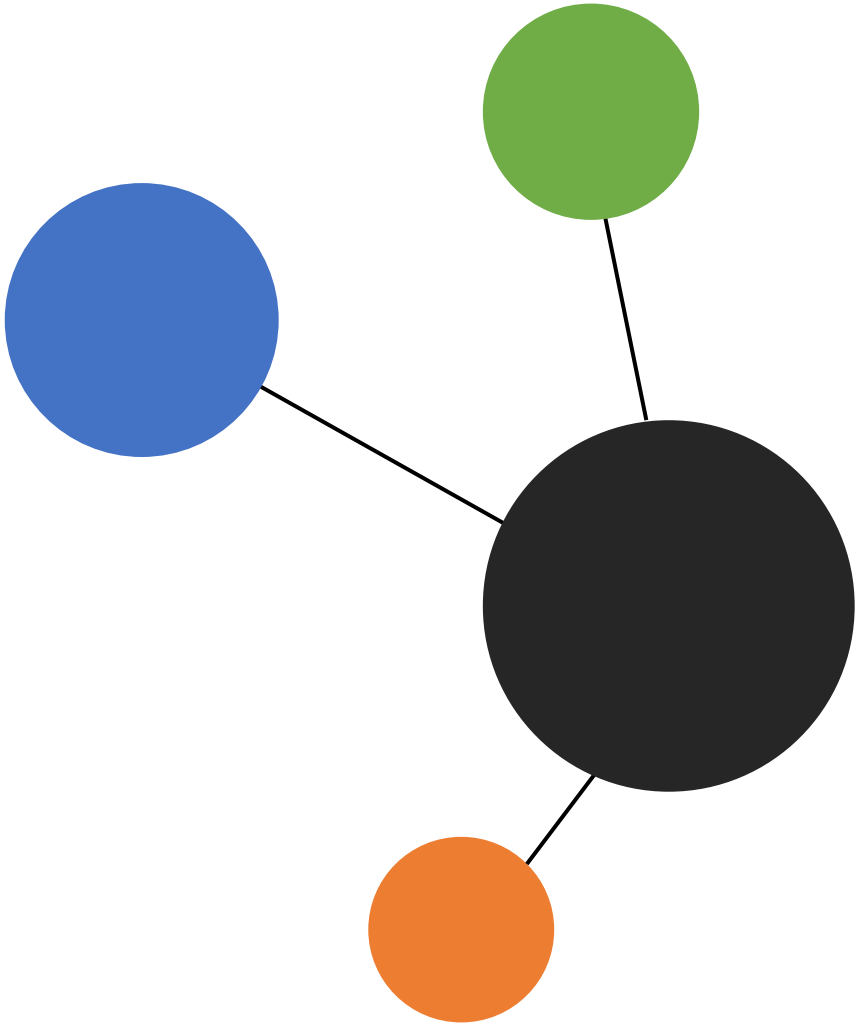
Brazilian ICPC Summer School

Fluxo Máximo

Augusto Damschi Bernardi



Definições e Notação

- Grafo orientado **G = {V, E}**
  - **V** é o conjunto de vertices.

  Ex: V = {A, B, C, D}

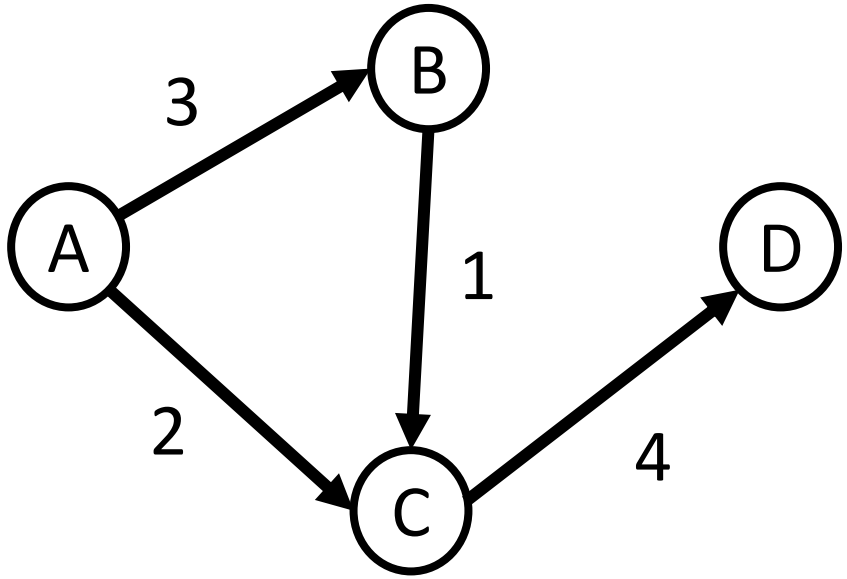
  - **E** é o conjunto de arestas (pares ordenados de vertices).

  Ex: E = {(A, B), (A,C), (B,C), (C,D)}

  - Grafo ponderado: cada aresta **e** tem um peso associado **w(e)**.

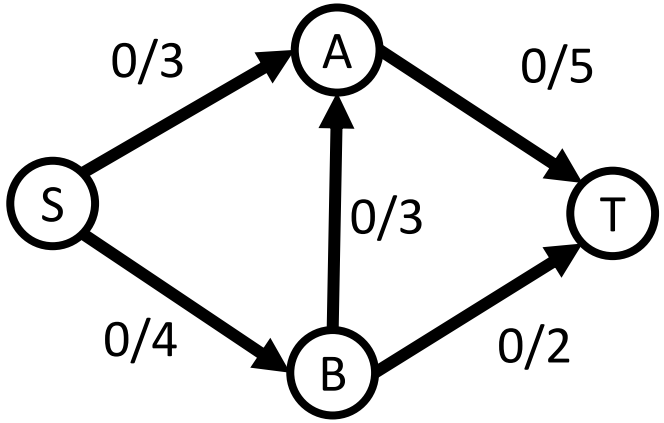
  Ex:

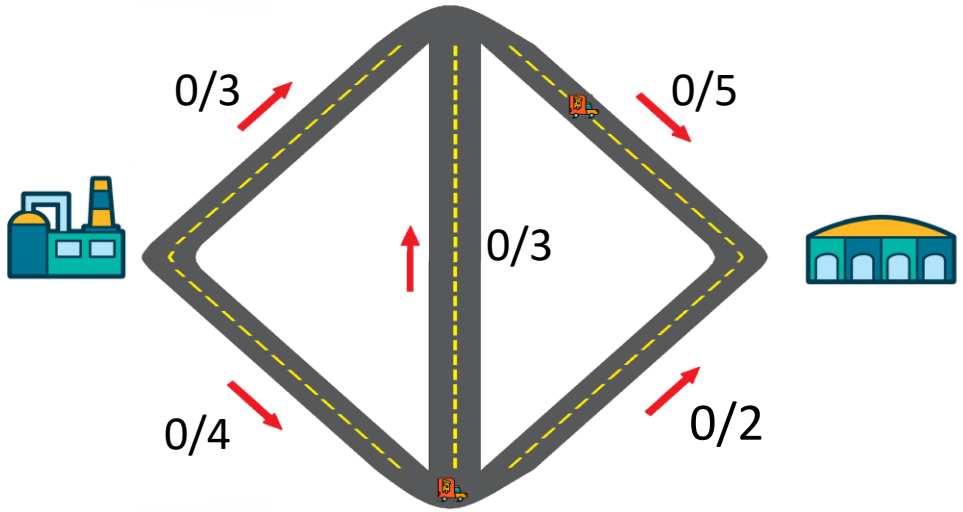
  ***w**((A, B)) = 3*
  ***w**((A,C)) = 2*
  ***w**((B,C)) = 1*
  ***w**((C,D)) = 4*



Redes

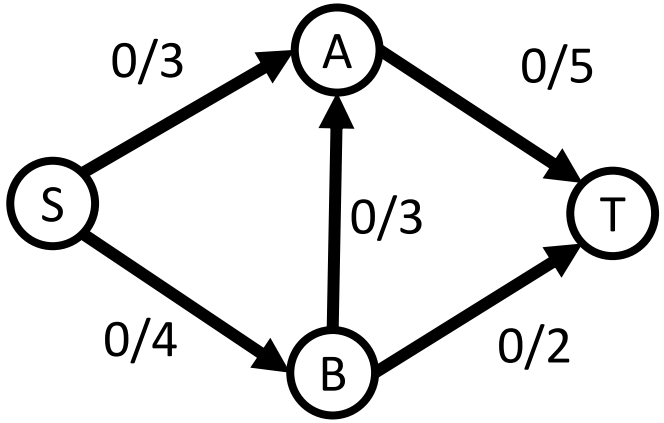
- Rede:
  - Grafo orientado **G = {V, E}**
  - Dois vertices especiais:
    - Fonte **S**
    - Dreno **T**
  - Cada aresta **e** tem uma capacidade **c(e)**
- Analogia: Sistema de estradas de mão única.
  - Fonte: Fábrica de onde saem caminhões
  - Dreno: Depósito onde os caminhões devem chegar
  - Cada estrada com capacidade em caminhões /minuto.

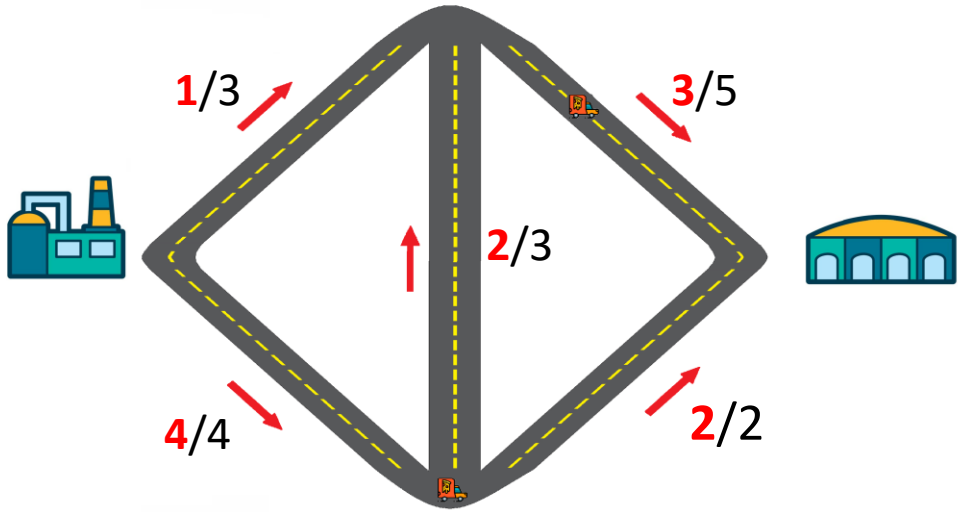




Redes

- Rede:
  - Grafo orientado **G = {V, E}**
  - Dois vertices especiais:
    - Fonte **S**
    - Dreno **T**
  - Cada aresta **e** tem uma capacidade **c(e)**
- Analogia: Sistema de estradas de mão única.
  - Fonte: Fábrica de onde saem caminhões
  - Dreno: Depósito onde os caminhões devem chegar
  - Cada estrada com capacidade em caminhões /minuto.

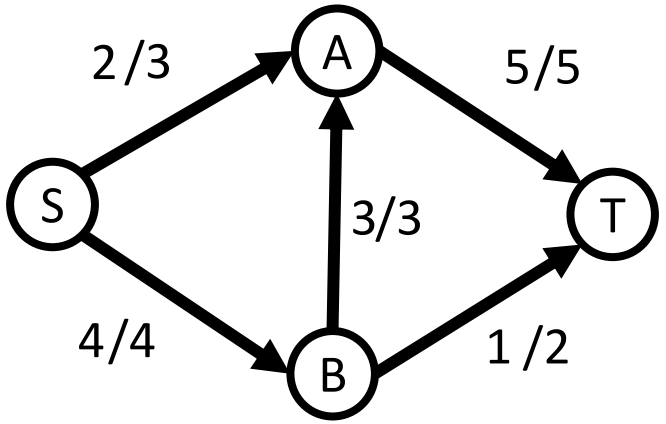


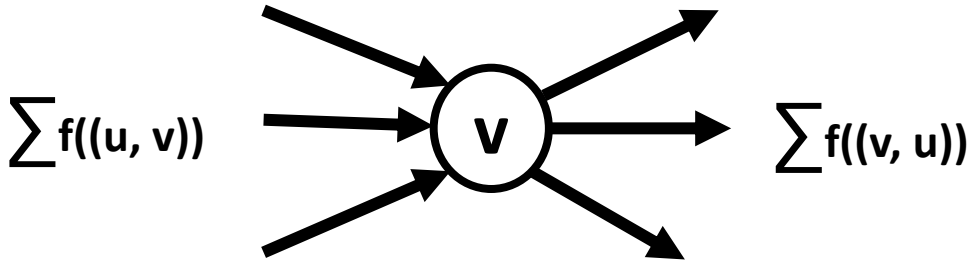


Fluxos

- Fluxo:
  - Cada aresta **e** recebe um valor **f(e)**
  - Restrições:
    - $\mathbf{0 \leq f(e) \leq c(e)}$
    - Para qualquer vértice $\mathbf{v \neq S, T}$:

$$\sum_{(u,\,v)\in E} f((u, v)) = \sum_{(v,\,u)\in E} f((v, u))$$



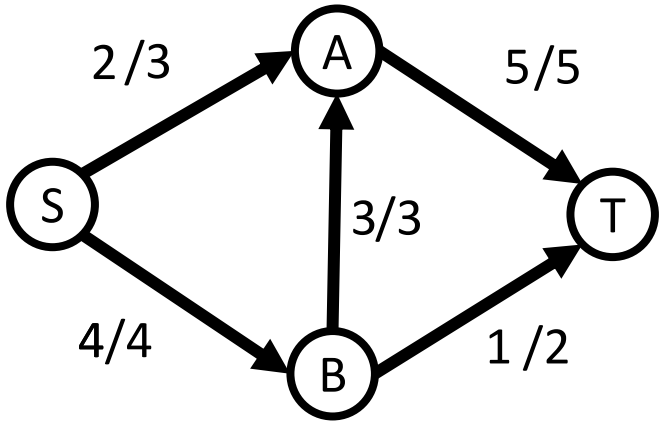


Fluxos

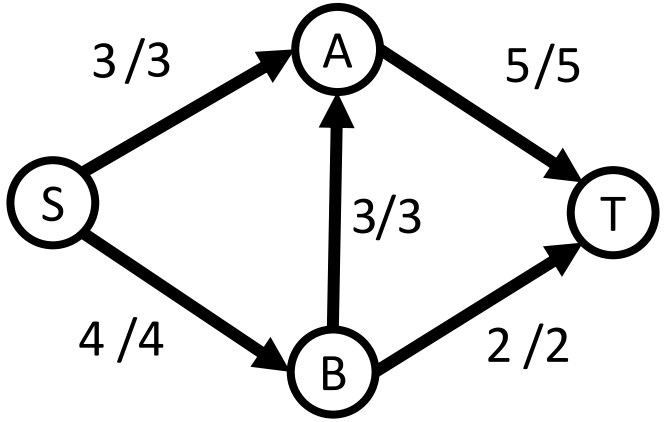
- Fluxo:
  - Cada aresta **e** recebe um valor **f(e)**
  - Restrições:
    - $\mathbf{0 \leq f(e) \leq c(e)}$
    - Para qualquer vértice $\mathbf{v \neq S, T}$:

$$\sum_{(u, v) \in E} f((u, v)) = \sum_{(v, u) \in E} f((v, u))$$

- Teorema:
  - $\sum_{(S, u) \in E} f((S, u)) = \sum_{(u, T) \in E} f((u, T))$ = Fluxo Total



Fluxo Total = 6

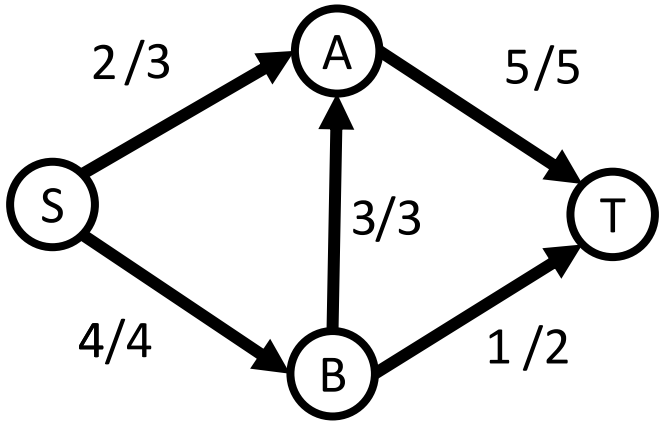


Fluxos

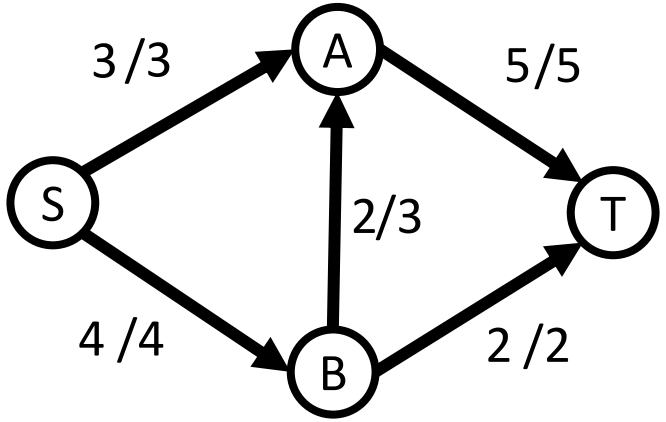
- Fluxo:
  - Cada aresta **e** recebe um valor **f(e)**
  - Restrições:
    - $\mathbf{0 \leq f(e) \leq c(e)}$
    - Para qualquer vértice $\mathbf{v \neq S, T}$:

$$\sum_{(u, v) \in E} f((u, v)) = \sum_{(v, u) \in E} f((v, u))$$

- Teorema:
  - $\sum_{(S, u) \in E} f((S, u)) = \sum_{(u, T) \in E} f((u, T))$ = Fluxo Total



