Fundamentos de Algoritmos e Estrutura de Dados – Lecture 06 – Max Flow

Prof. André Gustavo Hochuli

gustavo.hochuli@pucpr.br
aghochuli@ppgia.pucpr.br

Plano de Aula

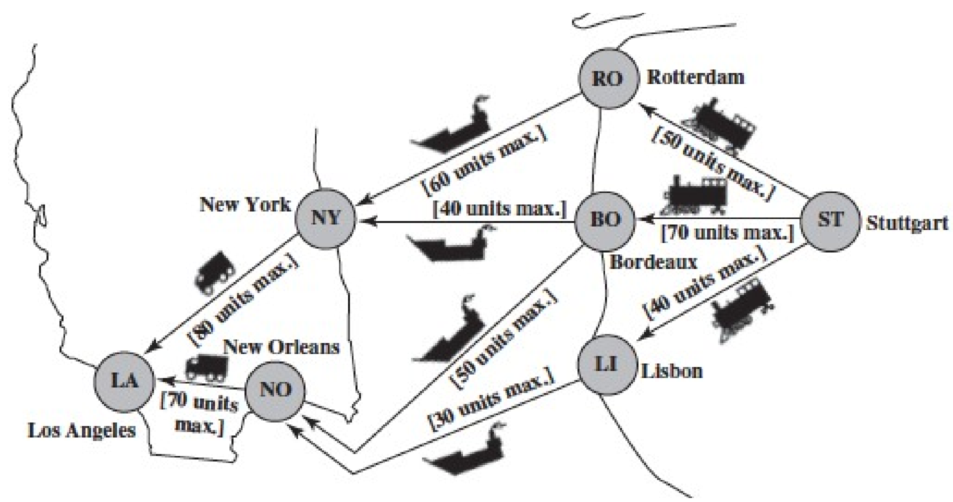
- **Lecture 05**
  - **Busca em Grafos**
  - **Caminho Mínimo**

- **Lecture 06**
  - **Fluxo Máximo: Ford-Fulkerson**
  - **Dinâmica de Grupos com Mentoria do Professor**

Problema de Fluxo Máximo
Algoritmo de Ford-Fulkerson

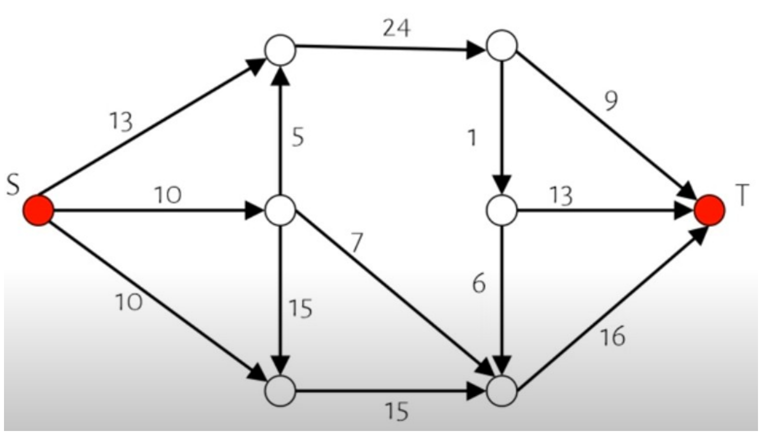
Fluxo Máximo

- Como otimizar a distribuição de recursos em redes (grafos) para maximizar o fluxo total entre origem e destino, considerando restrições de capacidade ?
  - Redes de tráfego (maximizar veÃculos que passam de um ponto a outro sem congestionamento).
  - Redes de dados (maximizar pacotes transmitidos entre servidores).
  - Logística de transporte (maximizar mercadorias distribuídas de um depósito a lojas).
  - Distribuição de Energia, etc.



Ford-Fulkerson

- **Busque um caminho da origem ao destino com capacidade disponível**
- **Determine o gargalo (capacidade mínima) do caminho.**
- **Atualizar rede residual**
- **Repetir ou finalizar:** Repita até não haver mais caminhos



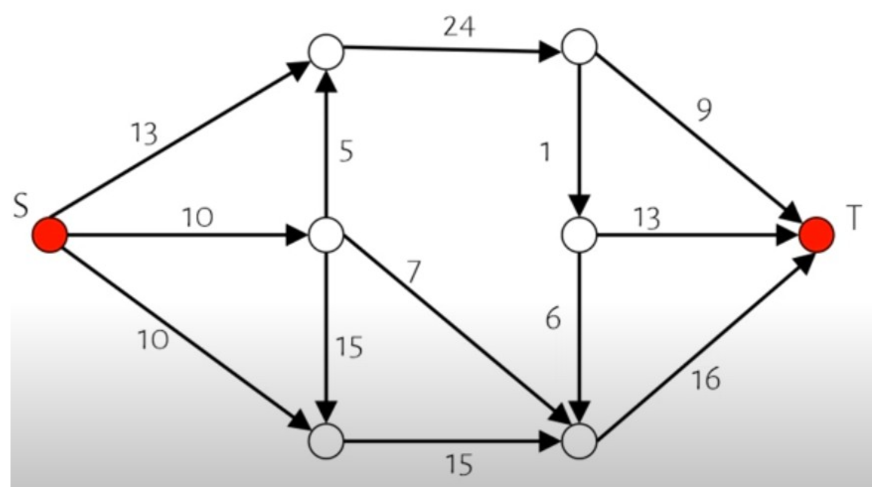
Agradecimento ao Igor pela elaboração do material de suporte!

https://www.youtube.com/watch?v=xC2tYIZvmgc

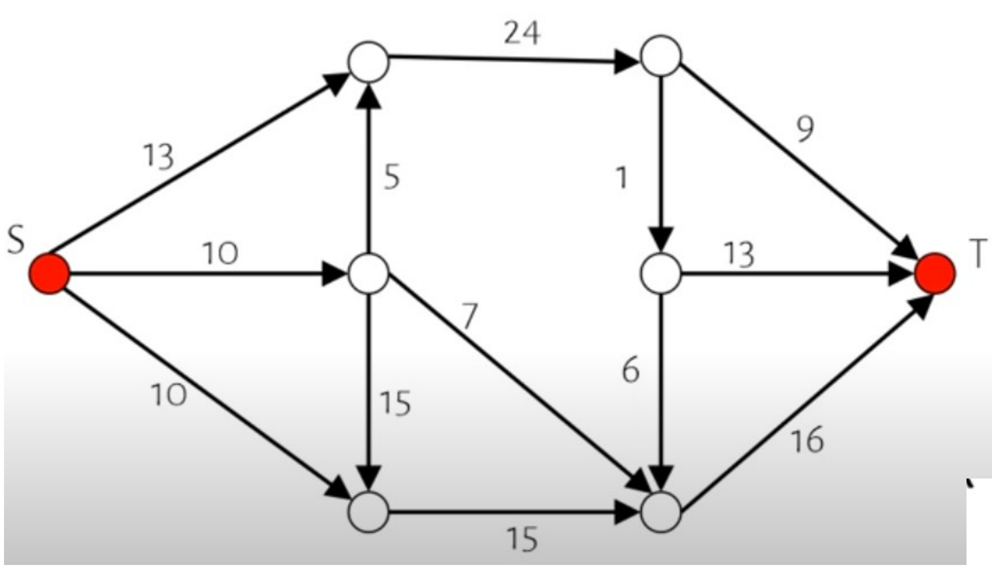
Ford-Fulkerson

Seja:
P={p1,p2,…,pk} o conjunto de caminhos aumentantes escolhidos pelo algoritmo
b(pi) o bottleneck (capacidade mínima residual) do caminho pip_ipi
Então o fluxo máximo total pode ser escrito como:

$$F_{\max} = \sum_{i=1}^{k} b(p_i)$$



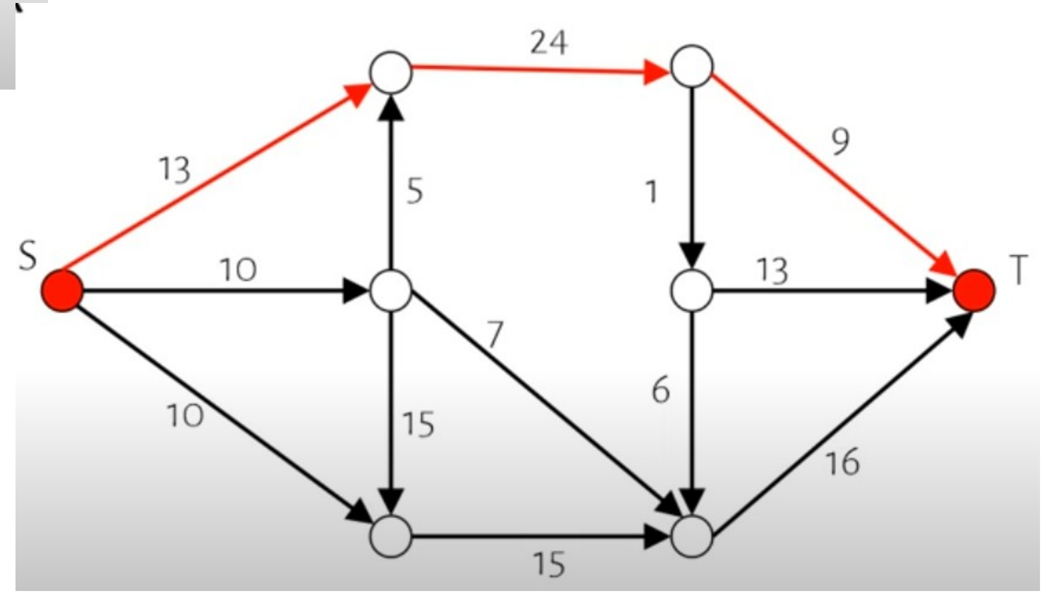
Ford-Fulkerson



INTERAÇÕES:
1 = 9

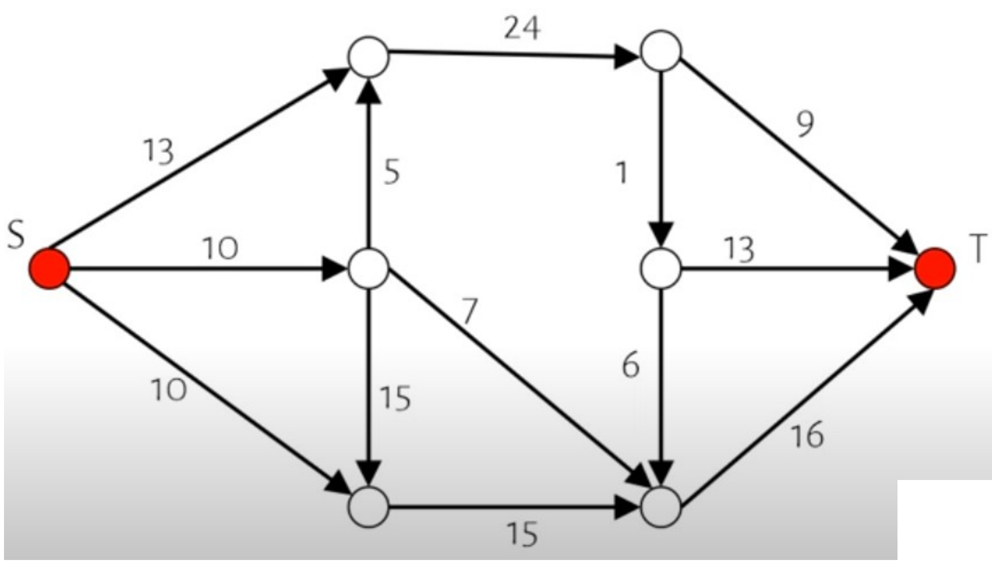
2. Determine o gargalo (capacidade mínima) do caminho.

Rede Residual



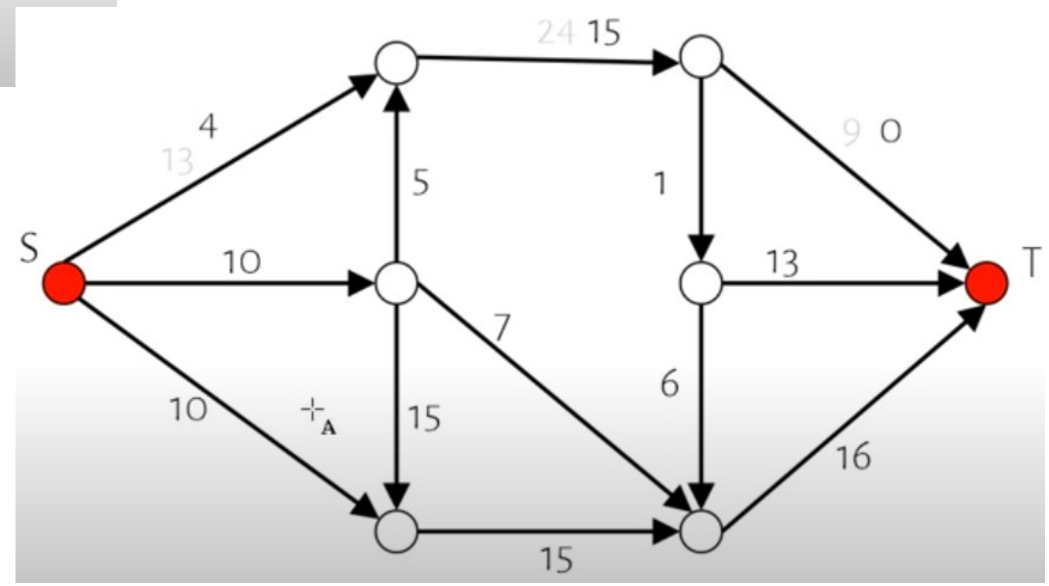
1. Busque um caminho da fonte ao destino com capacidade disponível.

