ALGORITMOS

Disciplina: Estruturas de Dados e Algoritmos

Professor: Rafael Marinho e Silva

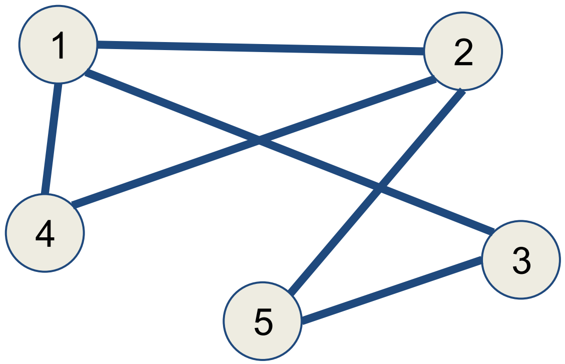
CRÉDITOS

O material a seguir contém trechos e adaptações dos originais criados pelos Professores:

- Professor Marcelo Zorzan
- Professora Rachel Reis
- Professor Marcelo Keese Albertini
- Professor André Back

Como representar **um conjunto de objetos** e as **suas relações**?

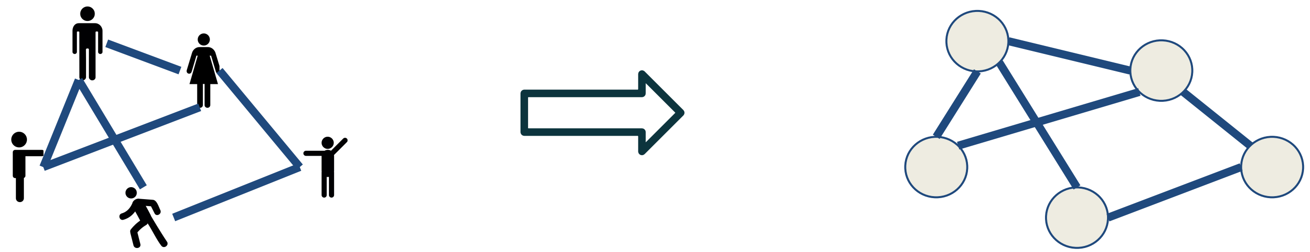
- **Um grafo é um modelo matemático que representa as relações entre objetos de um determinado conjunto.**

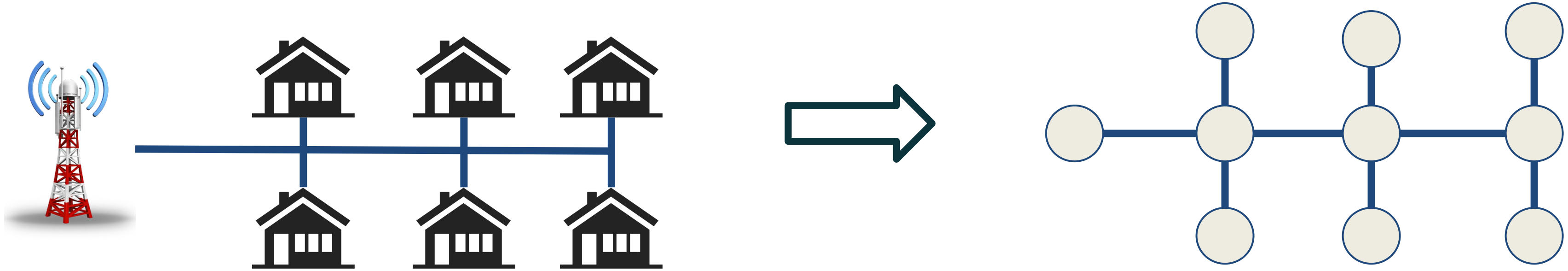


REVISÃO - GRAFOS

Grafos em computação

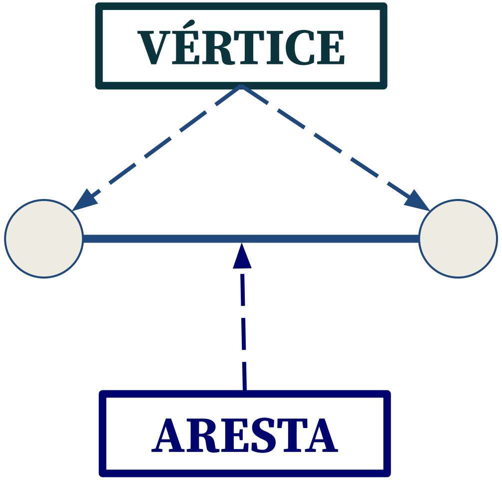
- Forma de **solucionar** problemas **computáveis**





Vértice

- É cada uma das entidades representadas em um grafo.
- Dependente da natureza do problema, podem ser pessoas, casas, etc.
- Dois vértices são adjacentes se existir uma aresta ligando eles.



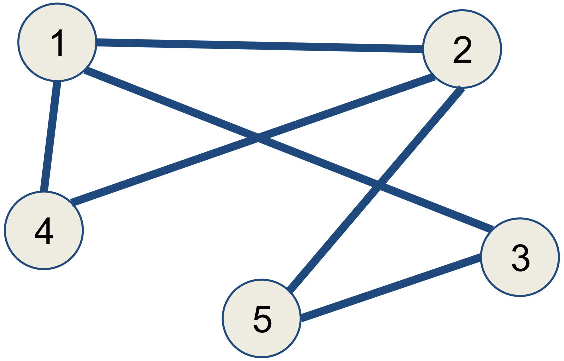
Aresta

- Está associada a dois vértices (v1, v2)
- Faz a ligação entre eles (tipo da relação entre eles)

REVISÃO

Grafos

- É definido como um conjunto de vértices e um conjunto de arestas que conectam qualquer par de vértices.
  - G = (V , A)
    - V é o conjunto de vértices (não vazio)
    - A é o conjunto de arestas

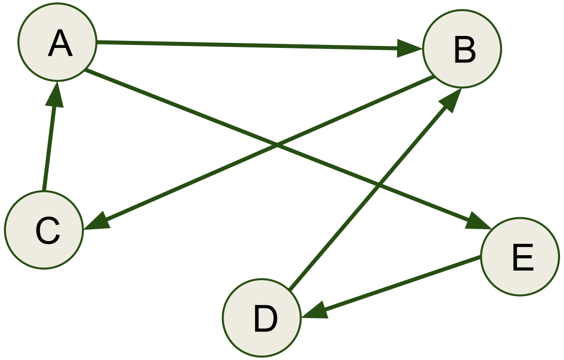


G(V,A)
V = {1, 2, 3, 4, 5}
A{{1, 2}, {1, 3}, {1, 4}, {2, 4}, {2, 5}, {3, 5}}

REVISÃO - GRAFOS

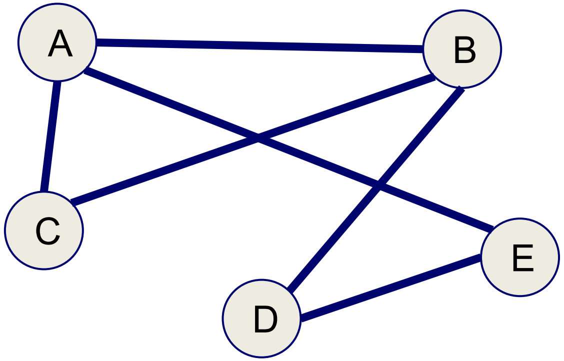
Grafo direcionado

- Existe uma orientação quanto ao sentido da aresta
- Se uma aresta liga “A” e “B”, podemos ir de “A” para “B”, mas não o contrário



Grafo não direcionado

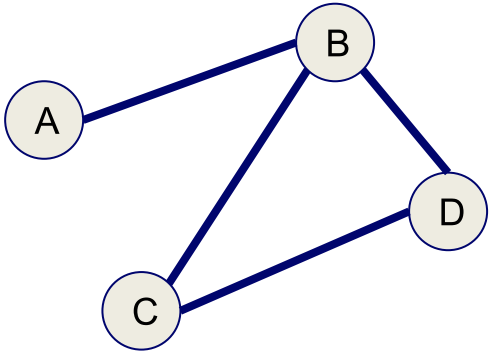
- Não existe nenhuma orientação quanto ao sentido da aresta. Podemos ir de “A” para “B” ou de “B” para “A”



Grau de um vértice

- É o número de arestas que chegam ou partem dele
- No caso dos **digrafos**, temos:
  - **“grau de entrada”**: arestas que chegam ao vértice
  - **“grau de saída”**: arestas que partem do vértice

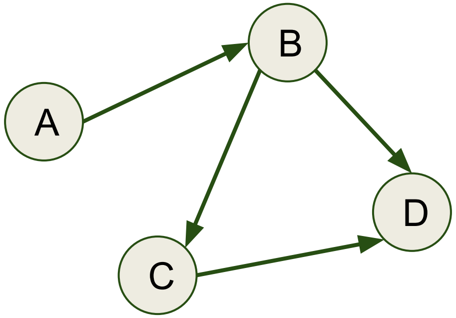
Grau
G(A) = 1
G(B) = 3
G(C) = 2
G(D) = 2



GRAFO

Grau Entrada
G(A) = 0
G(B) = 1
G(C) = 1
G(D) = 2

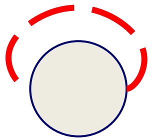
Grau Saída
G(A) = 1
G(B) = 2
G(C) = 1
G(D) = 0