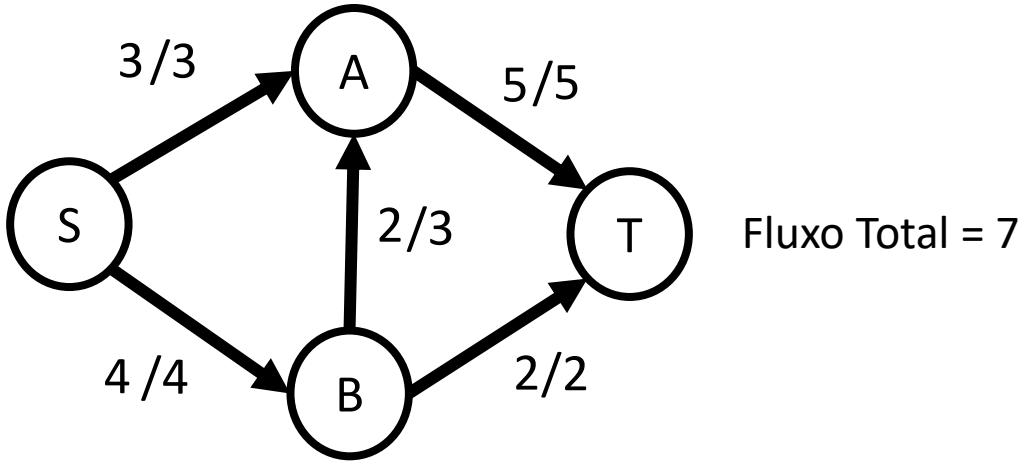
Fluxo Total = 6



Fluxo Total = 7

Problema do fluxo máximo

- Entrada:
  - Grafo orientado **G = {V, E}**
  - **S**, **T** ∈ **V**
  - Capacidade de cada aresta **c : E** → $\mathbb{Z}$
- Saída:
  - Fluxo em cada aresta **f** : **E** → $\mathbb{Z}$, tal que:
    - **f** respeite as restrições de fluxo
    - O fluxo total seja o máximo possível



* Teorema do fluxo inteiro

Exemplo

UVA 11418 - Clever Naming Patterns

- **N** problemas para serem nomeados
- $\mathbf{k_i}$ nomes possíveis para o **i-ésimo** problema.
- Escolher nomes para os problemas de maneira que um comece com A, outro com B, e assim até a N-ésima letra do alfabeto.

Exemplo:

  - *Problema 1: Abacate, Banana*
  - *Problema 2: Amora, Banana, Cenoura*
  - *Problema 3: Brócolis, Cenoura*

Exemplo

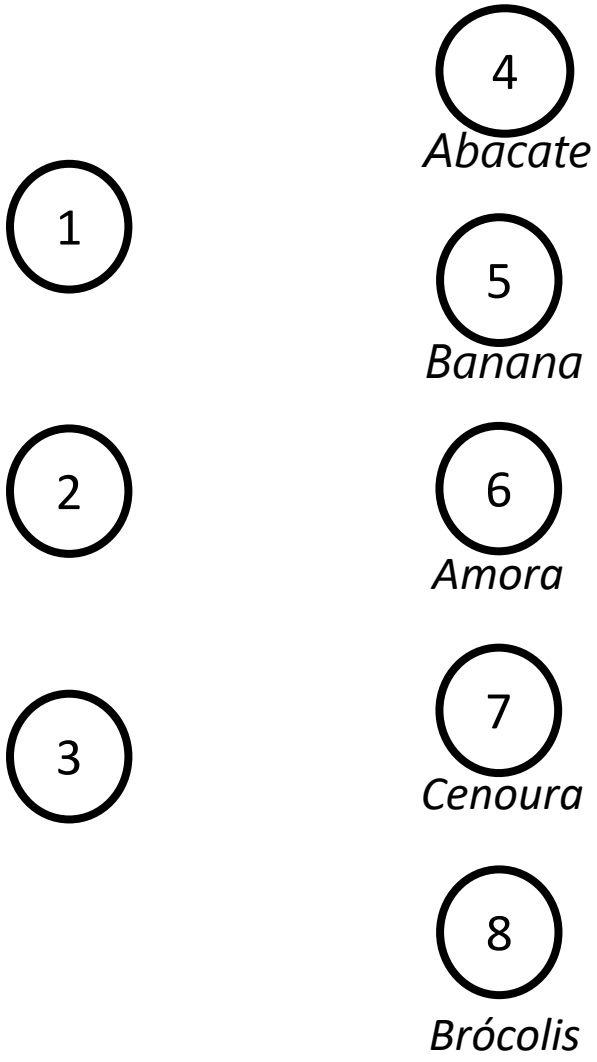
UVA 11418 - Clever Naming Patterns

- **N** problemas para serem nomeados
- $\mathbf{k_i}$ nomes possíveis para o **i-ésimo** problema.
- Escolher nomes para os problemas de maneira que um comece com A, outro com B, e assim até a N-ésima letra do alfabeto.

Exemplo:

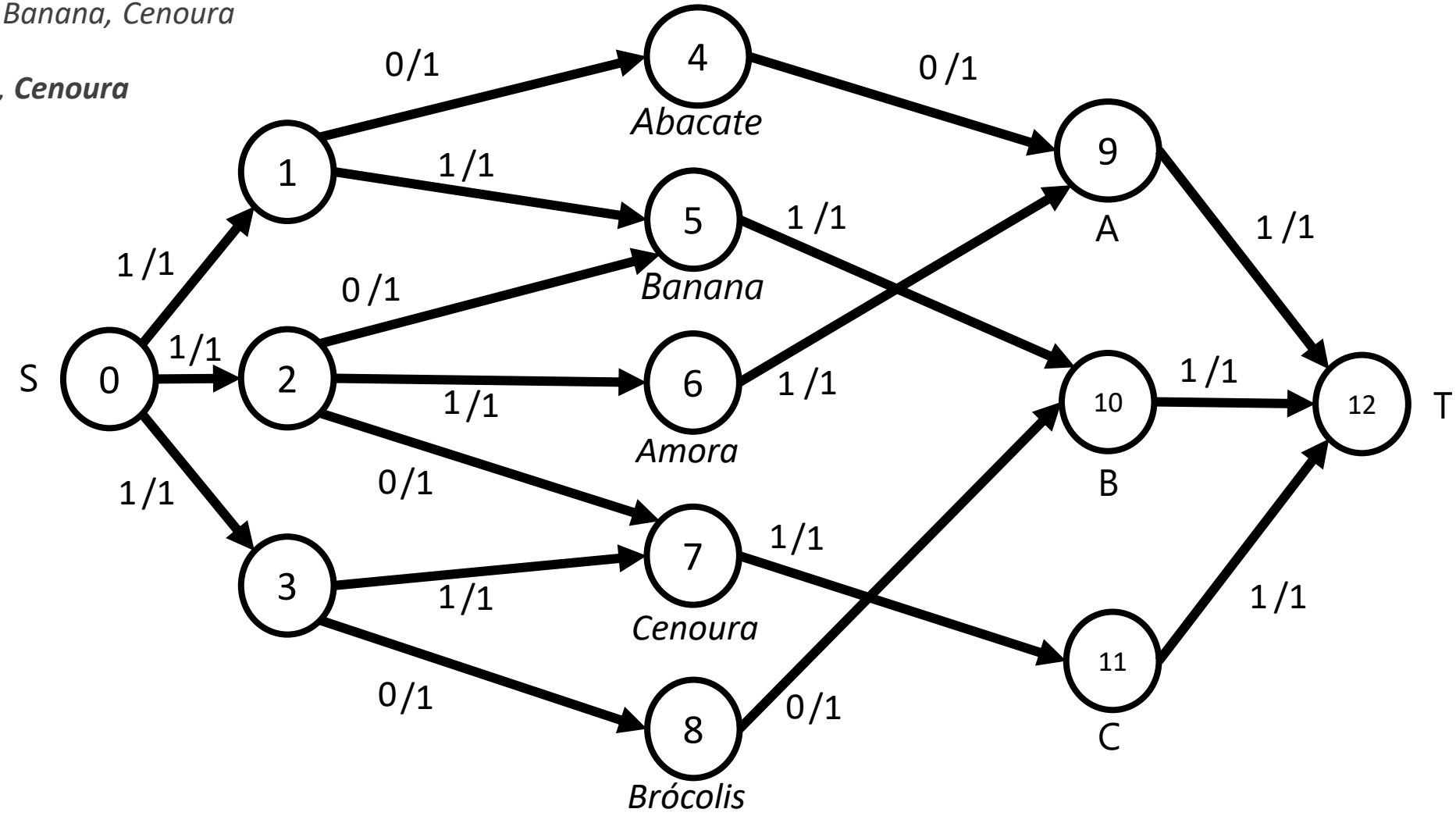
  - *Problema 1: Abacate,* ***Banana***
  - *Problema 2:* ***Amora****, Banana, Cenoura*
  - *Problema 3: Brócolis,* ***Cenoura***

- Suponha que pode existir mais de uma solução, ou nenhuma solução.



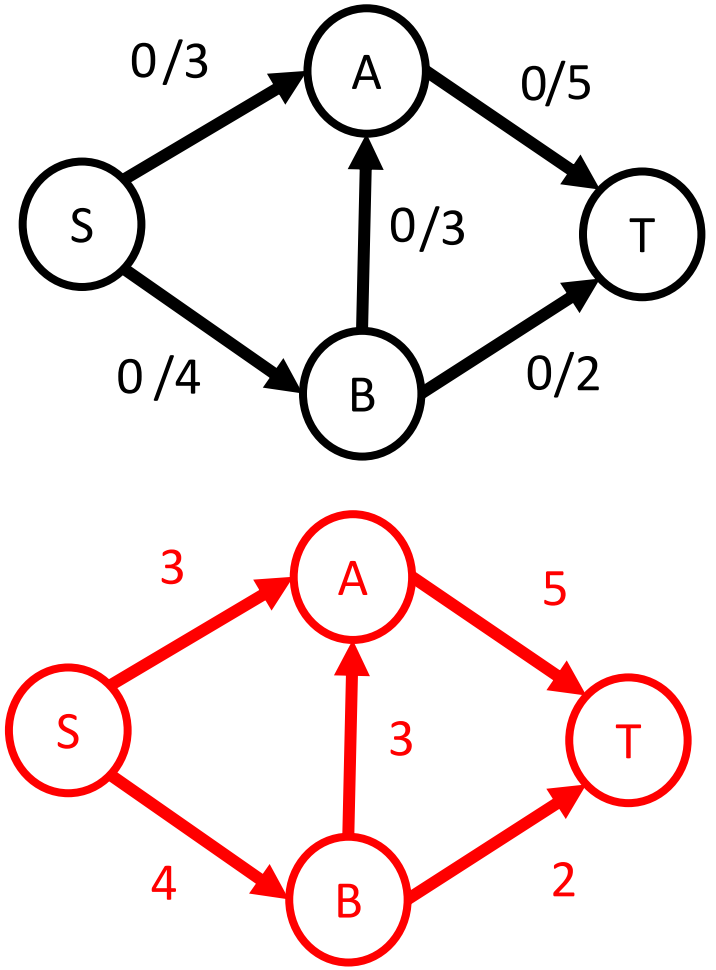
Exemplo

- *Problema 1: Abacate, **Banana***
- *Problema 2: **Amora**, Banana, Cenoura*
- *Problema 3: Brócolis, **Cenoura***



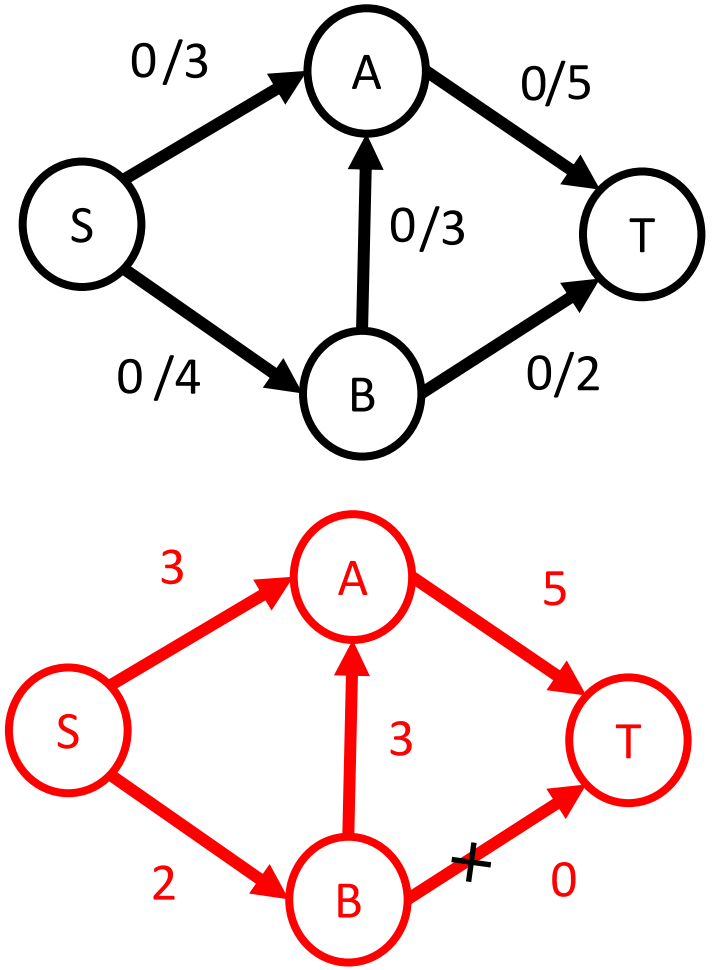
Algoritmo para encontrar fluxo máximo

- Entrada:
  - **G = {V, E}**
  - **c(e): E** $\rightarrow \mathbb{Z}$
- Ideia: Adicionar fluxo de **S** para **T** enquanto for possível.
  - Inicia fluxo em todas as arestas com zero
  - Cria grafo auxiliar $\mathbf{G^A}$ com pesos $\mathbf{c^A((u, v)) = c((u, v))}$
  - Enquanto existir um caminho **P** de **S** até **T** em $\mathbf{G^A}$
    - Seja **x** menor peso de uma aresta em **P**
    - Para toda aresta **(u, v)** em **P**:
      - $\mathbf{c^A((u, v)) \text{ -= } x}$
      - $\mathbf{f((u, v)) = c((u, v)) - c^A((u, v))}$



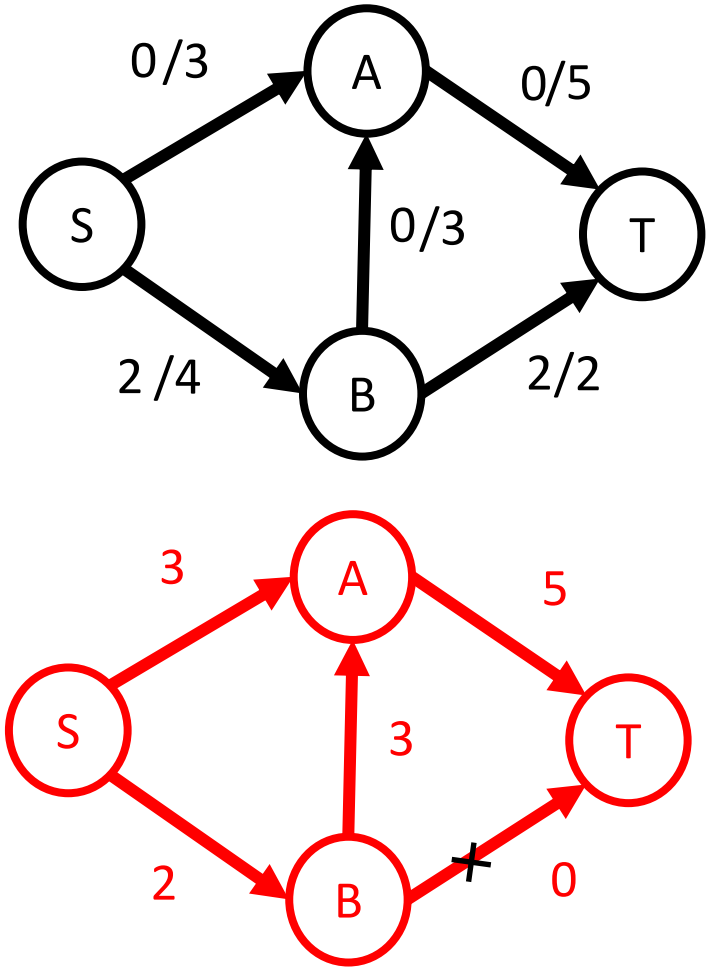
Algoritmo para encontrar fluxo máximo

- Entrada:
  - **G = {V, E}**
  - **c(e): E** $\rightarrow \mathbb{Z}$
- Ideia: Adicionar fluxo de **S** para **T** enquanto for possível.
  - Inicia fluxo em todas as arestas com zero
  - Cria grafo auxiliar $\mathbf{G^A}$ com pesos $\mathbf{c^A((u, v)) = c((u, v))}$
  - Enquanto existir um caminho **P** de **S** até **T** em $\mathbf{G^A}$
    - Seja **x** menor peso de uma aresta em **P**
    - Para toda aresta **(u, v)** em **P**:
      - $\mathbf{c^A((u, v)) \text{ -= } x}$
      - $\mathbf{f((u, v)) = c((u, v)) - c^A((u, v))}$



