Vitor Amadeu Souza

Desenvolvendo

Estrutura de dados

programado em

Python

© 2024 by Cerne Tecnologia e Treinamento Ltda.

© 2024 by Vitor Amadeu Souza

Nenhuma parte desta publicação poderá ser reproduzida sem autorização prévia e escrita de **Cerne Tecnologia e Treinamento Ltda.** Este livro publica nomes comerciais e marcas registradas de produtos pertencentes a diversas companhias. O editor utiliza as marcas somente para fins editoriais e em benefício dos proprietários das marcas, sem nenhuma intenção de atingir seus direitos.

Julho de 2024

Direitos reservados por:
Cerne Tecnologia e Treinamento Ltda
Produção: Cerne Tecnologia e Treinamento
E-mail da Empresa: cerne@cerne-tec.com.br
Home Page: www.cerne-tec.com.br.com.br
Atendimento ao Consumidor: sac@cerne-tec.com.br
Contato com o Autor: vitor@cerne-tec.com.br



FEITO NO BRASIL

"É como árvore plantada à beira de águas correntes: Dá fruto no tempo certo e suas folhas não murcham. Tudo o que ele faz prospera!"

Sl 1:3

Cerne Tecnologia

A Cerne Tecnologia tem uma equipe preparada para desenvolvimento de projetos eletrônicos em diversas áreas: Médica, Entretenimento, Industrial, Robótica, Científica, Automobilística, Aeronáutica, etc. Trabalhamos com tecnologia microcontrolada usando o PIC, ARM, AVR, 8051, dsPIC, PIC24, PIC32 além do Arduino, Raspberry, Beaglebone etc. Desenvolvemos o projeto desde sua concepção até a entrega do produto final, passando pelas etapas de esquema elétrico, protótipo e desenvolvimento de circuito impresso.

Desenvolvemos aplicativos para smartphones/tablets Android, iOS, Blackberry, Windows Phone e no desenvolvimento de softwares a nível PC para plataforma Windows, usando ferramentas como o Visual Basic, C# e C++.

Atuamos na parte de montagem de placas, onde podemos fornecer ambos os serviços de desenvolvimento de projetos e produção ou apenas um destes.

Desenvolvemos esquemas elétricos e layout de PCI, tanto em tecnologia convencional como SMD.

Temos a flexibilidade de customizar um de nossos produtos, de modo a atender a uma necessidade específica do cliente, tornando o custo de desenvolvimento menor se comparado a construção de um projeto desde a sua fase inicial.

Desenvolvemos e fornecemos kits didáticos para diversos microcontroladores além de apostilas, livros e e-books.

Na hora de desenvolver um projeto ou equipar seu laboratório não hesite em nos contatar. Entre em contato conosco através do endereço cerne-tec.com.br para obter mais informações.

Cerne
Conhecimento para o Desenvolvimento

cerne-tec.com.br

Sumário

Capítulo I Metodologia de desenvolvimento..... 7

Capítulo II Teoria dos Grafos..... 10

Capítulo III Programação em Python..... 109

Capítulo IV Recursão.....166

Capítulo V Busca Linear.....170

Capítulo VI Bubble Sort.....176

Capítulo VII Quick sort.....181

Capítulo VIII Insertion Sort.....187

Capítulo IX Quick select.....194

Capítulo X Selection Sort.....199

Capítulo XI Busca binária.....206

Capítulo XII Pesquisa binária.....217

Capítulo XIII Pilha.....230

Capítulo XIV Fila.....239

Capítulo XV BFS.....249

Capítulo XVI DFS.....255

Capítulo XVII Lista simplesmente encadeada.....260

Capítulo XVIII Lista duplamente encadeada.....268

Capítulo XIX Heap.....278

Capítulo XX Fila prioritária.....289

Capítulo XXI Hashtable Direto.....304

Capítulo XXII Hashtable Aberto.....315

Capítulo XXIII Hashtable em Lista....329
Capítulo XXIV Programação paralela....342
Capítulo XXV Programação Assíncrona com threading....352
Capítulo XVI Programação Assíncrona com asyncio framework....361
Capítulo XVII Árvore binária....372
Capítulo XVIII Árvore AVL....377
Capítulo XXIX Árvore B....392
Capítulo XXX Árvore Rubro-Negra....403
Capítulo XXXI Árvore Trie....411
Capítulo XXXII Heurística do Vizinho mais próximo....419
Capítulo XXXIII Algoritmo de Euclides....427
Capítulo XXXIV Algoritmo de Kruskal....432
Capítulo XXXV Algoritmo de Prim....443
Capítulo XXXVI Algoritmo de Dijkstra....453
Capítulo XXXVII Algoritmo de Bellman-Ford....465
Capítulo XXXVIII Algoritmo de Floyd-Warshall....478
Capítulo XXXIX YOLO....516
Capítulo XL Plotando gráficos....526
Capítulo XLI Função Sigmoide....550
Capítulo XLII Perceptron....552

Capítulo I
Metodologia de desenvolvimento

1. Introdução

Este livro oferece um panorama abrangente das principais estruturas e algoritmos de dados e técnicas de programação, proporcionando um guia essencial para o entendimento e a aplicação de conceitos fundamentais na ciência da computação e programação.

Iniciamos com o tema da Teoria dos Grafos, essencial para a análise e otimização de redes e estruturas conectadas. Em seguida, exploramos as práticas de Programação em Python, a linguagem amplamente utilizada por sua simplicidade e poder, e avançamos para tópicos mais complexos, como Recursão e Busca Linear, técnicas fundamentais para a solução de problemas e análise de algoritmos.

A seção dedicada a classificação de Algoritmos cobre diversos métodos de ordenação, incluindo Bubble Sort, Quick Sort, Insertion Sort, Quick Select, e Selection Sort, que são fundamentais para a organização e gerenciamento de dados. O texto também aborda técnicas de Busca Binária e Pesquisa Binária, que são essenciais para a recuperação eficiente de informações em estruturas ordenadas.