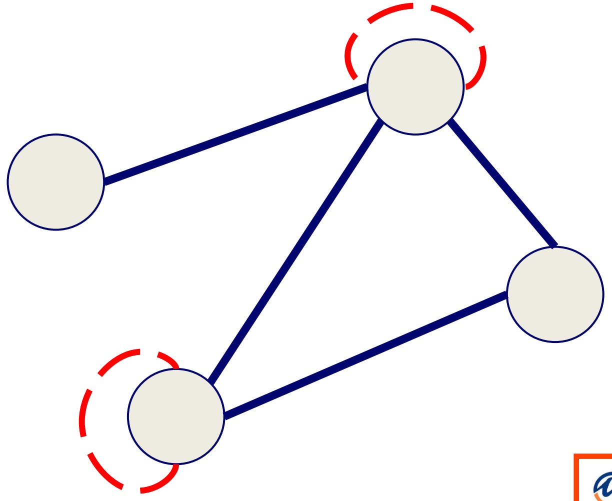


DIGRAFO

Laço

- Uma aresta é chamada de “**laço**” se seu vértice de partida é o mesmo que o de chegada, ou seja, a aresta conecta o vértice com ele mesmo (v, v)

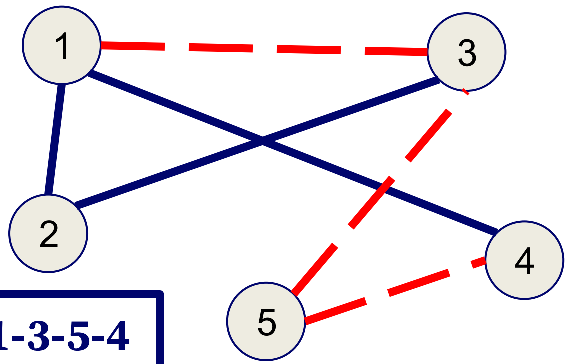




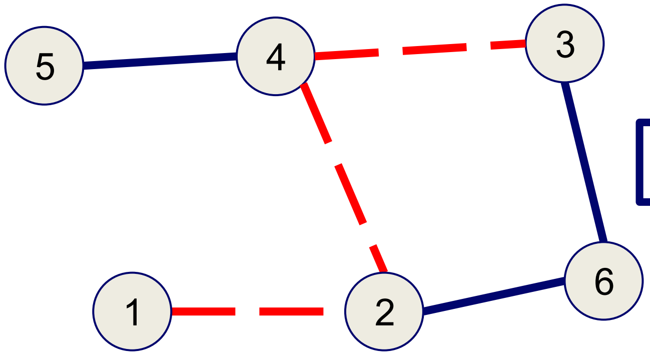
REVISÃO - GRAFOS

Caminho

- É uma sequência de vértices de modo que existe sempre uma aresta ligando o vértice anterior com o seguinte
  - **“Caminho simples”**
    - Nenhum dos vértices no caminho se repete
  - **“Comprimento do caminho”**
    - É o número de arestas que o caminho usa



Caminho: 1-3-5-4

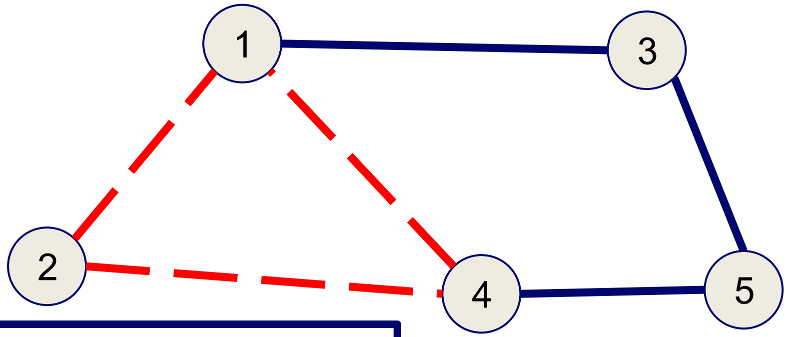


Caminho: 1-2-4-3

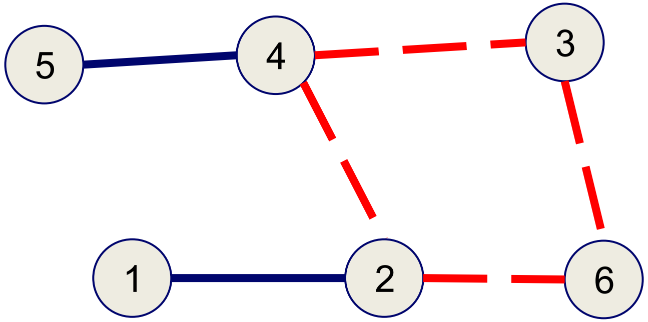
REVISÃO - GRAFOS

Caminho

- É uma sequência de vértices de modo que existe sempre uma aresta ligando o vértice anterior com o seguinte
  - **"Grafo ciclo"**
    - É um caminho que começa e termina no mesmo vértice
  - **"Grafo acíclico"**
    - Não contém "ciclo simples" (onde cada vértice aparece apenas uma vez)



Caminho: 1-2-4-1

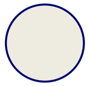


Caminho: 4-2-6-3-4

REVISÃO - GRAFOS

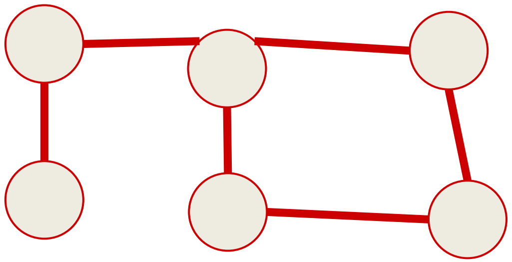
Grafo Trivial

- É um grafo com um único vértice e sem arestas



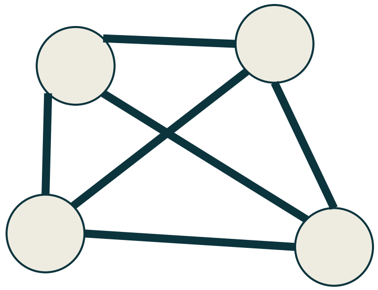
Grafo simples

- É um grafo não direcionado, sem laços e sem arestas paralelas



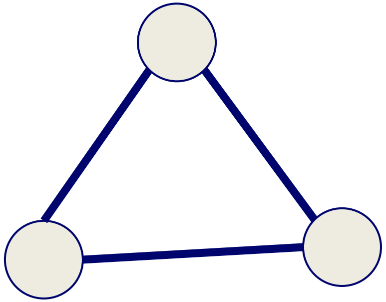
Grafo completo

- É um grafo simples onde cada vértice se conecta a todos os outros vértices

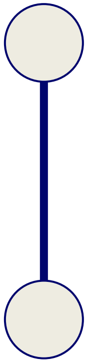


Grafo regular

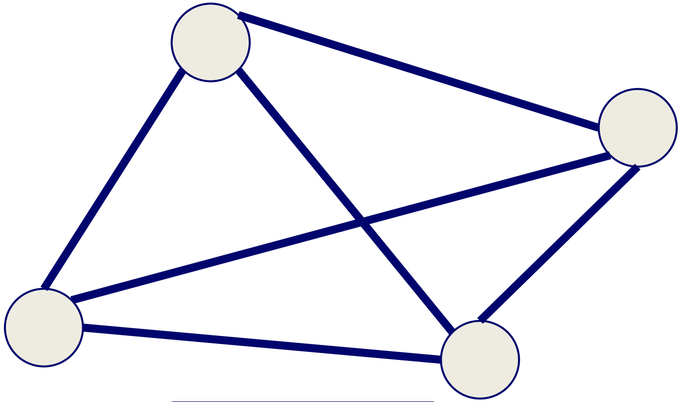
- É um grafo onde todos os vértices possuem o mesmo grau



Grau = 2



Grau = 1

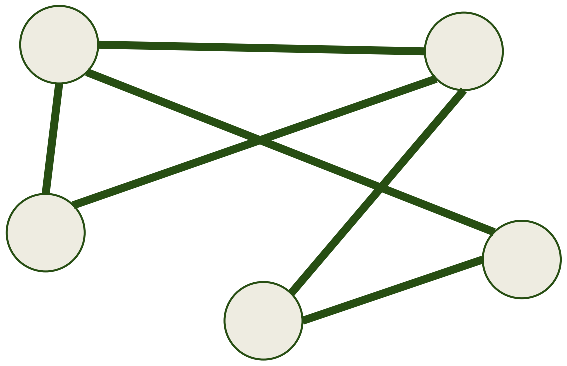


Grau = 3

REVISÃO - GRAFOS

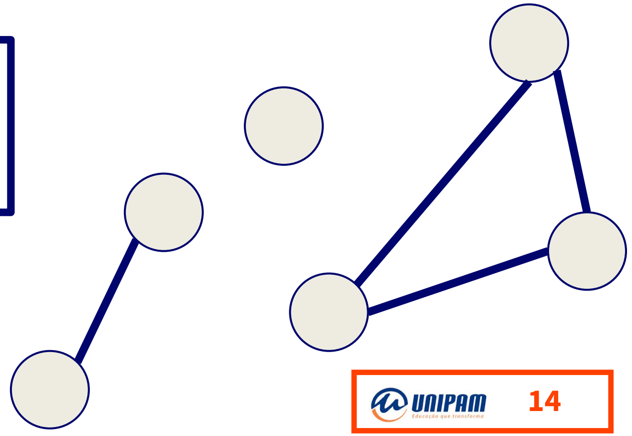
Grafo conexo

- Existe um caminho partindo de qualquer vértice até qualquer outro vértice do grafo



