

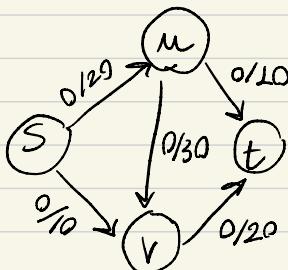
Resumo

- prova 3 -

Grafos

Ford-Fulkerson (Max Flow)

↳ Algoritmo usado para encontrar um fluxo de valor máximo com o melhor uso de capacidades possíveis.



$$\text{Caso 1} \rightarrow P = svt, v(f) = \underline{\underline{20}}$$

$$\text{Caso 2} \rightarrow P_1 = sut, v(f) = 20$$

$$P_2 = svt, v(f) = 20 + 10 = 20$$

$$P_3 = svt, v(f) = 20 + 10 = \underline{\underline{30}}$$

Gráfico Residual = $G_f = (V_f, E_f)$, gerado a partir de $G = (V, E, c)$

- i) $V_f = V$ (conjunto de vértices é o mesmo).
- ii) $E_f \neq E$, pois $\forall e \in E$ pode gerar até 2 arestas em E_f
 - a) forward-edge (nóma direção de e) = aresta e' com capacidade residual $c(e') = c(e) - f(e)$ (o que falta para saturar)
 - b) backward-edge (direção contrária a e) = aresta e'' com capacidade residual $c(e'') = f(e)$ (extraindo o que já passou).

Max Flow (G, s, t, c) {

$f = 0$ # fluxo global circulante do grafo setado como 0

for each $e \in E$

$f(e) = 0$ # fluxo inicial em cada aresta é nulo

while $\exists P_{st}$ in G_f { # enquanto existir caminho P_{st} no Grafo residual

| Let P_{st} be one of these Paths # seja P_{st} um desses caminhos

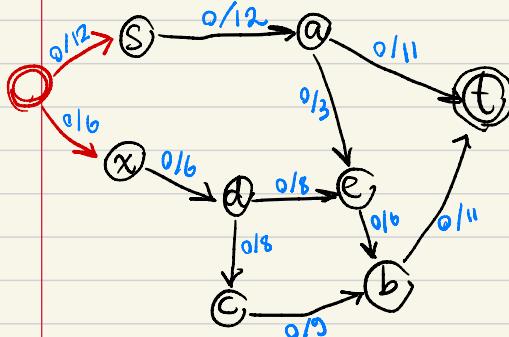
| $f' = \text{Augment}(f, P_{st})$ # amplia o caminho por onde P_{st}

| Update the residual Graph G_f # atualiza o Grafo Residual

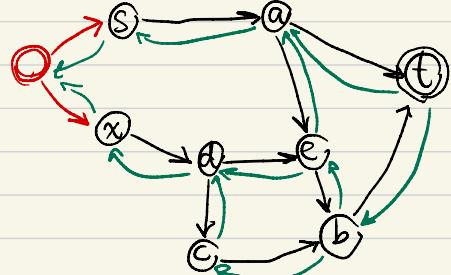
}

* a ordem de execução
não interfere no fluxo máximo

Exemplo de Ford - Fulkerson:

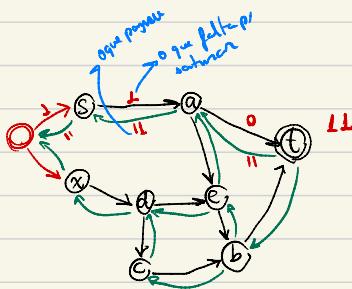
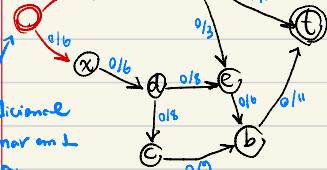


Grafo Residual:

