

## **ALGORITMO MAX FLOW FORD-FULKERSON**

Felipe Pinheiro (Fundação Hermínio Ometto - FHO) feliipepinheiro09@alunos.fho.edu.br

Gabriel Augusto Basso (Fundação Hermínio Ometto - FHO)  
gabrielaugustobasso@alunos.fho.edu.br

João Pedro Rebelato (Fundação Hermínio Ometto - FHO)  
joaopedrorebe7756@alunos.fho.edu.br

Julio Cezar da Cunha (Fundação Hermínio Ometto - FHO) juliocezar@alunos.fho.edu.br

### **Resumo**

A teoria dos grafos nos permite calcular algumas previsibilidades comuns no dia a dia, presente em redes sociais, navegação pelo GPS entre outras aplicações que são utilizadas comumente. Pensando em como verificar o fluxo máximo em um grafo, Lester Randolph Ford Jr. e Delbert Ray Fulkerson desenvolveram um algoritmo Max Flow que recebeu o nome de Ford-Fulkerson.

**Palavras-Chaves:** Ford-Fulkerson, Max flow, Grafo.

### **1. Introdução**

Algoritmos de max flow (fluxo máximo) tem sido utilizado cada vez mais no dia a dia, possibilitando calcular o máximo possível do uso de produtos, armazenamento em estoque, o consumo ideal máximo de matéria prima em produções de produtos das fábricas, bem como outras finalidades. Para que tudo isso fosse possível nos dias atuais Ford e Fulkerson uniram os pensamentos para desenvolver um algoritmo que a partir de um Grafo, uma fonte, um terminal e a capacidade das arestas fosse possível calcular os caminhos possíveis, não obrigando a escolha de um caminho mais curto ou com maiores capacidades, permitindo assim escolha de qualquer um dos caminhos.

### **2. Max flow**

O algoritmo de fluxo máximo é uma técnica fundamental na teoria dos grafos que identifica a maior quantidade possível de "material" que pode fluir de uma fonte a um sumidouro em uma rede capacitada. Desenvolvido por Ford e Fulkerson em 1956, o algoritmo funciona através da identificação iterativa de pseudo caminhos aumentadores, onde se busca caminhos não saturados entre a fonte e o sumidouro para incrementar o fluxo. A cada iteração, aument a-se

o fluxo pela capacidade residual do caminho encontrado (o mínimo entre as capacidades disponíveis dos arcos) até que não existam mais caminhos aumentadores. Neste ponto, os arcos saturados formam um corte mínimo que representa o gargalo da rede, e o valor do fluxo máximo é igual à soma das capacidades destes arcos, permitindo otimizar desde redes de distribuição até sistemas de telecomunicações.

## 2.1. Algoritmo

O algoritmo precisa do grafo, uma fonte, o final e carga ou capacidade, com essas informações é possível realizar o cálculo algorítmico, que inicia sentando o fluxo global como zero, para cada aresta “E” fluxo é igual a zero, depois começa as verificações, onde o processo se repete enquanto as partes do grafo houverem caminhos a serem percorridos, depois de cada verificação, atualizamos o grafo residual, após a atualização verificamos novamente os caminhos.

Exemplo:

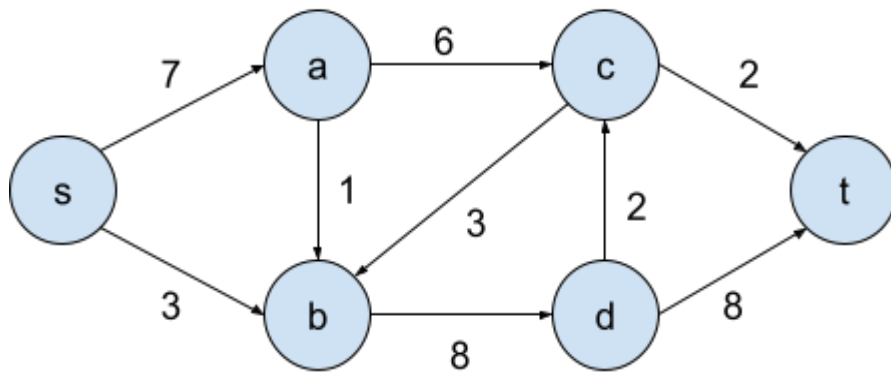
```
Fluxo Máximo(G, s, t, c){
    fluxo global = 0
    Para cada e  $\in$  E
        f(e) = 0
    Enquanto  $\exists$  Pst no Gf {
        Qualquer Pst # Qualquer caminho Pst
        f' = Aumentar(f, Pst)
        Atualizar o Grafo residual Gf
    }
}
```

Para começar as interações com o Grafo, escolhemos a aresta que possui maior peso ou capacidade, observamos a direção a partir do vértice (nó), usamos o mesmo critério a aresta que possui o maior peso, isso até chegar no final “t”, (ver grafo na figura 1).

