not marked and $0 < 3$

We choose to mark b
because $6 - 0 > 3 - 0$

Ford-Fulkerson Algorithm

- Flow + (Maximum Flow) = 0
- The research of an improving chain
- Mark S, mark the adjacent vertex if (it is not marked and the flux is strictly less than the capacity), choose the largest flux



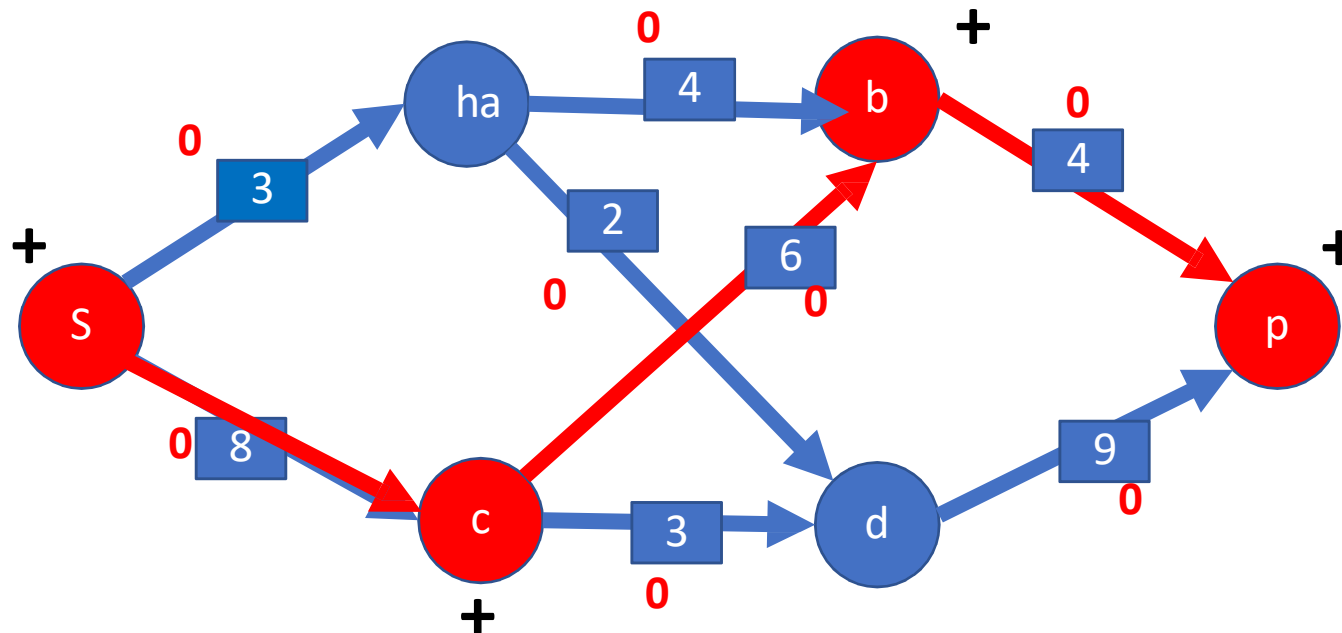
b is marked

P is not marked and $0 < 4$

**We choose to mark P
because $0 < 4$**

Ford-Fulkerson Algorithm

- Flow + (Maximum Flow) = 0
- The research of an improving chain
- Mark S, mark the adjacent vertex if (it is not marked and the flux is strictly less than the capacity), choose the largest flux

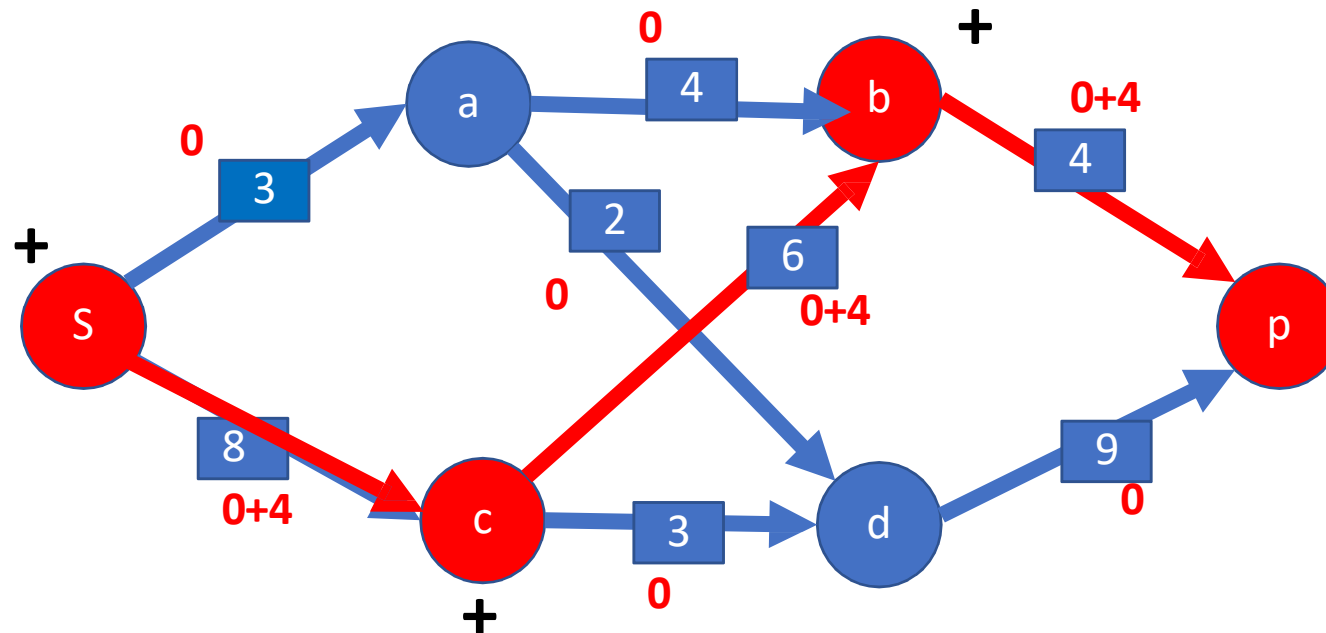


S c b P is an improving chain

**The flow can be increase
by $(\min(8-0, 6-0, 4-0) = +4)$**

Ford-Fulkerson Algorithm

- Flow + (Maximum Flow) = 0+4
- The research of an improving chain
- Mark S, mark the adjacent vertex if (it is not marked and the flux is strictly less than the capacity), choose the largest flux