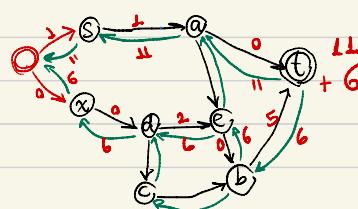
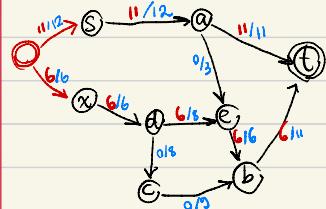


Passo 1:

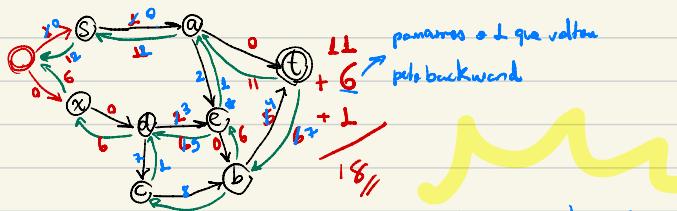
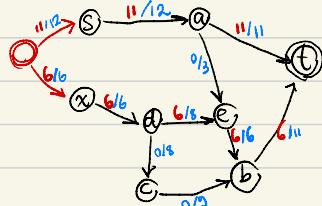
vértice adicional
p/ transformar em L
não entendi



Passo 2:



Passo 3:



volta pq pra um caminho é daí + pra passar o último L

tratarmos os arcos como arcos de caminho normal, entao se nenhuma caminho pelo arco, podemos seguir pelo arco!

caminho de aumento é um caminho de S até no grafo residual onde todos os arcos tem capacidade residual positiva.

(Só funciona pra direcionados)

Ford Fulkerson (Marcola):

- 1- Colocar o fluxo maximo igual a zero
- 2- Enquanto o grafo residual possuir caminho de aumento:
- 2.1- Encontrar um caminho de aumento P
- 2.2- Selecionar aresta v de P com menor capacidade
- 2.3- Somar o peso da aresta v no fluxo máximo
- 2.4- Atualizar o Grafo residual.

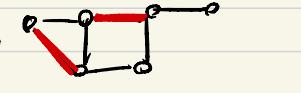
Ford Fulkerson (Prof):

- 1- Zerar os fluxos
- 2- Enquanto ouver caminho entre S e T:
- 2.1- Encontrar um caminho de aumento no grafo residual entre S e T
- 2.2- Atualizar o fluxo considerado caminho do aumento.

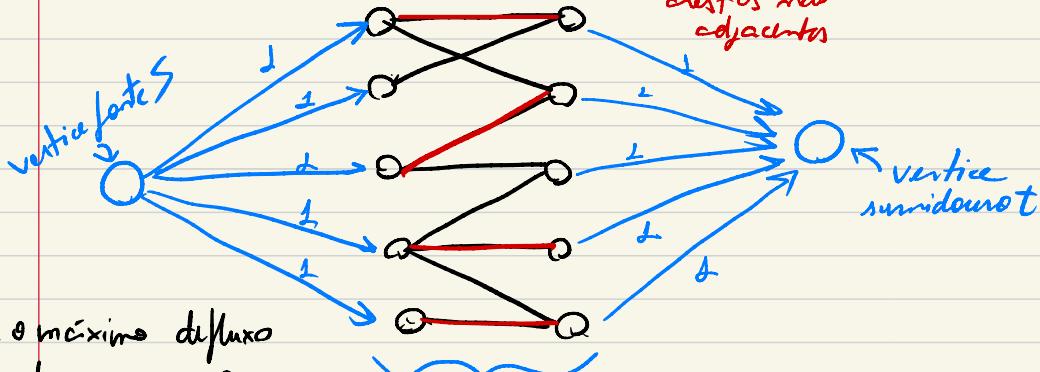
Ford-Fulkerson para o problema de casamento máximo em grafos bipartidos:

o objetivo é encontrar um emparelhamento máximo: o maior conjunto de arestas $M \subseteq E$ tal que nenhum vértice pertence a mais de uma aresta em M .

um emparelhamento máximo é dito perfeito se todos os vértices de grafo podem caber nos pares.



se um emparelhamento



para o máximo de fluxo corresponde ao uso o máximo de arestas

as arestas que permitem no fluxo não terão extremidades (vértices) em comum.

achando o fluxo máximo de S até, retomamos como matching as arestas de V_1 para V_2 que foram usadas no fluxo.