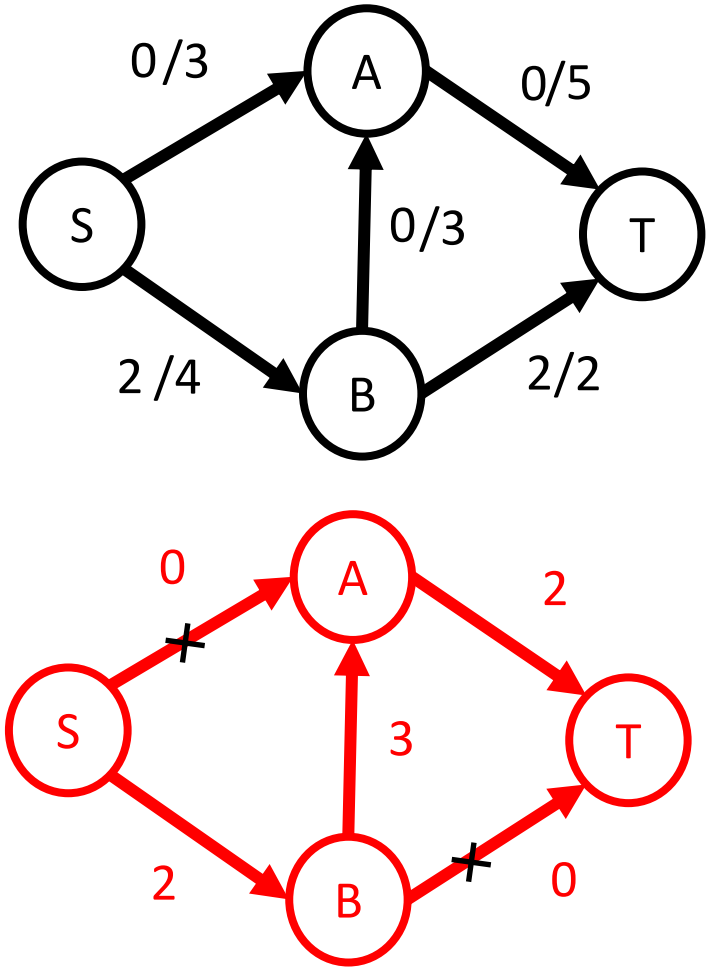
Algoritmo para encontrar fluxo máximo

- Entrada:
  - **G = {V, E}**
  - **c(e): E** $\rightarrow \mathbb{Z}$
- Ideia: Adicionar fluxo de **S** para **T** enquanto for possível.
  - Inicia fluxo em todas as arestas com zero
  - Cria grafo auxiliar $\mathbf{G^A}$ com pesos $\mathbf{c^A((u, v)) = c((u, v))}$
  - Enquanto existir um caminho **P** de **S** até **T** em $\mathbf{G^A}$
    - Seja **x** menor peso de uma aresta em **P**
    - Para toda aresta **(u, v)** em **P**:
      - $\mathbf{c^A((u, v)) \text{ -= } x}$
      - $\mathbf{f((u, v)) = c((u, v)) - c^A((u, v))}$



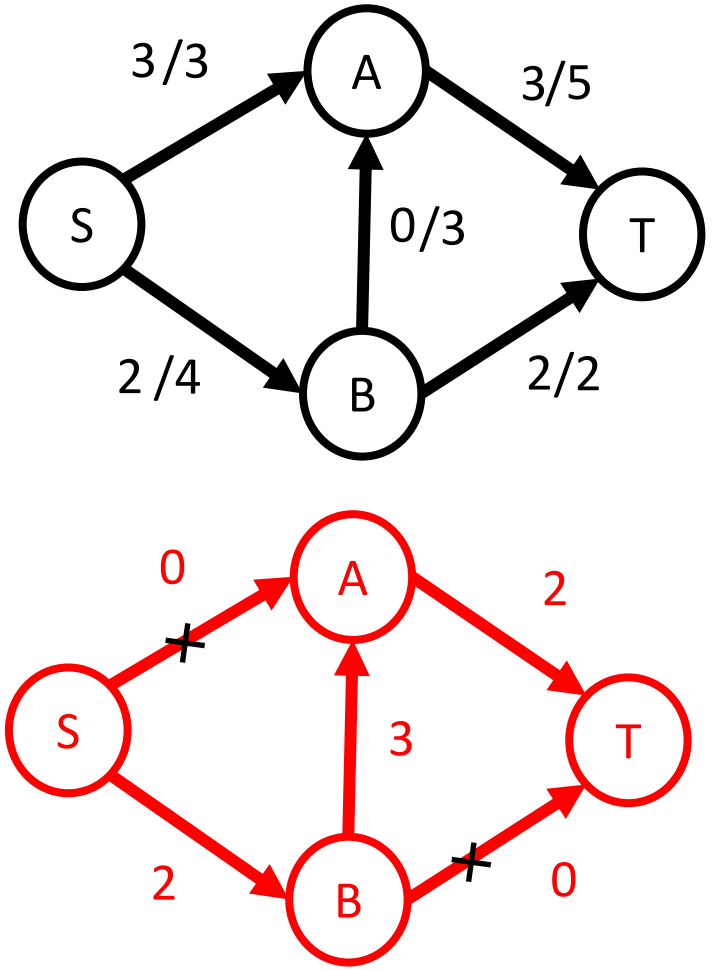
Algoritmo para encontrar fluxo máximo

- Entrada:
  - **G = {V, E}**
  - **c(e): E** $\rightarrow \mathbb{Z}$
- Ideia: Adicionar fluxo de **S** para **T** enquanto for possível.
  - Inicia fluxo em todas as arestas com zero
  - Cria grafo auxiliar $\mathbf{G^A}$ com pesos $\mathbf{c^A((u, v)) = c((u, v))}$
  - Enquanto existir um caminho **P** de **S** até **T** em $\mathbf{G^A}$
    - Seja **x** menor peso de uma aresta em **P**
    - Para toda aresta **(u, v)** em **P**:
      - $\mathbf{c^A((u, v)) \text{ -= } x}$
      - $\mathbf{f((u, v)) = c((u, v)) - c^A((u, v))}$



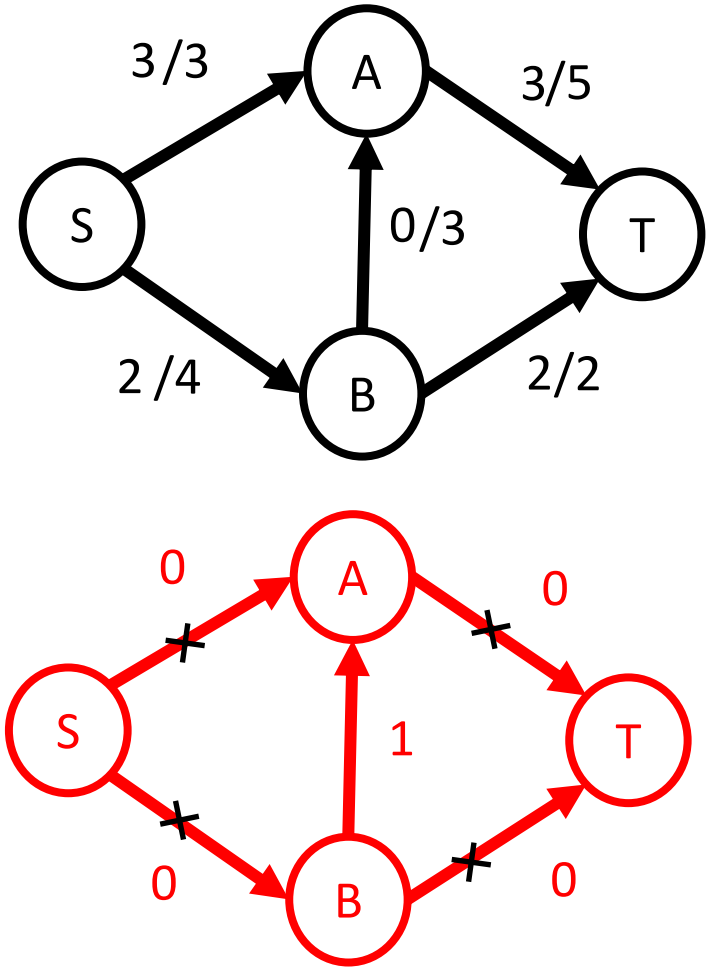
Algoritmo para encontrar fluxo máximo

- Entrada:
  - **G = {V, E}**
  - **c(e): E** $\rightarrow \mathbb{Z}$
- Ideia: Adicionar fluxo de **S** para **T** enquanto for possível.
  - Inicia fluxo em todas as arestas com zero
  - Cria grafo auxiliar $\mathbf{G^A}$ com pesos $\mathbf{c^A((u, v)) = c((u, v))}$
  - Enquanto existir um caminho **P** de **S** até **T** em $\mathbf{G^A}$
    - Seja **x** menor peso de uma aresta em **P**
    - Para toda aresta **(u, v)** em **P**:
      - $\mathbf{c^A((u, v)) -= x}$
      - $\mathbf{f((u, v)) = c((u, v)) - c^A((u, v))}$



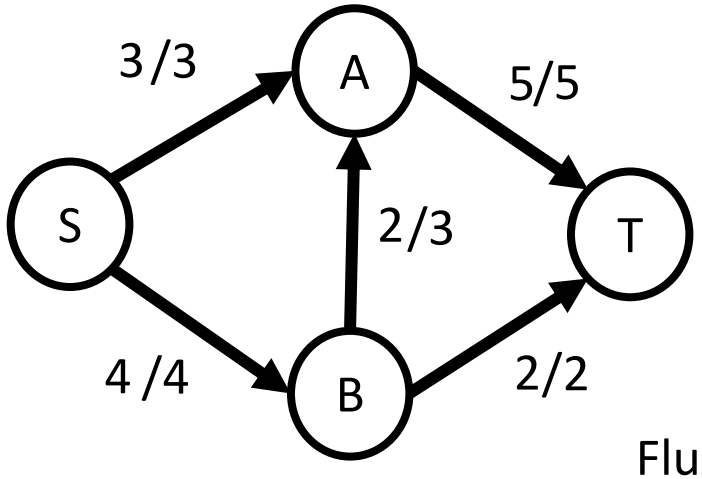
Algoritmo para encontrar fluxo máximo

- Entrada:
  - **G = {V, E}**
  - **c(e): E** $\rightarrow \mathbb{Z}$
- Ideia: Adicionar fluxo de **S** para **T** enquanto for possível.
  - Inicia fluxo em todas as arestas com zero
  - Cria grafo auxiliar $\mathbf{G^A}$ com pesos $\mathbf{c^A((u, v)) = c((u, v))}$
  - Enquanto existir um caminho **P** de **S** até **T** em $\mathbf{G^A}$
    - Seja **x** menor peso de uma aresta em **P**
    - Para toda aresta **(u, v)** em **P**:
      - $\mathbf{c^A((u, v)) \text{ -= } x}$
      - $\mathbf{f((u, v)) = c((u, v)) - c^A((u, v))}$

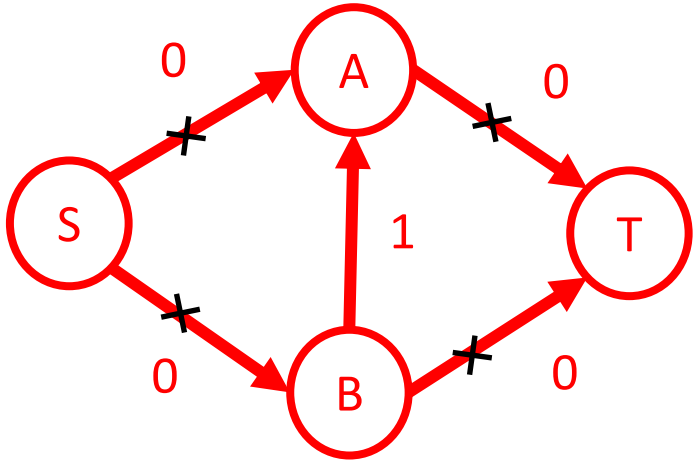


Algoritmo para encontrar fluxo máximo

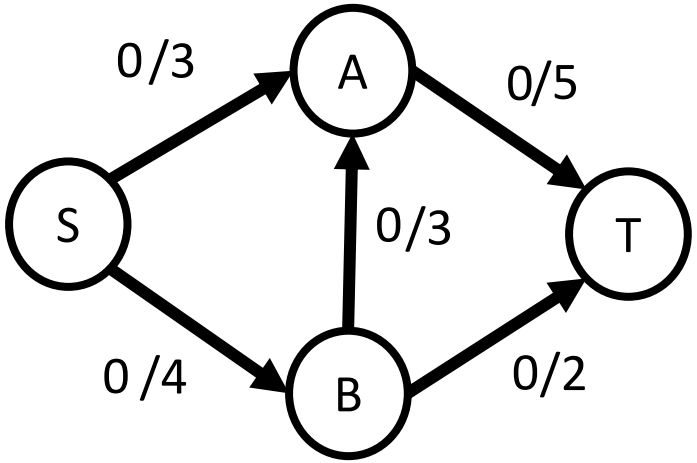
- Entrada:
  - **G = {V, E}**
  - **c(e): E** $\rightarrow \mathbb{Z}$
- Ideia: Adicionar fluxo de **S** para **T** enquanto for possível.
  - Inicia fluxo em todas as arestas com zero
  - Cria grafo auxiliar $\mathbf{G^A}$ com pesos $\mathbf{c^A((u, v)) = c((u, v))}$
  - Enquanto existir um caminho **P** de **S** até **T** em $\mathbf{G^A}$
    - Seja **x** menor peso de uma aresta em **P**
    - Para toda aresta **(u, v)** em **P**:
      - $\mathbf{c^A((u, v)) \text{ -= } x}$
      - $\mathbf{f((u, v)) = c((u, v)) - c^A((u, v))}$
- Essa estratégia sempre resulta no fluxo máximo? **NÃO!**