Respeitando as interações conforme descrito anteriormente, faremos o primeiro fluxo [s, a, c, b, d e t] e o valor máximo que conseguimos nesta interação é 3, pois a aresta com menor valor está localizada na intersecção c para b. Para atualizarmos o grafo residual, precisamos subtrair 3, o valor encontrado, de cada vértice gerando assim um novo grafo (ver grafo residual 1 figura 2).

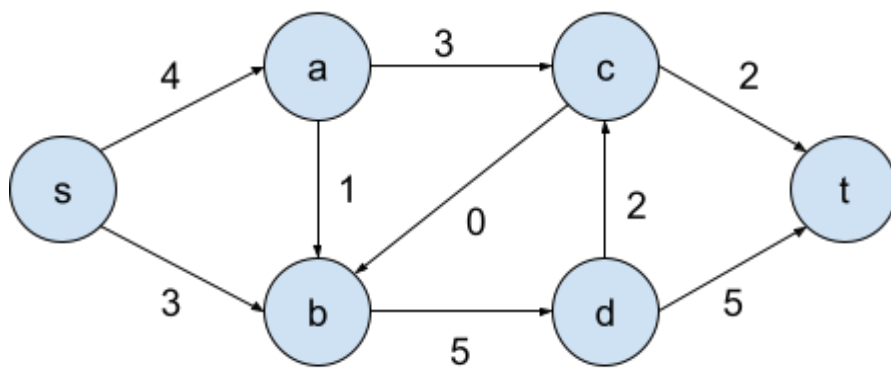
A partir do grafo residual repetimos o processo até que não seja possível passar por nenhum caminho, ou seja, até que os possíveis caminhos estejam com peso zero.

Figura 1 - Grafo inicial



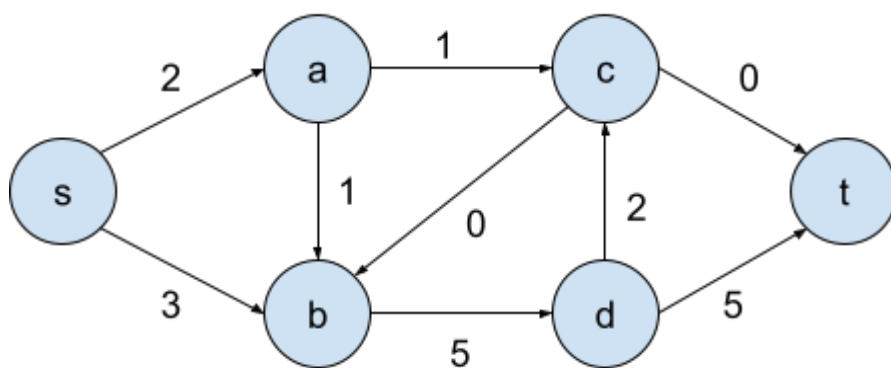
Fonte: Autor.

Figura 2 - Grafo residual 1



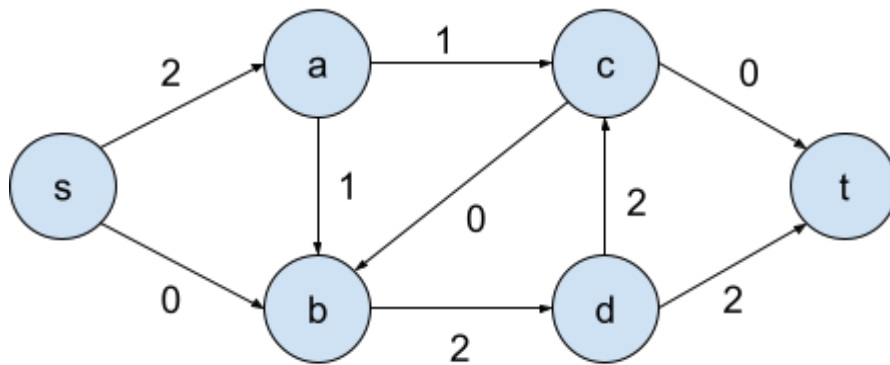
Fonte: Autor.