Edmonds-Karp

↳ Para resolver o problema de Ford-Fulkerson (de sempre visitar os piores caminhos possíveis, aumentando o custo computacional), Edmonds implementou a busca em largura (BFS) no Ford-Fulkerson para encontrar os caminhos aumentantes, garantindo sempre a escolha do caminho mais curto.

> Passo a passo:

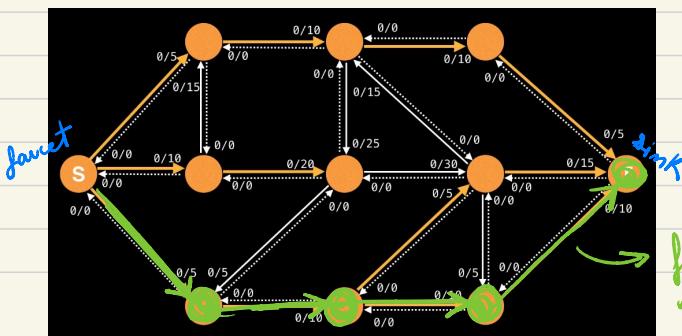
1- $\forall e \in E, f=0$ (inicializamos o fluxo de todos os arcos como 0)
↳ cria-se um grafo residual

2- Utiliza-se a BFS p/ encontrar um caminho de s a t onde ainda exista capacidade disponível em todos os arcos do caminho (um caminho aumentante), sendo que o caminho encontrado é sempre o mais curto (em n.º de arcos)

(menor capacidade residual ao longo do caminho)

3- Determina-se o gorgalo do caminho (a qtd máx de fluxo que pode ser enviada pelo caminho encontrado) e atualiza o fluxo no grafo residual.

4- Repita 2 e 3 até que não haja mais caminhos aumentantes no grafo residual.



fluxo c/ o menor n.º de arcos

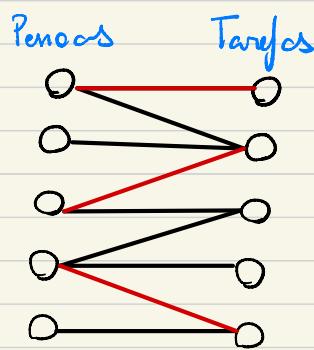
Bipartite-Matching

- ↳ É encontrar um subconjunto de um grafo bipartido em que nenhum vértice seja compartilhado por mais de uma aresta (cada vértice está conectado a, no máximo, uma aresta de matching) (ou seja, não há vértices adjacentes)
- matching máximo → um matching que contém o maior número possível de arestas.
- matching perfeito → um matching onde todos os vértices do grafo estão emparelhados (cada vértice pertence a 1 aresta de matching)

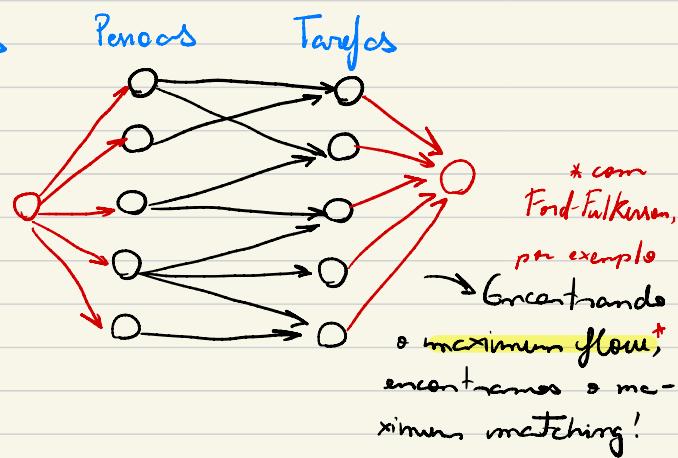
relembre

Gráfico Bipartido: É um gráfico $G = (V, E)$ em que seus vértices podem ser divididos em dois grupos independentes $U \cup S$ e todos os arestas de gráfico conectam vértices de $U \cup S$ ($u \in S$ e $s \in V$).

