Algoritmos e Estruturas de Dados

Marcelo Lobosco
DCC/UFJF

Fluxo Máximo em Redes

Parte 2 - Aula 03

Agenda

- Grafos
  - Fluxo Máximo em Redes
    - Introdução
    - Algoritmo de Fork-Fulkerson
    - DMKM

O Problema do Fluxo Máximo

- Um dos problemas mais importantes de otimização em grafos
  - Devido à freqüência com que surgem aplicações práticas
- Consiste em colocar uma “rede em equilíbrio” quanto à distribuição dos fluxos que podem ser enviados entre dois pontos da rede
  - Sujeito às limitações impostas, como capacidade de cada aresta

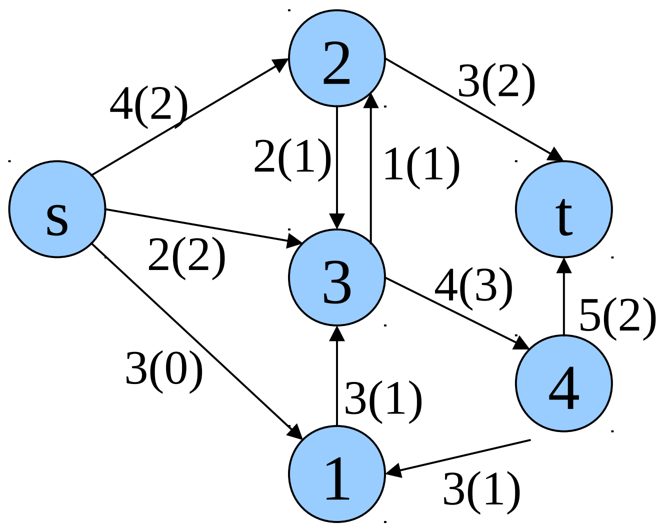
Modelo Matemático

- Maximizar f sujeito à

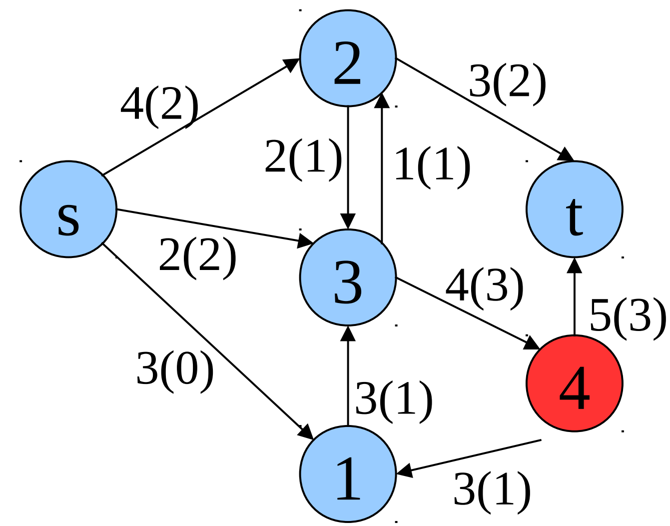
$$\begin{cases} \displaystyle\sum_{j=1}^{n} x_{ij} - \sum_{j=1}^{n} x_{ji} = \begin{cases} f, \text{ se } i = s \\ 0, \text{ se } i \neq s, i \neq t \\ -f, \text{ se } i = t \end{cases} \\ 0 \leq x_{ij} \leq u_{ij} \end{cases}$$

f = fluxo entre os nós s e t
x_{ij} = quantidade de fluxo enviada no arco (i,j)
u_{ij} = capacidade do arco (i,j)
Equação de conservação de Fluxos => A quantidade de fluxo que chega é igual a que sai

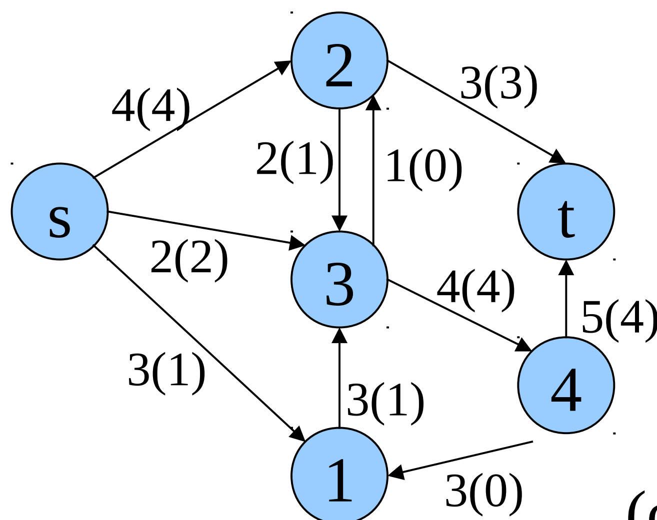
Fluxo em Redes



(a) um fluxo



(b) um fluxo ilegal



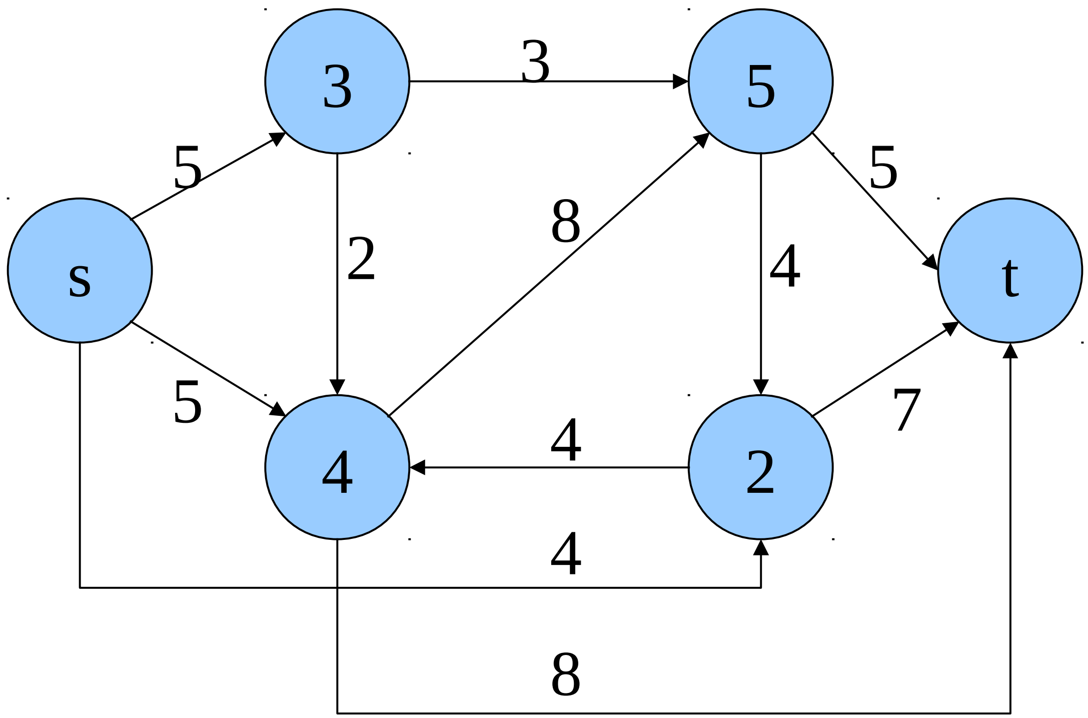
(c) um fluxo máximo

Corte Separador de G

- Dado o grafo direcionado G = G (N,A), com |N| = n, chamamos de corte separador $(X,\overline{X})$ de G o conjunto
  - $X \leq N : \overline{X} = N - X$ tal que $s \in X$, $t \notin X$
  - $(X,\overline{X}) = \{(i,j) \in A \,/\, i \in X, j \in \overline{X}\}$
- A capacidade de um corte separador $(X,\overline{X})$ de G é definido como

$$U(X,\overline{X}) = \sum_{(i,j) \in (X,\overline{X})} u_{ij}$$

Exemplo



X	$\overline{X}$	$(X,\overline{X})$	$u(X,\overline{X})$
s	2,3,4,5,t	(s,2), (s,3),(s,4)	14
s,4,2	3,5,t	(s,3),(2,t),(4,5),(4,t)	28

O Problema do Fluxo Máximo

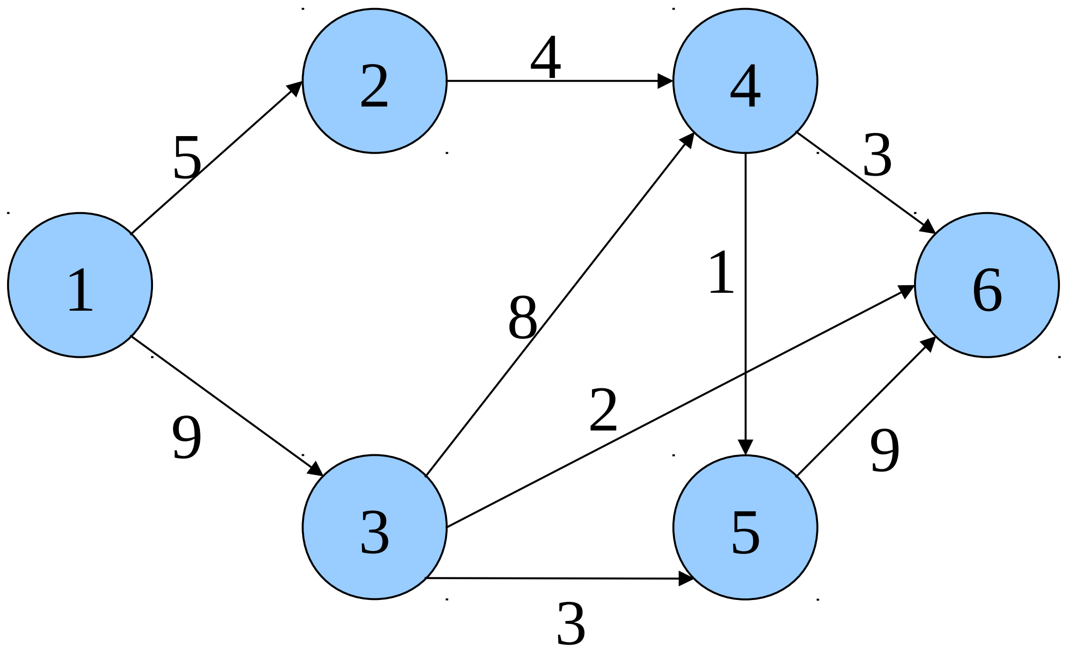
- Lema
  - O fluxo máximo de f será menor ou igual a capacidade de qualquer corte separador $(X,\overline{X})$ de G, isto é:

$u(X,\overline{X}) \geq f, \forall (X,\overline{X})$ corte separador de G

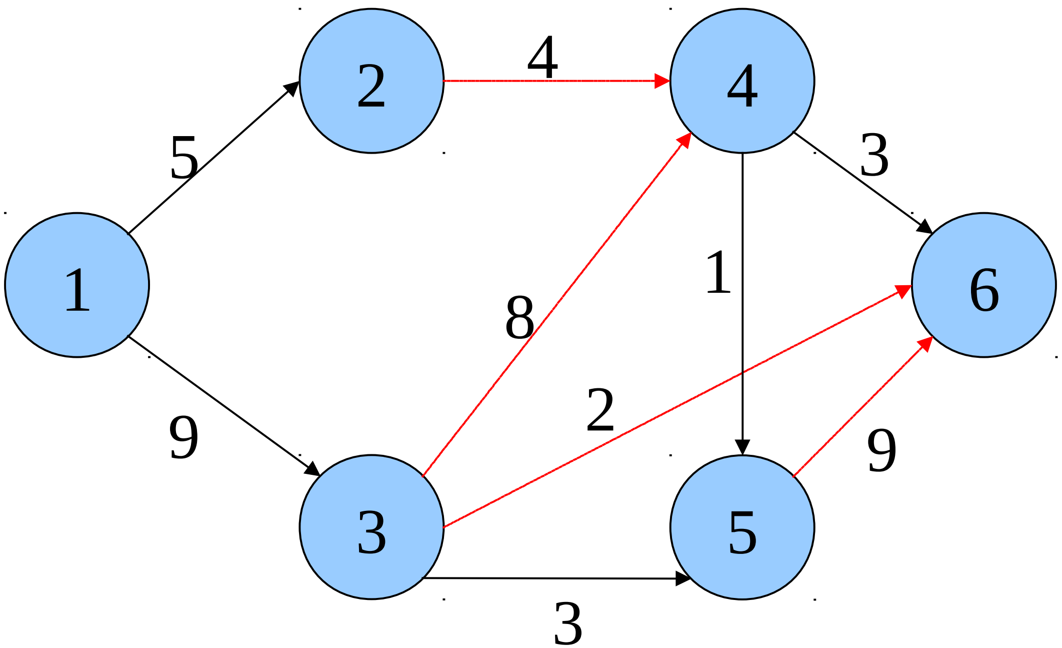
- Teorema
  - Seja $(X,\overline{X})$ o corte separador de G que possui a menor capacidade dentre todos os cortes separadores de G, então:

$U(X,\overline{X}) = f$

O Problema do Fluxo Máximo

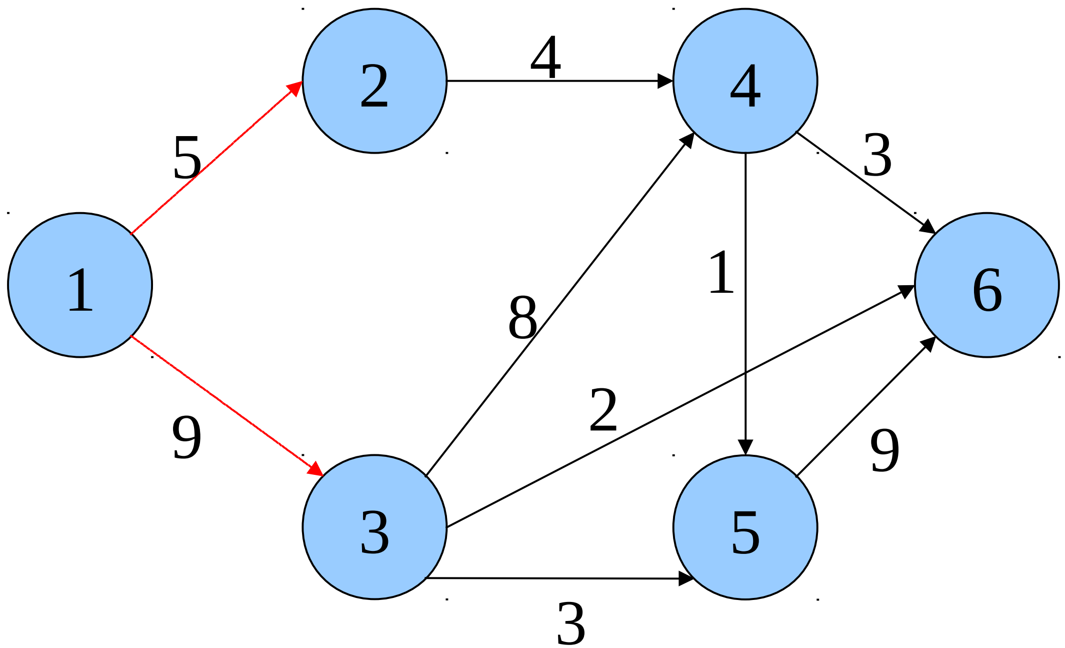


O Problema do Fluxo Máximo



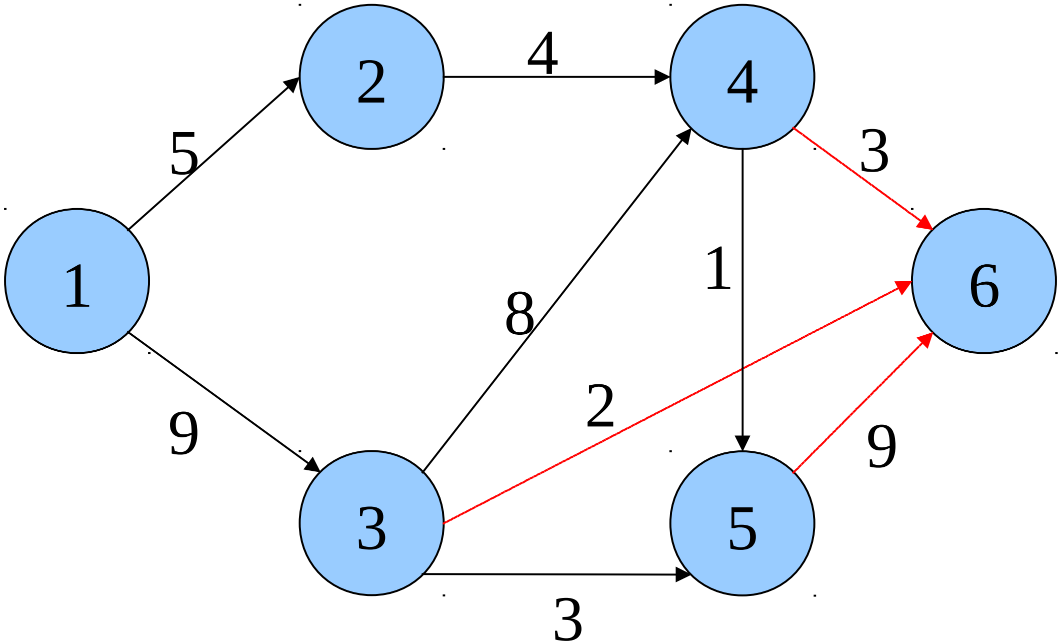
capacidade = 4 + 8 + 2 + 9 = 23

O Problema do Fluxo Máximo



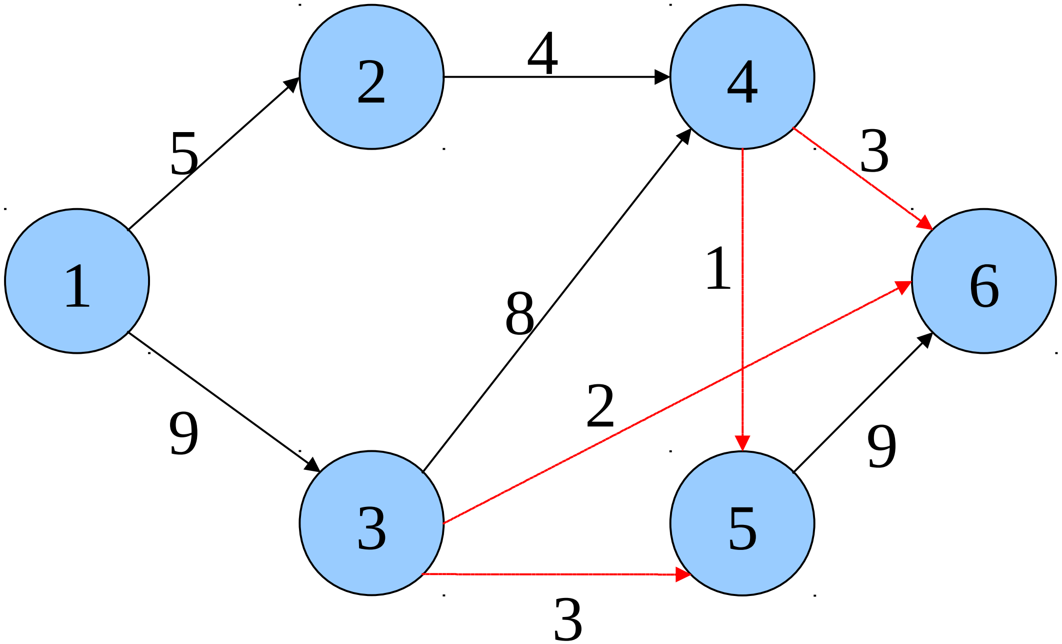
capacidade = 9 + 5 = 14

O Problema do Fluxo Máximo



capacidade = 9 + 2 + 3 = 14

O Problema do Fluxo Máximo



capacidade = 3 + 2 + 1 + 3 = 9

Algoritmo de Ford-Fulkerson

- Passo 1: Iniciar com uma distribuição de fluxo nula, isto é, fazer $x_{ij} = 0 \ \forall \ (i,j) \in A \Rightarrow f = 0$
- Passo 2: Tentar aumentar o valor de f efetuando a seguinte seqüência de passos:
  - Construir uma rede transformada G' a partir de G da seguinte forma:
    - Todos os nós de G estão em G'
    - Se $x_{ij} < u_{ij}$ na atual solução em G, então gerar arco (i,j) em G', com a alteração máxima de fluxos permitida $\Delta_{ij} = u_{ij} – x_{ij}$
    - Se $x_{ij} > 0$ na atual solução de G, gerar arcos (j,i) em G' com a alteração máxima de fluxos permitida $\Delta_{ij} = x_{ij}$

Algoritmo de Ford-Fulkerson

- Observe que se $0 < x_{ij} < u_{ij}$ então geramos dois arcos para i e j, um no sentido $i \rightarrow j$ e outro no $j \rightarrow i$
- Construindo a rede G' com os respectivos Δ_{ij}, verificar se existe um caminho P' em G' entre s e t
  - Se existe esse caminho P', seja P a cadeia em G' associada a P' de G'

Algoritmo de Ford-Fulkerson

- Análise de cada arco P' e de P
  - Se um arco (i,j) de P' corresponde a um arco (i,j) de P faça $X_{ij} \leftarrow X_{ij} + \Delta$
  - Se corresponde a um arco (j,i) faça $X_{ji} \leftarrow X_{ji} - \Delta$
  - $\Delta = \min \{\Delta_{ij} / ij \in P' \}$
- Com isso atualizamos a solução em G, e retornamos ao inicio do passo 2, construindo nova G' até que em G' não exista mais caminho entre s e t

Algoritmo de Ford-Fulkerson

- Algoritmo para um caminho entre s e t ou para detectar a não existência de 1 caminho
- Iniciação
  - pred (j) $\leftarrow \emptyset\ \forall\ j \in N$
  - L = {s}

Algoritmo de Ford-Fulkerson

- Iterações

Enquanto L ≠ ∅ e t não estiver rotulado faça

Selecionar um nó i ∈ L

Examinar (i)

Remover nó i de L

Se t estiver rotulado então use os pred(j) para recuperar o caminho P' de s até t

Senão não existe caminho entre s e t em G'

Algoritmo de Ford-Fulkerson

- Examinar (i)

Para todo j não rotulado: $x_{ij} < u_{ij}$, faça

Pred (j) = +i

rotular j = Δ_j, $\Delta_j = \min \{ \Delta_i ,\ u_{ij} - x_{ij} \}$

incluir j em L

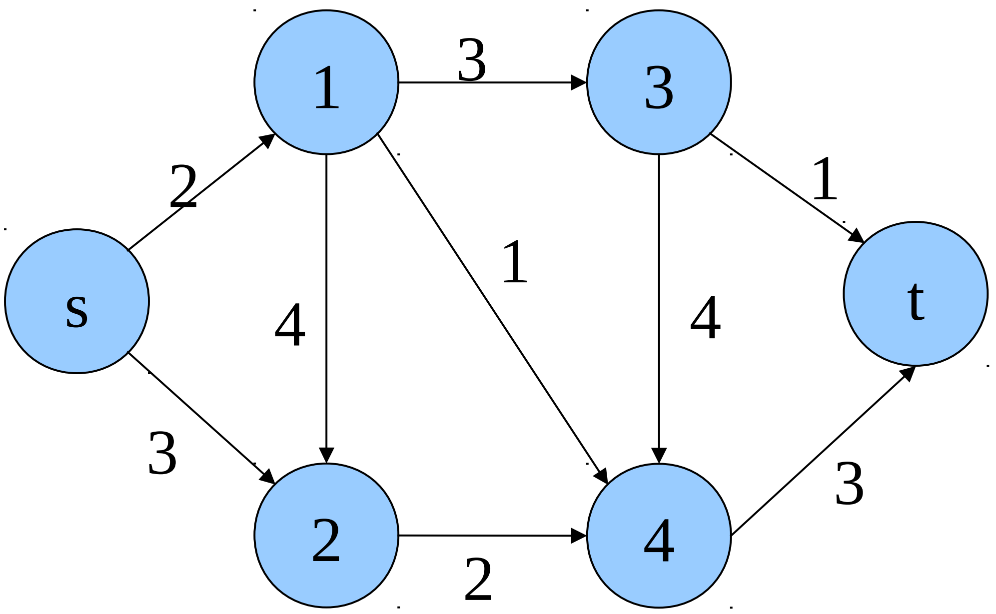
Para todo j não rotulado: $x_{ij} > 0$, faça

