

**Aluna:** Ana Cláudia Machado

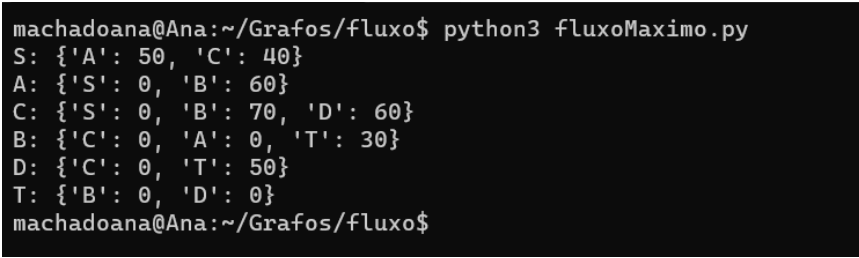
**Entrada:**

A entrada utilizada é um arquivo *.txt*, intitulado "graph", contendo as arestas e o fluxo que pode passar por elas. Assim, em cada linha do arquivo, têm-se, respectivamente, os vértices inicial e final daquela aresta e seu fluxo. Diferente dos trabalhos anteriores, a biblioteca *networkx* não foi utilizada para a leitura da entrada, pois considerou-se, para o desenvolvimento do algoritmo de Ford Fulkerson, mais fácil trabalhar com dicionários do que com a estrutura do grafo pronta.

Desse modo, criou-se um dicionário  $G$  para representar o grafo, com cada um de seus vértices como chaves. Assim, foi possível representar as arestas de  $G$ , pois para cada uma das chaves, criou-se um dicionário, no qual as chaves eram os vértices atingidos por arestas com origem em determinada chave de  $G$ . Por fim, o valor atribuído era o fluxo naquela aresta, o que facilitou a construção do grafo de folgas.

Para exemplificar, as linhas de entradas abaixo resultaram no seguinte grafo:

```
S A 50
S C 40
C B 70
A B 60
C D 60
B T 30
D T 50
```



```
machadoana@Ana:~/Grafos/fluxo$ python3 fluxoMaximo.py
S: {'A': 50, 'C': 40}
A: {'S': 0, 'B': 60}
C: {'S': 0, 'B': 70, 'D': 60}
B: {'C': 0, 'A': 0, 'T': 30}
D: {'C': 0, 'T': 50}
T: {'B': 0, 'D': 0}
machadoana@Ana:~/Grafos/fluxo$
```

**Arquivos .py:**

Neste trabalho, “fluxoMaximo.py” importa “auxiliar”, o qual também é um arquivo *.py* que, como o próprio nome sugere, é um arquivo que auxilia na implementação do algoritmo, pois conta com diversas funções que foram criadas para que o código fosse organizado e legível.

Assim, “fluxoMaximo.py” conta apenas com a leitura do arquivo de entrada, a implementação geral do algoritmo de Ford Fulkerson e a impressão na tela dos resultados obtidos, sendo que, as demais funções criadas, estão todas no arquivo auxiliar.

**Funcionamento do algoritmo:**

A priori, é importante ressaltar quais foram as estruturas utilizadas no desenvolvimento do algoritmo e, a posteriori, como elas se relacionam com a dada implementação. Como já foi dito na introdução, o grafo  $G$  em si é um dicionário de dicionários. Além disso, foram utilizados outros dois dicionários importantes: *visitado* e *antecessor*, ambos utilizando a mesma ideia de que cada chave é um vértice de  $G$ . Em *visitado*, a cada nova busca em largura (BFS), se um vértice já foi adicionado como parte possível do caminho, então sua chave possui valor *True*, caso contrário, permanece como *False*. Já em *antecessor*, assim que o vértice  $v$  é adicionado ao dito possível caminho, registra-se qual vértice é seu antecessor.

Agora, é possível explicar o funcionamento do algoritmo: dada a fonte  $S$  e o sorvedouro  $T$ , ambos são fornecidos ao BFS a fim de que o algoritmo saiba onde estão o início e o fim do caminho a ser encontrado. Desse modo, cria-se uma fila  $Q$  com  $S$  como seu primeiro elemento e, para cada um dos elementos de  $Q$ , retira-o da fila e verifica-se a possibilidade de adicionar na fila os vértices  $u$  alcançados por ele, de acordo com duas condições:  $u$  não pode já ter sido visitado e o fluxo do elemento em questão para  $u$  deve ser maior que zero, isto é, algum valor de fluxo pode passar por ali. Caso as condições sejam satisfeitas, então  $u$  é adicionado à fila. Esse processo é finalizado quando a fila estiver vazia e, se *visitado*[' $T$ '] = *True*, o caminho encontrado é válido.

Percebe-se, portanto, que o algoritmo de Ford Fulkerson, na verdade, controla, de forma sofisticada e eficiente, um mecanismo que, por meio de *antecessor* e *visitado*, torna possível reproduzir uma BFS cujo objetivo é que o primeiro caminho encontrado que chegue em  $T$  seja o

retornado. Dessa maneira, a cada vez que a BFS for chamada, um novo caminho será criado e, quando o fluxo máximo já tiver passado, a busca simplesmente não chegará a  $T$  quando a fila for esvaziada.

A partir do momento em que um caminho é definido, basta encontrar o menor valor de fluxo nas arestas que o compõem, haja vista que esse é o fluxo máximo naquele caminho, e atualizar o grafo de folgas de acordo com esse valor. Além disso, soma-o com o valor do fluxo máximo no grafo, iniciado como zero. Ao final da execução, o algoritmo será capaz de fornecer o fluxo máximo em  $G$  e os fluxos elementares das arestas.

### ***Saída:***

A saída é composta por 2 partes: primeiro, têm-se uma impressão do fluxo máximo no grafo e, em seguida, o fluxo elementar de cada aresta do grafo de folgas. Logo, para a entrada exemplificada nesta documentação, a saída prevista é:

```
machadoana@Ana:~/Grafos/fluxo$ python3 fluxoMaximo.py
O fluxo maximo eh 70
----- FLUXO ELEMENTAR EM CADA ARESTA -----
S-A: 20
S-C: 0
A-S: 30
A-B: 30
C-S: 40
C-B: 70
C-D: 20
B-C: 0
B-A: 30
B-T: 0
D-C: 40
D-T: 10
T-B: 30
T-D: 40
machadoana@Ana:~/Grafos/fluxo$ |
```