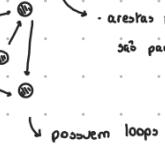


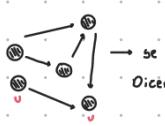
CONCEITOS



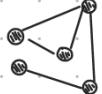
Podem ser dirigidos



↳ possuem loops



Podem ser não-dirigidos



↳ não possuem loops

(u,v) significa dizer que a aresta sai de u e entra em v .

→ Em grafos não-dirigidos a relação de adjacência é simétrica



Grau de vértice

Se G é um grafo não-dirigido:

O grau de um vértice v é igual ao número de arestas que incidem nele.



Se G é um grafo dirigido:

O grau de um vértice é igual ao número de arestas que saem dele somado ao número de arestas que saem dele.



Existe (u,v) mas não existe (v,u)



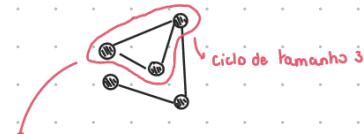
Ciclos
Em grafos dirigidos o caminho deve conter pelo menos uma aresta.



Gráfico cíclico → grafos em que há ao menos 1 ciclo

Gráfico acíclico → grafos em que não há ciclos

Um ciclo acontece quando, a partir de um vértice v , conseguimos percorrer um caminho que nos leva a esse mesmo vértice



Em gráfico não-dirigido um ciclo deve conter ao menos 3 vértices

Gráfico conexo

Em um gráfico não-dirigido:

Um gráfico G será conexo (ou conectado) se existir um caminho entre qualquer par de vértices



Gráfico conexo

Em um gráfico direcionado:

G será conexo se existir um caminho (u,v) ou um caminho (v,u) para cada par de vértice (u,v) .

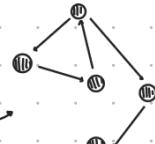
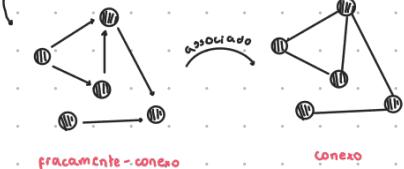


Gráfico conexo

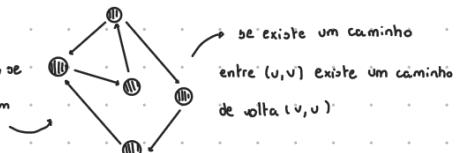
Em um gráfico direcionado:

G será fracamente conexo se a sua substituição de arestas mudando a sua direção (gráfico associado) for conexo.



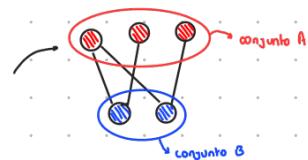
Em um gráfico direcionado:

G será fortemente conexo, se para cada par de vértice existir um caminho entre eles.



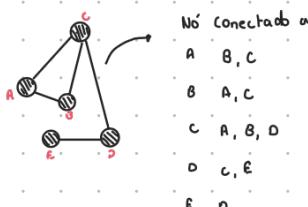
Grafo Bipartido

Um grafo em que os vértices podem ser separados em dois conjuntos distintos e não existem arestas entre vértices de um mesmo conjunto.

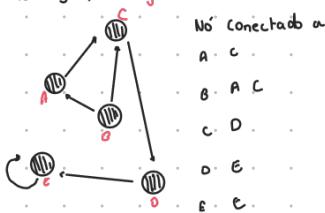


REPRESENTAÇÃO

Num grafo não-dirigido. → relação de adjacência simétrica



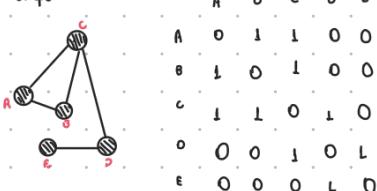
Num grafo dirigido



Matriz de Adjacência

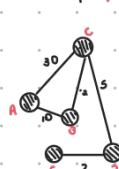
- Matriz quadrada e binária → existe simetria