O livro prossegue com a introdução de estruturas de dados mais avançadas, como Pilha e Fila, além de algoritmos de busca em grafos, especificamente BFS e DFS. Em seguida, discutimos as Listas Encadeadas e suas variações, como Lista Simplesmente Encadeada e Lista Duplamente Encadeada, proporcionando uma compreensão mais profunda das operações em listas dinâmicas.

Avançando, apresentamos o Heap e a Fila Prioritária, estruturas que otimizam a manipulação de dados com prioridades variadas. As seções seguintes discutem diferentes abordagens para tabelas hash, incluindo Hashtable Direto, Hashtable Aberto e Hashtable em Lista, fundamentais para a gestão eficiente de dados em mapas associativos.

Os capítulos seguintes focam em técnicas avançadas, como Programação Paralela e Programação Assíncrona, que são importantes para a criação de sistemas eficientes e responsivos. Abordamos também a implementação e os conceitos por trás de Árvore Binária, Árvore AVL, Árvore B, Árvore Rubro-Negra, e Árvore Trie, que são essenciais para a organização hierárquica e acesso rápido a dados.

Finalmente, exploramos técnicas heurísticas, como a Heurística do Vizinho Mais Próximo, e algoritmos clássicos de grafos, como o Algoritmo de Euclides, Algoritmo de Kruskal, Algoritmo de Prim,

Algoritmo de Dijkstra, Algoritmo de Bellman-Ford, e Algoritmo de Floyd-Warshall, que são fundamentais para a resolução de problemas complexos de otimização e caminhos em grafos. O livro conclui com uma introdução ao YOLO para detecção de objetos e técnicas de Plotagem de Gráficos e Função Sigmoide, abordando a visualização de dados e a modelagem matemática utilizada em redes neurais.

Cada capítulo deste livro é projetado para fornecer uma compreensão teórica detalhada, juntamente com exemplos práticos e implementações em Python, visando equipar o leitor com o conhecimento e as habilidades necessários para enfrentar desafios computacionais e avançar em sua jornada na ciência da computação.

Capítulo II

Teoria dos Grafos

1. História

O termo grafo foi introduzido por James Joseph Sylvester através de um artigo publicado na Nature em 1877.

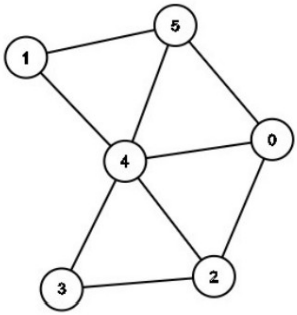


Entretanto, a resolução do problema das 7 Pontes de Königsberg em 1736 por Leonhard Euler é referida como a primeira publicação sobre o tema.



2. O que é um grafo?

A definição formal de grafo é: Conjunto de vértices (ou nodos), interconectados dois a dois por arestas (ou arcos).



Podemos abstrair este conceito que possibilita a codificação entre pares de objetos. Quais objetos? Qualquer um! Ex: Pessoas, estados, filmes, páginas web, aeroportos, átomos, ilha em PCI, países etc.

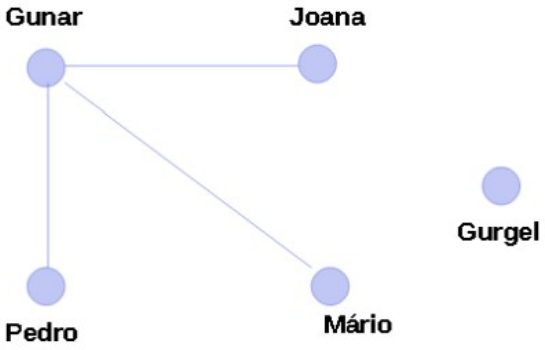
Quais relacionamentos? Qualquer um! Ex: Idioma, ideologia, amizade, profissão, genealogia, trilha em PCI, amizade etc.

Desta forma, um novo desenho pode ser feito no qual os objetos estão relacionados aos vértices do grafo e relacionamentos as arestas.

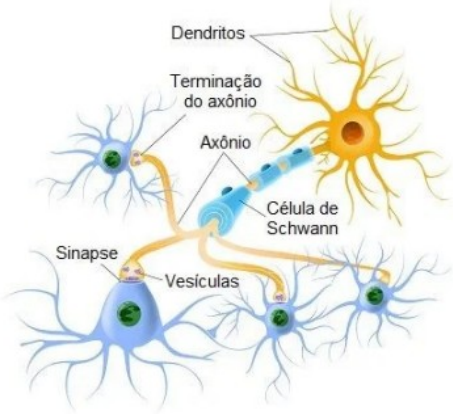


3. Aplicações

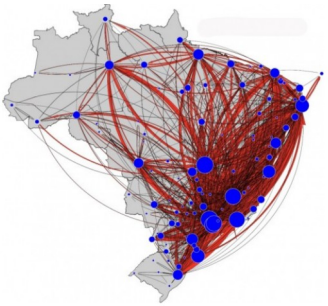
Pela representação do grafo abaixo fica claro observar que Gunar é o que possui mais amigos.



Imagine a quantidade de relacionamentos (arestas) ou sinapses há entre os neurônios (vértice) em uma rede neural.

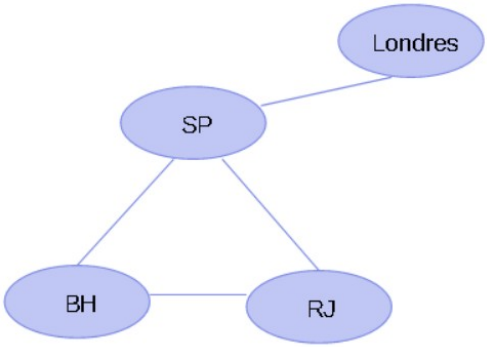


Pela figura abaixo podemos observar os relacionamentos entre os diversos aeroportos brasileiros.