Pred (j) = -i

rotular j = Δ_j , $\Delta_j = \min \{ \Delta_i ,\ x_{ij} \}$

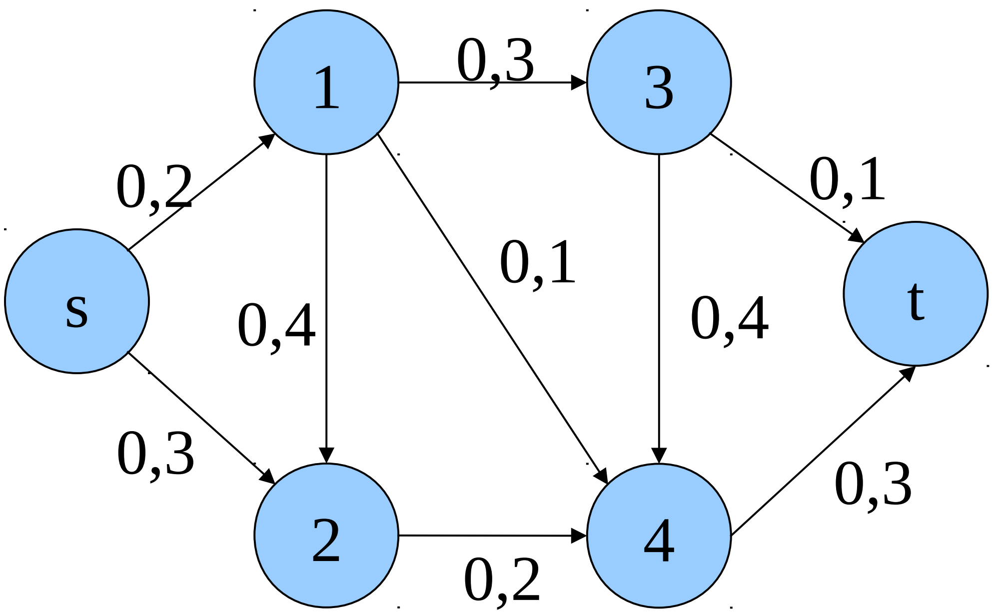
incluir j em L

Algoritmo de Ford-Fulkerson



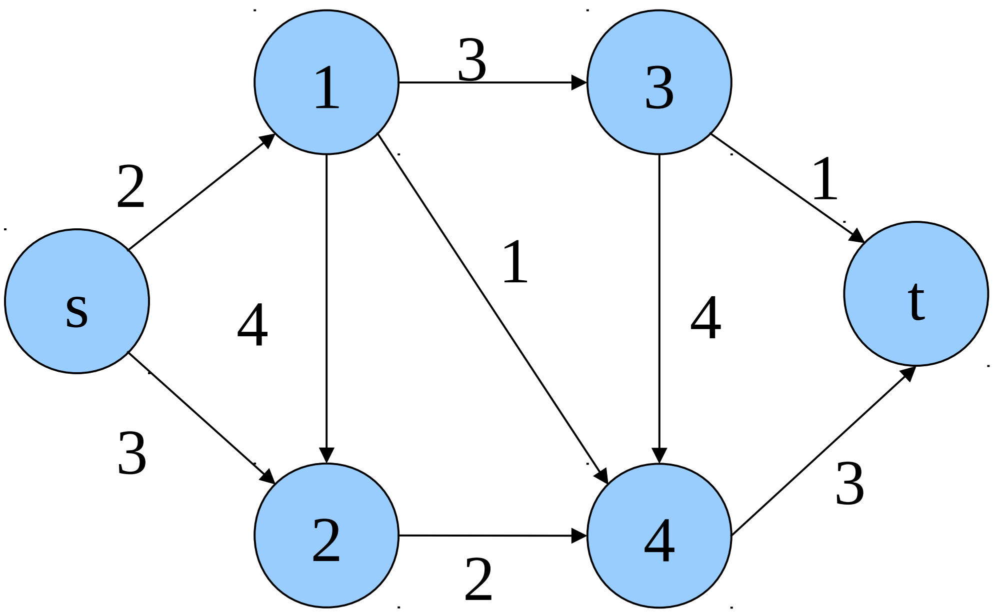
Algoritmo de Ford-Fulkerson

Iniciação



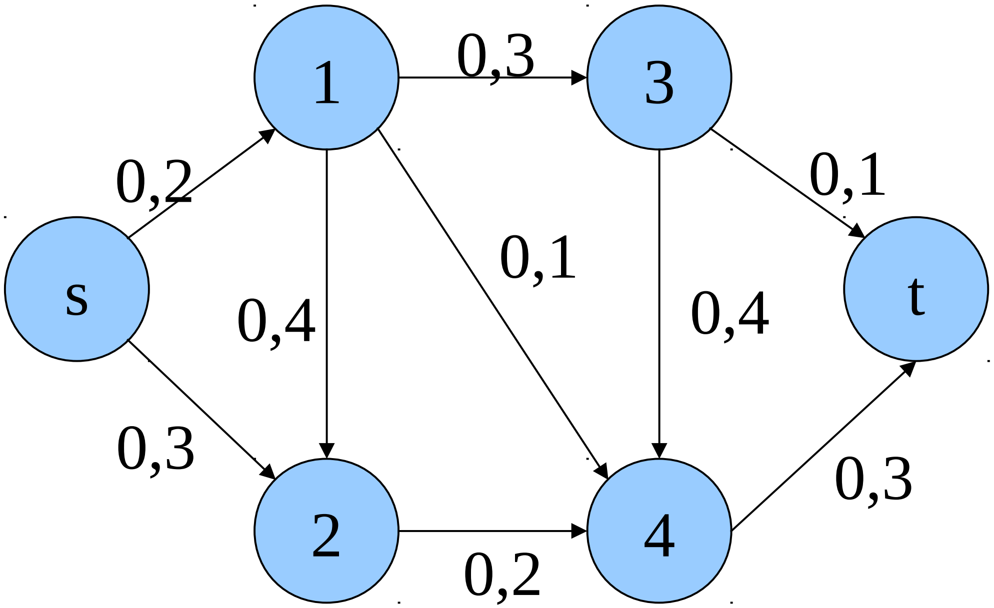
Algoritmo de Ford-Fulkerson

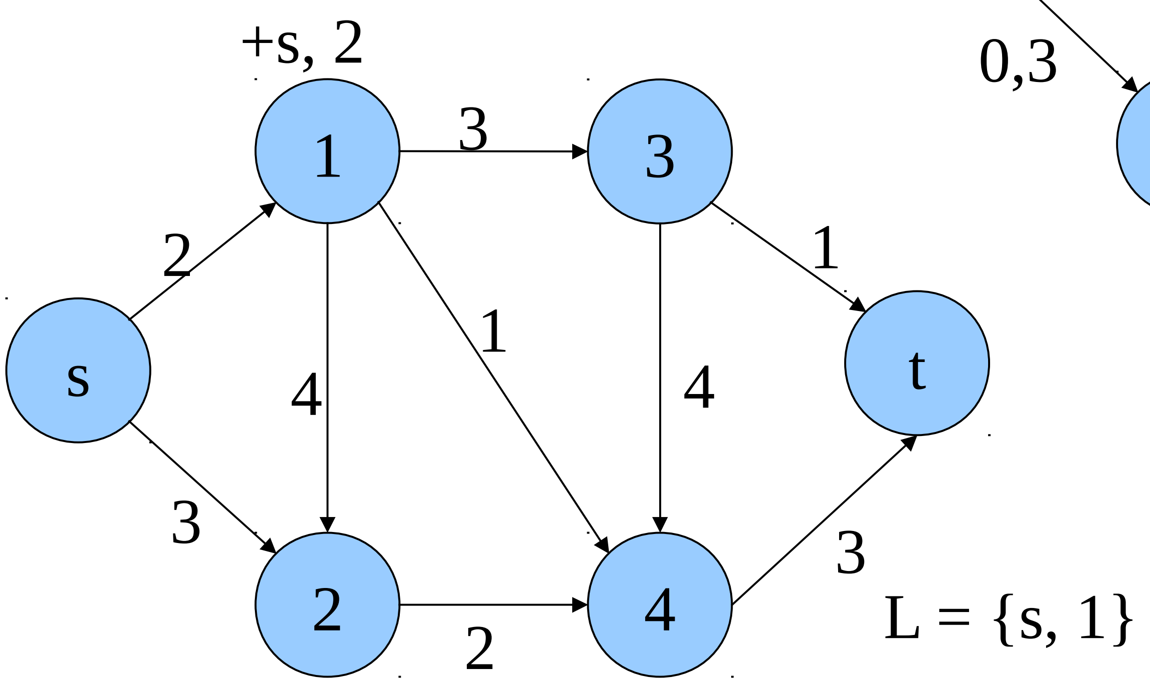
G’



