MATEMÁTICA PARA A COMPUTAÇÃO

FABRÍCIO GALENDE MARQUES DE CARVALHO

Sumário

PREFÁCIO

Esse material tem como objetivo fornecer ao estudante de cursos de computação uma visão geral de alguns tópicos da matemática superior com enfoque na computação.

O prefixo **EP** é usado para identificar **Exercícios e Problemas** que têm como objetivo fazer com que o estudante retenha termos, conceitos e definições estudados na unidade, além de aplicá-los à resolução de problemas hipotéticos e teóricos "no papel". Exercícios com prefixo **PC** são exercícios de **Prática Computacional** e tem como foco a aplicação dos tópicos estudados à resolução de problemas através da utilização de ferramentas computacionais, tais como simuladores, linguagens de programação, etc. Ao final de cada unidade são apresentados exercícios resolvidos, prefixo **ER**, que exemplificam a utilização de conceitos e técnicas estudados na seção imediatamente anterior.

1. INTRODUÇÃO

1.1. MOTIVAÇÃO PARA O ESTUDO DA MATEMÁTICA EM CURSOS DE COMPUTAÇÃO

Originalmente, computadores foram concebidos para efetuar cálculos. As aplicações que foram inicialmente desenvolvidas eram voltadas às finalidades científicas ou bélicas.

Matemáticos, engenheiros, físicos e outros cientistas de áreas correlatas tiveram papel crucial na criação dos dispositivos computacionais e, também, na concepção e análise de toda uma teoria por trás da computação. É natural, portanto, que a computação tenha permeado tais áreas do conhecimento.

Apesar do fato de computadores terem se tornado dispositivos capazes de executar aplicações com múltiplos propósitos, desde aplicativos de entrega de comida até mesmo sistemas de guiamento e controle de satélites, em sua essência, computadores continuam a operar sobre números e todas as aplicações existentes são baseadas em princípios bem fundamentados da matemática. Dessa forma, é fundamental que um desenvolvedor de sistemas computacionais, seja ele de software ou hardware, compreenda os fundamentos matemáticos vinculados à computação e que seja capaz de aplicá-los à concepção, construção, avaliação e análise de tais sistemas.

1.2. DIFERENÇA ENTRE MATEMÁTICA ELEMENTAR E MATEMÁTICA DO ENSINO SUPERIOR

Tipicamente, a matemática ensinada ao longo do ensino fundamental e ensino médio tem um enfoque em aspectos básicos, tais como operações aritméticas, álgebra elementar e geometria. As aplicações estudadas por vezes são simplificações extremas da realidade e, com frequência, não refletem a natureza de vários fenômenos reais. Além disso, essa matemática elementar possui um enfoque pouco analítico e muito mais intuitivo e computacional (i.e., no sentido de execução mecânica de cálculos).

A matemática do ensino superior, apesar de ser construída sobre os pilares da matemática elementar, tem um enfoque muito mais analítico, por vezes extremamente formal, e procura fornecer ao estudante ferramentas que sejam capazes de resolver problemas reais, cujas simplificações não comprometem a eficácia da solução obtida para um dado problema.

Em geral, os aspectos formais ficam evidenciados quando **definições precisas** são fornecidas e resultados são demonstrados através de **provas matemáticas.**

Para o caso de aplicações, em geral são construídos e estudados **modelos matemáticos** que devem ser próximos o suficiente da realidade de modo que forneçam dados e informações úteis à solução de problemas. Para o caso de aplicações computacionais, há modelos que se relacionam a áreas da computação tais como banco de dados, aprendizagem de máquina, simulação de fenômenos, computação gráfica, economia e finanças, etc.

1.3. ALGUMAS ÁREAS DE ESTUDO DA MATEMÁTICA E SUAS APLICAÇÕES À COMPUTAÇÃO

Apesar da matemática ser uma única ciência, é comum que esta seja dividida em diferentes disciplinas (ou áreas) para facilitar a definição do escopo de estudo e, também, auxiliar o estudante no processo de aprendizado e, consequentemente, na adequada seleção do ferramental necessário à resolução de um dado problema.

A seguir, são enumeradas algumas das áreas frequentemente estudadas pela matemática superior, com um enfoque em computação:

- **ARITMÉTICA:** Estuda as operações fundamentais existentes em um sistema numérico. As regras de precedência, agrupamento e de obtenção de resultados de operações numéricas estão entre as aplicações mais comuns na área de computação.
- **ÁLGEBRA ELEMENTAR:** Estuda as propriedades e operações envolvendo símbolos que representam objetos "arbitrários" denominados de variáveis. Equações matemáticas, polinômios, etc. , estão dentro do objeto de estudo da álgebra elementar. Expressões algébricas com frequência são utilizadas em modelos computacionais de sistemas.
- **ALGEBRA ABSTRATA:** Trata do estudo de estruturas matemáticas tais como anéis, grupos, corpos, etc. Estas estruturas são abstratas no sentido de não necessitarem de uma correspondência direta ou intuitiva com algo real, tangível. Geralmente o estudo da álgebra abstrata é conduzido nos cursos de bacharelado em matemática ou cursos de pós-graduação em ciências exatas tais como computação. Trata-se de um ramo da matemática formal e não computacional (i.e., não mecânica), mas cujos resultados abstratos e gerais são usados para fundamentar resultados específicos e computacionais (ou seja, uma demonstração abstrata e geral pode ser utilizada para se chegar a um resultado tangível particular, dado que as condições utilizadas na demonstração abstrata sejam satisfeitas pelo ente real.
- **ÁLGEBRA LINEAR:** Pode ser considerada como uma área específica da álgebra que estuda expressões, equações e sistemas de equações algébricas que possuem uma

forma específica com relação às variáveis (e.g: uma equação do tipo *y = 10x* é uma equação linear). A álgebra linear é uma das áreas da matemática que possui vasta aplicação em computação, podendo ser destacadas as áreas de aprendizagem de máquina, computação gráfica, entre outras.

- **ALGORITMOS:** Constituem a área da matemática que se preocupa com os métodos sistemáticos de computação para a obtenção de soluções aos problemas. Incluem a análise de existência de soluções expressas através de algoritmos (computabilidade), análise de desempenho, estruturas de dados, etc.
- **LÓGICA MATEMÁTICA:** É a área da matemática que está voltada à sistematização do processo de raciocínio na obtenção de conclusões. A lógica matemática lida com o processo de inferência, ou seja, a conclusão a partir de um certo conjunto de informações ou fatos (denominados de premissas). Os sistemas computacionais especialistas (e.g.: sistemas de diagnóstico) são fundamentados na lógica matemática.
- **MATEMÁTICA DISCRETA**: A noção de discreto tem a ver com separação clara e com contagem. A matemática discreta lida com estruturas/objetos que podem ser facilmente separados uns dos outros e facilmente contados. Um exemplo de objetos que podem ser facilmente contados são número de celulares armazenados em uma agenda, já a quantidade de água em uma garrafa, fica difícil de ser separada e contabilizada (e.g: 1L, 1.1L, 1.1000001L, etc.). Esse ramo da matemática é um dos pilares da computação, dado que computadores digitais modernos operam utilizando dados que são iminentemente discretos. Diferentemente de áreas como o cálculo e a análise matemática, a matemática discreta faz uso frequente de métodos numéricos (aproximados) para resolver problemas do mundo real que envolvem objetos contínuos.
- **TEORIA DE CONJUNTOS:** Estuda o agrupamento de itens (que podem ou não ter alguma relação entre si) e, também, a associação entre esse agrupamento a outros (operações entre conjuntos). A teoria de conjuntos e uma das bases dos sistemas de bancos de dados.
- **RELAÇÕES E FUNÇÕES:** É a área da matemática que está voltada a associação entre elementos que pertencem a dois ou mais conjuntos. Relações e funções são utilizados como base da modelagem e construção de sistemas computacionais (e.g.: definição de funções em uma linguagem de programação). Em geral, ao se modelar um fenômeno do mundo real, busca-se a construção de um modelo que reflita uma dependência funcional entre duas ou mais variáveis.
- **GEOMETRIA:** É a área da matemática que estuda elementos tais como formas, tamanhos e posição. Considera aspectos espaciais. Uma das especializações da geometria é a trigonometria, que estuda relações entre ângulos e dimensões considerando como base a figura geométrica de um triângulo. A geometria é uma das áreas que está presente em praticamente todas as aplicações computacionais modernas, pois as interfaces gráficas são construídas levando-se em conta aspetos geométricos.
- **CÁLCULO E ANÁLISE MATEMÁTICA:** O cálculo e a análise matemática estudam as funções sob o ponto de vista de variação, considerando elementos infinitamente pequenos. Enquanto o cálculo possui uma natureza mais prática e operacional, a análise efetua esse estudo de maneira formal. Essas áreas encontram aplicabilidade direta na simulação computacional incluindo o projeto e análise de sistemas aeroespaciais, na resolução de problemas de otimização, entre outros. Os modelos matemáticos que dependem de aspectos do cálculo tais como derivadas e integrais, são ditos modelos de sistemas dinâmicos.

- **PROBABILIDADE:** Estuda o quão factível um evento é ou não. Está atrelada ao estudo não só da aleatoriedade, mas também da regularidade presente em tais eventos. A teoria da probabilidade encontra vasta aplicação na área de simulação computacional, desenvolvimento de jogos, processamento de linguagem natural, entre outras.
- **ESTATÍSTICA:** A estatística lida com o agrupamento, organização, exibição, análise e interpretação de dados. A estatística é amplamente utilizada no processo de tomada de decisão automatizada por parte de sistemas computacionais "inteligentes". Além disso, a estatística é um dos principais pilares da análise de dados em sistemas computacionais.

EXERCÍCIO RESOLVIDO

ER.1.1. Identifique áreas da matemática que são aplicáveis à resolução do seguinte problema:

"O usuário de um sistema computacional abre uma imagem contendo sua foto e percebe que ao tirar sua selfie seu corpo ficou ligeiramente inclinado. Aplicando uma função de rotação do software editor de imagens, aplica uma rotação no sentido anti-horário, no valor de 15º e percebe que a imagem agora está corrigida."

Resolução: Trata-se de um problema envolvendo **relações espaciais** entre os elementos de uma imagem. Esse problema é abordado na geometria e, também, na **álgebra linear** para o caso de rotações simples em imagens.

EXERCÍCIOS

EP. 1.1. Cite uma referência (livro, artigo de internet, vídeo, etc.), para cada uma das áreas de estudo da matemática citadas a seguir:

a) Aritmética

b) Álgebra elementar

c) Álgebra linear

d) Algoritmos

f) Lógica matemática

g) Teoria dos conjuntos

h) Relações e funções

i) Geometria

j) Cálculo e análise matemática

k) Probabilidade

l) Estatística

EP.1.2. Ilustre uma fórmula matemática/função, ou equação, ou diagrama, ou símbolo, representativo das seguintes áreas da matemática:

a) Álgebra linear

b) Teoria de conjuntos

c) Probabilidade

d) Cálculo / análise matemática

e) Geometria (trigonometria)

f) Lógica matemática

EP. 1.3. Identifique uma área de conhecimento da matemática que pode ser utilizada para a resolução dos seguintes problemas (obs: caso envolva múltiplas áreas, cite-as e diga qual é a que aparenta ser predominantemente utilizada. Em alguns, casos mais de uma combinação de áreas é possível).

a) Determinação de quais palavras são mais utilizadas em uma língua, tal como o português ou inglês.

b) Desenvolvimento de um sistema de recomendação de produtos considerando o histórico dos produtos que são frequentemente comprados em um site de vendas. Esse sistema requer a organização, a análise e a adequada exibição desses dados.

c) Cálculo de trajetórias de lançamentos oblíquos de objetos sujeitos a atração gravitacional. Nesse caso, os objetos estão sujeitos a aspectos dinâmicos (velocidades, acelerações, etc.).

d) Determinação de cantos e bordas em imagens digitais, considerando a variação que ocorre na intensidade dos pixels da imagem. Nesse caso, considera-se que os pixels são estruturas discretas (ou seja, pertencentes a uma grade NxM) e que os valores das intensidades também são discretos.

e) Definição de um conjunto de passos computacionais a serem executados de modo a processar um conjunto de entradas e obter uma saída específica. Análise desse conjunto de passos quanto a sua correção, consumo de memória, tempo, etc.

f) Desenvolvimento de um jogo envolvendo objetos que se movem de modo “aleatório” em uma interface gráfica e que devem ser destruídos por um canhão direcionado através dos comandos de teclado fornecidos pelo usuário.

g) Um assistente pessoal digital, acionado através de comandos de voz. Os comandos de voz são processados de acordo com a análise e processamento de um histórico de dados utilizado para o treinamento do modelo (quanto mais dados forem utilizados no treinamento, mais acurado será o assistente pessoal digital).

EP.1.4. Uma prova matemática é um raciocínio dedutivo que faz uso de informações disponíveis para chegar a uma conclusão. Na matemática, em áreas como aritmética e álgebra, são comuns provas que fazem uso de propriedades simples tais como comutatividade, distributividade, utilização de elemento neutro na soma ou multiplicação (i.e., somar ou multiplicar ambos os lados de uma equação por um mesmo valor). Considere a seguinte prova matemática:

Provar que (-1) x (-1) = + 1

Prova:

1 -1 = 0	*Um número somado ao seu oposto é igual a zero.*
(-1) x (1-1) = (-1) x 0	*Multiplicar ambos os lados de uma equação por um número diferente de zero não altera a equação.*
(-1)x1 + (-1)x(-1) = 0	*Distributividade.*
-1 + (-1)x(-1) + 1= +1	*Somar um mesmo valor a ambos os lados de uma equação não altera a equação.*
-1 + (-1)x(-1) + 1= +1	*Um número somado ao seu oposto é igual a zero.*
(-1) x (-1) = +1	*Cqd (como queríamos demonstrar).*

Considerando o resultado anterior, escreva uma prova para a seguinte proposição:

Proposição:

$$(-10) * (-3) = + 30$$

EP.5. Explique o que é uma prova direta. Dê um exemplo básico envolvendo, por exemplo, teoria de conjuntos, ou álgebra elementar, ou aritmética básica.

EP.6. Explique o que é uma prova por contradição (também conhecida como prova por absurdo). Dê um exemplo envolvendo teoria de conjuntos, ou álgebra elementar, ou geometria ou aritmética básica.

EP.7. Explique o que é uma prova utilizando contrapositiva (ou contraposição). Dê um exemplo envolvendo teoria básica de conjuntos, ou álgebra elementar, ou aritmética, ou geometria.

EP.8. Provas matemáticas também são utilizadas na prova de correção de um algoritmo. Diga o que é um invariante e como ele é utilizado na prova que tem como objetivo indicar se um algoritmo está correto ou não. Dê um exemplo, utilizando um algoritmo, descrito em TypeScript, que retorna o maior elemento contido em um array não ordenado.

2. DEFINIÇÕES MATEMÁTICAS

Definições matemáticas são maneiras de se esclarecer o significado de objetos e símbolos utilizados em matemática. Evitam que equívocos sejam cometidos ao se ler ou utilizar um determinado objeto ou símbolo.

Em computação, definições são amplamente utilizadas para criação de **variáveis** que serão vinculadas a valores, criação de modelos representativos de objetos do mundo real (e.g.: classes), criação de símbolos fáceis de serem memorizados para representar conjuntos de valores menos intuitivos numericamente (e.g.: MES.JANEIRO correspondente ao valor 1), etc.

Uma definição é composta por um definiendum e por um definiens. Matematicamente é comum que seja representada tal como:

$$definiendum \equiv definiens$$

$$definiendum \triangleq definiens$$

$$definiendum \stackrel{\text{def}}{=} definiens$$

Definições podem ser denotativas ou conotativas.

