

Grafos

Aula 1

Um grafo é uma estrutura de dados composta por vértices e arestas. De forma um pouco mais rigorosa, um grafo é uma tripla ordenada formada por um conjunto de vértices V , um conjunto de arestas E e uma função delta que relaciona cada par de vértices por meio de uma das arestas. $G = (V, E, \delta)$. Por simplicidade, omitimos a função δ .

Ordem

A ordem de um grafo G , denotada por $\text{ordem}(G)$ nada mais é que a quantidade vértices de G .

Tamanho

O tamanho de um grafo G , denotado por $\text{tam}(G)$, é a quantidade de arestas de G .

Grau de um vértice

O grau de um vértice, denotado por $d(v)$ é definido pela quantidade de arestas que incidem sobre ele.

Um vértice de grau 0 é chamado de vértice *isolado* e um vértice de grau 1 é chamado de *folha*.

Grafo Vazio

Como o próprio nome sugere, é um grafo que não possui nenhum vértice, logo também não possui arestas.

Grafo Trivial

Um grafo é dito trivial, ou singular, quando possui apenas um vértice e nenhuma aresta.

Regular

Um grafo é dito regular, se todos os seus vértices têm o mesmo grau. Se todos os vértices têm grau k , dizemos que ele é k -regular.

Completo

Um grafo é dito completo se cada um de seus vértices está ligado a todos os outros demais vértices. O grafo completo de ordem n é chamado de

Kn.

Subgrafo e Supergrafo

Sendo $G = (V, E)$ e $G' = (V', E')$, onde $V' \subseteq V$ e $E' \subseteq E$, dizemos que G' é subgrafo de G . Podemos também dizer que G é super-raro de G' . Se $G' \neq G$, então dizemos que G' é subgrafo próprio de G e que G é um supergrafo próprio de G' .

Aula 2

Subgrafo induzido por arestas

Sendo $G = (V, E)$ e $E' \subseteq E$. O subgrafo de G induzido por E' , denotado por $G[E']$, é o subgrafo de G que contém todos os vértices de G mas apenas as arestas de E' .

Subgrafo induzido por vértices

Seja $V' \subseteq V$. O subgrafo de G induzido por V' , denotado por $G[V']$, é o grafo obtido a partir de G removendo todos os vértices que não pertencem a V' e naturalmente todas as arestas incidentes nos vértices removidos.

A ideia de indução apresentada gira em torno de gerar um grafo a partir de outro, retirando-se do grafo de origem vértices ou arestas.

Adjacência

O conceito de adjacência tem a ver com proximidade. Dizemos então que dois vértices são adjacentes se existe uma aresta entre eles.

Já as arestas, serão ditas adjacentes se elas incidirem num mesmo vértices.

Grau máximo

O grau máximo de um grafo G , denotado por $\Delta(G)$, é o grau de um vértice que tem o maior grau em G .

Grau mínimo

O grau mínimo de um grafo G , denotado por $\delta(G)$, é o grau de um vértice que tem o menor grau em G .

Obs.: Um grafo regular pode ser definido como um grafo em que o grau máximo e mínimo são iguais.

Laço

Um laço é uma aresta que incide duas vezes num mesmo vértice.

Arestas Múltiplas

Se duas arestas incidem num mesmo par de vértices, dizemos que elas são arestas múltiplas.

Grafos não simples

Dizemos que um grafo é não simples quando ele possui laços ou arestas múltiplas.

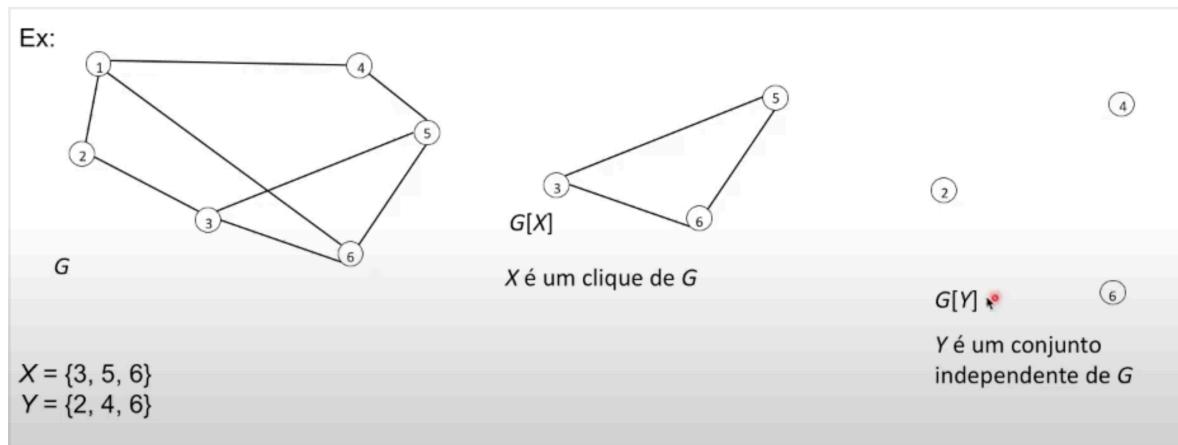
Clique

Seja X um conjunto de vértices de um grafo G . Dizemos que X é um clique de G se os vértices de X são todos dois-a-dois adjacentes. Note que $G[X]$ é um grafo completo.

Independente ou estável

Se os vértices de X são todos dois-a-dois não adjacentes dizemos que X é um conjunto independente (ou estável) de vértices de G . Observe que $G[X]$ contém apenas vértices isolados.

Exemplo de clique e independente:



Aula 3

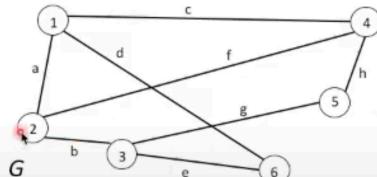
A aula 3 fala sobre as formas mais frequentemente utilizadas para representação de grafos.

Lista de Adjacências

Essa estrutura de dados, lista de adjacências (LA) de um grafo é

constituída de um vetor com uma posição para cada vértice do grafo. Nesse vetor, a posição correspondente ao vértice i aponta para uma lista ligada que contém todos os vértices adjacentes ao vértice i .

Ex:



1	→ 2 → 4 → 6
2	→ 1 → 3 → 4
3	→ 2 → 5 → 6
4	→ 1 → 2 → 5
5	→ 3 → 4
6	→ 1 → 3

LA de G

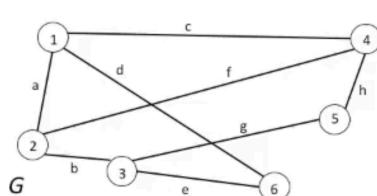
Obs.: Por questão de convenção do professor, a lista ligada será sempre ordenada, apesar da definição de lista de adjacências não exigir isso.

Obs2.: Com essa estrutura de dados, conseguimos remontar o grafo a partir da estrutura de dados, mas perdemos as informações de rótulos das arestas, assim dizemos que a LA é uma estrutura vértice-orientada.

Listas de Incidências

A lista de incidências(LI) de um grafo é constituída de um vetor com uma posição para cada vértice do grafo. Nesse vetor, a posição correspondente ao vértice i aponta para uma lista ligada que contém todas as arestas incidentes no vértice i .

Ex:



1	→ a → c → d
2	→ a → b → f
3	→ b → e → g
4	→ c → f → h
5	→ g → h
6	→ d → e

LA de G

Obs.: Também manter as arestas ordenadas por convenção do professor, apesar da definição não exigir isso.

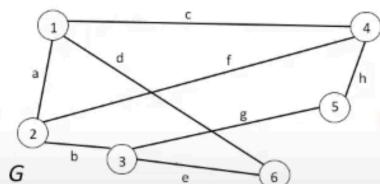
Obs2.: Conseguimos remontar completamente um grafo a partir da LI e por isso dizemos que a LI é uma estrutura aresta-orientada.

Quando necessitarmos das características positivas das duas formas de listas, podemos fazer uma combinação das estruturas onde em cada posição i do vetor apontaremos não para uma, mas sim para duas listas ligadas: uma de adjacências e outra de incidências.

Matriz de adjacências

A matriz de adjacências (MA) de um grafo é constituída de uma linha e uma coluna para cada vértice do grafo. A posição (i,j) da matriz indica se o vértice i é adjacente ao vértice j .

Ex:



	1	2	3	4	5	6
1		1		1		1
2	1		1	1		
3		1			1	1
4	1	1			1	
5			1	1		
6	1	1				

MA de G

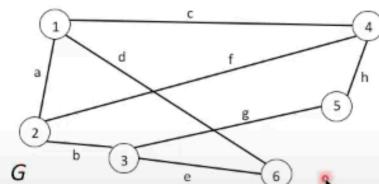
Obs.: A matriz de adjacências é uma matriz simétrica, ou seja, a linha i é igual a coluna i . Pode-se notar na imagem acima que a coluna 1 é igual à linha 1 e assim sucessivamente, ou seja, ela é igual à sua transposta.

Obs2.: Da mesma forma que a lista de adjacências temos aqui a perda de informação dos rótulos das arestas. Uma forma de corrigir isso seria guardar em vez de 0's e 1's, o rótulo da aresta.

Matriz de incidências

A matriz de incidências (MI) de um grafo é constituída de uma linha para cada vértice e uma coluna para cada aresta do grafo. A posição (i,j) da matriz indica se a aresta j incide no vértice i .

Ex:



	a	b	c	d	e	f	g	h
1	1		1	1				
2	1	1					1	
3		1			1		1	
4			1			1		1
5							1	1
6				1	1			

MI de G

Memória Necessária

Vamos denotar por n a quantidade de vértices e por m a quantidade de arestas de um grafo. A quantidade de memória requerida por cada uma das estruturas de dados é:

LA	$\Theta(n + m)$
LI	$\Theta(n + m)$
MA	$\Theta(n^2)$
MI	$\Theta(n.m)$

Obs.: Teta significa que será exatamente esse valor, ou seja, dessa ordem. No caso, para salvarmos as listas ligadas precisaremos de exatamente $2 * m$, pois uma aresta que liga os vértices UV fará com que V apareça na lista de U e que U apareça na lista de V. Ou seja, cada aresta é "duplicada". Isso acontece tanto nas Listas de adjacências como nas listas de incidências.

Grafos Esparsos

Seja C uma classe de grafos. Se $m \in O(n)$ (ou seja, o número de arestas é de no máximo, menor ou igual, da ordem de N), dizemos que C é uma classe de grafos esparsos, ou ainda, uma classe de grafos com poucas arestas. Exemplo: Arvores.

Grafos Densos

Se $m \in \Theta(n^2)$ dizemos que C é uma classe de grafos densos.

Nesses casos, a quantidade de memória requerida é:

	Grafos esparsos	Grafos densos
LA	$\Theta(n)$	$\Theta(n^2)$
LI	$\Theta(n)$	$\Theta(n^2)$
MA	$\Theta(n^2)$	$\Theta(n^2)$
MI	$O(n^2)$	$O(n^3)$

Obs.: A linha na foto sobre MI também é $\Theta()$, mas o professor errou colocando $O()$.

Obs2.: Existem classes de grafos que não se encaixam nos conceitos nem de Esparsos nem de Densos.

Tempo para operações

Corresponde ao tempo requerido para realizar algumas operações nessas estruturas de dados.

	LA	LI	MA	MI
Verificar se dois vértices u e v são adjacentes	$O(d(u)+d(v))$	$\Theta(d(u).log(d(u)))$	$O(1)$	$O(m)$
Determinar o grau de v	$\Theta(d(v))$	$\Theta(d(v))$	$\Theta(n)$	$\Theta(m)$
Inserir um novo vértice	$\Theta(n)$	$\Theta(n)$	$\Theta(n^2)$	$\Theta(n.m)$
Inserir uma nova aresta (u, v)	$O(d(u)+d(v))$	$O(d(u)+d(v))$	$O(1)$	$\Theta(n.m)$
Remover um vértice	$\Theta(n)$	$\Theta(n)$	$\Theta(n^2)$	$\Theta(n.m)$
Remover a aresta (u, v)	$O(d(u)+d(v))$	$O(d(u)+d(v))$	$O(1)$	$\Theta(n.m)$

Obs.: Para decidirmos qual estrutura de dados usar para representar o grafo devemos ver qual operação será mais utilizada(inserir e remover vértices por exemplo) e levar em consideração o tempo e espaço que cada estrutura leva para realizar essas operações e ai decidir qual será a mais econômica das estruturas de dados.

Obs2.: Podemos também fazer algum pre-processamento para melhorarmos o desempenho das nossas operações. Exemplo: Salvar no vetor de vértices o grau de cada vértice e assim o tempo para saber o grau de v seria constante sempre.

Aula 4

Percursos em grafos

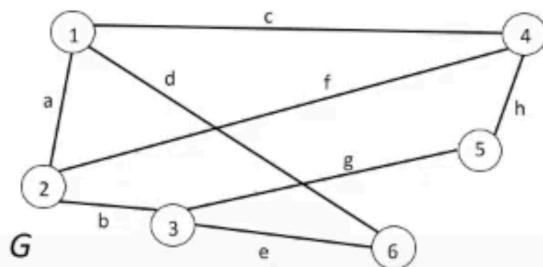
Passeio

Um passeio é uma sequência de arestas de um grafo tal que:

- Se duas arestas distintas são consecutivas no passeio então elas são adjacentes no grafo.
- Se três arestas distintas(a_1, a_2 e a_3) são consecutivas no passeio então a_3 não pode incidir no mesmo vértice que a_1 e a_2 incidem.

