Iniciação À Computação

Lógica, Estrutura de Dados, Algoritmos e Problemas Clássicos

Otávio Pavoni

Iniciação À Computação

1a Edição

Otávio Pavoni

Guia para Leitura / Nota do Autor

Algo que me foi ensinado (não diretamente) tanto em salas de aulas quanto nos mutirões solitários de estudos filosóficos e matemáticos que fiz é que nenhum conhecimento no mundo é de difícil discernimento, e uma boa explicação auxilia o processo. Se lhe é difícil: recue, estude mais as bases e tente novamente com um novo olhar. Este livro foi escrito com a máxima dedicação possível, e ele foi escrito também para o meu aprendizado. De aluno (eu) para aluno (você), temos de atingir expectativas o tempo todo. E a pior (pior que me refiro é no sentido de tamanho ou complexidade) das expectativas que devemos cumprir geralmente vem de nós mesmos. Se isso lhe aflige, reflita o seguinte: na frase que repetimos mentalmente sobre o nosso tal dever de cumprir metas, de onde vem tal dever? Certamente você irá estranhar a pergunta. É bem nítido que eu devo fazer X para bem performar em meu trabalho, faculdade, carreira, etc. Porém, é o suficiente? E de onde vem isso? Você estuda computação e está agora com esse livro na mão e/ou na mesa por uma demanda majoritariamente terceira (seu professor) ou você está se dedicando à tal leitura com um propósito além do que as vagas institucionais que títulos acadêmicos podem lhe oferecer? Digo, caso bonificações monetárias não estivessem em jogo, e você pudesse estudar o que quisesse sem se preocupar com o mercado de trabalho, você ainda estaria nessa área? O seu coração ainda arderia quando você estivesse próximo a um computador e demoraria a dormir só pensando nas inúmeras possibilidades de projetos que tal habilidade (conhecimentos técnicos computacionais) lhe forneceria? Claro, você pode estar aqui por uma indicação de um professor, mas creio que ficou evidente a mensagem que pretendi passar: você está aqui mais pelo dever ou pela ânsia imensa pelo conhecimento e auto-aprimoramento? Como eu estava demonstrando, se tal dever ao estudo que você tem é terceirizado e não parte de um fogo ou de um entusiasmo infinito, que transcende toda expectativa e demanda profissional para um nível incogniscível de êxtase, esse livro não é para você e, no geral, a computação também não. Somente a certeza e a possibilidade de você mudar o mundo e resolver problemas de

forma eficiente e intelectual simplesmente manuseando uma máquina extremamente complexa e tecnológica através de códigos, símbolos e lógica deveria ser a razão para todo estudante da ciência da computação viajar no mar de possibilidades e reminiscências oriundas do incrível intelecto humano. É necessário ou ideal que a tecnologia aqui estudada seja para você a Pasárgada de Manuel Bandeira.

Eu tentei o máximo possível ser didático em todos os assuntos aqui abordados – até porque eu também pretendo aprender indefinidamente com esse projeto. Para lê-lo, você deverá ter um computador com acesso à linguagem C, e é necessário que você compreenda os comandos básicos em inglês da mesma e recomendo que não lhe seja difícil compreender tal idioma, visto que muito conteúdo interessante para sua formação está na língua inglesa. Pessoalmente, não sei como seria a minha realidade caso eu não tivesse aprendido inglês cedo. Também será incluido, antes do Capítulo 1, uma breve introdução à linguagem C, juntamente com pequenos exercícios e aspectos principais da linguagem, tendo em vista que esse livro tem a intenção de ser uma iniciação à Computação.

Enquanto eu trabalhava nesse projeto, muitas dificuldades me apareciam e diversas vezes abandonava-o por semanas. No entanto, certamente devo muito de meu crescimento pessoal à ele, já que também aprendi muito com ele, assim como espero que você também o faça.

Essa iniciação conta com 20 capítulos, pretendendo ser uma obra-prima para aqueles que desejam algo denso, maciço e ainda assim ótimo. Espero que goste, e qualquer dúvida ficarei feliz em responder no meu e-mail: *otaviopavonimartins@gmail.com*.