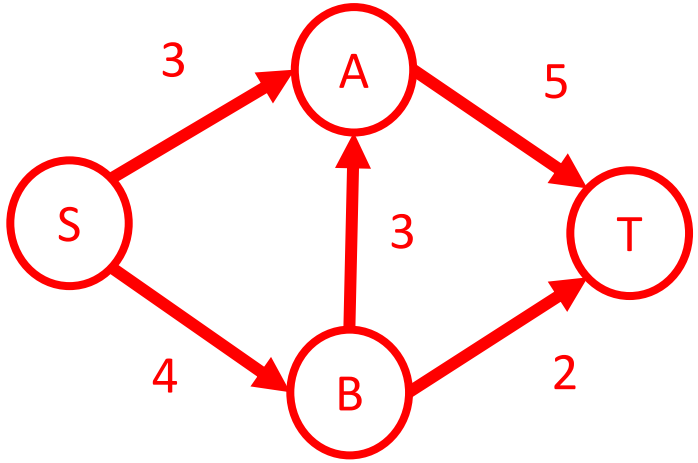


Fluxo Total = 7

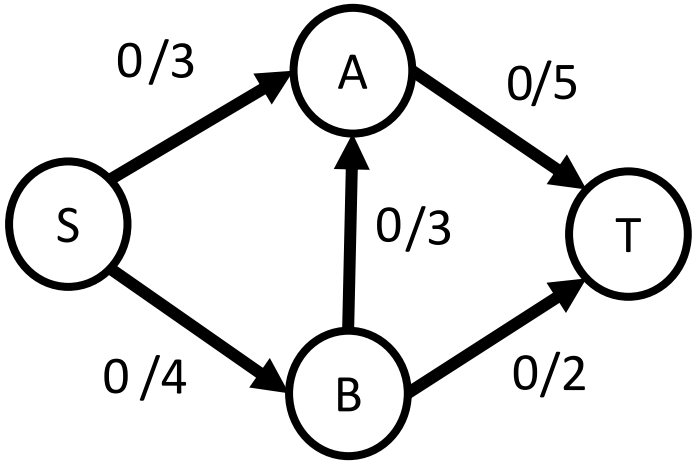


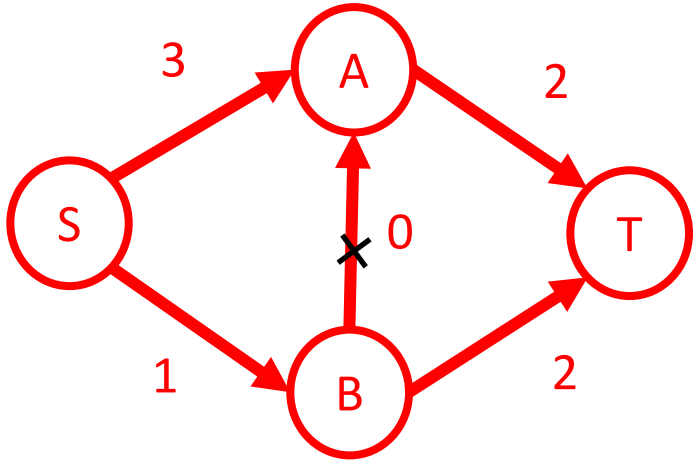
Algoritmo para encontrar fluxo máximo



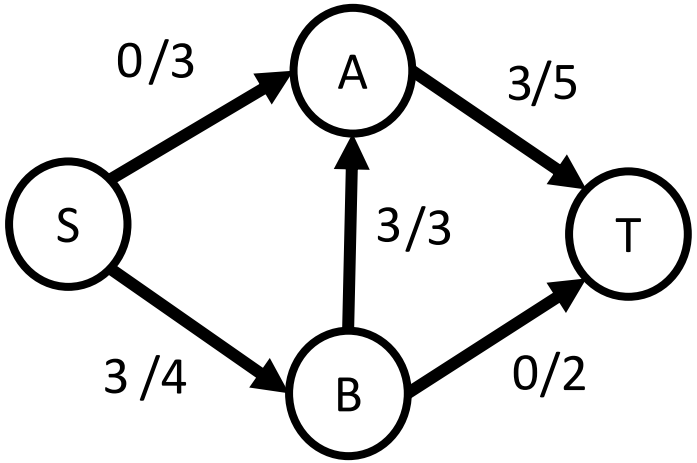


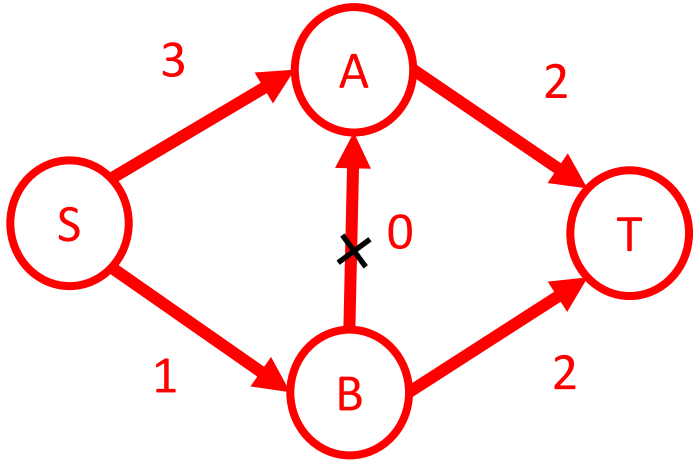
Algoritmo para encontrar fluxo máximo



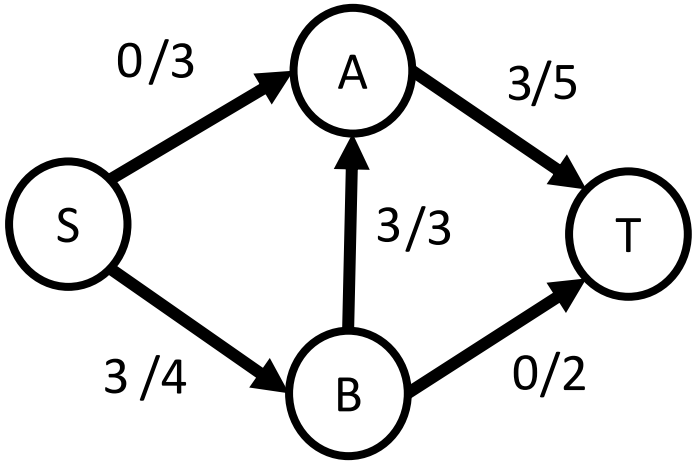


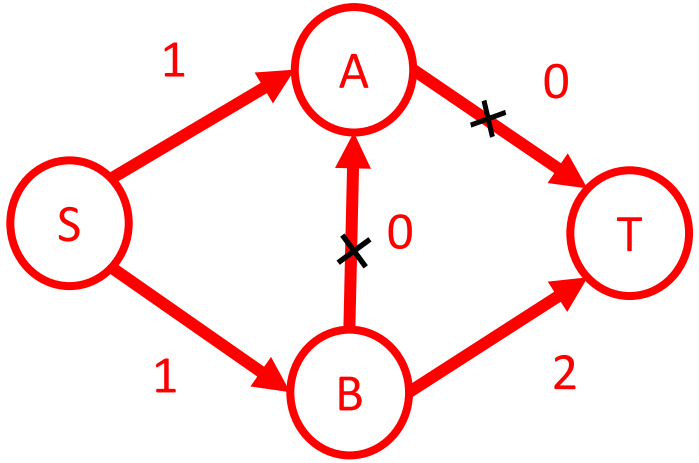
Algoritmo para encontrar fluxo máximo



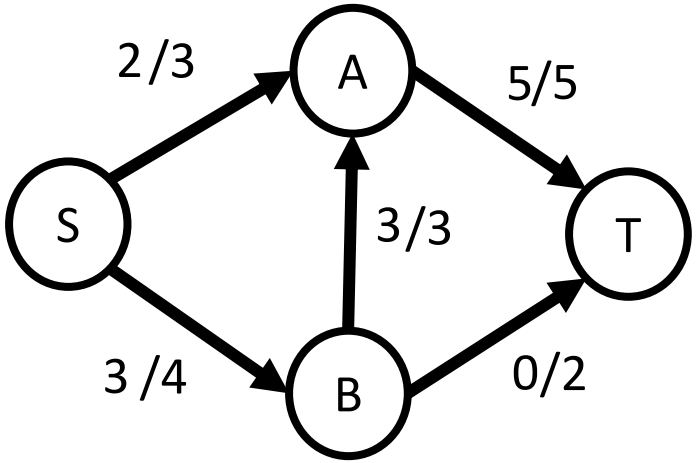


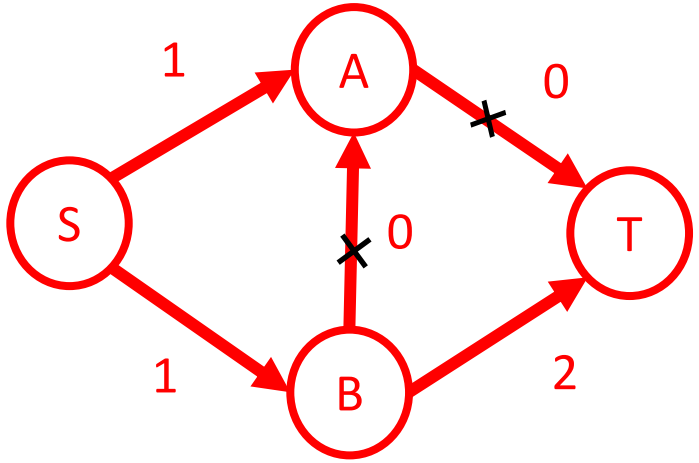
Algoritmo para encontrar fluxo máximo



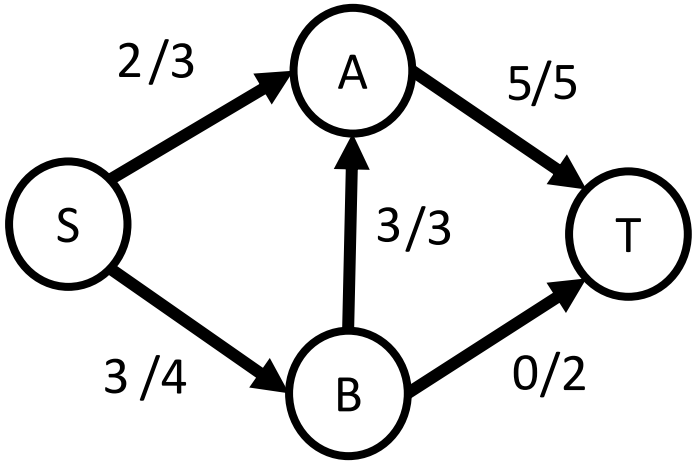


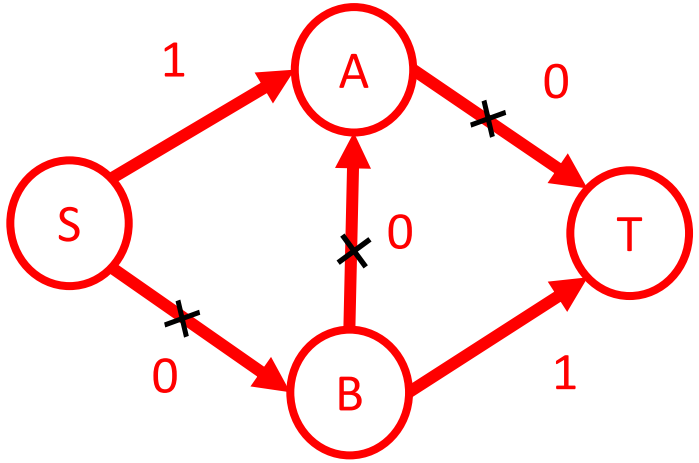
Algoritmo para encontrar fluxo máximo



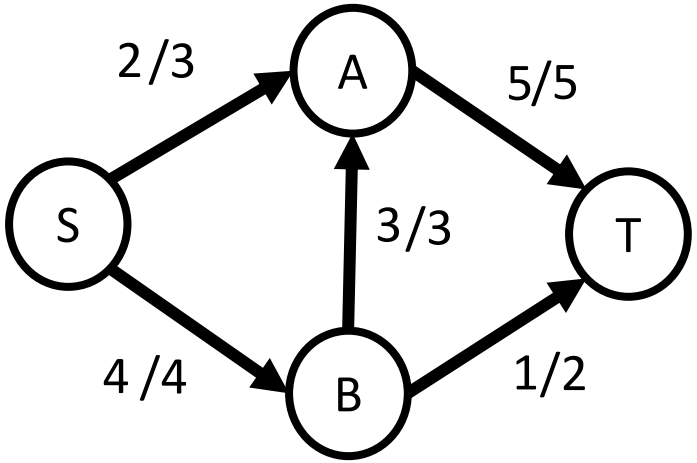


Algoritmo para encontrar fluxo máximo

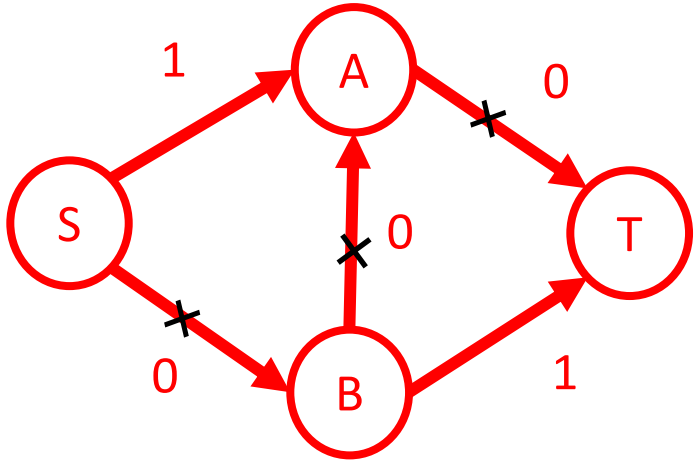




Algoritmo para encontrar fluxo máximo

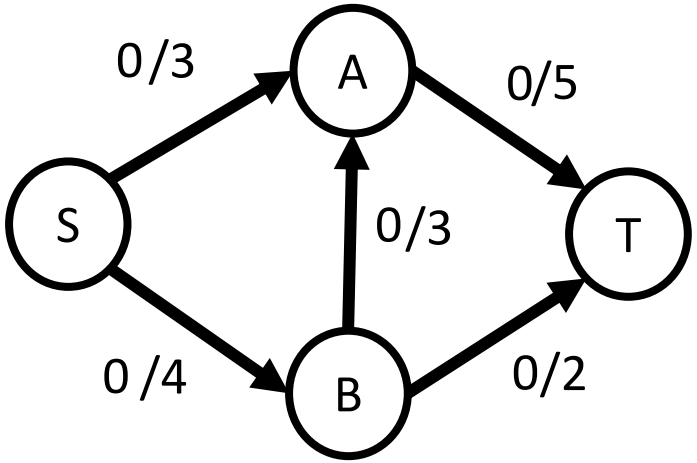


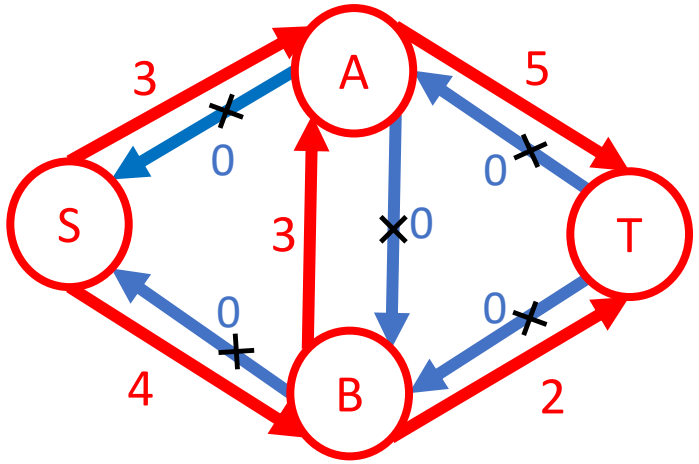
Fluxo Total = 6



- **Problema**: O algoritmo anterior pode acabar com os caminhos de **S** até **T** e ainda apresentar um fluxo sub-ótimo.
- **Ideia**: Dar uma maneira do algoritmo retirar fluxo já alocado nas arestas.

Algoritmo para encontrar fluxo máximo





- **Problema**: O algoritmo anterior pode acabar com os caminhos de **S** até **T** e ainda apresentar um fluxo sub-ótimo.
- **Ideia**: Dar uma maneira do algoritmo retirar fluxo já alocado nas arestas.
- Criar grafo G^R. Para cada aresta **(u, v)** em **G**:

  $c^R((u, v)) = c((u, v))$

  $c^R((v, u))$ = 0 se **(v, u)** não for parte de **G**

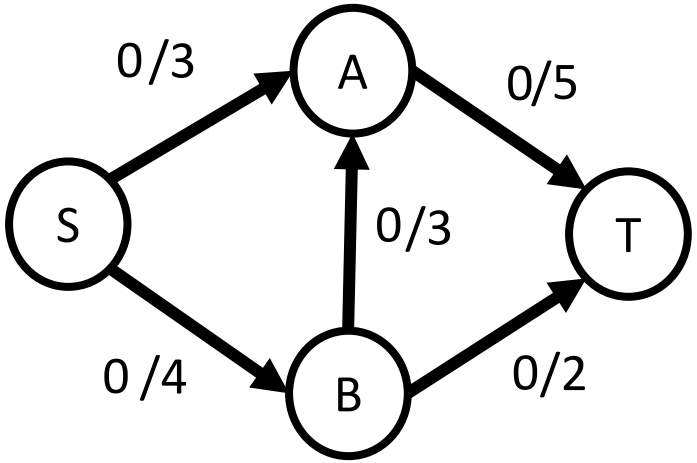
- e atualizar pesos com:

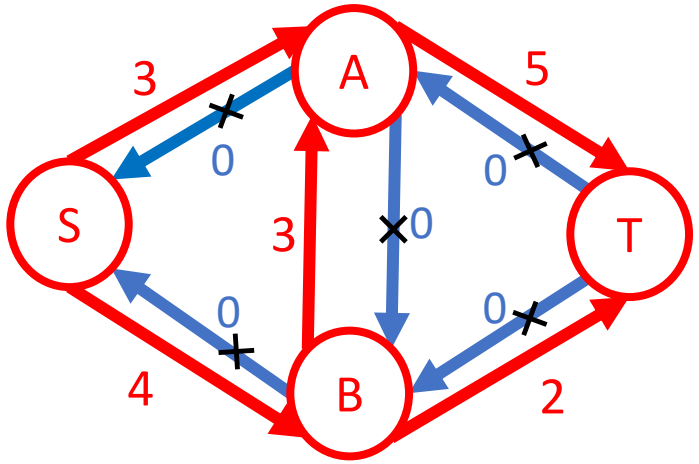
  $c^R((u, v)) \mathrel{-}= x$

  $c^R((v, u)) \mathrel{+}= x$

  $f((u, v)) = c((u, v)) - c^R((u, v))$

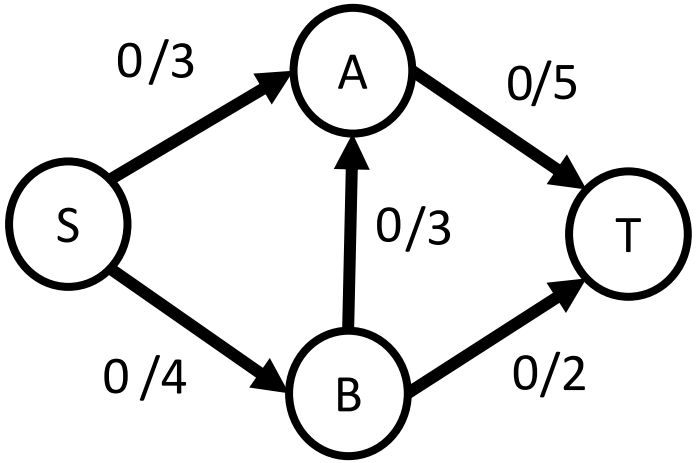
Algoritmo para encontrar fluxo máximo

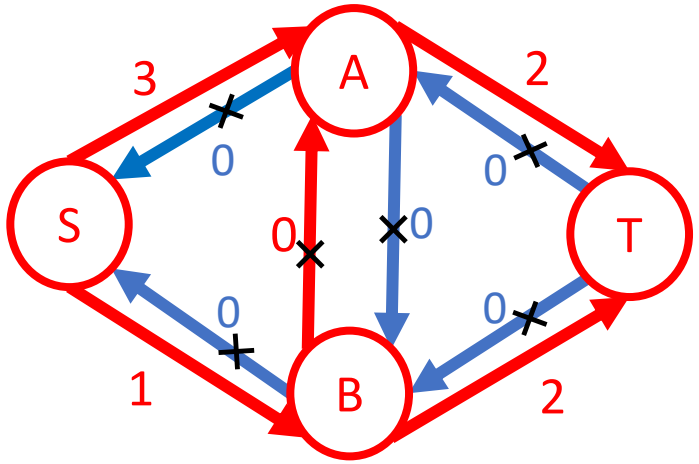




- Novo Algoritmo:
  - Inicia fluxo em todas as arestas com zero
  - Cria grafo auxiliar $\mathbf{G^R}$ com pesos $\mathbf{c^R((u, v)) = c((u, v))}$
  - Enquanto existir um caminho **P** de **S** até **T** em **G**
    - Seja **x** menor peso de uma aresta em **P**
    - Para todo **(u,v)** em **P**
      - $\mathbf{c^R((u, v)) \mathrel{-}= x}$
      - $\mathbf{c^R((v, u)) \mathrel{+}= x}$
      - $\mathbf{f((u, v)) = c((u, v)) - c^R((u, v))}$