Grafo não-direcionado



	A	B	C	D	E
A	0	1	1	0	0
B	1	0	1	0	0
C	1	1	0	1	0
D	0	0	1	0	1
E	0	0	0	1	0

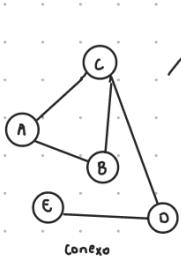
Grafo ponderado



	A	B	C	D	E
A	0	10	30	0	0
B	10	0	-2	0	0
C	30	-2	0	5	0
D	0	0	5	0	3
E	0	0	0	3	0

BUSCA EM PROFUNDIDADE

Começa em um vértice v_0 e explora a maior quantidade possível de vértices (navegando pelas arestas v_i, v_j) seguindo a regra: a partir do primeiro vértice visitar-se, recursivamente, seus vizinhos.



vértice	anterior	visitados
A	-	[A]
B	A	[A, B]
C	B	[A, B, C]
D	C	[A, B, C, D]
E	D	[A, B, C, D, E]

O caminho percorrido durante a busca só visitar os vértices

Deep-First Search - A partir de um vértice v_0 , marque v como explorado e adicioná-lo ao conjunto S de visitados. Para cada par (v, u) incidente de v , se u não estiver no conjunto S iniciar uma nova busca recursiva repitindo o processo.

Para que usar?

- Encontrar componentes conexos
- Encontrar componentes fortemente conexos
- verificar se um grafo é acíclico
- realizar ordenação topológica

PROFOUNDIDADE →

CORES

- branco (vértice ainda não visitado)
- cinza (vértice visitado que está sendo processado)
- preto (ja visitado e processado)

Tempo de descoberta → momento em que um vértice branco fica cinza.

Tempo de fim → momento em que um vértice cinza se torna preto.

Vértice anterior → salvar o vértice anterior a um vértice v.

Tipos de arestas

aresta de árvore → descobre um novo vértice

aresta de avanço → conecta um vértice v_1 a um

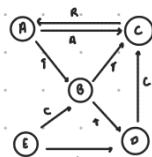
antecessor v_2 em uma árvore de

busca em profundidade

aresta de avanço → não pertencem a árvore de busca em profundidade mas conectam um vértice v_1

a um sucessor v_2 na árvore

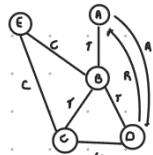
arestas de cruzamento → conectam vértices de árvores diferentes de busca



DPS-CORES

Nó	Anterior	TDesc	TFin	Cor
A	-	1	8	2
B	A	2	7	2
C	B	3	4	2
D	B	5	6	2
E	-	9	10	2

relações de árvore



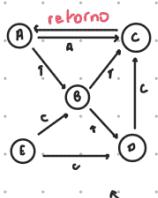
PROFOUNDIDADE

CICLOS

Uma forma de verificar se um grafo é assíndico:

Grafos não-direcionados só possuem arestas de retorno e árvore.

Verificar presença de arestas de retorno.

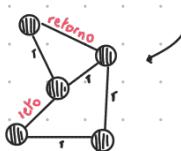


Em grafos dirigidos uma aresta de retorno é uma aresta que liga o vértice atual a um vértice de cor cinza na árvore de busca.

Se o vértice par pra preto então temos uma aresta de cruzamento ou avanço.

Verificar existência de arestas de retorno.

Em grafos não-dirigidos, uma aresta de retorno será uma aresta que liga o vértice atual a um vértice marcado como cinza, porém a aresta não pode ter sido usada para chegar ate o vértice atual.



PROFOUNDIDADE

COMPONENTES CONEXOS

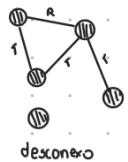
Em grafos não-direcionados, um grafo G será conexo se existir um caminho entre qualquer par de vértices



conexo

Um componente conexo é um subgrafo conexo

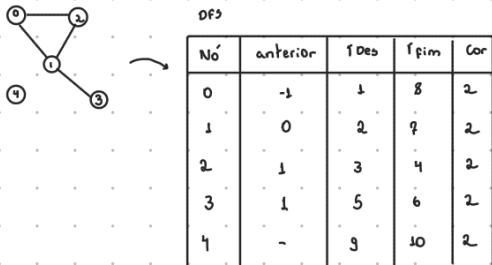
maximal de G .



desconexo

componentes:

cada árvore produz um componente conexo.