2.1. DEFINIÇÕES DENOTATIVAS

São definições que especificam o conjunto (agrupamento) de objetos que correspondem ao *definiendum*.

Alguns tipos de definições denotativas são através de enumeração, definição ostensiva e definição recursiva.

2.1.1. ENUMERAÇÃO

É feita através da exemplificação dos objetos correspondentes ao *definiens*.

Exemplo:

Número par $\stackrel{\text{def}}{=}$ 2, 4, 6, 8, 10, ...

Em computação, definições desse tipo são comuns em tipos de dados denominados de **enumerações.** Enumerações são utilizadas para tornar programas de computadores mais legíveis, evitando equívocos e facilitando o uso por parte do programador.

Em algoritmos e programas, enumerações são tipicamente definidas da seguinte forma:

```
Enumeração Nome_da_enumeração:
      símbolo 1 = valor para o símbolo 1
      símbolo 2 = valor para o símbolo 2
      ...
      símbolo N = valor para o símbolo N
Fim-enumeração
```

2.1.2. DEFINIÇÃO OSTENSIVA

Também conhecida como definição “apontando para o objeto”. Tipicamente é utilizada quando o *definiens* é difícil ou muito complicado de ser esclarecido através de palavras, símbolos, etc.

Exemplo:

Uma caneta esferográfica $\overset{\text{def}}{=}$ 

Nesse caso, imagine que aquele que efetua a definição observa e aponta para uma caneta enquanto que outra pessoa olha para a caneta e associa isso ao nome “caneta esferográfica).

2.1.3. DEFINIÇÃO RECURSIVA

São definições que se baseiam em uma definição simples inicial, denominado de caso base, efetuam uma generalização, denominada de cláusula indutiva, e, também, possuem uma cláusula que garante que objetos definidos de modo recursivos segundo as cláusulas anteriores sempre satisfarão a definição (cláusula externa ou de fechamento).

Exemplo:

Menor inteiro positivo $\overset{\text{def}}{=}$ 1 (caso base).

Se N é um número inteiro positivo, então N+1 também é (cláusula indutiva).

Número inteiro positivo $\overset{\text{def}}{=}$ Número que satisfaz as duas condições anteriores.

Definições recursivas são muito utilizadas em computação na criação de funções e estruturas de dados. Ao se definir uma estrutura ou função de modo recursivo, sempre é necessária a criação do caso base que, nesse caso, é o responsável por evitar que o algoritmo nunca termine. A ausência de término, nesse caso, acontece justamente por causa da cláusula indutiva.

Em **expressões matemáticas**, mais especificamente **fórmulas**, definições recursivas costumam utilizar subscritos para relacionar **variáveis** (i.e.: símbolos que representam números não explicitamente especificados) cujos valores dependem dessas mesmas variáveis mas em situações ou instantes distintos. Isso corresponde a cláusula indutiva ou relação de recorrência.

Exemplo:

Primeiro termo da sequência numérica $\overset{\text{def}}{=}$1

Termos da sequência numérica $\overset{\text{def}}{=} x_k$, com x_k expresso pela fórmula $x_k = 2.\ x_{k-1}$

Essa definição é equivalente a

Sequência numérica $\overset{\text{def}}{=}$ {1, 2, 4, 8, ...}

Que também pode ser expressa segundo a seguinte notação matemática para conjuntos:

Sequência numérica = $\{x_k \mid x_1 = 1 \text{ e } x_k = 2x_{k-1} \text{ para } k \geq 2\}$

2.2. DEFINIÇÕES CONOTATIVAS

São definições que especificam critérios que devem ser satisfeitos ou o intuito de determinado objeto para que possa corresponder a um dado *definiendum.*

2.2.1. DEFINIÇÃO POR SINÔNIMO

São definições que proveem um termo ou símbolo com um mesmo significado daquilo que está sendo definido.

Exemplo:

Enorme $\stackrel{\text{def}}{=}$ grande, extenso

2.2.2. DEFINIÇÃO OPERACIONAL

Utiliza atributos que podem ser observados e que estão atrelados a como o *definiendum* é utilizado em alguma operação ou procedimento. Tipicamente especifica algo que é empiricamente testável.

Exemplo:

Símbolo de adição $\stackrel{\text{def}}{=}$ é o símbolo utilizado em expressões de soma de dois números, tais como em $8 + 10$.

Ferro = metal que produz óxido/ferrugem ao ser deixado em contato com a água e com oxigênio.

2.2.3. DEFINIÇÃO POR GÊNERO E DIFERENÇA

Efetuada identificando-se uma classe ou categoria a qual o objeto pertence e depois refinando-se para suas características mais específicas dentro dessa categoria.

Exemplo:

Ser humano $\stackrel{\text{def}}{=}$ animal mamífero e racional

Na definição anterior, animal é a categoria mais geral enquanto que mamífero e racional são refinamentos ou especializações.

Em computação definições por gênero e diferenças são comuns em linguagens de programação com suporte a orientação a objetos quando se efetuam definições através de **herança**.

EXERCÍCIO RESOLVIDO

ER.2.1. Defina, utilizando enumeração, os números naturais ímpares. Nesse caso é possível enumerar todos esses números?

Resolução:

Números ímpares $\stackrel{\text{def}}{=}$ 1, 3, 5, 7, ...

Note-se que se trata de um conjunto enumerável mas cuja quantidade de elementos é infinita. Portanto, não é possível enumerar todos os elementos do conjunto, sendo utilizado nesse caso o símbolo ... (reticências, que significam que a sequência numérica continua indefinidamente seguindo o padrão já ilustrado, ou seja, cada elemento obtido somando-se 2 ao anterior.)

ER.2.2. Defina, utilizando recursão, os membros de uma família de pessoas que começou com um homem e uma mulher denominados de Adão e Eva.

Resolução:

Primeiros membros da família $\stackrel{\text{def}}{=}$ Adão e Eva (1)

Se uma pessoa é da família de Adão e Eva (geração k), então os filhos dessa pessoa (geração k+1) também são da família de Adão e Eva. (2)

Para ser da família de Adão e Eva, então as condições (1) e (2) devem ser satisfeitas simultaneamente.

EXERCÍCIOS

EP.2.1. Efetue uma definição, através de enumeração, de todos os membros da sua família, considerando apenas parentesco direto de pais, irmãos e filhos.

EP.2.2. Efetue uma definição, através de enumeração, de cursos de graduação existentes na instituição de ensino superior em que você está estudando.

EP.2.3. Efetue uma definição, através de enumeração, de números primos inferiores a 40. Um número é dito primo se só possui como divisor ele mesmo e o número um.

EP.2.4. Efetue uma definição ostensiva, de um colega de classe.

EP.2.5. Efetue uma definição recursiva de uma sequência numérica cujos elementos são obtidos multiplicando-se, a partir do segundo elemento, o elemento anterior por 3. Considere que o primeiro elemento vale 2.

PC.2.1. Utilizando linguagem de programação defina:

a) Uma enumeração para os meses do ano.

b) Uma enumeração para os dias da semana.

c) Uma função recursiva para o cálculo do fatorial de um número.

d) Uma definição que corresponda a definição do tipo gênero-diferença para um uma pessoa que estude em uma faculdade. Utilize uma linguagem que dê suporte a herança. Suponha que toda pessoa possua nome e idade e que um estudante de uma faculdade possui registro acadêmico e ano e semestre de ingresso.

PC.2.2. Um veículo possui a capacidade de se mover, expressa pela alteração na sua coordenada de longitude e latitude. Um veículo elétrico é um veículo que possui como fonte de energia primária a eletricidade (armazenada em uma bateria, que tem sua capacidade especificada em

Ampere-hora). Um veículo elétrico e voador é um veículo que também possui a capacidade de se mover na vertical, expressa pela alteração de sua altitude em relação ao solo. Represente um veículo elétrico e voador utilizando uma cadeia de herança. Defina o código-fonte representativo do modelo em um arquivo separado daquele que faz uso desse e, adicionalmente exemplifique o acesso e a modificação desses atributos através de chamada de suas operações (moverHorizontal, moverVertical).

PC.2.3. O que acontece ao se executar uma chamada a uma função recursiva que chama a si mesma um elevado número de vezes? Dê um exemplo utilizando o código-fonte da progressão aritmética fornecido pelo professor. Faça um comparativo escrevendo um algoritmo e código que sejam equivalentes ao recursivo em termos de entradas e saídas mas que utilizem iteração ao invés de recursão. Qual sua conclusão?

PC.2.4. Uma progressão geométrica é uma sequência numérica onde cada elemento, a partir do segundo, é obtido multiplicando-se o anterior por uma constante. Utilizando uma linguagem de programação que dê suporte a orientação a objetos, defina uma progressão geométrica e dê exemplo de geração de seus primeiros 50 termos.

PC.2.5. A sequência de Fibonacci é definida da seguinte forma: {1, 1, 2, 3, 5, 8, 13, ...}, ou seja, para $k>2$, $x_k = x_{k-1} + x_{k-2}$. Utilizando uma linguagem de programação com suporte a orientação a objetos, defina uma classe que modele a sequência de Fibonacci e exemplifique o cálculo de alguns de seus termos. Ilustre a chamada recursiva e identifique chamadas repetidas a um mesmo valor construindo uma árvore de chamadas.

PC.2.6. Modele, utilizando orientação a objetos, uma classe que representa uma sequência numérica onde os termos, a partir do 4º são obtidos somando-se os três termos anteriores. Considere que os três termos iniciais da sequência são parâmetros do construtor do objeto representativo da sequência. Utilize recursão e demonstre a exibição dos 15 primeiros termos da sequência, para diferentes valores de a_1, a_2 e a_3.

3. TEORIA DE CONJUNTOS

3.1. IDEIA GERAL DE CONJUNTO

Em geral conjunto remete a uma coleção ou classe de elementos. Nessa coleção, não há restrição referente ao tipo de elementos (e.g.: letras, números, palavras, etc.).

Em geral, na matemática, conjuntos são representados por letras maiúsculas e seus elementos por letras minúsculas.

Em um conjunto, não são consideradas nem a ordem nem a repetição de seus elementos.

Exemplos:

Conjunto A = {a, b} = {b, a}

Conjunto B = { x, y, z}

Conjunto C ={"carro", "moto", 10}

Quando um elemento *e* faz parte de um conjunto *E*, diz-se que o elemento pertence ao conjunto e representa-se da seguinte forma:

$$e \in E.$$

Lê-se "o elemento e pertence ao conjunto E", caso contrário diz-se que o elemento não pertence ao conjunto e representa-se por

$$e \notin E.$$

3.2. NOTAÇÃO E DESCRIÇÃO

Além da representação usual por letras maiúsculas, conjuntos também podem ser representados de modo mais detalhado através da enumeração e seus elementos (***roster notation***) ou através da descrição de uma propriedade característica desses mesmos elementos (***set builder notation***).

Exemplos:

Roster notation:

A ={ 2, 4, 6, 8, 10}

B = {a, b, c, d, e, f, g, ..., z}

Set builder notation:

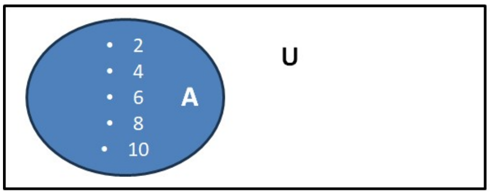
C = { x | x é letra do alfabeto português}

D = { y | (y = 2x -1) e (x > 1) e (x é número natural) }

D = { y | (y = 2x -1) $\wedge$ (x > 1) $\wedge$ (x é número natural) }

Observação: Na notação *set builder*, os conectivos *e* e *ou* são representados por $\wedge$ e $\vee$, respectivamente. Um $\wedge$ significa que ambas as condições (ou restrições) devem ser satisfeitas, já um $\vee$ significa que pelo menos uma das condições deve ser satisfeita, não requerendo que todas sejam satisfeitas simultaneamente.

Em termos gráficos, é comum que conjuntos sejam representados utilizando-se o diagrama de Veen.



Ao se utilizar a notação de Venn, é comum, também que o conjunto que contém todos os valores possíveis para um determinado conjunto também seja representado. Nesse caso, trata-se do conjunto universo (*U*), representado por um retângulo que contém internamente uma elipse representativa do conjunto específico. No caso do exemplo anterior, o conjunto *A*.

Alguns conjuntos numéricos possuem letras/símbolos especiais, são eles:

- ✓ Conjunto dos números **naturais**: $\mathbb{N}$
- ✓ Conjunto dos números **inteiros**: $\mathbb{Z}$
- ✓ Conjunto dos números **racionais**: $\mathbb{Q}$
- ✓ Conjunto dos números **irracionais**: ($\mathbb{R}$ - $\mathbb{Q}$)
- ✓ Conjunto dos números **reais**: $\mathbb{R}$
- ✓ Conjunto dos números **complexos**: $\mathbb{C}$

3.3. CASOS ESPECIAIS DE CONJUNTOS NUMÉRICOS

3.3.1. CONJUNTO DOS NÚMEROS NATURAIS

É formado pelos números utilizados em contagens simples.

$$\mathbb{N}=\{1, 2, 3, ...\}$$

Em algumas referências é comum que seja incluído o 0 como parte desse conjunto.

Possui propriedade de fechamento (*closure*) para soma e multiplicação, ou seja, a soma e a multiplicação de dois números naturais sempre resultam em um número natural.

!

- ✓ Em linguagens de programação, números naturais são frequentemente utilizados para indexar agrupamentos de objetos, tais como listas, vetores (arrays) e matrizes (arrays multidimensionais).
- ✓ Quando o índice utilizado para indexar coleções resulta de uma operação distinta de soma ou multiplicação envolvendo dois números naturais, não é garantido que o índice resultante será um número natural, portanto, pode ocorrer erro de indexação. Nesse caso, algumas linguagens de programação retornam valores indefinidos ou então há o lançamento de uma exceção.

3.3.2. CONJUNTO DOS NÚMEROS INTEIROS

Trata-se de uma extensão dos números naturais que inclui números negativos e o zero.

$\mathbb{Z} = \{..., -3, -2, -1, 0, 1, 2, 3, ... \}$

Além da adição e multiplicação, possui também fechamento para a subtração.

Em vários algoritmos, os números inteiros são utilizados para representar a indexação em sentido oposto ao usual em estruturas como vetores. Nesse caso, há a omissão do detalhe de indexação de modo que o usuário "percebe" um número negativo como um acesso no sentido do fim para o início.

Exemplo:

Considerando um vetor `x = [1, 2, 3, 4, 5]`, o elemento `x.get(-1)` pode significar o primeiro elemento começando-se do fim para o início. Ou seja, na verdade a indexação é feita da seguinte forma:

```
Classe array{
      valores: vetor [0..N-1] de números

      Função get(indice): número
            Se indice >=0 então
                  retorne valores[indice]
            Senão
                  indice ← N-indice
                  retorne valores[indice]
            Fim-se
      Fim-função
```

3.3.3. CONJUNTO DOS NÚMEROS RACIONAIS

Trata-se do conjunto formado pelos números inteiros e pelos números fracionários que podem ser escritos sob a forma p/q.

$$\mathbb{Q} = \{ p/q \mid p,q \in \mathbb{Z} \wedge q \neq 0 \}$$

Possui fechamento para a soma, subtração, multiplicação e divisão.

Inclui os números fracionários com as representações finitas e infinitas periódicas.

Os números pertencentes ao conjunto dos números racionais são amplamente utilizados nas mais diversas áreas de aplicação, incluindo finanças, engenharia, computação gráfica, etc.

Como os computadores são capazes somente de representar números utilizando um número finito de casas decimais, pode-se afirmar que os números racionais são utilizados em praticamente todas as aplicações computacionais. Quando um número não é racional, é necessário convertê-lo, por aproximação, para um número racional representável.