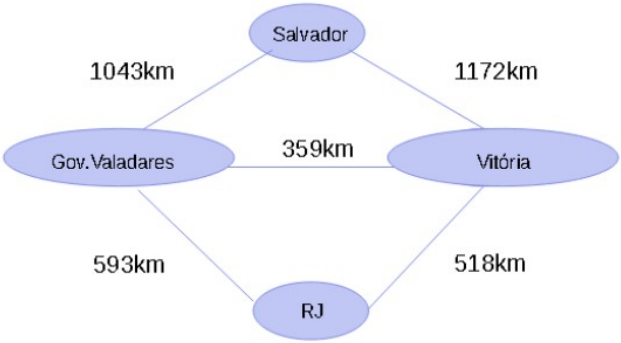


Fonte: https://journals.openedition.org/confins/46949?lang=pt

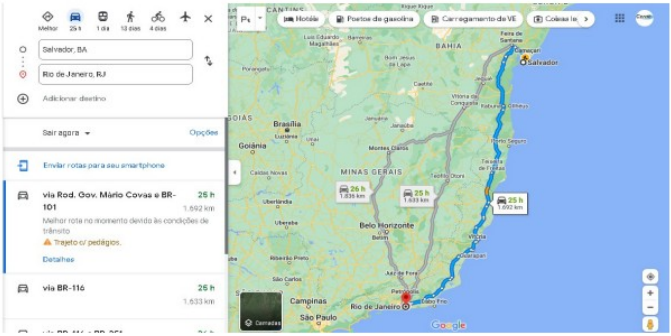
Observando a figura abaixo podemos ver que a relação com Londres é feita apenas através do aeroporto de SP.



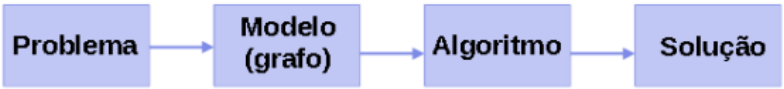
Voar entre qualquer cidade é possível? Qual o menor número de voos entre cidades? A resposta para tais indagações é dada através de um algoritmo. Vejamos outro exemplo prático: Qual a menor distância para uma viagem de carro entre a cidade do Rio de Janeiro e Salvador?



Usando o algoritmo do Google Maps, observe o resultado fornecido pelo algoritmo.



Podemos elaborar o diagrama abaixo, onde partindo de um problema e passando por um modelo dado por um grafo, é feito um algoritmo para se chegar a uma solução.



4. Representação de um grafo

Um grafo G=(V,E) é dado pelo conjunto de vértices (V) e arestas (E).

G=(V,E)

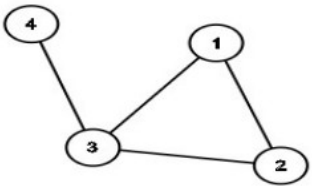
V= Conjunto de objetos (vértices ou nós)

E= Conjunto de par ordenado que forma a aresta ou arco

Como sugestão para desenhar o grafo, recomendamos a visitação ao endereço https://graphonline.ru/en/. Acompanhe alguns exemplos.

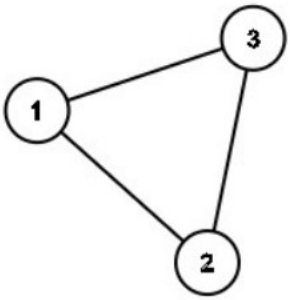
1)

```
G=(V,E)
 V={1,2,3,4}
 E={(1,2),(1,3),(2,3),(3,4)}
```



2)

```
G=(V,E)
 V={1,2,3}
 E={(1,2),(1,3),(2,3)}
```

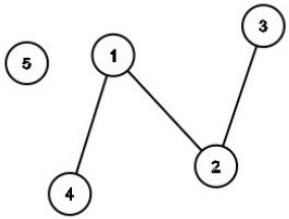


3)

G=(V,E)

V={1,2,3,4,5}

E={(1,2),(3,2),(1,4)}

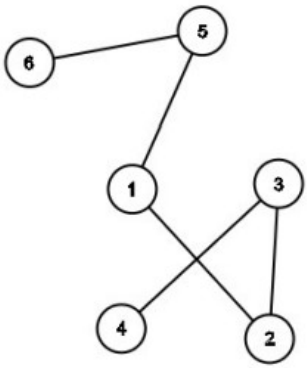


4)

G=(V,E)

V={1,2,3,4,5,6}

E={(1,2),(2,3),(3,4),(5,6),(5,1)}

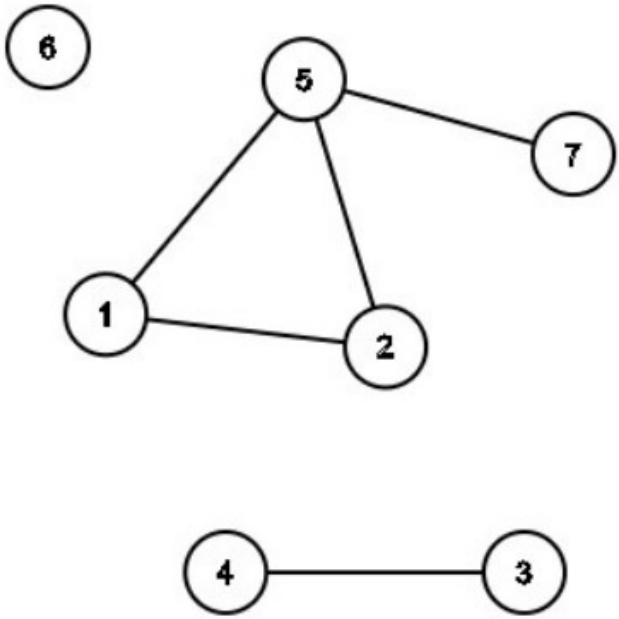


5)

G=(V,E)

V={1,2,3,4,5,6,7}

E={(1,2),(1,5),(2,5),(3,4),(5,7)}



5. Ordem de um grafo

A ordem de um grafo é dado pelo número de vértices V que formam o grafo G=(V,E). Acompanhe alguns exemplos.