Agora também manteremos um arranjo para guardar os vértices de um componente conexo. Um componente será formado quando a recursividade voltar para o vértice inicial de chamada da busca e terá todos os vértices que estiverem presentes na árvore. Um novo componente será formado quando uma nova pesquisa for iniciada por um vértice ainda não visitado.

PROFOUNDIDADE FORTEMENTE CONEXOS



Um gráfico direcionado é fortemente conexo quando existem caminhos de ida e de volta entre quaisquer pares de vértice.

Um componente fortemente conexo de G é um subgráfo fortemente conexo máximo de G .

1- Chamar a busca em profundidade em G para calcular os tempos de término de cada vértice.

2- Obtemos G^t

3- chamarmos busca em profundidade em G^t mas visitando os vértices, recursivamente, a partir daquele com maior tempo de término.

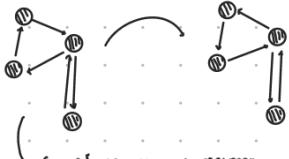
4- Cada uma das árvores de busca obtidas na busca em G^t terá ser um componente fortemente conexo.

Gráfico transposto

Um gráfico transposto de $G = (V, E)$ é um $G^t = (V, E^t)$ tal que E^t possui as direções inversas de E .

$G = (V, E)$

$G^t = (V, E^t)$

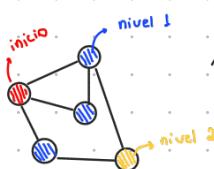


G e G^t possuem os mesmos componentes fortemente conexos

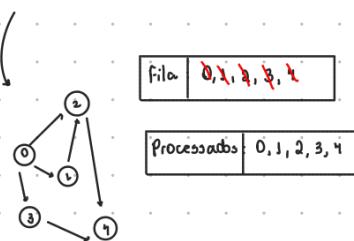
BUSCA EM LARGURA

Breadth-first search (BFS)

O algoritmo começa em um vértice e explora, inicialmente, todos os seus vizinhos, depois os vizinhos dos vizinhos e assim sucessivamente.

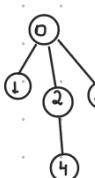


Só podemos visitar um vizinho por vez, primeiro visitaremos todos os vizinhos do nó atual para só depois visitarmos os vizinhos dos vizinhos.



A cada vértice visitado, colocaremos numa fila seu vizinhos, para que sejam visitados apenas após todos aqueles que já estão na fila.

A árvore produzida com as arestas a partir das quais visitamos cada vértice pela primeira vez permite a descoberta de menor caminho.



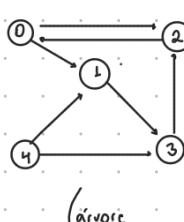
Vista da árvore

FILA → Precisamos 'enfileirar' os vizinhos de cada vértice visitado

LARGURA APLICAÇÕES

- encontrar o menor caminho entre 2 vértices (em termos de arestas percorridas).
- Encontrar componentes conexos
- Verificar se um grafo é bipartido
- Auxiliar no problema de fluxo máximo.

LARGURA MENOR CAMINHO

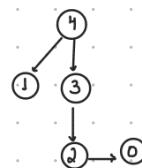


início v₀, fim v₀

Fila	4	1	3	2	0
------	---	---	---	---	---

Processado	4	1	3	2	0
------------	---	---	---	---	---

árvore



Anteriores dita
o caminho perco-
rrido.

Nº	Ant	Dist
0	2	3
1	4	1
2	3	2
3	4	1
4	-	0

GRAFOS EURELIANOS

trajeto
eureliano

Passa por cada vértice somente
uma vez

Um grafo G é dito eureliano se
existir um trajeto eureliano.

Um grafo G é semi-eureliano se
existir um trajeto eureliano.

Ciclo
Eureliano

é um caminho eureliano que começa e
termina no mesmo vértice, ou seja, um trajeto
fechado.

CONDICÃO
SUFICIENTE

1. Um grafo conexo é eureliano se e
sormente se todos os seus vértices
tiverem grau par.

2. Um grafo conexo é não-eureliano se
dois ou mais vértices de grau ímpar.

3. Um grafo conexo é semi-eureliano se e somente se
existirem exatamente dois vértices de grau ímpar.