Grafo desconexo

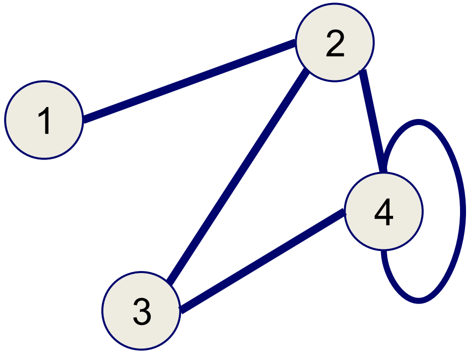
- Não existe um caminho ligando dois vértices



Matriz de adjacência

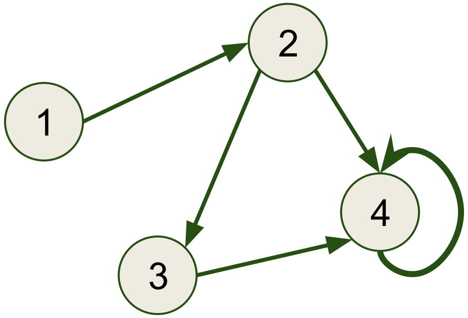
- Uma matriz N x N é utilizada para armazenar o grafo, onde N é o número de vértices
  - Alto custo computacional, **O(N^2)**
- Uma aresta é representada por uma “marca” na posição (i, j) da matriz
  - Aresta liga o vértice i ao j

	1	2	3	4
1	-	-	-	-
2	-	-	-	-
3	-	-	-	-
4	-	-	-	-



GRAFO

	1	2	3	4
1	-	-	-	-
2	-	-	-	-
3	-	-	-	-
4	-	-	-	-



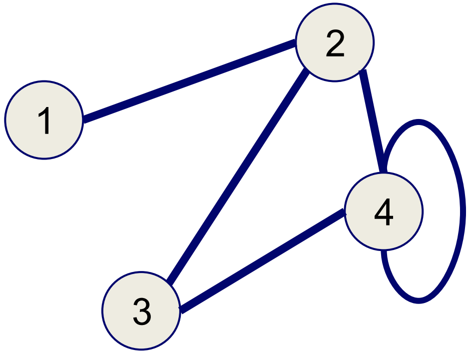
DIGRAFO

REVISÃO - REPRESENTAÇÃO DE GRAFOS

Matriz de adjacência

- Uma matriz N x N é utilizada para armazenar o grafo, onde N é o número de vértices
  - Alto custo computacional, **O(N²)**
- Uma aresta é representada por uma "marca" na posição (i, j) da matriz
  - Aresta liga o vértice i ao j

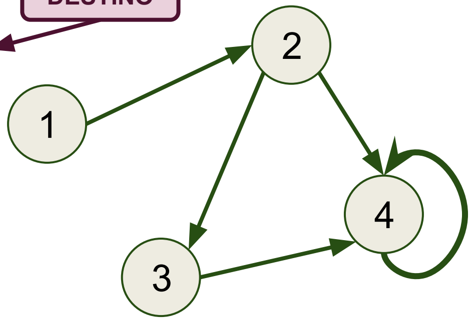
	1	2	3	4
1	0	1	0	0
2	1	0	1	1
3	0	1	0	1
4	0	1	1	1



GRAFO

DESTINO

ORIGEM	1	2	3	4
1	-	-	-	-
2	-	-	-	-
3	-	-	-	-
4	-	-	-	-



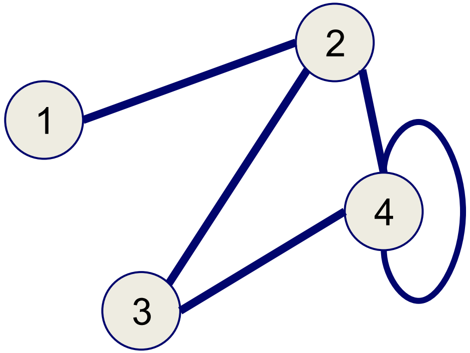
DIGRAFO

REVISÃO - REPRESENTAÇÃO DE GRAFOS

Matriz de adjacência

- Uma matriz N x N é utilizada para armazenar o grafo, onde N é o número de vértices
  - Alto custo computacional, **$O(N^2)$**
- Uma aresta é representada por uma “marca” na posição (i, j) da matriz
  - Aresta liga o vértice i ao j

	1	2	3	4
1	0	1	0	0
2	1	0	1	1
3	0	1	0	1
4	0	1	1	1

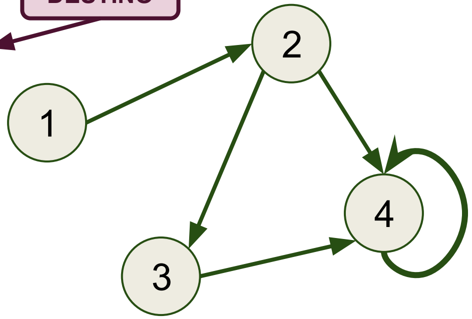


GRAFO

DESTINO

	1	2	3	4
1	0	1	0	0
2	0	0	1	1
3	0	0	0	1
4	0	0	0	1

ORIGEM

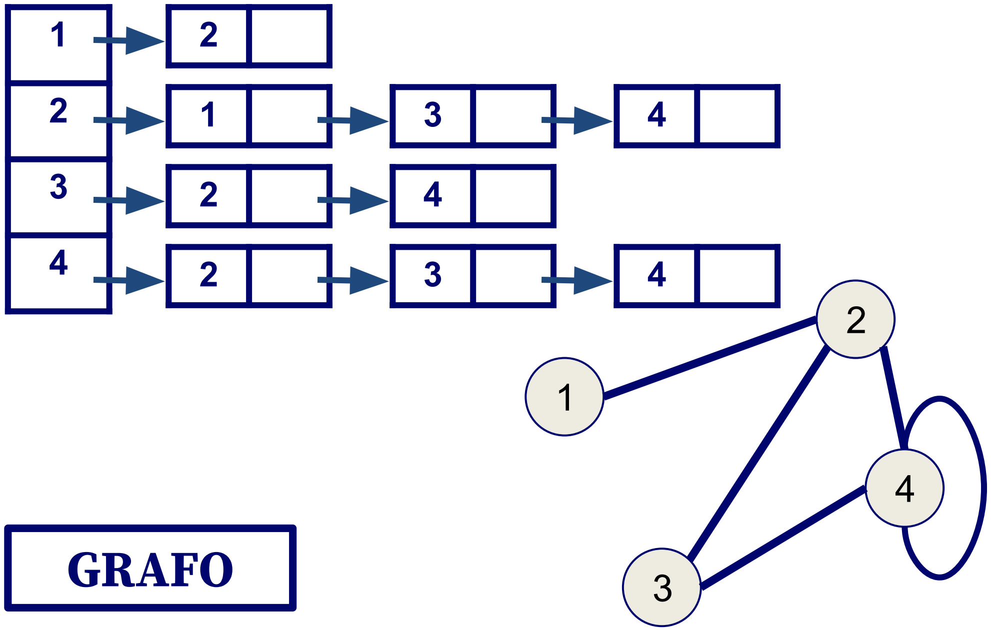


DIGRAFO

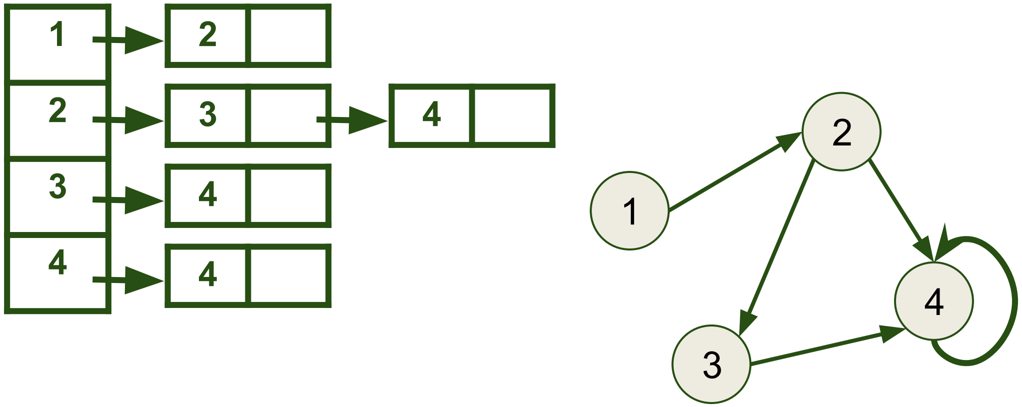
REVISÃO - REPRESENTAÇÃO DE GRAFOS

Lista de Adjacência

- Pode ser estática ou dinâmica
  - Cada vértice aponta para todos os vértices que ele tem conexão