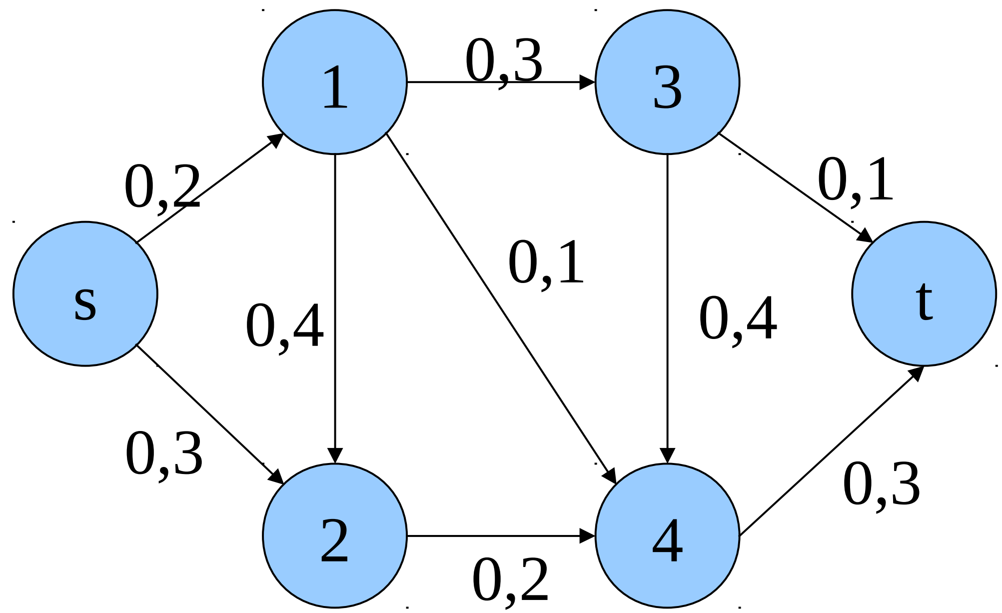
L = {s}

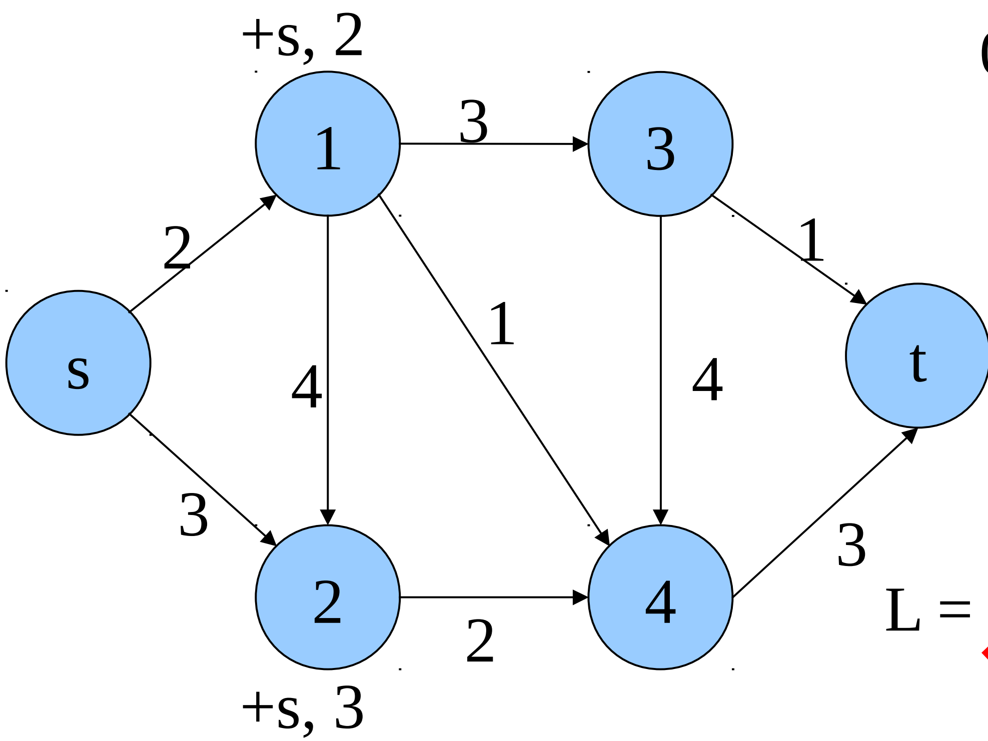
Algoritmo de Ford-Fulkerson





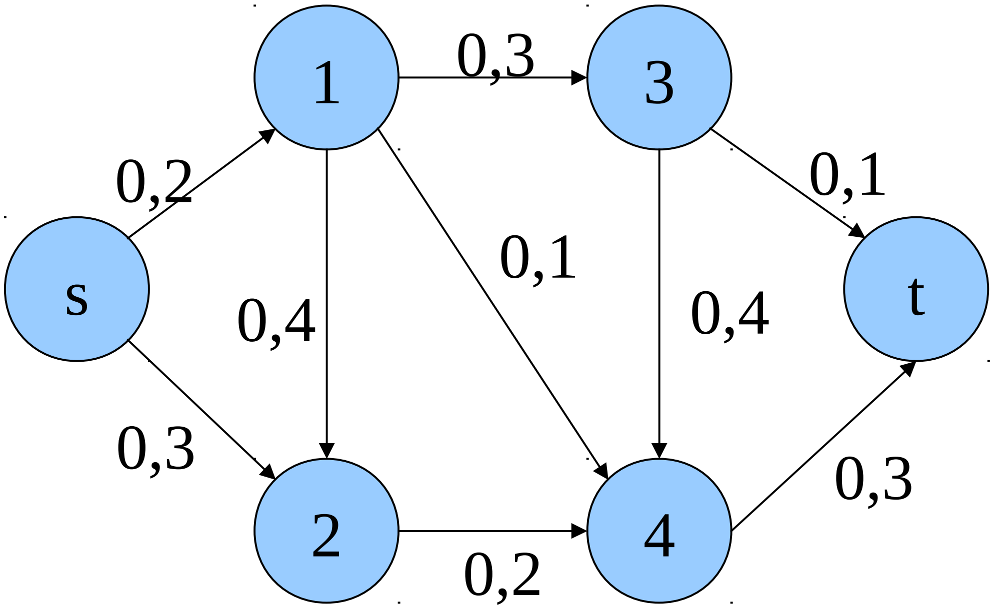
Algoritmo de Ford-Fulkerson

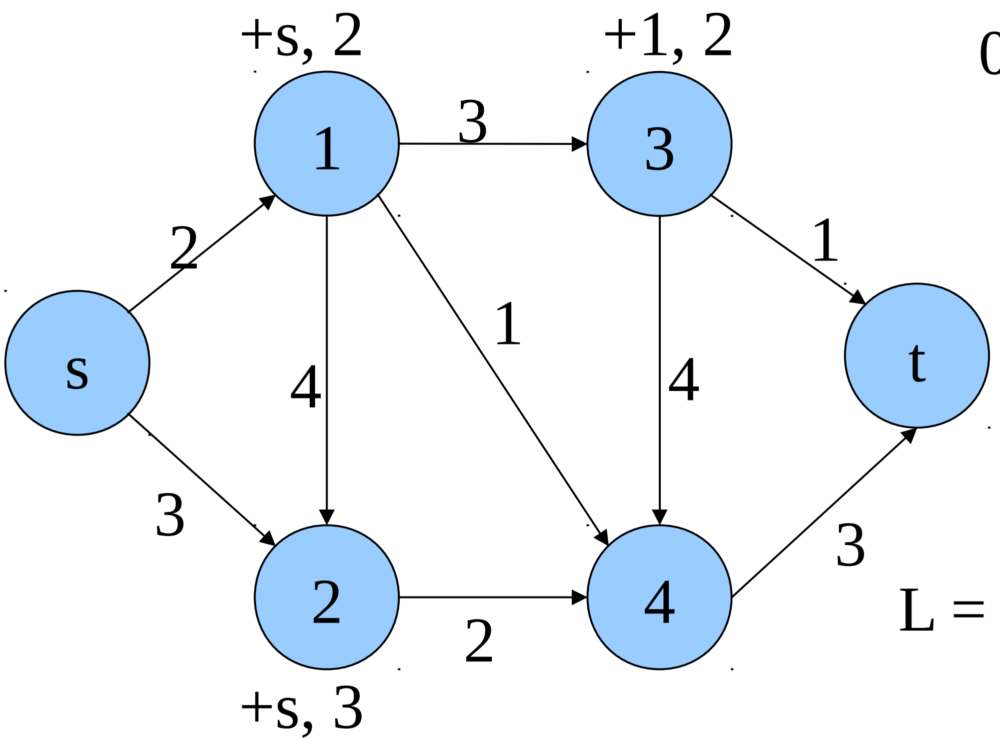




$L = \{\cancel{s}, 1, 2\}$

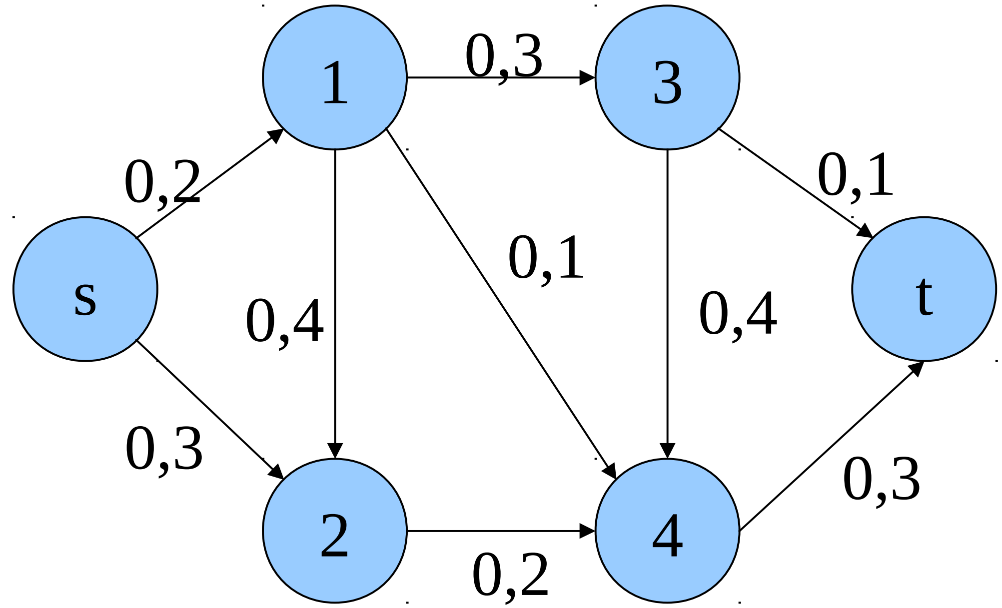
Algoritmo de Ford-Fulkerson

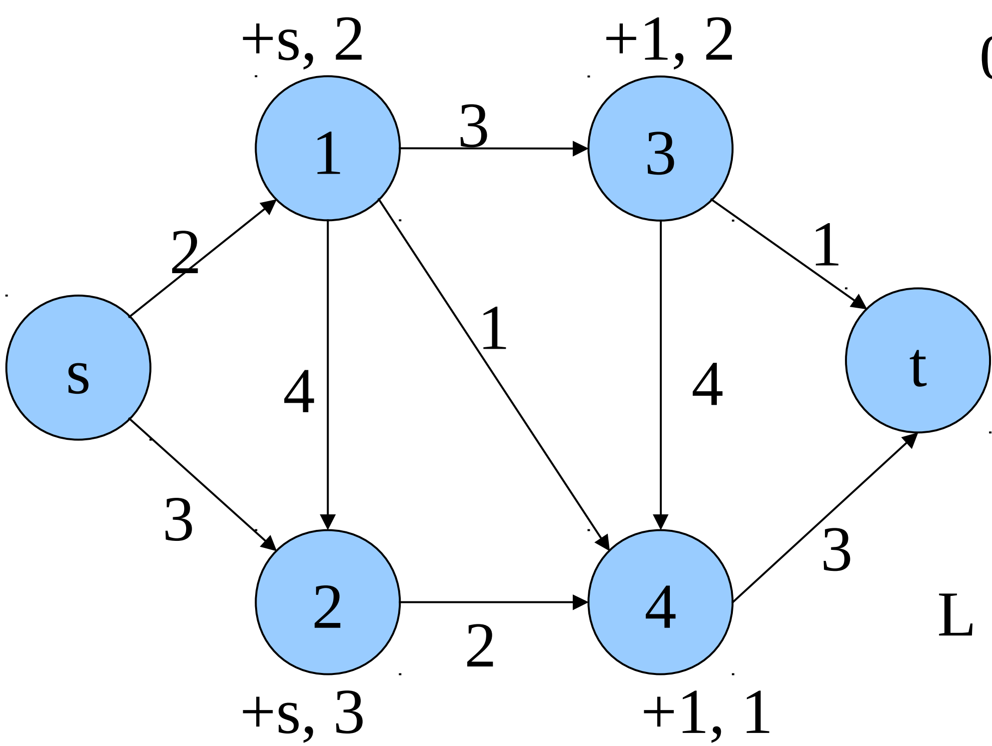




$L = \{\cancel{s}, 1, 2, 3\}$

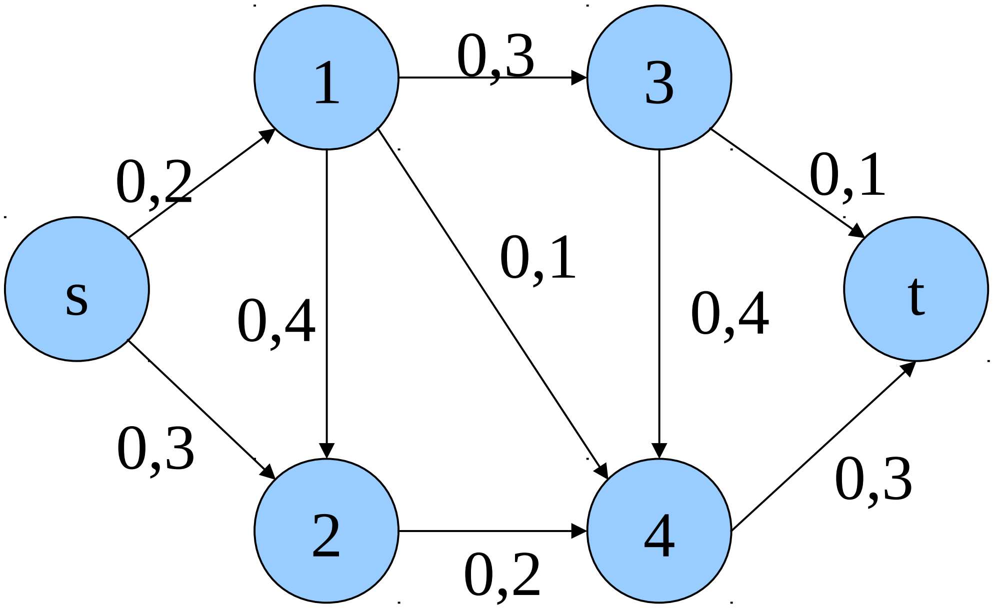
Algoritmo de Ford-Fulkerson

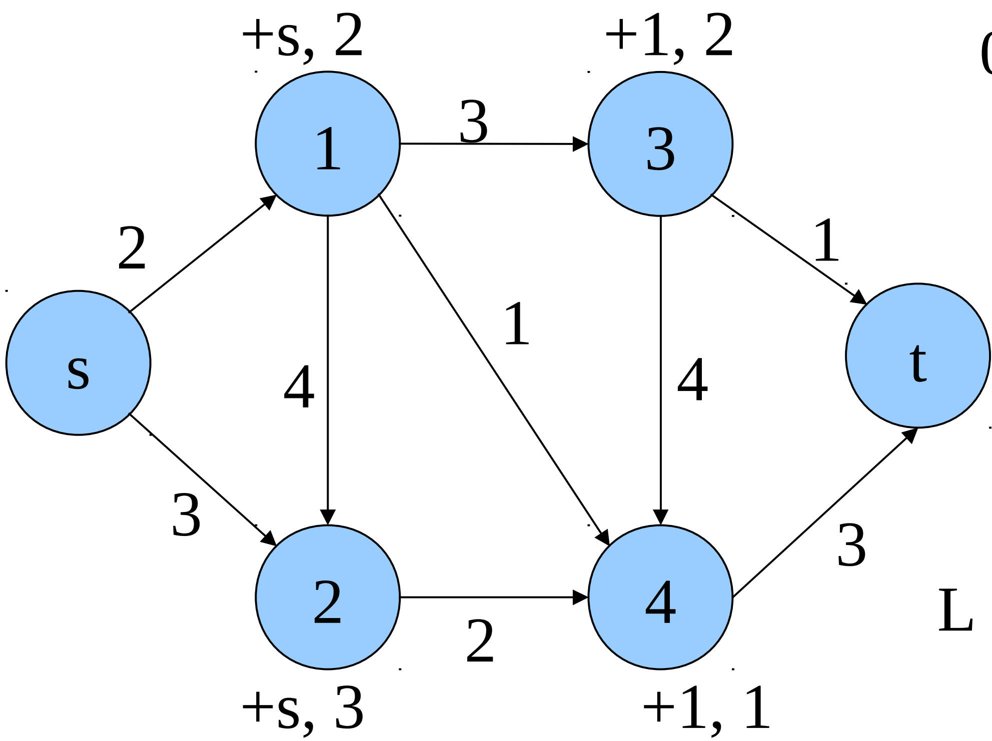




$L = \{\cancel{s}, \cancel{1}, 2, 3, 4\}$

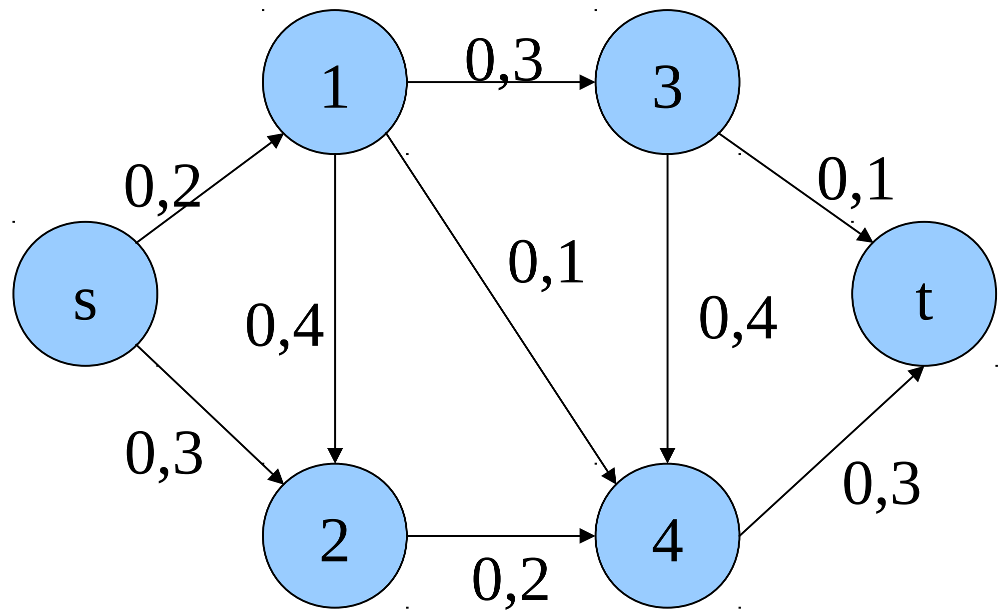
Algoritmo de Ford-Fulkerson

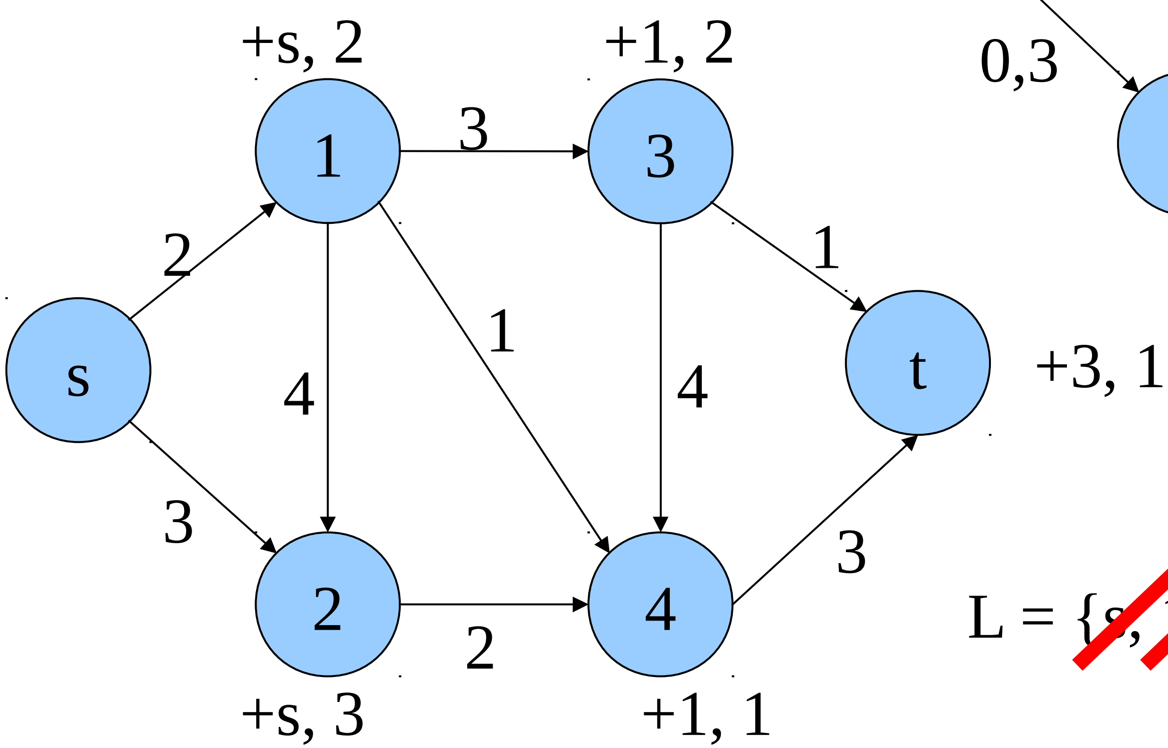




$L = \{\cancel{s}, \cancel{1}, \cancel{2}, 3, 4\}$

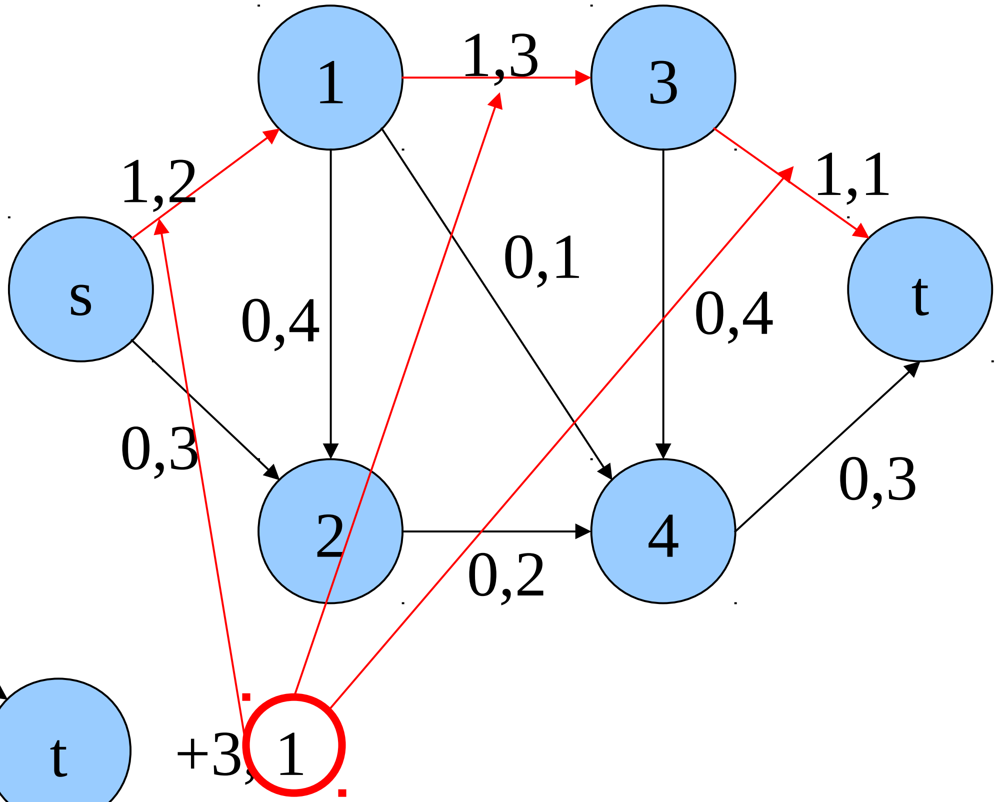
Algoritmo de Ford-Fulkerson

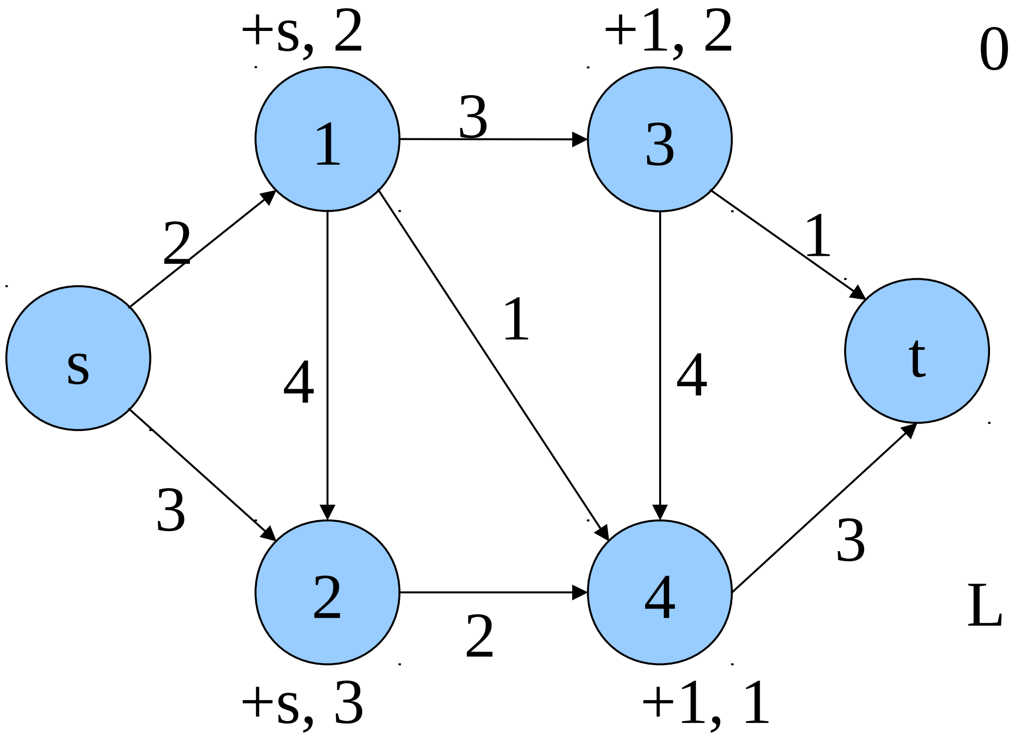




L = {~~s~~, ~~1~~, ~~2~~, ~~3~~, 4, t}

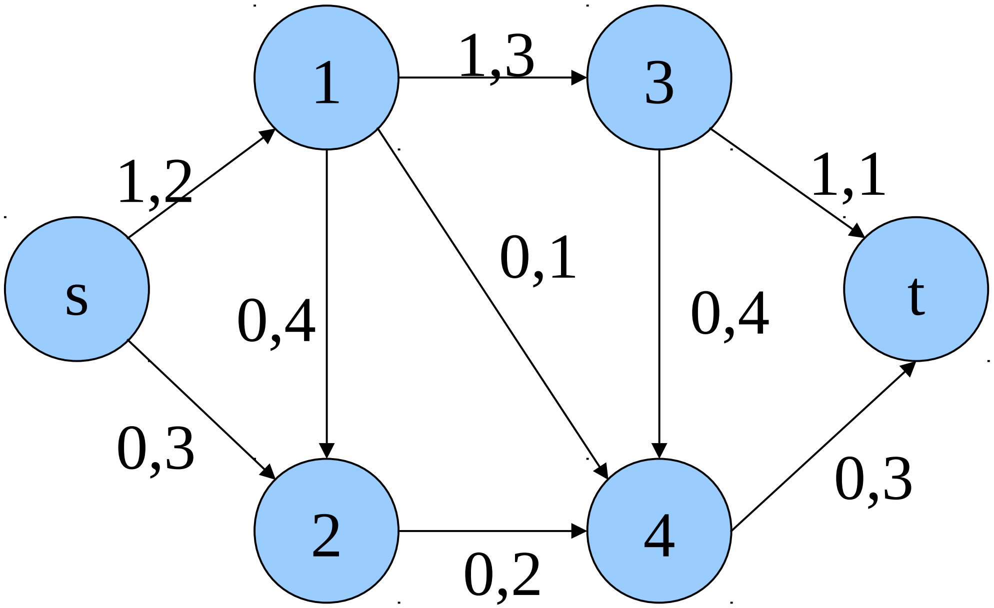
Algoritmo de Ford-Fulkerson

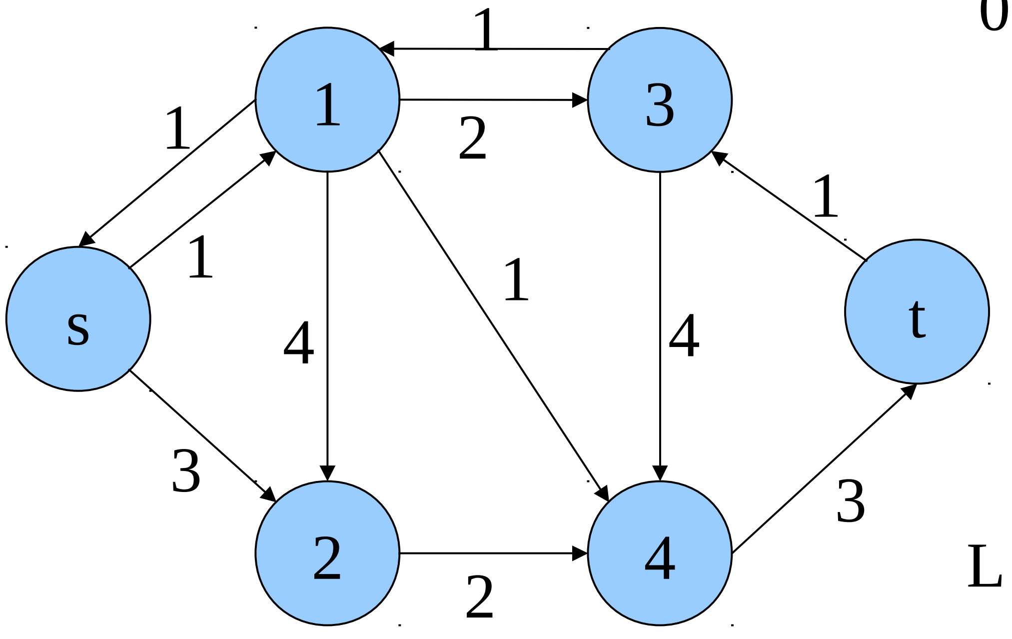




L = {~~s~~, ~~1~~, ~~2~~, ~~3~~, 4, t}

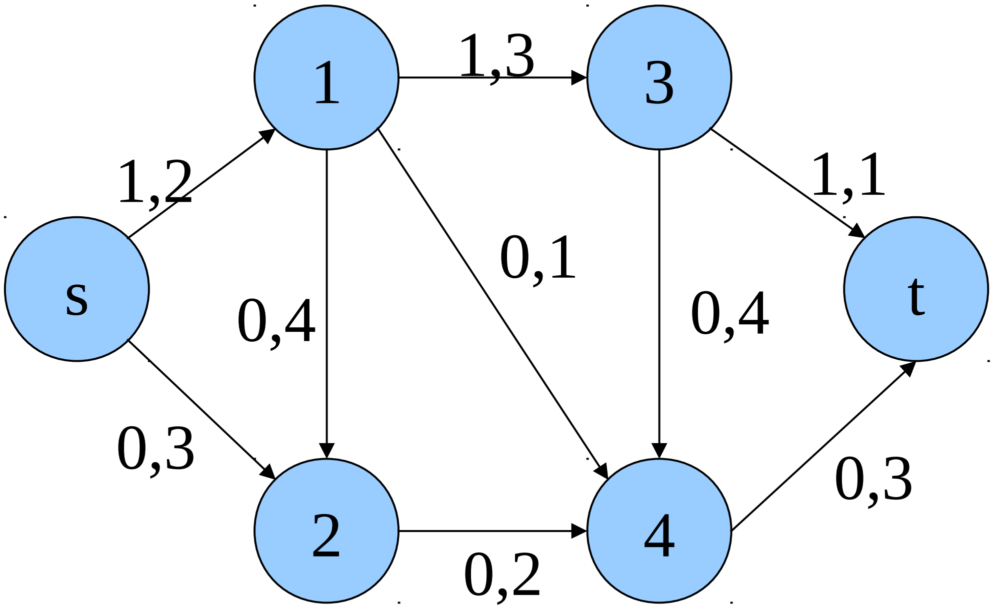
Algoritmo de Ford-Fulkerson

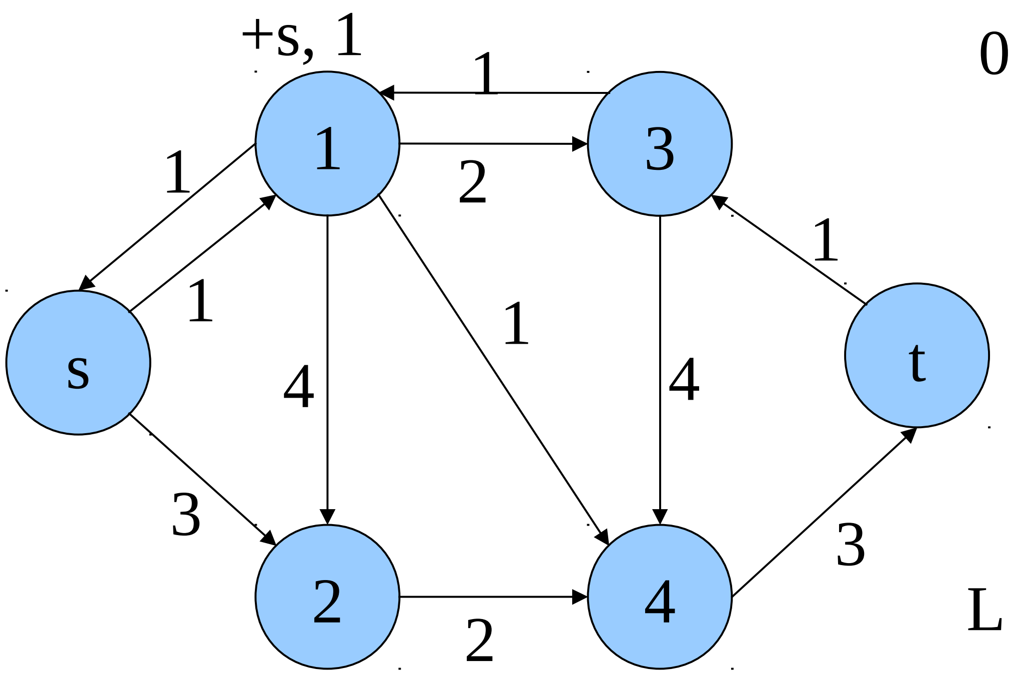




L = {s}

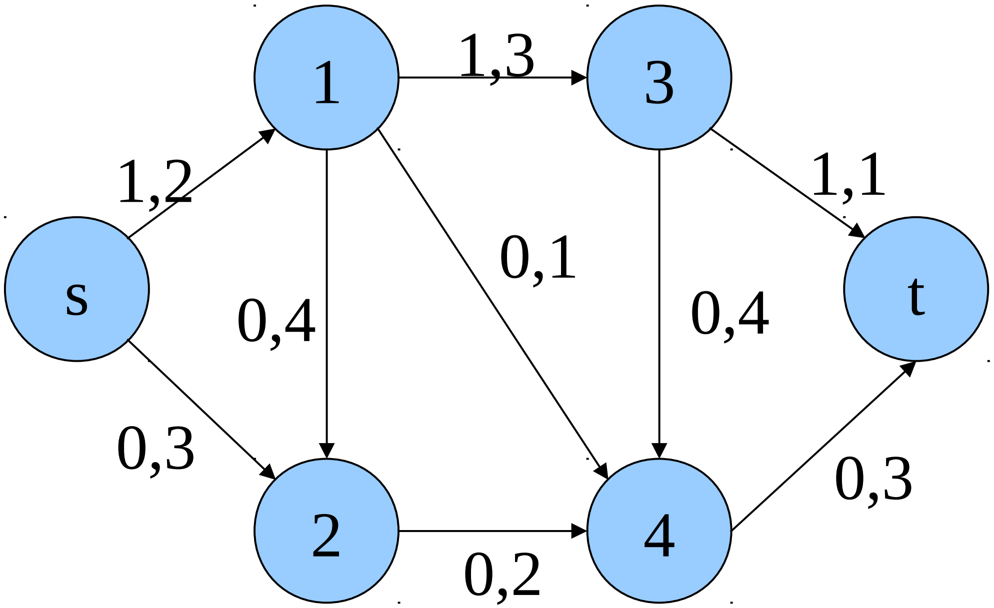