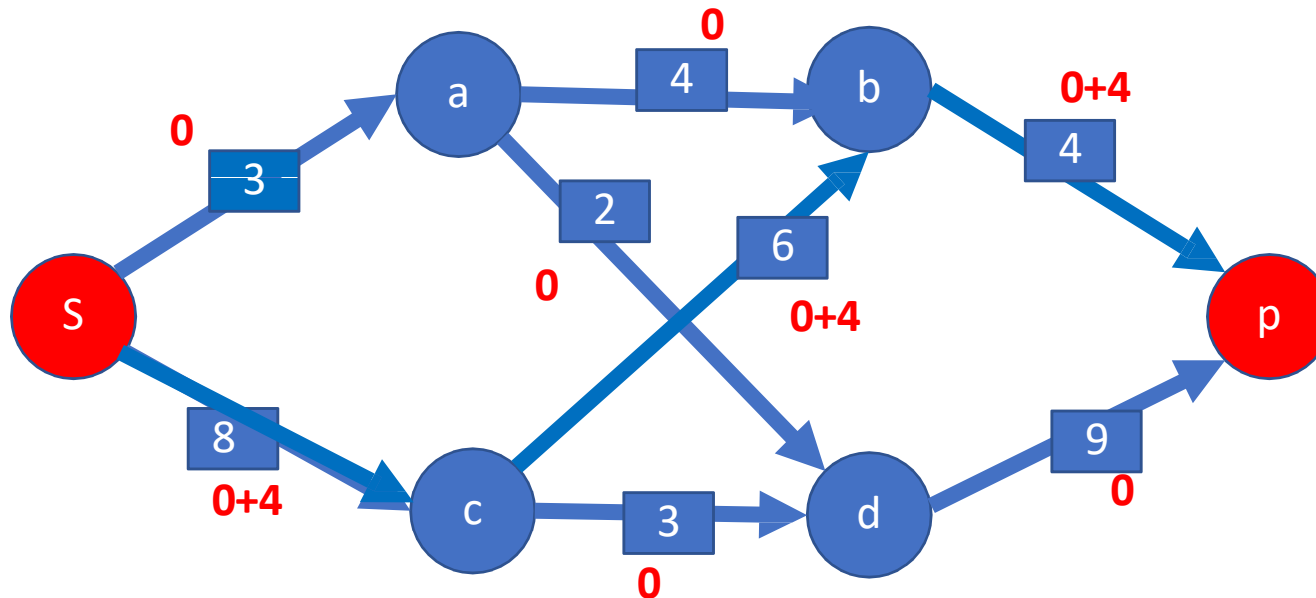


S c b p is an improving chain

**The flow can be increased by
($\min(8-0, 6-0, 4-0) = +4$)**

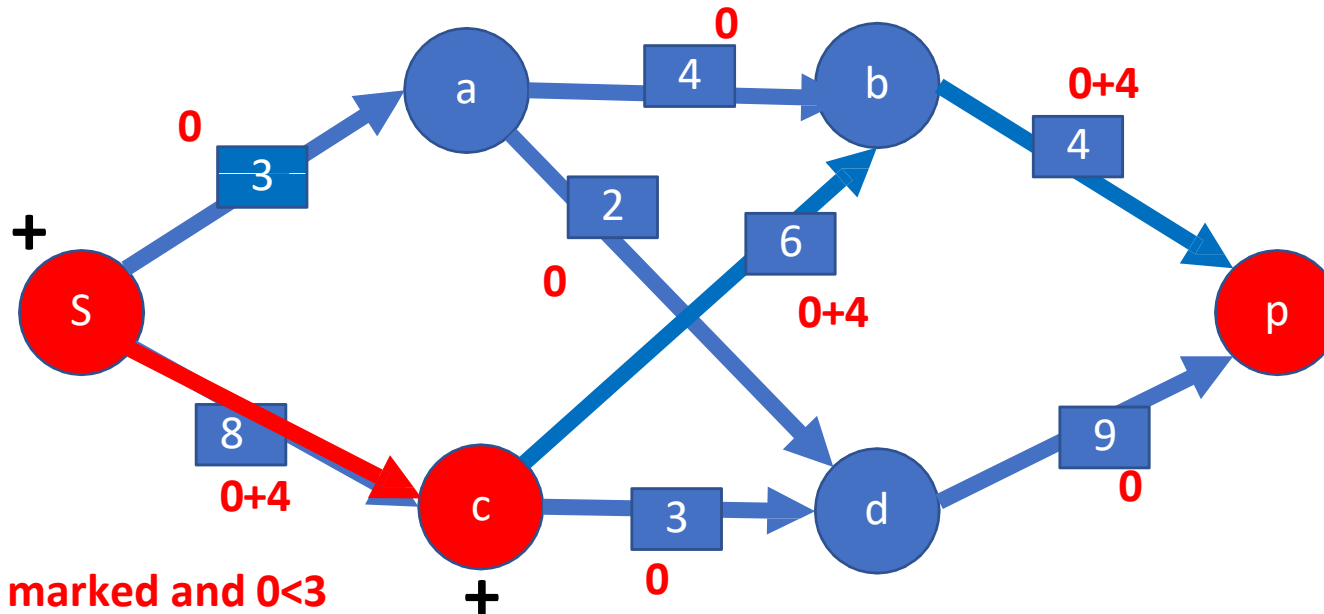
Ford-Fulkerson Algorithm

- Flow + (Maximum Flow)=0+4
- Erasing the marking
- The research of an other improving chain
- Mark S, mark the adjacent vertex if (it is not marked and the flux is strictly less than the capacity), choose the largest flux



Ford-Fulkerson Algorithm

- Flow + (Maximum Flow)=0+4
- Mark the source S
- The research of an improving chain
- Mark S, mark the adjacent vertex if (it is not marked and the flux is strictly less than the capacity), choose the largest flux

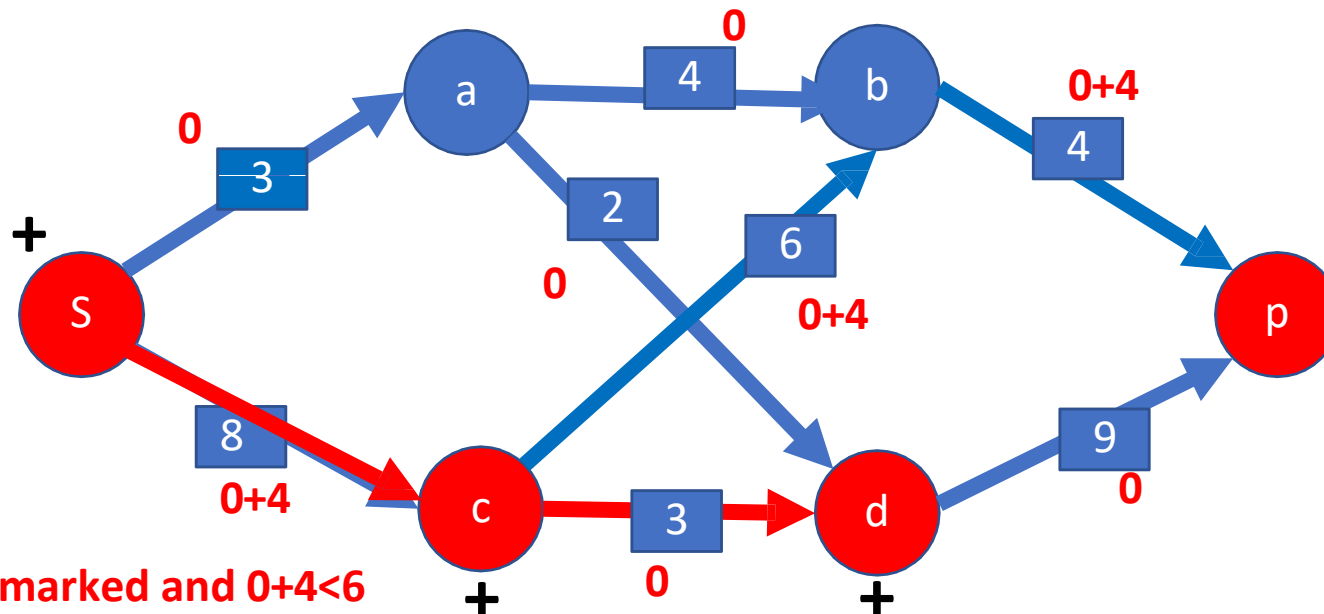


a is not marked and $0 < 3$
c is not marked and $0+4 < 8$

We choose to mark c because $8-4 > 3-0$

Ford-Fulkerson Algorithm

- $\text{Flow} + (\text{Maximum Flow}) = 0 + 4$
- C is marked
- The research of an improving chain
- Mark S , mark the adjacent vertex if (it is not marked and the flux is strictly less than the capacity), choose the largest flux