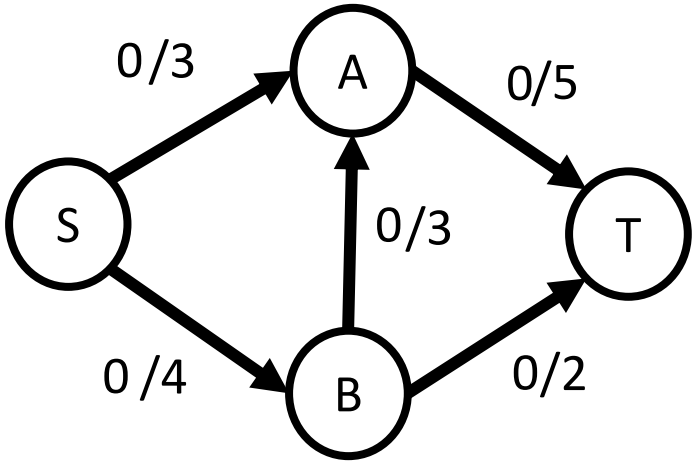
Algoritmo para encontrar fluxo máximo

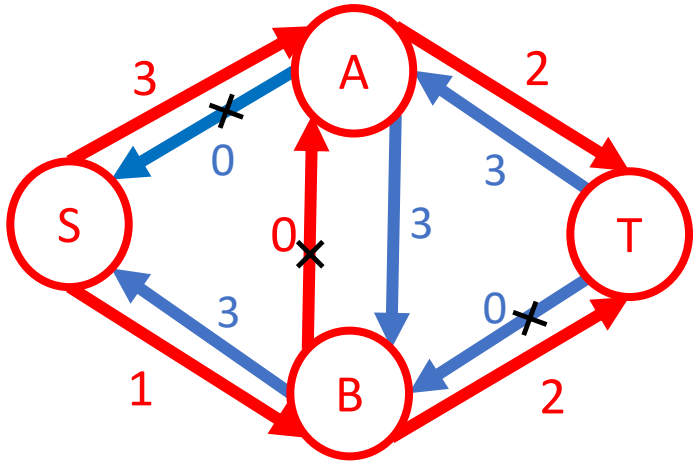




- Novo Algoritmo:
  - Inicia fluxo em todas as arestas com zero
  - Cria grafo auxiliar $\mathbf{G^R}$ com pesos $\mathbf{c^R((u, v)) = c((u, v))}$
  - Enquanto existir um caminho **P** de **S** até **T** em **G**
    - Seja **x** menor peso de uma aresta em **P**
    - Para todo **(u,v)** em **P**
      - $\mathbf{c^R((u, v)) \mathrel{-}= x}$
      - $\mathbf{c^R((v, u)) \mathrel{+}= x}$
      - $\mathbf{f((u, v)) = c((u, v)) - c^R((u, v))}$

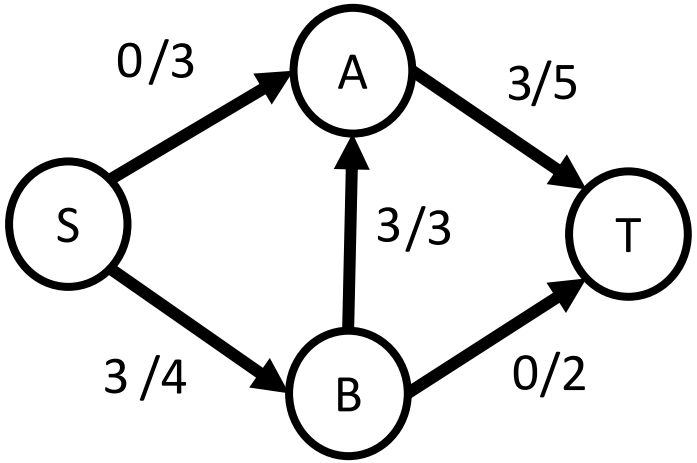
Algoritmo para encontrar fluxo máximo

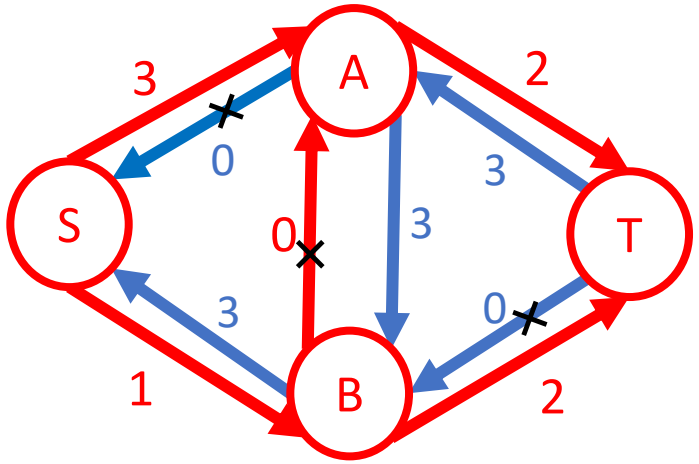




- Novo Algoritmo:
  - Inicia fluxo em todas as arestas com zero
  - Cria grafo auxiliar **G^R** com pesos **$c^R((u, v)) = c((u, v))$**
  - Enquanto existir um caminho **P** de **S** até **T** em **G**
    - Seja **x** menor peso de uma aresta em **P**
    - Para todo **(u,v)** em **P**
      - **$c^R((u, v))$ -= x**
      - **$c^R((v, u))$ += x**
      - **$f((u, v)) = c((u, v)) - c^R((u, v))$**

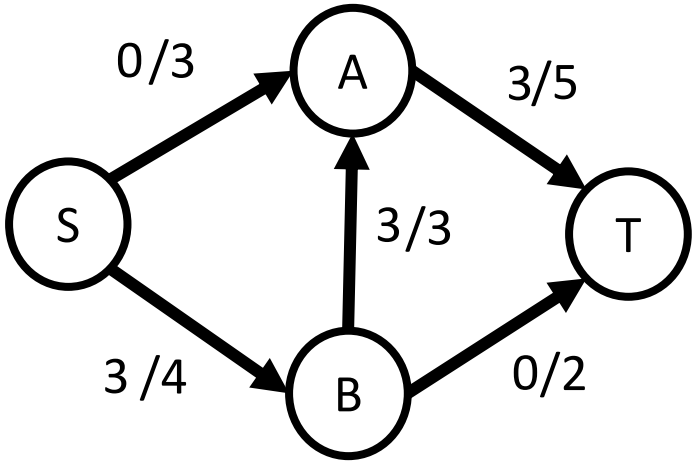
Algoritmo para encontrar fluxo máximo

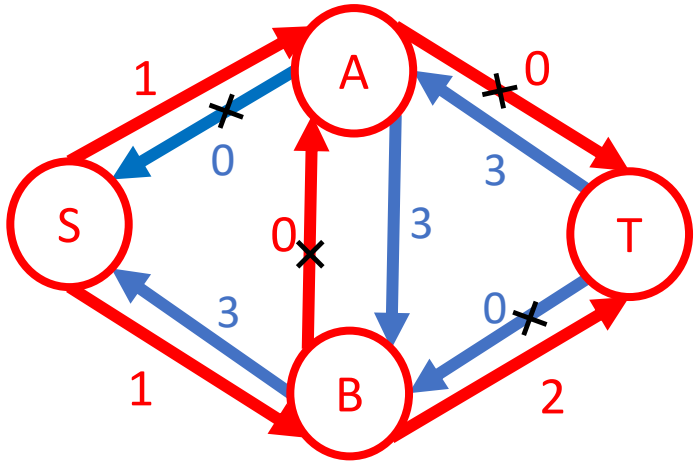




- Novo Algoritmo:
  - Inicia fluxo em todas as arestas com zero
  - Cria grafo auxiliar **G^R** com pesos **$c^R((u, v)) = c((u, v))$**
  - Enquanto existir um caminho **P** de **S** até **T** em **G**
    - Seja **x** menor peso de uma aresta em **P**
    - Para todo **(u,v)** em **P**
      - **$c^R((u, v))$ -= x**
      - **$c^R((v, u))$ += x**
      - **$f((u, v)) = c((u, v)) - c^R((u, v))$**

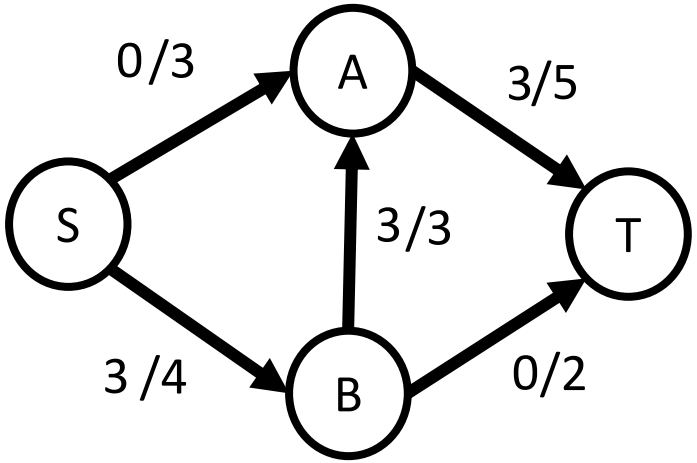
Algoritmo para encontrar fluxo máximo

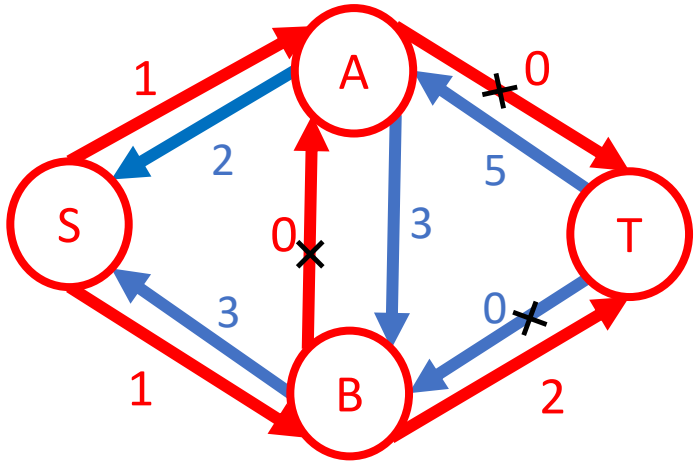




- Novo Algoritmo:
  - Inicia fluxo em todas as arestas com zero
  - Cria grafo auxiliar **G^R** com pesos **$c^R((u, v)) = c((u, v))$**
  - Enquanto existir um caminho **P** de **S** até **T** em **G**
    - Seja **x** menor peso de uma aresta em **P**
    - Para todo **(u,v)** em **P**
      - **$c^R((u, v))$ -= x**
      - **$c^R((v, u))$ += x**
      - **$f((u, v)) = c((u, v)) - c^R((u, v))$**

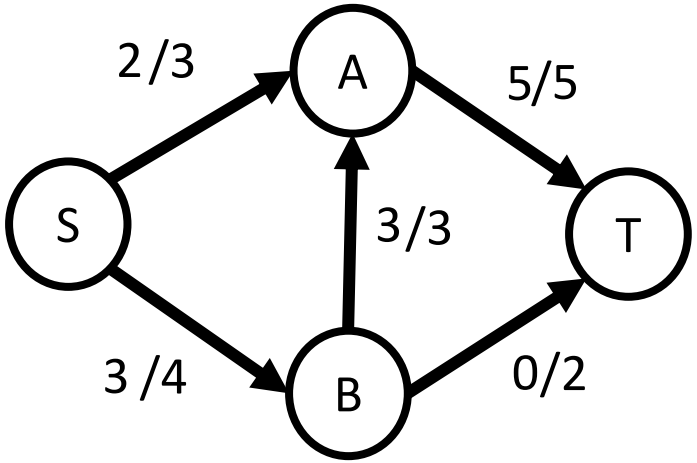
Algoritmo para encontrar fluxo máximo

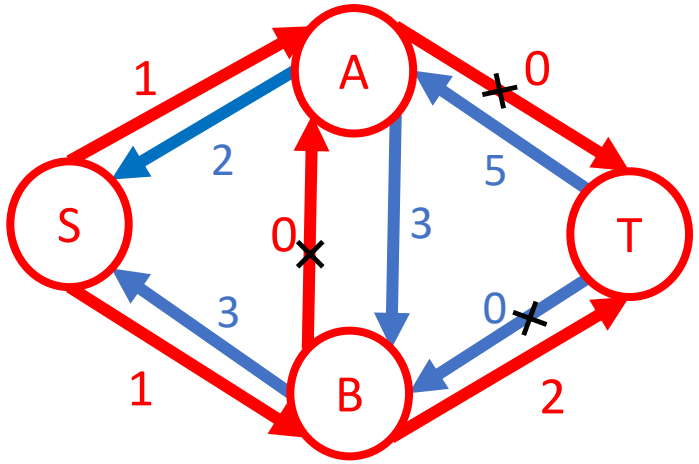




- Novo Algoritmo:
  - Inicia fluxo em todas as arestas com zero
  - Cria grafo auxiliar $\mathbf{G^R}$ com pesos $\mathbf{c^R((u, v)) = c((u, v))}$
  - Enquanto existir um caminho **P** de **S** até **T** em **G**
    - Seja **x** menor peso de uma aresta em **P**
    - Para todo **(u,v)** em **P**
      - $\mathbf{c^R((u, v)) \mathrel{-}= x}$
      - $\mathbf{c^R((v, u)) \mathrel{+}= x}$
      - $\mathbf{f((u, v)) = c((u, v)) - c^R((u, v))}$

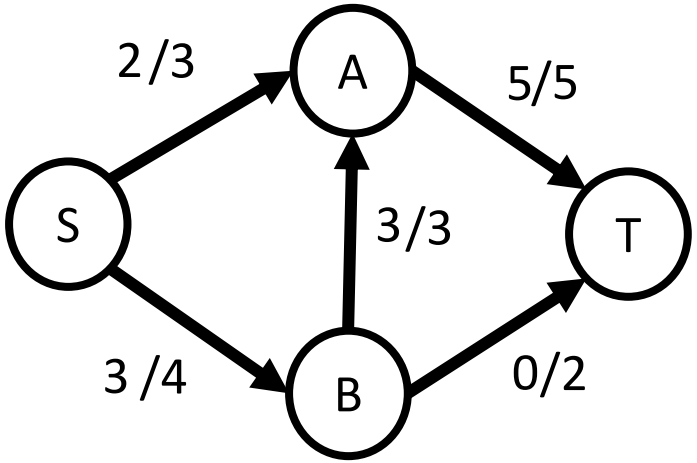
Algoritmo para encontrar fluxo máximo

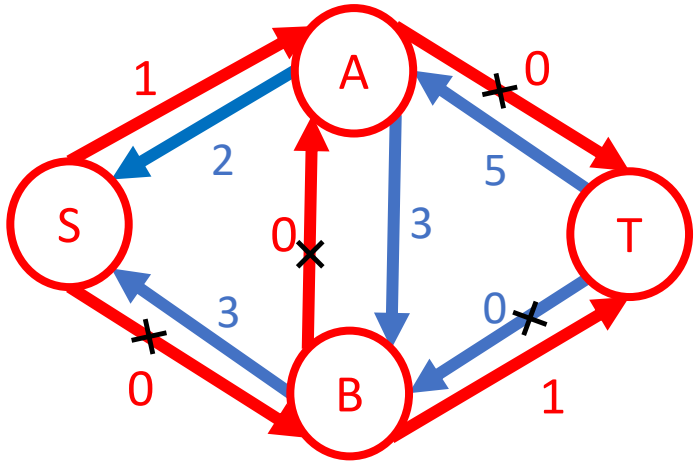




- Novo Algoritmo:
  - Inicia fluxo em todas as arestas com zero
  - Cria grafo auxiliar G^R com pesos $c^R((u, v)) = c((u, v))$
  - Enquanto existir um caminho **P** de **S** até **T** em **G**
    - Seja **x** menor peso de uma aresta em **P**
    - Para todo **(u,v)** em **P**
      - $c^R((u, v)) \mathrel{-}= x$
      - $c^R((v, u)) \mathrel{+}= x$
      - $f((u, v)) = c((u, v)) - c^R((u, v))$