Ford-Fulkerson

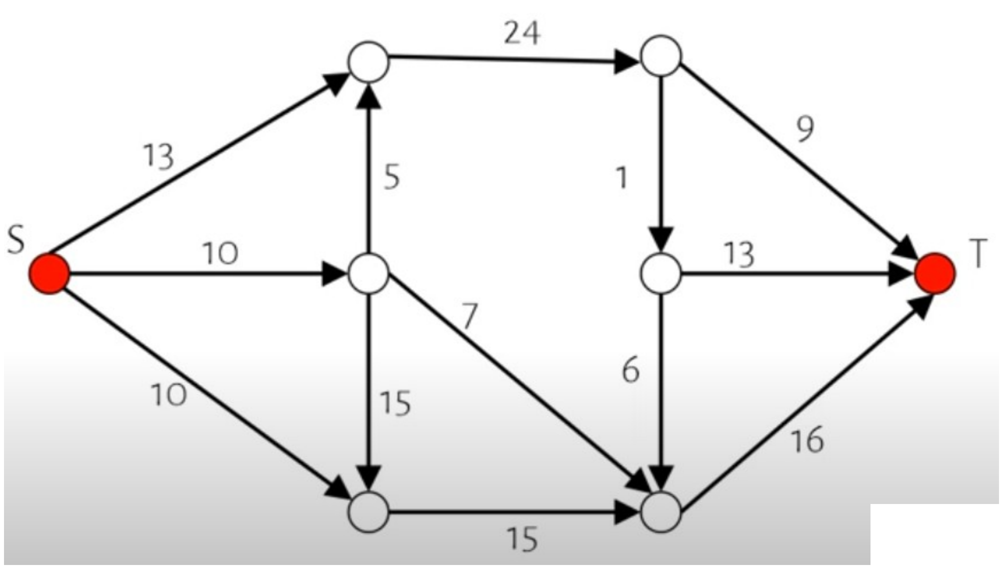


INTERAÇÕES:
1 = 9

Atualizar rede residual



Ford-Fulkerson



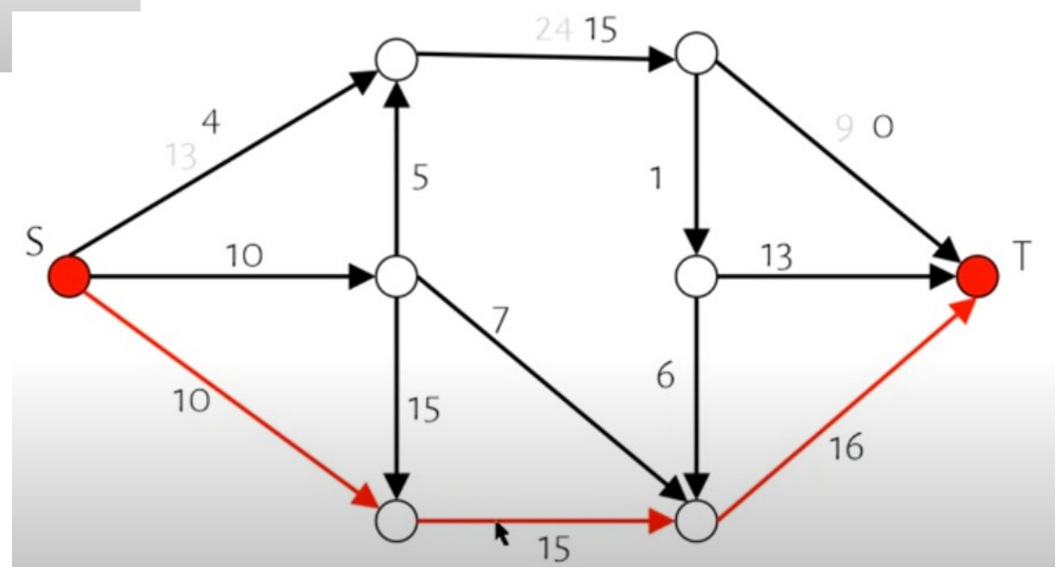
2. Determine o gargalo (capacidade mínima) do caminho.

INTERAÇÕES:

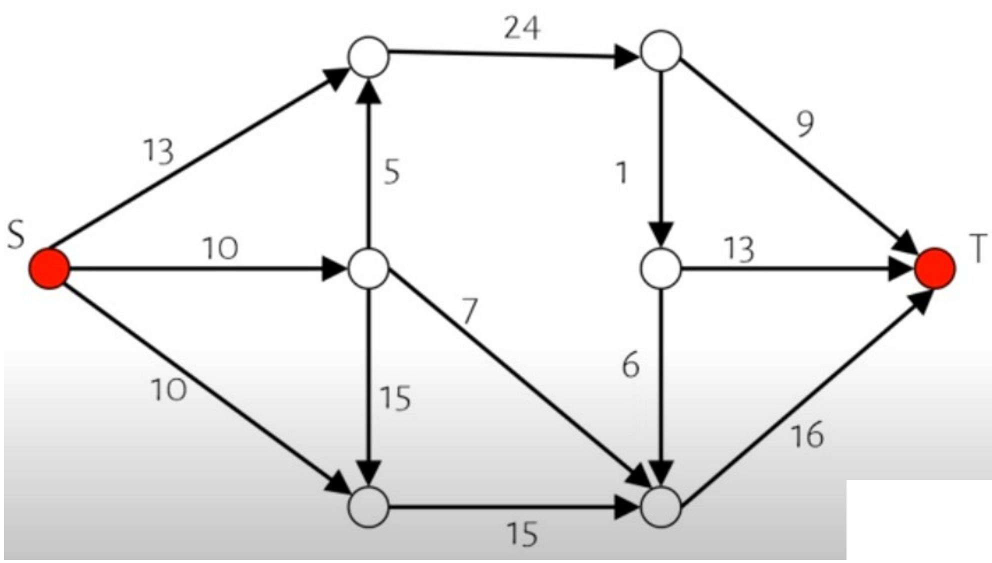
1 = 9

2 = 10

1. Busque um caminho da fonte ao destino com capacidade disponível.



Ford-Fulkerson

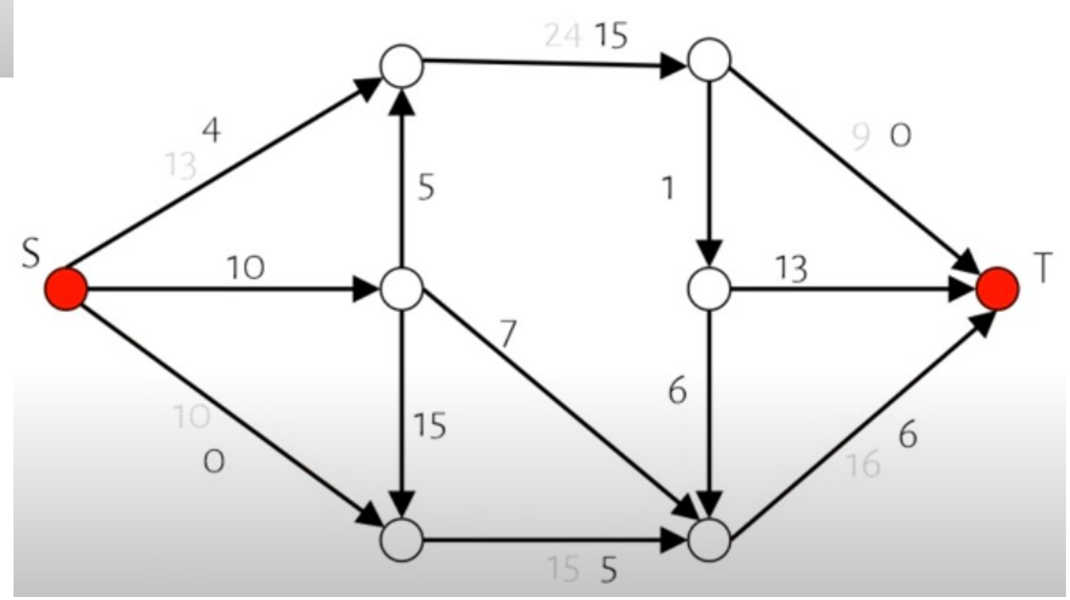


INTERAÇÕES:

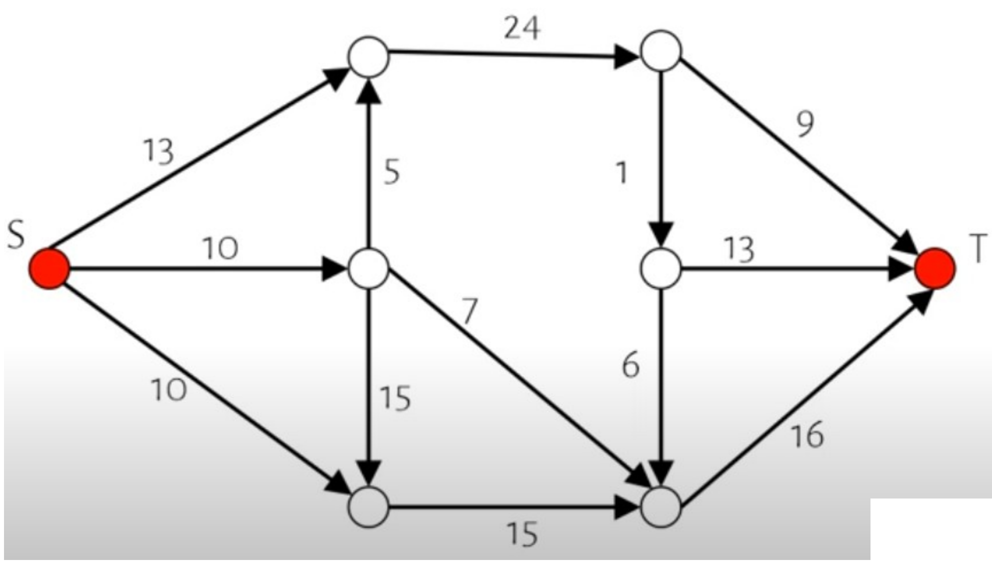
1 = 9

2 = 10

Atualizar rede residual



Ford-Fulkerson



2. Determine o gargalo (capacidade mínima) do caminho.

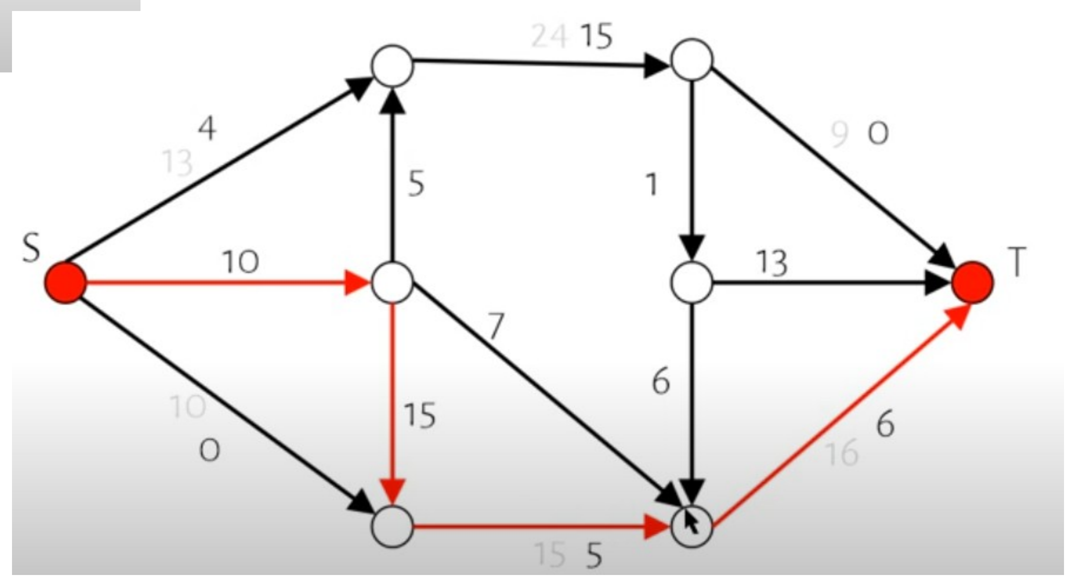
INTERAÇÕES:

1 = 9

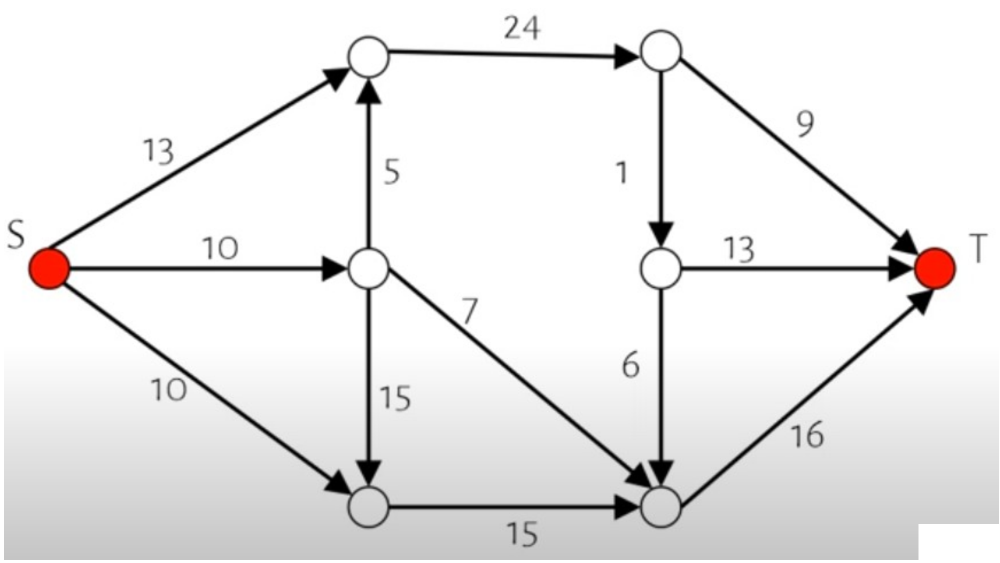
2 = 10

3 = 5

1. Busque um caminho da fonte ao destino com capacidade disponível.



Ford-Fulkerson



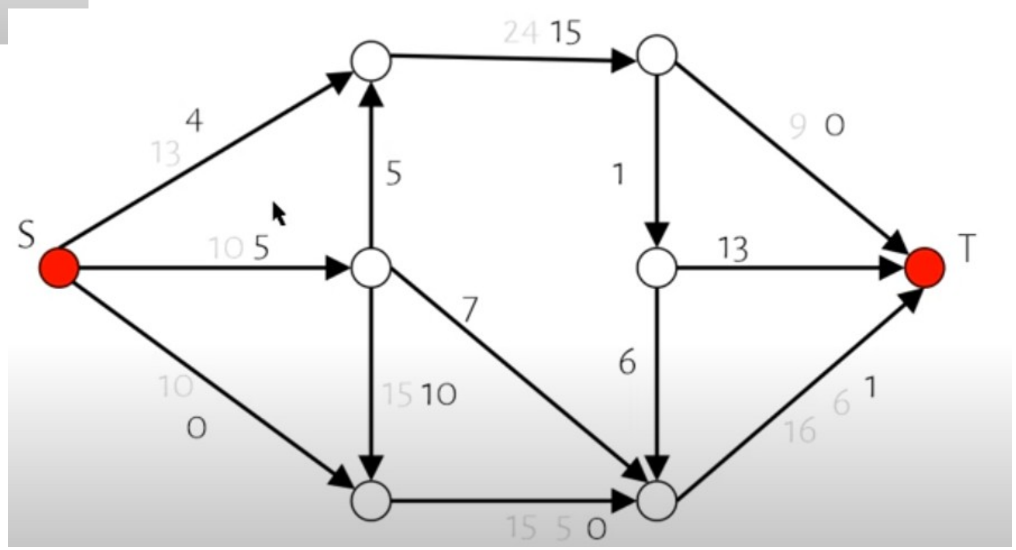
INTERAÇÕES:

1 = 9

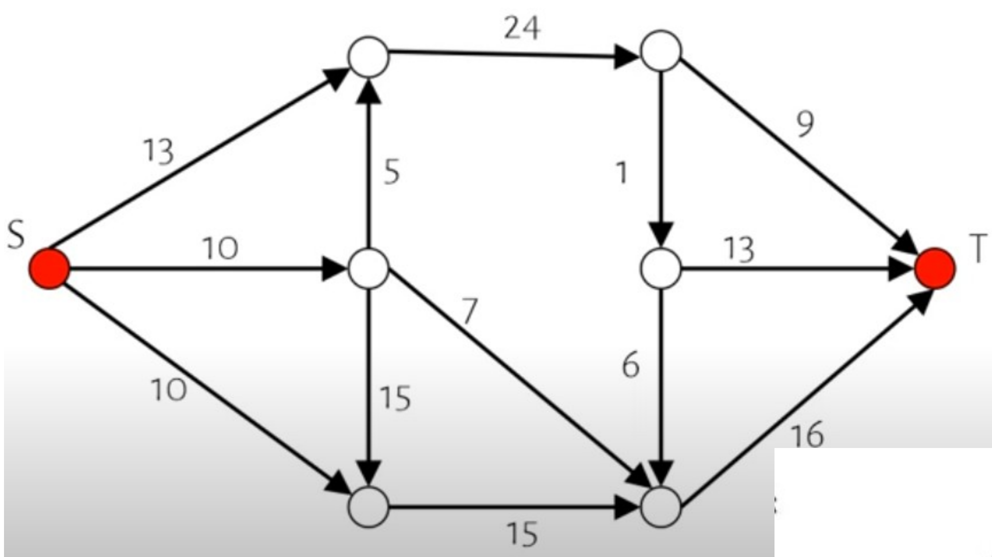
2 = 10

3 = 5

Atualizar rede residual



Ford-Fulkerson



2. Determine o gargalo (capacidade mínima) do caminho.

INTERAÇÕES:

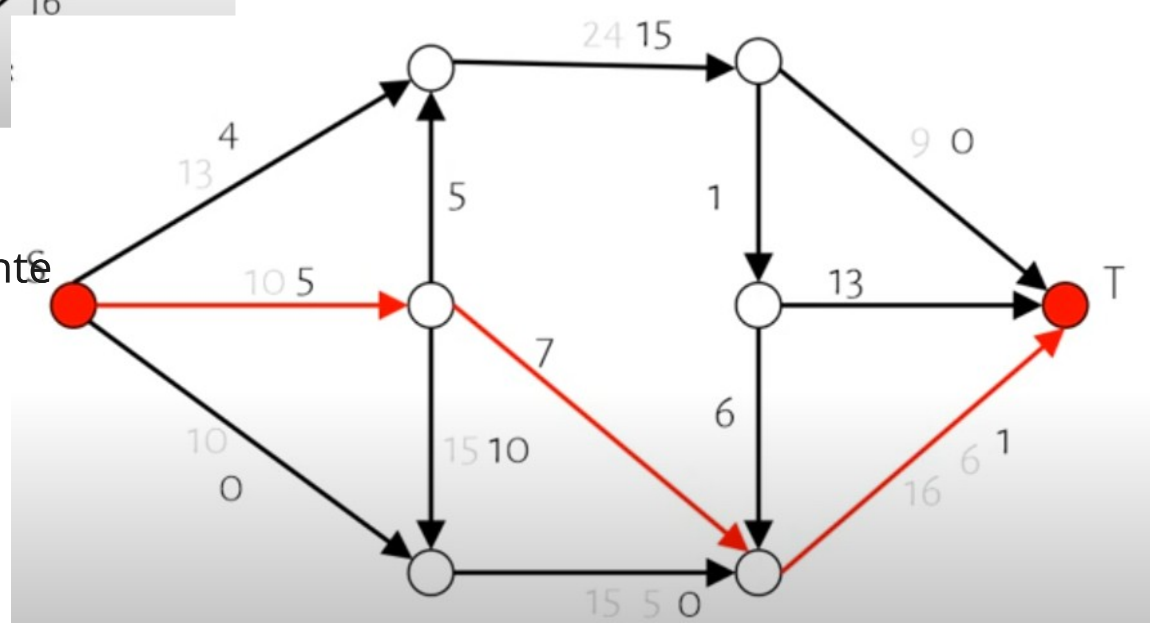
1 = 9

2 = 10

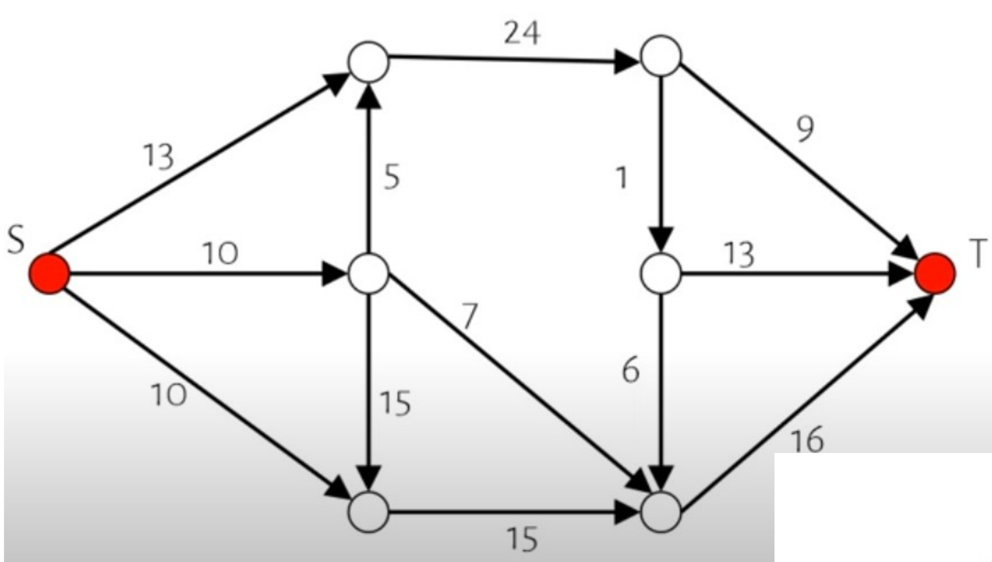
3 = 5

4 = 1

1. Busque um caminho da fonte ao destino com capacidade disponível.



Ford-Fulkerson



INTERAÇÕES:

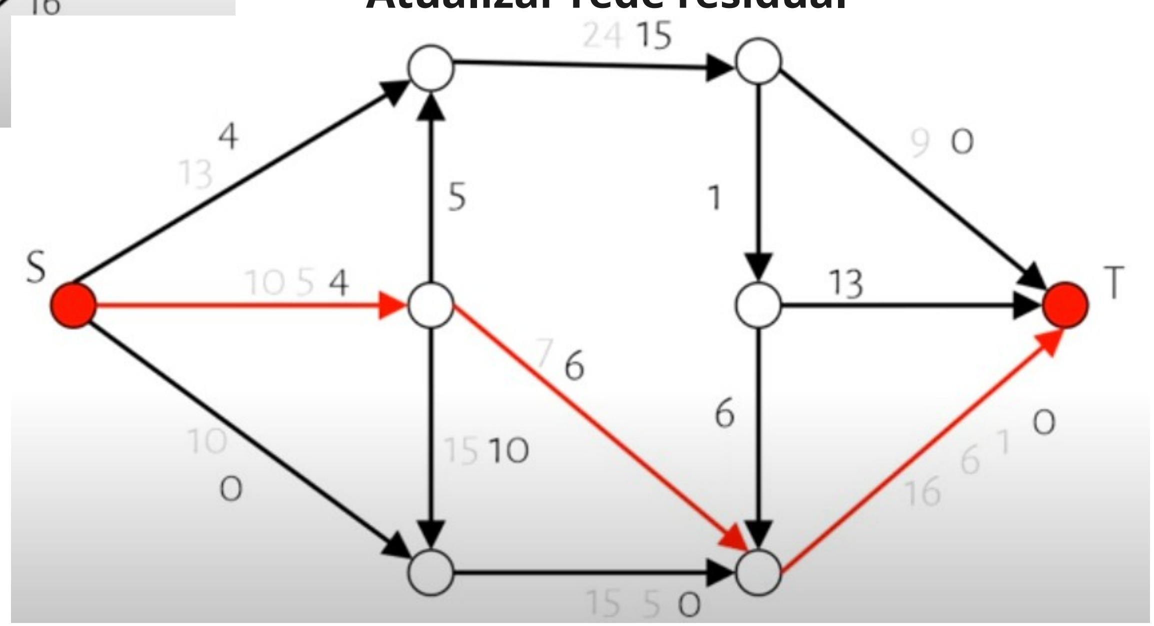
1 = 9

2 = 10

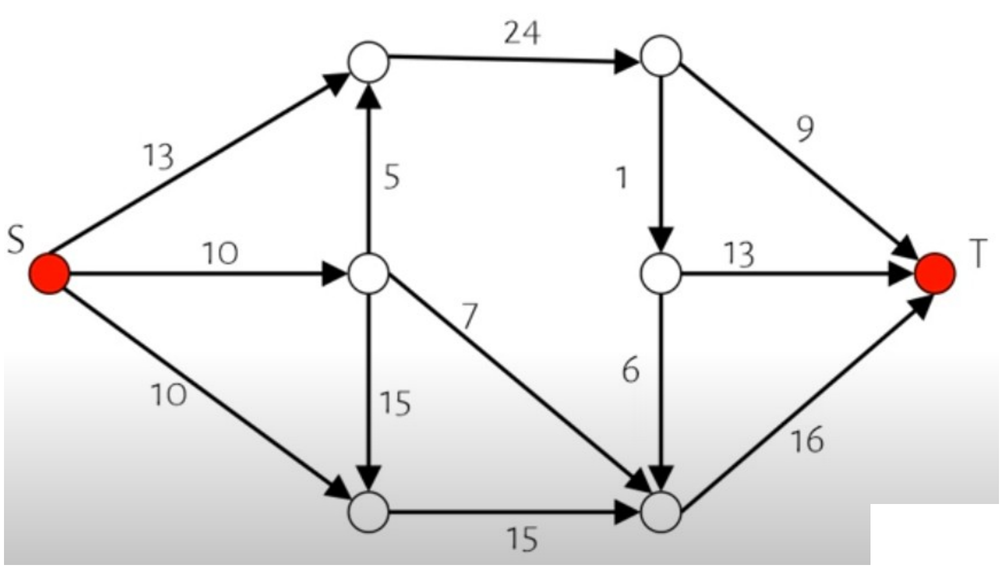
3 = 5

4 = 1

Atualizar rede residual



Ford-Fulkerson



INTERAÇÕES:

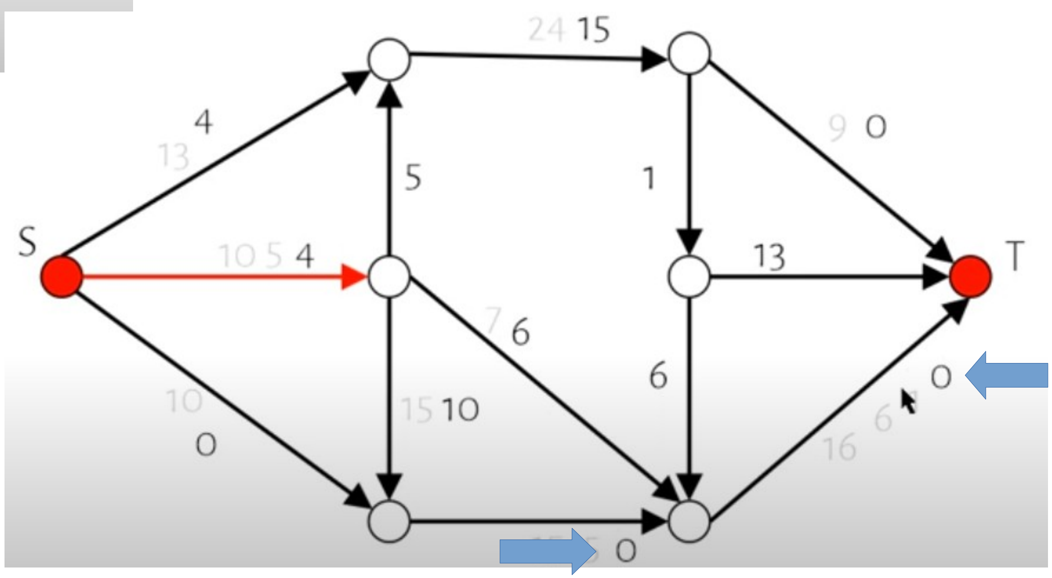
1 = 9

2 = 10

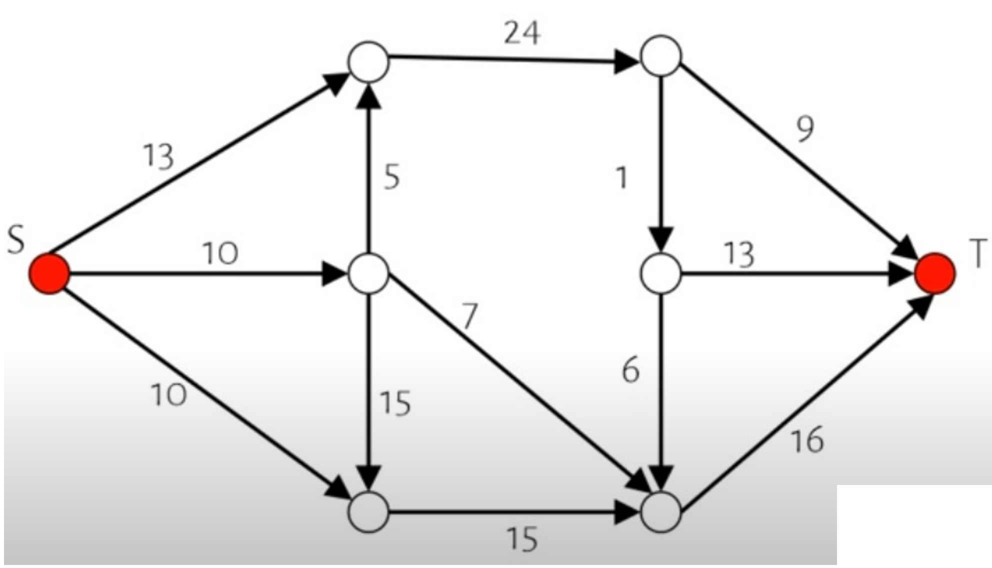
3 = 5

4 = 1

1. Busque um caminho da fonte ao destino com capacidade disponível.



Ford-Fulkerson



2. Determine o gargalo (capacidade mínima) do caminho.

INTERAÇÕES:

1 = 9

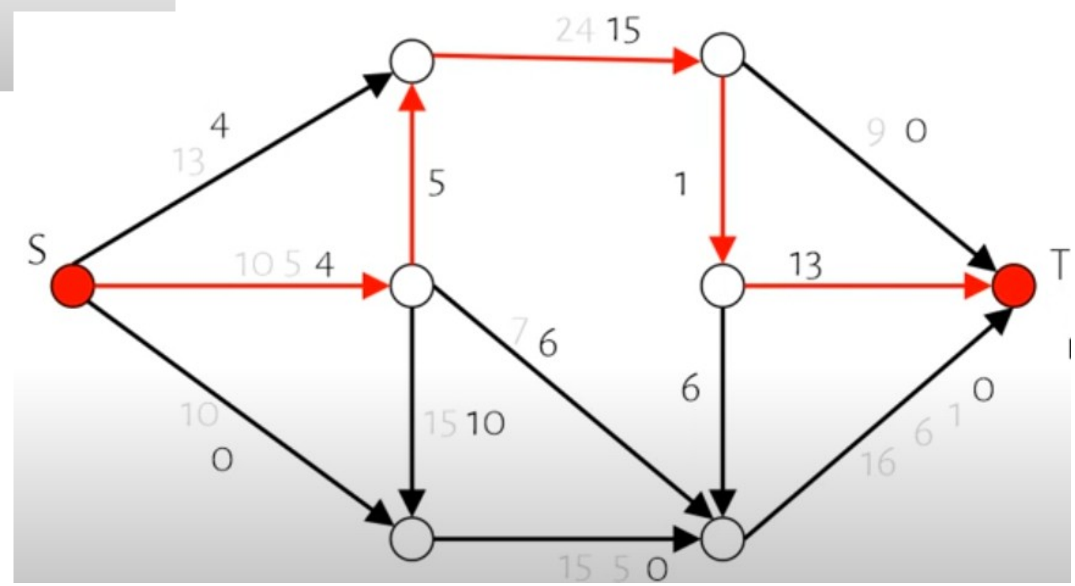
2 = 10

3 = 5

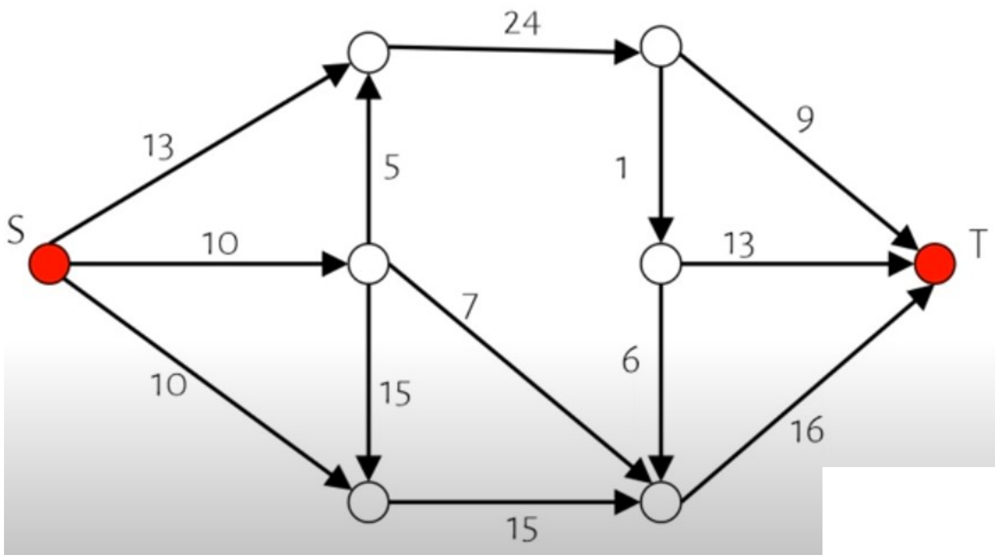
4 = 1

5 = 1

1. Busque um caminho da fonte ao destino com capacidade disponível.



Ford-Fulkerson



INTERAÇÕES:

1 = 9

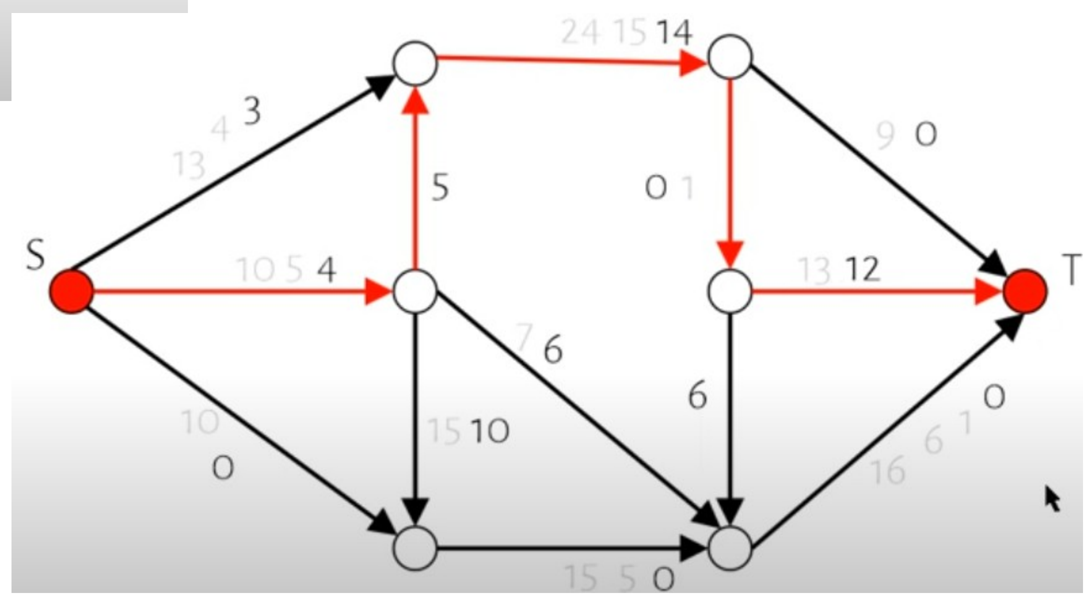
2 = 10

3 = 5

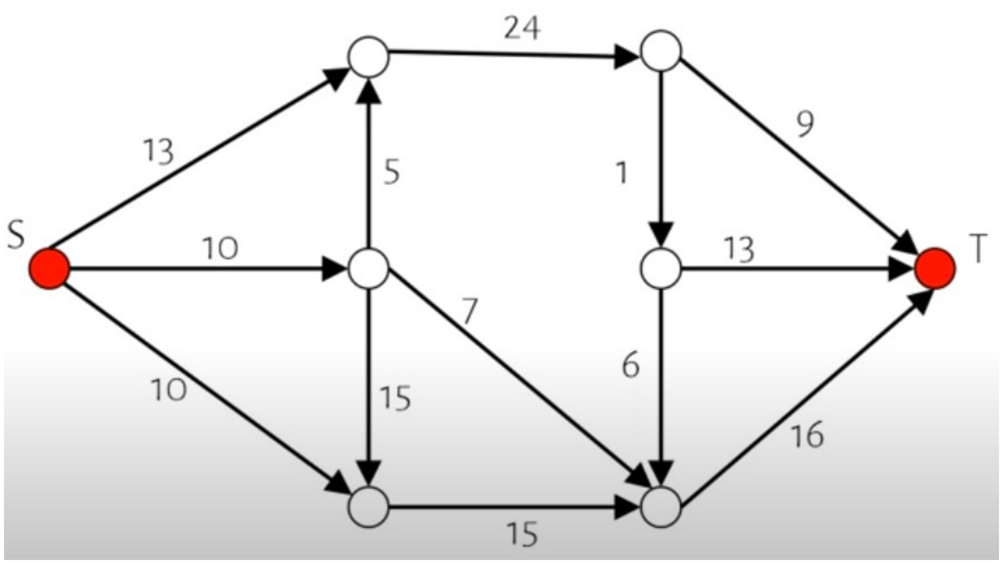
4 = 1

5 = 1

Atualizar rede residual



Ford-Fulkerson



INTERAÇÕES:

1 = 9

2 = 10

3 = 5

4 = 1

5 = 1

$$F_{\max} = \sum_{i=1}^{k} b(p_i)$$

26

Let's Code!

```python
class Graph:  1 usage
    def __init__(self, vertices):
        self.V = vertices
        self.graph = defaultdict(dict)

    def add_edge(self, u, v, w):  13 usages
        self.graph[u][v] = w
        if v not in self.graph or u not in self.graph[v]:
            self.graph[v][u] = 0  # Aresta reversa inicial

    # Busca em profundidade para encontrar caminho aumentante
    def dfs(self, s, t, visited, path):  2 usages
        if s == t:
            return path
        visited[s] = True
        for v in self.graph[s]:
            if not visited[v] and self.graph[s][v] > 0:
                res_path = self.dfs(v, t, visited, path + [(s, v)])
                if res_path is not None:
                    return res_path
        return None
```

```python
# Algoritmo Ford-Fulkerson
def ford_fulkerson(self, source, sink):  1 usage
    max_flow = 0
    while True:
        visited = [False] * self.V
        path = self.dfs(source, sink, visited,  path: [])
        if not path:
            break
        # Encontrar capacidade mínima do caminho (gargalo)
        flow = min(self.graph[u][v] for u, v in path)
        # Atualizar capacidades
        for u, v in path:
            self.graph[u][v] -= flow
            self.graph[v][u] += flow
        max_flow += flow
    return max_flow
```