0. Introdução - Cumprindo os Requisitos para a Iniciação

0.1. Aristóteles

Me é cômico ver toda a suposta preparação em lógica de programação que os estudantes têm – isso advém da ausência de conteúdo sobre Aristóteles em tais matérias. Mas qual a razão de tamanha importância de Aristóteles? Ele não é só mais um dos filósofos que tínhamos de decorar no Ensino Médio? De fato, reconhecer atualmente a importância de algum filósofo é complicado, tendo em vista que pouco aprendemos tradicionalmente sobre. E evidentemente, o antigo não teve contato com programação nem computadoras. Entretanto, ele formulou grande parte do que se consistia a lógica até a Modernidade com Leibniz. Aristóteles foi para a lógica o que Turing foi para a computação, o que Steve Jobs foi para a Interface do Usuário (GUI) e o que Bob Dylan foi para o folk e diretamente e indiretamente para o rock psicodélico devido sua influência aos Beatles.

O filósofo grego nasceu por volta do ano 384 antes de Cristo e viveu até o ano 322 a.C. Se a história do mundo ocidental lhe é familiar, esses anos marcam também algo importante para a nossa formação cultural e geopolítica: desenvolvimento, aperfeiçoamento, política e expansão territorial de Alexandre o Grande. Esse, por sua vez, também foi aluno de Aristóteles no ano 343 a.C., assim como você é ou será a partir dos conhecimentos lógicos que serão aqui apresentados. Antes disso, aos 17 anos, Aristóteles sai de sua cidade natal (Estagira na Grécia) e vai para Atenas estudar na Academia de Platão.

Aristóteles foi discípulo de Platão que, por sua vez, fora discípulo de Sócrates. Essa tríade formula a Filosofia ocidental. Na famosa obra Escola de Atenas (1511) podemos ver Aristóteles ao lado de Platão. Claro, tal obra é fictícia, pois nem todos nela presentes (como Diógenes, Heráclito, Parmênides e o próprio Sócrates) eram contemporâneos. No entanto, focando no centro,

encontramos os dois autores. No lado direito temos Aristóteles, o qual está com a palma da mão virada para baixo. No lado esquerdo, temos Platão, com um dedo apontando para cima. Por incrível que possa parecer, tal detalhe representa uma interessante parte da filosofia de ambos. Enquanto Platão focou sua filosofia em um mundo exterior ao nosso e, acima de tudo, perfeito, Aristóteles focou sua filosofia nesse nosso mundo o qual percebemos e vivemos em todo o nosso cotidiano. Além disso, ele é considerado pioneiro na Biologia pela sua classificação de diversas espécies; tem obras incríveis sobre retórica e ética etc. No entanto, vamos focar em sua obra sobre a lógica que, muito usada em debates conhecidos da Idade Média e sendo predominante até a Idade Moderna com Leibniz e Kant, que disse que tudo que havia para saber sobre lógica havia sido descoberto por Aristóteles e sendo claramente útil até hoje e certamente preparou a formulação da filosofia analítica e da lógica simbólica, a qual também iremos ver posteriormente nesse capítulo.

Para se ter noção da grandiosidade e importância da lógica aristotélica, achei interessante aqui mencionar um comentário que encontrei na Enciclopédia de Stanford do historiador lógico Prantl: "qualquer lógico depois de Aristóteles que disse qualquer coisa *nova* estava confuso, foi estúpido ou perverso".

É evidente que citei duas lógicas até agora: a aristotélica (formulada por Aristóteles há milhares de anos e muito usada cronologicamente pelos estóicos, árabes, cristãos medievais, modernistas e todos os pós-modernistas) e a lógica simbólica (formulada principalmente por Frege, filósofo alemão nascido em 1848). Ambas as lógicas se complementam, e certamente é necessário usar de uma lógica simbólica ou formal nesse estudo (devido sua similaridade com a lógica da programação) invés de somente a aristotélica, porém, não se pode falar dessa outra sem se falar primeiro da uma.

0.2. Influência Aristotélica

Pouco teria do método científico, do conhecimento biológico, das artes da retórica e oratória, ética e ciência política se não fosse por

Aristóteles. Seu trabalho inclui a categorização de 500 espécies de animais diferentes, incluindo mamíferos, pássaros, peixes, insetos e outros invertebrados. Deu uma alta e excelente classificação de vida na água e descreve a anatomia interna de mais de 100 animais, e dissecou e estudou cerca de 35 desses.

Na verdade, Aristóteles influenciou o mundo não só científico, mas também religoso. Por muito tempo Aristóteles foi o principal filósofo que os árabes utilizavam, mas na segunda metade da Idade Média quando esses conhecimentos chegaram ao mundo do Ocidente, ele também muito bem influenciou o Cristianismo juntamente com o Santo Tomás de Aquino.