3.3.4. CONJUNTO DOS NÚMEROS IRRACIONAIS

É formado pelos números que não podem ser postos na forma de um número racional.

Esses números podem resultar da razão de dois números não comensuráveis (i.e., que não podem ser postos em uma referência comum de comparação), raízes de números primos, etc.

$$\mathbb{R} - \mathbb{Q} = \{ \pi, -\pi, \sqrt{2}, \sqrt{3}, \ldots\}$$

!

- ✓ Em várias linguagens de programação, há bibliotecas numéricas que possuem constantes contendo aproximações para os números racionais com um número razoável de casas decimais. Sempre que possível, é recomendável utilizar essas constantes aproximadas no desenvolvimento de aplicações.
- ✓ Caso a linguagem de programação não possua tal recurso, em sistemas de alto desempenho, é recomendável que o desenvolvedor efetue o cálculo prévio, armazene em arquivo e carregue os valores aproximados nas aplicações, objetivando assim evitar o *overhead* de cálculo e aproximação.

3.3.5. CONJUNTO DOS NÚMEROS REAIS

É formado pela intersecção entre o conjunto dos números racionais e irracionais.

Esse conjunto é representado pela letra $\mathbb{R}$.

Em computação, os números reais possuem sempre representação fracionária finita e são conhecidos como números de ponto flutuante (*float point number*).

3.3.6. CONJUNTO DOS NÚMEROS COMPLEXOS

Formado por números com o formato *(a+bi)*, sendo que a e b são números reais, chamados de parte real e imaginária, respectivamente.

$$\mathbb{C} = \{a + bi \mid (a,b \in \mathbb{R}) \wedge (i = \sqrt{-1}\ )\}$$

Números complexos são utilizados em sistemas computacionais de engenharia, física, etc.

Um número real pode ser visualizado em um sistema de coordenadas cartesianas onde o número *b* é representado no eixo das ordenadas *(y)* e o número a no eixo das abscissas *(x).*

Exemplo:

Número imaginário *n = 3 + 8i*

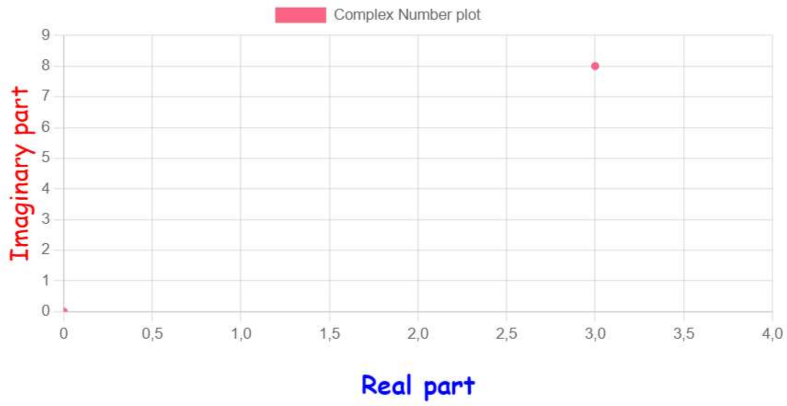


Figura 1. Representação de um número imaginário no plano cartesiano.

Um outro exemplo de aplicação dos números complexos é a geração de imagens com padrões que se repetem e que são interessantes. Algumas dessas imagens são denominadas de fractais.

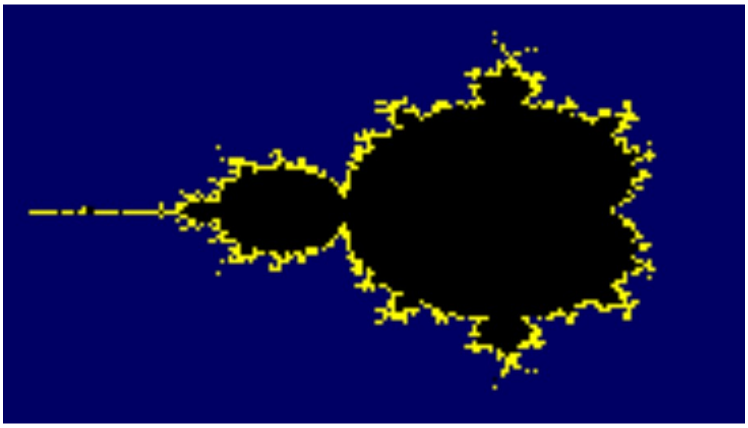


Figura 2. Conjunto de Mandelbrot.

3.4. SÍMBOLOS ESPECIAIS PARA ALGUNS CONJUNTOS

Seja X um conjunto numérico arbitrário, então:

X^* é o mesmo conjunto excluindo-se o 0.

X_+ é o mesmo conjunto excluindo-se os elementos negativos.

$X\_$ é o mesmo conjunto excluindo-se os elementos positivos.

Um conjunto que também tem uma representação especial é o conjunto sem elementos, denominado de conjunto vazio:

$$\{\ \} \text{ ou } \varnothing$$

3.5. CONJUNTOS FINITOS E CARDINALIDADE

É um conjunto que contém um número finito de elementos. Nesse caso, o número de elementos do conjunto é denominado de cardinalidade do conjunto, que, para um conjunto *E* é representado por:

$$| E |$$

3.6. RELAÇÕES E OPERAÇÕES ENTRE CONJUNTOS

3.6.1. IGUALDADE ENTRE CONJUNTOS

Dois conjuntos A e B são ditos iguais se e somente se possuem os mesmos elementos.

$$A = B \leftrightarrow x \; (x \in A \wedge \; x \in B)$$

Se existir ao menos um elemento que não faça parte de ambos os conjuntos, então esses conjuntos são diferentes ($A \neq B$)

3.6.2. INCLUSÃO/SUBCONJUNTOS

Um conjunto *A* é subconjunto (***subset***) de B quando todo elemento de *A* é também elemento de B.

$$A \subseteq B \leftrightarrow \forall x \; (x \in A \rightarrow x \in B)$$

Quando A é diferente de B, diz-se que A é um subconjunto próprio de B (***proper subset***).

$$A \subset B \leftrightarrow \forall x \; (x \in A \rightarrow x \in B) \wedge (A \neq B)$$

3.6.3. CONJUNTO DAS PARTES DE UM CONJUNTO

Também conhecido como Power Set, é o conjunto que é construído a partir de todos os subconjuntos de um conjunto *A*:

$$P(A) = \{X \mid X \subseteq A\}$$

Exemplo: Determinar o *power set* do conjunto A = {1, 2}

$$P(A) = \{ \varnothing, \{1\}, \{2\}, \{1,2\}\}$$

3.6.4. UNIÃO ENTRE CONJUNTOS

Considerando dois conjuntos *A* e *B*, a união é formada construindo-se o conjunto contendo elementos que pertencem a *A*, *B* ou ambos.

$$x \in (A \cup B) \leftrightarrow (x \in A \vee x \in B)$$

> **!**
>
> ✓ Em sistemas gerenciadores de bancos de dados (SGBDs) é comum a existência de operações de união entre conjuntos de resultados a consultas ao banco de dados (***union***).

Exemplo:

usuario 1

select * from usuario

	nome	cidade
1	fabricio	sjc
2	ronaldo	bel
3	ana	sp
4	pedro	palmas

```
select * from usuario where nome = 'fabricio'  union  select  *
from usuario where cidade = 'sp';
```

Results 1

select * from usuario where nome = 'fabricio'

	nome	cidade
1	ana	sp
2	fabricio	sjc

3.6.5. INTERSECÇÃO ENTRE CONJUNTOS

Considerando dois conjuntos A e B, a intersecção é formada construindo-se o conjunto dos elementos que pertencem a A **e** pertencem a B.

$$x \in (A \cap B) \leftrightarrow (x \in A \wedge x \in B)$$

> **!**
>
> ✓ Em sistemas gerenciadores de bancos de dados (SGBDs) é comum a existência de operações de intersecção entre atributos comuns presentes em tabelas distintas (***joins***).

Exemplo:

usuario 1

select * from usuario

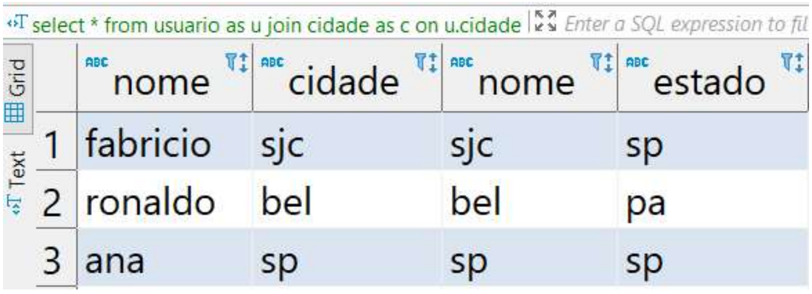
	nome	cidade
1	fabricio	sjc
2	ronaldo	bel
3	ana	sp
4	pedro	palmas

cidade 1

select * from cidade

	nome	estado
1	sjc	sp
2	bel	pa
3	sp	sp
4	manaus	am

```
select * from usuario as u join cidade as c on u.cidade =
c.nome;
```


select * from usuario as u join cidade as c on u.cidade

	nome	cidade	nome	estado
1	fabricio	sjc	sjc	sp
2	ronaldo	bel	bel	pa
3	ana	sp	sp	sp

3.6.6. DIFERENÇA ENTRE CONJUNTOS

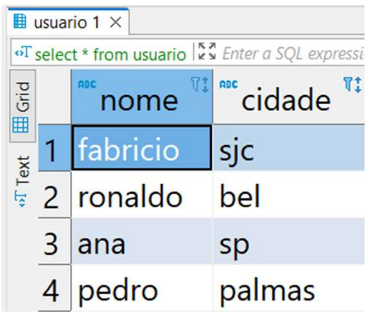
A diferença entre dois conjuntos *A* e *B* é definida como sendo os elementos que pertencem a *A* e não pertencem a *B*.

$$x \in (A - B) \leftrightarrow (x \in A \wedge x \notin B)$$

!

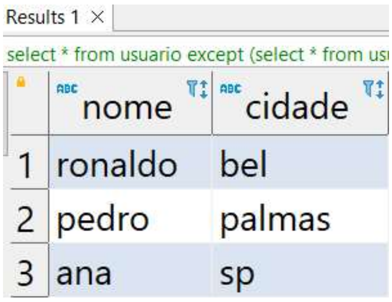
- ✓ Em sistemas gerenciadores de bancos de dados (SGBDs) é comum a existência de operações de diferença entre conjuntos que operam sobre o conjunto de resultados das consultas (*except*).

Exemplo:


usuario 1
select * from usuario

	nome	cidade
1	fabricio	sjc
2	ronaldo	bel
3	ana	sp
4	pedro	palmas

```
select * from usuario except (select * from  usuario where
nome = 'fabricio');
```


Results 1

select * from usuario except (select * from usı

	nome	cidade
1	ronaldo	bel
2	pedro	palmas
3	ana	sp

3.6.7. COMPLEMENTAR DE UM CONJUNTO EM RELAÇÃO A OUTRO

Considerando um subconjunto *A* de *B*, o complementar de *A* em relação a *B* é definido como sendo *(B – A)*.

3.6.8. PRODUTO ENTRE DOIS CONJUNTOS

O produto entre dois conjuntos *A* e *B*, conhecido como produto cartesiano e representado por $A \times B$, é formado por todos os pares (*a*, *b*), com $a \in A$ e $b \in B$.

$$A \times B = \{ (a,b) \mid a \in A \wedge b \in B\}.$$

Os pares *(a,b)* são chamados de pares ordenados ou duplas.

Para o caso do produto envolvendo N conjuntos tem-se as chamadas *n*-uplas (“ênuplas”, por vezes também chamadas de “tuplas”).

EXERCÍCIO RESOLVIDO

E.R.3.1. Exercício Resolvido: Para os conjuntos seguintes, determine: a)$A \cap B$; b) $A \cup B$; c) A-B; d) $P(A)$

$$A = \{ 1, 2, 3, 10\}$$

$$B = \{20, 1, 2, 3, -5\}$$

Resolução:

a)

A = { 1, 2, 3, 10}
B = {20, 1, 2, 3, -5}

$A \cap B$ ={ 1, 2, 3}

Marcar os elementos que se repetem em ambos os conjuntos auxilia na determinação do resultado das operações.

A intersecção é formada pelos elementos que aparecem em ambos os conjuntos.

b)

$A \cup B$ = {1, 2, 3, 10, 20, -5}

A união é formada pelos elementos que aparecem em um ou ambos os conjuntos.

c)

A-B = {10}

A diferença A-B é formada pelos elementos que pertencem a A mas não pertencem a B.

d)

P(A) = {∅, {1}, {2}, {3}, {10}, {1,2}, {1,3}, {1,10}, {2,3}, {2,10},{3,10}, {1,2,3}, {1,2,10}, {1,3,10} {2,3,10}, {1,2,3,10}}

O conjunto das partes de A deve incluir o conjunto vazio e todos os subconjuntos contendo 1, 2, 3 e 4 elementos (próprio A), formados a partir dos elementos de A.

EXERCÍCIOS

EP.3.1. Para os conjuntos A, B e C seguintes, determine o resultado das operações indicadas.

A = { 1, 2, 3, "moto"}

B= { "carro", "moto", 2}

a) A ∪ B

b) A ∩ B

c) A – B

d) B – A

e) P(B)

EP.3.2. Para os problemas seguintes, indique quais tipos numéricos são os mais adequados na hora de se implementar uma solução computacional (indique a qual dos conjuntos numéricos notáveis eles devem pertencer). Justifique sua resposta e forneça um exemplo para cada item.

a) Contagem do número de linhas de um arquivo de log.

b) Definição do preço, incluindo centavos, de mercadorias de uma loja.

c) Processamento de sinais de voz ou de comunicação em um sistema de engenharia de telecomunicações.

d) Marcação de temperaturas, em graus celsius, sem considerar frações de graus, em lugares de elevada variação de temperatura entre as estações do ano.

EP.3.3. Para ser elegível à participação em uma promoção, a idade do consumidor, subtraída de 18 deve ser superior a 5. Utilizando ***set builder notation***, descreva o conjunto das idades de todos os consumidores que podem participar dessa promoção.

EP.3.4. Se uma tabela em um banco de dados relacional, contendo ***n*** linhas cujas combinações de valores das colunas sejam únicas, pode ser considerada como um conjunto, diga qual a cardinalidade desse conjunto. Qual a cardinalidade do conjunto formado pelos conjuntos de resultado de todas as consultas possíveis? Ilustre sua resposta com um exemplo numérico simples através de uma tabela contendo 4 tuplas.

PC.3.5. Desenvolva uma classe, denominada de "Conjunto" que faça uso de um array para armazenar os elementos de um conjunto (i.e.: deve possuir o mesmo comportamento da interface Set do TypeScript. Adicionalmente, deve implementar os métodos denominados de inserir, pertence, união e intersecção. Implemente sua classe utilizando o arquivo denominado set_model.ts, fornecido pelo professor. Faça uso de tipos genéricos.

PC.3.6. Considere uma base de dados relacional que possua uma tabela de usuários com os seguintes valores:

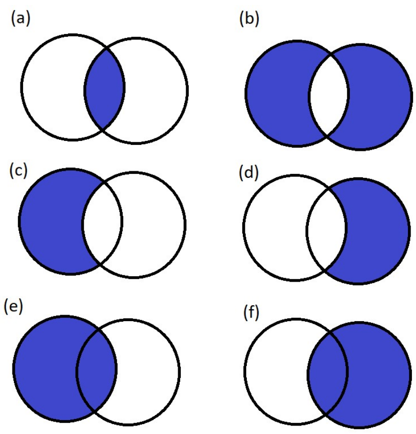
Nome	Telefone
“fabrício”	111
“beatriz”	222
“fabíola”	333

Utilizando consultas SQL, mostre que não é possível efetuar uma quantidade de consultas que seja superior a P(A), sendo A o conjunto formado pelas n-uplas que compõem as linhas da tabela. Para esse caso, ilustre todas as consultas e resultados possíveis.