Algoritmo de Ford-Fulkerson

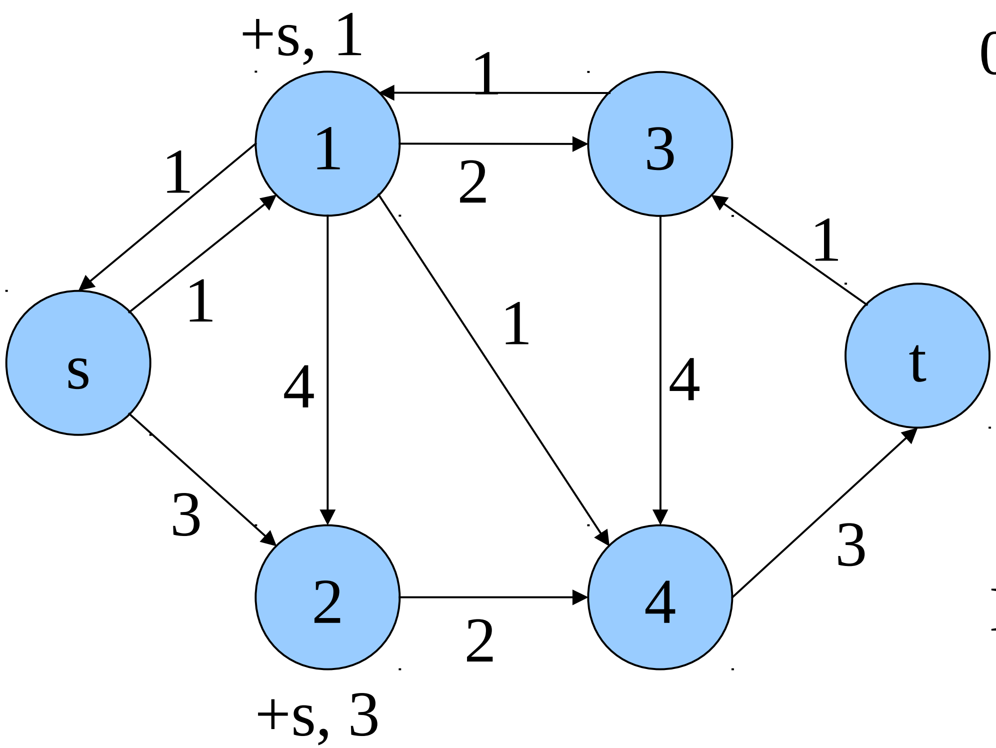




L = {s, 1}

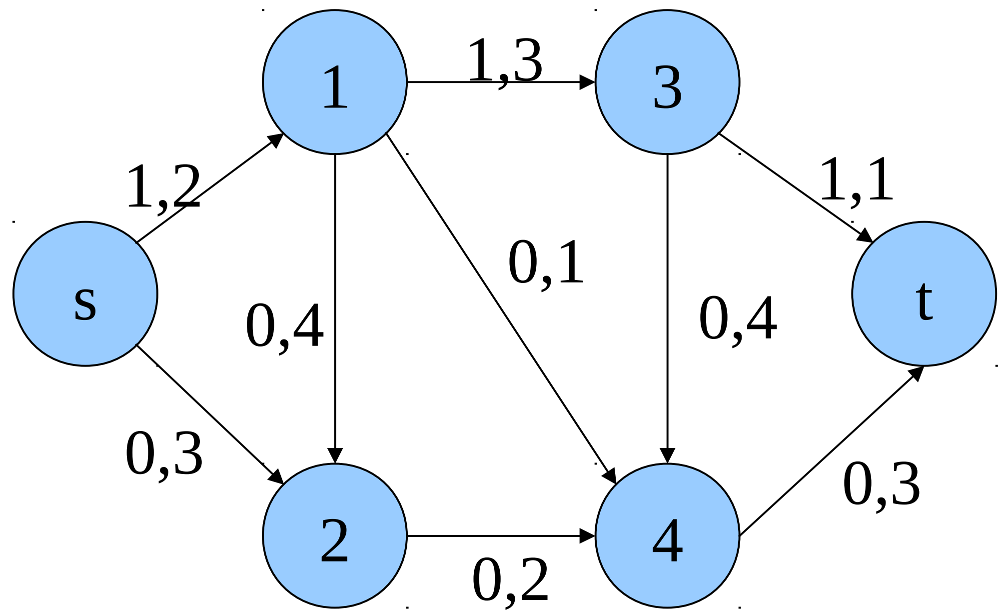
Algoritmo de Ford-Fulkerson

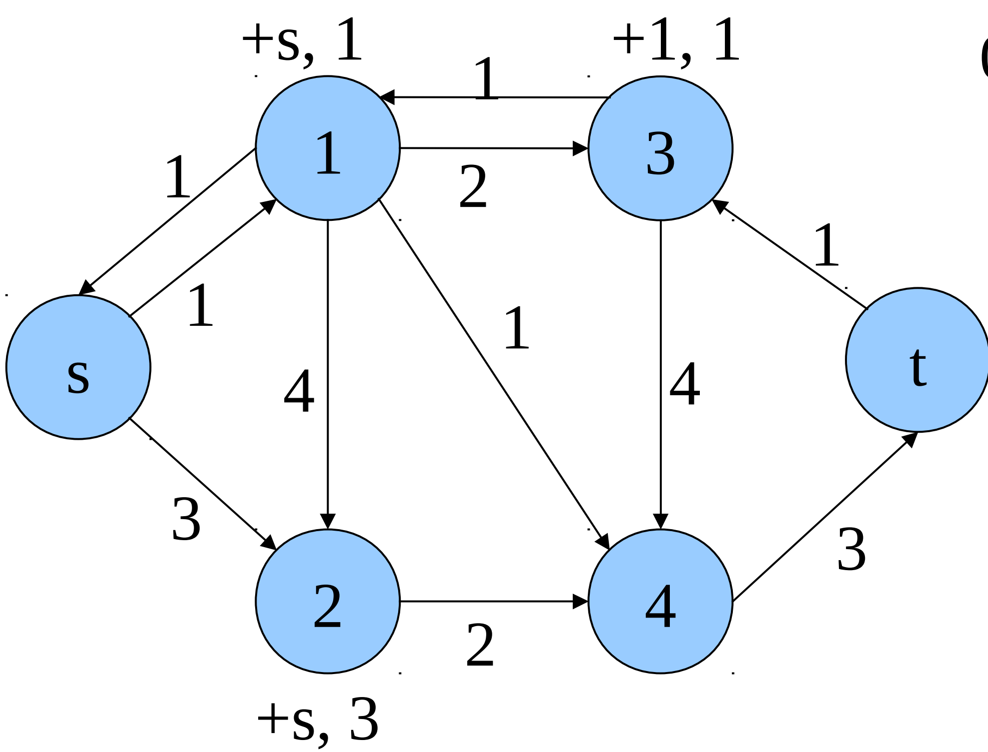




L = {~~s~~, 1, 2}

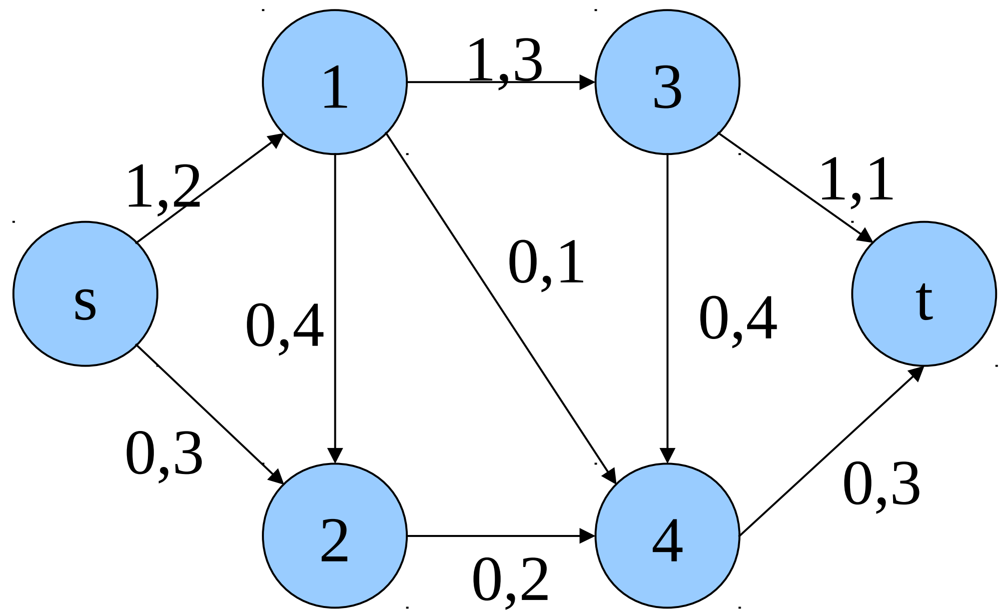
Algoritmo de Ford-Fulkerson

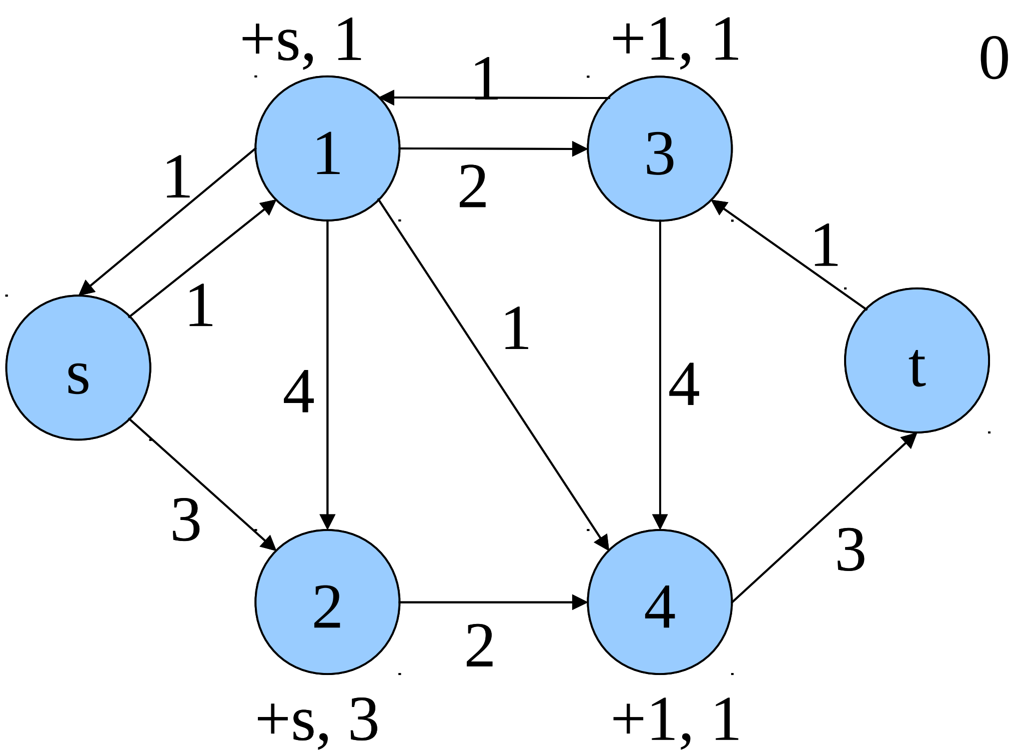




$L = \{\cancel{s}, 1, 2, 3\}$

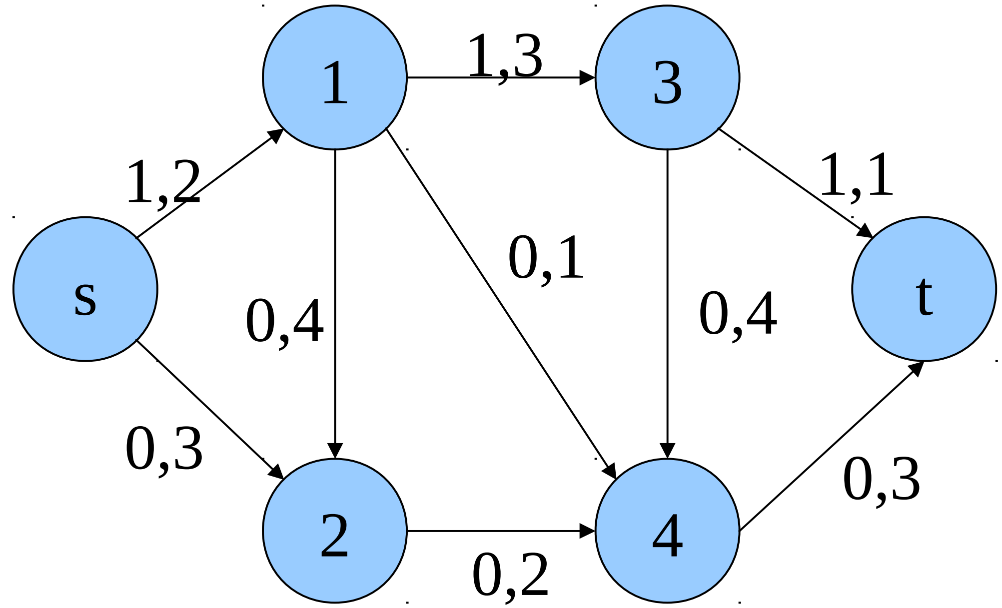
Algoritmo de Ford-Fulkerson

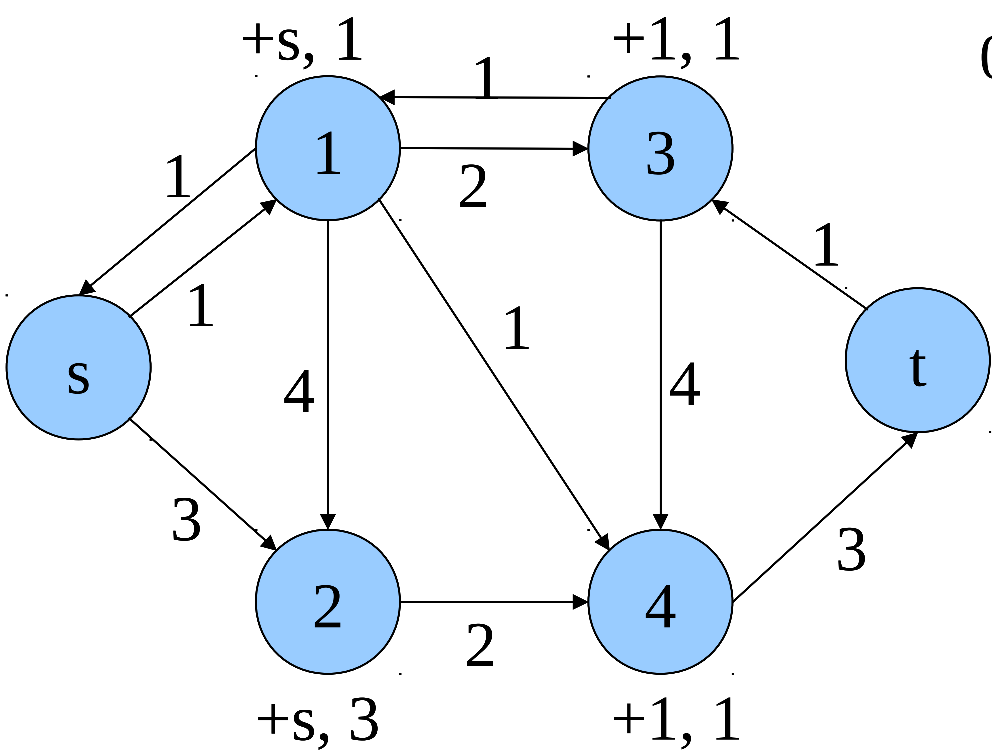




$L = \{\cancel{s}, \cancel{1}, 2, 3, 4\}$

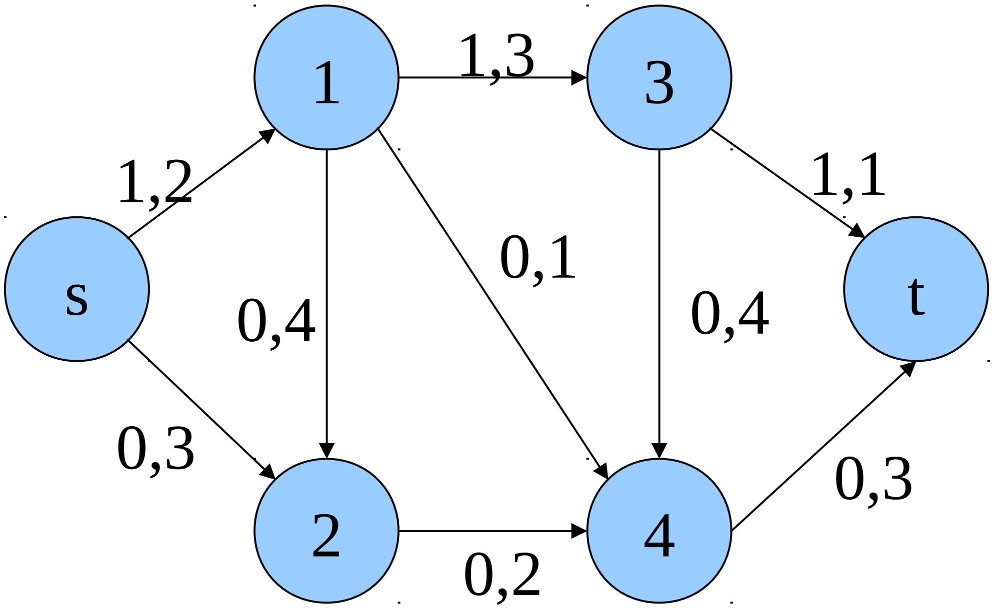
Algoritmo de Ford-Fulkerson

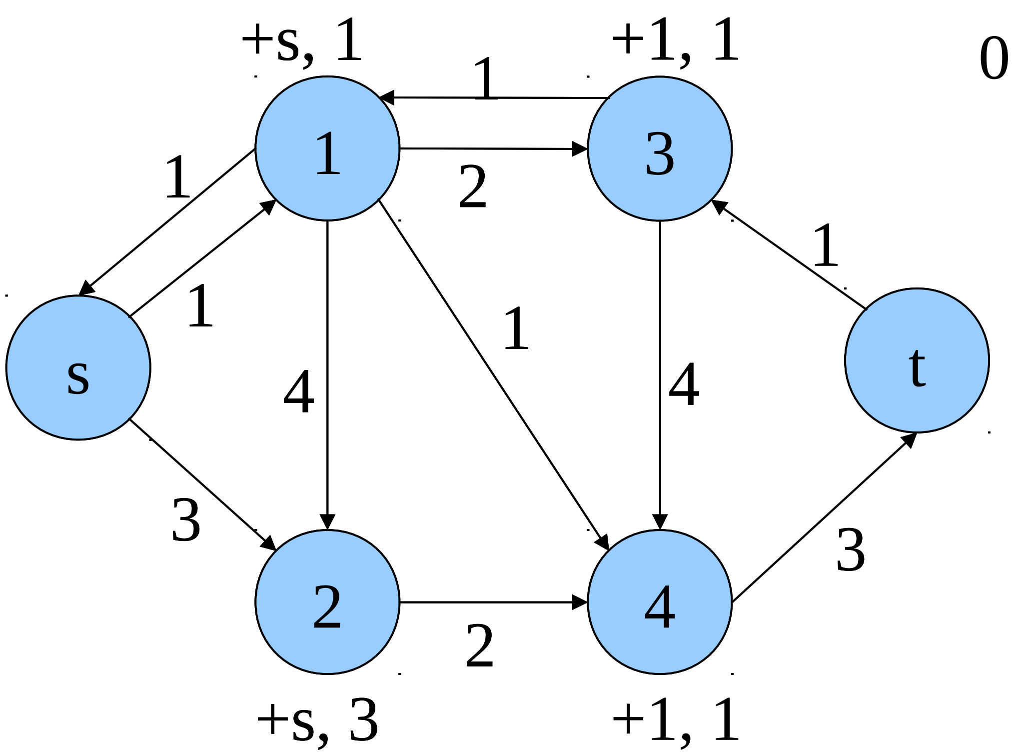




L = {~~s~~, ~~1~~, ~~2~~, 3, 4}

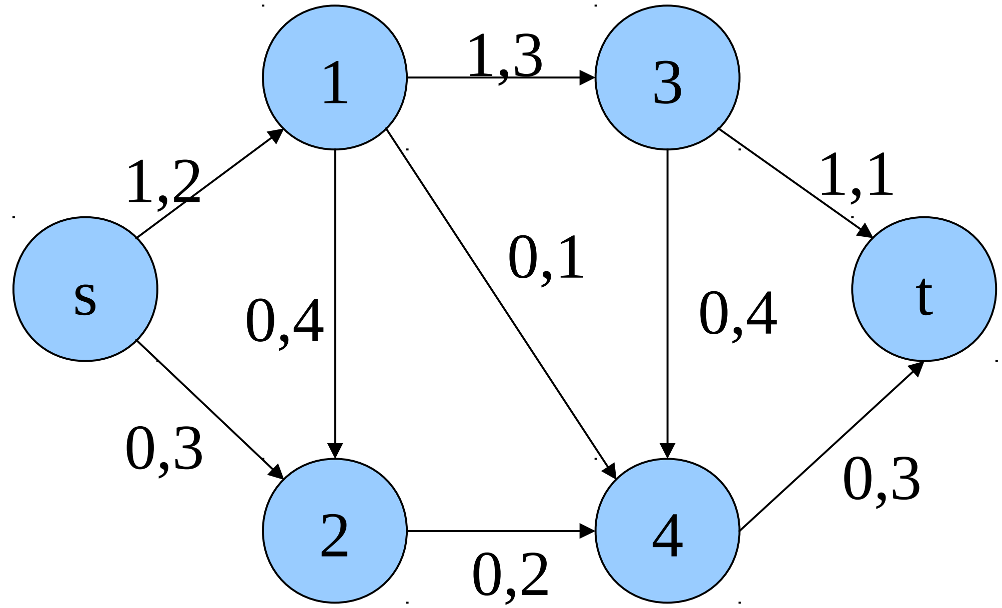
Algoritmo de Ford-Fulkerson

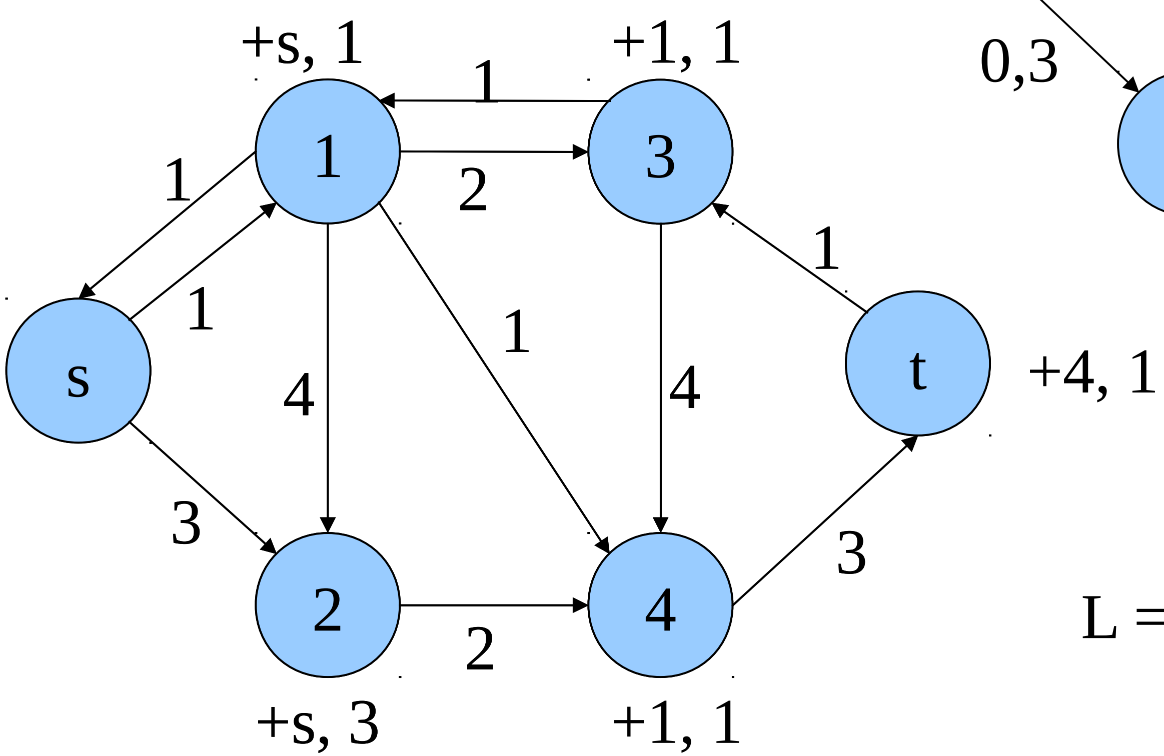




L = {~~s~~, ~~1~~, ~~2~~, ~~3~~, 4}

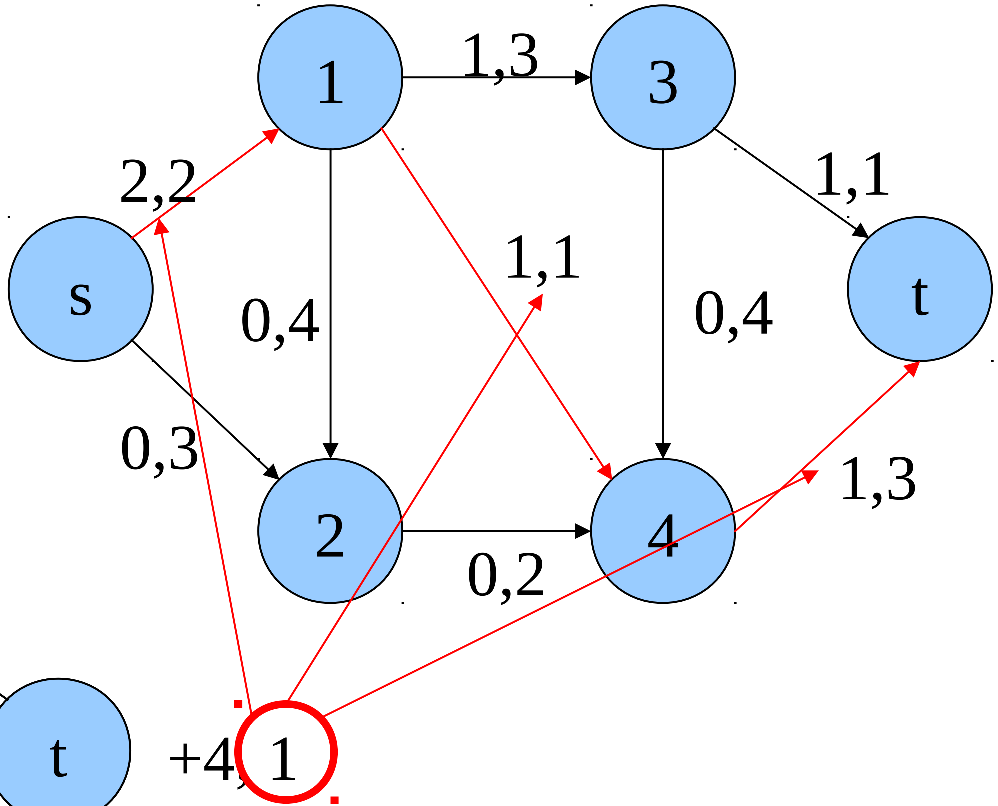
Algoritmo de Ford-Fulkerson

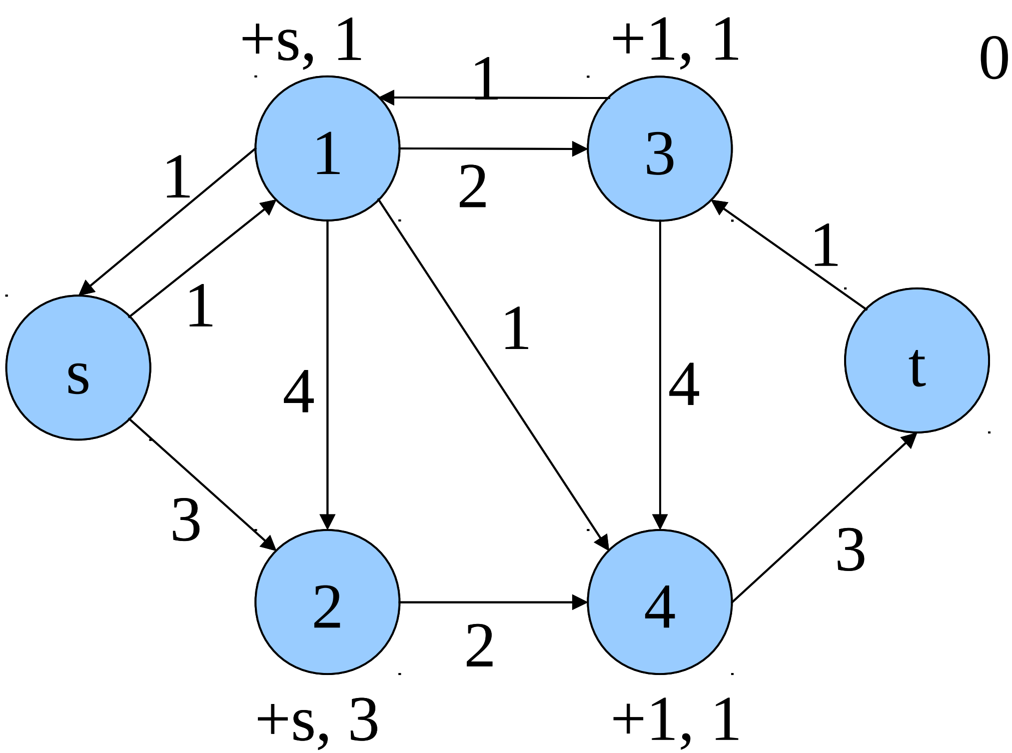




$L = \{\cancel{s}, \cancel{1}, \cancel{2}, \cancel{3}, \cancel{4}, t\}$

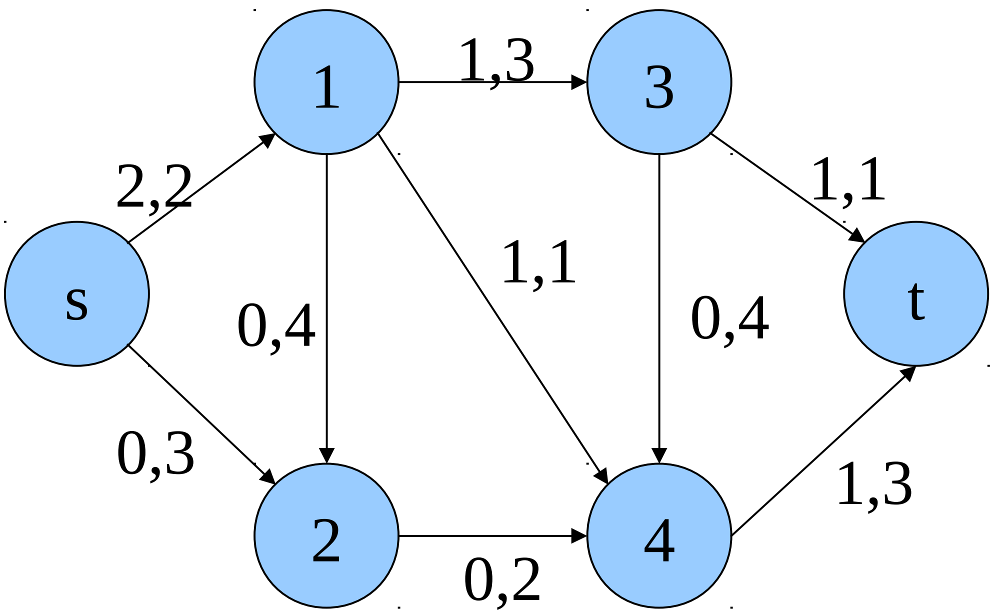
Algoritmo de Ford-Fulkerson

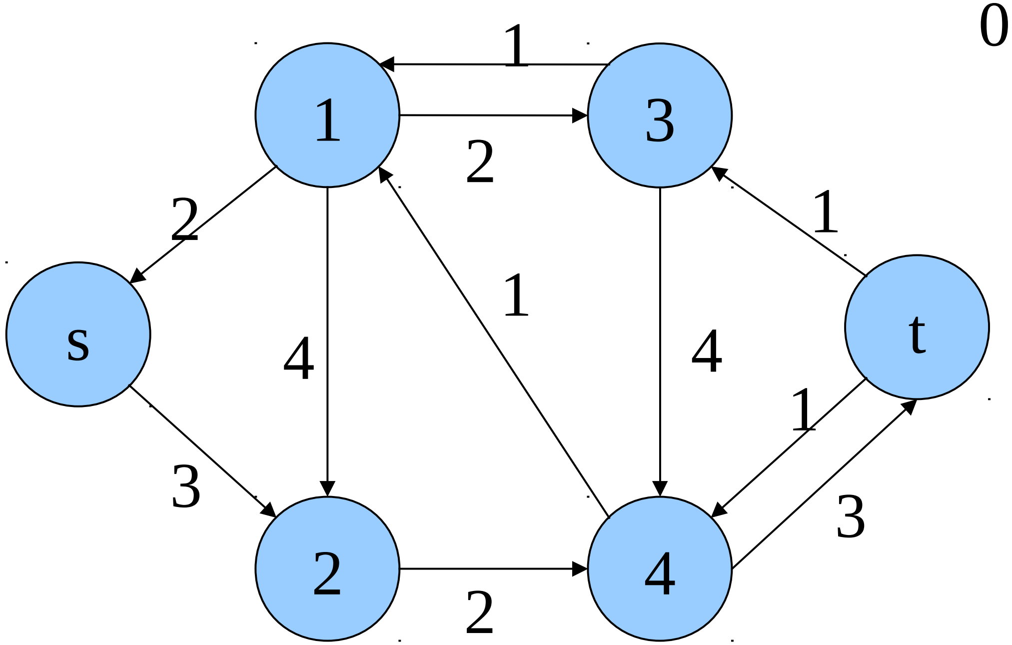




L = {~~s~~, ~~1~~, ~~2~~, ~~3~~, ~~4~~, t}

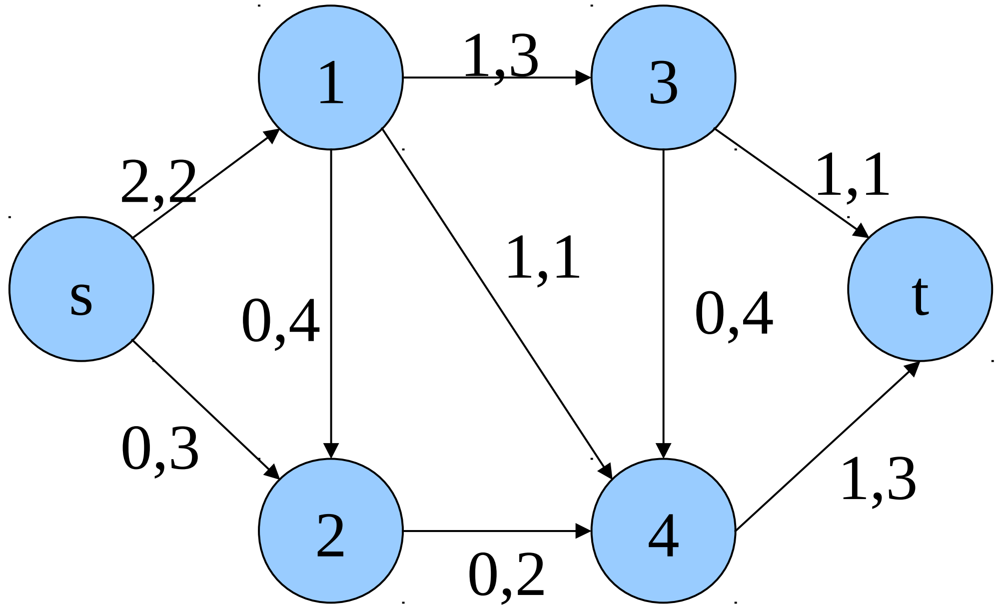
Algoritmo de Ford-Fulkerson