↳ Outras definições: gráfico em que não há 2 cores na coloração ou não há ciclos de tamanho ímpar.

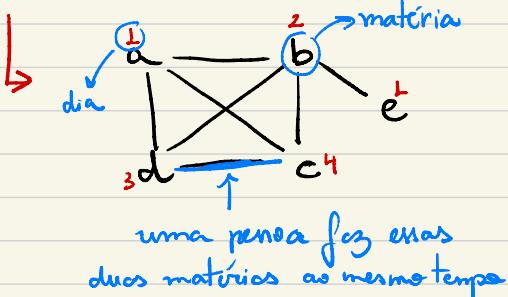


é um matching,
mas não máximo.



- 1 Given bipartite graph $G = (A \cup B, E)$, direct the edges from A to B .
- 2 Add new vertices s and t .
- 3 Add an edge from s to every vertex in A .
- 4 Add an edge from every vertex in B to t .
- 5 Make all the capacities 1.
- 6 Solve maximum network flow problem on this new graph G' .

Problema inicial: Sejam n alunos de um curso. Cada aluno faz um conjunto de disciplinas e a turma pede para não ter duas ou + provas no mesmo dia. Como uma organização pode ser modelada para atender a esse pedido?



Soluções:

- 1- **Colorações do Grafo** (atribuindo cores aos vértices de modo a minimizar a qntd. de cores).
- 2- Encontrando o maior subgrafo completo desse grafo (\circ clique). → não é ótimo!!!

* Se há um clique com R vértices no Grafo, então pelo menos R dias serão necessários, já que todas as disciplinas nesse clique estão em conflito entre si. - Mas não há garantia de que não há outros conflitos fora do clique.



Conceptos:

Clique (Limite Inferior)	Coloração (Solução Completa)
Determina o número mínimo de cores necessárias (limite inferior). <small>→ (mais & menor)</small>	Determina exatamente o número mínimo de cores necessárias para todo o grafo.
Baseia-se apenas no maior subgrafo completo.	Considera toda a estrutura do grafo e todas as conexões entre os vértices.
Não resolve conflitos <small>fora do clique</small> .	Resolve todos os conflitos no grafo globalmente.

prevenindo-lo é
força bruta ou
backtracking.

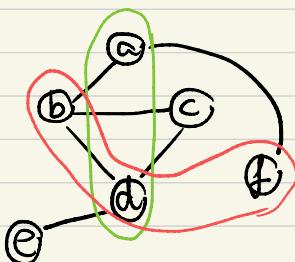
Busca
gulosa
não recursiva

clique → Maior subgrafo induzido completo (subconjunto de vértices em que todas as pares de vértices estão conectados por uma aresta).

conjunto independente → Maior subgrafo induzido nulo (o maior subconjunto de vértices que não têm conexão entre si).

cobertura de vértices → Menor subconjunto de vértices tal que esse conjunto de vértices é vizinho a todos o meu conjunto de cores). - minimiza arestas

conjunto dominante → Menor subconjunto de vértices que são vizinhos a todos os vértices. - minimiza vértices



★ - cobertura de vértices

★ - conjunto dominante

► **Subgrafo induzido** = é um subconjunto de vértices do grafo original, mas inclui todas as arestas entre esses vértices que estão no grafo original.

Subgrafo Normal	Subgrafo Induzido
Você escolhe quais vértices e quais arestas incluir.	Você escolhe apenas os vértices; as arestas são determinadas automaticamente.
Pode excluir algumas arestas mesmo que conectem os vértices escolhidos.	Inclui obrigatoriamente todas as arestas entre os vértices escolhidos no grafo original .
Mais flexível.	Mais restritivo.