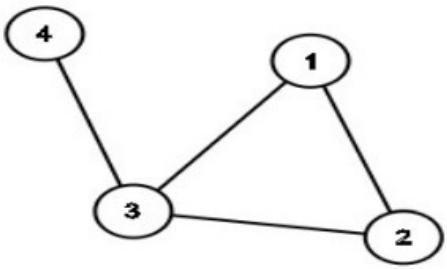
1)

$G=(V,E)$

$V=\{1,2,3,4\}$

$E=\{(1,2),(1,3),(2,3),(3,4)\}$

$|V(G)|=4$



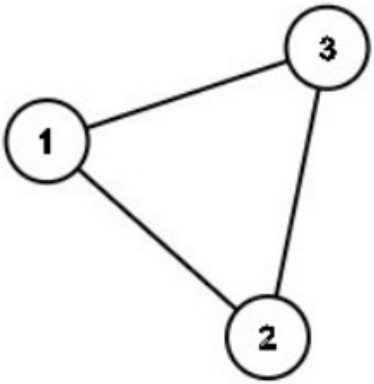
2)

G=(V,E)

V={1,2,3}

E={(1,2),(1,3),(2,3)}

|V(G)|=3



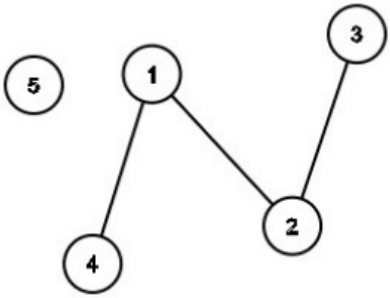
3)

G=(V,E)

V={1,2,3,4,5}

E={(1,2),(3,2),(1,4)}

|V(G)|=5



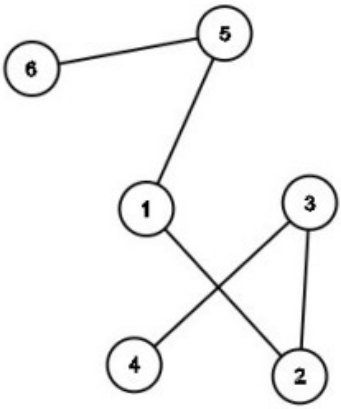
4)

G=(V,E)

V={1,2,3,4,5,6}

E={(1,2),(2,3),(3,4),(5,6),(5,1)}

|V(G)|=6



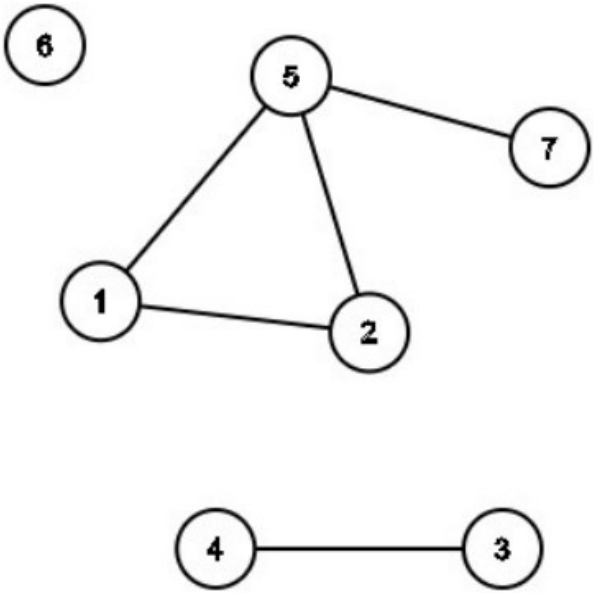
5)

G=(V,E)

V={1,2,3,4,5,6,7}

E={(1,2),(1,5),(2,5),(3,4),(5,7)}

|V(G)|=7

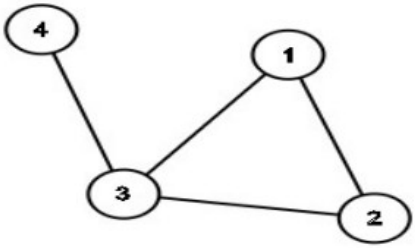


6. Número de arestas

É a quantidade de arestas que formam um grafo.

1)

```
G=(V,E)
  V={1,2,3,4}
  E={(1,2),(1,3),(2,3),(3,4)}
  |V(G)|=4
  |E(G)|=4
```



2)

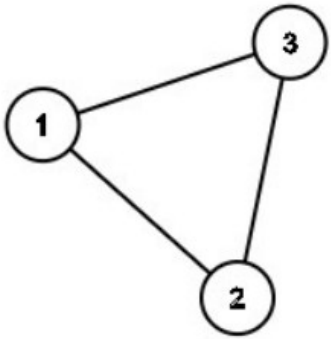
G=(V,E)

V={1,2,3}

E={(1,2),(1,3),(2,3)}

|V(G)|=3

|E(G)|=3



3)

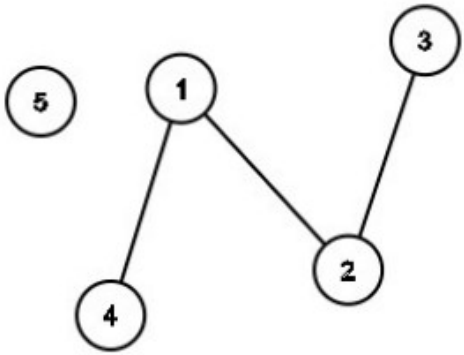
G=(V,E)

V={1,2,3,4,5}

E={(1,2),(3,2),(1,4)}

|V(G)|=5

|E(G)|=3



4)