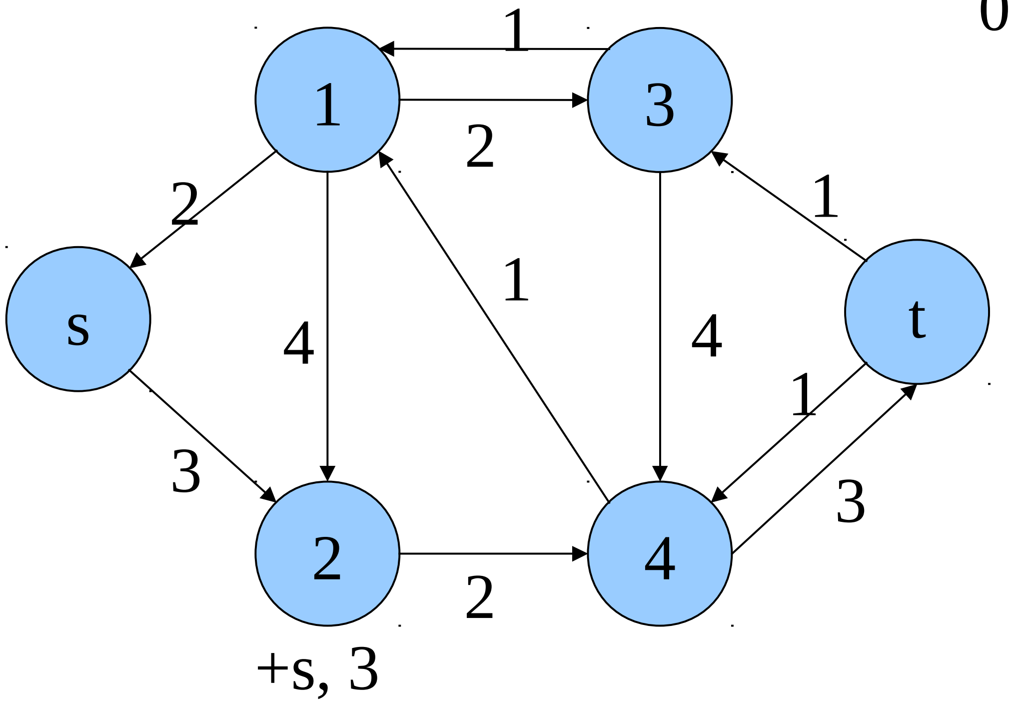




L = {s}

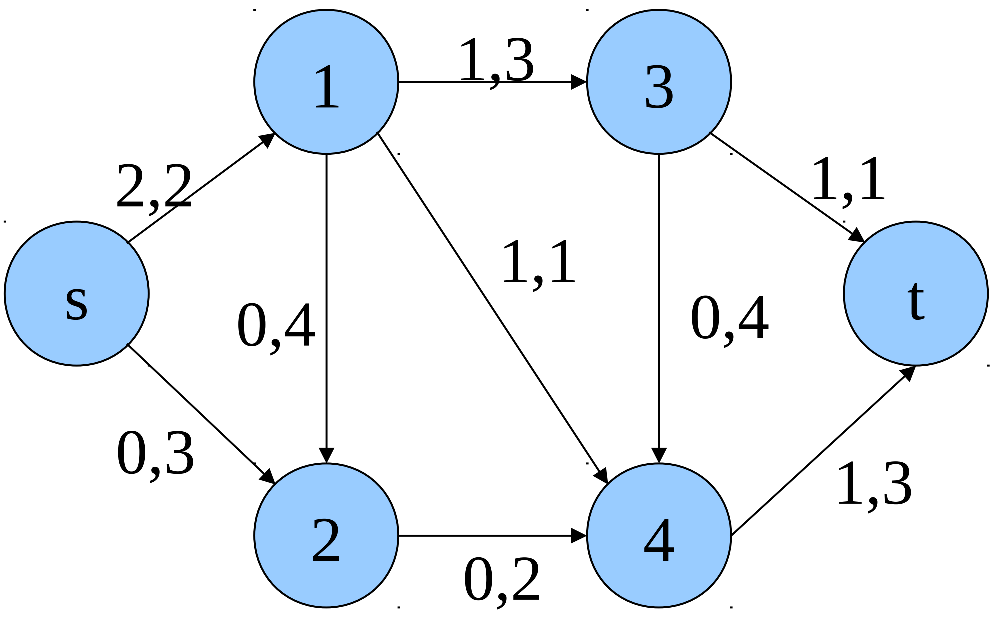
Algoritmo de Ford-Fulkerson

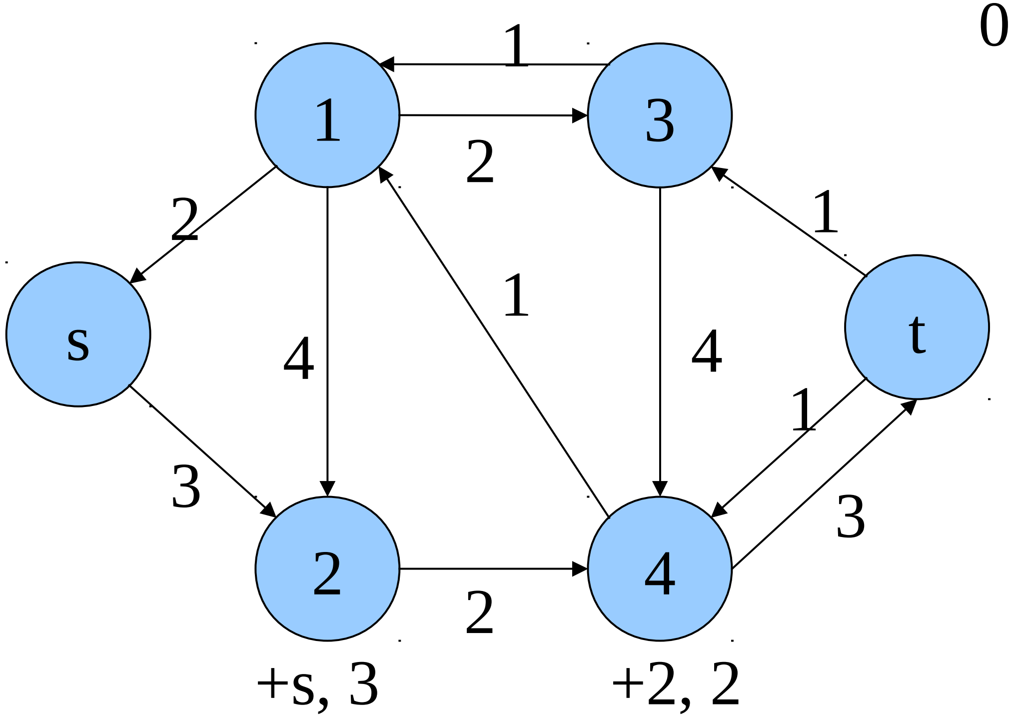




L = {~~s~~, 2}

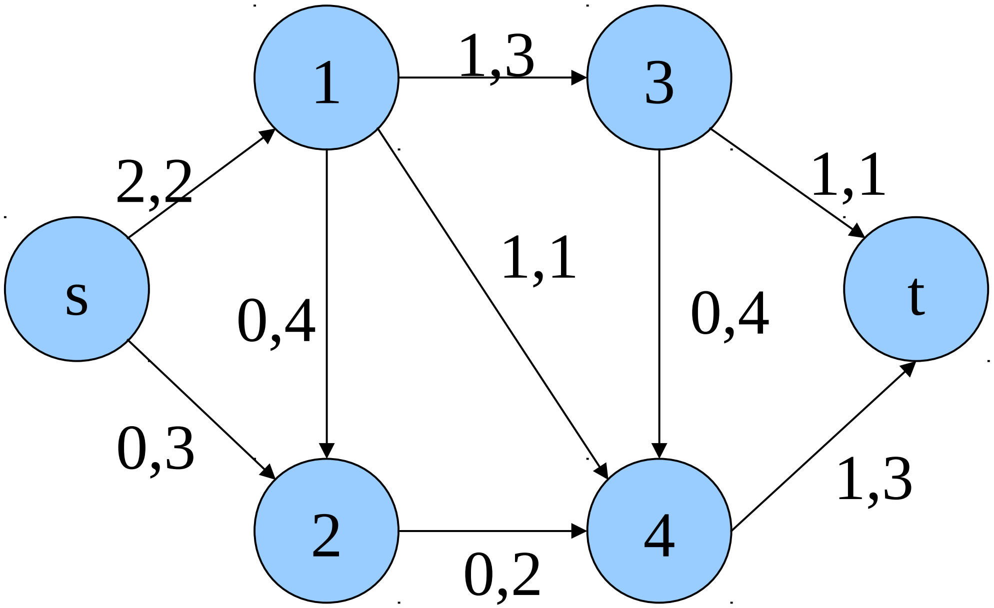
Algoritmo de Ford-Fulkerson

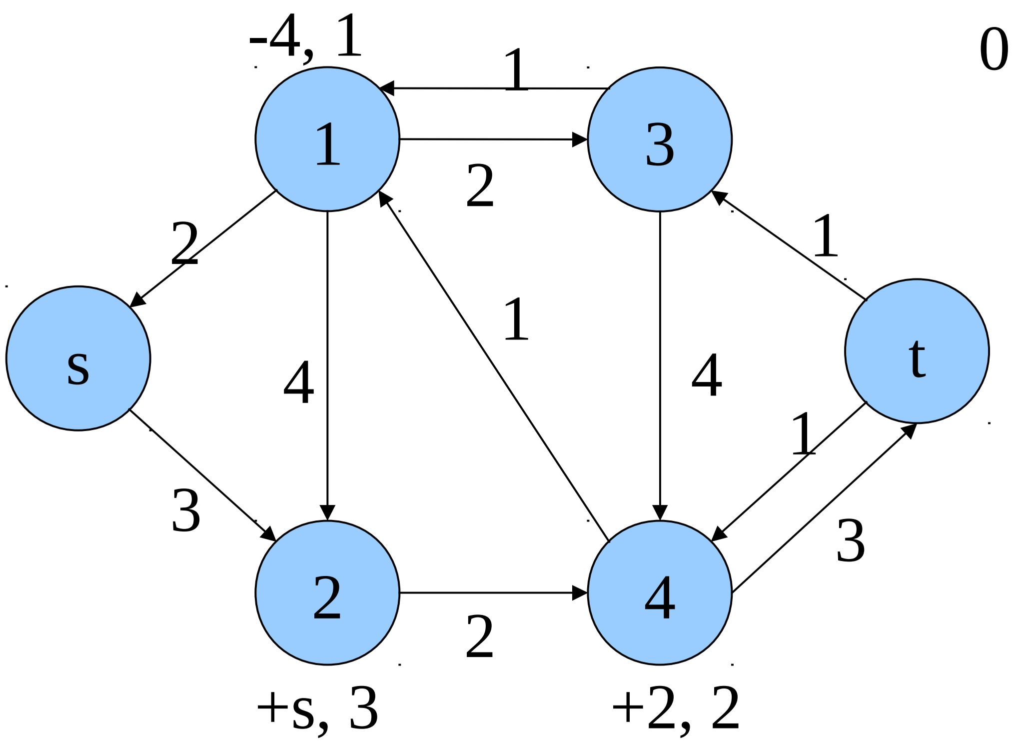




L = {~~s~~, ~~2~~, 4}

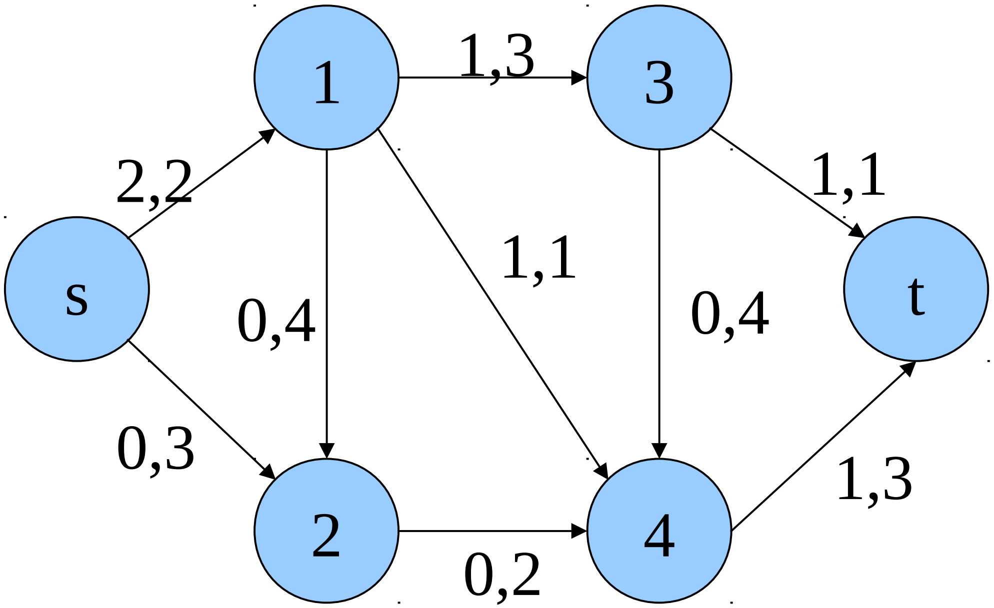
Algoritmo de Ford-Fulkerson

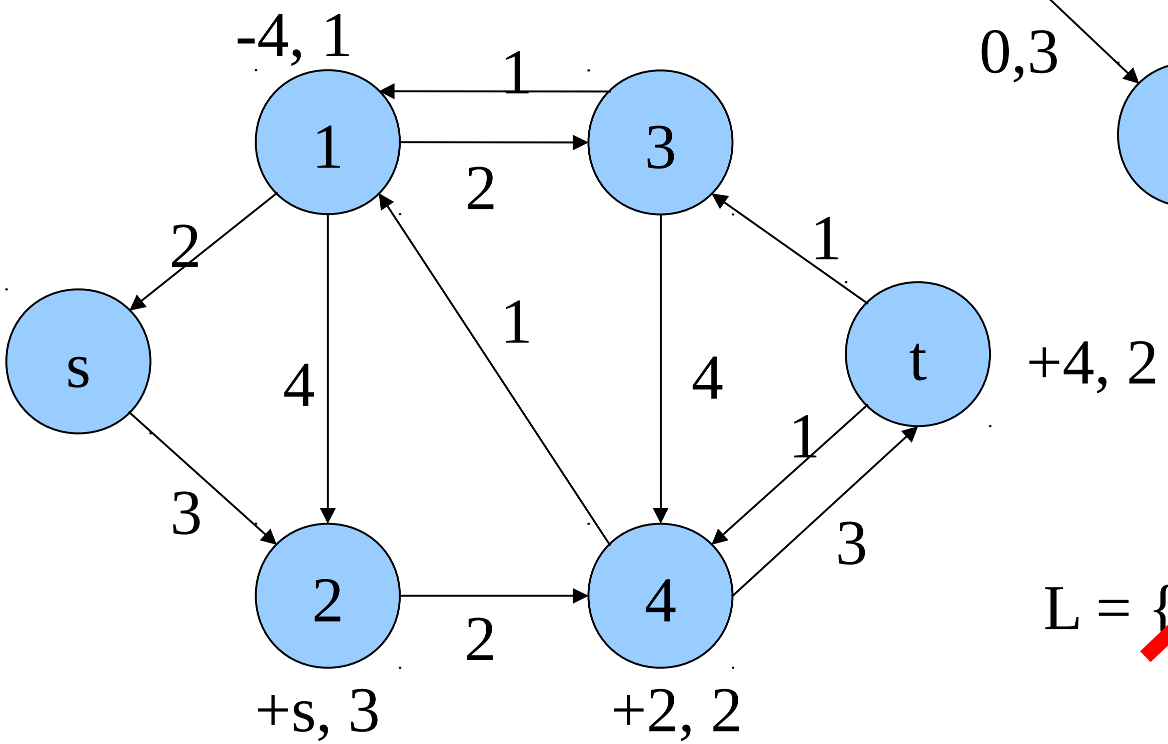




L = {~~s~~, ~~2~~, 4, 1}

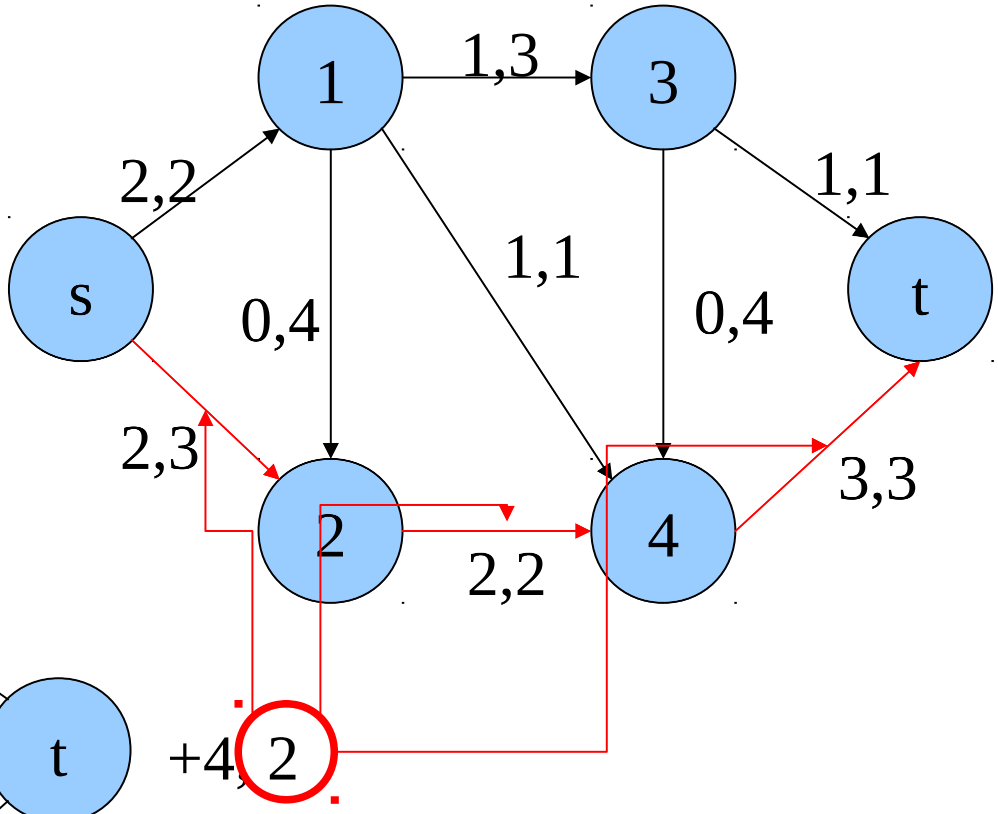
Algoritmo de Ford-Fulkerson

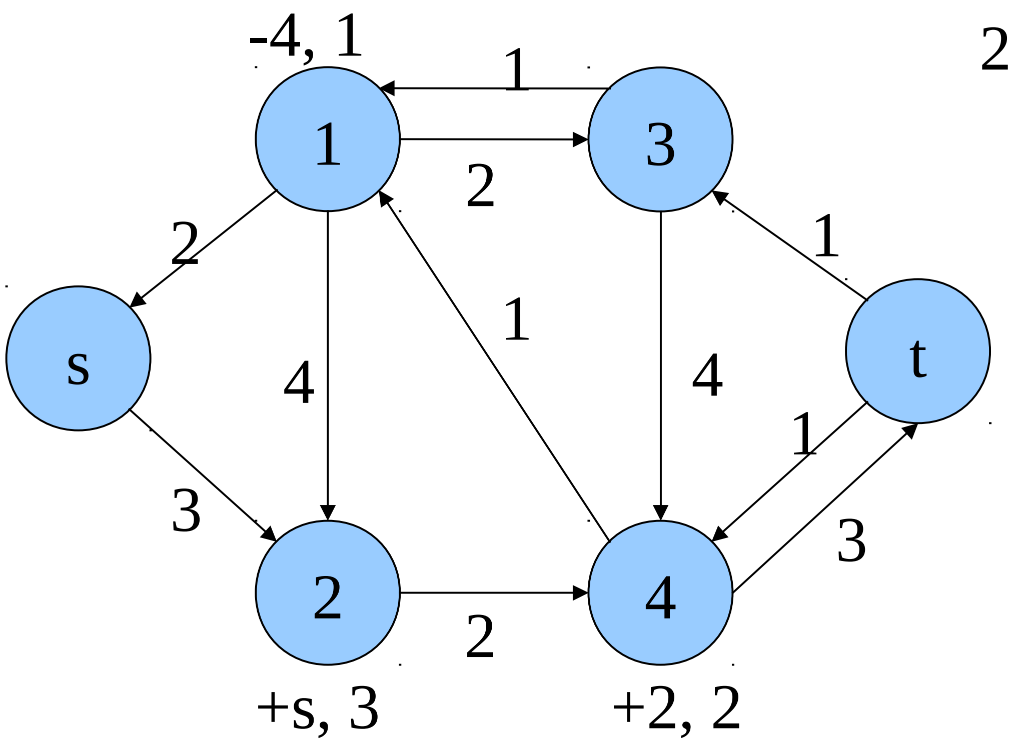




L = {~~s~~, ~~2~~, ~~4~~, 1, t}

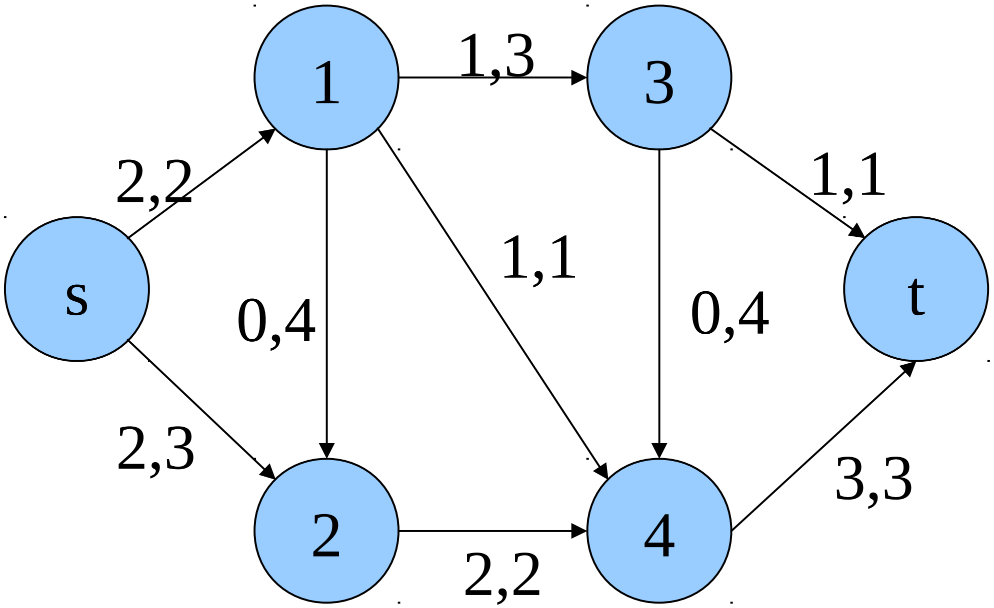
Algoritmo de Ford-Fulkerson

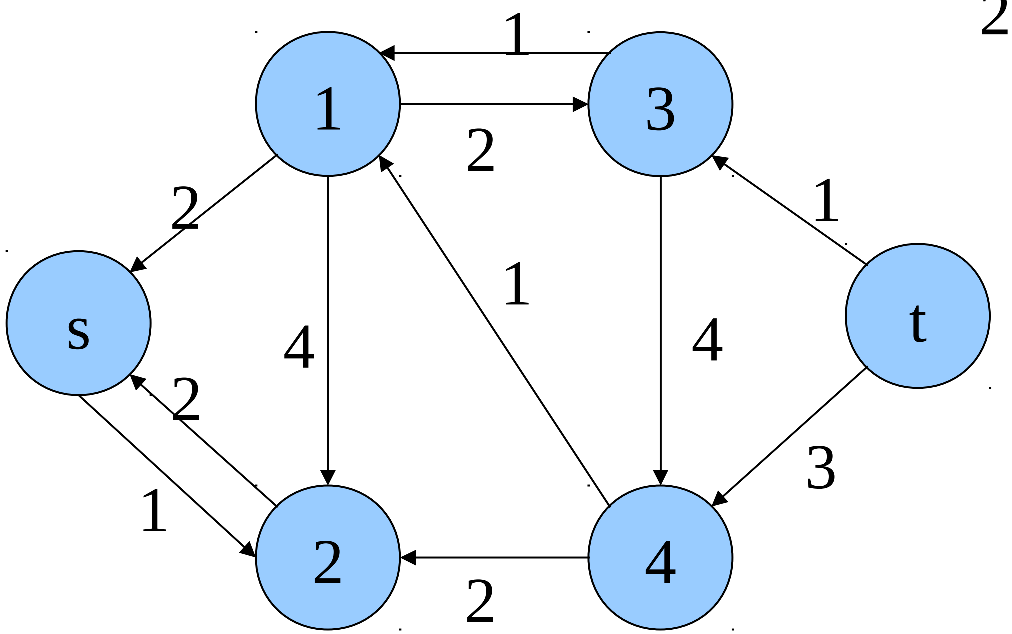




L = {~~s~~, ~~2~~, ~~4~~, 1, t}

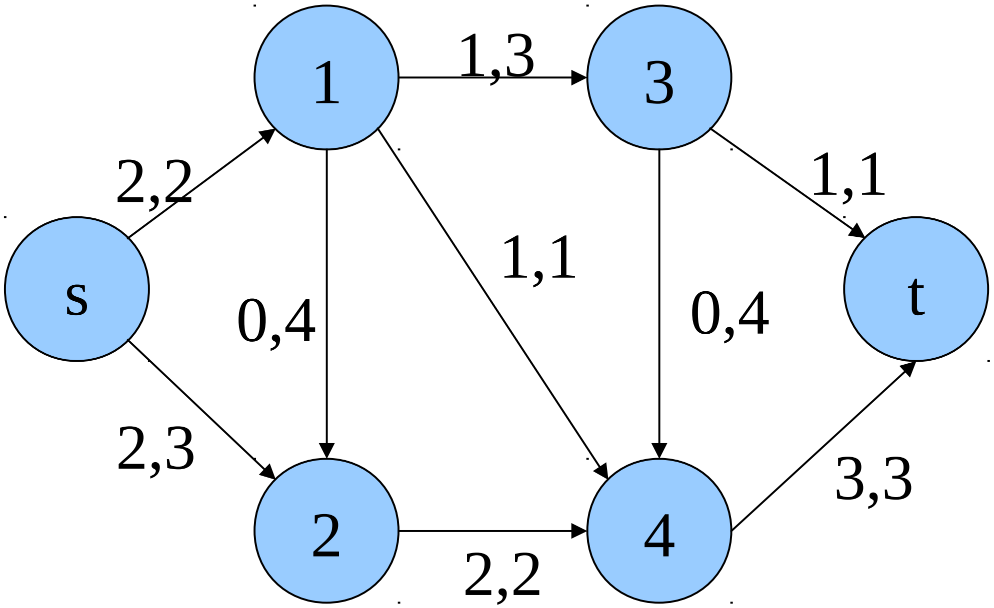
Algoritmo de Ford-Fulkerson

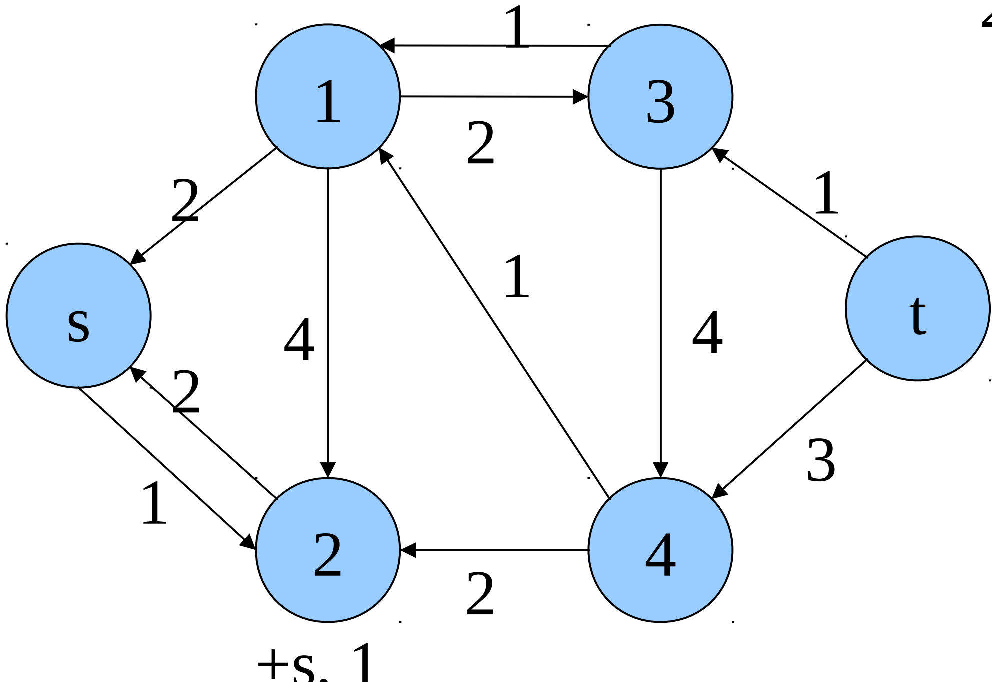




L = {s}

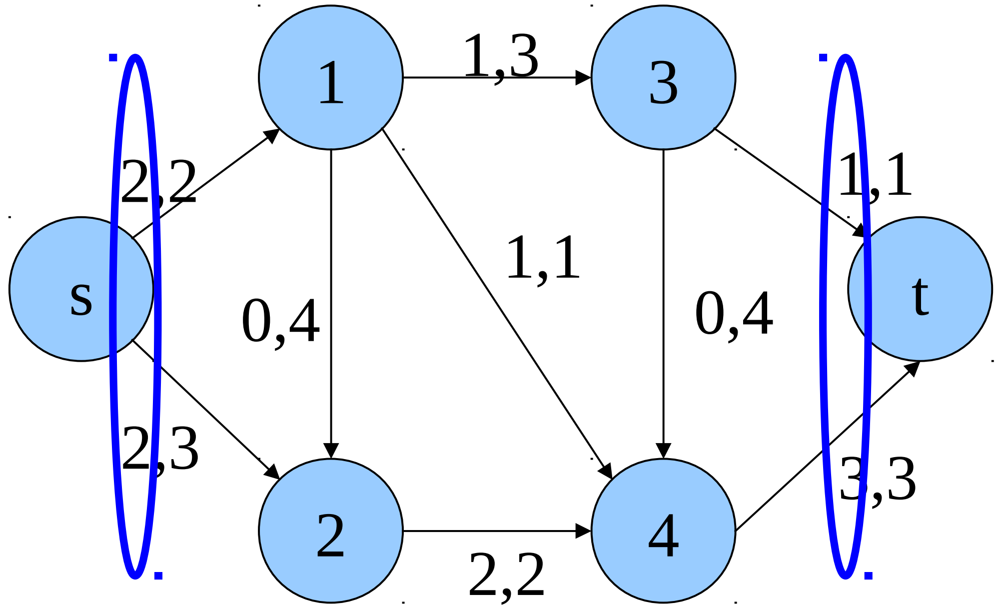
Algoritmo de Ford-Fulkerson

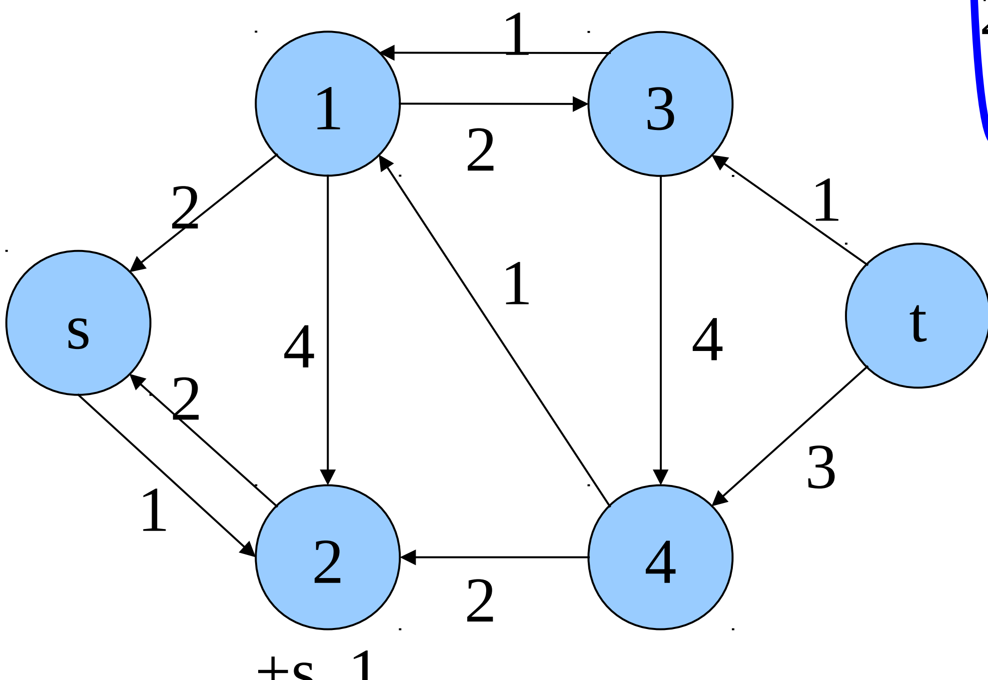




L = {~~s~~, ~~2~~}

Algoritmo de Ford-Fulkerson

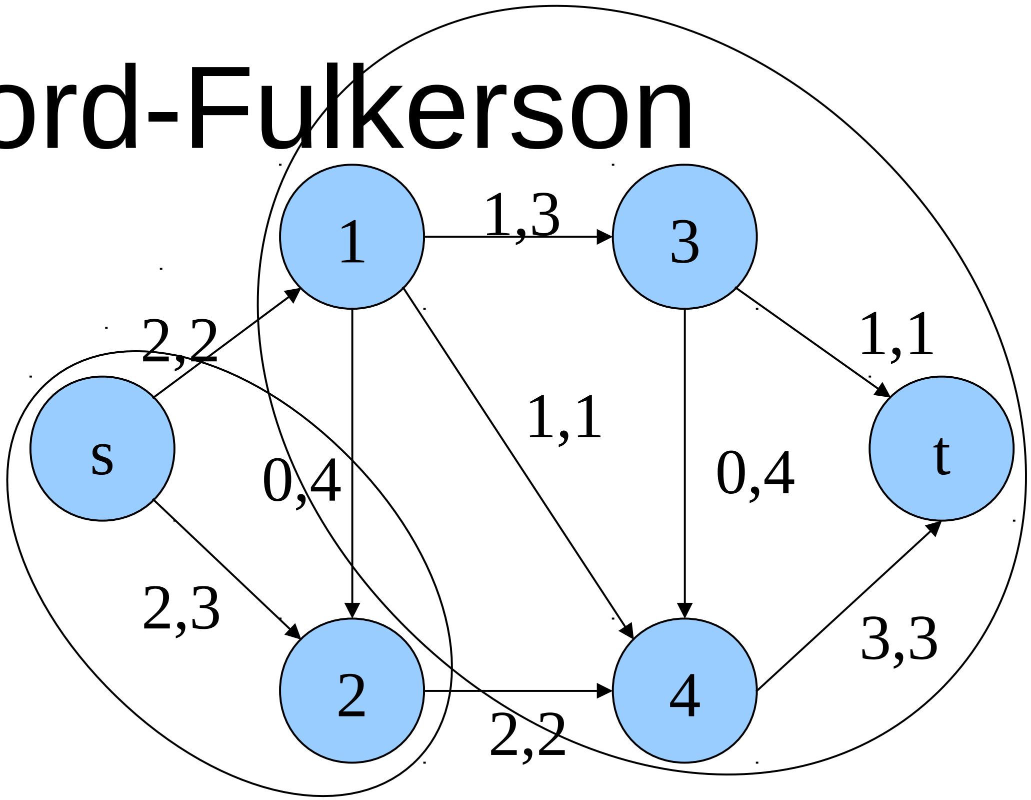


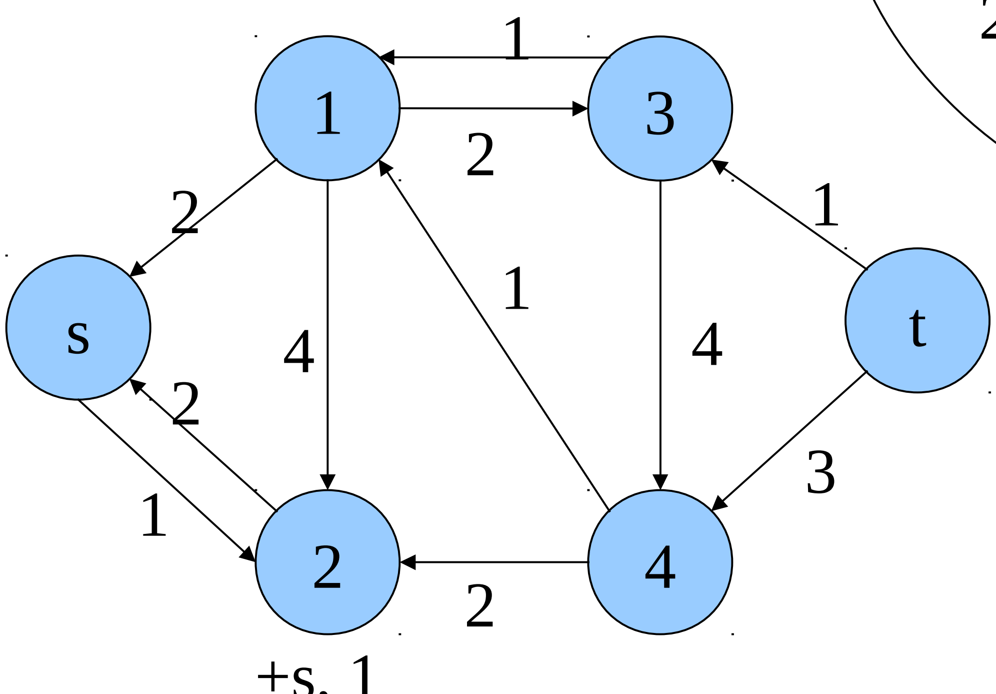


Fluxo máximo = 4

L = {~~s~~, ~~2~~}

Algoritmo de Ford-Fulkerson