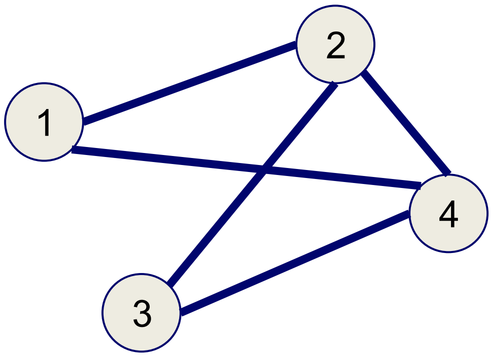
GRAFO



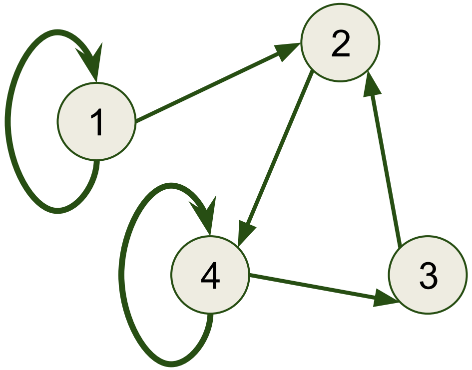
DIGRAFO

Exercícios

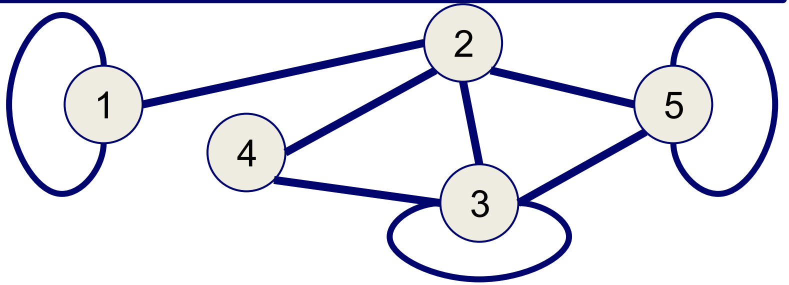
- Faça uma matriz de adjacência para os seguintes grafos



	1	2	3	4
1	-	-	-	-
2	-	-	-	-
3	-	-	-	-
4	-	-	-	-



	1	2	3	4
1	-	-	-	-
2	-	-	-	-
3	-	-	-	-
4	-	-	-	-

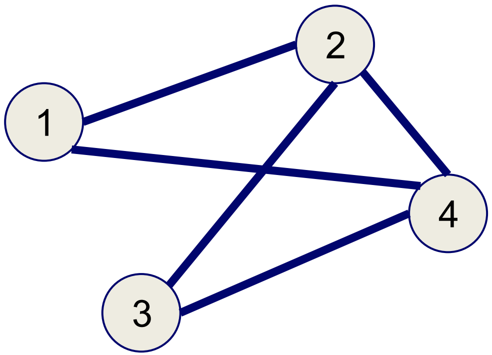


	1	2	3	4	5
1	-	-	-	-	-
2	-	-	-	-	-
3	-	-	-	-	-
4	-	-	-	-	-
5	-	-	-	-	-

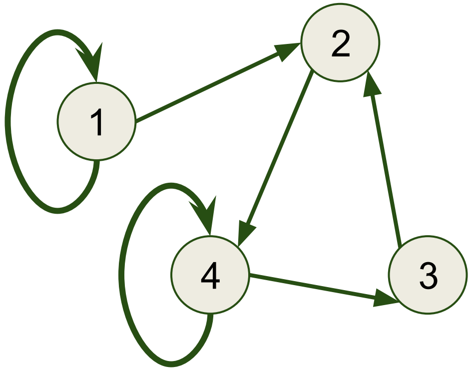
REVISÃO - REPRESENTAÇÃO DE GRAFOS

Exercícios

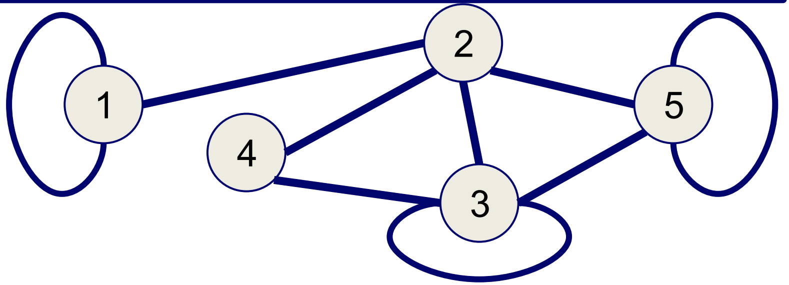
- Faça uma matriz de adjacência para os seguintes grafos



	1	2	3	4
1	0	1	0	1
2	1	0	1	1
3	0	1	0	1
4	1	1	1	0



	1	2	3	4
1	-	-	-	-
2	-	-	-	-
3	-	-	-	-
4	-	-	-	-

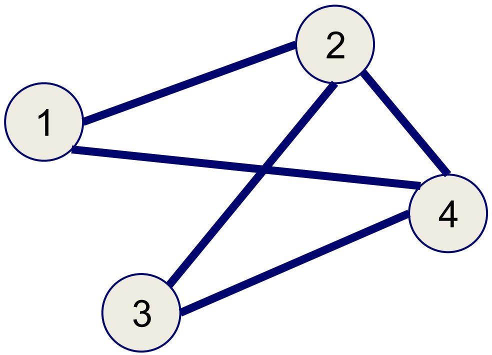


	1	2	3	4	5
1	-	-	-	-	-
2	-	-	-	-	-
3	-	-	-	-	-
4	-	-	-	-	-
5	-	-	-	-	-

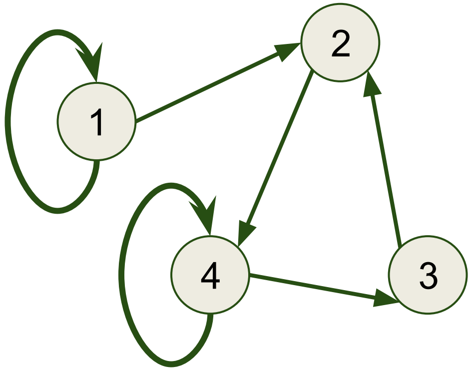
REVISÃO - REPRESENTAÇÃO DE GRAFOS

Exercícios

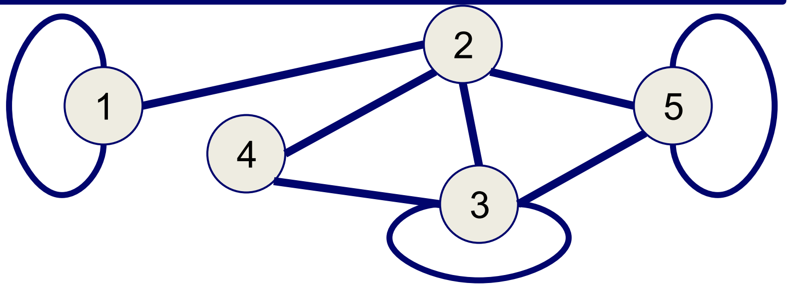
- Faça uma matriz de adjacência para os seguintes grafos



	1	2	3	4
1	0	1	0	1
2	1	0	1	1
3	0	1	0	1
4	1	1	1	0



	1	2	3	4
1	1	1	0	0
2	0	0	0	1
3	0	1	0	0
4	0	0	1	1

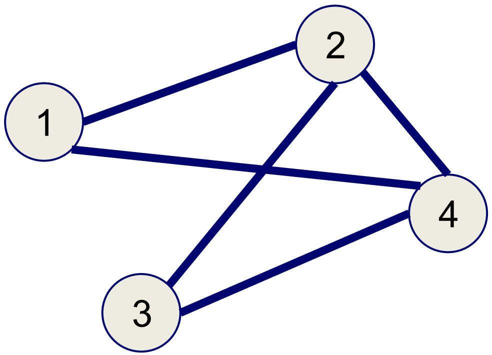


	1	2	3	4	5
1	-	-	-	-	-
2	-	-	-	-	-
3	-	-	-	-	-
4	-	-	-	-	-
5	-	-	-	-	-

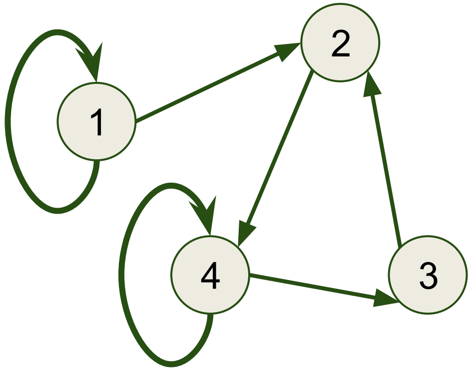
REVISÃO - REPRESENTAÇÃO DE GRAFOS

Exercícios

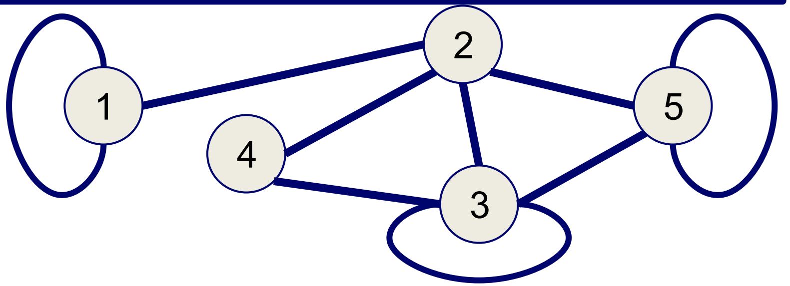
- Faça uma matriz de adjacência para os seguintes grafos