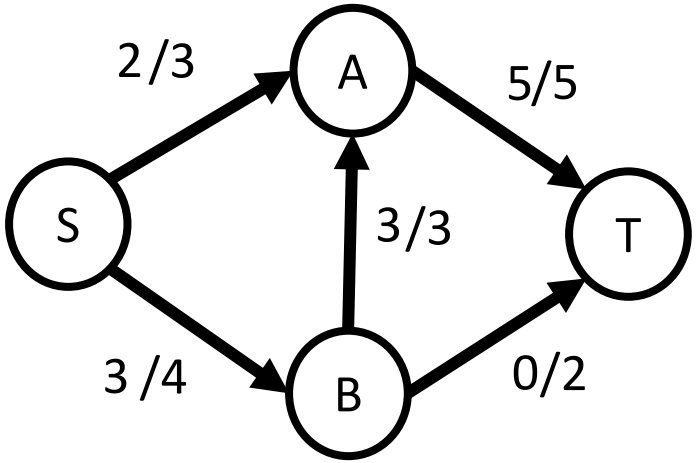
Algoritmo para encontrar fluxo máximo

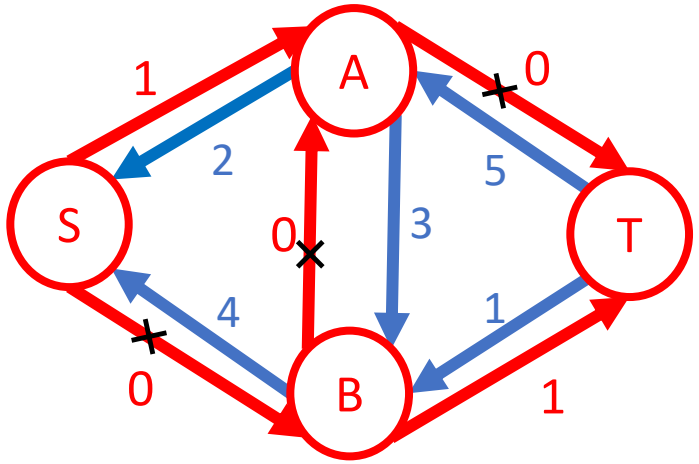




- Novo Algoritmo:
  - Inicia fluxo em todas as arestas com zero
  - Cria grafo auxiliar $\mathbf{G^R}$ com pesos $\mathbf{c^R((u, v)) = c((u, v))}$
  - Enquanto existir um caminho **P** de **S** até **T** em **G**
    - Seja **x** menor peso de uma aresta em **P**
    - Para todo **(u,v)** em **P**
      - $\mathbf{c^R((u, v)) \mathrel{-}= x}$
      - $\mathbf{c^R((v, u)) \mathrel{+}= x}$
      - $\mathbf{f((u, v)) = c((u, v)) - c^R((u, v))}$

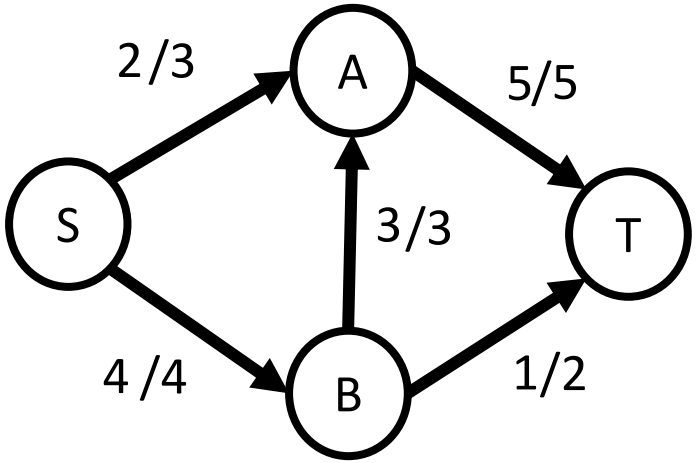
Algoritmo para encontrar fluxo máximo

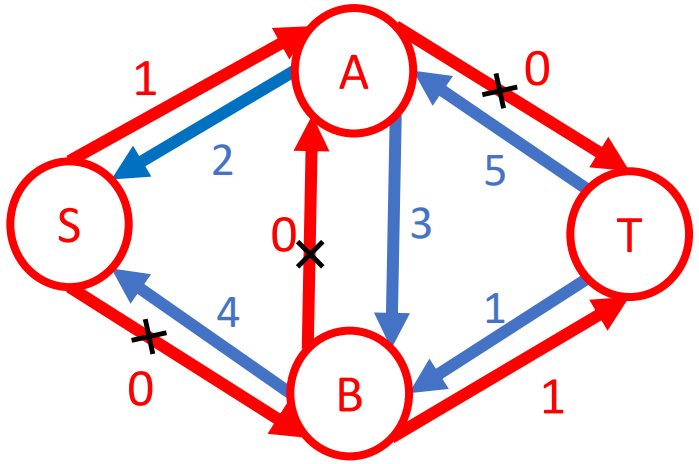




- Novo Algoritmo:
  - Inicia fluxo em todas as arestas com zero
  - Cria grafo auxiliar $\mathbf{G^R}$ com pesos $\mathbf{c^R((u, v)) = c((u, v))}$
  - Enquanto existir um caminho **P** de **S** até **T** em **G**
    - Seja **x** menor peso de uma aresta em **P**
    - Para todo **(u,v)** em **P**
      - $\mathbf{c^R((u, v)) \mathrel{-}= x}$
      - $\mathbf{c^R((v, u)) \mathrel{+}= x}$
      - $\mathbf{f((u, v)) = c((u, v)) - c^R((u, v))}$

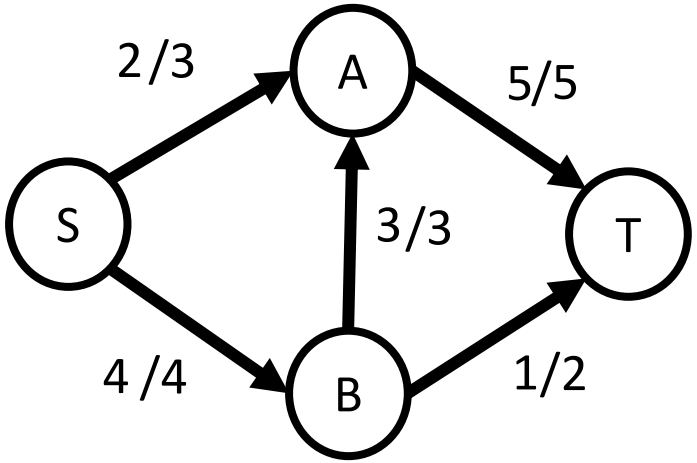
Algoritmo para encontrar fluxo máximo

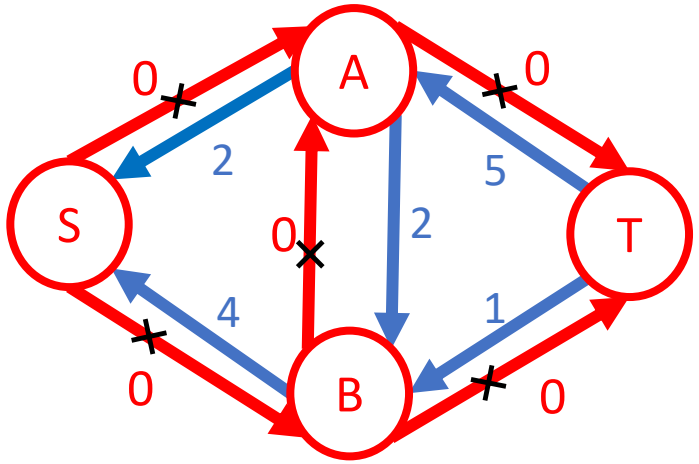




- Novo Algoritmo:
  - Inicia fluxo em todas as arestas com zero
  - Cria grafo auxiliar $\mathbf{G^R}$ com pesos $\mathbf{c^R((u, v)) = c((u, v))}$
  - Enquanto existir um caminho **P** de **S** até **T** em **G**
    - Seja **x** menor peso de uma aresta em **P**
    - Para todo **(u,v)** em **P**
      - $\mathbf{c^R((u, v)) \mathrel{-}= x}$
      - $\mathbf{c^R((v, u)) \mathrel{+}= x}$
      - $\mathbf{f((u, v)) = c((u, v)) - c^R((u, v))}$

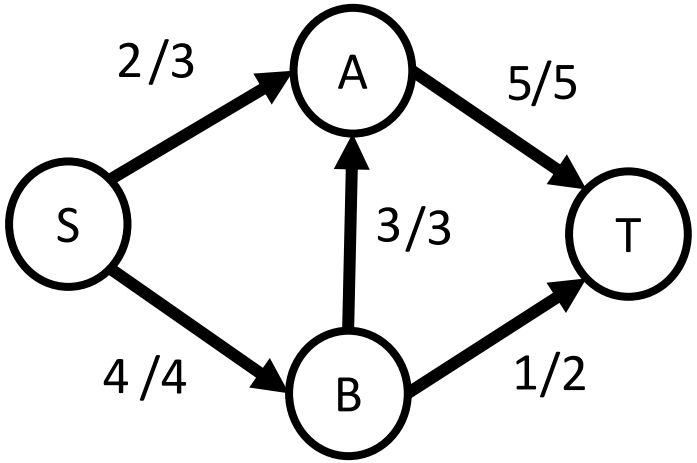
Algoritmo para encontrar fluxo máximo

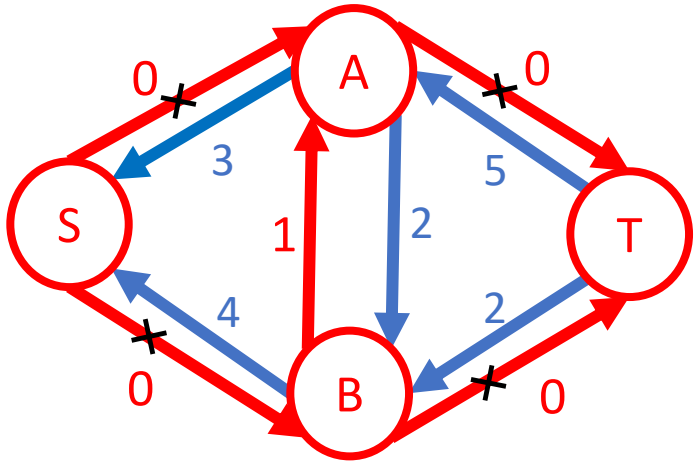




- Novo Algoritmo:
  - Inicia fluxo em todas as arestas com zero
  - Cria grafo auxiliar $\mathbf{G^R}$ com pesos $\mathbf{c^R((u, v)) = c((u, v))}$
  - Enquanto existir um caminho **P** de **S** até **T** em **G**
    - Seja **x** menor peso de uma aresta em **P**
    - Para todo **(u,v)** em **P**
      - $\mathbf{c^R((u, v)) \mathrel{-}= x}$
      - $\mathbf{c^R((v, u)) \mathrel{+}= x}$
      - $\mathbf{f((u, v)) = c((u, v)) - c^R((u, v))}$

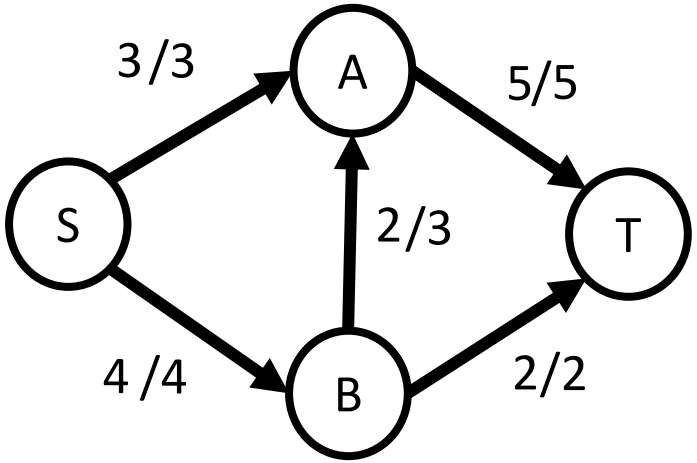
Algoritmo para encontrar fluxo máximo

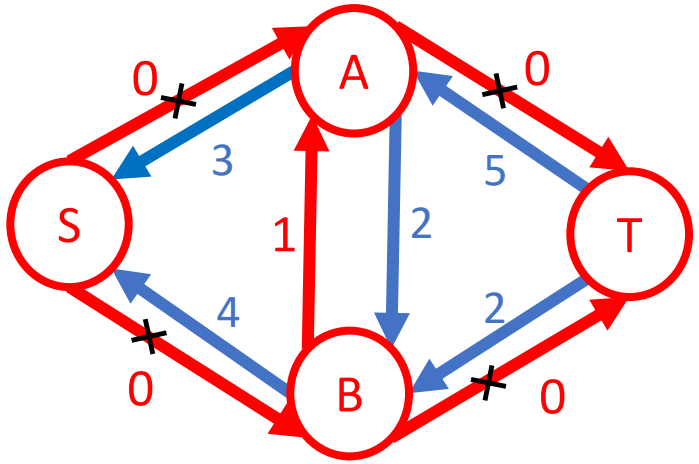




- Novo Algoritmo:
  - Inicia fluxo em todas as arestas com zero
  - Cria grafo auxiliar $\mathbf{G^R}$ com pesos $\mathbf{c^R((u, v)) = c((u, v))}$
  - Enquanto existir um caminho **P** de **S** até **T** em **G**
    - Seja **x** menor peso de uma aresta em **P**
    - Para todo **(u,v)** em **P**
      - $\mathbf{c^R((u, v)) \mathrel{-}= x}$
      - $\mathbf{c^R((v, u)) \mathrel{+}= x}$
      - $\mathbf{f((u, v)) = c((u, v)) - c^R((u, v))}$

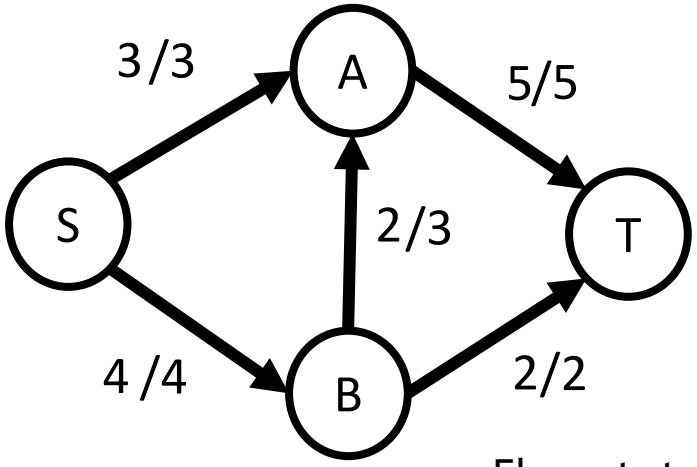
Algoritmo para encontrar fluxo máximo



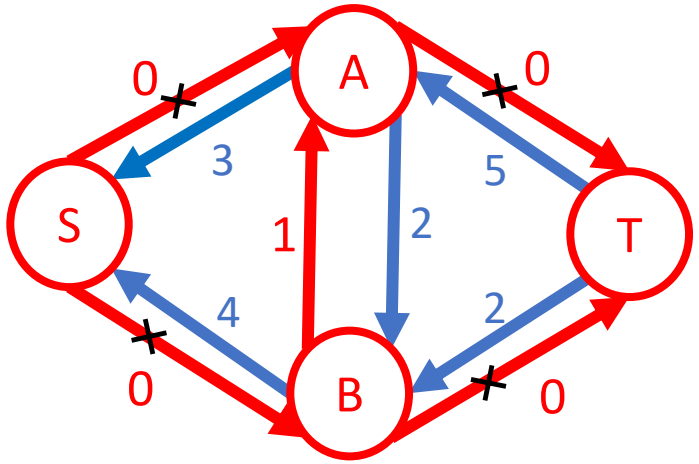


- Novo Algoritmo:
  - Inicia fluxo em todas as arestas com zero
  - Cria grafo auxiliar $\mathbf{G^R}$ com pesos $\mathbf{c^R((u, v)) = c((u, v))}$
  - Enquanto existir um caminho **P** de **S** até **T** em **G**
    - Seja **x** menor peso de uma aresta em **P**
    - Para todo **(u,v)** em **P**
      - $\mathbf{c^R((u, v)) \mathrel{-}= x}$
      - $\mathbf{c^R((v, u)) \mathrel{+}= x}$
      - $\mathbf{f((u, v)) = c((u, v)) - c^R((u, v))}$

Algoritmo para encontrar fluxo máximo



Fluxo total = 7



Algoritmo de Ford-Fulkerson

- Novo Algoritmo:
  - Inicia fluxo em todas as arestas com zero
  - Cria grafo auxiliar $\mathbf{G^R}$ com pesos $\mathbf{c^R((u, v)) = c((u, v))}$
  - Enquanto existir um caminho **P** de **S** até **T** em **G**
    - Seja **x** menor peso de uma aresta em **P**
    - Para todo **(u,v)** em **P**
      - $\mathbf{c^R((u, v)) \text{ -= } x}$
      - $\mathbf{c^R((v, u)) \text{ += } x}$
      - $\mathbf{f((u, v)) = c((u, v)) - c^R((u, v))}$

Complexidade: O(|E|F)

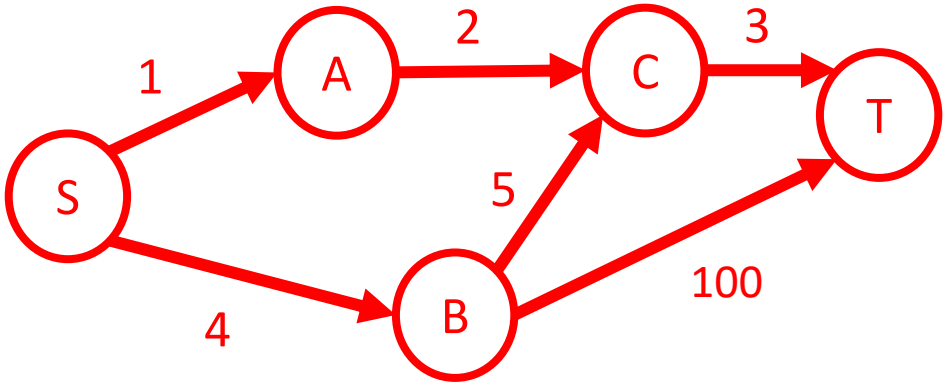
BFS ou DFS O(|E|)