Corte mínimo

L = {~~s~~, ~~2~~}

DMKM

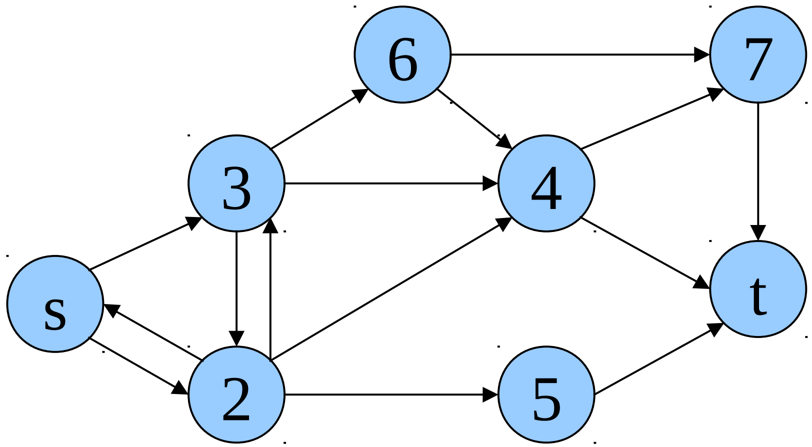
- Desenvolvido por Malhottra, Kumar e Maheshwari
  - Posteriormente melhorada por Dinic
- Considerado um dos algoritmos mais eficientes para resolver o problema de fluxo máximo em redes
- Idéia é iniciar colocando uma certa quantidade de fluxos da origem s até o destino t
  - Arcos não saturados são analisados, alocando-se mais fluxos para estes
    - Até que nenhum fluxo adicional possa ser alocado nos arcos não saturados
  - Procedimento deve ser cuidadoso e sistemático, para não gerar fluxos que não representam o fluxo máximo real

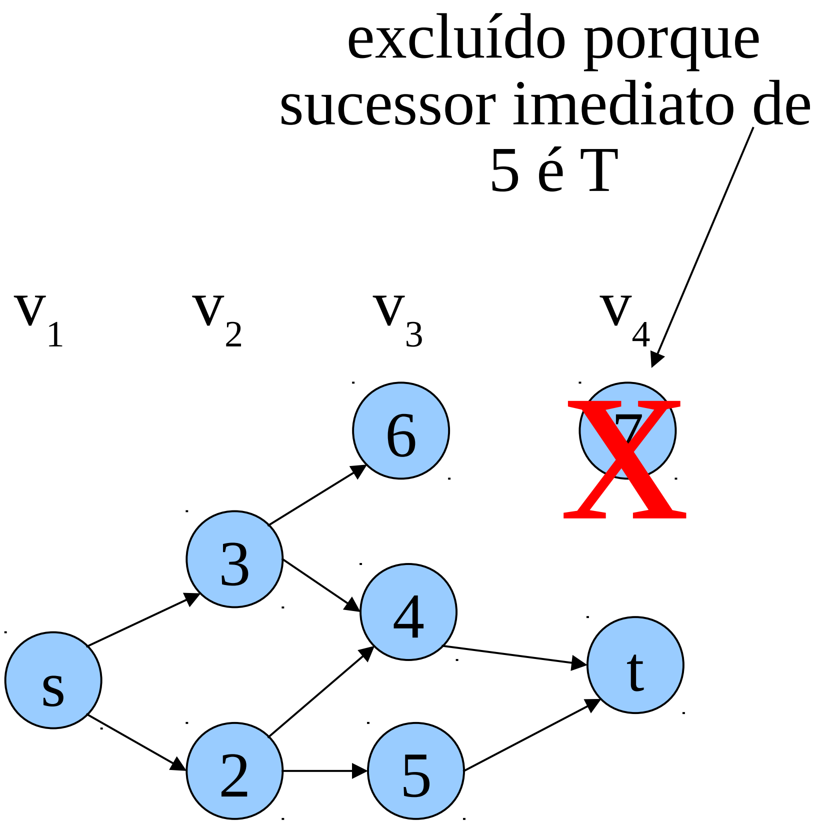
Rede Particionada (G_2)

- Trata-se de uma rede acíclica na qual todos os nós de N são particionados em níveis V_1, V_2, ..., V_k
  - Por definição, primeiro nível contém só a origem s do grafo
  - Nível V_2 consiste de todos os sucessores imediatos dos nós em V_1
  - O nível V_l, $2 \leq l \leq k-1$, consiste de todos os nós sucessores imediatos do nível V_{l-1} e que não estejam contidos nos níveis inferiores
  - Nível V_k consiste somente do nó destino t
  - Alguns nós do grafo original podem ser excluídos na transformação deste em um grafo particionado