	1	2	3	4
1	0	1	0	1
2	1	0	1	1
3	0	1	0	1
4	1	1	1	0



	1	2	3	4
1	1	1	0	0
2	0	0	0	1
3	0	1	0	0
4	0	0	1	1



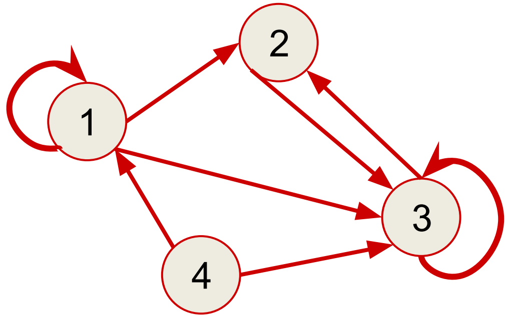
	1	2	3	4	5
1	1	1	0	0	0
2	1	0	1	1	1
3	0	1	1	1	1
4	0	1	1	0	0
5	0	1	1	0	1

ALGORITMOS

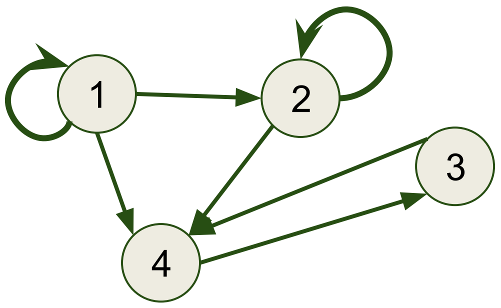
REVISÃO - REPRESENTAÇÃO DE GRAFOS

Exercícios

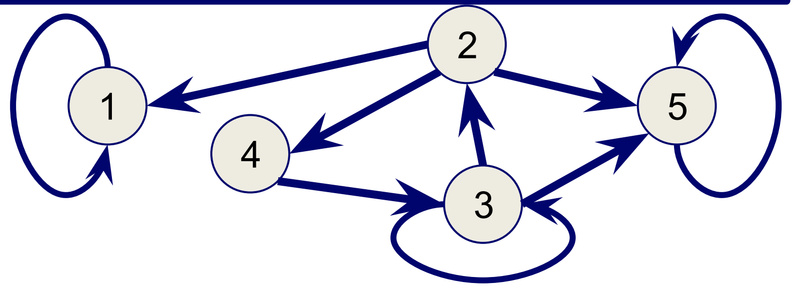
- Faça uma matriz de adjacência para os seguintes grafos



	1	2	3	4
1	-	-	-	-
2	-	-	-	-
3	-	-	-	-
4	-	-	-	-



	1	2	3	4
1	-	-	-	-
2	-	-	-	-
3	-	-	-	-
4	-	-	-	-



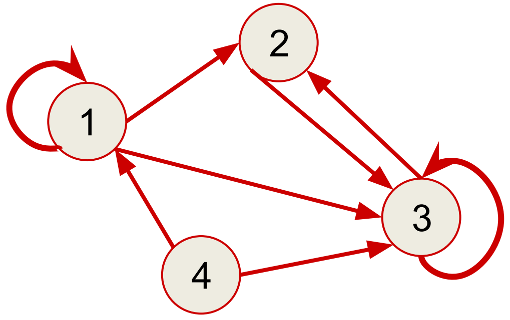
	1	2	3	4	5
1	-	-	-	-	-
2	-	-	-	-	-
3	-	-	-	-	-
4	-	-	-	-	-
5	-	-	-	-	-

ALGORITMOS

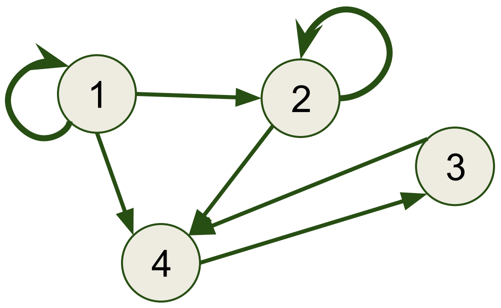
REVISÃO - REPRESENTAÇÃO DE GRAFOS

Exercícios

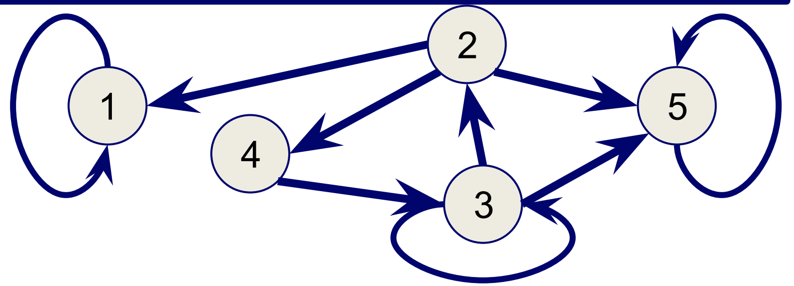
- Faça uma matriz de adjacência para os seguintes grafos



	1	2	3	4
1	1	1	1	0
2	0	0	1	0
3	0	1	1	0
4	1	0	1	0



	1	2	3	4
1	-	-	-	-
2	-	-	-	-
3	-	-	-	-
4	-	-	-	-



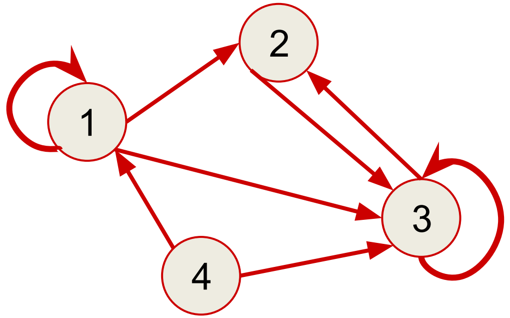
	1	2	3	4	5
1	-	-	-	-	-
2	-	-	-	-	-
3	-	-	-	-	-
4	-	-	-	-	-
5	-	-	-	-	-

ALGORITMOS

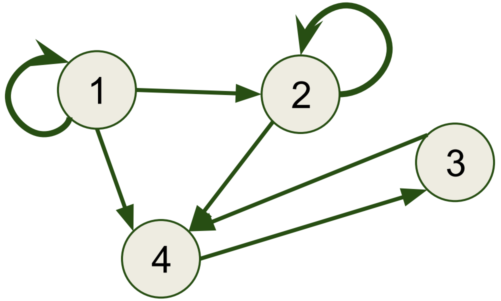
REVISÃO - REPRESENTAÇÃO DE GRAFOS

Exercícios

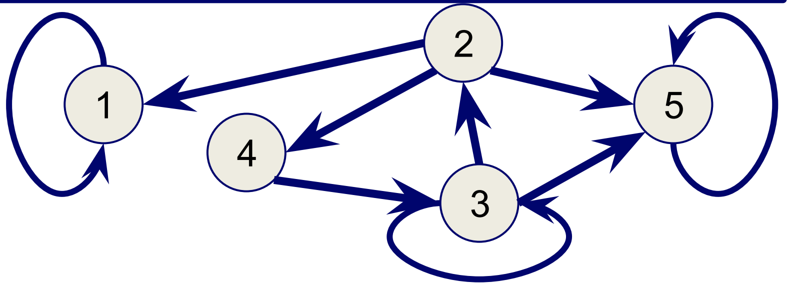
- Faça uma matriz de adjacência para os seguintes grafos



	1	2	3	4
1	1	1	1	0
2	0	0	1	0
3	0	1	1	0
4	1	0	1	0



	1	2	3	4
1	1	1	0	1
2	0	1	0	1
3	0	0	0	1
4	0	0	1	0

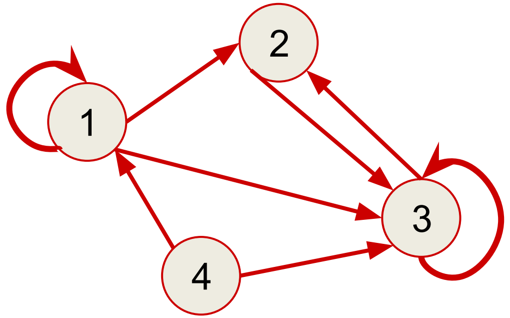


	1	2	3	4	5
1	-	-	-	-	-
2	-	-	-	-	-
3	-	-	-	-	-
4	-	-	-	-	-
5	-	-	-	-	-

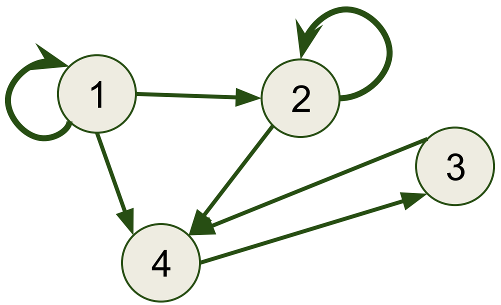
REVISÃO - REPRESENTAÇÃO DE GRAFOS

Exercícios

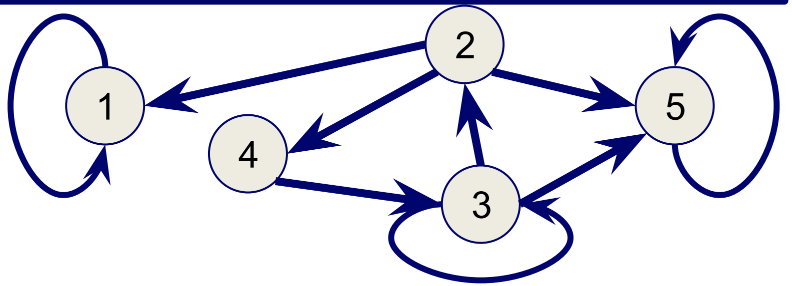
- Faça uma matriz de adjacência para os seguintes grafos