PC.3.7. Considerando as seguintes tabelas contendo um cadastro de usuários e um cadastro de cidades, utilize consultas SQL, junções e operadores de conjuntos para ilustrar o resultado das operações mostradas nos diagramas de Venn.

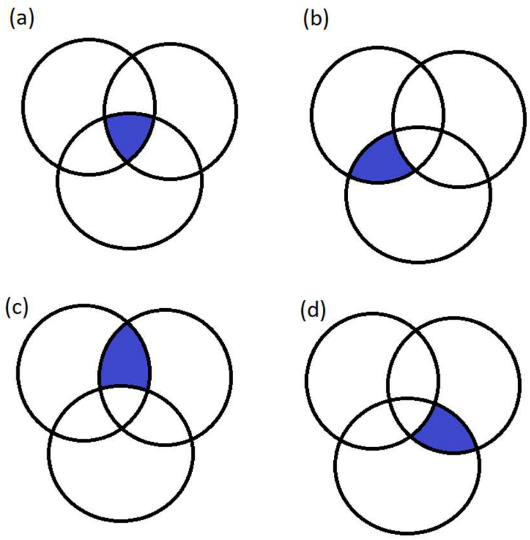
```
MariaDB [sets]> select * from usuario;
+----------+--------+
| nome     | cidade |
+----------+--------+
| fabricio | sjc    |
| thales   | sjc    |
| beatriz  | bel    |
| inacio   | palmas |
+----------+--------+
4 rows in set (0.000 sec)
```

```
MariaDB [sets]> select * from cidade;
+--------+-----------+
| cidade | regiao    |
+--------+-----------+
| bsb    | centro o. |
| bel    | norte     |
| sjc    | sudeste   |
+--------+-----------+
3 rows in set (0.000 sec)
```



PC.3.8. Repita o exercício anterior mas considerando a tabela adicional seguinte. Assegure-se de acrescentar valores às tabelas de forma que todas as regiões fiquem claramente ilustradas ao se efetuarem as consultas propostas:

```
MariaDB [sets]> select * from cidade_estado;
+--------+--------+
| cidade | estado |
+--------+--------+
| sjc    | sp     |
| bel    | pa     |
| bsb    | df     |
| manaus | am     |
+--------+--------+
4 rows in set (0.000 sec)
```



PC.3.9. Ilustre ao menos três diferentes tipos de Julia sets a partir do código-fonte fornecido pelo professor. Altere o esquema de coloração de modo a tornar a forma o mais próxima possível dos resultados mostrados em aplicações da Internet. Na sua resposta, inclua os valores iniciais para z e os demais parâmetros de ajuste da implementação do algoritmo.

PC.3.10. Utilizando a biblioteca ChartJS, adapte o gráfico de linhas ou dispersão XY para que exiba uma linha sólida desde a origem do sistema de coordenadas até as cordenadas do número complexo. Seu programa deve ser capaz de mostrar números complexos contendo magnitude menor ou igual a 10.

4. RELAÇÕES

4.1. DEFINIÇÃO DE RELAÇÃO

Uma relação *R* é um subconjunto que está contido no produto cartesiano de dois conjuntos *A* e *B*. *A* é dito conjunto de partida da relação e *B* é dito conjunto de chegada da relação ou contradomínio.

$$R \subseteq A \times B$$

Exemplo:

$$A = \{ 1, 2, 3\}$$

$$B = \{ 1, 5 \}$$

$$A \times B = \{(1,1), (1,5), (2,1), (2,5), (3,1), (3,5)\}$$

$$R = \{(1,1), (1,3)\}$$

$$R \subseteq A \times B$$

!

- ✓ Bancos de dados que fazem uso de tabelas são ditos relacionais pelo fato das tabelas serem relações entre conjuntos de valores de cada coluna.

Exemplo:

A = Gênero = { "Masculino", "Feminino"}

B = Nome = { "Ronaldo" , "Beatriz", "Jane"}

R = Cadastro = {("Masculino", "Ronaldo"), ("Feminino", "Jane"), ("Feminino", "Beatriz")}

$$R \subseteq A \times B$$

No banco de dados relacional que contenha uma tabela chamada de usuário, ter-se-ia:

Usuário

Gênero	Nome
Masculino	Ronaldo
Feminino	Jane
Feminino	Beatriz

4.2. REPRESENTAÇÃO ATRAVÉS DE FUNÇÃO PROPOSICIONAL

Quando o conjunto de partida e chegada de uma relação *R* é formado pelo mesmo conjunto *A*, pode-se representar a relação por:

$$R = \{(x,y) \in (A \times A) \mid P(x,y) \},$$

ou

$$R = \{(x,y) \in A^2 \mid P(x,y)\}.$$

Nesse caso, *P(x,y)* descreve como os elementos devem ser selecionados para compor a relação (assume valor verdadeiro para pares pertencentes a *R*).

Exemplo: Seja $R = \{(x,y) \in \mathbb{R}^2 \mid y = x^2\}$. Para esse caso, *P(x,y)* informa que *y* é um número real que deve corresponder ao quadrado de *x*. Dessa forma, os pares (1,1) e (2,4) pertenceriam a relação, mas o par (2,9) não pertenceria.

4.3. DOMÍNIO, IMAGEM E REPRESENTAÇÃO COM DIAGRAMA

O **domínio** de uma relação é o subconjunto, do conjunto de partida, que possui os elementos relacionados a algum elemento do conjunto de chegada na relação.

Exemplo: Sejam os conjuntos *A* = {1, 2, 3}, *B* = {3, 4} e a relação *R* = {(1, 3), (1,4), (3,3)}. Para esse caso

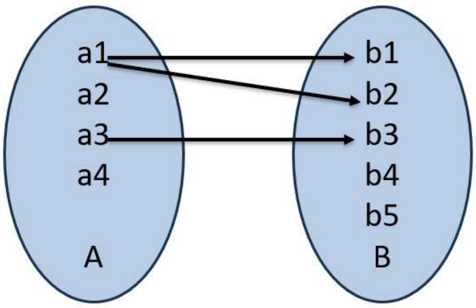
D(R) = {1, 3}.

A **imagem** da relação é o subconjunto, do conjunto de chegada, que está relacionado a pelo menos um elemento do conjunto de partida na relação.

Exemplo: Sejam os conjuntos *A* = {1, 2, 3}, *B* = {3, 4, 5} e a relação *R* = {(1, 3), (1,4), (3,3)}. Para esse caso tem-se

Im(R) = {3, 4}

Uma relação pode ser representada também utilizando-se uma notação gráfica, com diagramas elipses para representar os conjuntos e setas representativas para relacionamento entre os elementos dos conjuntos.



R ={(a1,b1), (a1,b2), (a3,b3)}

D(R) = {a1, a3}
Im(R) = {b1, b2, b3}

Conj. de partida = *A*
Contradomínio = *B*

Figura 3. Representação de relação com diagrama de setas.

4.4. RELAÇÕES E ÁLGEBRA RELACIONAL

Considerando que um par (x_0, y_0), pertencente a uma relação *R* pode ser representado em um plano cartesiano por um ponto, a sua **projeção** pode ser visualizada como sendo o valor daquele ponto considerando-se somente um dos eixos do sistema de coordenadas.

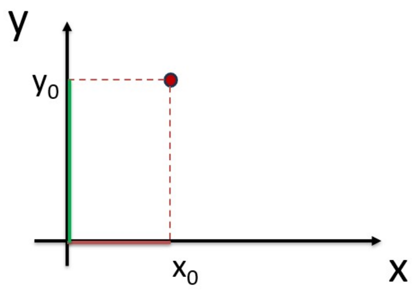


Figura 4. Projeção de um ponto de relação.

Considerando que uma relação pode ser utilizada na representação de uma tabela de um banco de dados relacional, pode-se definir a operação de projeção de modo análogo à projeção do par (x0, y0). Para esse caso, a projeção corresponde à extração de um único atributo do par. Simbolicamente, a projeção é representada da seguinte forma:

$$\pi_{\text{lista de atributos}}(\text{tabela})$$

Exemplo:

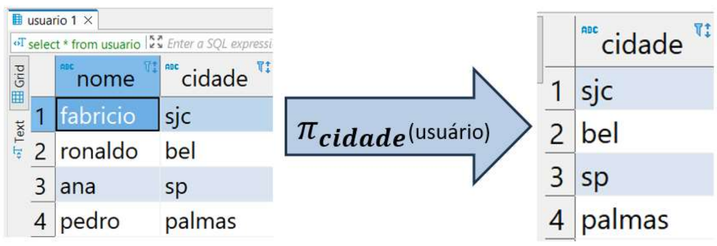


Figura 5. Exemplo de projeção de atributo.

Note-se que em uma consulta do tipo `select (atributo) from tabela`, a operação de projeção corresponde à parte `(atributo)` na sintaxe da consulta.

Uma relação R_2 é uma **<u>sub-relação</u>** de outra R_1 se toda tupla pertencente a R_2 mandatoriamente pertence a R_1.

Exemplo: Sejam $R_1 = \{(1,1), (1,2), (2,3)\}$ e $R_2 = \{(1,1), (2,3)\}$, então tem-se que R_2 é sub-relação de R_1.

O conceito de sub-relação é utilizado para definição do operador relacional de seleção, que é representado da seguinte forma:

$$\sigma_{\text{predicado}}(\text{tabela})$$

Exemplo:

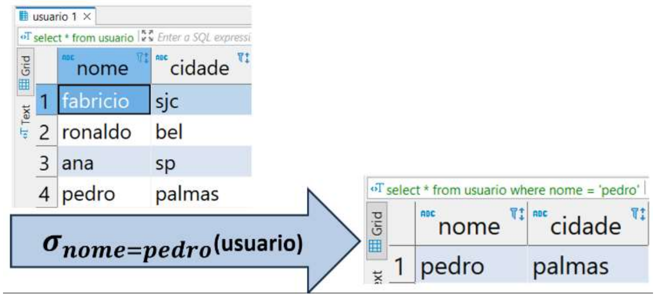


Figura 6. Exemplo de operação de seleção.

Note-se que em uma consulta do tipo `select * from tabela where atributo = valor`, a operação de seleção corresponde à parte `where atributo = valor`.

EXERCÍCIO RESOLVIDO

ER.4.1. Determine o domínio, a imagem e represente a relação *R* utilizando uma função proposicional, sabendo-se que o conjunto de partida e chegada da relação são os números naturais.

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

Resolução:

O domínio da relação é formada por todos os elementos do conjunto de partida que fazem parte dos pares da relação. Logo, $D(R) = \{1, 2, 3\}$.

A imagem da relação é formada por todos os elementos do conjunto de chegada que fazem parte da relação. Logo, $Im(R) = \{2, 3, 4\}$

A função proposicional pode ser considerada como uma combinação das condições satisfeitas pelos elementos dos pares *(a, b)*.

Observando-se o domínio, tem-se que *a* é natural e é menor do que 4.

Observando-se a imagem, tem-se que *b* é natural e é maior do que 1 e menor do que 5.

Observando-se os pares da relação, nota-se que, em todos os casos, *a* é menor do que *b*. Portanto, tem-se a seguinte representação para a relação:

$$R = \{(a, b) \in \mathbb{N}^2 \mid (a<4) \wedge (1<b<5) \wedge (a<b)\}.$$

A representação $\mathbb{N}^2$ é equivalente a $(\mathbb{N} \times \mathbb{N})$.

EXERCÍCIOS

EP.4.1. Considere a seguinte a relação $R = \{(1,1), (1,2),(1,3),(2,1), (2,2), (2,3), (3,1)\}$. Determine:

a) O domínio da relação.

b) A imagem da relação.

c) Considerando que o conjunto de partida é formado pelos elementos de $A = \{1, 2, 3, 4, 5, 6\}$ e que o contradomínio é formado pelos elementos de $B = \{0, 1, 2, 3, 4\}$, desenhe o diagrama de setas representativo da relação.

d) Uma relação é dita relação injetora, se para cada elemento do domínio é mapeado um elemento diferente do conjunto imagem. Diga se essa relação é injetora. Justifique.

e) Uma relação binária R^{-1} é dita relação inversa de *R*, se para cada par (x,y) pertencente à relação R^{-1}, o par *(y,x)* pertence à relação R. Determine a relação inversa R^{-1} para o caso da relação *R* especificada.

EP.4.2. Considere a seguinte relação: *R = {(1, 2), (1,3), (1,4), (2,3), (2,4), (3,4)}*. Determine uma sub-relação, diferente de R e descreva essa sub-relação através de uma função proposicional. Para essa mesma sub-relação, determine a sua inversa.

EP.4.3. Considere a seguinte reação: *R* = {(1, 1), (1,2), (1,3), (1,4), (2,2),(2,3), (2,4), (3,3), (3,4), (4,4)}. Determine:

a) A relação inversa de R.

b) O domínio e a imagem da relação.

c) A sua representação utilizando uma função proposicional.

d) O conjunto projeção da relação com relação ao eixo das ordenadas.

EP.4.4. Represente, utilizando a notação de álgebra relacional, a operação sobre a tabela de entrada que produz a tabela de saída.

Entrada:

Cadastro

id_empresa	razão_social	data_cadastro
4	Faculdade J SA	20/10/2020
55	Industria SA	10/01/2019
300	Comércio SA	01/01/2021

Saída:

55	Industria SA
300	Comércio SA

PC.4.5. Crie um programa que, informados dois conjuntos de entrada A e B, e a função proposicional associada à relação desejada, gera os pares *(a,b)* pertencentes à relação. Seu programa deve conter uma classe, denominada **Relação**, que deve possuir um método, denominado **`constuir_pares(number[], number[], object),`** que deve receber como argumentos dois conjuntos ou arrays numéricos e mais uma função que aceita como argumentos valores numéricos. O retorno deve ser um objeto (ex. JSON) contendo os pares da relação.

PC.4.6. Crie um programa que, informados os pares de uma relação, através de um array de pares ordenados, constrói a relação inversa. Denomine sua classe de Relação, e o método que executa tal operação de **`construir_rel_inversa(object),`** sendo **`object`** a relação original. O retorno deve ser a estrutura ou objeto contendo os pares da relação inversa.

PC.4.7. Crie as consultas em SQL que representam as seguintes operações sobre a tabela cadastro:

a) $\sigma_{id<55}$(Cadastro)

b) $\pi_{(razão\_social,\ data\_cadastro)}(\sigma_{(data\_cadastro>03/04/2019)}$(Cadasto))

Cadastro

id_empresa	**razão_social**	**data_cadastro**
4	Faculdade J SA	20/10/2020
55	Industria SA	10/01/2019
300	Comércio SA	01/01/2021

PC.4.8. Exemplifique, utilizando 2 tabelas contendo pelo menos 3 registros, a sintaxe, em álgebra relacional, utilizando uma composição de junção natural seguida de uma projeção. A seguir, monte a consulta SQL e ilustre o resultado.

5. FUNÇÕES

5.1. DEFINIÇÃO DE FUNÇÃO

Em linhas gerais, uma função é uma regra que associa a cada elemento de um conjunto *A* um único elemento de um conjunto *B*.

Notar que se houver um único elemento do conjunto *A* associado a mais de um elemento do conjunto *B* então a regra não descreve uma função.