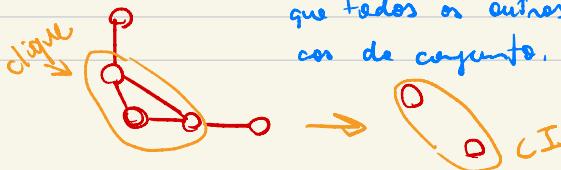
Relações:

↳ Clique e Conjunto Independente são opostos. Logo, se há um clique no grafo original, ele é um conjunto independente no grafo complementar a ele.

↳ Selecionando uma Cobertura de Vértices, os demais vértices são independentes. (Extraindo a CV, os vértices que restam não mudam)

- Se S é CLIQUE em $G \rightarrow S$ é CI em \overline{G} .
- Se S é CI em $G \rightarrow V \setminus S$ é CV em \overline{G} .
- Se é CV, é CD, mas nem todo CD é CV.

↳ porque cobre todos os vizinhos garantindo que todos os outros vértices não adjacentes aos de conjunto.



$$(V \setminus CV = CI)$$

$$CI + CV = V$$

Conceito	Definição	Problema Associado
Clique	Subgrafo completo onde todos os pares de vértices estão conectados.	Encontrar o maior clique.
Conjunto Independente	Subconjunto onde nenhum par de vértices está conectado por uma aresta.	Encontrar o maior conjunto independente.
Cobertura de Vértices	Subconjunto onde todas as arestas têm pelo menos um extremo no conjunto.	Encontrar a menor cobertura.
Conjunto Dominante	Subconjunto onde todos os vértices estão no conjunto ou adjacentes a ele.	Encontrar o menor conjunto dominante.

Coloração de Vértices

↳ Atribuições de cores em tabelas aos vértices de um grafo de forma que nenhum par de vértices (conectados por uma aresta, ou seja, vizinhos) tenham a mesma cor.

↳ O objetivo é minimizar o número de cores utilizadas, chamados de número cromático do grafo

↳ Uma coloração é considerada própria se nenhuma par de vértices compartilha a mesma cor (colorido de forma válida)

Propriedades:

- um grafo bipartido sempre pode ser colorido com no máximo 2 cores.
- um grafo com ciclos ímpares exige ao menos 3 cores
- o número cromático ($\chi(G)$) de um grafo determina o menor número de cores necessárias p/ obter uma coloração própria.
- o número cromático é sempre maior ou igual ao tamanho do clique do grafo (pq o clique é o limite inferior!)
- qualquer grafo plano pode ser colorido com no máximo 4 cores

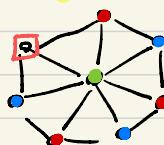


Algoritmos para coloração:

► algoritmo guloso: atribui cores aleatoriamente desde que não viole a regra. — Não garante a solução ótima.

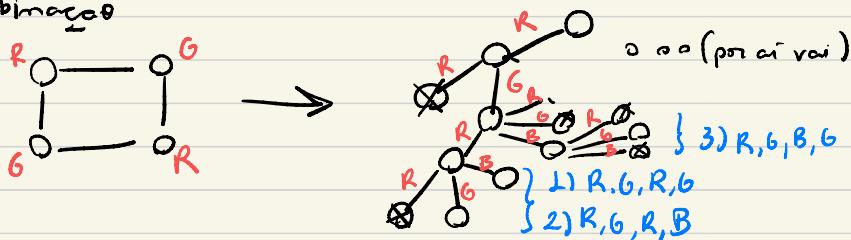
(o menor número de cores).

► ordenar por grau não descendente (seleciona os menores primeiros) não é ótimo, pois há contra-exemplos:



► backtracking: explora todas as combinações possíveis (exponente potência) para encontrar a solução ótima (é computacionalmente caro).