	1	2	3	4
1	1	1	1	0
2	0	0	1	0
3	0	1	1	0
4	1	0	1	0



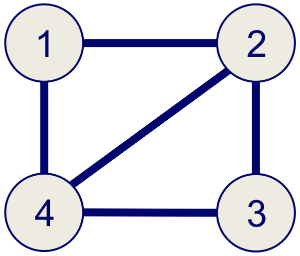
	1	2	3	4
1	1	1	0	1
2	0	1	0	1
3	0	0	0	1
4	0	0	1	0



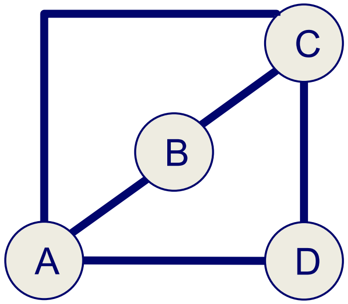
	1	2	3	4	5
1	1	0	0	0	0
2	1	0	0	1	1
3	0	1	1	0	1
4	0	0	1	0	0
5	0	0	0	0	1

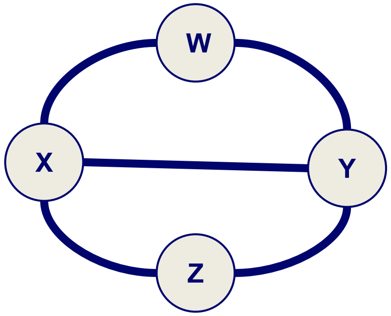
Grafos isomorfos

- Dois grafos, “**G1(V1, A1)**” e “**G2(V2, A2)**”, são ditos “**isomorfos**” se existe uma função que faça o mapeamento de vértices e arestas de modo que os dois grafos se tornem coincidentes



Grau
f(1) = A
f(2) = B
f(3) = C
f(4) = D

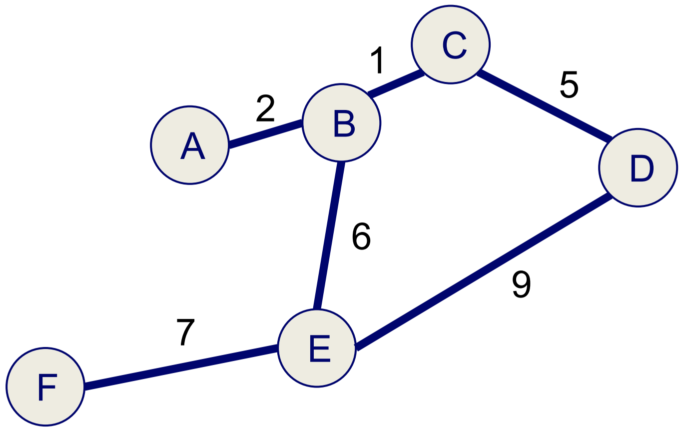


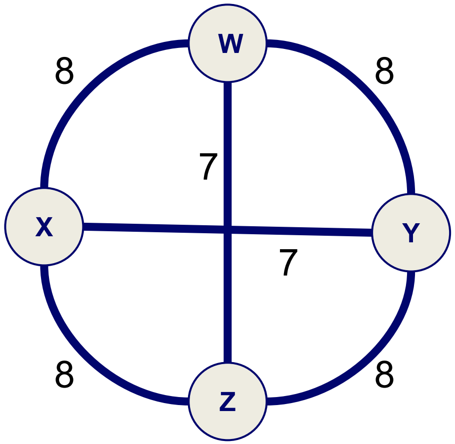


Grau
f(1) = X
f(2) = W
f(3) = Y
f(4) = Z

Grafo ponderado

- É o grafo que possui "**pesos**" associados a cada uma de suas arestas

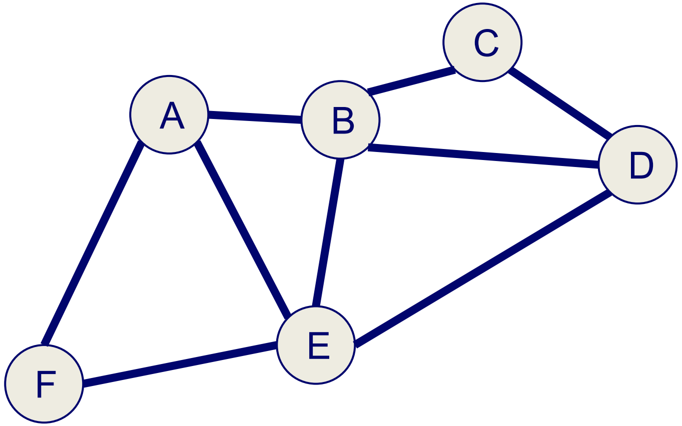




Problema de otimização

- É um problema em que cada solução possível tem um valor associado e desejamos encontrar a melhor solução com relação a esse valor.

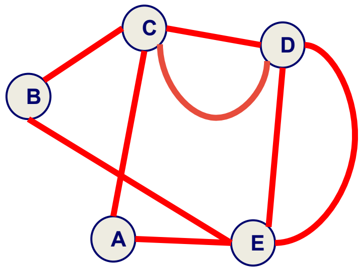
- **Exemplo:** Dado um grafo e dois vértices **f** e **a**, deve-se encontrar o caminho de **f** até **a** que tenha o menor número de arestas.



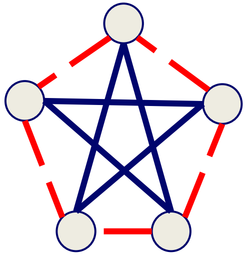
Problema de decisão

- Problemas em que a resposta é simplesmente **sim** ou **não**
- Existe uma solução para um determinado problema?

- **Exemplo**
  - Problema do **grafo euleriano**



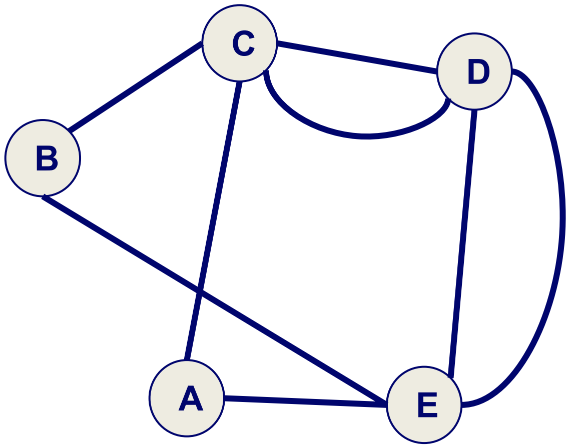
  - Problema do **grafo hamiltoniano**



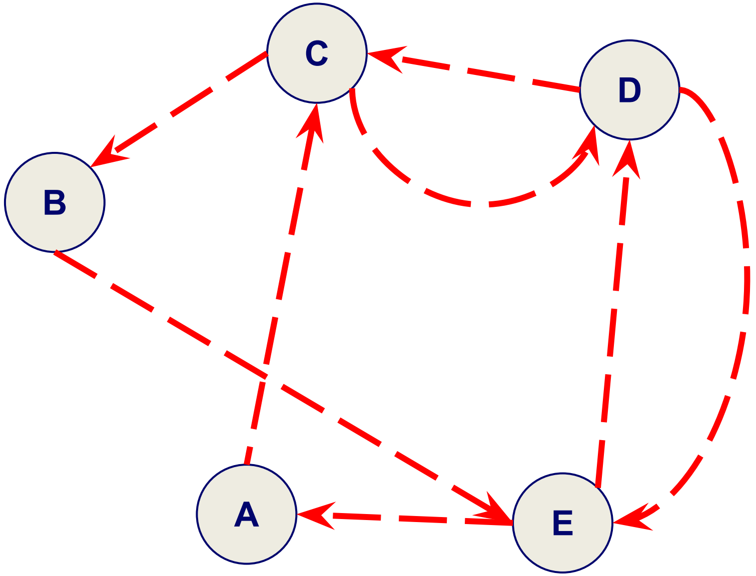
REVISÃO - GRAFOS

Grafo euleriano

- É o grafo que possui um "**ciclo**" que visita cada "**aresta**" apenas uma vez



GRAFO EULERIANO

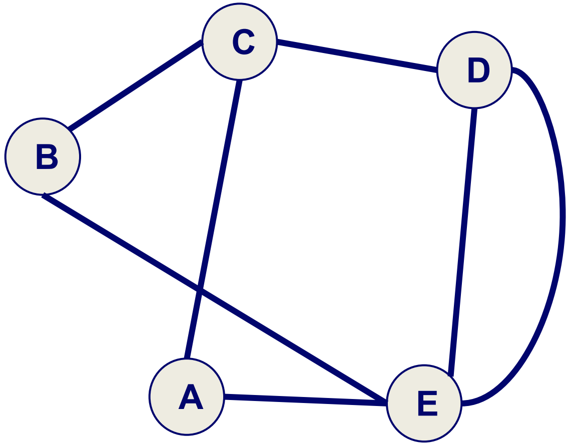


CICLO EULERIANO
"(D,E), (E,D), (D,C), (C,B), (B,E), (E,A), (A,C), (C,D)"