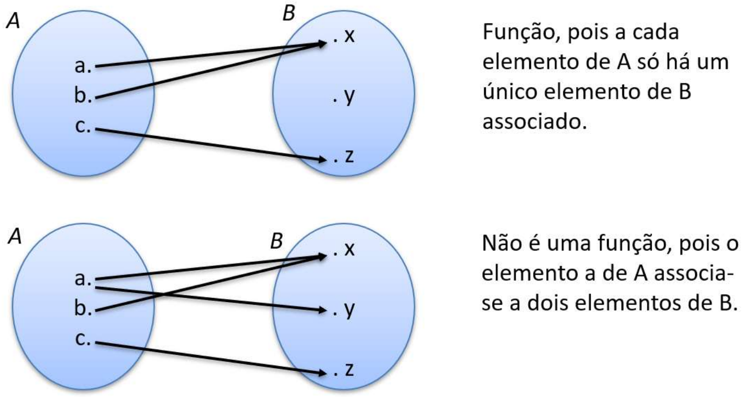


Figura 7. Diagrama ilustrativo de uma relação funcional (superior) e uma não funcional (inferior).

Formalmente, uma função é uma relação entre dois conjuntos *A* e *B* tal que para cada elemento do conjunto *A* existe **<u>um único elemento correspondente no conjunto</u>** *B*, tal que *(a,b)* pertence a essa relação.

$$\forall a \in A, \exists | \ b \in B \ \ | \ (a,b) \in F$$

Representa-se $F: A \rightarrow B$

!

✓ Funções são utilizadas para descrever relacionamentos entre variáveis. Nesse caso, funções correspondem a modelos e são amplamente utilizados na computação, física, economia, etc.

!

✓ A grande maioria das linguagens de programação de alto nível possui elementos de sintaxe que permitem ao programador definir e utilizar funções.

Uma função pode ser representada de diferentes formas.

Algumas dessas formas são: tabelas, diagramas de seta, gráfico de espalhamento, gráfico e equações (fórmulas).

Diagrama de seta:

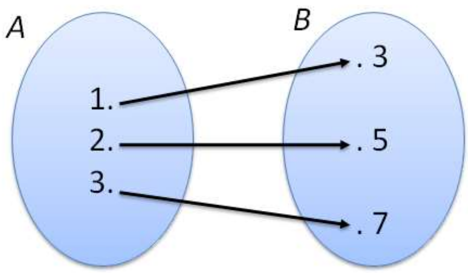


Figura 8. Representação de função com diagrama de seta

Tabela:

x	y
1	3
2	5
3	7

Gráfico de dispersão:

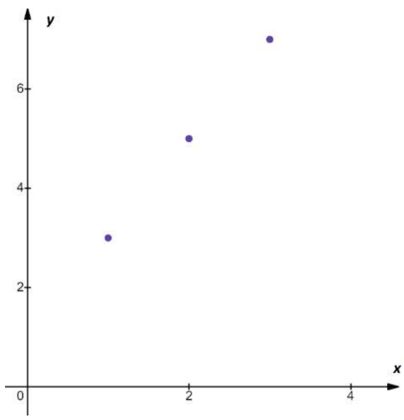


Figura 9. Representação de função através de gráfico de dispersão.

Gráfico (as vezes chamado de gráfico de linha):

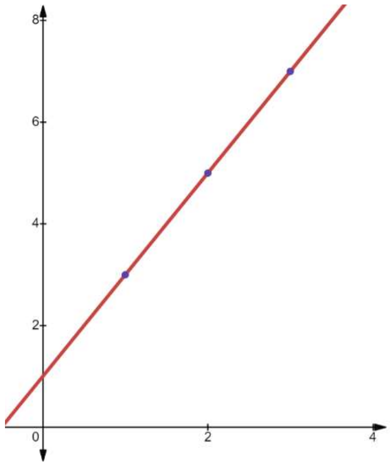


Figura 10. Representação de função através de gráfico de linha.

Fórmulas/equações:

$$y = 2x + 1$$

ou

$$f(x) = 2x + 1$$

Em todas as representações anteriores a variável x é dita variável independente, enquanto *y* é dita variável dependente (pois seu valor é obtido em função do valor da variável *x*).

Obter o valor da variável dependente a partir de um valor específico da variável independente é dito avaliar a função y = f(x) para o dado valor de x.

Exemplo: Avaliar a função *y = 2x+1* para *x = 9*.

$$f(9) = 2.9 + 1$$
$$f(9) = 19$$

Nesse caso, no plano cartesiano a função seria representada pelo ponto (9, 19).

!

- ✓ Uma função que associa a um valor de x a uma tupla é dita uma função vetorial. Ex. $f(x) = (x, x^2) = $ <x, x^2>
- ✓ Em aplicações, essas funções descrevem movimentos no plano, espaço 3d, etc.

5.2. DOMÍNIO E IMAGEM DE UMA FUNÇÃO

O domínio de uma função é o conjunto de todos os valores possíveis de *x*, em *A*, para os quais a função é definida.

A imagem de uma função é o conjunto de todos os valores de *y*, em *B*, que possuem correspondência em *A* através da função.

Exemplo: Seja a função y = (1/x). Então o domínio da função são todos os números reais, exceto x = 0. A imagem são todos os números reais exceto 0.

> **!**
> ✓ Em aplicações computacionais reais, tentar calcular valores para uma função (avaliar a função) para valores de x que não pertencem ao domínio da função tipicamente geram exceções (e.g.: divisão por zero, raiz quadrada de números negativos, etc.).

Na matemática, há uma infinidade de possibilidades para as funções. Algumas das mais utilizadas para modelagem de fenômenos são as funções polinomiais, exponenciais, logarítmicas e trigonométricas. Essas funções são descridas em linhas gerais a seguir.

5.3. FUNÇÕES ALGÉBRICAS E FUNÇÕES TRANSCENDENTAIS

Em uma **expressão algébrica** (formada combinando-se variáveis e constantes utilizando-se operações tais como soma, multiplicação, subtração, divisão e radicais), um termo é definido como um produto entre variáveis e/ou constantes.

Expressão Algébrica	Número de termos
$5x^2 + 4xy$	2 (binômio), contendo 2 variáveis
$\sqrt{x} + xy + 4$	3 (trinômio), contendo 2 variáveis
$\frac{10}{x} + z^2 + y$	3 (trinômio), contendo 3 variáveis

Obs: Uma expressão que não seja algébrica e que contenha funções exponenciais, logaritmos, senos, cossenos e combinações dessas funções são ditas expressões **transcendentais**.

Um **polinômio** é formado pela soma de um ou mais termos, sendo que os expoentes das variáveis devem ser números inteiros não negativos e não pode haver variáveis no denominador.

Exemplos:

$x^2 + x + 10$ é um polinômio (trinômio)

$\sqrt{x} + x + 10$ não é um polinômio

5.4. FUNÇÕES POLINOMIAIS

São funções algébricas que são obtidas através de expressões polinomiais e possuem o seguinte formato geral (para o caso de uma única variável *x*):

$$y = f(x) = a_n x^n + a_{n-1} x^{n-1} + \ldots + a_1 x^1 + a_0$$

As funções lineares, quadráticas e cúbicas são casos particulares de funções polinomiais.

5.4.1. FUNÇÕES LINEARES

São um caso particular de funções polinomiais. Possuem a seguinte forma geral:

$$y = f(x) = ax + b$$

Nesse caso, *a* e *b* são ditas constantes enquanto y e x são denominados de variáveis.

!

- ✓ Funções lineares são vastamente utilizadas na modelagem de fenômenos. A obtenção de modelos lineares para previsão de valores é tipicamente chamada de análise de regressão linear.

Um dos algoritmos mais populares para se ajustar os valores de pontos a uma função linear é o chamado algoritmo dos mínimos quadrados. Esse algoritmo é descrito a seguir:

```
Algoritmo mínimos_quadrados(x: vetor[1..N] de número,
                            y_r: vetor[1..N] de número[],
                            eta:número, épocas: número):
                            vetor[1..2] de número
        a: número ← 0
        b: número ← 0
        Para n = 1..épocas faça
              Para i = 1..N faça
                    y ← a*x[i] + b
                    erro ← y_r[i] - y
                    a ← a + eta*erro*x[i]
                    b ← b + eta*erro
              Fim-para
        Fim-para
        retorne [a, b]
```

O algoritmo recebe como entradas vetores que correspondem aos pares *(x,y)*, o fator de ajuste η e o número de épocas de ajuste. A saía do algoritmo é um vetor contendo as estimativas para os parâmetros *a* e *b* de um modelo linear.

5.5. FUNÇÕES EXPONENCIAIS

São funções em que a variável independente aparece no expoente.

Possuem a seguinte forma geral:

$$y = f(x) = Ca^x$$

Com $C \neq 0$, $a > 0$ e $a \neq 1$.

Nessa função, *a* é denominado de **base** e C é denominado de **valor inicial**.

Para que as funções exponenciais sejam definidas para todos os valores de x, deve-se ter a base $a > 0$.

Exemplo: O valor do montante para um capital C_0 que cresce a uma taxa de juros compostos *i*, num intervalo de tempo de *t* períodos de capitalização, pode ser modelado por:

$$M(t) = C_0(1 + i)^t$$

!

- ✓ As funções exponenciais são muito utilizadas na modelagem de fenômenos de engenharia, biologia, finanças etc.
- ✓ Pode-se provar que uma função exponencial com base a pode ser representada de modo equivalente por uma função exponencial com a base e = 2,7182818285...

5.6. FUNÇÕES LOGARÍTMICAS

O **logaritmo** de um número *a* é o nome dado a um **expoente** ao qual se deve elevar um número *b* para se obter o número *a*. Ou seja, se $log_b(a) = c$. O número *c* é o logaritmo, *b* é a base e *a* o logaritmando. Por definição, tem-se

$$a = b^c.$$

Ou seja, a operação de logaritmo é a operação inversa da operação de exponenciação.

A função logarítmica é a função onde a variável independente *x* aparece no lugar de *a* e *y* aparece no lugar de c.

$$y = f(x) = log_b(x)$$

Para essa função, as restrições são que $b > 0$ e $b \neq 1$.

A seguir, são mostradas as formas gerais dos gráficos das funções exponenciais e logarítmicas.

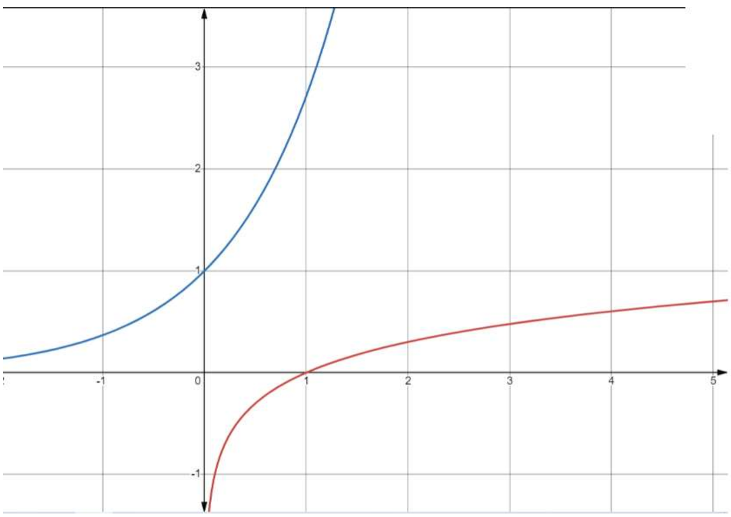


Figura 11. Forma geral do gráfico da função exponencial (azul, base > 1), e logarítmica (vermelho).

Quando a base do logaritmo é o número 10, tem-se um logaritmo decimal. Quando a base do logaritmo é o número e = 2,718281... diz-se que se trata do logaritmo neperiano.

- Funções logarítmicas são utilizadas sempre que se deseja obter um dado inverso referente a um dado que foi submetido a alguma operação de exponenciação.
- Um exemplo típico é a obtenção de períodos aos quais um determinado capital ficou sujeito a ocorrência de juros compostos.

5.7. FUNÇÕES TRIGONOMÉTRICAS BÁSICAS

As funções trigonométricas tiveram sua origem no estudo dos triângulos, mais especificamente no estudo das relações existentes entre os lados dos triângulos e os ângulos internos desses mesmos triângulos.

São seis as funções trigonométricas básicas: seno, cosseno, tangente, cotangente, secante e cossecante.

A seguir, ilustra-se a relação existente entre essas funções e as dimensões de um triângulo retângulo e, também as coordenadas no plano cartesiano.

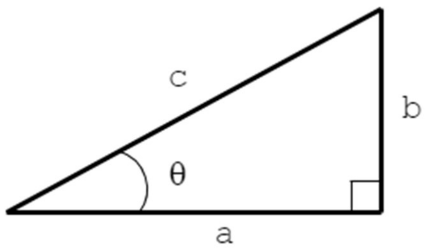


Figura 12. Triângulo retângulo.

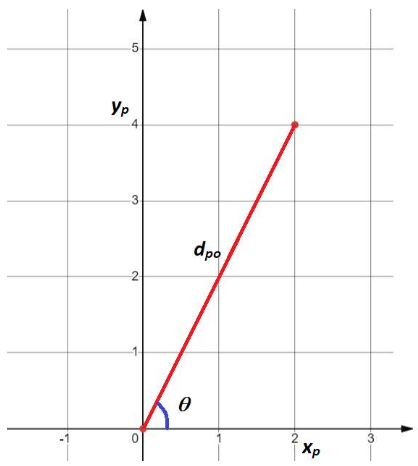


Figura 13. Representação de um ponto associado a um ângulo no sistema de coordenadas cartesianas.

Seis funções trigonométricas básicas.	
$sen(\theta) = \frac{b}{c} = \frac{y_p}{d_{po}}$	$cossec(\theta) = \frac{c}{b} = \frac{d_{po}}{y_p} = \frac{1}{sen(\theta)}$
$cos(\theta) = \frac{a}{c} = \frac{x_p}{d_{po}}$	$sec(\theta) = \frac{c}{a} = \frac{d_{po}}{x_p} = \frac{1}{cos(\theta)}$
$tan(\theta) = \frac{sen(\theta)}{\cos(\theta)} = \frac{b}{a} = \frac{y_p}{x_p}$	$cot(\theta) = \frac{cos(\theta)}{\text{sen}(\theta)} = \frac{a}{b} = \frac{x_p}{y_p} = \frac{1}{tan(\theta)}$

Para se obter uma função trigonométrica *y = f(x)*, substitui-se *x* no lugar do ângulo θ, ou seja, a função trigonométrica cosseno é, por definição *y = cos(x)*, *x* é a medida do ângulo em radianos. As demais funções são obtidas da mesma forma.

> **!**
>
> - ✓ Considerando uma dada circunferência, a medida correspondente a um radiano é aquela em que o tamanho do arco da circunferência é igual a um raio. Matematicamente, a relação de conversão de graus para radianos é $\theta_{\text{radianos}} = \frac{\theta_{graus}}{180}\pi$
> - ✓ Para testar se um sistema de cálculo, biblioteca, etc, está expresso em graus ou radianos, basta efetuar a seguinte conta utilizando a função seno: *sen(1,57)*. Se o valor der próximo de zero, então o sistema está utilizando graus. Se o valor der próximo de 1, então o sistema está utilizando radianos.

As funções trigonométricas são ditas funções periódicas, pois são funções que se repetem em intervalos regulares da variável *x*. Esses intervalos são chamados de períodos das funções. Para o caso da função seno, por exemplo o período, por vezes representado por T, é igual a 2π.

A figura abaixo ilustra o gráfico da função *sen(x)*.