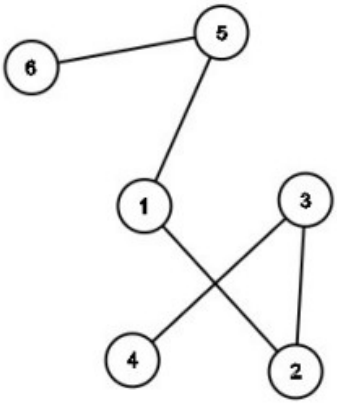
G=(V,E)

V={1,2,3,4,5,6}

E={(1,2),(2,3),(3,4),(5,6),(5,1)}

|V(G)|=6

|E(G)|=5



5)

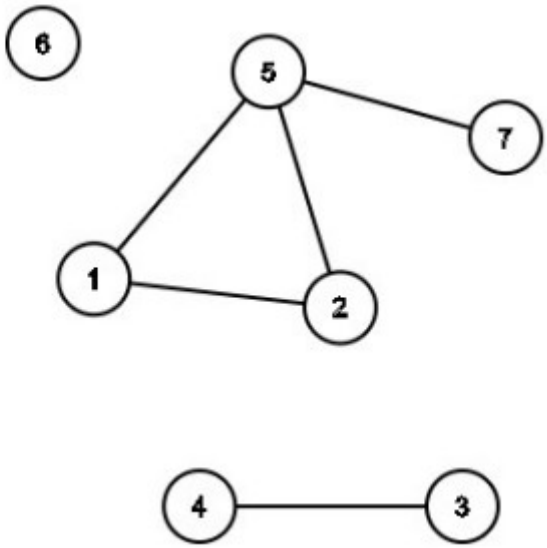
G=(V,E)

V={1,2,3,4,5,6,7}

E={(1,2),(1,5),(2,5),(3,4),(5,7)}

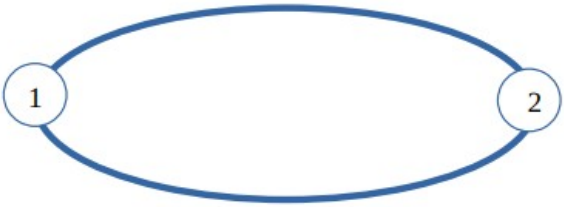
|V(G)|=7

|E(G)|=5



7. Multigrafos

Quando há arestas múltiplas ou paralelas chamamos de multigrafo. Acompanhe o exemplo:



$G=(V,E)$

$V=\{1,2\}$

$E=\{(1,2),(2,1)\}$

$|V(G)|=2$

$|E(G)|=2$

8. Grafo trivial

Quando há vértice e não há arestas temos um grafo trivial. Acompanhe o exemplo:



$G=(V,E)$

$V=\{v1\}$

$E=\emptyset$

$|V(G)|=1$

$|E(G)|=0$

9. Grafo vazio

É aquele não há vértices nem aresta.

$$G=(\emptyset,\emptyset)$$

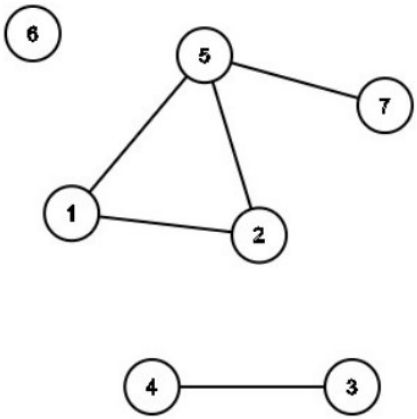
10. Laço

Quando há uma aresta que possui o mesmo vértice como extremo.



11. Vértices adjacentes

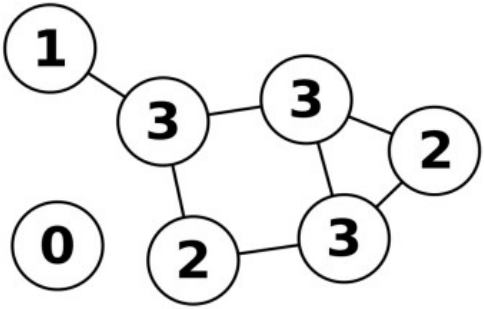
Esta situação ocorre quando um vértice for o extremo de uma mesma aresta.



No exemplo acima o vértice 3 é adjacente a 4 e este adjacente a 3. Entretanto, o vértice 5 não é adjacente a 4 e 7 não é adjacente a 2.

12. Grau ou valência de um vértice

Refere-se ao número de arestas incidentes para com o vértice.



Teorema do aperto de mãos: a soma dos graus dos vértices é sempre igual ao dobro do número de arestas.

$$2|E| = \sum_{u \in V} d_G(u).$$

No exemplo acima pode-se observar que a soma dos graus dos vértices é igual a 14. Neste caso, há 7 arestas.

1) G=(V,E)

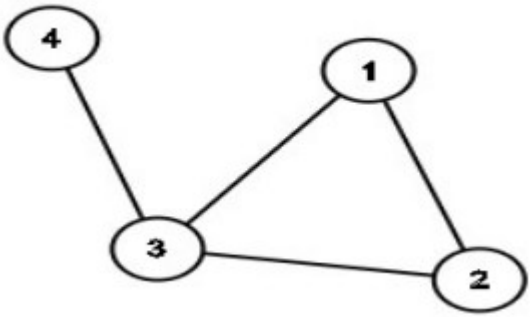
V={1,2,3,4}

E={(1,2),(1,3),(2,3),(3,4)}

|V(G)|=4

|E(G)|=4

dG(1)=2, dG(2)=2, dG(3)=3, dG(4)=1



2) G=(V,E)

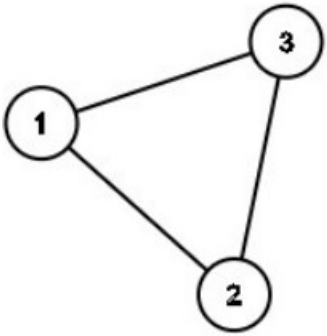
V={1,2,3}

E={(1,2),(1,3),(2,3)}

|V(G)|=3

|E(G)|=3

dG(1)=2, dG(2)=2, dG(3)=2



3) G=(V,E)