↳ mais lento, mas garante a solução correta
↳ constrói uma árvore com todas as possibilidades de combinação



Coloração de arestas:

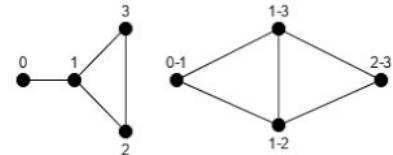
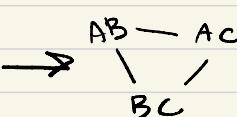
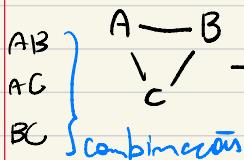
► podemos realizar a coloração de arestas implementando a coloração de vértices no grafo de linha do grafo original.

Coloração de
arestas no grafo =

original

coloração de
vértices no
Grafo de Linha
 $L(G)$

► **grafo de limha** = todos os arestas do grafo original não representadas como vértices no grafo limha.



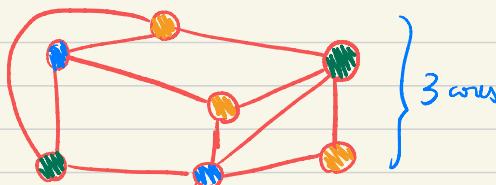
► Quantas arestas existem em $L(G)$ para K_m ?

$$\text{limite superior} = \binom{|E|}{2} \quad (\text{combinacões de todos os arestas } 2=2.)$$

$$\text{valor ótimo} = \frac{\sum (\text{grau de cada vértice})^2}{2} - E$$

► Grafo de limha de um plânio é plânio também?
→ SIM!

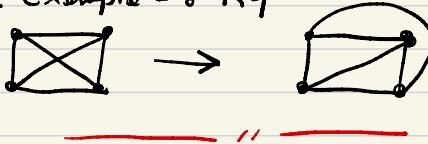
► **Teorema dos 4 cores** = afirma que, para qualquer grafo plânio, é possível colorir os vértices usando no máximo 4 cores (de forma que mesmo assim a coloração ainda seja própria).



Planaridade

↳ É uma propriedade que indica se um grafo pode ser desenhado no plano sem que haja um cruzamento entre arestas.

► Um grafo é planar se ele puder ser desenhado sem cruzamentos. Exemplo = K_4



(Se ele tiver uma representação planar sem cruzamentos, é planar).



* todo grafo planar possui pelo menos uma representação com arestas "retas".

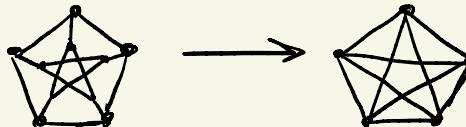
► Grafo dual = é um grafo construído a partir de um grafo planar, definido da seguinte forma:

- cada face do grafo G corresponde a um vértice em G^* .
- para cada aresta e em G , se e separa duas faces f_1 e f_2 , então existe uma aresta em G^* que conecta os vértices f_1 e f_2 .
- a dualidade é bidirecional $(G^*)^* = G$ (o grafo dual da dual retorna o grafo original)



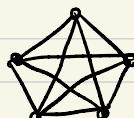
► homeomorfismo = é uma função $f: V_G \rightarrow V_H$, tal que, \forall aresta $(u, v) \in E_G$, temos que $(f(u), f(v)) \in E_H$.

↳ ou seja, se dois vértices não são adjacentes em G , seus correspondentes em H também devem ser adjacentes. Ele preserva as adjacências entre os vértices de um grafo.