Rede Particionada (G_2)

- Exemplo: Determine G_2 de G





Definições

- Definição: O arco (i,j) é dito arco saturado se $x_{(i,j)} = u_{(i,j)}$
- Definição: Um caminho P de s a t é dito saturado se existe em P pelo menos um arco saturado
- Definição: Um fluxo fs de um grafo G é dito fluxo de saturação se todo caminho de s até t for um caminho saturado. Obviamente todo fluxo máximo deve ser um fluxo saturado, mas a recíproca nem sempre é verdadeira, isto é, nem sempre um fluxo de saturação é máximo

Definições

- Definição: Definimos como potencial de um nó V o fluxo adicional máximo que pode ser passado por V. Essa quantidade potencial de V (pot(V)) é o mínimo entre dois valores: a) a quantidade de fluxo que ainda pode chegar em V (input(V)) e b) a quantidade de fluxo que ainda pode sair de V (output(V)). Por definição, dizemos que pot(s) = output(s) e que pot(t) = input(t)
- Definição: O nó r de N_2 de G_2 que possui o menor potencial é dito nó de referência

Etapas do Algoritmos DMKM

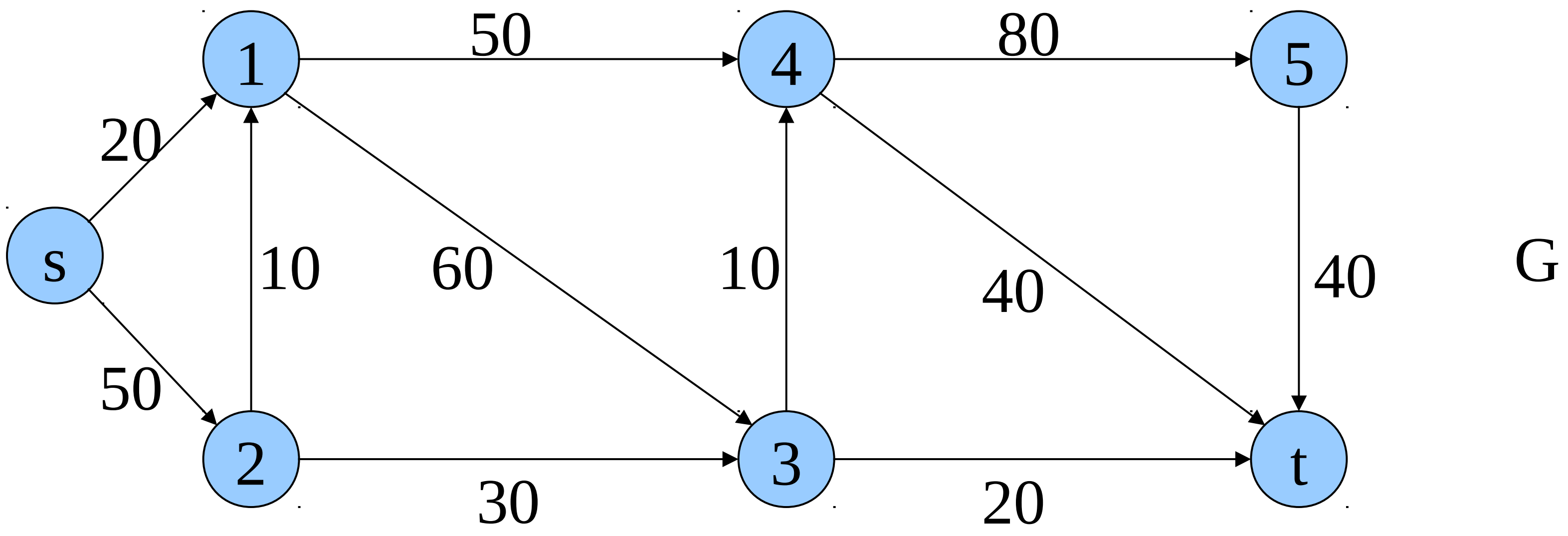
- Etapa 1:
  - Objetivo é determinar o fluxo de saturação de uma rede particionada G_2. Em G_2, determinamos o pot(r), onde r é um nó de referência e acrescentamos a quantidade de fluxo pot(r) nos caminhos de s até t via r no grafo original G

Etapas do Algoritmos DMKM

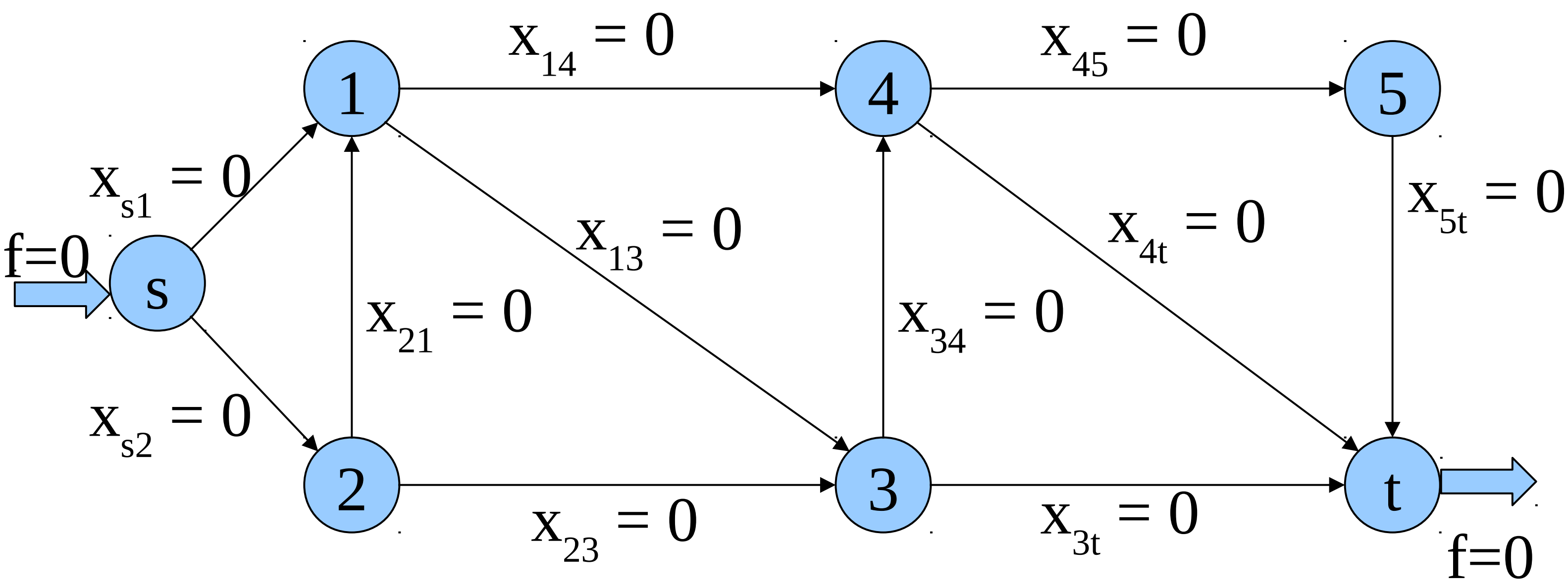
- Etapa 2:
  - Objetivo é gerar o fluxo máximo f da rede original. Feito por estágios
    - Inicialmente partimos com uma solução inicial s_1 idêntica a de f em G
    - A partir dessa solução inicial geramos a rede particionada G_2^1. Determinamos o fluxo de saturação fs_1 de G_2^1 e a quantidade pot(r) é acrescida nos caminhos de s a t via r em S_1, gerando uma solução S_2 em G com fluxo parcial f = fs_1 e G'
    - A partir de G', geramos uma nova rede G_2, aplicamos a fase 1 em G_2^2, e assim sucessivamente até que dada uma solução S_k em G com fluxo $f = \sum_{j=1}^{k} fs_j$, não seja mais possível construir uma G_2^{k+1}. O fluxo atual f de S_k seria então o fluxo máximo de s a t em G

DMKM

- Determinar o fluxo máximo da rede abaixo

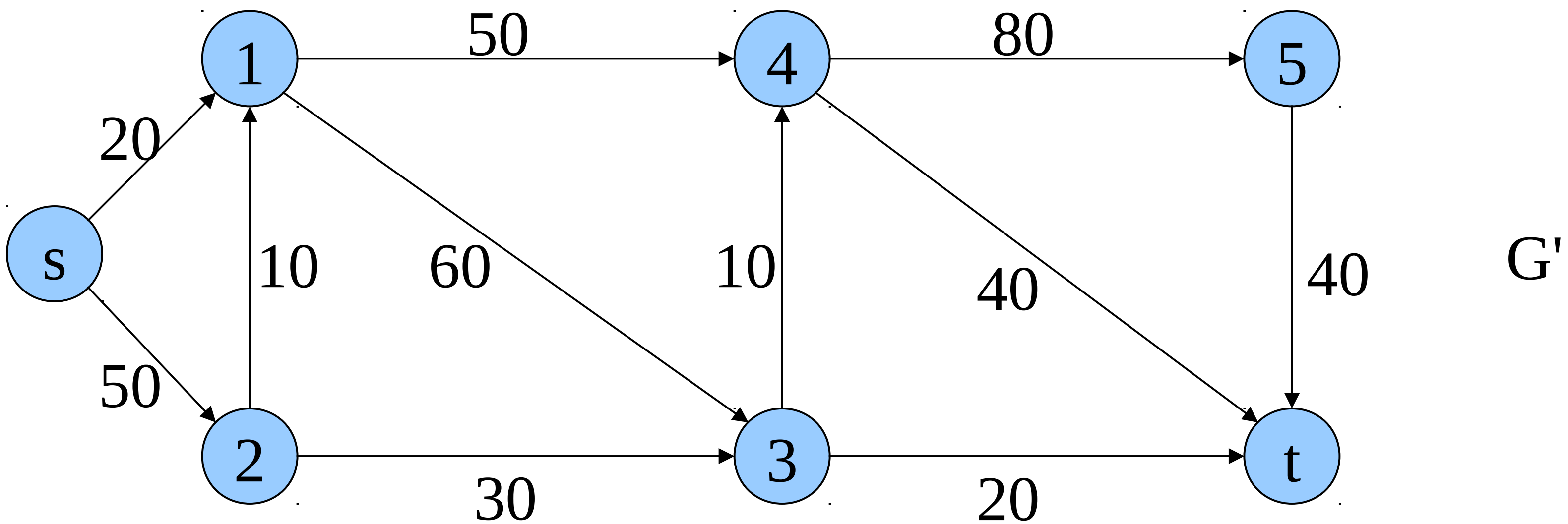


DMKM

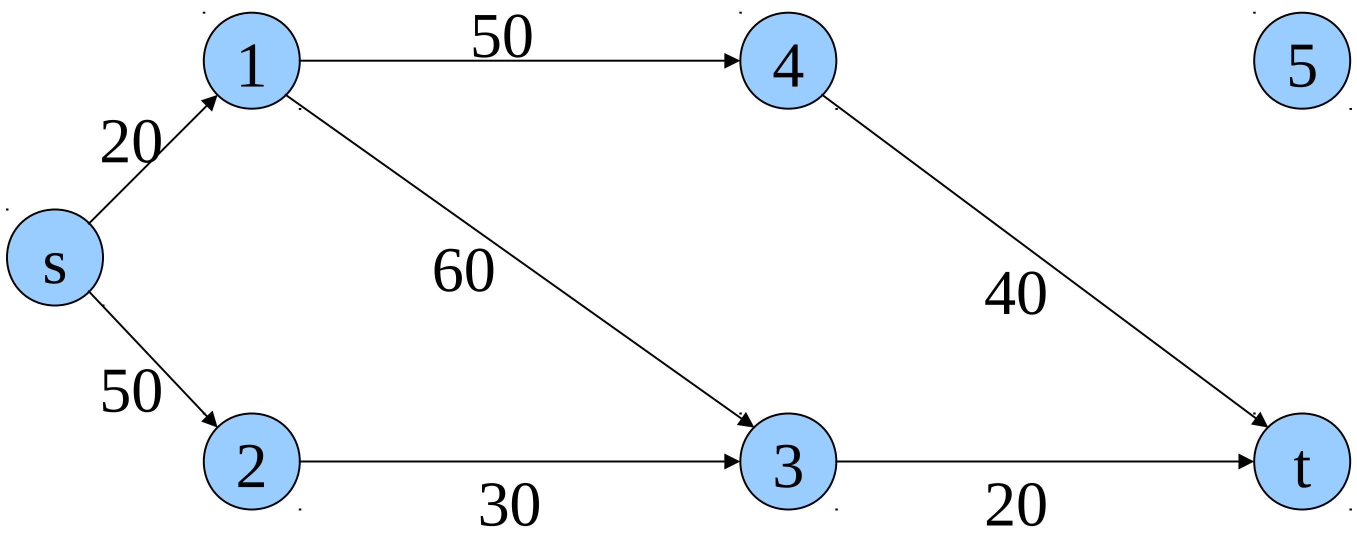


Solução S_0

DMKM



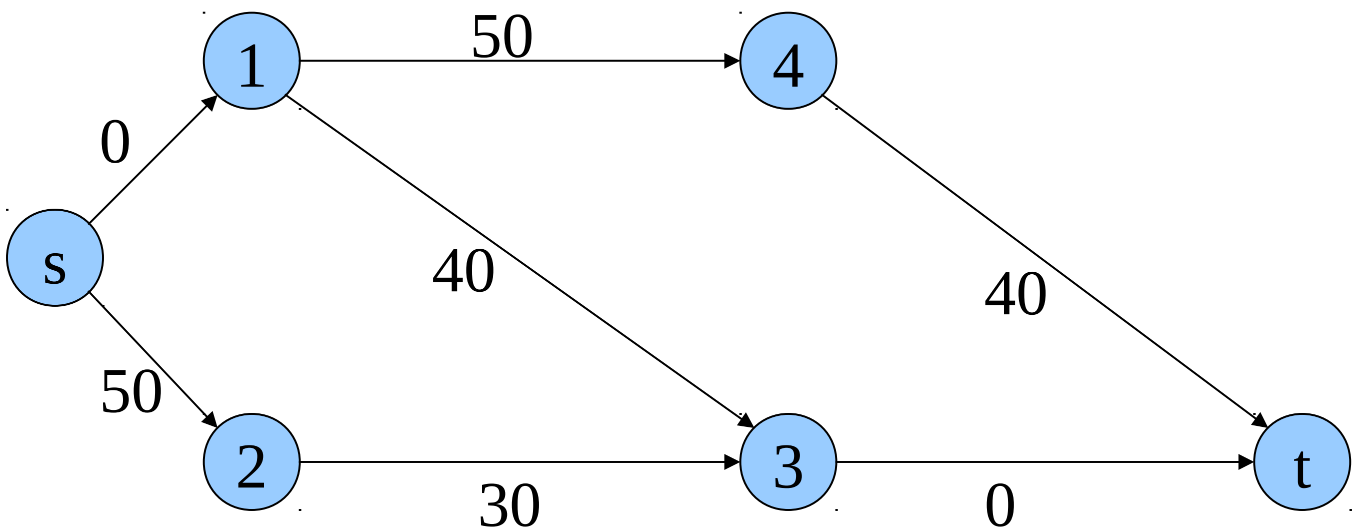
DMKM



G_2

Rede particionada
r = 1
f = 20

DMKM



G_2^1

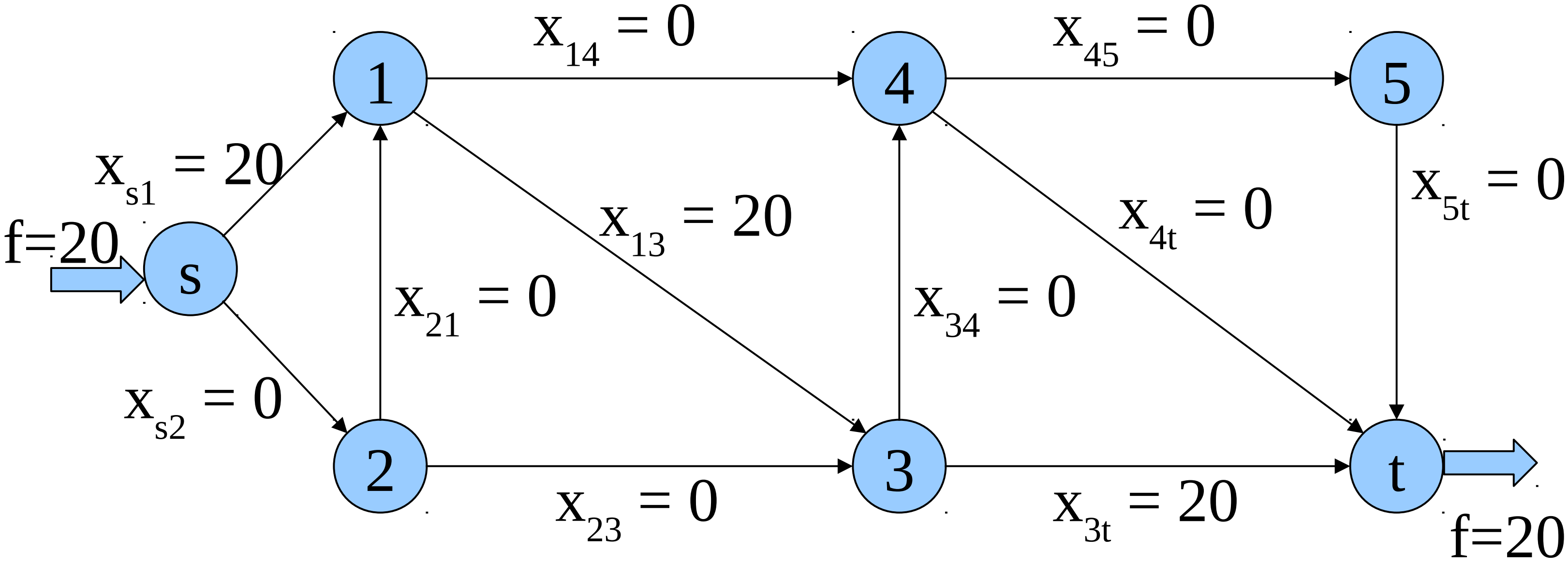
DMKM

- Apagar por saturação
  - Arcos s1 e 3t: ambos valem zero
  - Nó 1 e arcos 14 e 13: ninguém passa a chegar em 1
  - Nó 4 e arco 4t: ninguém passa a chegar em 4
  - Nó 3: ninguém sai de 3
  - Nó 2: ninguém sai de 2
  - Nós s e t: ninguém chega ou sai de t

DMKM

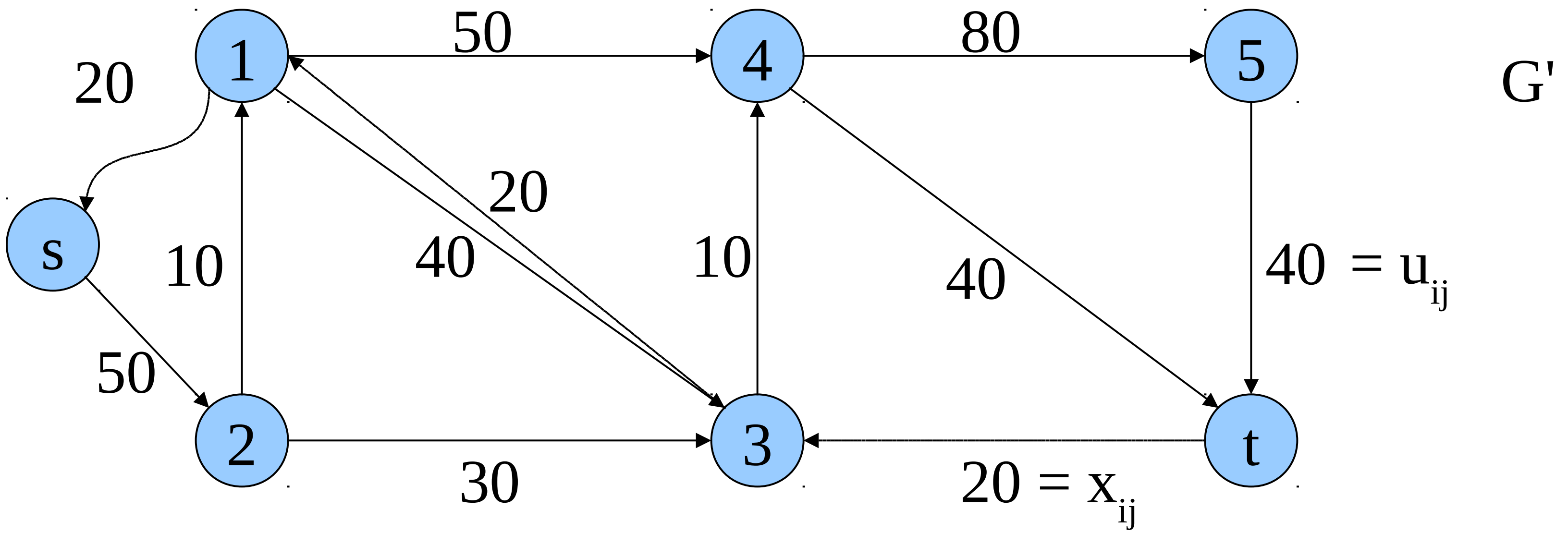
- Em G_2 colocamos um arco (i,j) se:
  - na rede G existe (i,j) com $x_{ij} < u_{ij}$ ou
  - se existe arco (j,i) em G com $x_{ij} > 0$
- No primeiro caso adicionamos fluxo em (i,j), enquanto no segundo reduzimos o fluxo de (j,i), isto é, adicionamos o fluxo de (i,j)