A primeira regra significa que deve ter continuidade no caminho, e a segunda regra significa que a aresta a_3 não pode (ao mesmo tempo) incidir no mesmo vértice que a_1 e a_2 incidem, ou seja, no vértice que liga a_1 e a_2 . (Lembrando que estamos trabalhando com grafos simples, sem laços e sem arestas múltiplas). (É permitido usar a mesma aresta para ir e voltar).

Ex:



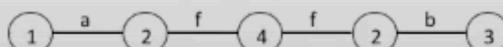
(a, f, g) não é passeio

(a, f, b) não é passeio

(a, f, c, a, b) é passeio

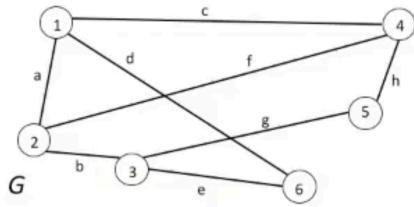


(a, f, f, b) é passeio

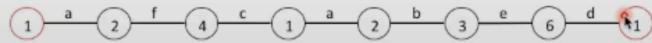


Passeio fechado

É um passeio (ou seja, segue as regras acima), mas que tem a característica de iniciar e terminar no mesmo vértice.



Passeio fechado: (a, f, c, a, b, e, d)

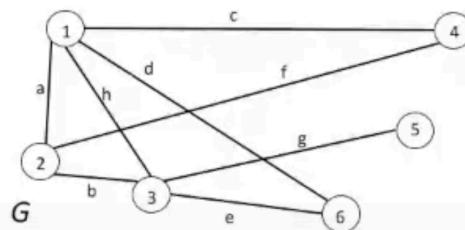


Trilhas

As trilhas também são um tipo específico de passeio, mas ela não permite que arestas sejam repetidas.

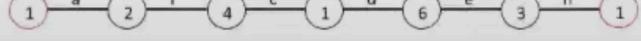
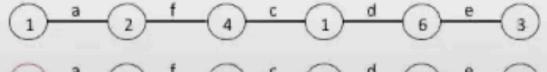
Trilha fechada(ou ciclo)

É uma trilha que começa e termina no mesmo vértice.



Trilha: (a, f, c, d, e)

Trilha fechada: (a, f, c, d, e, h)



Trilha de Euler

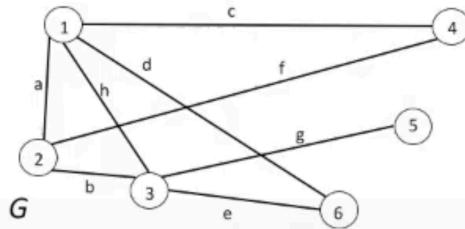
A trilha de Euler é uma trilha em um grafo que passa por todas as arestas.

Caminho

O caminho é uma trilha que não passa mais de uma vez pelo mesmo vértice.

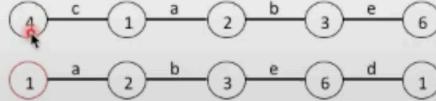
Caminho fechado(ou circuito)

É um caminho que começa e termina no mesmo vértice.



Caminho: (c, a, b, e)

Círcuito: (a, b, e, d)



Caminho Hamiltoniano

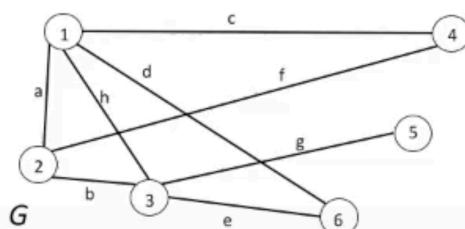
É um caminho onde se passar por todos os vértices de um grafo.

Comprimento de passeios

O comprimento de um passeio é a quantidade de elementos da sequencia de arestas que compõem o passeio.(Não importa se o elemento é repetido)(ou seja, é a quantidade de arestas do caminho)

Cintura

A cintura de um grafo G , denotada por $G(G)$, é o comprimento de um caminho mais longo de G .



Caminho mais longo: (g, e, d, c, f)

$G(G) = 5$



OBS.: Todo caminho é uma trilha mas nem toda trilha é um caminho. Toda trilha é um passeio mas nem todo passeio é uma trilha.

Quanto mais a direita mais específico:

Passeio -> Trilha -> Caminho

Aula 5

Conexidade

Um grafo é dito conexo se para todo par de vértices do grafo sempre existe um caminho entre eles.(Isso não significa que você vai ter uma aresta ligando cada par de vértices mas que é possível percorrer uma série de arestas até chegar no destino).



Obs.: O grafo vazio e os grafos triviais são conexos!

Como testar a conexidade de um grafo

Para fazer isso, escolhemos um vértice e a partir dele percorremos em todas as direções possíveis até chegar em todos os outros vértices. Caso não seja possível alcançar um vértice, o grafo não é conexo.

Algoritmo:

Algoritmo Conexo

Entrada: um grafo G com n vértices e m arestas

Saída: *Sim*, se G é conexo; *Não*, caso contrário para $i = 2$ até n

 visitado[i] = falso

 visitado[1] = verdadeiro

 crie uma fila F com n posições

 insira o vértice 1 em F

 enquanto F não estiver vazia

 remova de F obtendo u

 para cada vértice w adjacente a u

 se visitado[w] = falso

 visitado[w] = verdadeiro

 insira w em F

 Se existir vértice v tal que visitado[v] = falso

 devolva *Não*

 se não

 devolva *Sim*

OBS.: O ALGORITMO CONEXO REQUER TEMPO $O(n + m)$ E ESPAÇO

$\Theta(n)$

Obs.: O algoritmo utiliza uma fila por fins didáticos, para fim de eficiência deveria ser utilizada uma pilha.

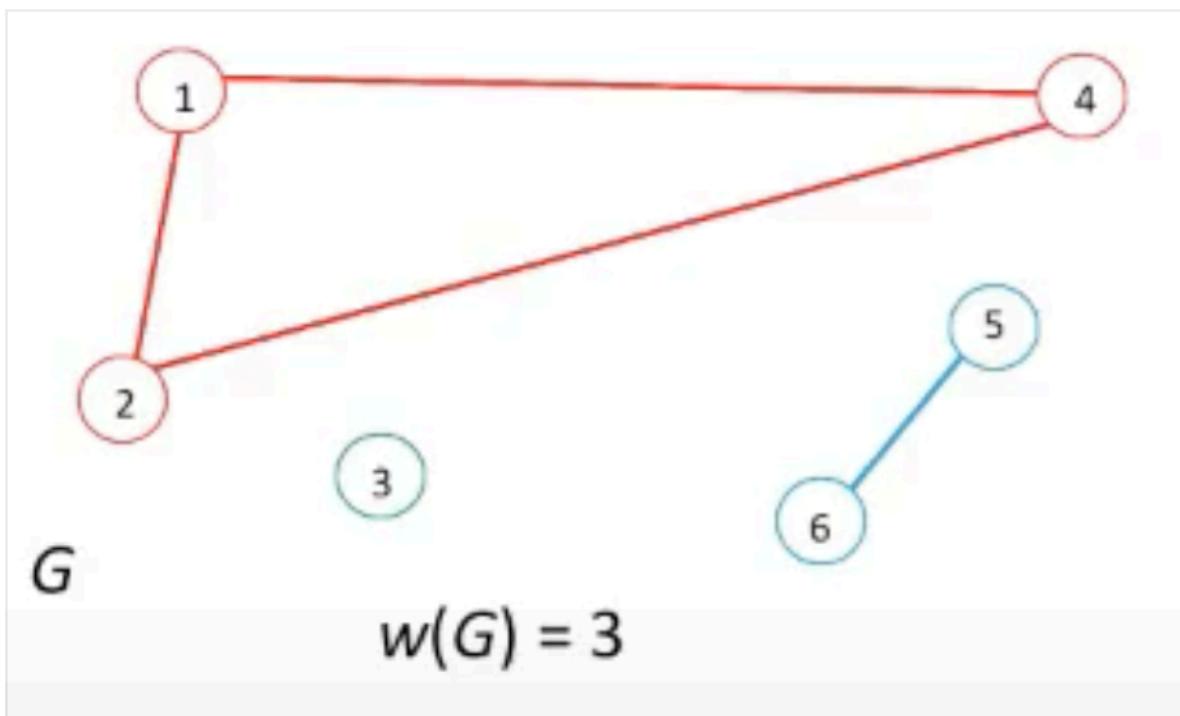
Componente Maximal

Uma componente conexa de G é um subgrafo não vazio de G que é conexo e maximal.(NÃO ENTENDI O QUE SIGNIFICA MAXIMAL)

A quantidade de componentes conexas de G é denotada por $w(G)$.

(Pelo que entendi, um subgrafo maximal é um subgrafo de G onde foi possível pegar todo o grafo conexo daquele trecho).

Exemplo:



Outro exemplo de componente maximal, pelo que entendi, seria {6,5}

Aula 6

Aresta Conexidade

Esse problema tem a ver com a seguinte pergunta: Quantas arestas podemos retirar de um grafo e ele ainda continuar conexo?

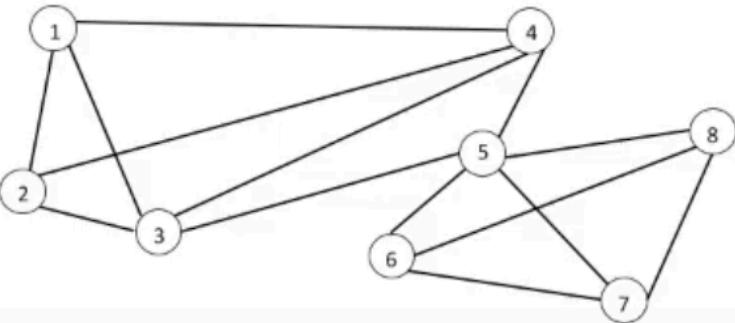
Contextualizando: Levando em consideração o sistema de distribuição brasileiro, quantas conexões de energia poderiam ser desfeitas e ainda assim a energia chegar a todos os estados?

Obs.: Atualmente, apenas o Amapá não faz parte de toda a rede de distribuição de energia brasileira e o estado no inicio de novembro ficou apagado.(Hoje é dia 15 de novembro de 2020 mas não sei se o problema ja foi resolvido).

Definição

Dizemos que um grafo é k-aresta-conexo se é preciso remover pelo menos k de suas arestas para desconta-lo.

Exemplo de grafo 2-aresta-conexo:



Grafo 2-aresta-conexo

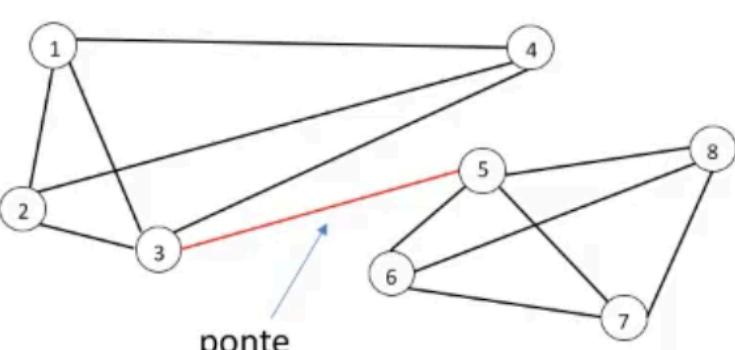
Obs.: Se um grafo é 3-aresta-conexo, ele também é 2-aresta-conexo e 1-aresta-conexo, pois pela definição "é preciso remover pelo menos k" então se temos que tirar pelo menos 3, também temos que tirar pelo menos 2 e pelo menos 1.

Conclusão:

A aresta-convexidade de um grafo G é o maior k tal que G é k -aresta-conexo.

Ponte

Uma ponte é uma aresta cuja remoção aumenta o número de componentes conexas do grafo.



Proposição 1: Uma aresta é ponte se e somente se ela não está contida em nenhum ciclo do grafo.

Proposição 2: Um grafo conexo é 2-aresta-conexo se e somente se ele não contém nenhuma ponte.

Proposição 3: A aresta-conexidade do K_n é $n-1$.

Proposição 4: A aresta-convexidade de um grafo G é no máximo $\delta(G)$

(Grau mínimo de G)

Obs.: A proposição 4 significa que pegando o vertice com menos arestas incidentes, se tirarmos essas arestas desse vertice com o menor grau, o grafo será desconexo.

Verificando se um grafo é k-aresta-conexo

Algoritmo:

Algoritmo Aresta-conexo

Entrada: um grafo G com n vértices e m arestas e um inteiro positivo k

Saída: Sim, se G é k -aresta-conexo; Não, caso contrário

```
se  $k > \delta(G)$ 
    devolva Não e pare
se  $k = 1$ 
    devolva conexo( $G$ ) e pare
para  $i = 1$  até  $m$ 
     $G' = G - \{\text{aresta } i\}$ 
    se Aresta-conexo( $G'$ ,  $k - 1$ ) = Não
        devolva Não e pare
    devolva Sim
```

Obs.: O algoritmo Aresta-conexo utiliza o algoritmo da aula passada, conexo(G).