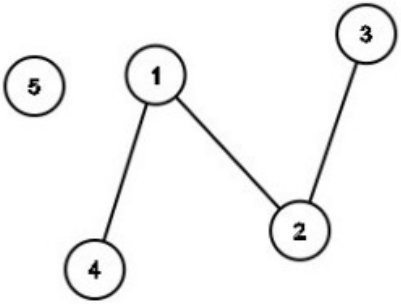
V={1,2,3,4,5}

E={(1,2),(3,2),(1,4)}

|V(G)|=5

|E(G)|=3

dG(1)=2, dG(2)=2, dG(3)=1, dG(4)=1, dG(5)=0



4) G=(V,E)

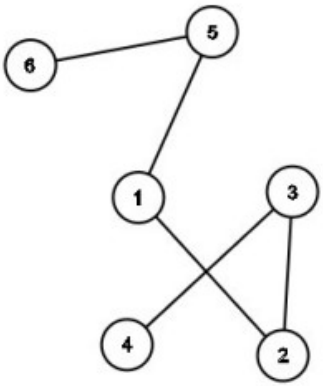
V={1,2,3,4,5,6}

E={(1,2),(2,3),(3,4),(5,6),(5,1)}

|V(G)|=6

|E(G)|=5

dG(1)=2, dG(2)=2, dG(3)=2, dG(4)=1, dG(5)=2, dG(6)=1



5) G=(V,E)

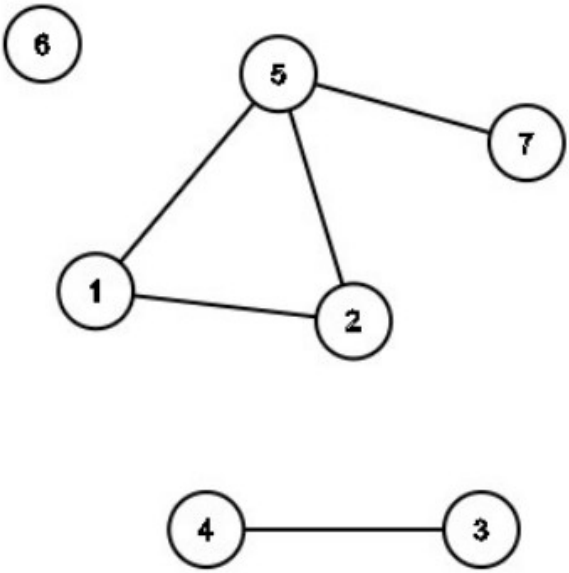
V={1,2,3,4,5,6,7}

E={(1,2),(1,5),(2,5),(3,4),(5,7)}

|V(G)|=7

|E(G)|=5

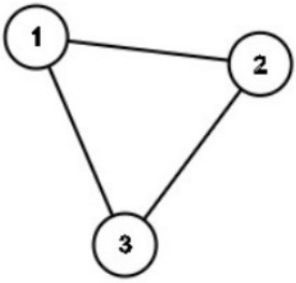
dG(1)=2, dG(2)=2, dG(3)=1, dG(4)=1, dG(5)=3, dG(6)=0, dG(7)=1



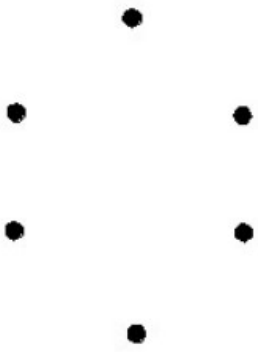
13. Grafo regular

Quando todos os vértices possuem o mesmo grau temos um grafo regular. Neste caso, chamamos o grafo de n-regular, onde n representa o grau de cada vértice.

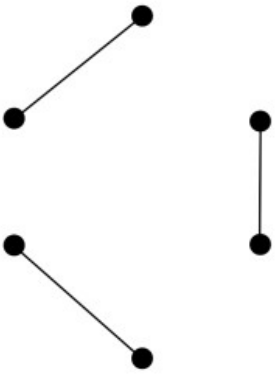
1) G é 2-regular



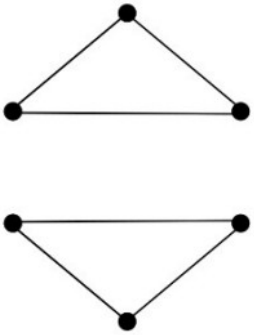
2) G é 0-regular

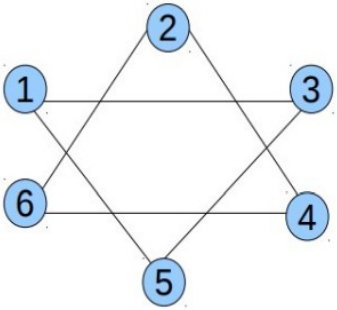


3) G é 1-regular

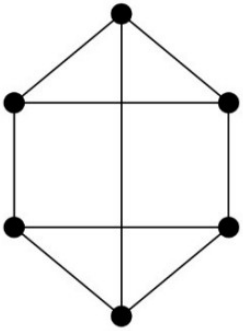


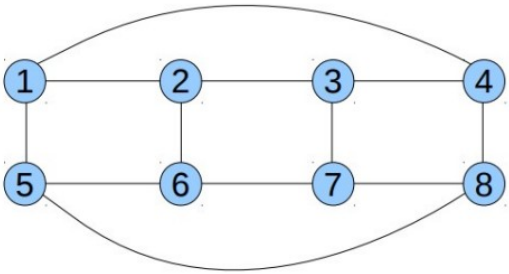
4) G é 2-regular





5) G é 3-regular





14. Grafo completo

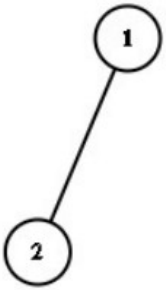
É aquele com todos os vértices adjacentes, ou seja, aresta conectando-os. Chamamos este grafo de Kn em que n representa o número de vértices. Em um grafo completo o número de arestas é dado por:

$$e=n(n-1)/2$$

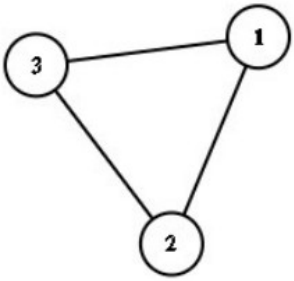
1) K_1, n=1, e=0



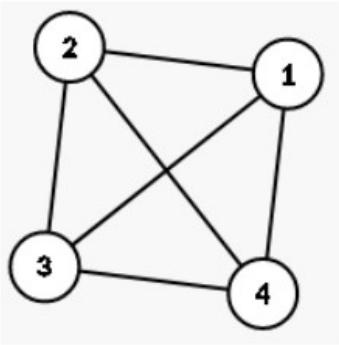
2) K_2**,** n=2, e=1



3) K_3**,** n=3, e=3



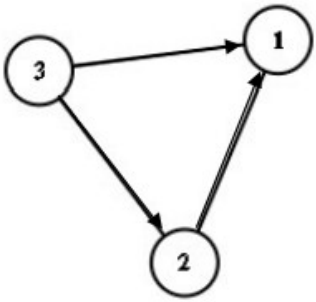
4) K_4, n=4, e=6



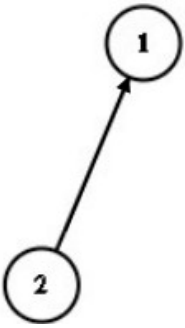
15. Grafo orientado ou dígrafo

É aquele cuja aresta e=(x,y) possui uma única direção, como mostra os exemplos abaixo.

1)

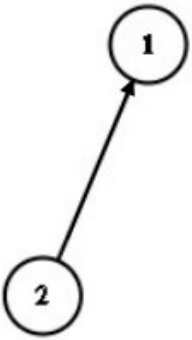


2)

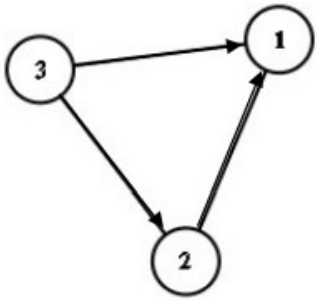


16. Aresta convergente e divergente

Irá depender da direção da aresta em um dígrafo e seu ponto de referência. Na figura abaixo, a aresta (1,2) é divergente do vértice 1, entretanto é convergente ao vértice 2.

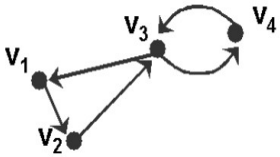


No próximo exemplo a aresta (2,3) é convergente ao vértice 2 e divergente ao vértice 3.



17. Grau de um grafo orientado

Depende do número de arestas divergentes e convergente de um vértice, onde $d_{out}(v)$ é o número de arestas divergentes e $d_{in}(v)$ o de arestas convergentes. Como exemplo, acompanhe o próximo grafo.




```
din(v3)=2

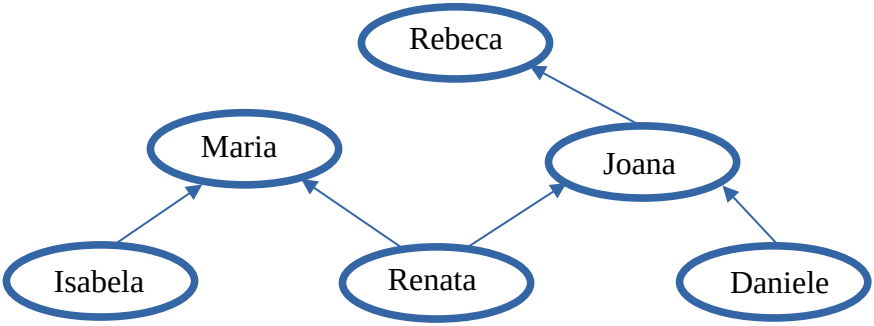
dout(v3)=2

din(v1)=din(v2)=din(v4)=1

dout(v1)=dout(v2)=dout(v4)=1
```

18. Sumidouro ou sorvedouro

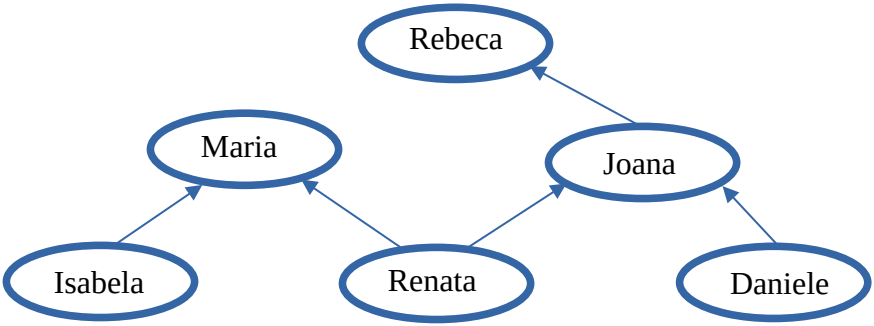
Sempre que o $d_{out}(v)$ de um vértice for 0 este será considerado um sumidouro. No exemplo abaixo, quais os vértices sumidouros?



No grafo acima os vértices Maria e Rebeca possuem $d_{out}(v)=0$, logo estes são sumidouro.