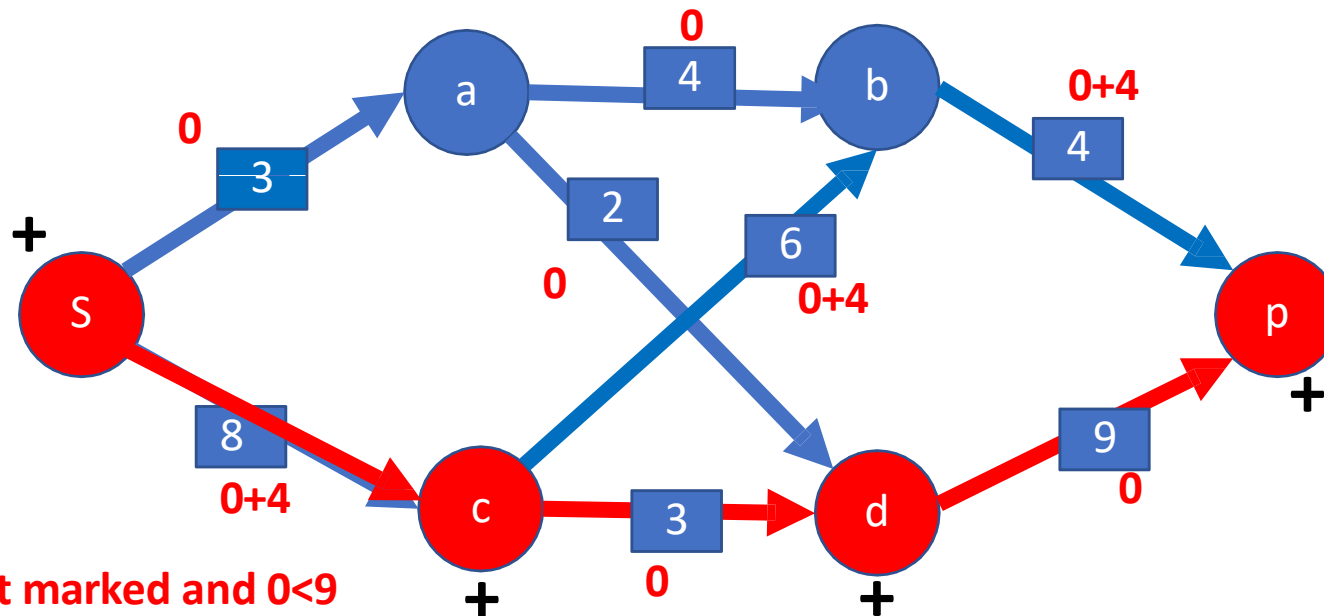
b is not marked and $0 + 4 < 6$

d is not not marked and $0 < 3$

We choose to mark d because $3 - 0 > 6 - 4$

Ford-Fulkerson Algorithm

- Flow+ (Maximum Flow) = 0 + 4
- d is marked
- The research of an improving chain
- Mark S, mark the adjacent vertex if (it is not marked and the flux is strictly less than the capacity), choose the largest flux