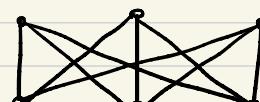


► Teorema de Kuratowski = um grafo é planar se, e somente se, ele não contém um subgrafo que seja uma subdivisão de K_5 ou $K_{3,3}$. (grafos não-planares fundamentais)

(que possua um K_5 ou $K_{3,3}$ como contração)



K_5



$K_{3,3}$

* Como identificar se um grafo é planar analisando a quantidade de vértices e arestas?

$$V - A + F = 2$$

o em relação ao K_5 :

2 faces vizinhas compartilham
a mesma aresta

$$\hookrightarrow e = 3f/2$$

$$V - e + \frac{2e}{3} = 2 \Leftrightarrow f = 2v/3$$

o em relação ao $K_{3,3}$:

1 face = 4 arestas
 $\hookrightarrow e = 4f/2$

$$V - e + \frac{e}{2} = 2 \Leftrightarrow$$

$$e \leq 3v - 6$$

$$e \leq 2v - 4$$

Logo:

se o grafo satisfizer
uma das equações,
ele é PLANAR.

MST = árvore geradora que conecta todos os vértices de gráfio minimizando o peso total das arestas
 ↳ (e temete c/ arestas originais) **garoto**

A Árvore Geradora com Bottleneck Mínimo = é uma árvore em que o peso da aresta + peso da é o menor possível entre todos os arvores geradoras

Se elas permitem todos os pesos distintos

• não são os menores, há contra-exemplos.



vértice pendente \Rightarrow tem grau igual a 1

vértice de corte \Rightarrow removê-lo causa desconexão
 (tira uma aresta ponte)

cut-set = conjunto de corte \rightarrow conjunto de arestas cuja remoção causa desconexão (e aumenta o n° de componentes conexos).

Aspecto	Ciclo Euleriano	Ciclo Hamiltoniano
Críterio	Percorre todas as arestas <u>do gráfico</u> exatamente uma vez	Visita todos os vértices exatamente uma vez
Condições para Existência	Grau par para todos os vértices (não-direcionado)	Não há condições gerais simples
Complexidade Computacional	Determinar a existência é polinomial existe alg. pr. identificar	Determinar a existência é NP-completo existe algoritmo "
Relação com Arestas	Todas as arestas devem ser percorridas	Nem todas as arestas precisam ser usadas

mº de circuitos hamiltonianos
 $\downarrow (n-1)!$

↳ que iniciam no mesmo V
 (n iniciam no próximo c/ 10!)

Um grafo simples, conexo e não direcionado com n vértices e m arestas deve satisfazer a seguinte condição:

- O número mínimo de arestas em um grafo conexo é $n - 1$ (uma árvore).
- O número máximo de arestas em um grafo simples é $\binom{n}{2} = \frac{n(n-1)}{2}$, que ocorre quando o grafo é completo.

Caminhos Eulerianos

- 1- Contar o grau de entrada de cada vértice;
- 2- Caso tenha no grafo mais de dois vértices de grau ímpar, não há caminho euleriano, pois para ter caminho euleriano ou todos os vértices tem o grau de entrada pares ou tem no max 2 ímpares;
- 3- Caso seja o caso de termos 2 de grau ímpar, o caminho euleriano deverá começar em um destes vértices e deverá terminar no outro;
- 4- Caso seja o caso de termos todos de grau par, o caminho euleriano deverá começar em qualquer vértice;
- 5- Depois de escolhermos o vértice inicial, enquanto houver arestas não visitadas no caminho, escolhemos a aresta não visitada e caminharemos por ela e adicionamos em nosso caminho atual (dando prioridade de escolha as arestas que não são pontes). Logo em seguida, removeremos a aresta caminhada do nosso grafo;
- 6- Quando não houver mais arestas não visitadas no nosso caminho atual, apenas retornaremos ao nosso último vértice que tinha arestas não visitadas e damos continuidade para o nosso processo;
- 7- No final desse caminhamento nos teremos nosso caminho euleriano.

*caminho euleriano
em um gráfico*

fechado ↙ ↘ aberto :

- se todos os vértices
possuem grau par.

- se exatamente 2
vértices possuem grau
ímpar.

(e ele deve começar em 1
ímpar e terminar no outro)

$$A \subseteq B$$

↓
 $A \in \text{subconjunto de } B$

União p/ fazer \Rightarrow
intersecção

$$\overline{(A \cup B)}$$