Obs2: O algoritmo Aresta-Conexo requer tempo $O(m^k)$ e espaço $O(n)$

Aula 7

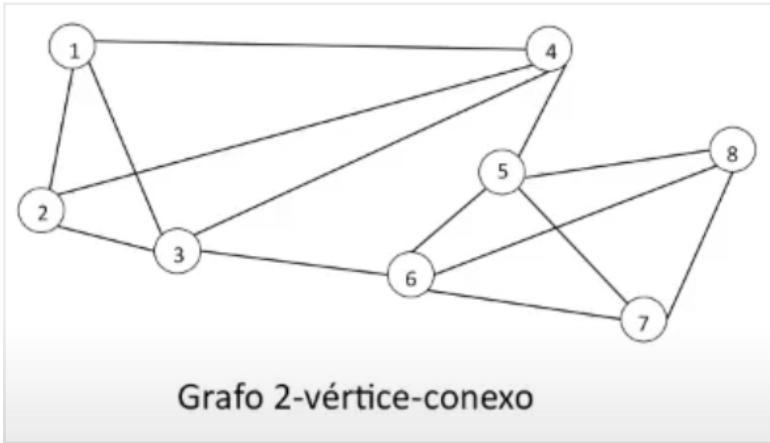
Vértice-Conexidade

Similar a questão da aresta conexidade, na questão da vidente-convexidade queremos saber quantos vértices podemos retirar de um grafo e ele continuar conexo.

Definição

Dizemos que um grafo é k -vérteice-conexo (ou simplesmente k -conexo) se é preciso remover pelo menos k de seus vértices para desconectá-lo.

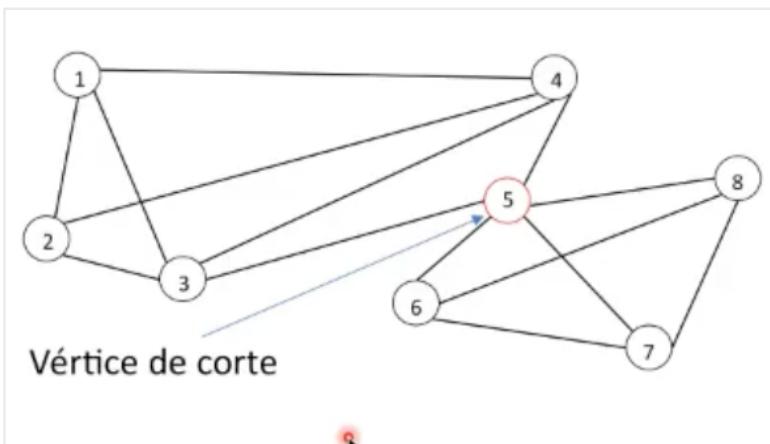
Exemplo:



A vértice-convexidade de um grafo G é o maior k tal que G é k -conexo. O grafo de exemplo tem vértice-convexidade 2.

Vértice de corte

Um vértice de corte é um vértice cuja remoção aumenta o número de componentes conexas de um grafo.(versão em vertice de uma ponte).



Proposição 1: Um grafo conexo é 2-conexo se e somente se ela não contém nenhum vértice de corte.

Não é possível desconectar um grafo completo removendo alguns de seus vértices. Dizemos que a vértice-convexidade dos grafos completos é infinita. Isso porque tirando um vértice de K_n (um grafo completo), ficamos com um K_{n-1} (que também é completo).

Proposição 2: Se G não é completo então a vértice-convexidade de G é no máximo $\delta(G)$

Como saber se um grafo é K -vértice-conexo?

Algoritmo Vértice-Conexo:

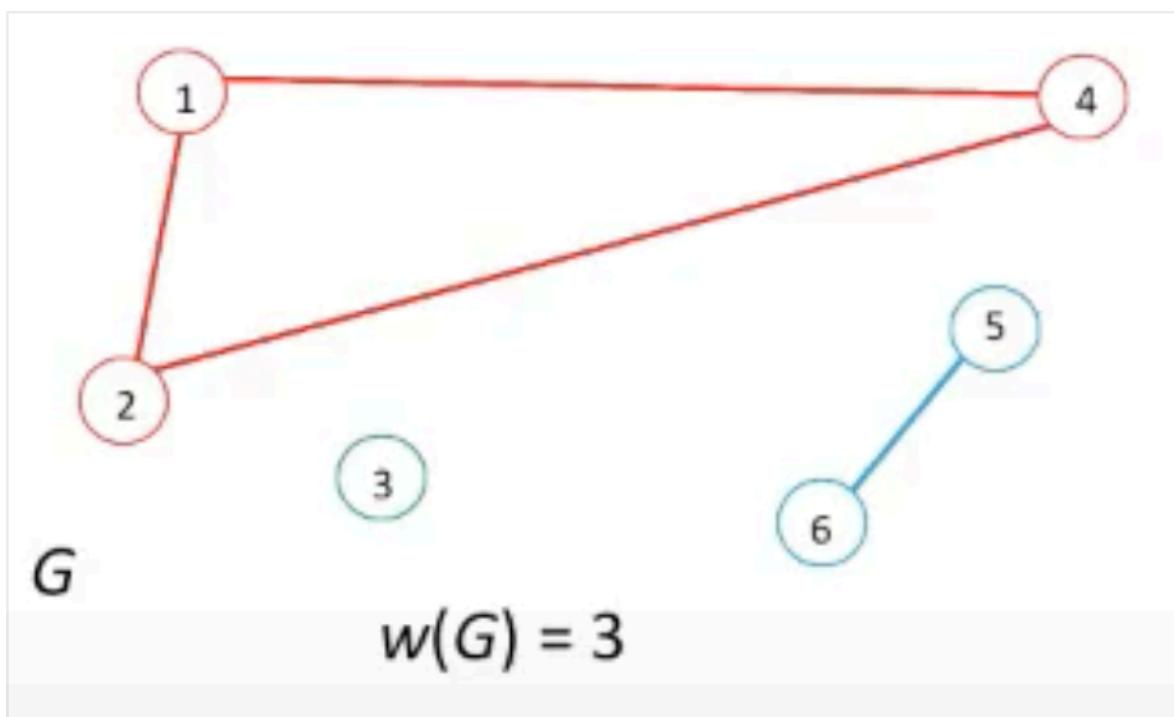
Algoritmo Vértice-conexo

Entrada: um grafo G com n vértices e m arestas e um inteiro positivo k
Saída: Sim, se G é k -vértice-conexo; Não, caso contrário

```
se  $G$  é completo
    devolva Sim e pare
se  $k > \delta(G)$ 
    devolva Não e pare
se  $k = 1$ 
    devolva conexo( $G$ ) e pare
para  $i = 1$  até  $n$ 
     $G' = G - \{\text{vértice } i\}$ 
    se Vértice-conexo( $G', k - 1$ ) = Não
        devolva Não e pare
devolva Sim
```

Obs.: O algoritmo Vértice-Conexo requer tempo $O(n^k+1)$ e espaço $O(n)$

OBS.: Componente conexa é um “pedaço” solto de um grafo.



Nesse exemplo ai, temos 3 componentes conexas, ou seja, três pedaços.

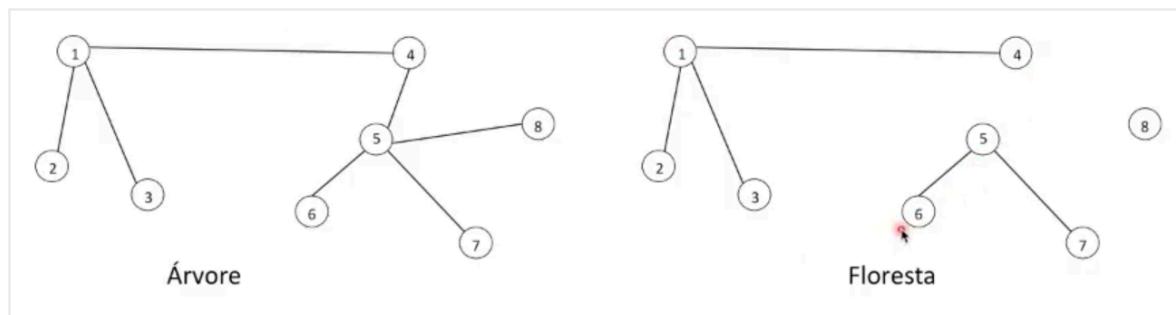
Aula 8

Ciclos em grafos

Árvore

Uma árvore é um grafo conexo e sem ciclos. Um grafo que não contém ciclos é chamado de floresta porque cada uma de suas componentes conexas é uma árvore. Ou seja, uma árvore é um tipo particular de floresta que possui apenas uma componente conexa.

Exemplo:



- **Teorema:** Seja G uma árvore não vazia com n vértices. Então $\text{tam}(G) = n - 1$.
- **Corolário:** Seja F uma floresta não vazia com n vértices. Então $\text{tam}(F) = n - w(F)$. Obs.: $w(F)$ é o número de componentes conexas do grafo.

Como verificar se um grafo é ou não uma floresta?

Algoritmo Floresta:

Algoritmo Floresta

Entrada: um grafo G com n vértices e m arestas

Saída: Sim, se G é floresta; Não, caso contrário

se $m \geq n$ devolva Não e pare

para $i = 1$ até n

visitado $[i]$ = falso, anterior $[i]$ = 0

crie uma fila F com n posições

Enquanto existir vértice i tal que visitado $[i]$ = falso

visitado $[i]$ = verdadeiro

insira o vértice i em F

enquanto F não estiver vazia

remova de F obtendo u

para cada vértice w adjacente a u

se anterior $[u] \neq w$

se visitado $[w]$ = falso

visitado $[w]$ = verdadeiro

anterior $[w]$ = u

insira w em F

se não

devolva Não e pare

devolva Sim

Obs: O Algoritmo Floresta requer tempo $O(n)$ e espaço $\Theta(n)$

Obs.: O algoritmo floresta requer tempo $O(n)$ e espaço $\Theta(n)$.

Aula 9

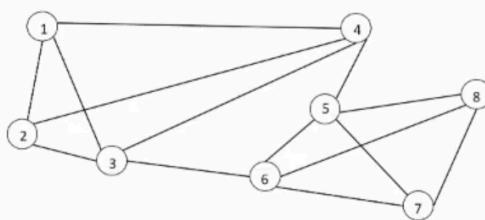
Árvores Geradoras Mínimas

Árvore Geradora

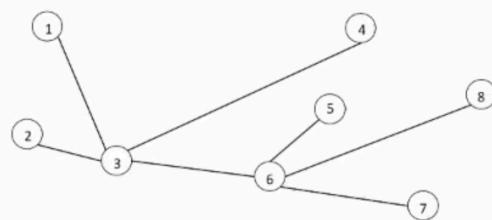
Seja G um grafo conexo. Uma árvore geradora de G é um subgrafo de G que é conexo, acíclico e que contém todos os vértices de G .

Exemplo:

Ex:



G



Árvore geradora de G

Problema da árvore geradora mínima

No problema da árvore geradora mínima(AGM) temos um grafo conexo no qual cada aresta possui um custo e queremos encontrar uma árvore geradora no grafo na qual a soma dos custos das arestas seja o menor possível.

Podemos resolver esse problema usando o Algoritmo de Kruskal. Trata-se de um algoritmo guloso surpreendentemente simples que, se bem implementado, é extremamente rápido.

Nesse algoritmo, colocamos as arestas em ordem crescente de custo e depois escolhermos para a AGM as arestas nessa ordem sem permitir a formação de ciclos.

Apesar do algoritmo ser simples, a demonstração de sua corretude não é trivial.

Algoritmo de Kruskal

Algoritmo de Kruskal

Entrada: um grafo conexo G com n vértices e m arestas no qual cada aresta possui um custo
Saída: um conjunto de arestas T tal que $G[T]$ é uma AGM de G

```
coloque as arestas de  $G$  em ordem crescente de custo
 $i = 1$ 
 $T = \{\}$ 
enquanto  $|T| < n - 1$ 
    Se  $G[T \cup \{\text{aresta } i\}]$  é floresta
         $T = T \cup \{\text{aresta } i\}$ 
     $i++$ 
devolva  $T$ 
```

Obs.: Uma implementação ingênuia do Algoritmo de Kruskal tem complexidade temporal $O(mn)$.

Se utilizarmos uma estrutura de dados conhecida como union-find, podemos implementar o Algoritmo de Kruskal de modo que sua complexidade temporal seja $O(m\log n)$.

(Essa estrutura basicamente pinta da mesma cor vértices que estão conectados). Esse método de pintura substitui o algoritmo de verificar se um grafo tem ciclo ou não, basicamente não permitimos que sejam ligados dois vértices que tem a mesma cor, pois caso isso aconteça, um ciclo é formado.

Aula 10

Algoritmo de Prim

Podemos resolver o problema da AGM usando o Algoritmo de Prim. Tal algoritmo também é guloso, simples e rápido. Antes de descrevê-lo, precisamos de uma definição.

Seja Z um conjunto de vértices. Dizemos que uma aresta uv está na fronteira de Z se $u \in Z$ e $v \notin Z$ ou $v \in Z$ e $u \notin Z$.

Exemplo:

Ex:



Iniciamos o Algoritmo de Prim com um conjunto Z de vértices contendo um único vértice. A cada iteração escolhemos a aresta da fronteira de Z que possui o menor custo para fazer parte da AGM. Seja uv a aresta escolhida, com $u \in Z$ e $v \notin Z$. Incluímos o vértice v em Z e iniciamos uma nova iteração.

Algoritmo de Prim

Algoritmo de Prim

Entrada: um grafo conexo G com n vértices e m arestas no qual cada aresta possui um custo
Saída: um conjunto de arestas T tal que $G[T]$ é uma AGM de G

```
 $T = \{ \}$ 
 $Z = \{\text{vértice } 1\}$ 
enquanto existir aresta na fronteira de  $Z$ 
    Seja  $uv$  uma aresta da fronteira de  $Z$  que possua custo mínimo, com  $u \in Z$  e  $v \notin Z$ 
     $T = T \cup \{uv\}$ 
     $Z = Z \cup \{\text{vértice } v\}$ 
devolva  $T$ 
```

Obs.: Uma implementação ingênuia do Algoritmo de Prim tem complexidade temporal $O(mn)$.