Figura 3 - Grafo residual 2



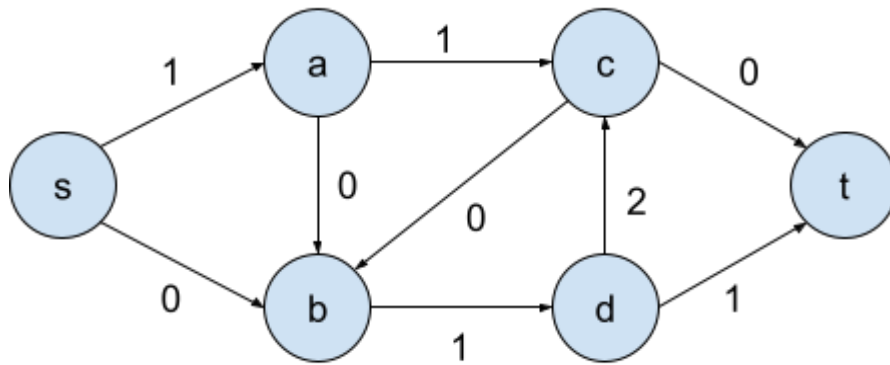
Fonte: Autor.

Figura 4 - Grafo residual 3



Fonte: Autor.

Figura 5 - Grafo final



Fonte: Autor.

Somando os valores que foram subtraídos nas interações dos grafos até chegar no último, temos o valor de 9, o que corresponde ao fluxo máximo do grafo apresentado.

### 3. Desenvolvimento

Para a proposta deste trabalho precisa-se desenvolver o grafo sem o uso de bibliotecas que permita a programação realizar o cálculo diretamente, sendo assim, gerando a necessidade de implementação deste cálculo diretamente.

Para comparativo e validação dos acertos, necessitamos de um parâmetro ou um projeto base, pois a partir dele será possível atestar a genuinidade do método e dos cálculos realizados internamente.

Foi escolhida a linguagem em Python para o desenvolvimento deste trabalho, com ela é possível gerar uma classe do grafo que vai realizar o cálculo de busca do maior fluxo para o valor inserido no código.

O primeiro passo foi definir a classe e estruturar a mesma, pois a partir disto será possível a identificação das arestas e capacidades. Dentro desta estrutura está a função que usa o algoritmo de Ford-Fulkerson para encontrar o fluxo máximo e obter a resposta esperada.

Depois escrevemos o grafo através de uma matriz, que possui o tamanho de acordo com a quantidade de vértices, por exemplo, o grafo encontrado na figura 1, possui 6 nós (arestas), sendo assim a matriz correspondente é 6 x 6 (ver tabela 1 - Matriz grafo).

Tabela 1 - Matriz do grafo inicial

	s	a	b	c	d	t
s	0	7	3	0	0	0
a	0	0	1	6	0	0
b	0	0	0	0	8	0
c	0	0	3	0	0	2
d	0	0	0	2	0	8
t	0	0	0	0	0	0

Fonte: Elaborado pelo autor (2025).

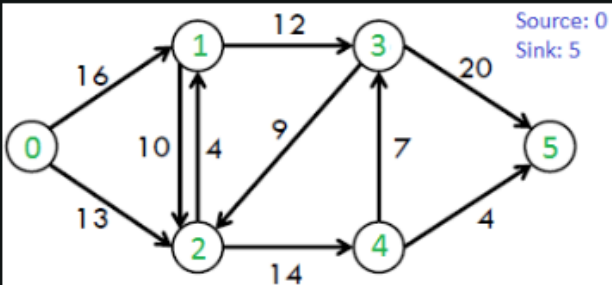
### 3.1. Base de comparação

Para a validação deste projeto utilizamos como base o canal do Youtube de Lucas Mattos, que fez uso do blog Geeks, neste blog tem um exercício que é resolvido através do algoritmo de Ford-Fulkerson, buscando o fluxo máximo do grafo plotado na página.

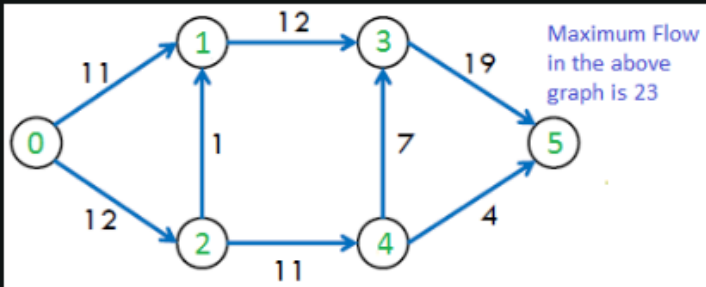
Conforme o autor o grafo (ver na Figura 6), precisa apresentar o resultado de 23 para o fluxo máximo encontrado, e, quando aplicamos o conhecimento adquirido no item **2.1. Algoritmo** podemos chegar ao resultado do fluxo igual a 23.

Embora o código desenvolvido não gere partes gráficas, ao inserir as informações de capacidade em forma de matriz, o código precisa resultar no valor de 23 como fluxo máximo encontrado (ver retorno do código na Figura 7).

For example, consider the following graph from the CLRS book.



The maximum possible flow in the above graph is 23.