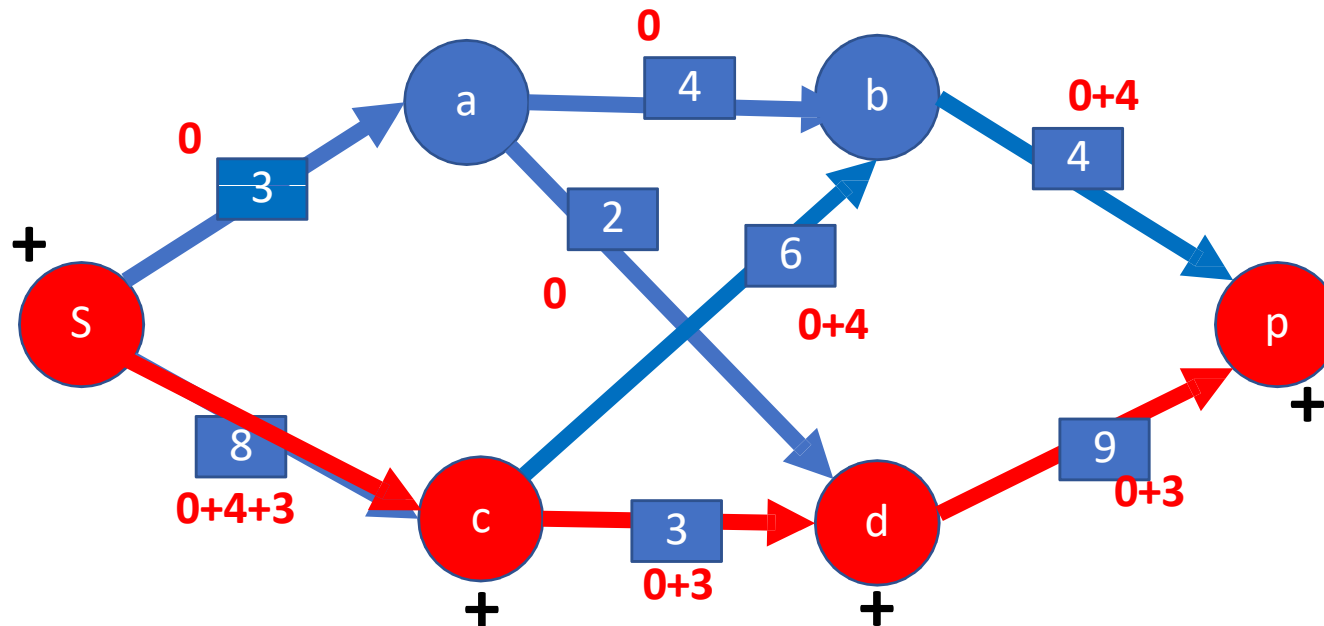


P is not marked and $0 < 9$

We mark p

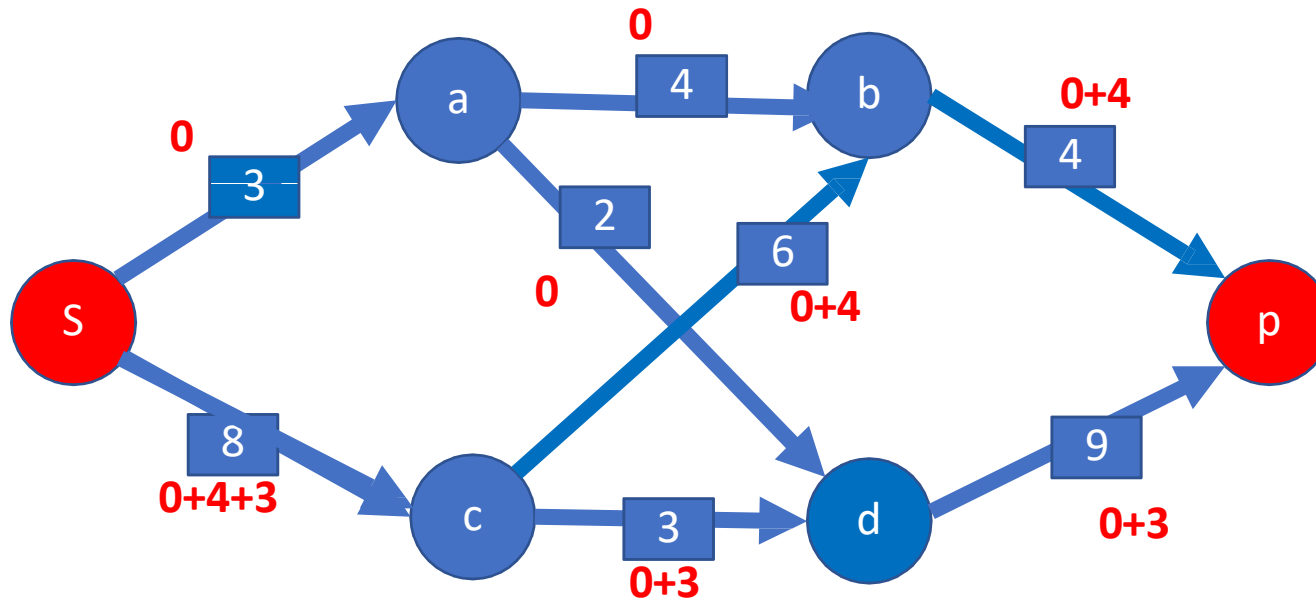
Ford-Fulkerson Algorithm

- Flow+ (Maximum Flow) = 0 + 4
- P is marked and the improving chain is S c d P
- We can increase the flow of $\min(8-4, 3-0, 9-0) = +3$
- Flow+ = 0 + 4 + 3



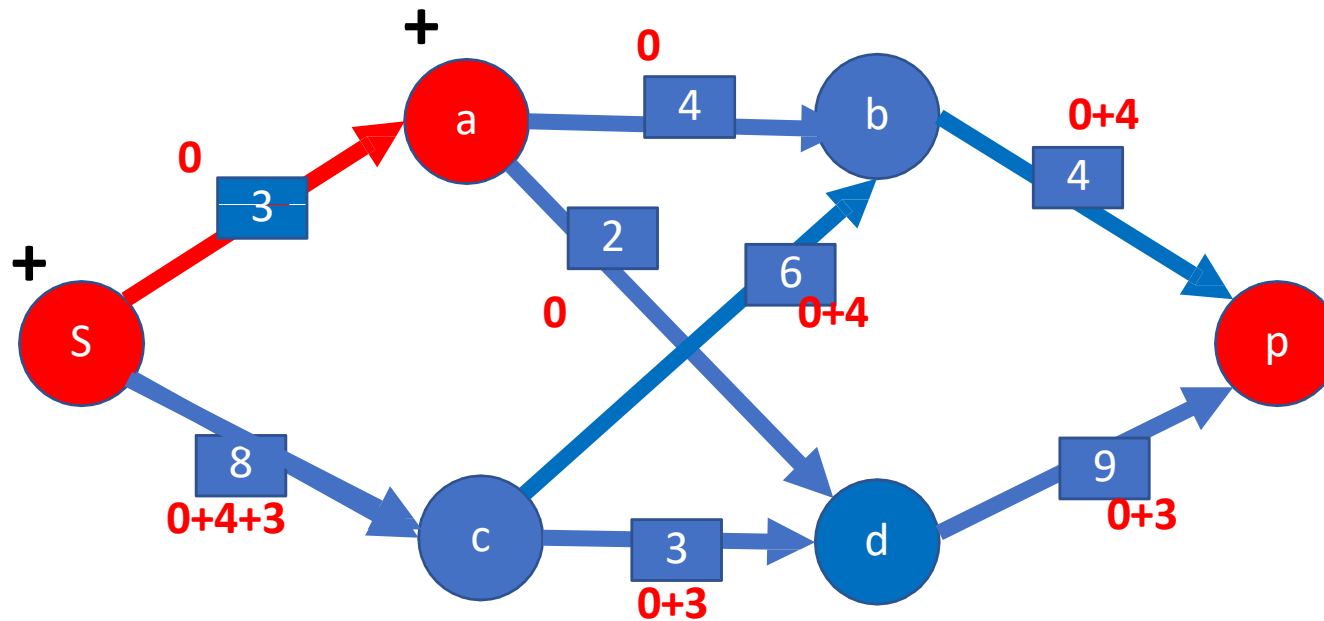
Ford-Fulkerson Algorithm

- Flow += 0+4+3
- Delete the marking and research for an other improving chain



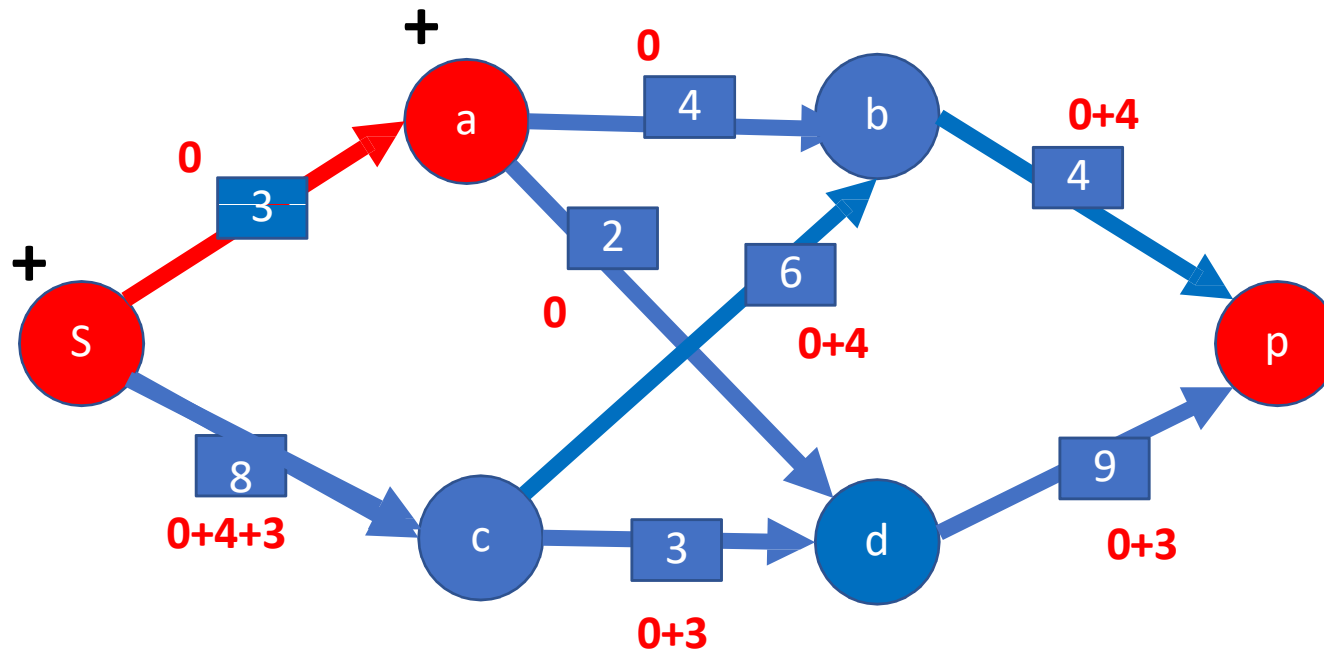
Ford-Fulkerson Algorithm

- Flow += 0+4+3
- Mark S
- We can mark either c or a but we choose a because $\max(3-0, 8-7)=3$