Se usarmos uma estrutura de dados conhecida como heap binário para armazenar as arestas da fronteira de Z , podemos implementar o Algoritmo de Prim de modo que sua complexidade temporal seja $O(m\log n)$.

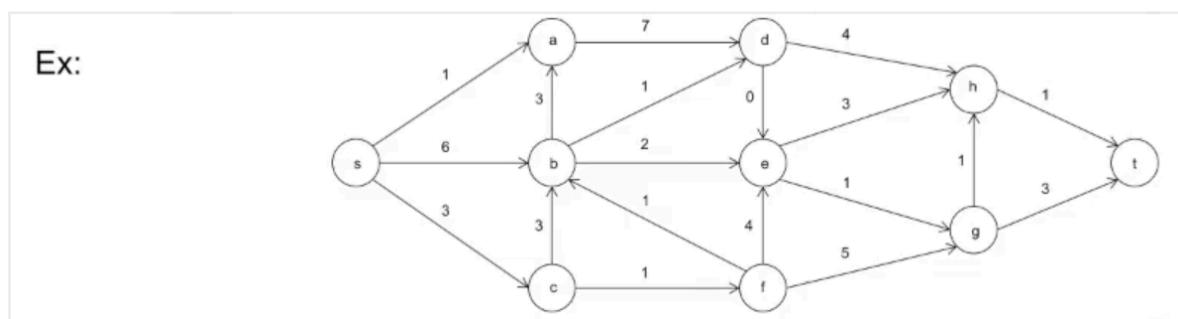
Aula 11

Grafos dirigidos

Um arco é uma aresta dirigida. O arco uv tem origem no vértice u e destino no vértice v .

Um grafo que contém arcos é chamado de grafo dirigido ou digrafo. Dado um grafo dirigido G :

Exemplo:



As definições de passeio dirigido, trilha dirigida e caminho dirigido são análogas às definições de passeio, trilha e caminho. Nos percursos dirigidos, os arcos têm que estar todos na mesma direção.

Problema do caminho mínimo

No Problema do Caminho Mínimo(PCM) é dado um grafo dirigido no qual cada arco e possui um custo C_e , e dois vértices especiais s (origem) e t (destino).

Desejamos encontrar um caminho dirigido de s a t que possua custo mínimo.

Antes de descrever o Algoritmo de Dijkstra, precisamos de uma definição,

Seja Z um conjunto de vértices. Dizemos que um arco uv está na fronteira de Z se $u \in Z$ e $v \notin Z$.

Algoritmo de Dijkstra

O Algoritmo de Dijkstra funciona para arcos com custos não negativos.

Inicialização: Associe a cada vértice v um valor $c(V)$ que indica o custo mínimo de um caminho dirigido de s a v . Faça $c(s) = 0$ e $c(v) = +\infty$, para todo $v \neq s$. Para cada vértice v faça $\text{anterior}(v) = 0$. Defina um conjunto de vértices Z contendo inicialmente apenas s .

Em cada iteração escolha um arco uv da fronteira de Z tal que $c(u) + C_{uv}$ seja mínimo. Faça $c(v) = c(u) + C_{uv}$, $\text{anterior}(v) = u$ e inclua v em Z .

O algoritmo para quando incluirmos t em Z ou quando não houver mais arcos na fronteira de Z . Ao final da execução $c(t)$ indicará o custo mínimo de um caminho dirigido de s a t (obs: se $c(t) = +\infty$ então não existe caminho dirigido de s a t).

Podemos reconstruir o caminho mínimo de s a t usando o vetor anterior.

Complexidade temporal:

Uma implementação ingênuia do Algoritmo de Dijkstra requer tempo $O(nm)$.

Usando heap binário para armazenar os arcos da fronteira de Z , a complexidade cai para $O(m\log n)$.

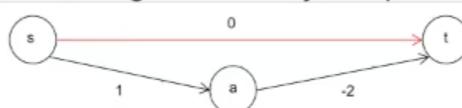
Em grafos esparsos a complexidade é $O(n)$.

Aula 12

Se o grafo tiver arco de custo negativo, o Algoritmo de Dijkstra pode não funcionar.

Exemplo:

Ex:



Nesse exemplo o Algoritmo de Dijkstra vai indicar que o caminho mínimo é usar o arco st , o que é incorreto.

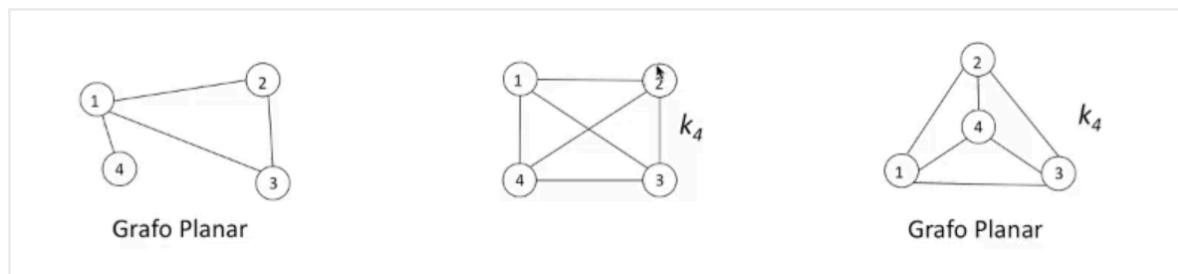
Uma maneira de resolver esse problema seria adicionar 2 ao custo de todos os arcos, fazendo com que não existisse mais arco de custo negativo, e então aplicar o Algoritmo de Dijkstra, certo?

Infelizmente o Algoritmo de Dijkstra pode continuar indicando o caminho errado.

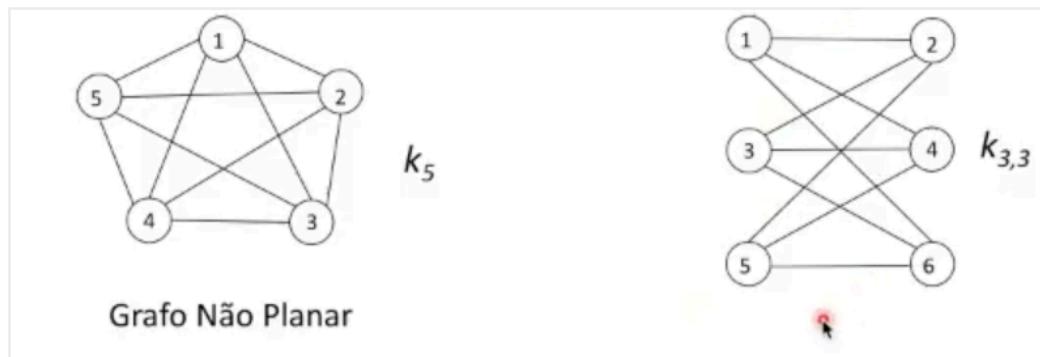
Quando existem arcos de custo negativo, podemos usar o Algoritmo de Bellman-Ford ou o Algoritmo de Floyd-Warshall.

Planaridade

Um grafo é dito planar se ele pode ser representado no plano sem que suas arestas se cruzem.



Obs.: O k_4 tem tanto representação planar como não planar.



Obs.: $K_{3,3}$ não é planar.

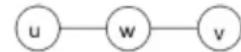
Teorema de Kuratowski

O glorioso Teorema de Kuratowski estabelece uma condição que é necessária e suficiente para que um grafo seja planar. Antes de enunciá-lo precisamos de duas definições.

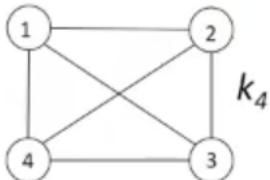
Subdividir uma aresta uv significa criar um novo vértice, digamos w , e substituir a aresta uv pelas arestas uw e wv .



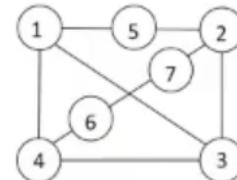
Subdivisão de $uv \Rightarrow$



Dizemos que um grafo G' é uma subdivisão de G se podemos obter G' a partir de G fazendo uma sequência de subdivisões de arestas.



Subdivisão do $K_4 \Rightarrow$



Teorema de Kuratowski: Um grafo é planar se e somente se ele não possui uma subdivisão K_5 nem do $K_{3,3}$.

Teorema: Seja G um grafo planar com n (≥ 3) vértices e m arestas. Então $m \leq 3n - 6$.

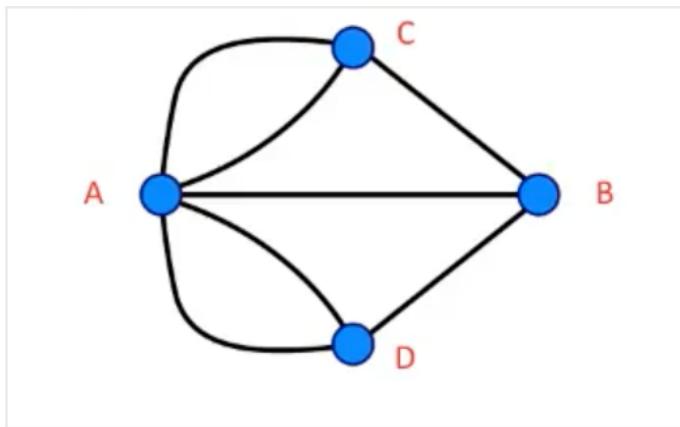
Consequentemente, os grafos planares são esparsos.

Aula 13

Trilhas de Euler

Na cidade de Konigsberg (atualmente Kaliningrado) existiam sete pontes sobre o Rio Pregel ligando duas ilhas ao restante da cidade. Os moradores da cidade se divertiam tentando resolver o seguinte desafio: Realizar um passeio passando por todas as pontes uma única vez.

Podemos modelar esse desafio através de um grafo com quatro vértices, dois deles representando as ilhas e outros dois para as margens, e sete arestas representando as pontes.



Observe que esse grafo é não simples. A solução do problema seria uma trilha contendo todas as arestas do grafo.

Em 1736 Leonhard Euler provou que tal passeio era *impossível*.

Teorema de Euler

Uma Trilha Fechada de Euler(TFE) é uma trilha fechada que contém todas as arestas de um grafo. O Teorema de Euler estabelece uma condição que é necessária e suficiente para que exista uma TFE num grafo num grafo conexo.

Teorema de Euler: Um grafo conexo possui uma TFE se e somente se todos os seus vértices têm grau par.

Uma Trilha Aberta de Euler(TAE) é uma trilha aberta que contém todas as arestas de um grafo.

Corolário: Um grafo conexo possui uma TAE se e somente se ele possui exatamente dois vértices de grau ímpar.

Algoritmo de Fleury

O algoritmo de Fleury permite encontrar uma Trilha de Euler num grafo euleirano. Se todos os vértices têm grau par, a trilha pode começar em qualquer vértice. Se o grafo tem dois vértices de grau ímpar, a trilha deve começar em um dos vértices de grau ímpar.

Seja u o vértice corrente no início de uma iteração. Escolha uma aresta incidente em u que não seja ponte, se houver essa possibilidade, caso contrário escolha uma ponte incidente em u .

Seja uv a aresta escolhida. Insira uv na trilha, remova uv do grafo e considere v o novo vértice corrente.

O algoritmo para quando todas as arestas do grafo estiverem na trilha.

Algoritmo Componentes-Conexas

Entrada: Um grafo G com n vértices e m arestas.

Saída: Quantidade de componentes conexas de G.

Para i = 1 até n

 visitado[i] = falso, anterior[i] = 0

componentes_conexas = 0

Crie uma fila F com n posições

Enquanto existir vértice i tal que visitado[i] = falso

 visitado[i] = verdadeiro

 insira o vértice i em F

 componentes_conexas = componentes_conexas + 1

 enquanto F não estiver vazia

 remova de F obtendo u

 para cada vértice w adjacente a u

 se visitado[w] = falso

 visitado[w] = verdadeiro

 insira w em F

Devolva componentes_conexas

Aula 1 - N2

Coloração de Arestas

É a atribuição de cores para as arestas de um garfo.

Uma restrição adotada é que arestas adjacentes não podem ter a mesma cor, assim temos uma colocação própria ou apropriada de arestas.

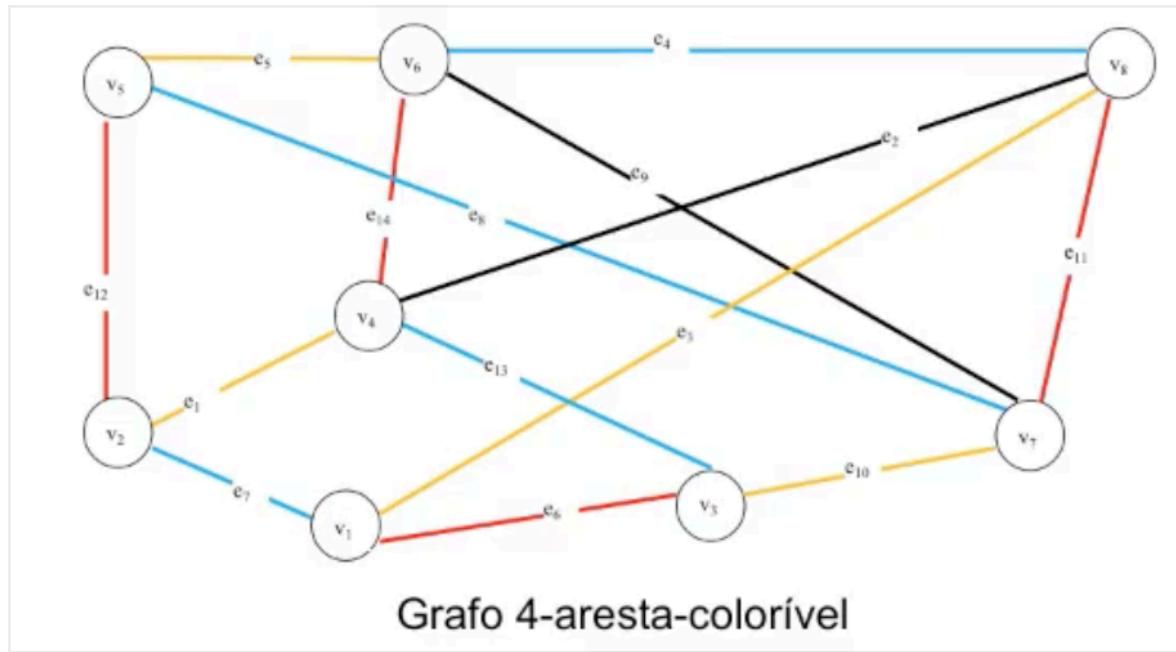
Vários problemas podem ser resolvidos encontrando uma coloração

própria de arestas.