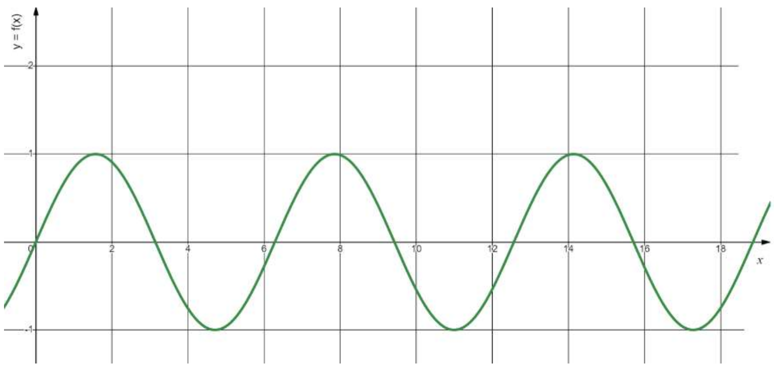


Figura 14. Gráfico da função sen(x)

!

✓ Funções trigonométricas são amplamente utilizadas em simulação e computação gráfica. Movimentos de rotação e decomposição de velocidades e forças utilizam funções trigonométricas.

EXERCÍCIOS

EP.5.1. Diga o que é uma função par, ímpar, injetora, sobrejetora e bijetora. Ilustre um exemplos através de gráficos de funções, mostrando tanto funções que atendem a classificação como as que não atendem. Dê preferência a funções que foram abordadas em aula.

EP. 5.2. Uma loja fornece descontos progressivos da seguinte forma: caso o cliente compre até 3 itens idênticos, não há desconto. A partir do 4º item, o desconto global da compra cresce 5% por item adicional ao terceiro, considerando o mesmo item, tal como mostrado na tabela seguinte:

Número de itens	Preço pago
1	Preço do item
2	Soma dos preços dos itens
3	Soma dos preços dos itens
4	Soma dos preços dos itens - 5%
5	Soma dos preços dos itens – 10%
...	...

Os descontos progressivos são aplicados até um limite de 40%. Dito isso:

a) Faça o gráfico da função que modela o preço total pago pelo cliente como função do número de itens idênticos presentes em seu carrinho de compras, considerando os descontos progressivos.
b) Sabendo que essa função é definida por partes, escreva sua expressão matemática para cada intervalo de itens pertinentes.
c) Escreva a expressão matemática da função que relaciona o preço final *p*, pago pelo consumidor, o número de itens idênticos *n* e o preço unitário p_u , considerando que n varia numa faixa de 3 a 11.
d) Mostre como calcular o número máximo de itens que geram desconto adicional ao cliente a partir da expressão matemática que descreve a função.

EP. 5.3. Dado um gráfico de uma função y = f(x), genérica, diga o que acontece com o gráfico quando se faz:

a) y = f(x-a), sendo a uma constante positiva
b) y = f(x+a), sendo a uma constante positiva
c) y = f(x) + a, sendo a uma constante positiva

d) y = f(x) – a, sendo a uma constante positiva
e) y = -f(x)
f) y = f(-x)

Para cada um dos itens, forneça um exemplo utilizando um dos gráficos das funções mostradas no capítulo (polinomial, exponencial, trigonométrica ou logarítmica).

EP.5.4. Determine o domínio e a imagem das seguintes funções:

a) $f(x) = \frac{1}{x^2+4x+2}$

b) $f(x) = \frac{1}{\sqrt{x}}$

c) $f(x) = \cos(x) + 10$

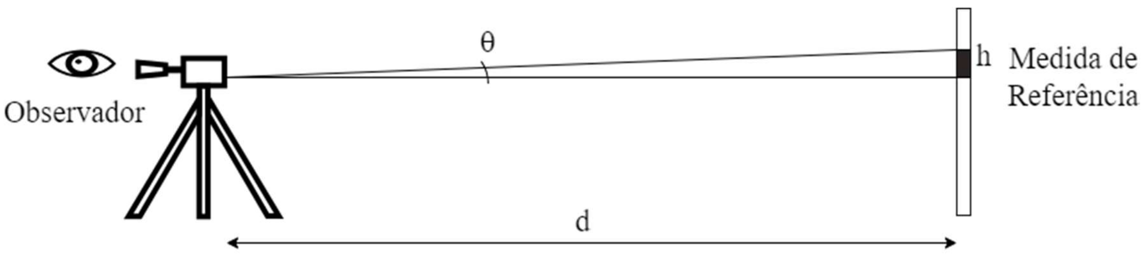
EP. 5.5. Um capital é investido a uma taxa de 0,5% ao mês sob o regime de juros compostos. Calcule o valor do montante pago após um período de:

a) 12 meses

b) 24 meses.

EP. 5.6. Um capital é investido a uma taxa de 1% ao mês sob o regime de juros compostos. Calcule em quanto o montante correspondente ao investimento dobra de valor em relação ao capital investido originalmente.

EP. 5.7. A imagem seguinte ilustra um esquema típico de medição de distância utilizado em topografia.



O observador utiliza um aparelho que é capaz de medir variações angulares θ ao apontar para uma barra com uma marcação de referência, medindo *h*. Determine:

a) A função que relaciona a distância ***d*** entre o observador e a base que sustenta a régua com a medida de referência ***h*** e o ângulo **θ.**
b) Calcule o valor da distância d, para o caso em que θ = 0,0286º e h = 50cm**.**

PC.5.8. Considerando o exercício EP. 5.2. Desenvolva um programa, utilizando funções, que receba como entradas a quantidade de itens de um determinado produto, o preço unitário do produto e as faixas de descontos considerando o número de itens. A saída do programa deverá ser o valor total da compra, considerando os descontos, o valor do desconto aplicado e a economia do cliente (diferença entre o preço sem desconto e com desconto). Seu programa deverá funcionar corretamente para uma quantidade arbitrária de itens comprados.

PC.5.9. Utilizando uma biblioteca tal como chartjs, ilustre o gráfico das funções trigonométricas básicas, exponencial, logarítmica, cúbica, quadrática e linear. Para cada função, ilustre diferentes valores para os coeficientes constantes (deixe parametrizado no seu código para poder redesenhar rapidamente um gráfico diferente). Assegure-se de criar gráficos com faixas de valores que permitam a adequada visualização do formato essencial da função.

PC.5.10. Baseando-se no código-fonte do exemplo fornecido pelo professor, que simula um círculo movendo-se na tela com velocidade formando um determinado ângulo com os eixos de coordenadas, desenvolva uma simulação de "gotas de chuva" que aparecem a partir da extremidade superior da tela e caem a partir de pontos pseudoaleatórios superiores. Num determinado instante, devem surgir simultaneamente de 10 a 100 gotas de chuva por vez. Seu programa deve receber como entrada a quantidade de gotas que devem surgir de uma vez só e deve ser possível informar se a simulação representará ou não o efeito do vento (nesse caso, as gotas devem cair de modo inclinado e a velocidade do vento deve ser informada pelo usuário.

PC.5.11. Baseando-se no código-fonte fornecido pelo professor, que simula uma esfera se movendo na tela, faça uma modificação para que a esfera surja em um ponto pseudoaleatório na extremidade superior da tela, com uma velocidade que forme um certo ângulo com os eixos x e y. A esfera, ao tocar em uma extremidade da tela deve ter sua velocidade alterada da seguinte forma: caso toque as bordas verticais da tela, sua velocidade na direção x muda de sinal e ela é refletida nessa direção (ou seja, $v_x = - v_x$). O caso análogo deve ocorrer para toques nas extremidades horizontais superiores e inferiores ($v_y = -v_y$). Seu programa deve receber como entradas: a cor de fundo, a cor da esfera, o módulo da velocidade e o ângulo inicial formado entre o vetor velocidade e a parte superior da tela.

PC.5.12. Baseando-se no código-fonte fornecido pelo professor, desenvolva uma espécie de jogo onde objetos caem a partir de pontos pseudoaleatórios da parte superior da tela e, quando interceptados pela plataforma retangular na base, movida a partir de eventos do teclado, os objetos são destruídos e uma pontuação de 10pts é acrescentada para o jogador. Caso o objeto atinja a base da tela e não seja interceptado pela plataforma, o jogo é encerrado. Seu programa deve ser parametrizado em termos de velocidade de surgimento dos objetos.

PC.5.13. Repetir o PC.5.12 para o caso de objetos surgindo da lateral esquerda da tela e a plataforma se movimentando verticalmente na parte direita da tela.

PC.5.14. Em sistemas modernos de processamento de linguagem natural, é comum que o texto de entrada seja pré-processado e, com base em algumas estatísticas de ocorrência de palavras, ele seja convertido em outro eliminando-se elementos com pouco ou nenhum significado. Faça um sistema que, dado um texto de entrada (arquivo .txt), gere um gráfico onde o eixo das abscissas correspondam a uma palavra e o eixo das ordenadas correspondam ao número de vezes que essa palavra aparece no texto. Em termos de probabilidade, esse gráfico associa a uma dada palavra sua probabilidade de ocorrência no texto (função distribuição de probabilidade) e serve para que sejam selecionadas as chamadas stop words (palavras muito frequentes e com baixo significado). Ilustre a execução desse sistema para um texto de entrada arbitrário (.txt) fornecido pelo usuário. Não esqueça de converter todos os caracteres para minúsculo e efetuar a remoção de caracteres tais como pontos, etc. Para facilitar a visualização desse gráfico, elabore uma legenda e atribua valores numéricos para palavras ao invés de escrevê-las no eixo *x*.

PC. 5.15. Desenvolva um programa que simule o efeito de uma onda se deslocando para a direita. Para isso, plote em um objeto canvas a função seno($x - \theta(t)$) para valores de θ variando ao longo do tempo (aumentando), em intervalos de 1 segundo.

PC. 5.16. Desenvolva um programa, utilizando funções e o componente canvas, que simule o estouro de fogos de artifício. O “estouro” de um fogo de artifício específico deve acontecer no centro do canvas e os “raios luminosos” devem ser separados por ângulos configuráveis (ex. 30º, 45º, 60º , etc.). Além da separação dos raios, seu programa deve ser configurável com relação às cores de fundo e cores dos raios luminosos.

PC. 5.17. Desenvolva um programa que desenhe no canvas uma circunferência, um ponteiro de segundos de um relógio. Esse ponteiro deve se mover tal como os ponteiros de segundo de um relógio analógico. A cada ciclo completo (60s) esse programa deve alterar a cor de fundo da circunferência e do ponteiro.

PC. 5.18. Desenvolva um programa que, informados o prazo da aplicação e a taxa de juros (percentual, a juros compostos) retorna o valor do montante.

PC. 5.19. Desenvolva um programa que, informado um montante, um capital inicial e a taxa de juros calcula o tempo mínimo necessário para que o capital inicial se torne o montante. Seu programa deve considerar valores inteiros para os períodos de capitalização (saída inteira).

PC. 5.20. Desenvolva um programa que calcule a distância, medida pelo instrumento topográfico do exercício EP.7. A entrada para o programa deve ser o ângulo, em graus, e a medida de referência h, em centímetros. A saída deve ser a distância, em metros.

PC. 5.21. Utilizando o código-fonte fornecido pelo professor, que efetua o ajuste utilizando o algoritmo dos mínimos quadrados, obtenha dados de algum fenômeno que possa ser representado por um modelo linear e efetue o ajuste dos parâmetros do modelo. Plote o gráfico representativo do modelo comparado com a dispersão XY dos dados originais utilizados para ajustar o modelo.

PC.5.22. Vários modelos não lineares (ex. cúbicos) podem ser aproximados por modelos lineares se o intervalo de valores para a variável x for pequeno. Gere dados para modelos exponenciais, cúbicos, quadráticos e senoidais e obtenha uma aproximação linear através da utilização do método dos mínimos quadrados, fornecido pelo professor. Plote os dados comparativos dos diferentes modelos comparados com o modelo linear aproximado. Qual sua conclusão?

6. LÓGICA MATEMÁTICA

A área da matemática conhecida como Lógica Matemática lida com o processo de obtenção de conclusões a partir de informações previamente disponíveis. Esse processo de obtenção de uma conclusão a partir de fatos previamente conhecidos, **denominados de premissas**, é conhecido como **inferência matemática.**

No processo de inferência matemática, a organização do raciocínio a partir das **premissas** até que se chegue a uma **conclusão** constrói um **argumento lógico**. Nesse caso, premissas são proposições (declarações que podem ser classificadas como verdadeiras ou falsas) que são assumidas como verdadeiras e a conclusão é uma proposição que se pode concluir a partir das premissas.

Esquematicamente, um argumento lógico possui a seguinte estrutura geral:

Premissa 1
Premissa 2
....
Premissa N
Conclusão

Não existe número mínimo ou máximo de premissas para dar suporte a uma conclusão.

Quando não são necessárias premissas para dar suporte a uma conclusão dizemos que a conclusão é **autoevidente** (ou **verdade autoevidente)**. Um exemplo de conclusão ou verdade autoevidente é a seguinte: "Qualquer coisa é idêntica a si mesma" (Lei da identidade).

Um argumento lógico é dito **válido**, quando a conclusão é consequência direta das premissas. Caso contrário o argumento é dito **inválido**.

A argumentação lógica pode, ainda, ser agrupada em duas categorias. A **argumentação dedutiva** e a **argumentação indutiva**.

Um **argumento dedutivo** é aquele em que a veracidade das premissas determina totalmente a veracidade da conclusão.

Exemplo:

Premissa 1. Sempre que chove a rua fica molhada.

Premissa 2. Hoje está chovendo

Conclusão. A rua está molhada.

Argumentos dedutivos são os pilares da matemática moderna. Exemplos de argumentos dedutivos são os que são mostrados em **teoremas.**

Um **argumento indutivo** é aquele em que a veracidade das premissas torna muito provável a veracidade da conclusão, mas não assegura totalmente tal veracidade.

Exemplo:

Premissa 1. Quando se investe dinheiro em fundos diversificados, o risco é reduzido.

Premissa 2. A carteira de investimento do cliente está diversificada.

Conclusão. É provável que o risco do cliente seja reduzido.

Os argumentos indutivos são muito utilizados nos estudos experimentais empíricos e na modelagem matemática, pois possibilitam a obtenção de generalizações a partir de uma quantidade reduzida de informações que são analisadas.

O objeto de estudo da Lógica Matemática encontra ampla aplicabilidade na área de computação. Entre essas aplicações, pode-se destacar a construção de programas através da lógica de programação, os sistemas especialistas baseados em regras, as análises de dados, etc.

Dentro da área da lógica matemática, há três grandes áreas que lidam com o processo de inferência de modo distinto: a **Lógica Categórica**, a **Lógica Proposicional** e a **Lógica de Predicados**.

6.1. LÓGICA CATEGÓRICA

A Lógica Categórica lida com o processo de inferência relacionada à classificação ou categorização de objetos.

Dadas certas informações relacionadas a um determinado objeto, o objetivo é concluir se esse objeto pode ou não ser enquadrado como um objeto pertencente a uma certa categoria ou classe, o que consequentemente implica nesse objeto ser dotado de certas propriedades. Essas categorias ou propriedades tipicamente lidam com qualidades ou quantidades

Exemplo: Todos os estudantes da faculdade possuem um número de matrícula. Maria é uma estudante da faculdade, logo, pode-se concluir que Maria possui um número de matrícula.

O processo de conclusão utilizando lógica categórica tipicamente usa uma série de inferências imediatas que dependem da estrutura da proposição categórica original.