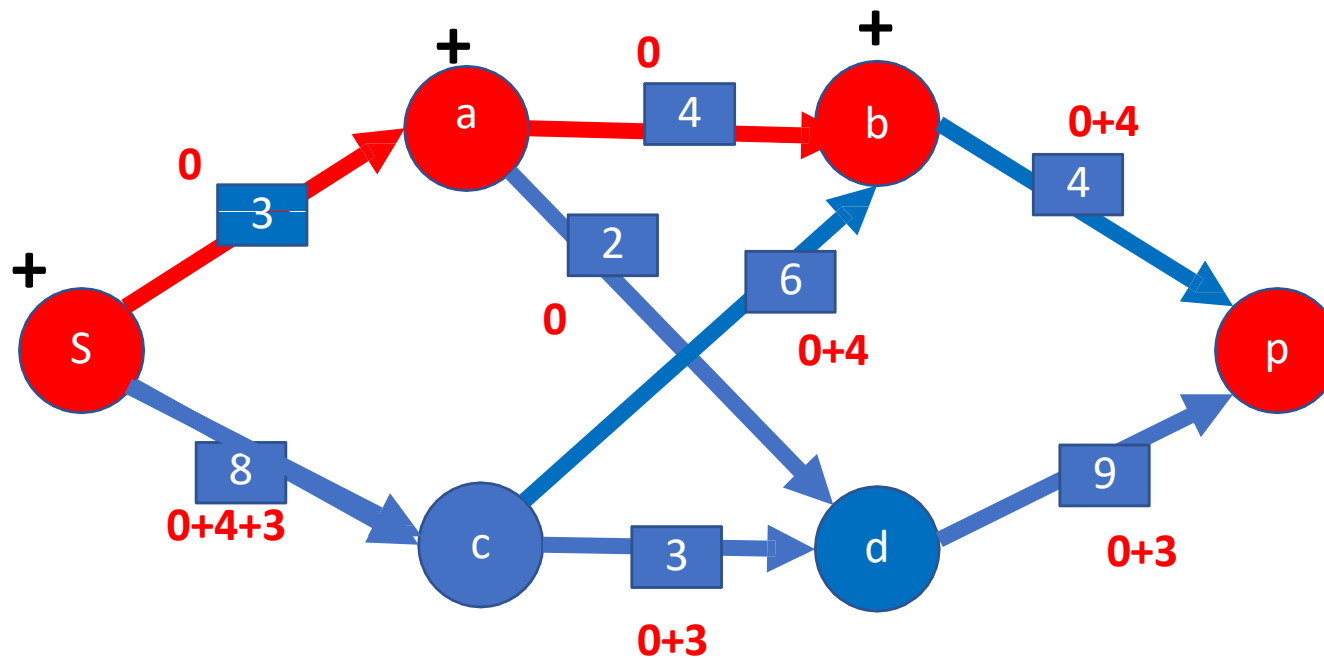
Ford-Fulkerson Algorithm

- Flow += 0+4+3
- S is marked
- a is marked
- We choose to mark b because $2-0 < 4-0$



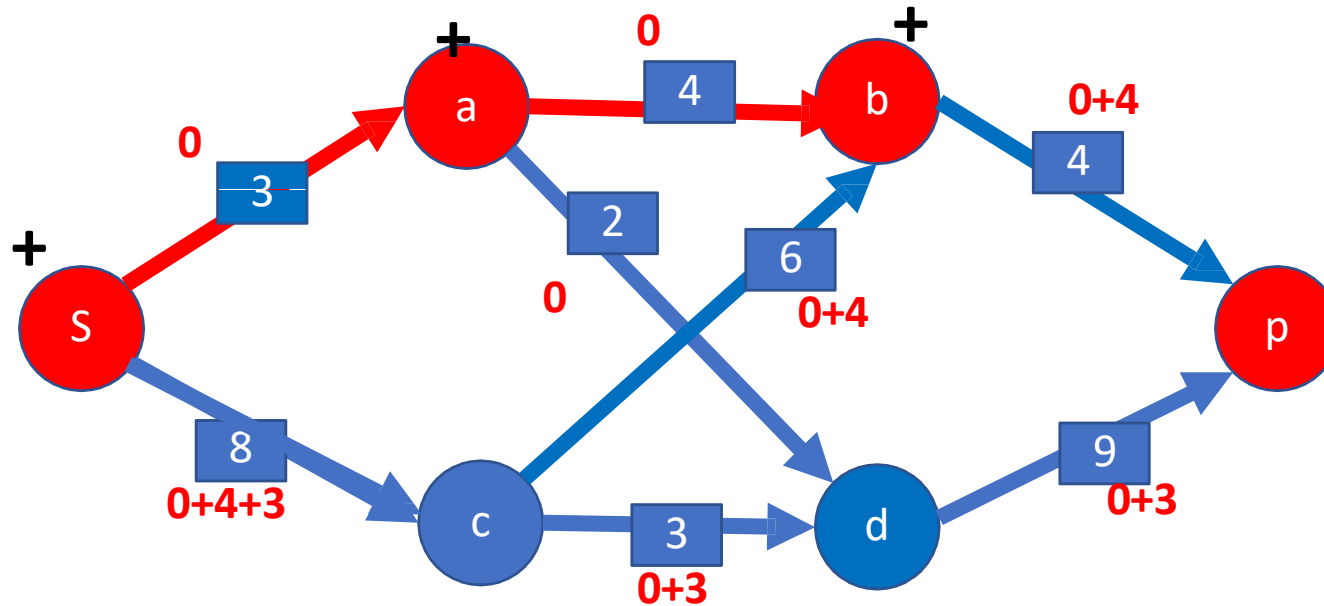
Ford-Fulkerson Algorithm

- Flow += 0+4+3
- S is marked
- a is marked
- b is marked



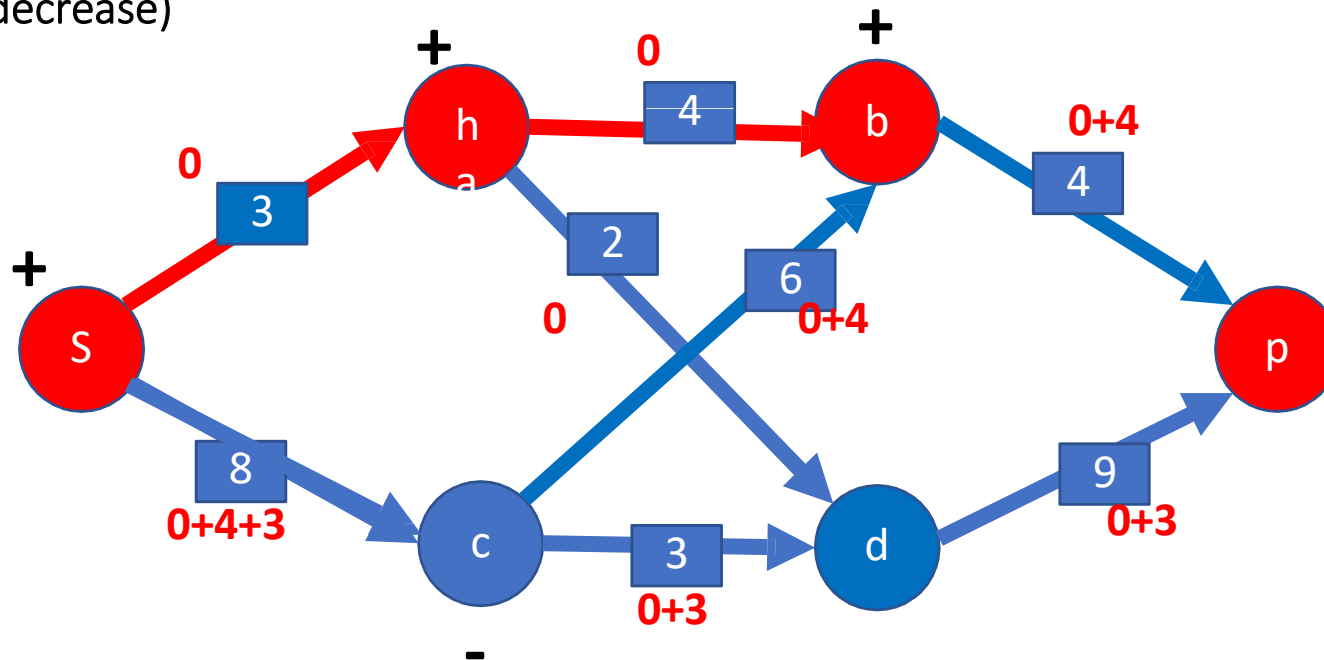
Ford-Fulkerson Algorithm

- Flow += 0+4+3
- S is marked
- a is marked
- b is marked
- We cannot mark P because the edge (b p) is saturated