Se uma coloração própria de arestas utiliza k cores dizemos que ela é uma k -aresta-coloração.

Se um grafo admite uma k -aresta-coloração, dizemos que ele é k -aresta-colorível.

Exemplo:



Para verificarmos se um grafo é k -aresta-colorível basta vermos se arestas de cores iguais incidem em um mesmo vértice.

Para saber por exemplo quantas cores precisamos no mínimo, devemos olhar o numero máximo de arestas chegando em um vertice.

Índice Cromático

O índice cromático de G , denotado por $\chi'(G)$, é o menor k tal que G é k -aresta-colorível.

Teorema de Vizing:

$$\Delta(G) \leq \chi'(G) \leq \Delta(G) + 1.$$

O indice cromático de G vai ser sempre maior ou igual a "deltao de G " e menor ou igual "deltao de $G + 1$ "

Deltao de G significa grau máximo do grafo.

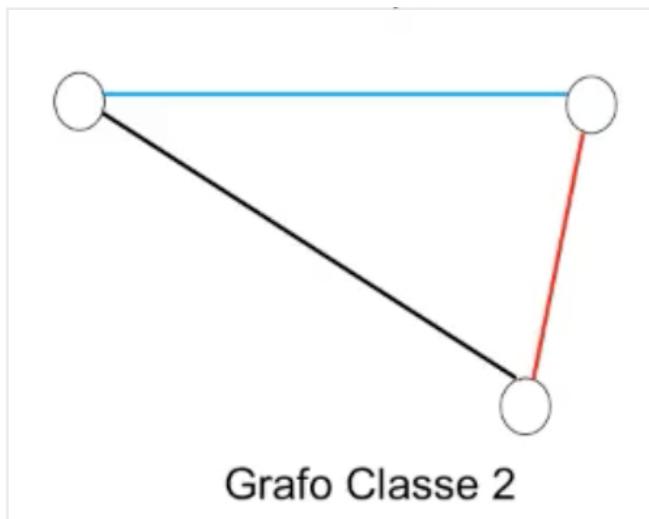
Um grafo é Classe 1 se $\chi'(G) = \Delta(G)$ e é classe 2 se $\chi'(G) = \Delta(G) + 1$

Os grafos planares com $\delta(G) \geq 7$, os grafos bipartidos, os grafos completos de ordem par são classe 1.

Os circuitos de comprimento ímpar e os grafos completos de ordem ímpar são classe 2.

Determinar se um grafo é classe 1 ou classe 2 é um problema NP-Completo. :(

Exemplo:



Coloração equilibrada de arestas:

Seja C uma k -aresta-coloração que utiliza as cores $1, 2, \dots, k$. Seja c_i a quantidade de arestas da cor i ($i = 1, 2, \dots, k$). Dizemos que C é equilibrada se $|c_i - c_j| \leq 1$ ($i = 1, 2, \dots, k$ e $j = 1, 2, \dots, k$).

Ou seja, a diferença entre o número de vezes que uma cor e outra são utilizadas é um.

O índice cromático equilibrado de G , denotado por $X_{eq}'(G)$, é o menor k tal que existe uma k -aresta-coloração equilibrada de G .

Fato: $X_{eq}'(G) \geq X'(G)$

Teorema: Se G é completo então $X_{eq}'(G) = X'(G)$.

Aula 2 - N2

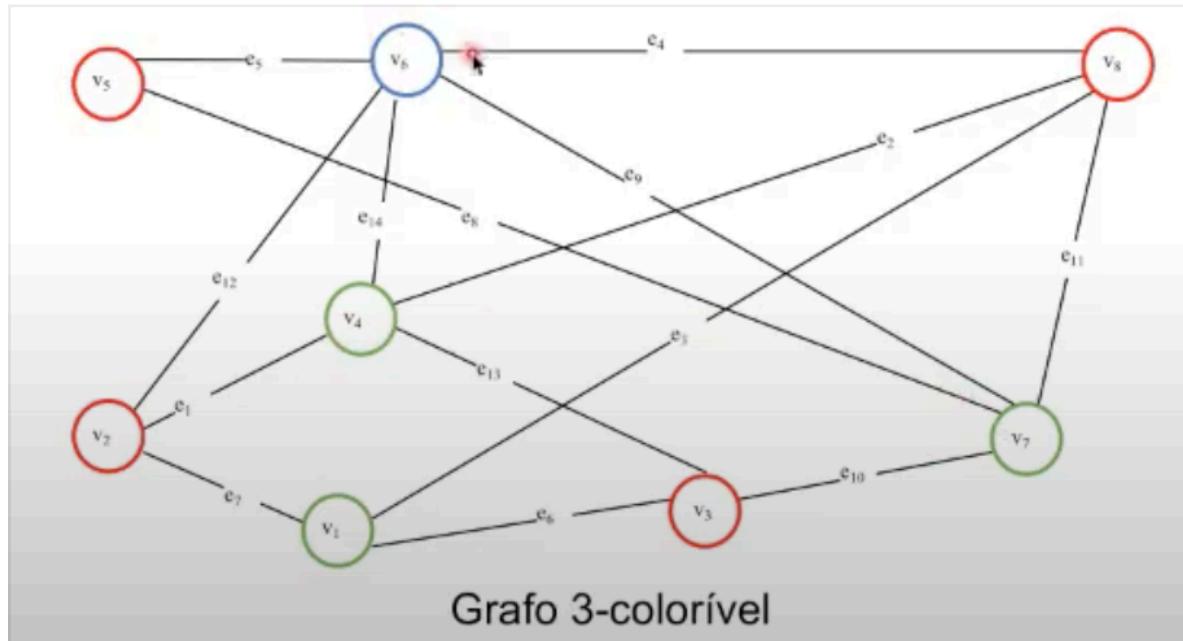
Coloração de Vértices

É uma atribuição de cores a cada um dos vértices de um grafo.

Uma coloração de vértices será dita própria se vértices adjacentes sempre tem cores diferentes.

Se uma coloração própria de vértices utiliza k cores, dizemos que ela é uma k -vertice-coloração ou simplesmente k -coloracao.

Se um grafo admite uma k -coloração dizemos que ele é k -colorível

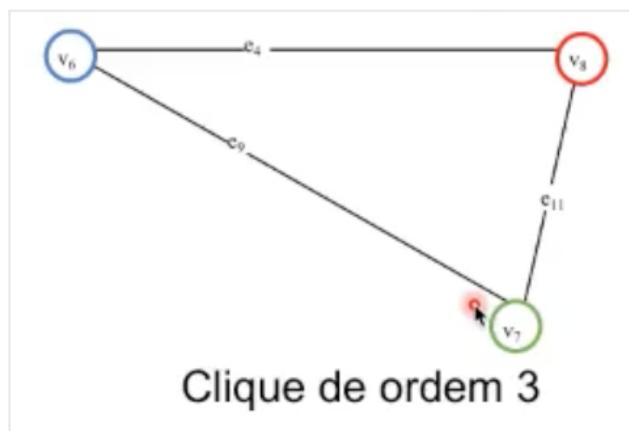


Número Cromático

O numero cromático de G , denotado por $\chi(G)$, é o menor k tal que G é k -colorível.

Um clique de G é um subgrafo completo contido em G .

Teorema 1: Se G contém um clique de ordem n , então $\chi(G) \geq n$.

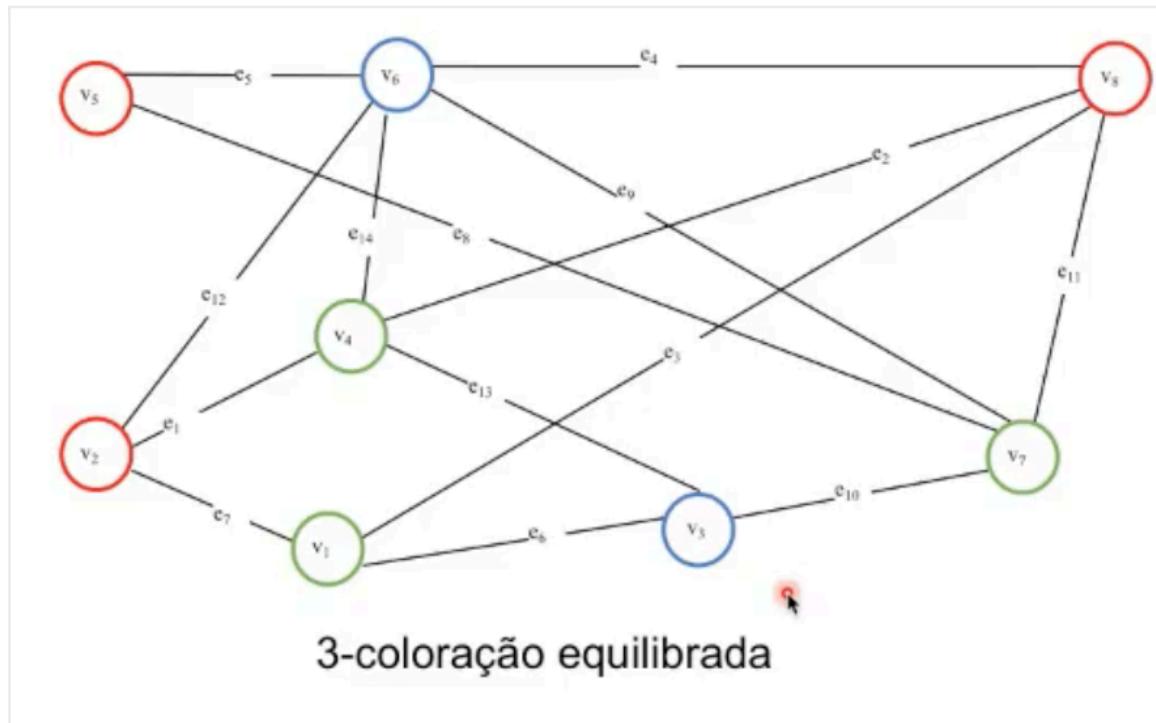


Teorema 2: Se F é uma floresta $\chi(F) \leq 2$.

Teorema 3: $\chi(K_n) = n$

Coloração equilibrada de vértices

Seja C uma k -coloração que utiliza as cores $1, 2, \dots, k$. Seja c_i a quantidade de vértices da cor i ($i = 1, 2, \dots, k$). Dizemos que C é equilibrado se $|c_i - c_j| \leq 1$ ($i = 1, 2, \dots, k$ e $j = 1, 2, \dots, k$).



O número cromático equilibrado de G , denotado por $X_{eq}(G)$, é o menor k tal que existe uma k -coloração equilibrada de G .

Fato: $X_{eq}(G) \geq X(G)$

Teorema das 5 cores

Dado um mapa dividido em regiões, é possível colorir as regiões do mapa de forma que regiões vizinhas sempre tenham cores diferentes usando no máximo 5 cores.

Teorema das 4 cores

Dado um mapa dividido em regiões, é possível colorir as regiões do mapa de forma que regiões vizinhas sempre tenham cores diferentes usando no máximo 4 cores.

Aula 3 - N2

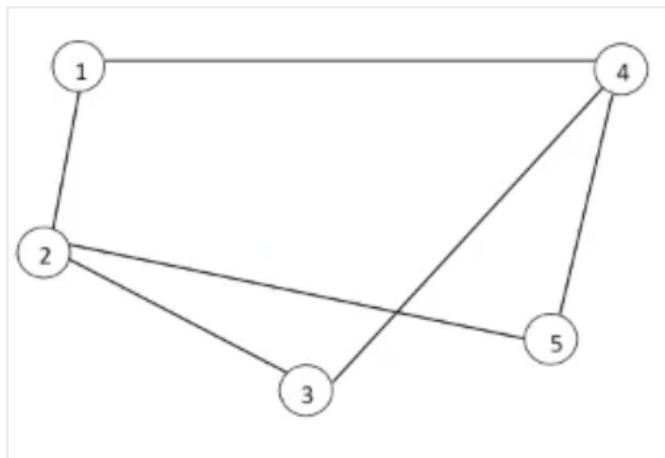
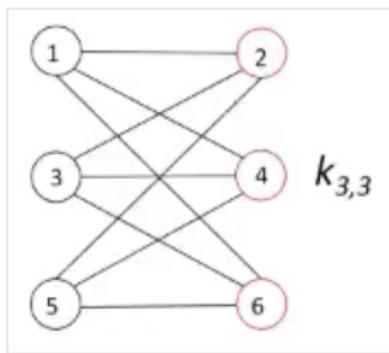
Emparelhamento em grafos bipartidos

Grafos bipartidos

São grafos que podem ser dividido em dois grupos ditos X e Y e as arestas sempre vão ligar vértices do tipo X a vértices do tipo Y.