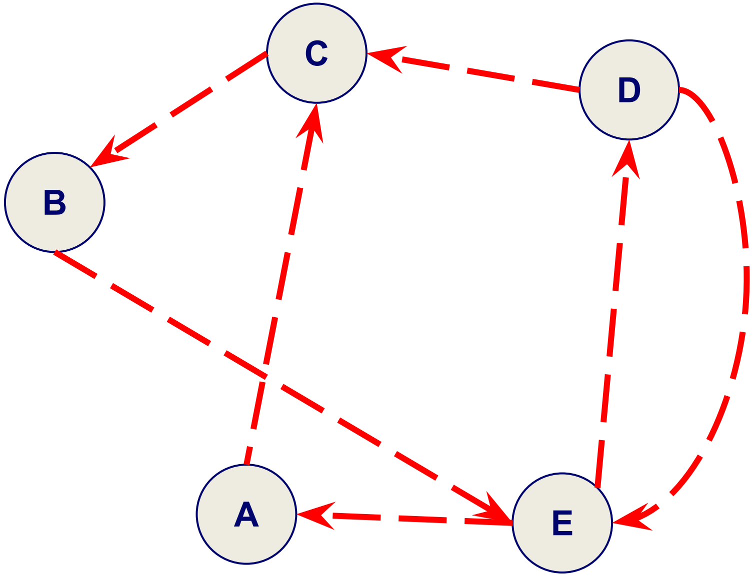
Euler mostrou que um **grafo conexo** tem um **ciclo euleriano** se e somente se ele **não** tem vértices de **grau ímpar.**

Grafo semi-euleriano

- É o grafo que possui um “**caminho**” que visita cada “**aresta**” apenas uma vez



GRAFO SEMI-EULERIANO

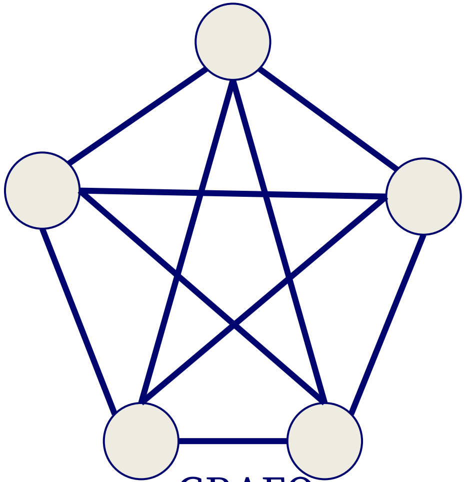


CAMINHO EULERIANO
“**(D,E), (E,D), (D,C), (C,B), (B,E), (E,A), (A,C)**”

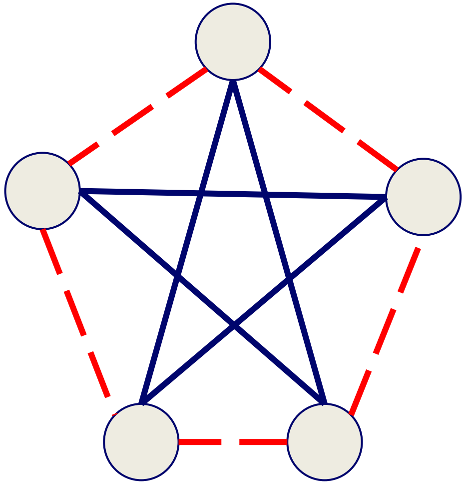
REVISÃO - GRAFOS

Grafo hamiltoniano

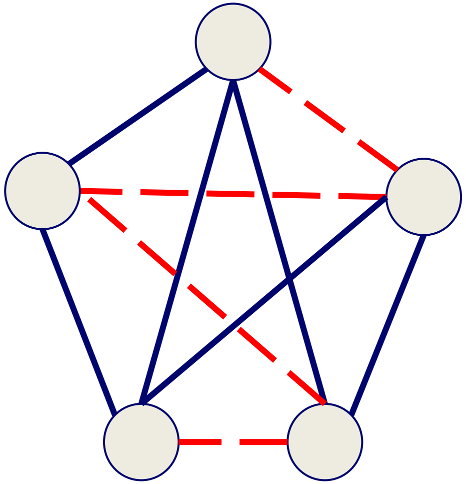
- É o grafo que possui um “**caminho**” que visita cada “**vértice**” apenas uma vez
  - Sua detecção é uma tarefa extremamente árdua
- Um “**ciclo hamiltoniano**” visita cada vértice do grafo uma única vez, terminando no vértice de início.



GRAFO HAMILTONIANO



CICLO HAMILTONIANO



CAMINHO HAMILTONIANO

Problema de otimização x problema de decisão

Exemplo:

- Dado um grafo e dois vértices **u** e **v**, encontrar o caminho de **u** até **v** que tenha o **menor número de arestas.**

- Dado um grafo e dois vértices **u** e **v,** existe um caminho de **u** até **v** que tenha **menos de 8 arestas?**

Problema de otimização x problema de decisão

Exemplo:

- Dado um grafo e dois vértices **u** e **v**, encontrar o caminho de **u** até **v** que tenha o **menor número de arestas.**

Otimização

- Dado um grafo e dois vértices **u** e **v**, existe um caminho de **u** até **v** que tenha **menos de 8 arestas?**

Decisão

ALGORITMO DETERMINÍSTICO X ALGORITMO NÃO DETERMINÍSTICO

Algoritmo Determinístico

- Para as mesmas entradas de um problema, teremos sempre as mesmas saídas.

Algoritmo Não Determinístico

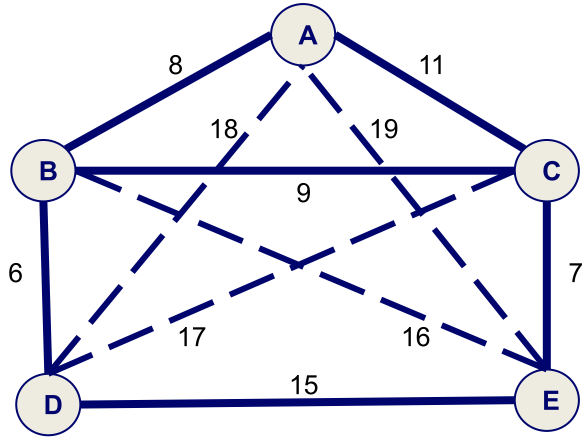
- Pode fornecer diferentes saídas para a mesma entrada em diferentes execuções.

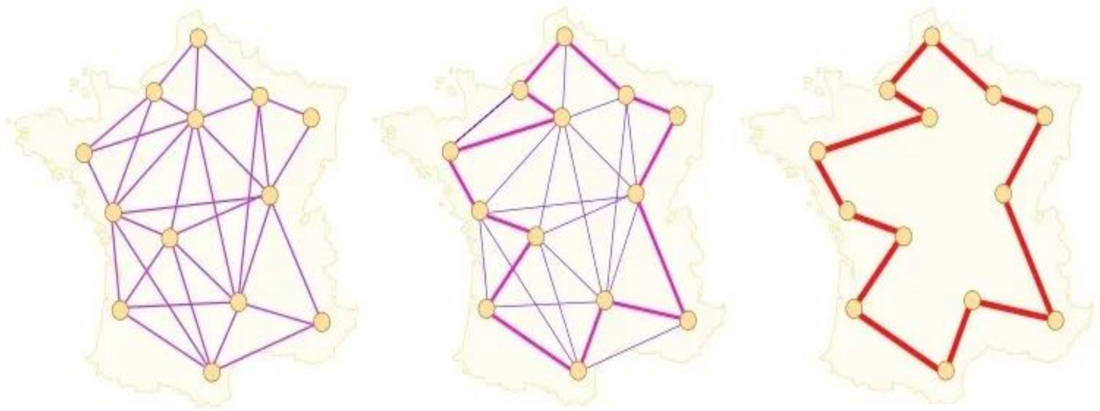
O PROBLEMA DO CAIXEIRO VIAJANTE

Problema clássico

- Conhecido como “**Travelling Salesman Problem**” (**TSP**)
- Um caixeiro viajante precisa visitar “**n**” cidades diferentes. Não importa a ordem com que as cidades são visitadas.De cada cidade pode-se ir diretamente a qualquer outra.

- Deve-se descobrir **a rota** que torna **mínima** a viagem total do caixeiro, que sairá de uma cidade, passará por **todas** as demais exatamente uma vez e retornará à cidade de início.