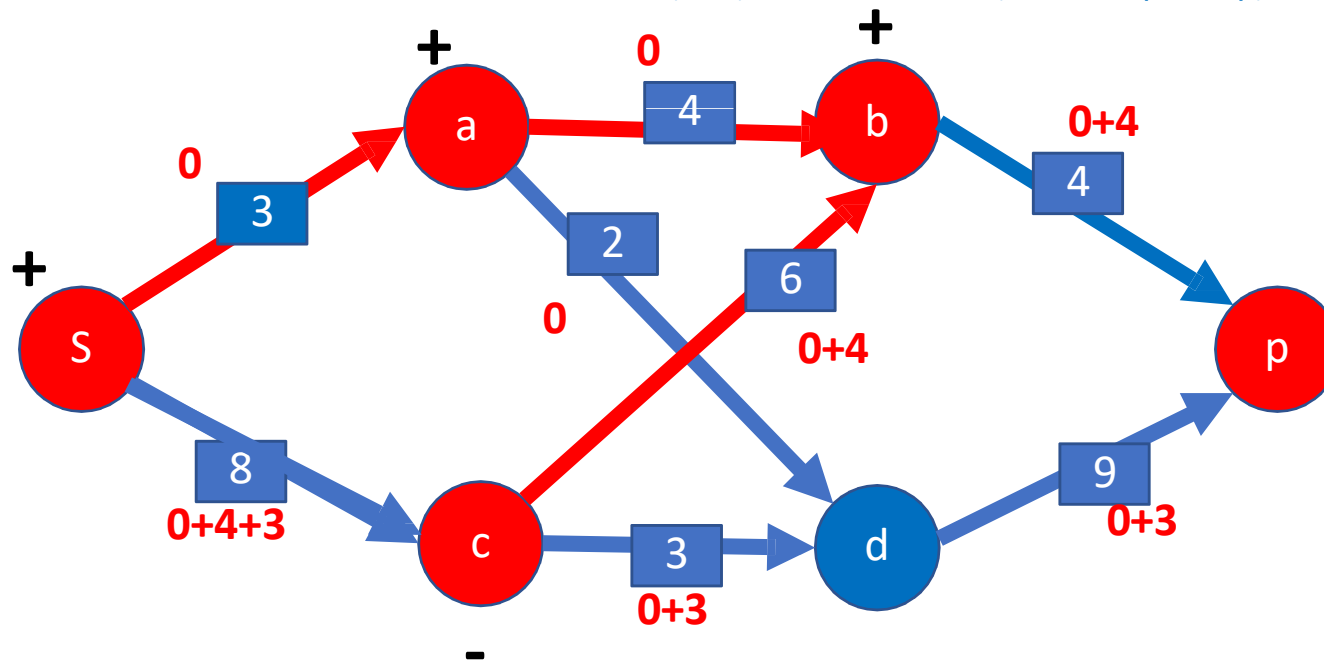
Ford-Fulkerson Algorithm

- Flow += 0+4+3
- S is marked
- a is marked
- b is marked
- We mark c by − because it is an **inverse edge** and the flux can decrease (c is the origin of an edge whose end is marked and the flux which is greater than 0 can decrease)



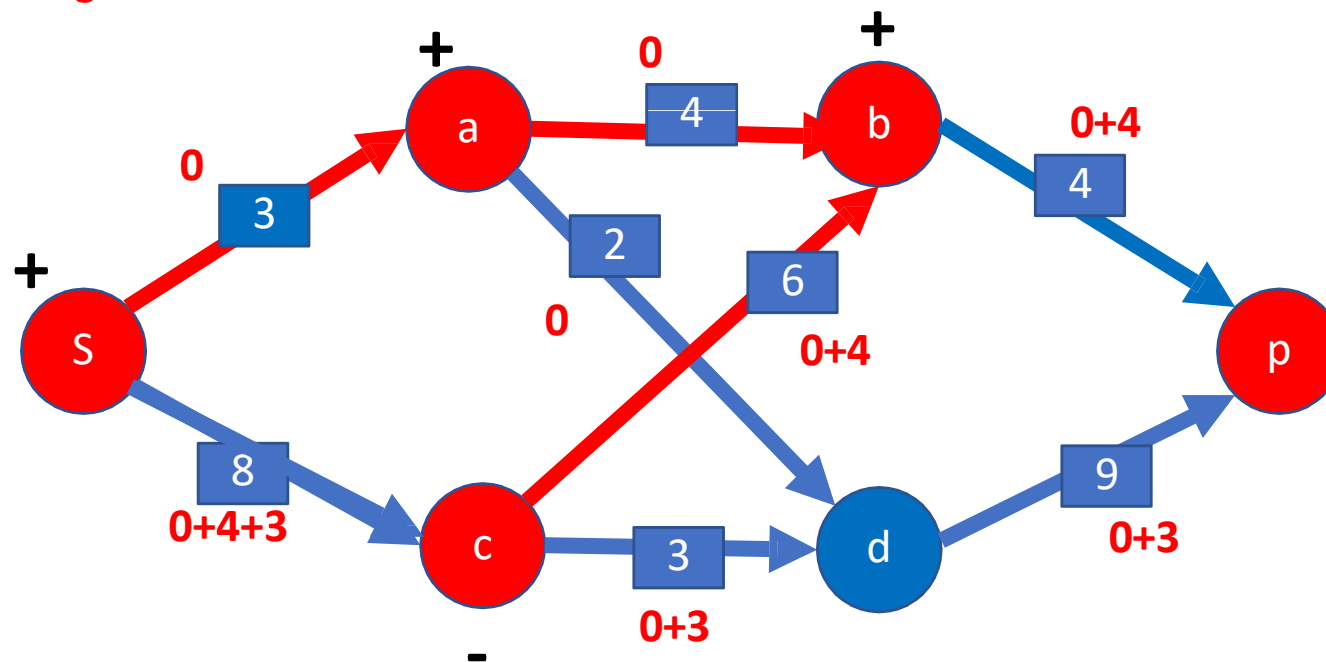
Ford-Fulkerson Algorithm

- Flow += 0+4+3
- S is marked
- a is marked
- b is marked
- c is marked by −
- We cannot mark d because the arc (c d) is saturated (flux=capacity)