Um grafo é dito bipartido (ou biparticionável) se é possível partitionar seu conjunto de vértices em dois subconjuntos, digamos X e Y, de tal maneira que toda aresta conecte um vértice de X a um vértice de Y.

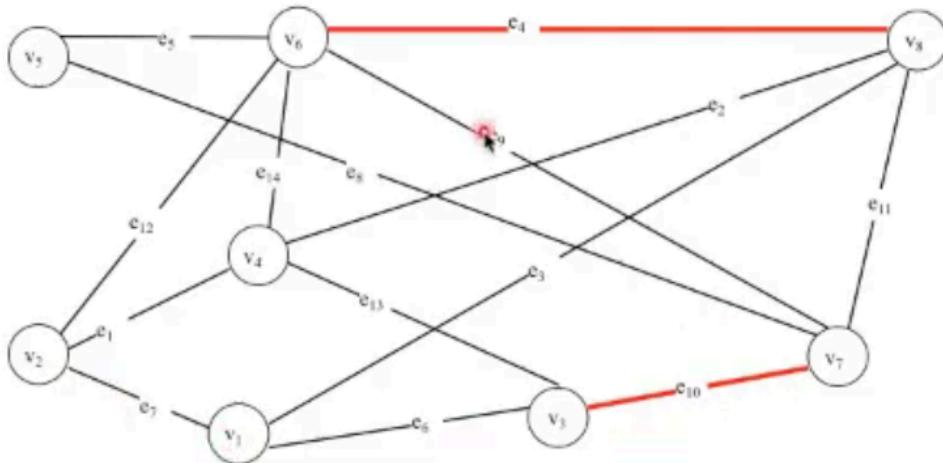
Exemplos de grafos bipartidos:



Teorema: Um grafo é bipartido se e somente se ele não possui circuito de comprimento ímpar.

Emparelhamento

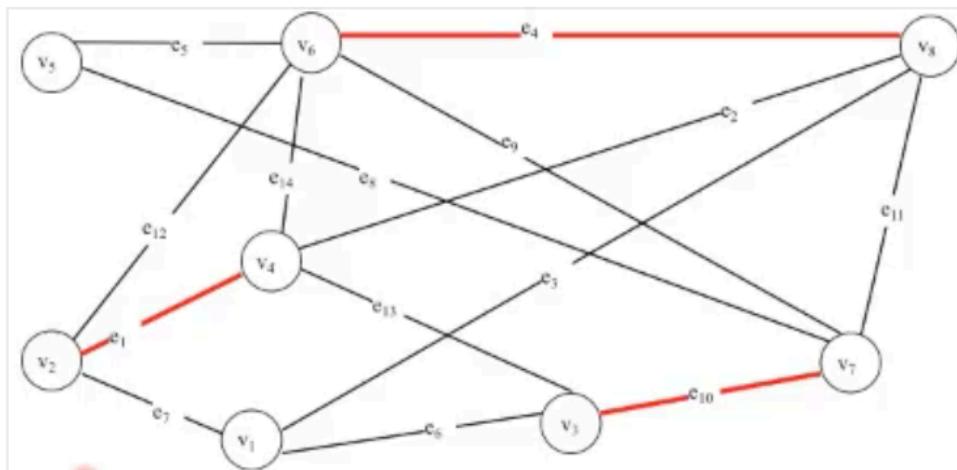
Emparelhamento é um conjunto de arestas não adjacentes entre si.



Emparelhamento

Dizemos que um emparelhamento é maximal se ele não está propriamente contido em nenhum outro emparelhamento. Ou seja, não existe emparelhamento maior que ele contendo ele.

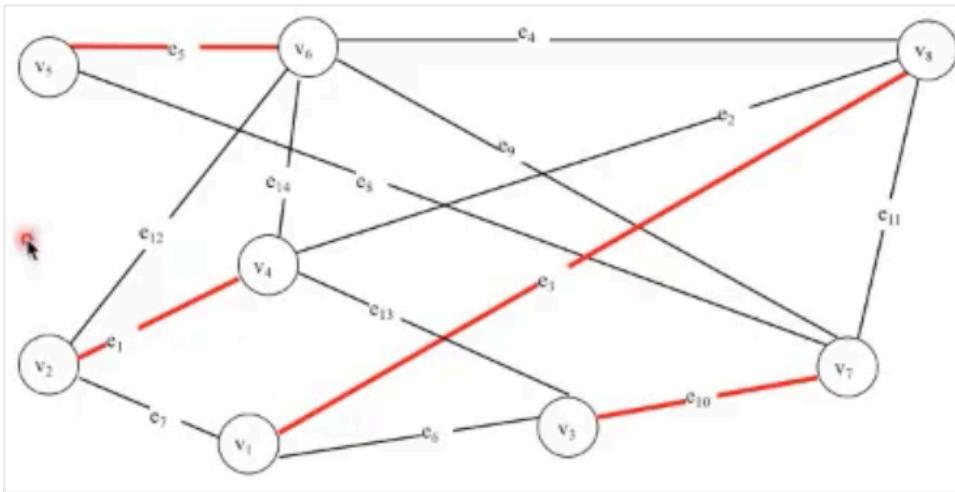
Exemplo de emparelhamento maximal:



Seja M um emparelhamento. Se uma aresta uv pertence a M , dizemos que M cobre u e v .

Um emparelhamento é perfeito se ele cobre todos os vértices de um grafo.

Exemplo de emparelhamento perfeito:



Todo emparelhamento perfeito é máximo.

Entretanto, nem todo emparelhamento máximo é perfeito.

Nem todo emparelhamento maximal é perfeito, mas todo emparelhamento perfeito é maximal.

Seja G um grafo bipartido com bipartição (X, Y) . Quais as condições para que exista um emparelhamento que cubra todos os vértices de X ?

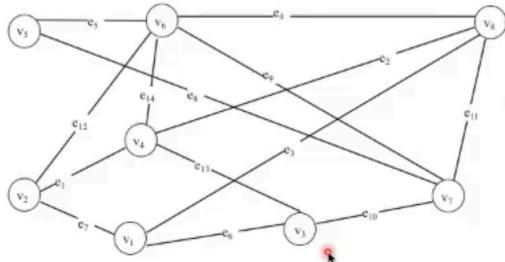
Aula 4 - N2

Teorema de Hall

Seja G um grafo bipartido com bipartição (X, Y) . O teorema de Hall estabelece uma condição que é necessária e suficiente para que exista um emparelhamento que cubra todos os vértices de X . Antes de enunciá-lo e provar-lo, precisamos de algumas definições.

Seja S um conjunto de vértices. Chamamos de vizinhança de S , denotada por $N(S)$, o conjunto de todos os vértices adjacentes a algum vértice de S .

Ex:



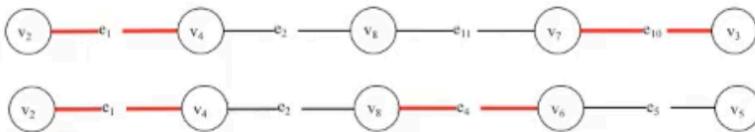
$$S = \{v_1, v_2, v_3\}$$

$$N(S) = \{v_2, v_3, v_8, v_1, v_4, v_6, v_7\}$$

Fato: Seja G um grafo bipartido com bipartição (X, Y) . Se $S \subseteq X$, então $S \cap N(s) = \emptyset$

Sejam M um emparelhamento e p um caminho num grafo G . Dizemos que p é M -alternante se as arestas de p se alternam na propriedade de pertencer a M , ou seja, se uma aresta de p pertence a M a aresta consecutiva não pertence a M , e se uma aresta de p não pertence a M então a aresta consecutiva pertence a M .

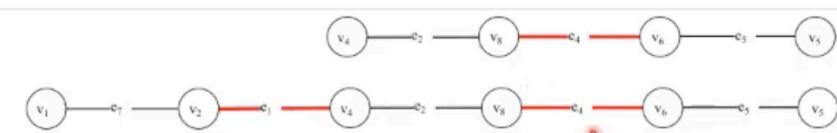
Exemplo:



Não é M -alternante

É M -alternante

Dizemos que um caminho M -alternante é M -aumentador se os vértices nas extremidades do caminho não estão cobertos por M .



Não é M -aumentador

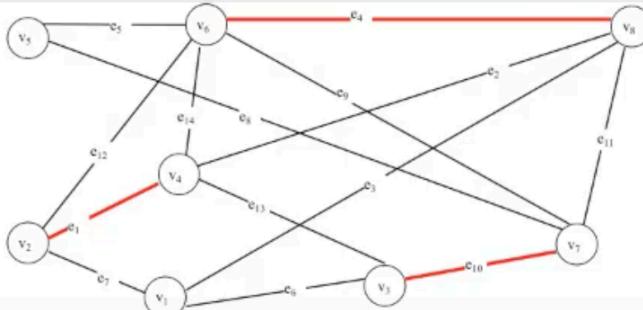
É M -aumentador

O caminho é dito aumentador porque a gente pode ter um emparelhamento maior.

Podemos usar um caminho M -aumentador para aumentar o emparelhamento M . Para isso, devemos remover de M as arestas do caminho que pertencem a M e inserir em M as arestas do caminho que não pertencem a M .

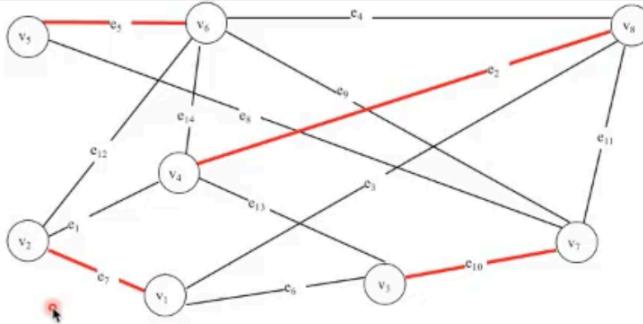
Antes:

Ex: M é o conjunto das arestas vermelhas



Depois:

Ex: M é o conjunto das arestas vermelhas



Teorema de Hall

Seja G um grafo bipartido com bipartição (X, Y) . Existe um emparelhamento que cobre todos os vértices de X se e somente se para todo $S \subseteq X$ temos que $|S| \leq |N(S)|$.



Prova: (\Rightarrow) Suponha que existe um emparelhamento que cobre todos os vértices de X . Note que para todo $S \subseteq X$ temos que ter $|S| \leq |N(S)|$, caso contrário não seria possível cobrir todos os vértices de S .

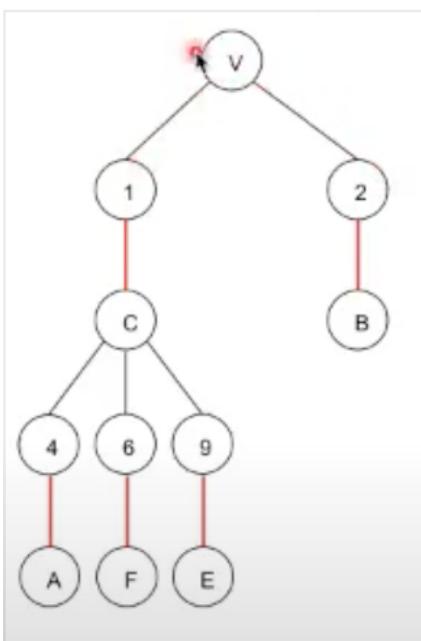
(\Leftarrow) Suponha agora que para todo $S \subseteq X$ temos que $|S| \leq |N(S)|$.

Queremos provar que existe um emparelhamento que cobre todos os vértices de X .

Suponha, por absurdo, que não exista um emparelhamento que cobre

todos os vértices de X . Seja M um emparelhamento máximo em X e $v \in X$ um vértice não coberto por M .

Construa uma árvore maximal enraizada em v na qual todos os caminhos iniciados na raiz sejam M -alternantes. Claramente, tal árvore não pode conter nenhum caminho M -aumentador, caso contrário M não seria máximo. Assim, nessa árvore, todas as folhas são vértices cobertos por M .



Considerando que v está no nível 1 da árvore, não há folhas nos níveis pares da árvore. Seja S o conjunto dos vértices nos níveis ímpares. Observe que $N(S)$ é exatamente o conjunto de vértices dos níveis pares da árvore. Note ainda que $|S| = |N(S)| + 1 > |N(S)|$, o que é uma contradição $\Rightarrow \Leftarrow$.

Aula 5 - N2

Método Húngaro

Pseudo Código:

Método Húngaro

Entrada: Um grafo bipartido G com bipartição (X, Y)

Saída: um emparelhamento que cobre todos os vértices de X ou um conjunto S tal que $|S| > |N(S)|$

$M = \{ \}$

enquanto existir vértice $v \in X$ não coberto por M

construa uma árvore maximal enraizada em v na qual todos os caminhos iniciados em v sejam M -alternantes

se existir na árvore um caminho p que seja M -aumentador

$A = \{\text{arestas de } p \text{ que pertencem a } M\}$

$B = \{\text{arestas de } p \text{ que não pertencem a } M\}$

$M = \{M \cup B\} - A$

se não

$S = \{\text{vértices da árvore que pertencem a } X\}$

devolva S e pare

devolva M

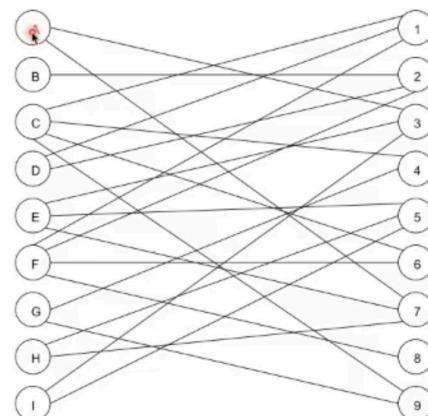
Problema da designação de tarefas

No problema da designação de tarefas temos n funcionários e n tarefas. Sabemos quais tarefas cada funcionário está apto a executar. Desejamos designar exatamente 1 tarefa para cada funcionário de modo que cada funcionário tenha que executar uma tarefa para a qual está habilitado.