Fonte: Logistica Descomplicada.com

O PROBLEMA DO CAIXEIRO VIAJANTE

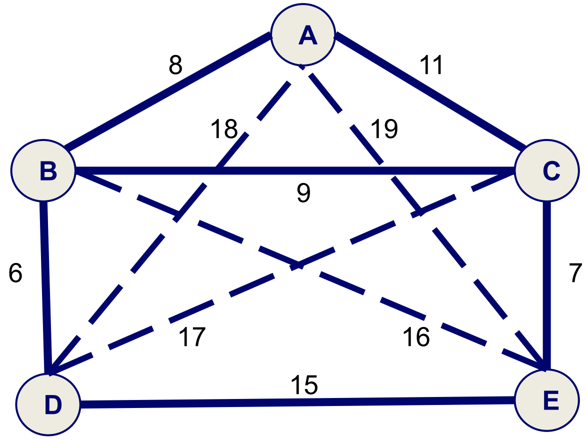
Problema clássico

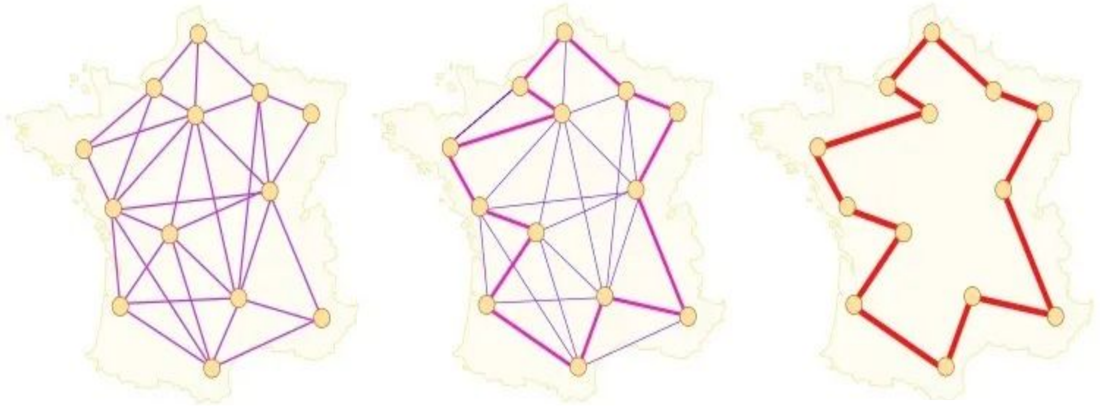
- É um problema de **otimização combinatória**
  - Rotas de transporte como: ônibus municipais, chamadas de emergência, entregras de produtos, etc.

- Como determinar todas as rotas do caixeiro?
- Como saber qual delas é a menor?

- Para enumerar todas as rotas:

(n-1)!

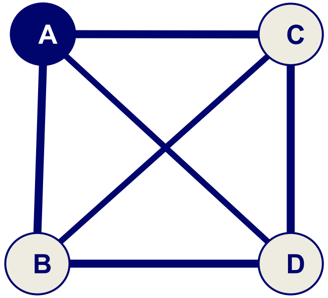




Fonte: Logistica Descomplicada.com

Exemplo

- Para quatro cidades A, B, C e D, considere que o caixeiro saia de A, visite as demais cidades em qualquer ordem e retorne a A no menor custo.



Rotas válidas

- ABCDA
- ADCBA
- ACBDA
- ADBCA
- ABDCA
- ACDBA

- Para enumerar todas as rotas (**simétricas**): **(n-1)!**
  = (n-1)!
  = (4-1)!
  = 3!
  = (3 x 2 x 1)
  = (6)
  = 6

Fonte: Logistica Descomplicada.com

O PROBLEMA DO CAIXEIRO VIAJANTE

Crescimento da complexidade

- O problema do caixeiro é um clássico exemplo de problema de otimização combinatória.
- Uma possibilidade para resolver esse tipo de problema é reduzí-lo a um problema de enumeração.
  - Achar todas as rotas possíveis e calcular o comprimento de cada uma delas e então verificar qual é a menor.

- Encontrar o número R(n) de rotas para o caso de n cidades, usando um raciocínio combinatório simples.
- No caso de n = 4 cidades, a primeira e última posição são fixas; na segunda posição pode ser uma das 3 cidades restantes B, C e D. Na terceira posição pode ser qualquer uma das 2 restantes.
  - O número de rotas é igual a **1x3x2x1 = 3x2x1 = 6**

- **(n-1) x (n-2) x ... x 2 x 1**
- Notação de fatorial: **R(n) = (n - 1)!**

O PROBLEMA DO CAIXEIRO VIAJANTE

Exemplo

- Um computador faz 1 bilhão de operações por segundo.
- No caso de 20 cidades, o computador precisa apenas de 19 operações para dizer qual o comprimento de uma rota. Logo, ele será capaz de calcular **10^9 / 19 = 53 milhões** de rotas por segundo.
- Mas ele precisa examinar **19!** de rotas possíveis (**121.645.100.408.832.00** ou **1.2×10^{17}**).

n	rotas/segundo	(n-1)!	Cálculo total
5	250 milhões	24	insignificante
10	110 milhões	362.880	0.003 seg
15	71 milhões	87 bilhões	20 min
20	53 milhões	1.2×10^{17}	73 anos
25	42 milhoes	6.2×10^{23}	470 milhões de anos

1.2×10^{17} / (53 milhões) = 2.3×10^9 segundos
≅ 73 anos

ALGORITMOS

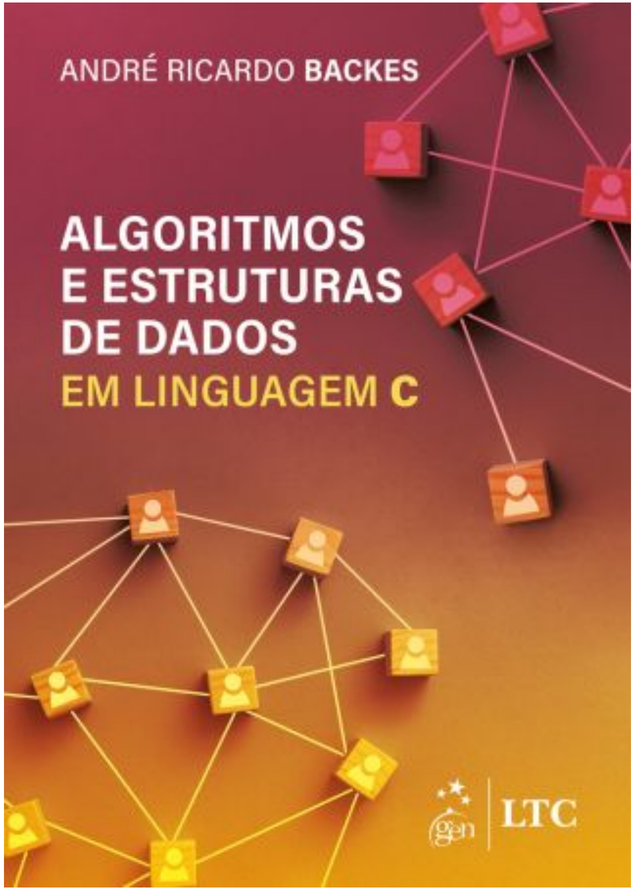
ÁRVORE

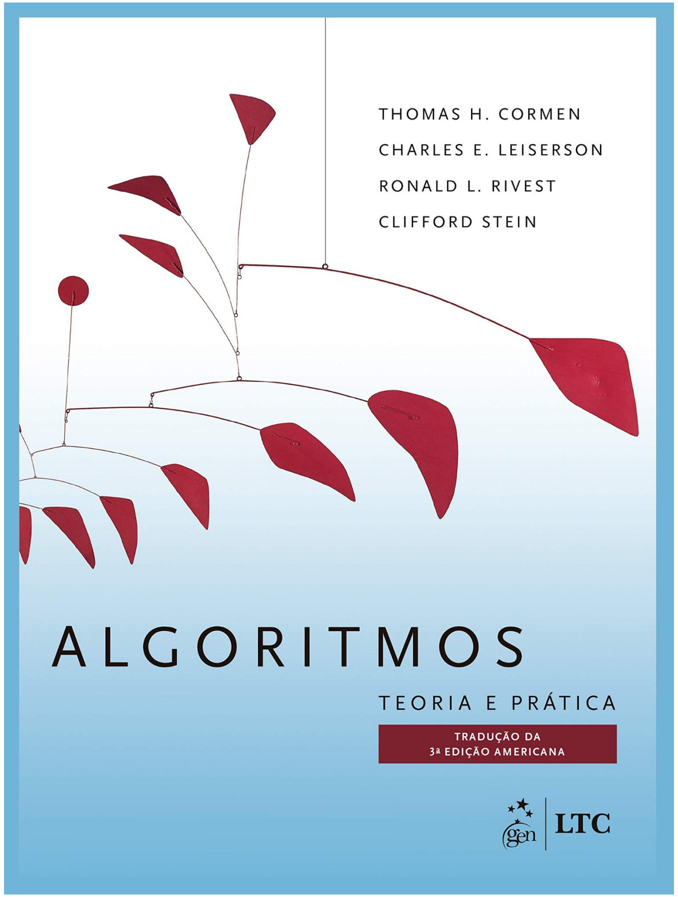
PÁGINAS DO LIVRO ALGORITMOS E ESTRUTURA DE DADOS EM LINGUAGEM C

- Capítulo 11 - Árvore - Link
  - Busca em profundidade - págs: 344 e 345
  - Busca em largura - págs: 346 e 347

ATIVIDADES

LEITURAS RECOMENDADAS





Agradecimentos

OBRIGADO.

Rafael Marinho e Silva
rafaelmarinho@unipam.edu.br