<https://www.geeksforgeeks.org/ford-fulkerson-algorithm-for-maximum-flow-problem/>

Figura 7 - Execução código python

Fonte: Autor.

## 4. Resultados

Após a realização dos testes no código em Python do algoritmo de Ford-Fulkerson, obtivemos um resultado satisfatório e que entrega o valor de fluxo máximo calculado pelo algoritmo.

O desafio deste desenvolvimento está em codificar este algoritmo de grafo sem o uso de bibliotecas auxiliares já existentes, uma vez que o uso delas não permitiria o conhecimento passo a passo do que estaria sendo executado nas linhas de código.

Como uma segunda verificação de validação do código, inserimos a matriz equivalente do exemplo utilizado no item 2.1. *Algoritmo*, que gerou o resultado 9 para o fluxo máximo, que foi calculado previamente (ver resultado na Figura 8).

Figura 8 - Fluxo máximo exemplo item 2.1. Algoritmo

```
58 grafo = [[0, 7, 3, 0, 0, 0],
59          [0, 0, 1, 6, 0, 0],
60          [0, 0, 0, 0, 8, 0],
61          [0, 0, 3, 0, 0, 2],
62          [0, 0, 0, 2, 0, 8],
63          [0, 0, 0, 0, 0, 0]]
64
65 g = Grafo(grafo)
66 s = 0
67 t = 5
68 print ("O fluxo máximo possível é %d " % g.fordFulkerson(s, t))
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS QUERY RESULTS Python D

```
PS D:\Documentos\documentos\fho\Teoria dos Grafos> d;; cd 'd:\Documentos\documentos\fho\Teoria dos Grafos'; & 'c:\Users\
hon\Python313\python.exe' 'c:\Users\gabriel\.vscode\extensions\ms-python.debugpy-2025.8.0-win32-x64\bundled\libs\debugpy\la
tos\documentos\documentos\fho\Teoria dos Grafos\FordFulkerson.py'
in32-x64\x5cbundled\x5clibs\x5cdebugpy\x5claucher' '52020' '--' 'D:\x5cDocumentos\x5cdocumentos\x5cfho\x5cTeoria dos Gra
70-9e35-48ce-aa81-b6b64de9319c0 fluxo máximo possível é 23
● PS D:\Documentos\documentos\fho\Teoria dos Grafos> d;; cd 'd:\Documentos\documentos\fho\Teoria dos Grafos'; & 'c:\Users\
hon\Python313\python.exe' 'c:\Users\gabriel\.vscode\extensions\ms-python.debugpy-2025.8.0-win32-x64\bundled\libs\debugpy\la
tos\documentos\documentos\fho\Teoria dos Grafos\FordFulkerson.py'
in32-x64\x5cbundled\x5clibs\x5cdebugpy\x5claucher' '52457' '--' 'D:\x5cDocumentos\x5cdocumentos\x5cfho\x5cTeoria dos Gra
70-9e35-48ce-aa81-b6b64de9319c0 fluxo máximo possível é 9
○ PS D:\Documentos\documentos\fho\Teoria dos Grafos> []
```

Fonte: Autor.

## REFERÊNCIAS BIBLIOGRÁFICAS

LEVADA, Alexandre. **O algoritmo de Ford-Fulkerson**. [S.I.]: YouTube, 2021. Disponível em: [https://www.youtube.com/watch?v=TPK\\_PWd\\_xhA&t=160s](https://www.youtube.com/watch?v=TPK_PWd_xhA&t=160s). Acesso em: 09 de maio de 2025.

IME-USP. "O problema do fluxo máximo num grafo". Disponível em: [https://www.ime.usp.br/~pf/algoritmos\\_para\\_grafos/aulas/flow.html](https://www.ime.usp.br/~pf/algoritmos_para_grafos/aulas/flow.html).

IME-USP. "Algoritmo de Ford-Fulkerson para fluxo máximo num grafo". Disponível em: [https://www.ime.usp.br/~pf/algoritmos\\_para\\_grafos/aulas/flow-FF.html](https://www.ime.usp.br/~pf/algoritmos_para_grafos/aulas/flow-FF.html).

FORD-FULKERSON Algorithm for Maximum Flow Problem. **Geeks for Geeks**, 01 junho de 2023. Disponível em:

<https://www.geeksforgeeks.org/ford-fulkerson-algorithm-for-maximum-flow-problem/>.

Acesso em: 19 maio 2025.

MATTOS, Lucas. **Algoritmo de Ford-Fulkerson (Fluxo máximo)**. [S.I.]: YouTube, 15 junho de 2022. Disponível em: [https://www.youtube.com/watch?v=-Exigir\\_8c](https://www.youtube.com/watch?v=-Exigir_8c) Acesso em: 20 maio 2025