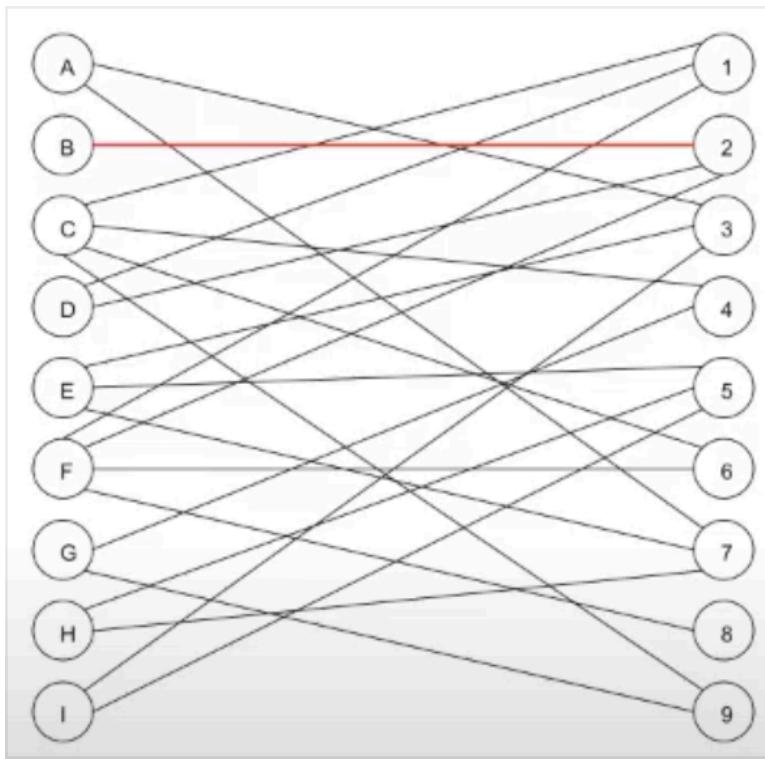
Como resolver:

Ex:

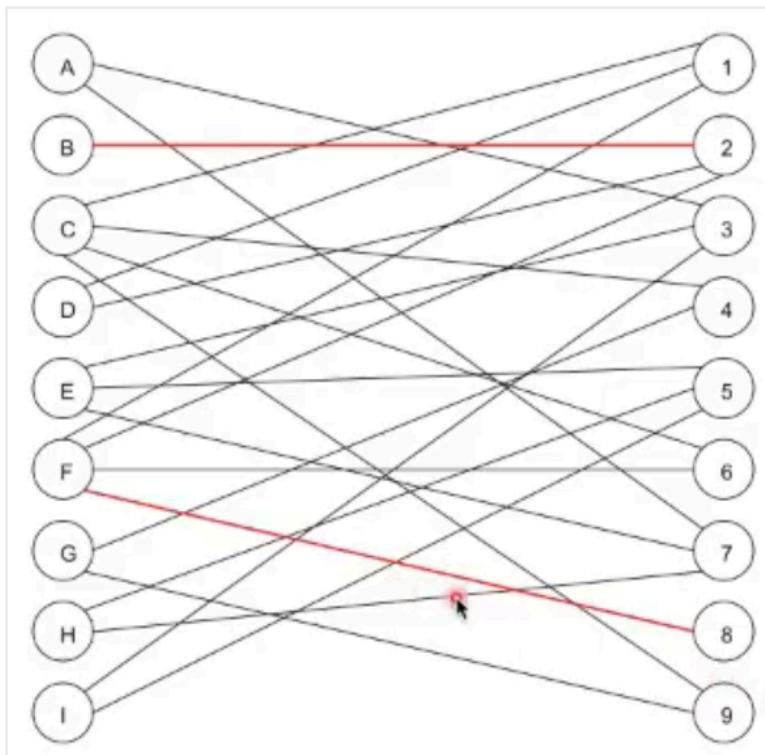
	1	2	3	4	5	6	7	8	9
Ana			✓				✓		
Bel		✓							
Cid	✓			✓		✓			✓
Davi	✓	✓							
Eva			✓	✓		✓			
Fred	✓	✓				✓		✓	
Gil				✓					✓
Hugo					✓		✓		
Iara			✓	✓					



Primeiro verificamos se tem alguém que só pode executar uma tarefa, se sim, designamos essa tarefa a essa pessoa.

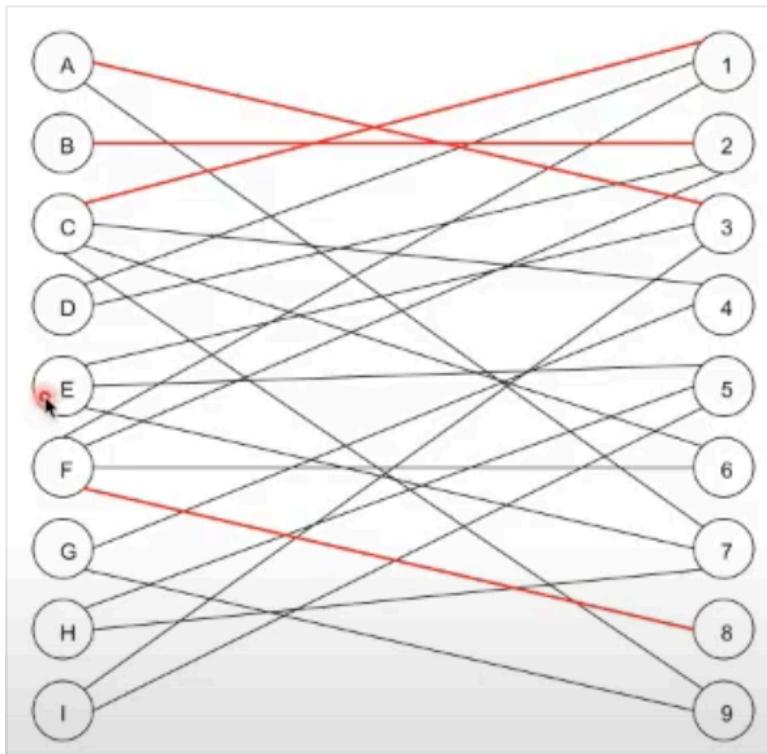


Depois fazemos o mesmo para as tarefas. Se existir uma tarefa que só pode ser executada por um funcionário, entregamos essa tarefa a esse funcionário.



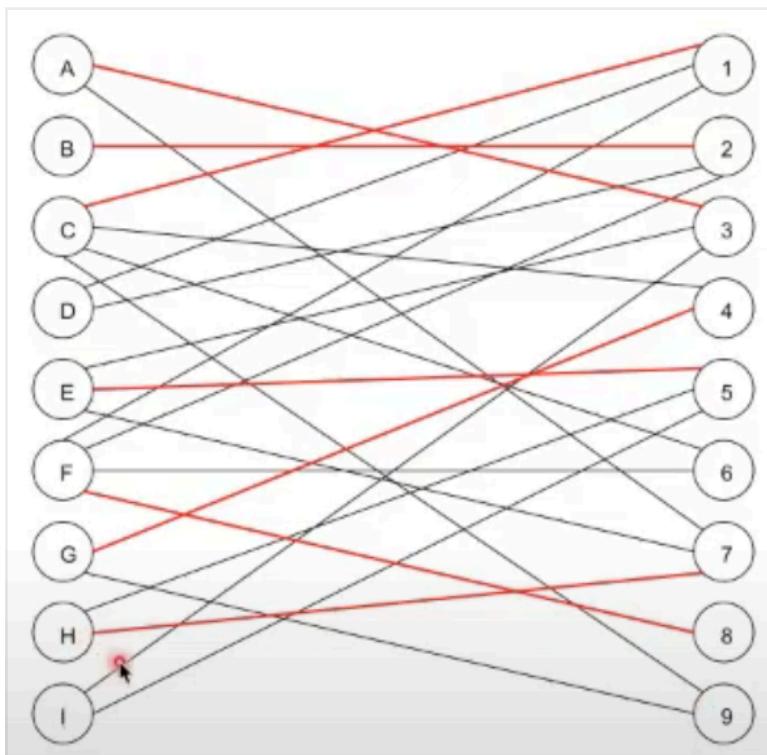
Agora podemos fazer uma decisão que pode ser ruim mas vamos atribuir a cada um dos restantes dos trabalhadores a primeira tarefa que ele pode executar. Se for ruim voltamos atrás.

Entregando para A e C uma tarefa:



Não foi possível designar uma tarefa para D, então pulamos e vamos para E.

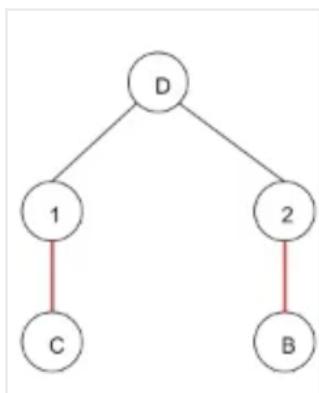
Atribuindo tarefas para E, G e H:



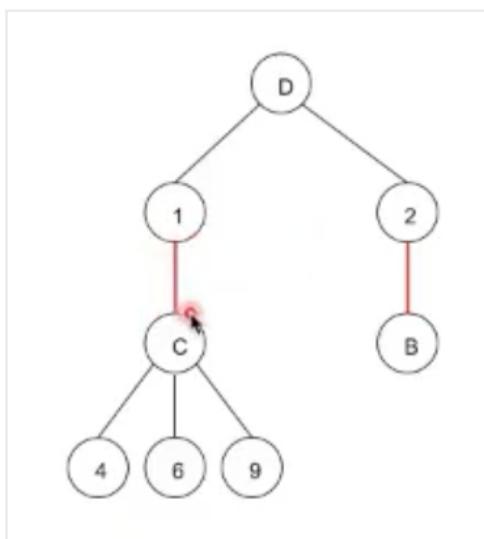
Não foi possível atribuir tarefas para I.

Agora sim vamos utilizar o método Hungaro.

Construiremos a arvore do método a partir de um vértice que não está sendo contemplado, no caso temos D e I. Vamos colocar D como raiz.

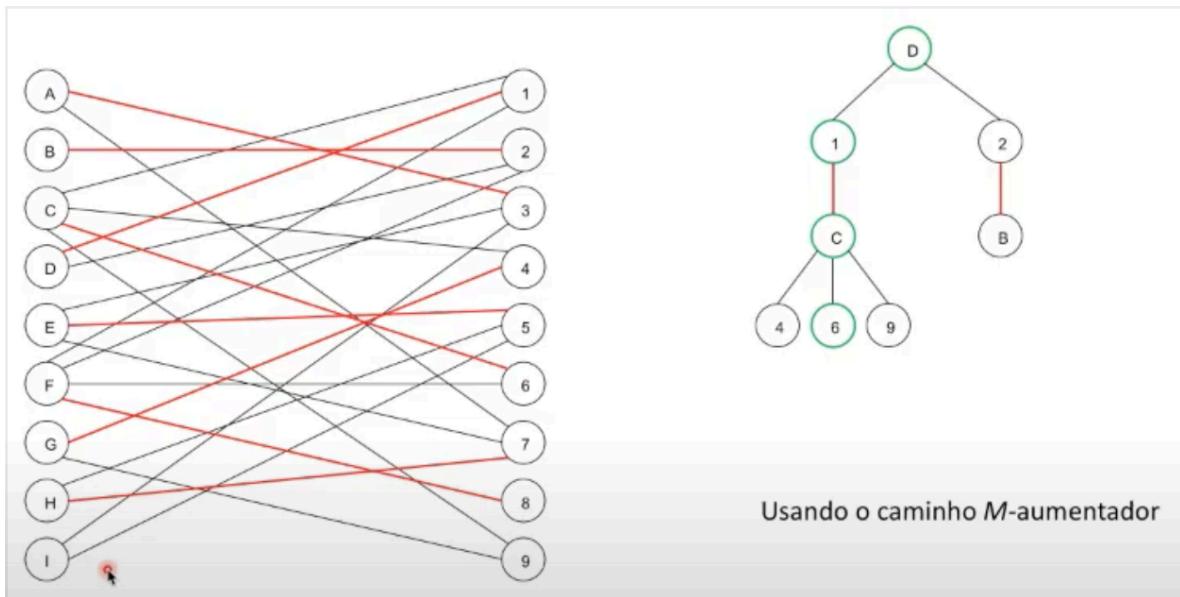


Seguindo com as adições em busca de um caminho aumentador:

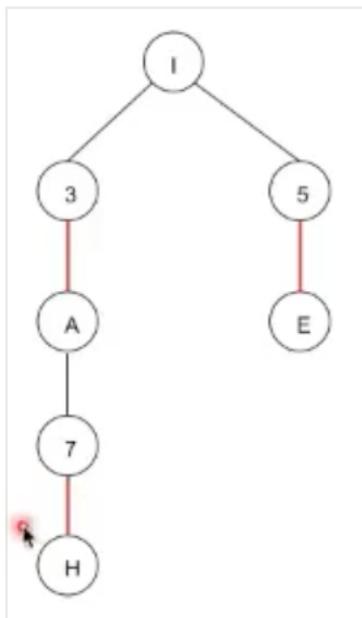


Temos na foto em cima então que o caminho que termina em 6 é um caminho aumentador, pois 6 não foi designado a ninguém e para corrigirmos isso fazemos a alternância de arestas atribuídas para aumentar nosso emparelhamento.