19. Fonte

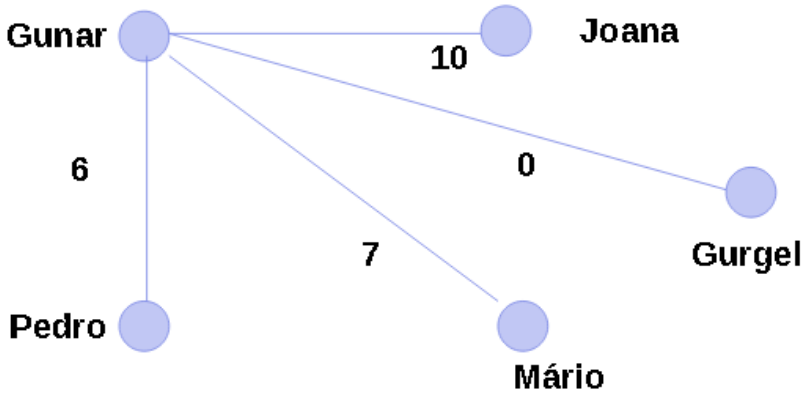
Sempre que o $d_{in}(v)$ de um vértice for 0 este será considerado uma fonte. No exemplo abaixo, quais os vértices fonte?



No grafo acima os vértices Isabela, Renata e Daniele possuem $d_{in}(v)=0$, logo estes são fonte.

20. Grafo valorado

Além dos vértices há um valor real valorando a ligação entre os objetos, neste caso formando uma tripla (v,w,z) onde v e w são vértices e z um número real. No grafo abaixo, além do relacionamento de Gunar com outras pessoas há um valor de 0 a 10 que estabelece a ligação entre eles.



Fica claro pelo grafo que o apreço maior de Gunar em seu relacionamento é por Joana e o menor por Gurgel.

21. Matriz adjacência

Com um grafo com poucos vértices fica relativamente fácil representá-lo de forma gráfica. Entretanto, conforme o número de pontos cresce faz-se necessário obter novos meios de representação, sendo a matriz adjacência uma forma de representação. Além disso, a matriz adjacência é uma forma de fornecer dados ao computador.

Para formar a matriz distribui-se o número de vértices em linhas e colunas e deixa-se o elemento em 1 caso haja aresta adjacente ao vértice e 0 caso contrário. Matriz n x n (número de vértices):

a_{ij}=1, se existe aresta

a_{ij}=0, caso contrário

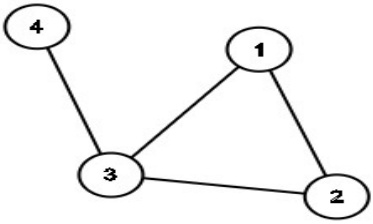
Acompanhe os exemplos propostos.

1)

G=(V,E)

V={1,2,3,4}

E={(1,2),(1,3),(2,3),(3,4)}

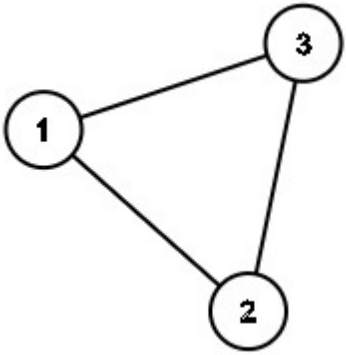


	1	2	3	4
1	0	1	1	0
2	1	0	1	0
3	1	1	0	1
4	0	0	1	0

Podemos observar algumas propriedades interessantes nesta matriz, como o fato de ser simétrica e a soma da linha fornecer o grau do vértice.

2)

```
G=(V,E)
  V={1,2,3}
  E={(1,2),(1,3),(2,3)}
```



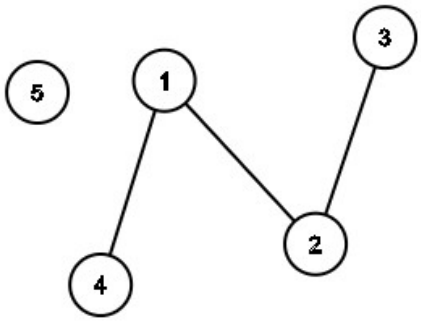
	1	2	3
1	0	1	1
2	1	0	1
3	1	1	0

3)

G=(V,E)

V={1,2,3,4,5}

E={(1,2),(3,2),(1,4)}



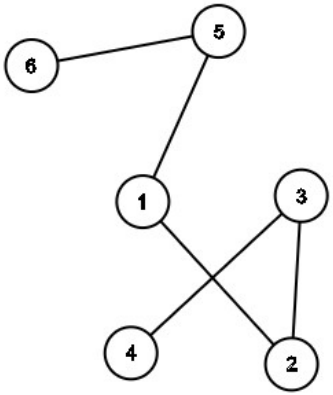
	1	2	3	4	5
1	0	1	0	1	0
2	1	0	1	0	0
3	0	1	0	0	0
4	1	0	0	0	0
5	0	0	0	0	0

4)

G=(V,E)

V={1,2,3,4,5,6}

E={(1,2),(2,3),(3,4),(5,6),(5,1)}



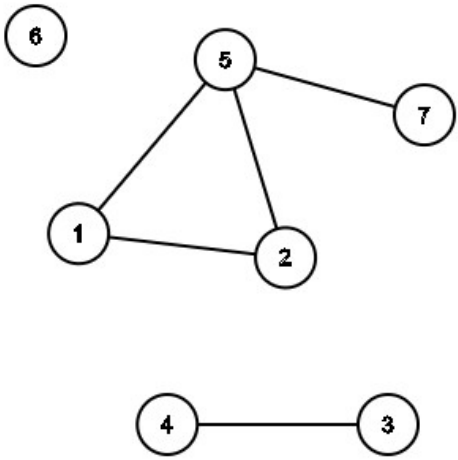
	1	2	3	4	5	6
1	0	1	0	0	1	0
2	1	0	1	0	0	0
3	0	1	0	1	0	0
4	0	0	1	0	0	0
5	1	0	0	0	0	1
6	0	0	0	0	1	0

5)

G=(V,E)

V={1,2,3,4,5,6,7}

E={(1,2),(1,5),(2,5),(3,4),(5,7)}



	1	2	3	4	5	6	7
1	0	1	0	0	0	0	0
2	1	0	0	0	0	0	0
3	0	0	0	1	0	0	0
4	0	0	1	0	0	0	0
5	0	0	0	0	0	0	1
6	0	0	0	0	0	0	0
7	0	0	0	0	1	0	0