Exemplo:

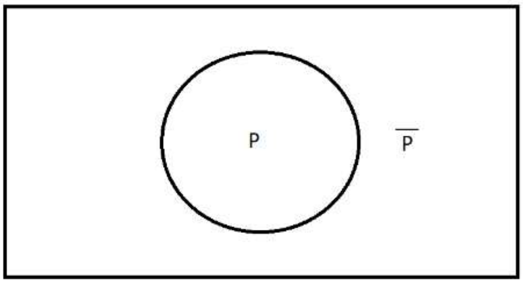
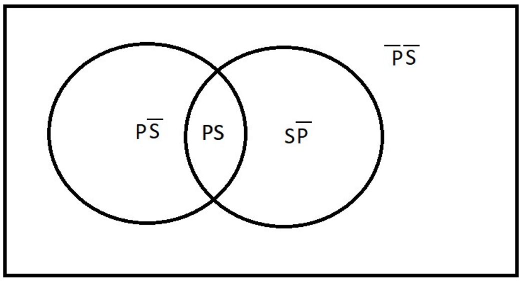
Premissa 1. **Todos** os seres humanos vivos **respiram.**

Conclusão imediata: **Não existe** ser humano vivo **que não respire.**

Conclusão imediata: **Alguns** seres humanos vivos **respiram**.

Diagramas de Venn são comumente utilizados na análise e na verificação de argumentos que utilizam lógica categórica, em especial argumentos que fazem uso de duas premissas categóricas e obtém uma conclusão, nesse caso denominado de **silogismo categórico**.

As seguintes notações são utilizadas para representar proposições e suas inter-relações:

	O círculo representa o conjunto de objetos que pertencem a uma certa categoria P. Os objetos que não fazem parte desse círculo, estão na região externa e são representados por $\overline{P}$.
	Objetos que pertencem a mais de uma categoria são indicados pela intersecção entre conjuntos.

Para verificar se um silogismo categórico é válido ou não, desenha-se o diagrama de Venn correspondente, marcam-se as áreas correspondentes às afirmações e procede-se com a verificação se a conclusão encontra-se indicada em uma das áreas devidamente assinaladas.

Durante a marcação das áreas nos conjuntos, é comum utilizar o sombreado ou hachurado para indicar inexistência, o ☑ para se indicar a existência e uma ? para se indicar a incerteza.

Exemplo:

Efetuar a análise de validade do seguinte silogismo categórico:

Premissa 1. Todas as ocasiões de trovão (T) são ocasiões de chuva (C).

Premissa 2. Todas as ocasiões de relâmpagos (R) são ocasiões de trovão.

Conclusão. Todas as ocasiões de relâmpago são ocasiões de chuva.

O diagrama abaixo ilustra a representação do silogismo categórico. Notar que a única região possível para a ocorrência de relâmpago se dá quando houver chuva e é consequência direta das duas premissas, que excluem as áreas em cinza e amarelo.

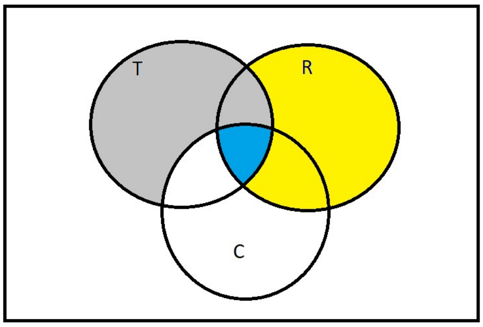


Figura 15. Análise de silogismo categórico utilizando diagrama de Venn.

Exemplo:

Analisar o seguinte silogismo categórico:

Premissa 1. Alguns gatos pretos capturam ratos.

Premissa 2. Todo gato que captura rato é um bom gato.

Conclusão. Portanto, alguns gatos pretos são bons gatos.

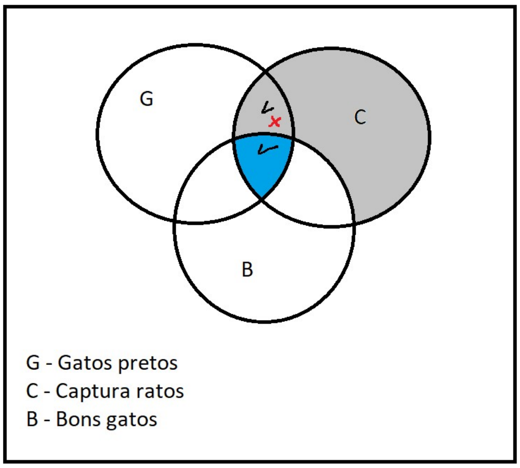


Figura 16. Diagrama de Venn para silogismo categórico. Exemplo 2.

6.2. LÓGICA PROPOSICIONAL

A **lógica proposicional,** também conhecida como **lógica sentencial** ou **cálculo proposicional**, considera as proposições como unidades fundamentais. Enquanto a lógica categórica tem uma classe ou categoria como unidade fundamental (e.g.: um objeto é ou não pertencente a tal categoria), a lógica proposicional tem como unidade fundamental uma declaração/proposição.

Exemplo:

Lógica categórica, análise das partes da declaração.

Todos os gatos são felinos (**quantificador + sujeito + cópula + predicado**)

Lógica proposicional

Todos os gatos são felinos (**P**)

Nesse caso, a declaração é uma unidade atômica e detalhes internos não são considerados para a análise. Para o exemplo anterior, uma outra declaração do tipo.

Todos os felinos possuem garras (**Q**)

Nada teria a ver com a proposição **P**, ou seja, não há relação entre elas sob o ponto de vista de análise da argumentação.

Na lógica proposicional, o objetivo é verificar como o valor verdade da proposição conclusão veria de acordo com o valor verdade das premissas.

Para representar uma proposição P (uma declaração que pode ser assumida como verdadeira ou falsa), utiliza-se a chamada tabela verdade. Nesse caso, os valores possíveis são como ilustrados a seguir.

Tabela 1. Valores verdades possíveis para uma proposição.

Valor de P	Significado
V	Proposição verdadeira
F	Proposição falsa.

Uma proposição pode ser atômica, ou seja, possuir um dos valores V ou F, ou resultar da combinação de outras proposições utilizando uma função verdade.

A seguintes funções verdades fundamentais são utilizadas no cálculo proposicional:

Negação ( )

P	¬ P
V	F
F	V

Conjunção (^ , E)

A disjunção é uma operação simétrica, ou seja P ^ Q = Q ^ P.

P	Q	P ^ Q
F	F	F
F	V	F
V	F	F
V	V	V

Disjunção (v , OU)

Essa função proposicional é também conhecida como **ou inclusivo**. Notar que a relação resultante entre as proposições é simétrica, ou seja, P v Q = Q v P.

P	Q	P v Q
F	F	F
F	V	V
V	F	V
V	V	V

Implicação material (→, SE ENTÃO)

Na implicação material, a proposição do lado esquerdo do símbolo → é denominada de proposição **antecedente**, a proposição do lado esquerdo desse símbolo é denominada de **consequente**.

P	Q	P → Q
F	F	V
F	V	V
V	F	F
V	V	V

Diferentemente das funções E e OU, a implicação não é simétrica, ou seja, $P \rightarrow Q \neq Q \rightarrow P$

A proposição antecedente é dita proposição **suficiente.** Já a proposição consequente é dita proposição **necessária.**

Notar que, para os casos onde o antecedente é verdadeiro e o consequente também é verdadeiro, temos uma relação de implicação que é disponibilizada em linguagens de programação através dos blocos **SE ... ENTÃO** (**IF ... THEN**)

Equivalência ($\leftrightarrow$, SE E SOMENTE SE)

A função de equivalência é uma função simétrica, ou seja $P \leftrightarrow Q = Q \leftrightarrow P$. Trata-se de uma composição entre duas funções mais fundamentais, no caso $P \rightarrow Q$ E $Q \rightarrow P$.

P	**Q**	**$P \leftrightarrow Q$**
F	F	V
F	V	F
V	F	F
V	V	V

Para se avaliar uma proposição composta de outras proposições atômicas utilizando as funções proposicionais deve-se montar a tabela verdade considerando todas as combinações possíveis para cada proposição atômica.

Exemplo:

Avaliar a proposição resultante de $\neg (P \wedge Q)$.

P	**Q**	**P^Q**	**$\neg$ (P ^Q)**
V	V	**V**	**F**
V	F	**F**	**V**
F	V	**F**	**V**
F	F	**F**	**V**

Uma das principais aplicações da lógica proposicional está na construção de sistemas especialistas baseados em regras. Nesse caso, argumentos de entrada para o sistema são convertidos em proposições (que podem ser avaliadas como verdadeiras ou falsas) e, caso uma regra do sistema (composta pela combinação dessas proposições) assuma a condição de verdadeira para os argumentos informados, o sistema fornece como resposta a ação correspondente.

Esquematicamente, para os sistemas especialistas baseados em regras tem-se:

Regra:

Antecedente $\rightarrow$ Consequente, que equivale a **SE** Antecedente **ENTÃO** consequente

Antecedente = Combinações de argumentos de entrada convertidos para proposições.

Consequente = Ação a ser executada quando o antecedente assume o valor verdadeiro.

Exemplo:

ler(temperatura)

Se temperatura > 37 **então**

imprimir(está com febre)

Para esse caso o antecedente seria a proposição (temperatura > 37) avaliada para a temperatura informada como argumento de entrada. O consequente seria imprimir("está com febre")

6.3. LÓGICA DE PREDICADO

6.3.1. CONCEITUAÇÃO

É um sistema lógico capaz de lidar tanto com a estrutura interna como externa associada às proposições.

Nesse sistema, **predicados** são os qualificadores e, em geral, são representados por letras capitais.

Exemplo: M, F, etc. são qualificadores que podem representar relações entre elementos, tais como "é do sexo masculino" ou "é do sexo feminino".

Sujeitos são objetos que são representados, tipicamente, por constantes (a, b, c..) ou variáveis (x, y, z, ...).

Para se formar uma proposição, aplica-se um predicado a uma constante ou a uma variável. Caso seja aplicado a uma variável, tem-se as chamadas funções proposicionais (cujo valor depende da variável).

Exemplo: Seja x uma pessoa genérica (variável) e seja R o predicado referente à respiração respira. Para se representar a proposição "Pessoa respira" escreve-se

$$R(x)$$

Notar que a veracidade de tal proposição depende de valores assumidos por x. Se x assume o valor de "Pedro" e essa pessoa respira, escreve-se

R(Pedro)

Que nesse caso assume o valor **verdadeiro.**

Uma função proposicional que não possui como argumentos um conjunto de constantes, só pode ser avaliada como verdadeira ou falsa no caso de se utilizarem os chamados quantificadores que são aplicados às variáveis.

6.3.2. QUANTIFICADORES

QUANTIFICADOR EXISTENCIAL: Utilizado para representar a existência de um determinado objeto que será utilizado na avaliação da função proposicional.

Exemplo: Existe um estudante x que estuda na faculdade. Seja E o predicado estuda na faculdade.

$$\exists x\ E(x)$$

QUANTIFICADOR UNIVERSAL: Utilizado para representar a generalidade da veracidade de uma função proposicional considerando um determinado conjunto de valores assumidos pelo sujeito.

Exemplo: Todos os estudantes x possuem um número de matrícula. Seja M o predicado possui um número de matrícula e x o conjunto de todos os estudantes de uma certa instituição de ensino.

$$\forall x\ M(x)$$

!

- ✓ A lógica de predicados, bem como os quantificadores, são amplamente utilizados na descrição matemática de algoritmos e sistemas computacionais

EXERCÍCIO RESOLVIDO

ER.6.1. Para o texto seguinte, identifique as premissas, a conclusão e a avalie a validade do argumento do silogismo categórico utilizando um diagrama de Venn.

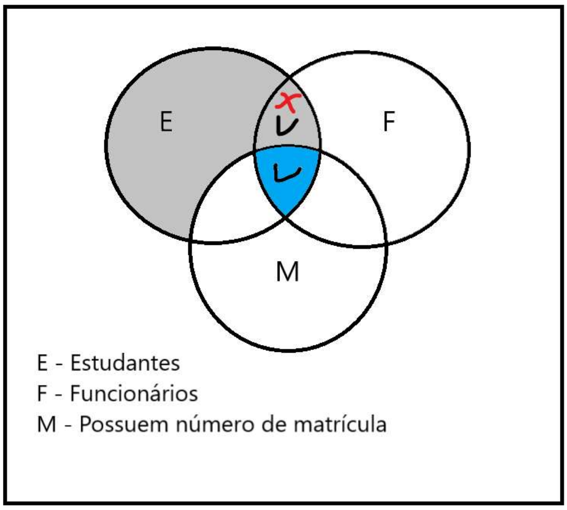
É comum e há casos de funcionários das faculdades de tecnologia que se tornaram estudantes dessas faculdades, consequentemente, possuindo um número de matrícula. Nessas faculdades de tecnologia, todos aqueles que são alunos devem possuir um número de matrícula.

Resolução:

Premissa 1. Todos os estudantes possuem número de matrícula.

Premissa 2. Existem funcionários estudantes.

Conclusão. Existem funcionários que possuem número de matrícula.



Para esse caso, o argumento é válido e está provado no diagrama de Venn.

ER.6.2. Prove que a função de equivalência entre duas proposições P e Q (P ↔ Q) é igual a conjunção entre as proposições P → Q e Q →P

Resolução:

A tabela verdade demonstra a igualdade:

P	**Q**	**P → Q** (1)	**Q →P** (2)	**(1) ^ (2)**	**P ↔ Q**
F	F	V	V	V	V
F	V	V	F	F	F
V	F	F	V	F	F
V	V	V	V	V	V

ER.6.3. Representar a seguinte declaração utilizando a lógica de predicados: Se uma pessoa específica a é filha de uma pessoa específica b e outra pessoa c também é filha da pessoa b, então a e b são irmãos.

Resolução:

Relação é filho(a): F(x,y)

Relação é irmão(ã): I(x,y)

$$F(a,b) \wedge F(c,b) \rightarrow I(a,c)$$

ER.6.4. Generalizar a proposição do exercício anterior utilizando quantificadores.

Resolução: Dado que SEMPRE que duas pessoas compartilham um pai ou mãe são consideradas irmãs, então deve-se utilizar o quantificador universal.

$$\forall x \, \forall y \, \forall z \, [\, (\, ((F(x,y) \wedge F(z,y)) \wedge (x \neq z) \,) \rightarrow I(x,z) \,]$$

EXERCÍCIOS

EP.6.1. Nos argumentos seguintes, identifique as premissas e a conclusão. Reescreva os argumentos de modo a deixar o processo de inferência claro, fluindo das premissas para a conclusão.

a) Alguns usuários acessam o sistema após as 23h. Todos os acessos após as 23h tem baixo tempo de espera. Dessa forma alguns usuários tem baixo tempo de espera.

b) Percebe-se que todos os objetos que estão marcados com as etiquetas (TxyE) no estoque foram entregues pela transportadora Txy.

c) O fato do log do sistema apresentar um grande volume de dados registrados significa que houve um grande número de tentativas de violação de segurança.

EP.6.2. Represente usando diagrama de Venn e determine a validade dos seguintes argumentos (especifique quem são as premissas e quem é a conclusão no silogismo categórico):

a) Todos os consumidores de barras cereal estão acima do peso. Alguns praticantes de musculação consomem barras de cereal. Portanto, alguns praticantes de musculação estão acima do peso.

b) Nenhum estudante aprovado em estrutura de dados fica desempregado. Existem pessoas felizes que foram aprovadas em estruturas de dados. Dessa forma, existem pessoas felizes empregadas.

c) Todos os homens são humanos. Alguns humanos são franceses. Portanto, alguns homens são franceses.

PC.6.3. Suponha que você foi contratado para efetuar uma análise de dados para ajudar a entender o perfil dos consumidores de um mercado varejista. Explique como você faria, em termos de consulta de dados em uma base de dados, para dar suporte ao seguinte argumento que subsidia uma estratégia para impulsionar as vendas:

Todos os consumidores que adquirem café, também adquirem leite. Todos os consumidores que compram adoçante, também adquirem café. Logo, todos os consumidores que adquirem adoçante, também adquirem leite. Logo, uma boa estratégia para promover a compra dos adoçantes é fornecer desconto para o leite e o café sempre que houver promoções de adoçantes.