Fazendo as alterações:



Agora construimos uma outra outra árvore maximal para I :



Percebemos assim que não conseguimos formar um caminho aumentador e segundo o algoritmo devemos chamar os níveis ímpares da árvore de S :

$$S = \{I, A, E, H\}$$

$$N(S) = \{3, 5, 7\}$$

Isso significa que encontramos 4 funcionários que só conseguem executar 3 tarefas.

Conclusão: Não tem como designar tarefas para cada um deles, um sempre ficará de fora.

Aula 6 - N2

Fluxos em Redes

Uma rede é um grafo dirigido com dois vértices especiais s (origem) e t (destino), no qual cada arco i possui uma capacidade de vazão c_i .

Denotamos por

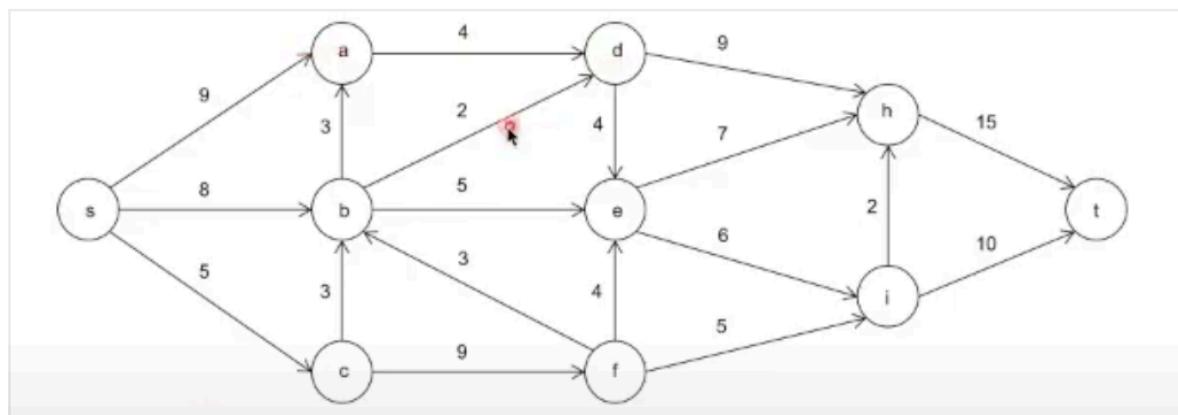
$$\delta^+(v)$$

o conjunto de todos os arcos que chegam em v e por

$$\delta^-(v)$$

o conjunto de todos os arcos que saem de v .

Exemplo:



$$\text{Ex: } \delta^+(d) = \{\vec{ad}, \vec{bd}\}, \delta^-(d) = \{\vec{dh}, \vec{de}\}$$

Um fluxo numa rede é uma atribuição de fluxo a cada um dos arcos da rede. Vamos denotar por f_i o fluxo no arco i . Um fluxo é viável se:

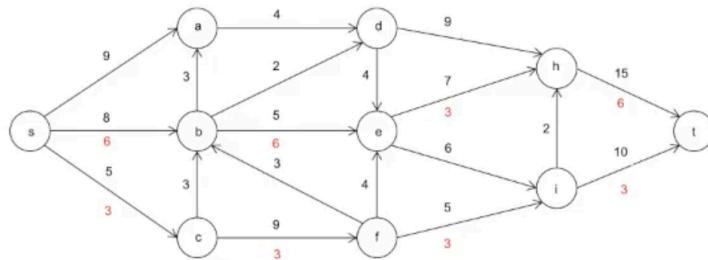
Para todo arco i , $0 \leq f_i \leq c_i$

Para todo vértice v (diferente de s e de t),

$$\sum_{i \in \delta^+(v)} f_i = \sum_{i \in \delta^-(v)} f_i \text{ (Lei de Kirchoff)}$$

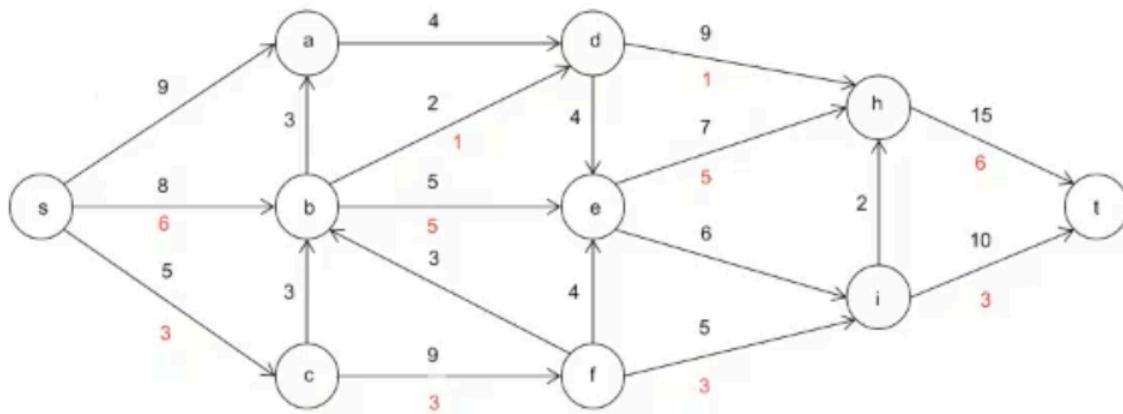
Exemplo de fluxo não viável:

No exemplo a seguir os valores em vermelho representam os fluxos nos arcos, sendo que os arcos que não têm o fluxo indicado na figura têm fluxo 0.



O fluxo é não viável pq o fluxo no arco BE é maior que sua capacidade e segundo a lei de kirchoff, no vertice E e H a lei é violada.

Exemplo de fluxo viável:



Fluxo viável

O volume de um fluxo F , denotado por $\text{vol}(F)$, é definido como sendo:

$$\text{vol}(F) = \sum_{i \in \delta^-(s)} f_i - \sum_{i \in \delta^+(s)} f_i$$

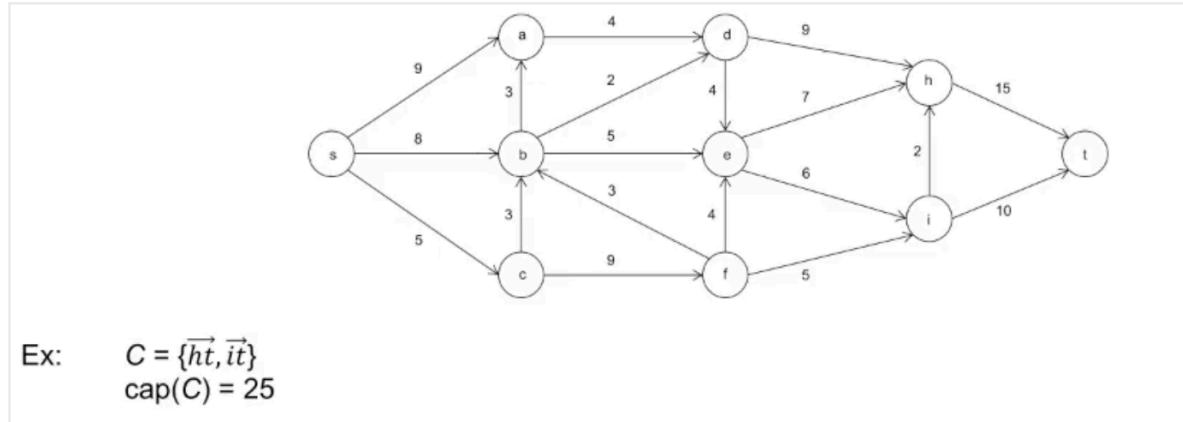
Ou seja, é a quantidade de fluxo que sai de S - fluxo que chega em S.

$$\text{vol}(F) = \sum_{i \in \delta^-(s)} f_i - \sum_{i \in \delta^+(s)} f_i = \sum_{i \in \delta^+(t)} f_i - \sum_{i \in \delta^-(t)} f_i$$

Ex: O volume do fluxo do exemplo acima é 9.

Um (s,t) -corte numa rede é um conjunto de arcos cuja remoção desconecta s de t . A capacidade de um (s,t) -corte, denotada por $\text{cap}(C)$, é a soma das capacidades dos arcos que compõem C .

Ex:



Teorema: Se F é um fluxo viável e C é um (s,t) -corte, então $\text{vol}(F) \leq \text{cap}(C)$.

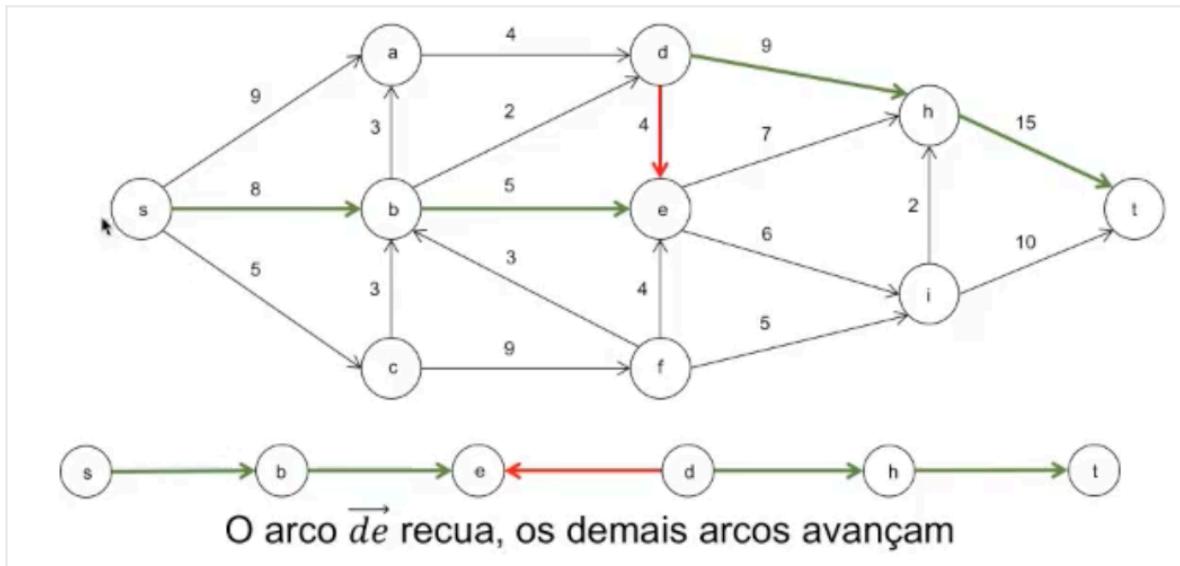
Aula 7 - N2

Teorema de Ford-Fulkerson

O teorema de Ford-Fulkerson estabelece uma condição que é necessária e suficiente para que um fluxo possua volume máximo. Antes de enunciá-lo e prova-lo, precisamos de algumas definições.

Sejam F um fluxo viável e p um caminho(não necessariamente dirigido) de s a t . Dizemos que um arco de p avança se ele está na direção de s a t e dizemos que ele recusa se está na direção de t a s .

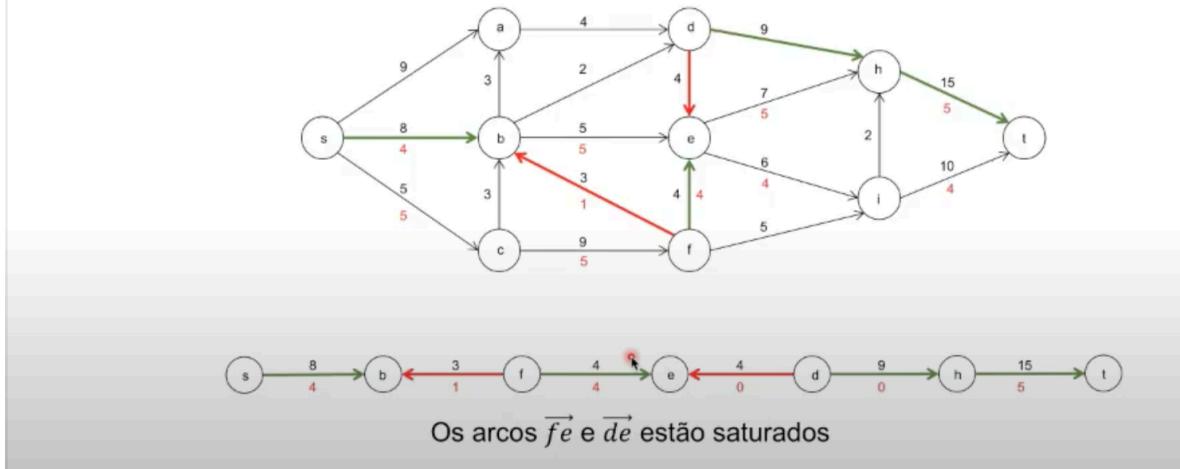
Exemplo:



Um arco i que avança está saturado se $f_i = c_i$. Um arco i que recua está saturado se $f_i = 0$.

Exemplo:

Ex: os valores em vermelho representam os fluxos nos arcos, sendo que os arcos que não têm o f_i indicado na figura têm fluxo 0.



...