Pelo menos +1 de Fluxo a cada iteração → O(F) iterações

Algoritmo para encontrar fluxo máximo

Algoritmo de Edmonds-Karp

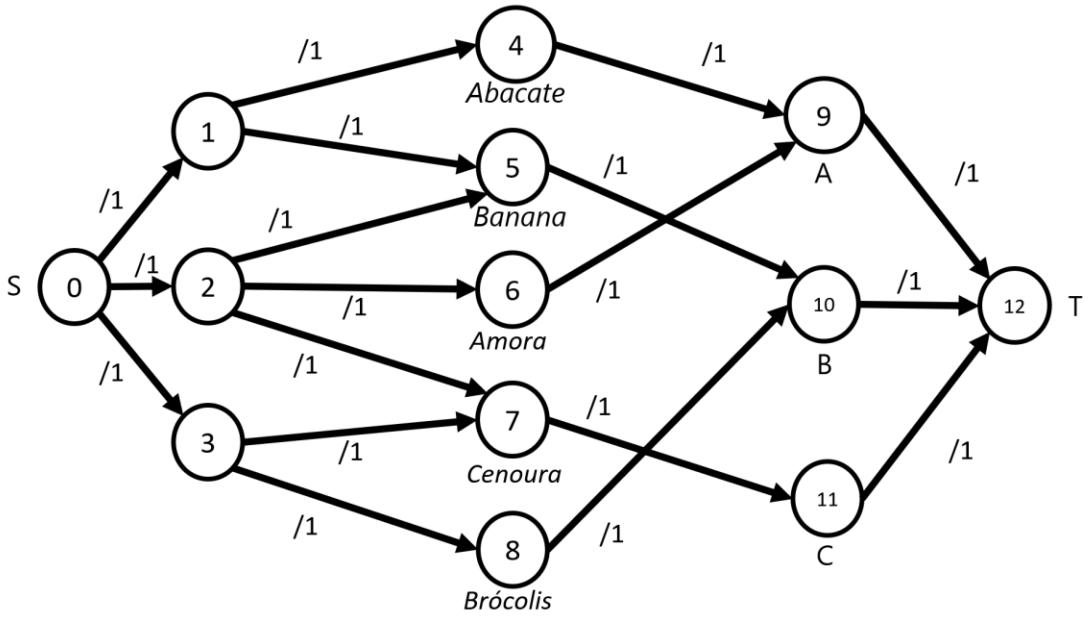
- **Ford-Fulkerson** usando uma busca em largura (BFS) para encontrar o caminho mínimo em G^R entre **S** e **T** a cada iteração.
- Caminho mínimo em número de arestas, não em soma dos pesos!
- Complexidade: $O(E^2 V)$



Algoritmo para encontrar fluxo máximo

Algoritmo de Dinic

- Também usa o conceito de grafo residual
- Complexidade:
  - **O(E V^2)** no caso geral
  - **O(E $\sqrt{V}$)** para redes nas quais:
    - Todas as capacidades são 1
    - Com exceção de **S** e **T**, todos os vértices tem grau de entrada ou saída 1.



Exemplo

URI 2354 – Kill the Werewolf

- **N** (<= 50) jogadores, um deles é secretamente é um lobisomem
- Duas fases:
  - Na primeira cada jogador escolhe dois alvos (outros jogadores)
  - Depois da primeira o lobisomem revela sua identidade
  - Nas segunda cada jogador escolhe um dos dois alvos que definiu na primeira. O alvo que tiver mais votos “morre”.
  - Lobisomem é o último a votar e vence se não for o jogador com mais votos (vence em empates)
- Entrada: Alvos de cada um dos N jogadores
- Saída: Quantos jogadores venceriam se fossem o lobisomem e todos jogassem de forma ótima

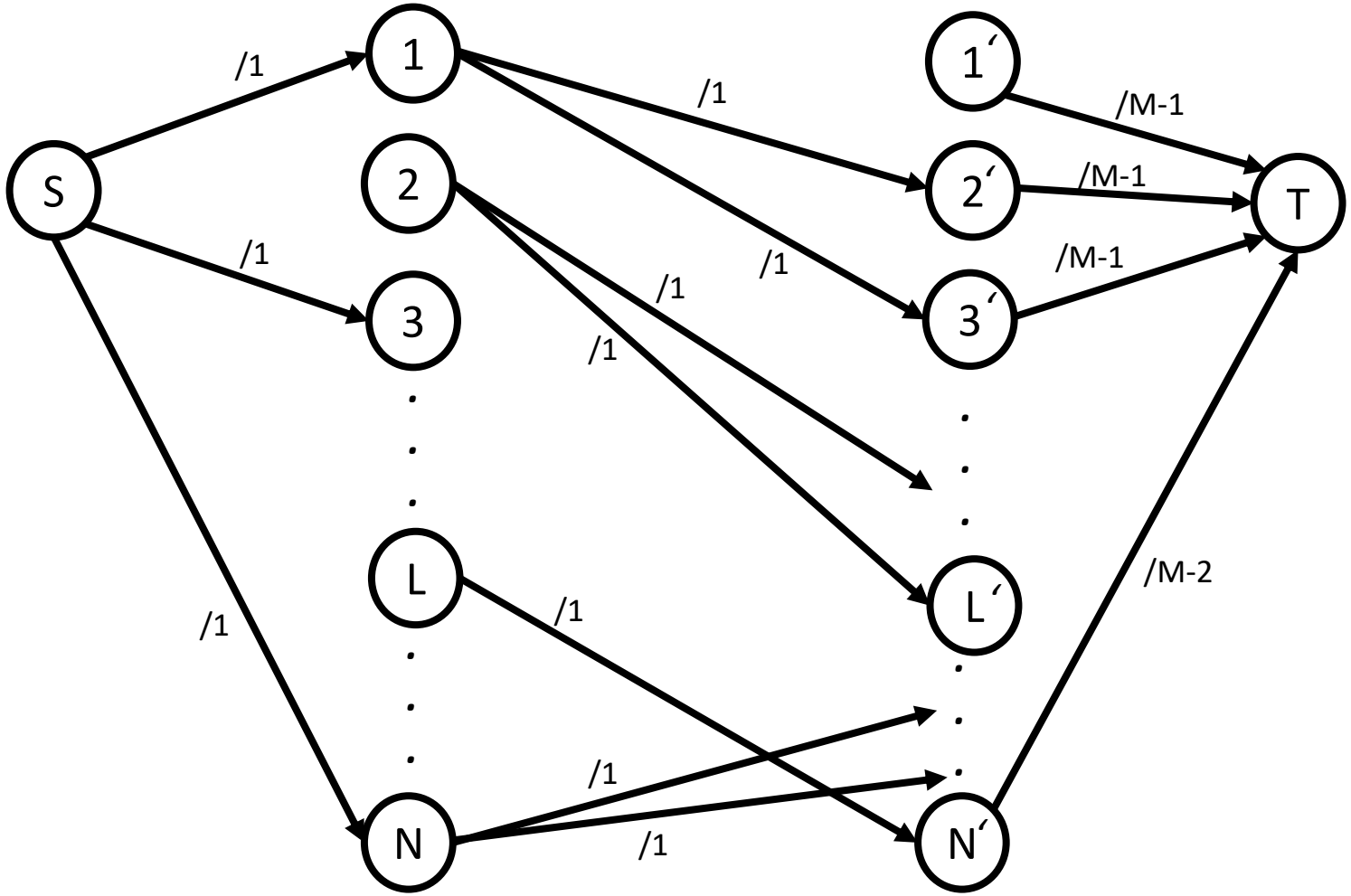
Ex:

N = 4
1 -> 3, 4
2 -> 1, 4
3 -> 4, 1
4 -> 3, 1

2 Jogadores vencem como lobisomem: 2 e 3

Exemplo

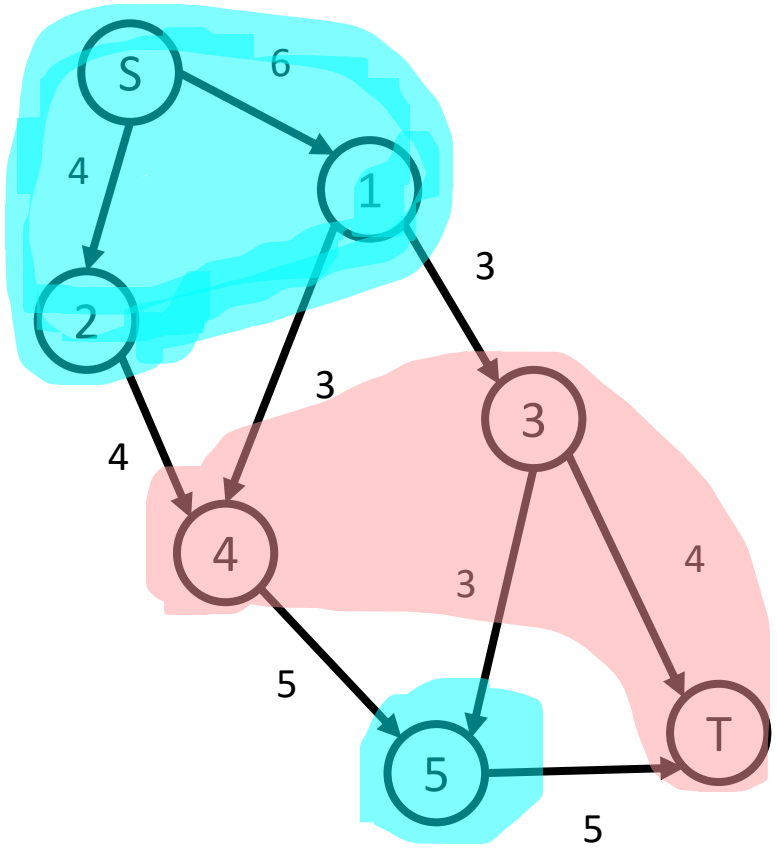
- *Resolver para cada jogador sendo o lobisomem.*
- *Todos os **M** jogadores que escolheram o lobisomem como alvo votam nele.*
- *Se **k** tem **a** e **b** como alvos, **k** liga em **a'** e **b'** com cap. 1.*
- ***S** liga em **k** com cap. 1 se:*
  - ***k**-ésimo jogador não é o lobisomem*
  - ***k**-ésimo jogador não tem lobisomem como alvo*
- Se **k´** não é alvo do lobisomem, liga em **T** com cap. **M-1**
- Caso contrário, liga em **T** com **M-2**
- Lobisomem ganha se o fluxo total < **N-M-1**



Aplicação: Corte Mínimo

- Problema do corte mínimo
  - Entrada: um grafo orientado **G = {V, E}** e dois vértices **S**, **T**.
  - Um corte é definido por **A** e **B**:
    - **A, B ⊂ V**
    - **A ∩ B = Ø**
    - **A U B = V**
    - **S ϵ A**
    - **T ϵ B**
  - Seja o valor de um corte a soma dos pesos das arestas que saem de vértices em **A** e vão para vértices em **B**.
  - Queremos achar o corte com menor valor.
- Teorema do fluxo máximo e corte mínimo:

Fluxo Total Máximo = Valor do Corte Mínimo



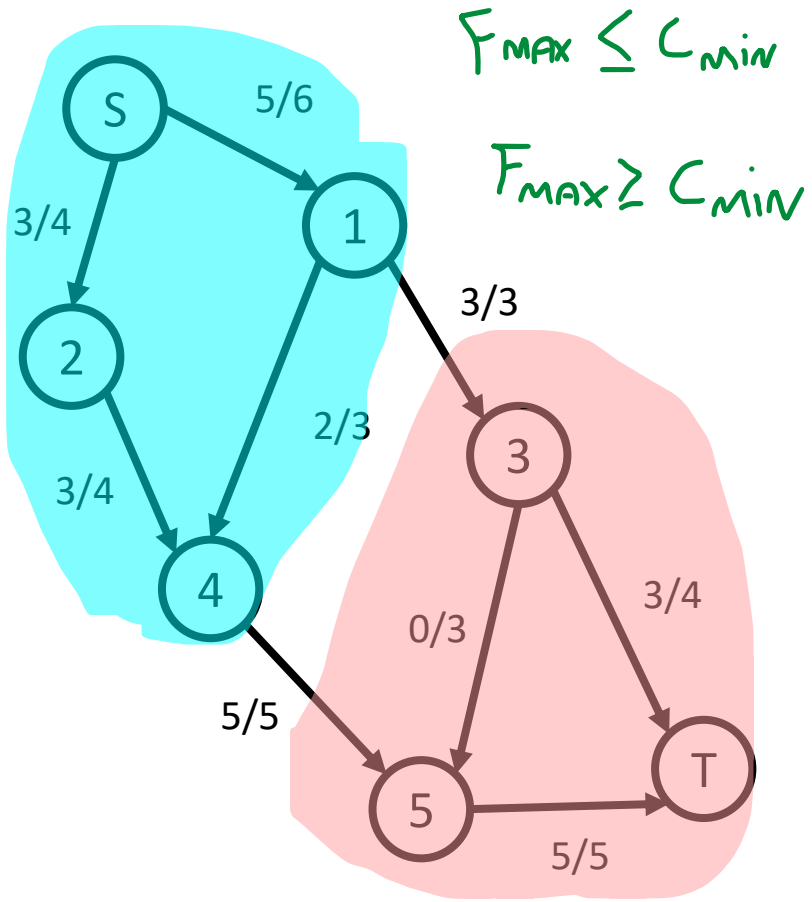
A = {S, 1, 2, 5}
B = {3, 4, T}
Valor do Corte: 4 + 3 + 3 + 5 = 15

Aplicação: Corte Mínimo

- Problema do corte mínimo
  - Entrada: um grafo orientado **G = {V, E}** e dois vértices **S**, **T**.
  - Um corte é definido por **A** e **B**:
    - **A, B ⊂ V**
    - **A ∩ B = Ø**
    - **A U B = V**
    - **S ϵ A**
    - **T ϵ B**
  - Seja o valor de um corte a soma dos pesos das arestas que saem de vértices em **A** e vão para vértices em **B**.
  - Queremos achar o corte com menor valor.
- Teorema do fluxo máximo e corte mínimo:

Fluxo Total Máximo = Valor do Corte Mínimo

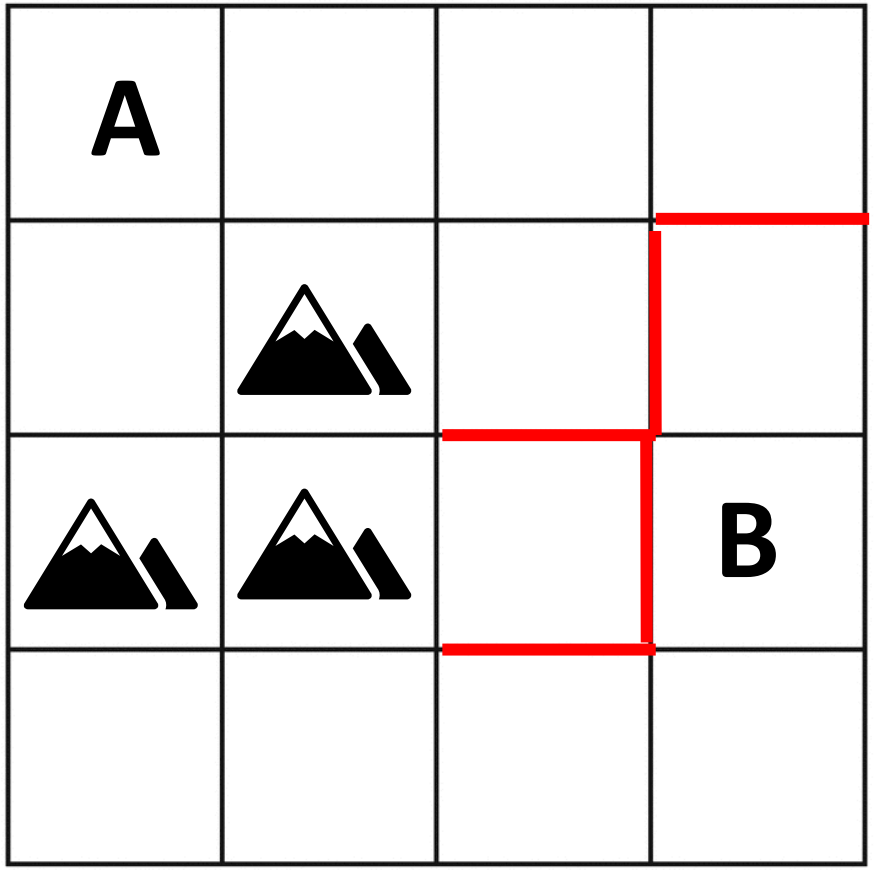
- Para encontrar **A**: todos os vértices aos quais se pode chegar de **A** sem usar arestas com a capacidade completamente ocupada