Na sua resposta, explique:

a) A lógica da conclusão do argumento utilizando diagramas de Venn.

b) Como a conclusão referente à validade ou não do argumento pode ser comprovada a partir dos dados presentes em um banco de dados de transações.

c) Como isso é feito considerando a base de dados fornecida pelo professor, que apresenta dados de transações estruturados da seguinte forma:

Id transação	**Produto**
1	Produto 1
1	Produto 2
1	Produto 3
2	Produto 1
2	Produto 4
...	
N	Produto n

Utilize o código-fonte e a base de dados fornecidos pelo professor, explicando seu funcionamento e a sua contribuição ao código para possibilitar o suporte à conclusão do argumento (é verdadeiro ou falso?).

Para restaurar: mariadb db_name < market_data.sql

PC.6.4. Baseando-se no exemplo do código fornecido pelo professor, desenvolva um pequeno sistema especialista, configurável, que informa se um usuário está ou não com sintomas que podem ser diagnosticados como uma determinada doença. Seu sistema especialista deve possuir uma base de regras configurável que não seja ***hard coded***. Exemplifique a utilização desse sistema para pelo menos 2 doenças com sintomas bem distintos (devem conter mais de um sintoma característico na base de regras).

7. INTRODUÇÃO À ANÁLISE DE ALGORITMOS

7.1. INTRODUÇÃO

Muitas vezes o desenvolvedor de sistemas se depara com situações onde deve tomar certas decisões tais como, por exemplo, a escolha de algum algoritmo já disponível para um dado fim ou então sobre como projetar um novo algoritmo. Há diversos critérios que podem ser utilizados para justificar uma determinada escolha, entre eles pode-se citar a facilidade de compreensão, a eficiência na utilização de recursos, o reuso de código, entre outros. Um critério significativo e que será brevemente explicado é a chamada análise de **complexidade do algoritmo**.

7.2. COMPLEXIDADE DE ALGORITMOS

A complexidade de um algoritmo possui relação com a quantidade de recursos necessários à resolução de determinado problema e com a eficiência na utilização desses recursos.

Dependendo do tipo de recurso analisado, pode-se ter, por exemplo a complexidade associada ao tempo de execução (**complexidade de tempo**) ou à memória utilizada (**complexidade de espaço**).

Figura 17. A complexidade de tempo (complexidade temporal) tem relação com o tempo de execução do algoritmo.

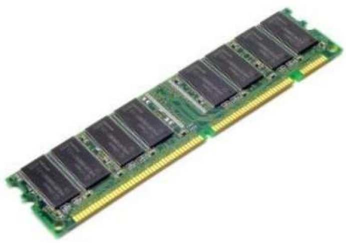


Figura 18. A complexidade de espaço tem relação com a quantidade de memória principal utilizada durante a execução do algoritmo. O termo espaço tem a ver com o menor ou maior requisito de espaço físico do dispositivo de memória.

7.2.1. COMPLEXIDADE ESPACIAL

Considerando um bloco funcional com uma entrada e saída bem definidos, a complexidade espacial de um algoritmo depende tanto da quantidade de memória necessária para armazenar a entrada (**dados de entrada**) como da quantidade de memória interna ao bloco funcional (**dados auxiliares**) que é requerida para o completo processamento.

Em geral, o tamanho da saída é excluído do cálculo da complexidade (pode já está incluído implicitamente no espaço auxiliar).

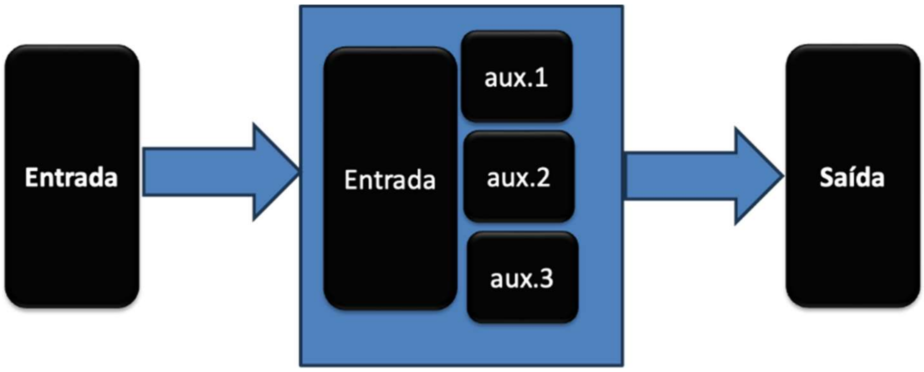


Figura 19. A complexidade espacial considera, em geral, os dados de entrada e os dados auxiliares do algoritmo.

Exemplo: Calcular a complexidade espacial do seguinte algoritmo que corresponde a uma função que efetua a cópia de um array.

```
Função copiar(a: Array [0..n-1] de número): Array [1..n-1] de número
      copia: Array [0..n-1] de número ← [ ]
      Para i:número = 0 .. (n-1)
            copia[i] ← a[i]
      Fim-para
      retorne copia
Fim-função
```

Para esse caso, tem-se que serão necessárias n posições de memória interna para o array que é cópia mais 1 posição de memória para a variável i, logo, a complexidade espacial do algoritmo será dada por $C_e(n) = n+1$.

7.2.2. COMPLEXIDADE DE TEMPO

Tipicamente, em análise de algoritmos, a análise de tempo (**complexidade de tempo**) não é feita em termos absolutos ou em relação a uma máquina particular, mas sim em termos de um certo número de operações básicas definidas (e.g.: comparações, movimentação de dados, etc.) para um dado tamanho n da entrada. A análise é feita desta forma para evitar que os resultados sejam influenciados por características específicas de uma determinada máquina (e.g.: uma máquina nova e mais rápida pode fazer com que o projetista erroneamente ache que um determinado algoritmo é mais eficiente do que outro quando, na verdade, o que houve foi aumento do poder computacional e não a melhoria, de fato, do algoritmo).

Exemplos de operações básicas que são utilizadas na análise de algoritmos incluem: atribuições (incluindo inicializações), operações matemáticas (somas, multiplicações, divisões, etc.), operações relacionais, operações lógicas, movimentações de posições de memória, entre outras. Apesar dessas operações demandarem tempos diferentes, é comum que se considere que várias delas apresentam o mesmo consumo por parte de uma determinada máquina. Essa suposição facilita o cálculo da complexidade de tempo do algoritmo analisado.

Além disso, a análise de tempo pode depender da instância do problema que está sendo resolvido para esse tamanho de entrada *n*. Esta instância pode ser o **pior caso**, o **caso médio** ou o **melhor caso**.

- Na análise do **pior caso**, para uma entrada de tamanho *n*, é requerido o maior número de operações básicas possíveis para se resolver o problema.
- Na análise do **melhor caso**, para uma dada entrada de tamanho *n*, é requerido o menor número de operações básicas para se resolver o problema.
- Na análise de **caso médio**, avalia-se o número de operações básicas para uma entrada de tamanho *n* considerando-se a média de entradas possíveis.

Exemplo: Para o caso de busca sequencial de elementos em um vetor, explicitar o melhor caso, o caso médio e o pior caso.

Solução:

- O melhor caso ocorre quando o elemento buscado está na primeira posição do vetor.
- O caso médio ocorre quando o elemento buscado está na posição central do vetor.
- O pior caso ocorre quando o elemento está na última posição do vetor.

No cálculo de complexidade de tempo de algoritmos, tipicamente é assumido que operações básicas tais como somas, multiplicações, divisões, trocas de elementos em vetores, etc. possuem custo computacional fixo de t_0 unidades de tempo.

A partir dessa hipótese, busca-se obter uma expressão que expresse como essas t_0 unidades de tempo são relacionadas com outros parâmetros, tais como o tamanho da entrada *n*, obtendo-se o custo computacional como função de *n*, para t_0 fixo.

Exemplo: Calcular a complexidade de tempo para o seguinte algoritmo que representa uma função de cópia de array, considerando sua implementação em TypeScript.

```
Função copiar(a: Array [0..n-1] de número): Array [1..n-1] de número
      copia: Array [0..n-1] de número ← [ ]
      Para i:número = 0 .. (n-1)
            copia[i] ← a[i]
      Fim-para
      retorne copia
Fim-função
```

```
function copiar(a: number[]){
    let copia: number[] = [];
    for (let i:number=0; i< a.length; ++i ){
        copia.push(a[i]);
```

```
    }
    return copia;
}
```

Solução:

Note-se, que, para este exemplo, foi considerada uma implementação em uma determinada linguagem de programação (no caso o TypeScript). Isso foi feito para deixar claro quais operações básicas serão consideradas, de acordo com as estruturas presentes na linguagem.

Para se calcular a complexidade de tempo do algoritmo em questão, procede-se com a contabilização do número de operações básicas presentes no algoritmo:

Passo do programa	**Custo computacional (unidades de t_0)**
Inicialização da variável copia[]	$C_1 = t_0$
Inicialização da variável i	$C_2 = t_0$
Atribuição de um valor a cada elemento do array copia[]	$C_3 = nt_0$
Atribuições a i	$C_4 = nt_0$
Incrementos (+) de i	$C_5 = nt_0$
Comparação de i com a.length (inicial + uma para cada passo do laço)	$C_6 = t_0 + nt_0$

A complexidade de tempo é obtida através do somatório dos tempos consumidos por cada um dos passos do programa:

$$C(n, t_0) = \sum C_i$$

$$C(n, t_0) = t_0 + t_0 + nt_0 + nt_0 + nt_0 + nt_0 + t_0$$

Agrupando-se os termos que contém n, chegamos a seguinte expressão:

$$C(n, t_0) = 4nt_0 + 3t_0$$

$$C(n, t_0) = (4n + 3)t_0$$

Se cada passo computacional tem um custo de t_0 segundos, então o custo total será de

$$C(n,t_0) = (4n+3)\ t_0 \text{ seg}$$

Pelo fato de t_0 ser, em geral, considerado constante na análise, omite-se esse valor tanto na análise como na função do custo computacional, que ficaria

$$C(n) = (4n+3)$$

Ou seja, para o exemplo mostrado, o algoritmo é de natureza linear em relação ao tamanho da entrada.

7.3. NOTAÇÃO BIG O E ORDEM

Ainda com relação à análise de complexidade de algoritmos, em geral a análise é conduzida em termos de avaliações de limites superiores para as funções que representam a complexidade do algoritmo como função do tamanho da entrada (i.e., C(n)). Além disso, as entradas de interesse são tipicamente grandes (i.e, valores elevados de n).

Uma maneira de se expressar a complexidade de um algoritmo e que guarda relação com a complexidade temporal é através da utilização da ordem do algoritmo.

Uma função assintoticamente não negativa ***f(n)*** é dita estar na ordem ***O*** (e escreve-se ***f(n)=O(g(n))***) de outra função assintoticamente não negativa ***g(n)*** se ***f(n) < c. g(n)*** para algum *c* positivo e para um *n* arbitrariamente grande.

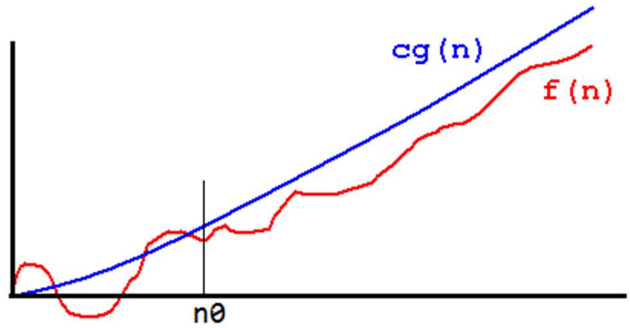


Figura 20. Visualização do conceito: f(n) está na ordem de g(n).

Dessa forma, ao se analisar a complexidade de um algoritmo para uma entrada de tamanho *n*, pode-se avaliar sua ordem em função do tamanho dessa entrada.

Esta avaliação permite não só comparar os algoritmos em termos de eficiência computacional, mas também estimar a o tempo que será consumido.

EXERCÍCIO RESOLVIDO

ER.7.1. Comparar as seguintes funções que medem a complexidade de tempo de dois algoritmos distintos e dizer qual algoritmo é o mais eficiente:

Algoritmo 1 (A1), linear: *C(n)* = 100*n*

Algoritmo 2 (A2) , polinomial quadrático: $C(n) = 2n^2$

Solução: Para pequenos valores de n, as complexidades dos algoritmos podem ser comparadas como segue:

A1: C(1) = 100

A2: C(1) = 2

Aparentemente, o algoritmo A1 consome menos tempo para entradas pequenas. Entretanto, para valores altos de n (n>50), tem-se:

A1: C(51) = 100.51 = 5100

A2: $C(26) = 2 * 26^2 = 5202$

Ou seja, para grandes entradas o algoritmo A1 consome menos tempo do que o algoritmo A2, logo, o algoritmo A1 é mais eficiente.

Note-se que, de acordo com a definição, o algoritmo A1 é da ordem O(n), já o algoritmo A2 é da ordem $O(n^2)$. Considerando que uma função quadrática cresce mais rápido do que uma função linear, pode-se concluir que, quanto maior a entrada, mais rapidamente o tempo necessário para o algoritmo A2 executar irá aumentar, quando comparado ao algoritmo A1. Logo, o algoritmo A1 é mais eficiente e consumirá menos tempo para entradas grandes, a partir de um certo valor de n ($n > 50$). Dessa forma, chega-se a mesma conclusão sem ter que se estipular valores para n, apenas observando-se a “forma” do crescimento da função que representa a ordem do algoritmo.

EXERCÍCIOS

EP.7.1. Esboce em um mesmo gráfico as seguintes funções, representativas das complexidades computacionais de tempo de certos algoritmos e diga qual delas representa o algoritmo mais eficiente e o menos eficiente. Justifique sua resposta.

a) $C(n) = n$

b) $C(n) = n^2$

c) $C(n) = n^3$

d) $C(n) = \log_2(n)$

e) $C(n) = n.\log_2(n)$

EP.7.2. Para qual caso um algoritmo com desempenho de complexidade temporal $C(n) = 81n$ apresenta desempenho idêntico ao de um algoritmo de complexidade temporal $C(n) = n^3$? A partir de qual valor o algoritmo de complexidade cúbica apresenta desempenho inferior?

PC.7.3. Escreva um programa computacional simples que apresente complexidade computacional quadrática. Para diferentes tamanhos das entradas, efetue a medição dos tempos de execução e crie um gráfico cartesiano que mostre tamanho da entrada x tempo de execução. Diga se a curva é coerente com a análise de complexidade do algoritmo.

PC.7.4. Repita o problema PC.7.3. para um algoritmo de complexidade cúbica.

PC.7.5. Repita o problema PC.7.3. para um algoritmo de complexidade $C(n) = \log_2(n)$.

REFERÊNCIAS

[1] GitHub da disciplina:

https://github.com/fabriciogmc/mathematics_for_computer_science

[2] GERSTING, J.L. Fundamentos Matemáticos para a ciência da computação: Matemática Discreta e Suas Aplicações.7 ed. São Paulo: LTC, 2016.

[4] Lee, S. F. Logic – a complete introduction. United Kingdom, Teach Yourself, 2017.

[5] IEZZI, Gelson et al. Fundamentos de matemática elementar. Vols 1-3, 9a Ed. São Paulo, Saraiva, 2019.

[6] TUSSY, Alan; Gustafson, R. David. Elementary and intermediate algebra. 5a Ed, US, Cengage Learning, 2012.

[7] Wikipedia, Lists of mathematics topics. Disponível em <https://en.wikipedia.org/wiki/Lists_of_mathematics_topics> Acesso em 07 de julho de 2023.