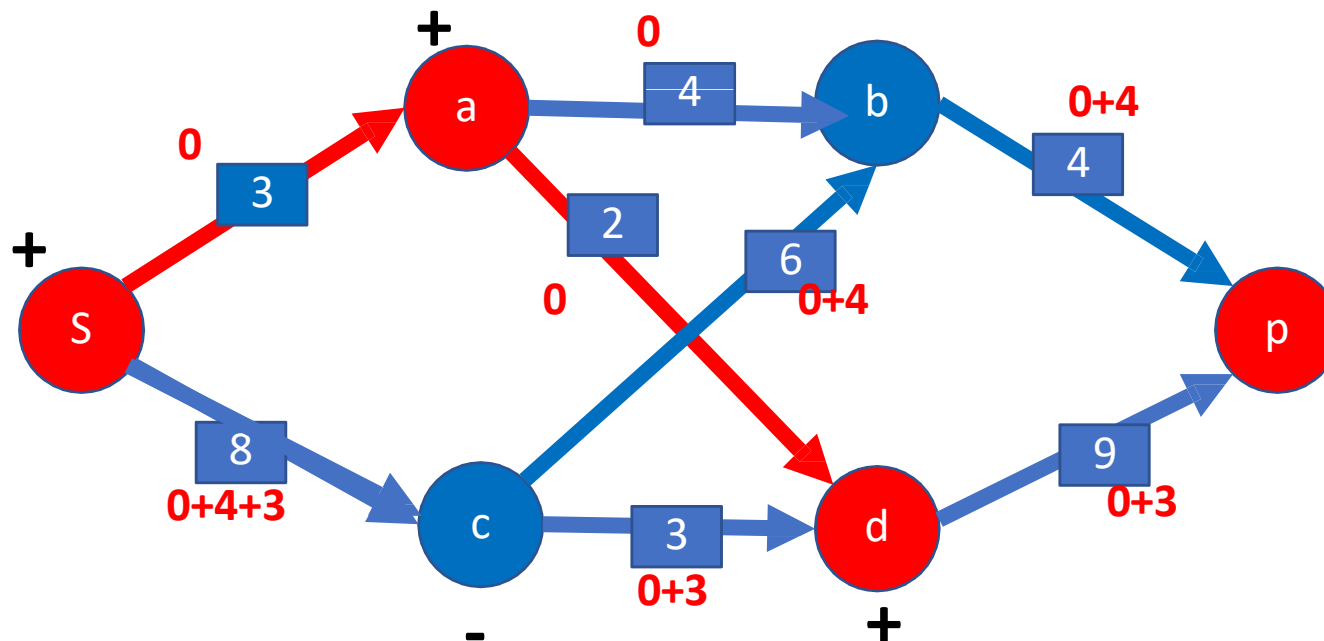
Ford-Fulkerson Algorithm

- Flow += 0+4+3
- S is marked
- a is marked
- b is marked
- c is marked by -
- We go back



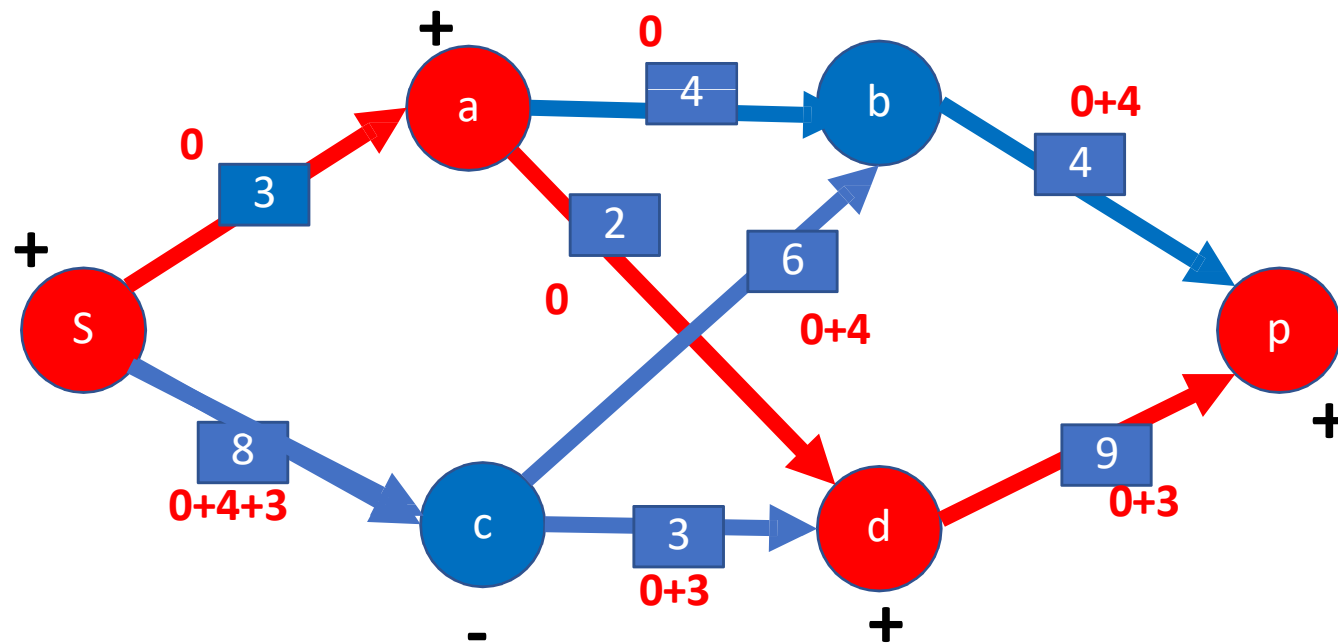
Ford-Fulkerson Algorithm

- Flow += 0+4+3
- S is marked
- a is marked
- We mark d from a because $0 < 2$



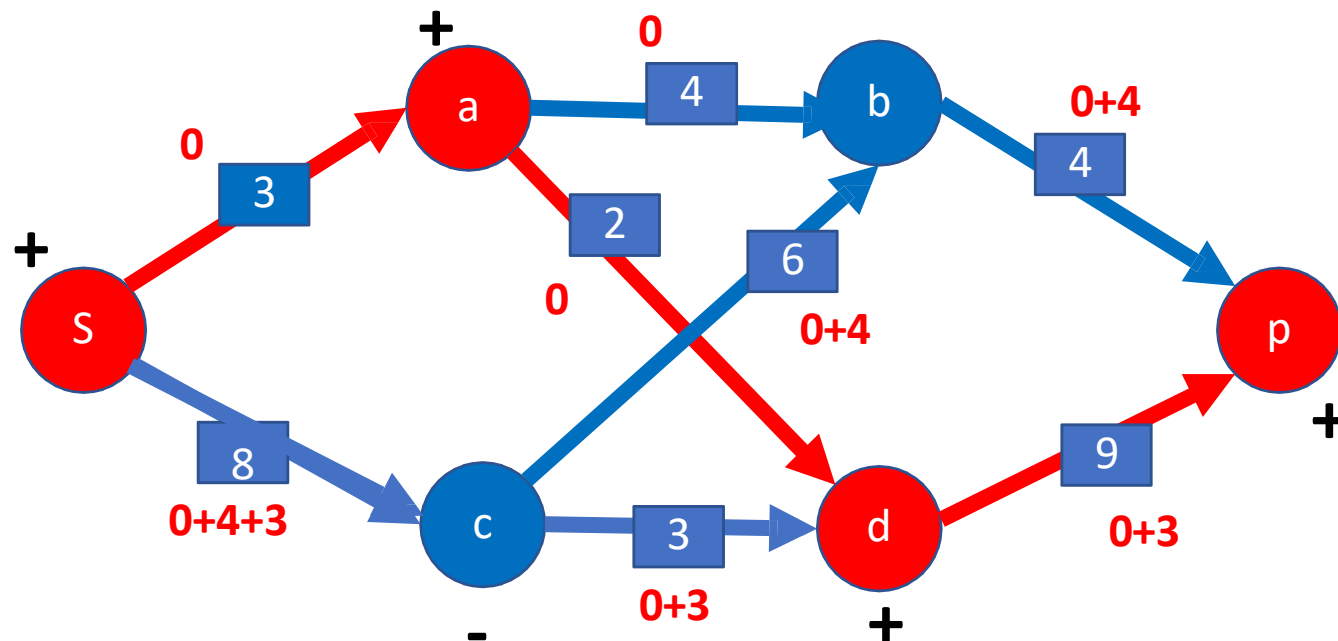
Ford-Fulkerson Algorithm

- Flow += 0+4+3
- S is marked
- a is marked
- d is marked
- We mark P from d because $0+3 < 9$