A = {S, 1, 2, 4}
B = {3, 5, T}
Valor do Corte: 5 + 3 = 8

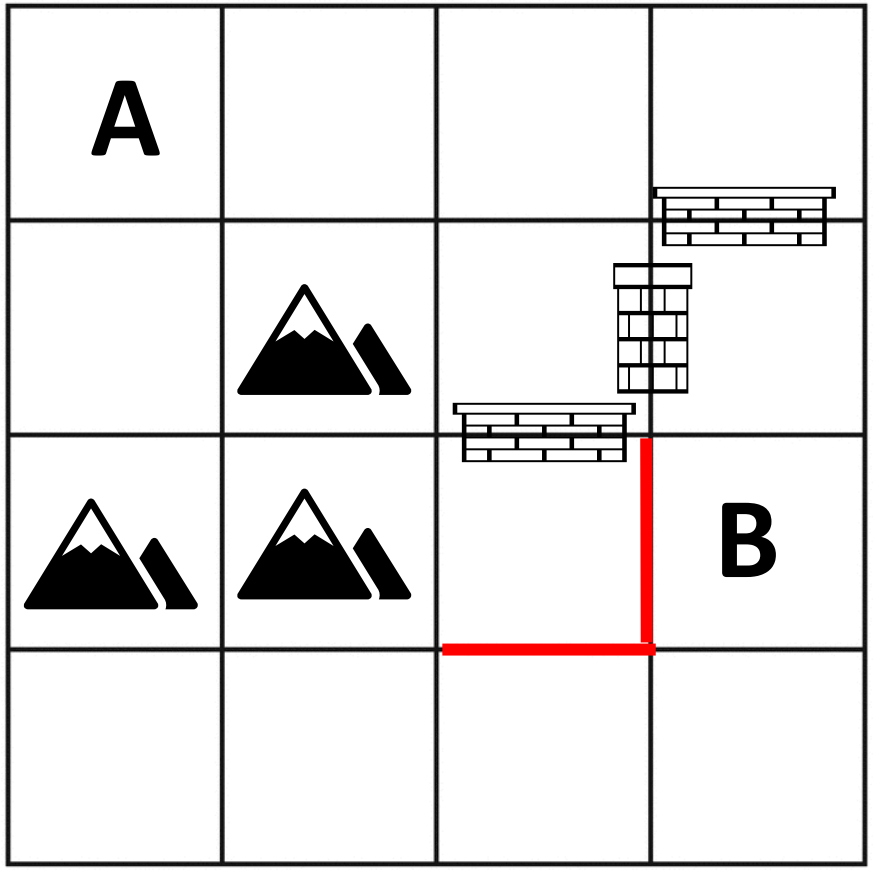
Exemplo

- Território retangular **N x M**
- Duas posições distintas são as cidades **A** e **B**
- É possível se mover entre duas posições com uma aresta em comum
- Algumas posições são montanhas, não é possível passar por elas
- É possível construir muros em adjacências especificas



Exemplo

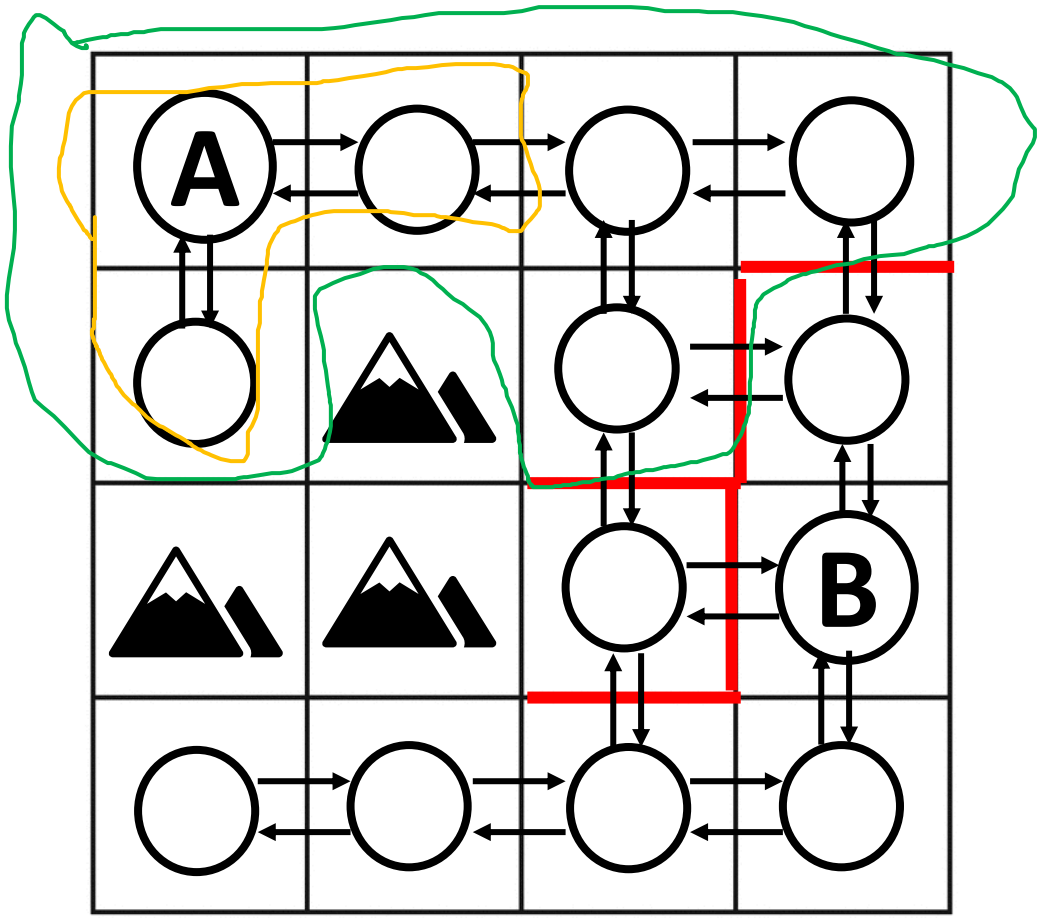
- Território retangular **N x M**
- Duas posições distintas são as cidades **A** e **B**
- É possível se mover entre duas posições com uma aresta em comum
- Algumas posições são montanhas, não é possível passar por elas
- É possível construir muros em adjacências especificas
- Qual o menor número de adjacências nos quais teríamos que construir muros para que não seja possível ir de uma cidade à outra?



Exemplo

- Cada posição que não as montanhas é um vértice, com A e B sendo a fonte e o dreno.
- Arestas nos dois sentidos em cada adjacência.
  - Adjacências onde pode se construir muros: capacidade = 1
  - Adjacências onde não se pode construir muros: capacidade = ∞
- Solução: Corte mínimo = Número mínimo de muros

Obs: cuidado com overflow em casos que não existe solução!



Exemplo

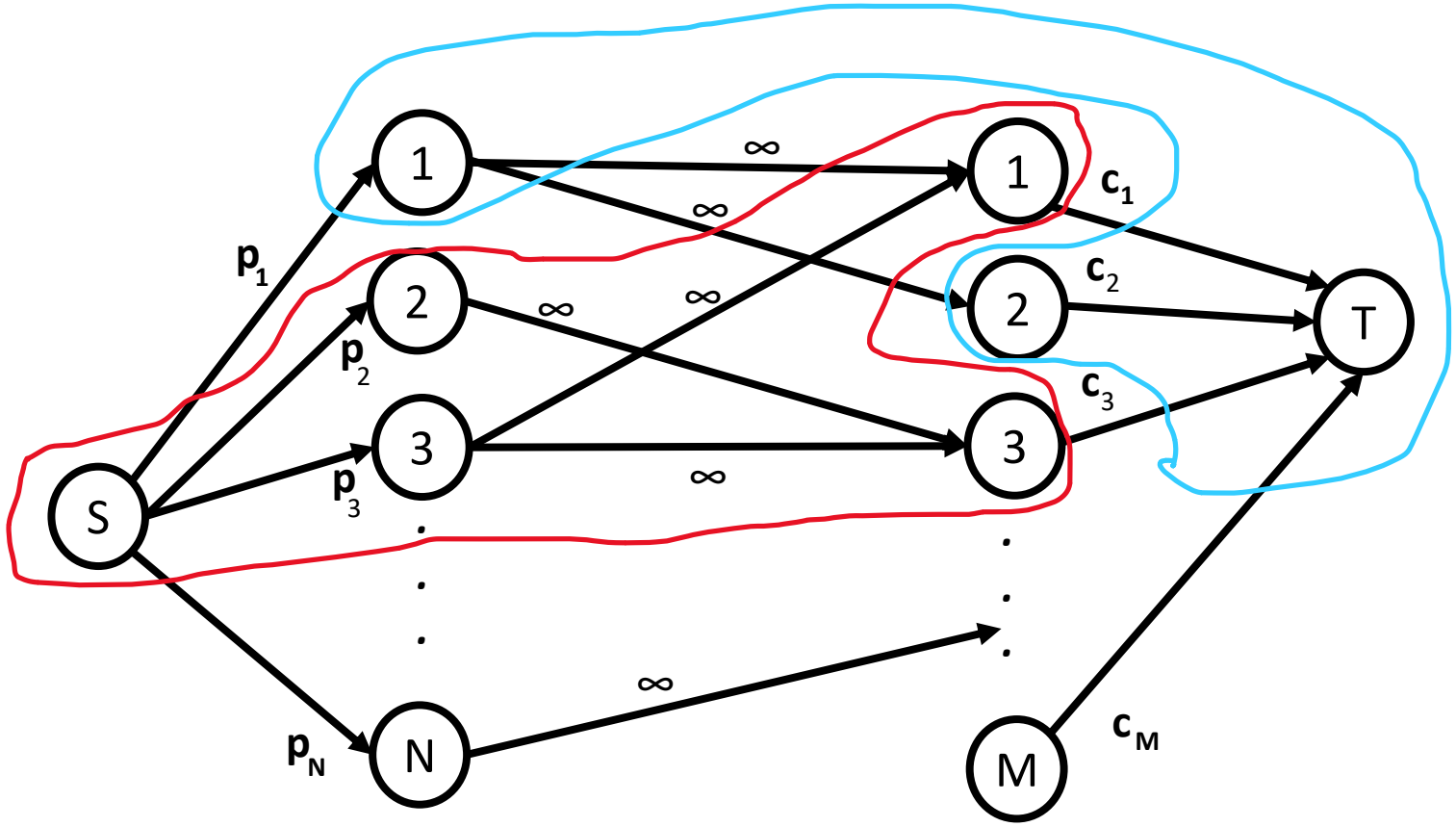
- Uma fábrica pode aceitar **N** serviços, cada um com pagamento $\mathbf{p_i}$
- Existem **M** máquinas que podem ser compradas
- Para fazer o **i**-ésimo serviço a fábrica precisa comprar $\mathbf{k_i}$ máquinas $\mathbf{m_{i1}}$, $\mathbf{m_{i2}}$, ..., $\mathbf{m_{ik_i}}$
- Cada máquina tem um custo $\mathbf{c_i}$, e só precisa ser comprada uma vez (pode ser compartilhada entre diferentes serviços).
- Escolhendo os serviços de forma ótima, qual o maior lucro que a fábrica poder ter?

Restrições:

- $1 \leq \mathbf{N}, \mathbf{M} \leq 100$
- $1 \leq \mathbf{p_i}, \mathbf{c_i} \leq 5000$
- $1 \leq \mathbf{k_i} \leq \mathbf{M}$

Exemplo

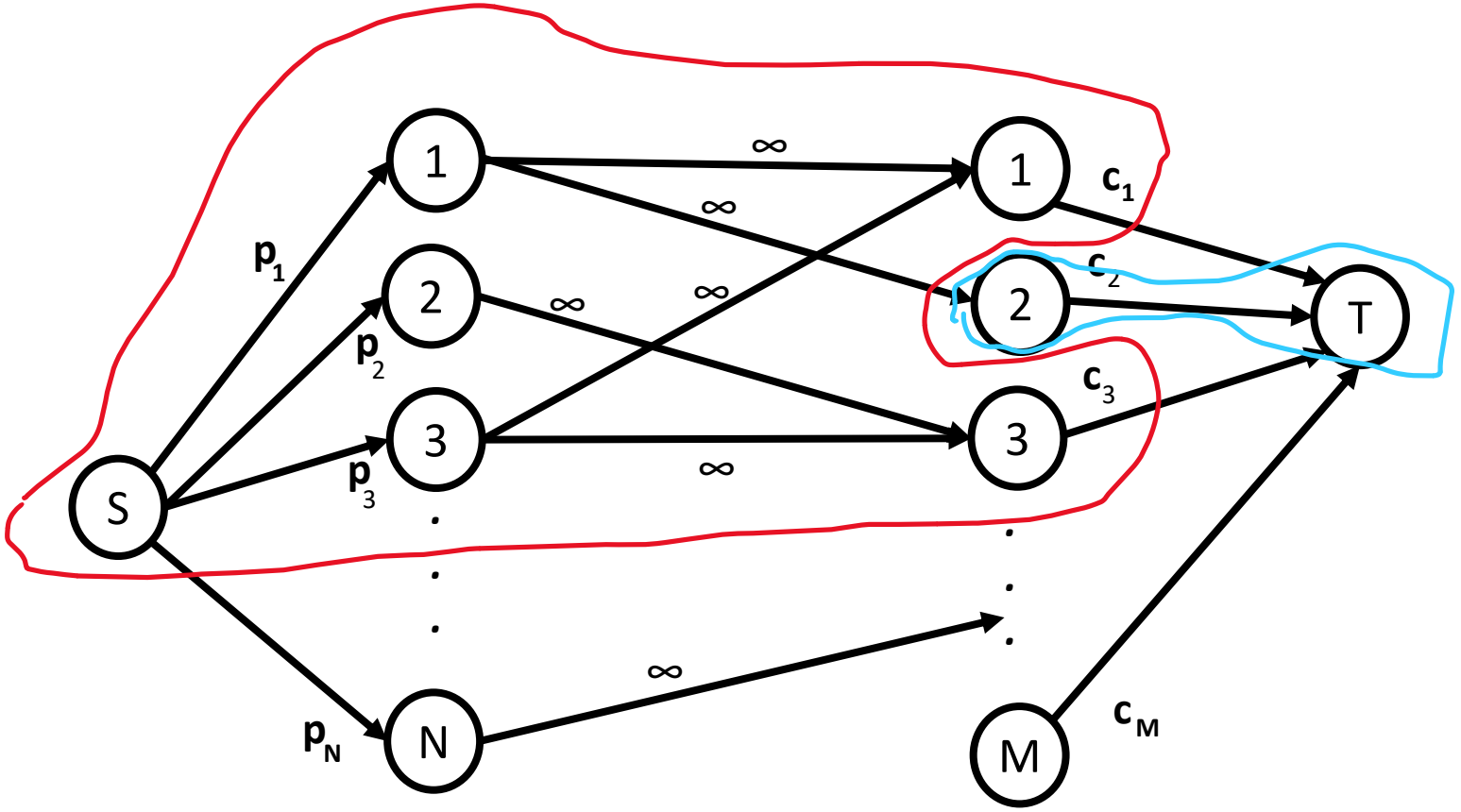
- Vértices
  - Vértices para os serviços e máquinas
  - Lado da fonte: serviços escolhidos
  - Lado do dreno: serviços não escolhidos
- Arestas
  - Entre a fonte e os serviços: capacidade é o pagamento do serviço
  - De cada máquina ao dreno: capacidade é o custo da máquina
  - Entre cada serviço e as máquinas necessárias: capacidade infinita
- Seja **P** a soma de todos os pagamentos possíveis
- Solução: **P** – Corte Mínimo



1 -> 1, 2
2 -> 3
3 -> 1, 3

Exemplo

- Vértices
  - Vértices para os serviços e máquinas
  - Lado da fonte: serviços escolhidos
  - Lado do dreno: serviços não escolhidos
- Arestas
  - Entre a fonte e os serviços: capacidade é o pagamento do serviço
  - De cada máquina ao dreno: capacidade é o custo da máquina
  - Entre cada serviço e as máquinas necessárias: capacidade infinita
- Seja **P** a soma de todos os pagamentos possíveis
- Solução: **P** – Corte Mínimo



1 -> 1, 2
2 -> 3
3 -> 1, 3

Referências

Explicação + Implementação

- **CP Algorithms:** Maximum flow - Ford-Fulkerson and Edmonds-Karp

  https://cp-algorithms.com/graph/edmonds_karp.html

  https://bit.ly/357js9f

- **CP Algorithms:** Maximum flow - Dinic's algorithm

  https://cp-algorithms.com/graph/dinic.html

  https://bit.ly/3nPVLZw

Demonstrações

- **Simon Fraser University:** The Ford-Fulkerson Method

  https://coursys.sfu.ca/2020fa-cmpt-307-d1/pages/slides23/view

  https://bit.ly/3fP22jL

- **Brilliant**: Edmonds-Karp Algorithm Complexity Proof

  https://brilliant.org/wiki/edmonds-karp-algorithm/#complexity-proof

  https://bit.ly/3nN9qAA

Referências

Problemas para treino

- **SPOJ:**

  https://www.spoj.com/problems/POTHOLE/

  https://bit.ly/3FNrOQ1

- **Maratona Unicamp:** Idade dos Impérios

  https://codeforces.com/group/DZKrfFzIDL/contest/298591/problem/I

  https://bit.ly/3AjVIKI

- **SPOJ:** COCONUTS

  https://www.spoj.com/problems/COCONUTS/

  https://bit.ly/3KxMoYa

- **Online Judge:** Lista de problemas

  https://onlinejudge.org/index.php?option=com_onlinejudge&Itemid=8&category=565

  https://bit.ly/357s2EZ

Referências

Tópicos Avançados

- **CP Algorithms:** Push-relabel algorithm

  https://cp-algorithms.com/graph/push-relabel.html

  https://bit.ly/3KAih2k

- **CP Algorithms:** Maximum flow - Push-relabel method improved

  https://cp-algorithms.com/graph/push-relabel-faster.html

  https://bit.ly/3KBDyZq

- **CP Algorithms:** Flows with demands

  https://cp-algorithms.com/graph/flow_with_demands.html

  https://bit.ly/3Iwng2r

- **kactl**: GomoryHu

  https://github.com/kth-competitive-programming/kactl/blob/main/content/graph/GomoryHu.h

  https://bit.ly/35flVP9