DMKM

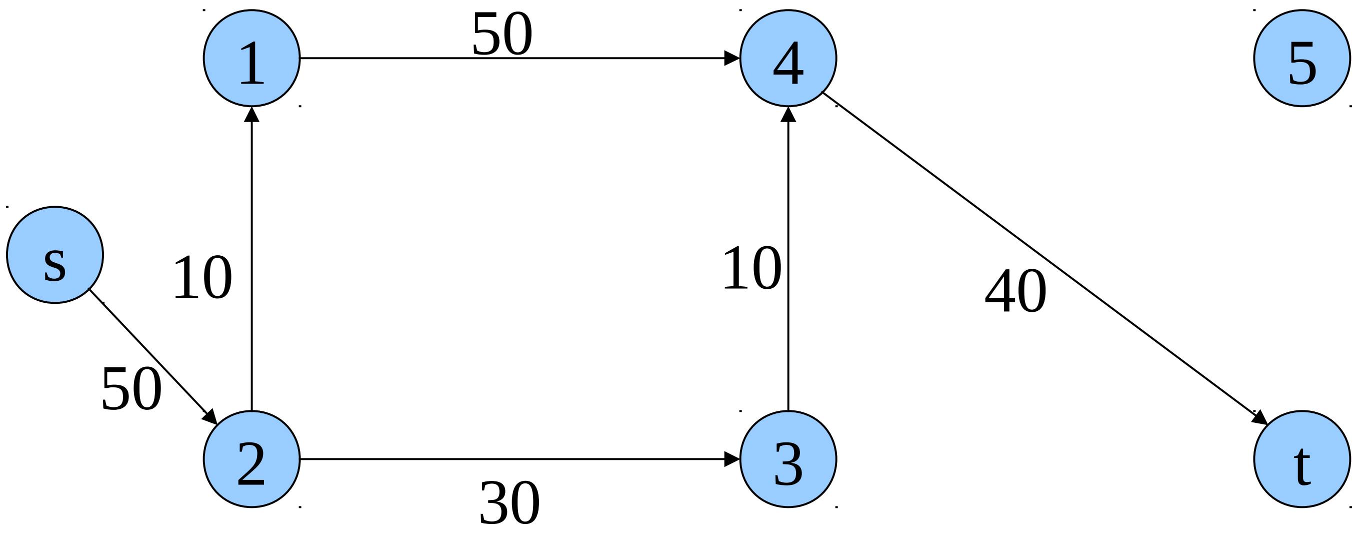


Solução S_1

DMKM



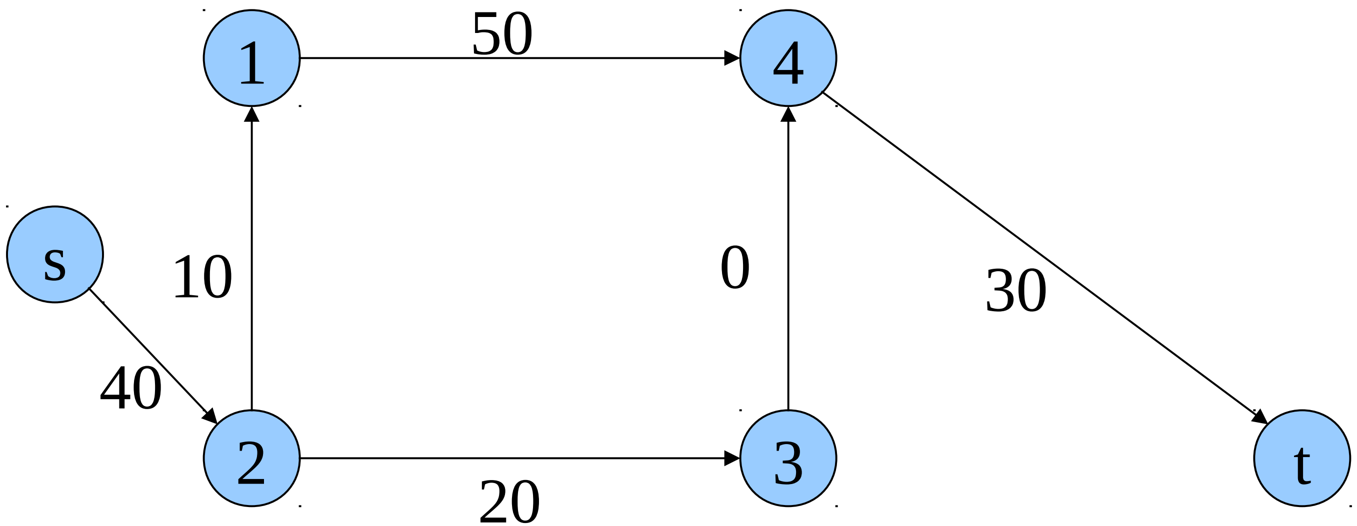
DMKM



G_2^2

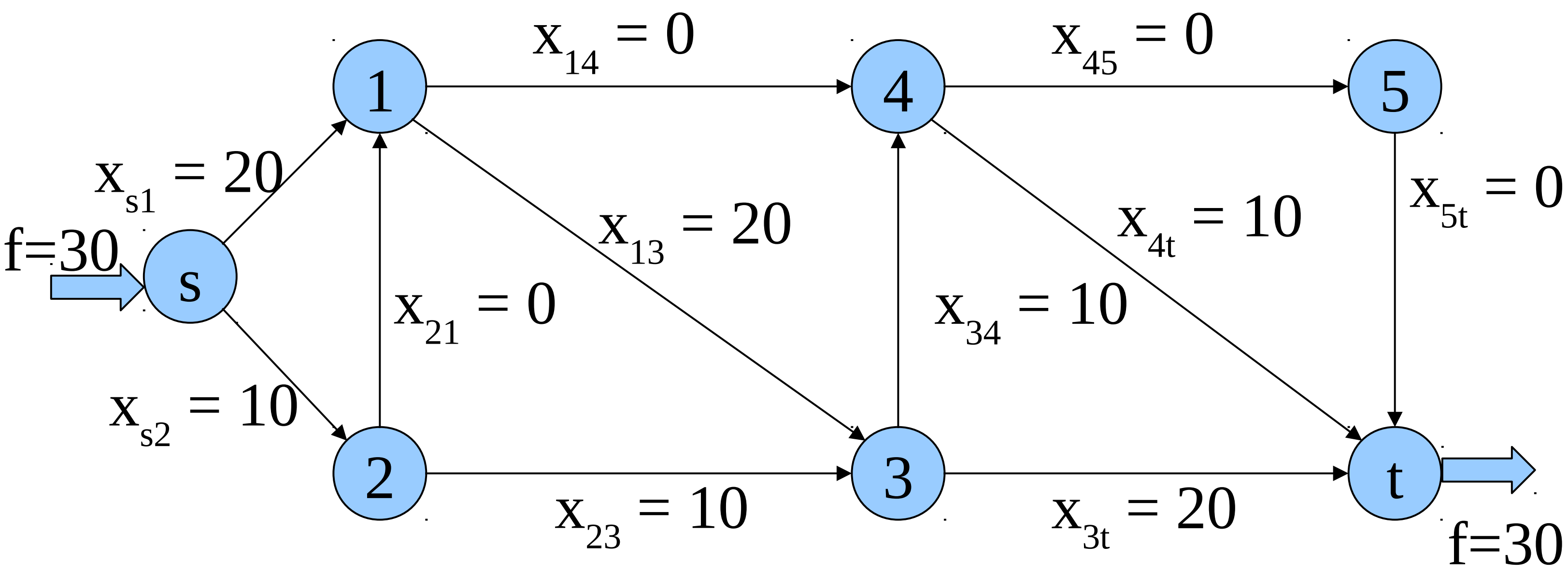
r = 4
f = 10

DMKM



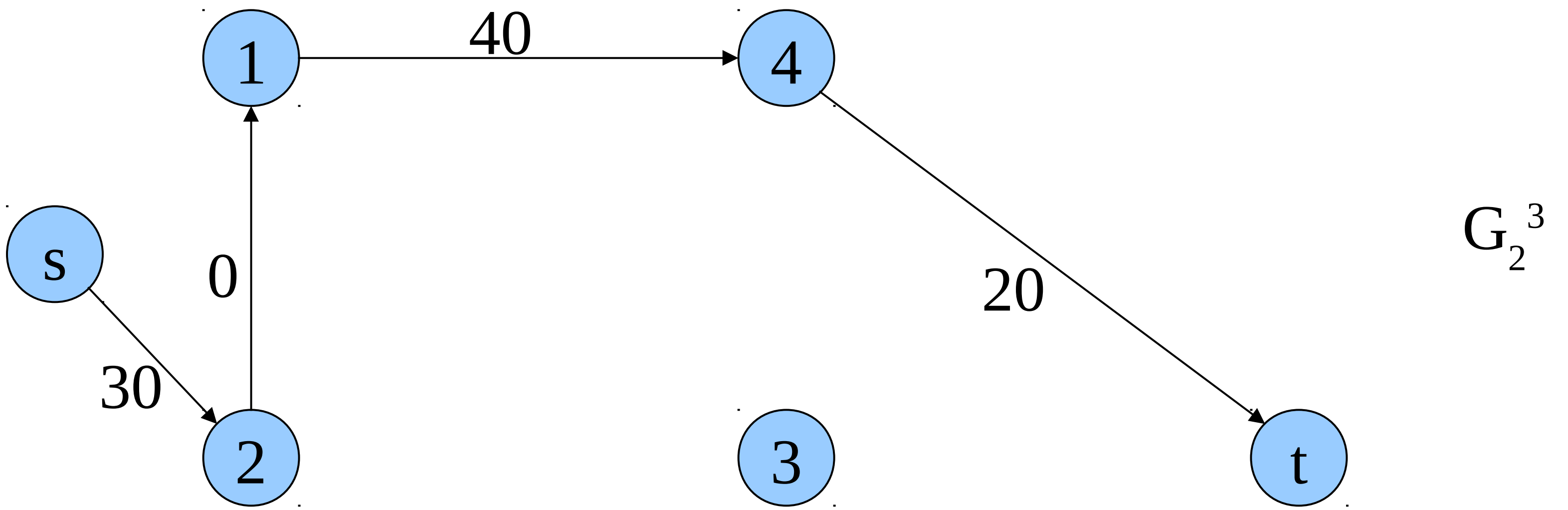
r = 1
f = 10

DMKM

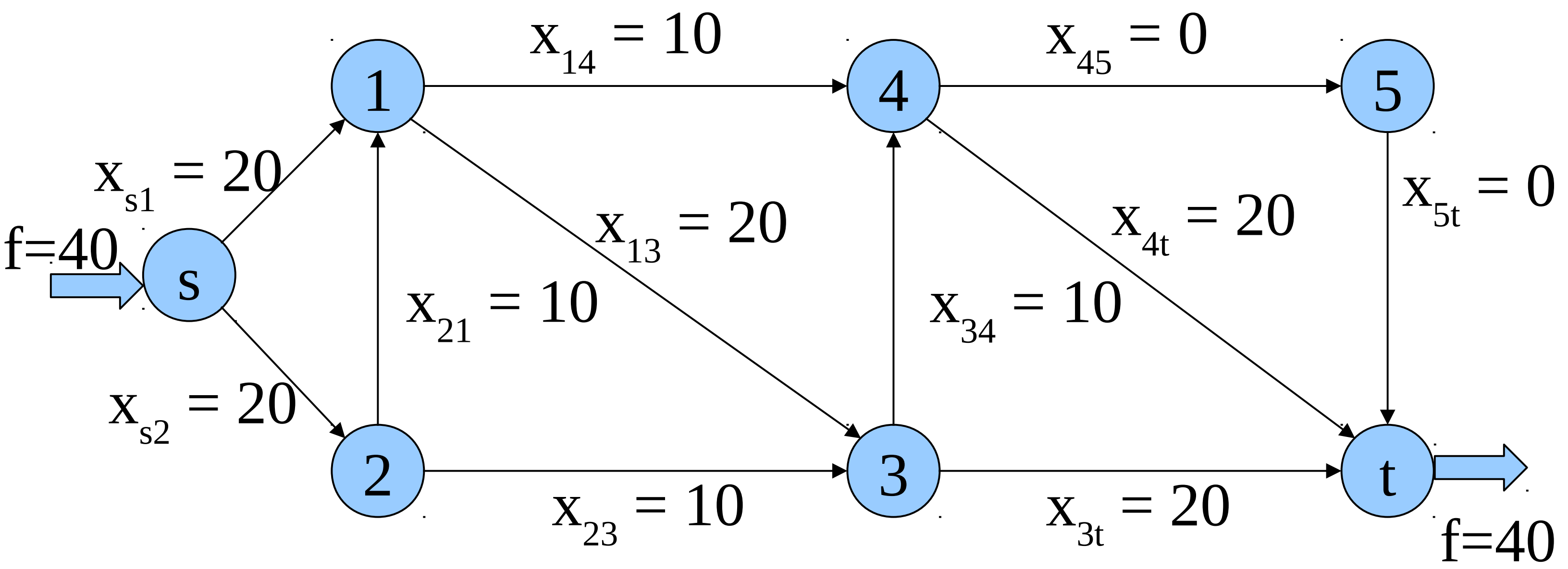


Solução S_2

DMKM

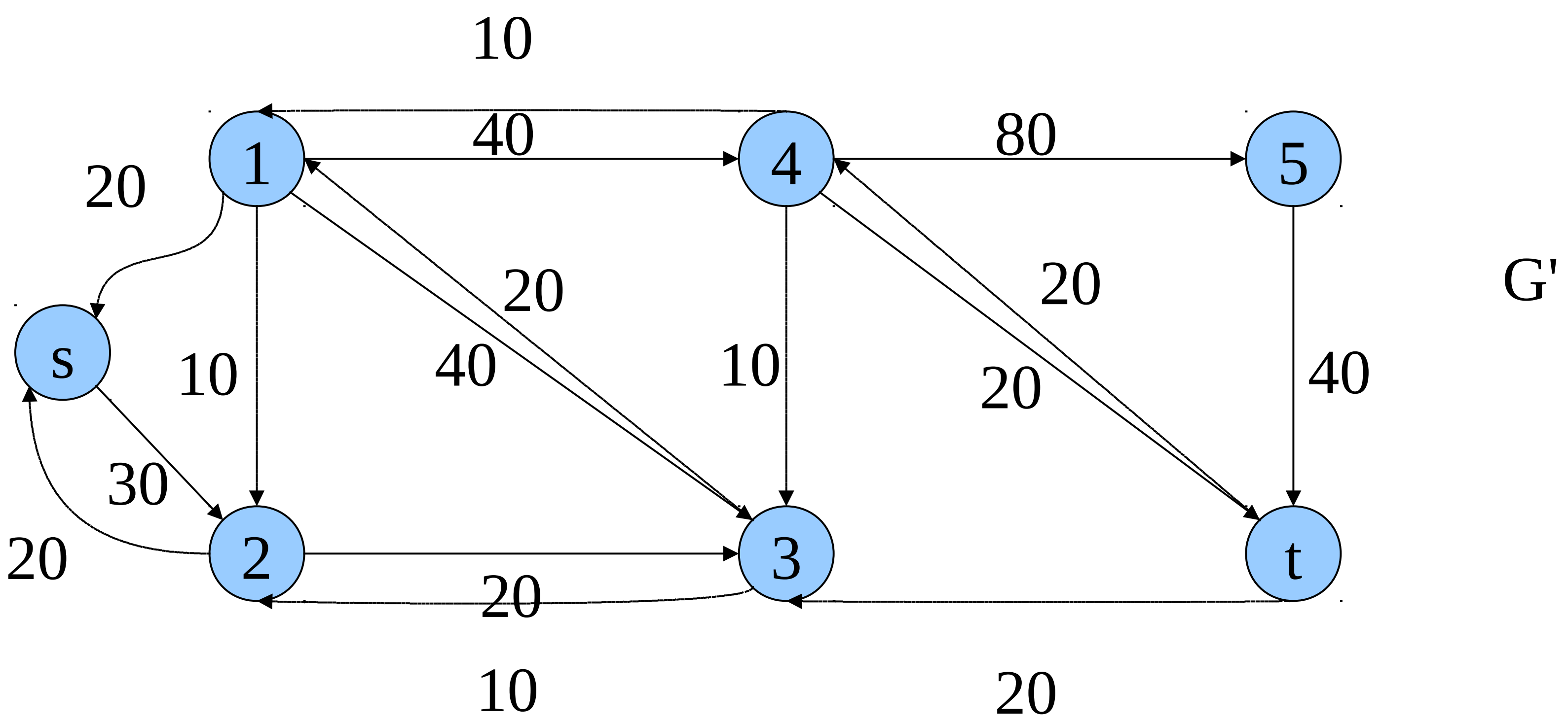


DMKM

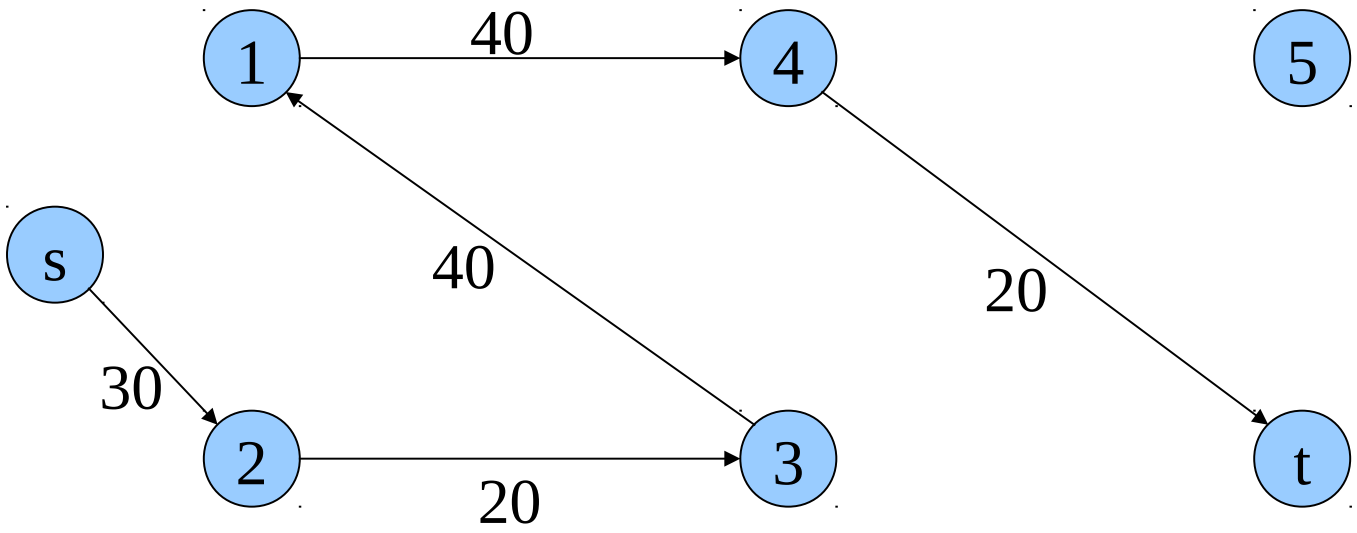


Solução S_2

DMKM



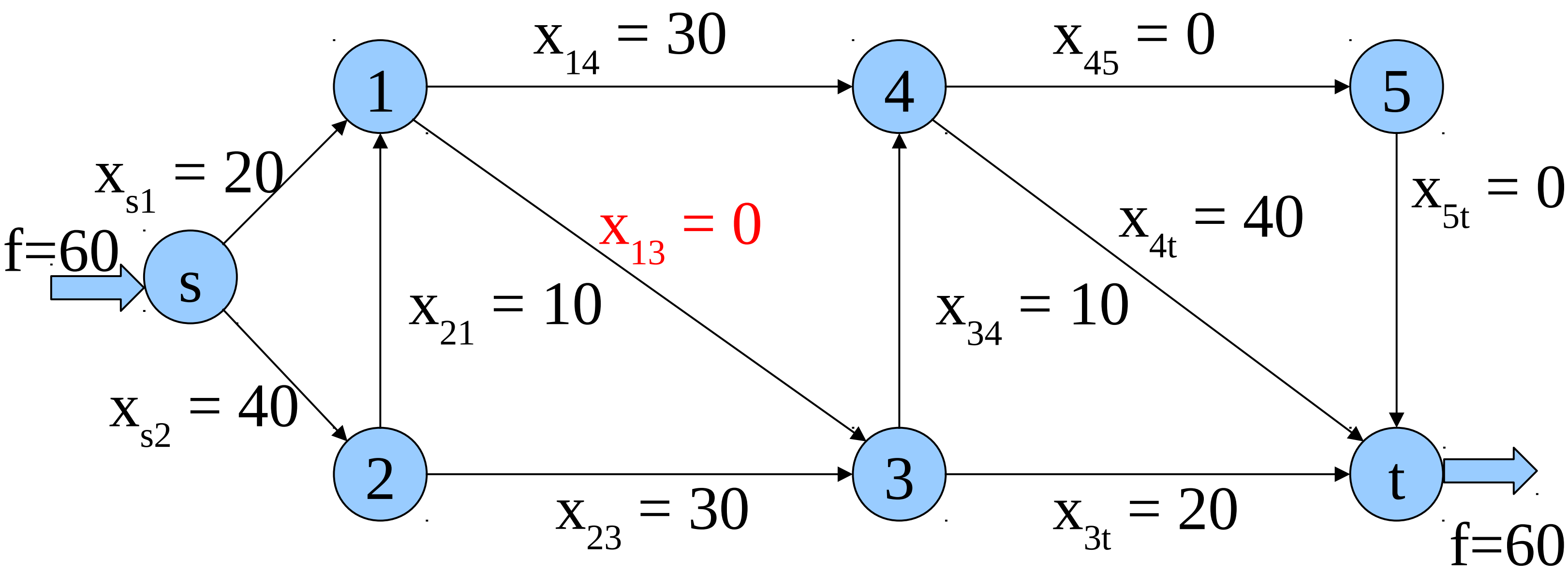
DMKM



G_2^3

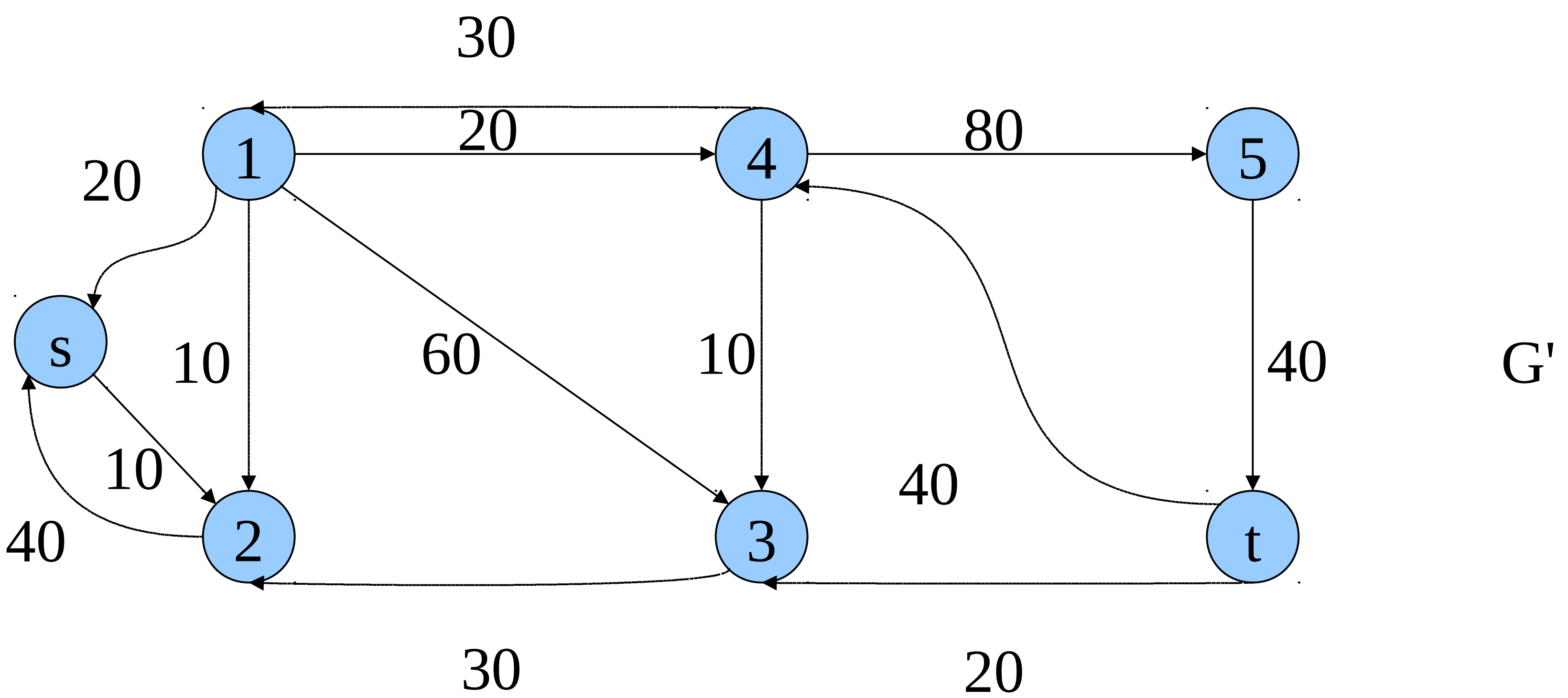
r = 3
f = 20

DMKM



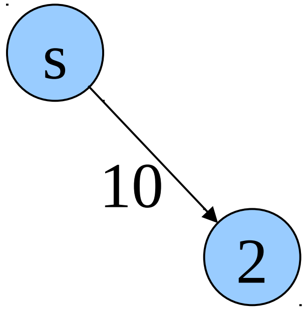
Solução S_3

DMKM



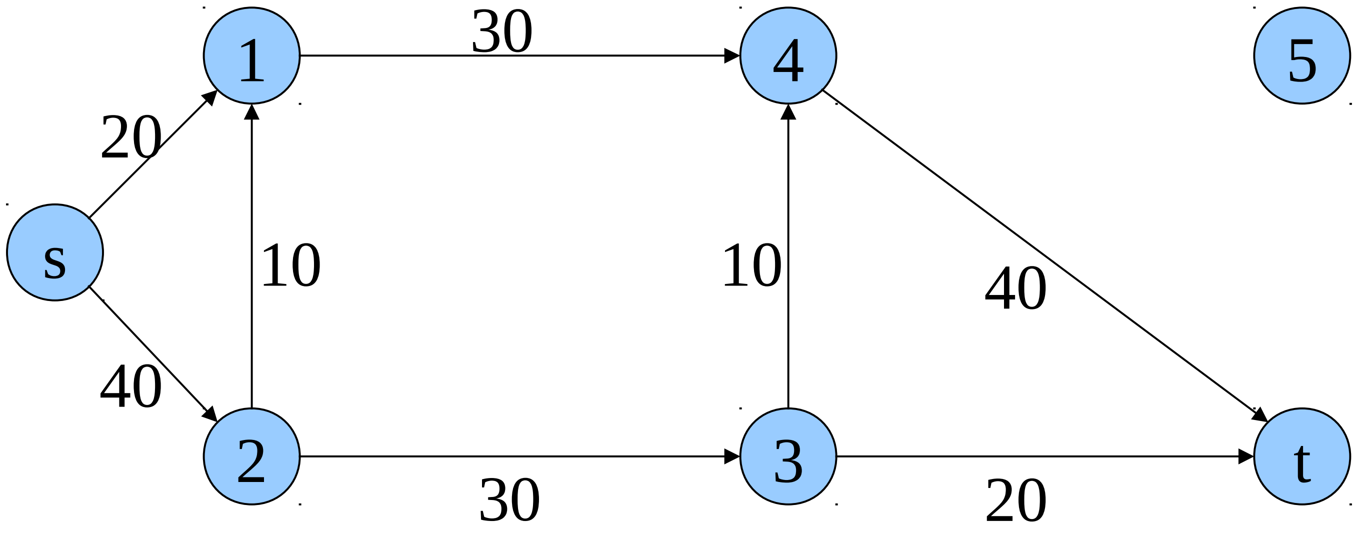
DMKM

G_2^4 ???



DMKM

- Não existe G_2^4 (não há como chegar a t)
  - Logo o fluxo máximo é 60



Próxima Aula...

- Algoritmos de ordenação