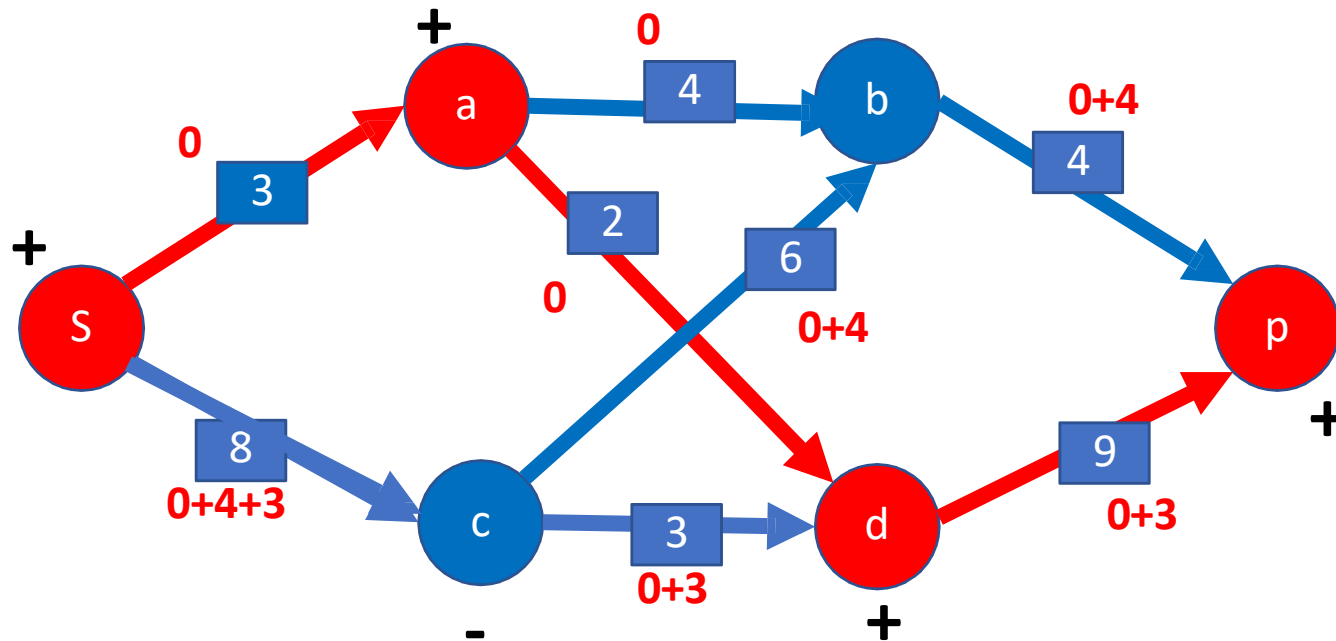
Ford-Fulkerson Algorithm

- Flow+ = 0+4+3
- S is marked
- a is marked
- d is marked
- P is marked
- The improving chain is S a d P with an increase in flux = $\min(3-0.2-0.9-3)=2$



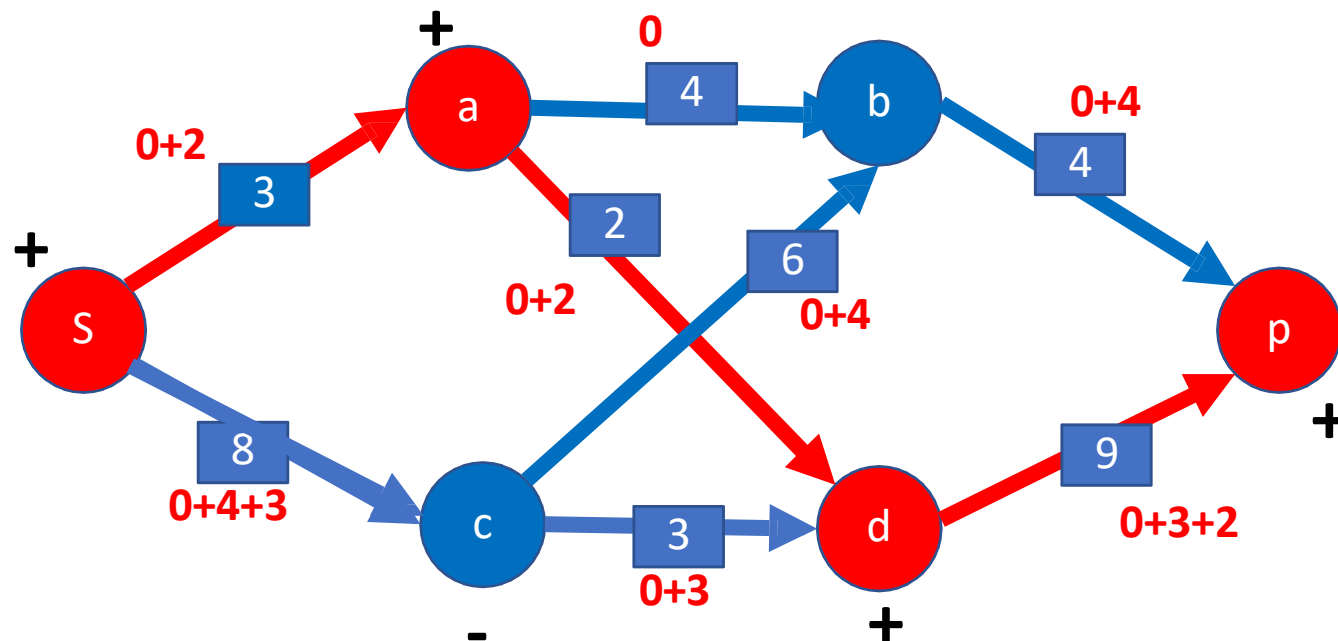
Ford-Fulkerson Algorithm

- $\text{Flow} += 0 + 4 + 3 + 2$
- S is marked
- a is marked
- d is marked
- P is marked
- The improving chain is $S \rightarrow a \rightarrow d \rightarrow P$ with an increase in flux = $\min(3 - 0.2 - 0.9 - 3) = 2$



Ford-Fulkerson Algorithm

- Flow+ = 0+4+3+2
- S is marked
- a is marked d is
- marked
- P is marked
- The improving chain is S a d P with an increase in flux = $\min(3-0.2-0.9-3)=2$



Ford-Fulkerson Algorithm

- $\text{Flow} = 0 + 4 + 3 + 2$
- If we no longer find an improving chain (this is the case for this network because (b p) (a d) and (c d) are saturated)
- So, the flow maximum = $0 + 4 + 3 + 2 = 9$