Aqui nos aprofundaremos em sua lógica antes de adentrarmos em lógica da programação e eventualmente em estruturas de dados e algoritmos.

Pouco se parece, mas estruturar bem seus conhecimentos no coração da lógica antes de sequer começar a programar irá lhe salvar muitas dores de cabeça no futuro. Uma metáfora atribuida à Abraham Lincoln apresenta bem isso, da qual ele formula a seguinte frase: "Se eu tivesse seis horas para derrubar uma árvore, passaria as primeiras quatro horas afiando o machado". Isto é, é evidente que o esforço requerido para afiar um machado por quatro horas é bem menor que o esforço requerido para insistir em bater em uma árvore por quatro horas com um machado cego. Saindo da abstração, se você não pular nenhuma etapa e, mesmo com dúvidas e aflições intelectuais, insistir e persistir no seu próprio aprendizado, evidentemente será melhor no futuro quando você ter de colocar isso tudo em prática no seu dia-a-dia, mesmo de maneira indireta (como a Filosofia é posta à pratica, por exemplo).

0.3. Lógica Categórica

Pouco teria do método científico, do conhecimento biológico, das artes da retórica e oratória, ética e ciência política se não fosse por Aristóteles. Aqui nos aprofundaremos em sua lógica antes de adentrarmos em lógica simbólica e posteriormente em lógica da

programação e eventualmente em estruturas de dados e algoritmos.

Não é em totalidade correto chamar em todas as vezes a lógica aristotélica de lógica categórica ou vice-e-versa, mas no intuito que lhes é mostrado tal conhecimento, não vejo como seria errôneo. Esse nome "categoria" provavelmente é oriundo da metafísica de Aristóteles, que muito usa de categoria. No entanto, irei formular essa lógica sem precisar me adentrar em metafísica, tendo em vista que pouco seria do interesse do leitor, mesmo que tal assunto seja, em minha opinião, essencial para a formação de todos.

Para Aristóteles, o mundo é composto de substâncias. Todo Ente tem uma substância. Essa substância é algo individual dos indivíduos e outras espécies, além de objetos inanimados. Tais substâncias estão sujeitas a formulações de propriedades e categorizaões. As substâncias são divididas em duas: as *primárias* e *secundárias*. As primárias consistem basicamente de individuais, isto é, de termos como um homem, um sapo e uma pedra. Fica mais fácil de entender dessa forma: ao andar na natureza, você vê apenas substâncias primárias. Você não vê todas as pedras do mundo ao andar na natureza, e nem em grandes pesquisas você consegue fazer tal proeza (isso será muito discutido por Hume e outros filósofos modernos, caso você tenha interesse), você vê apenas pedras individuais. As substâncias secundárias são os grupos definidos mais abrangentes, como o homem, os sapos e as pedras.

A importância do parágrafo anterior se encontra no encontro da metafísica aristotélica com a sua lógica. A lógica aristotélica é suposta a definir o que existe no mundo, e existe, na filosofia que estamos estudando, cerca de dez categorias das quais são usadas para descrever alguma coisa:

1. Substância (ou essência – o nome "essência" é mais agradável à nossa compreensão devido a sua bagagem literaria lusófona);
2. Quantidade;
3. Qualidade;
4. Relação;

5. Aonde;
6. Quando;
7. Posição;
8. Posse;
9. Ato (ação);
10. Sendo.

Para Aristóteles, tais dez tópicos (categorias) são as formas de se descrever algo na natureza: Na primeira, uma essência é algo único do indíviduo, aquilo que nenhum outro pode ter. Sócrates, por exemplo, é um homem, ou o Sócrates. Essa é a essência de Sócrates. Agora, se fossemos falar "o homem", por exemplo, estaríamos abrangendo uma substância secundária, da qual Sócrates está incluso, mas não é único de si. A primeira é a mais complicada de se compreender com totalidade, já que envolve uma diferenciação metafísica da linguagem. A segunda, terceira, quarta, quinta, sexta são auto-explicatórias. A sétima é interessante e é literalmente o que parece ser: a posição do Ente do qual a sentença está pretendendo descrever, exemplo: o homem está deitado. A oitava descreve posse, isto é, uma sentença que descreve que um Ente tem tal coisa, por exemplo: você tem esse livro ou a posse momentânea dele. A nona implica em um ato que certas sentenças podem descrever. A décima é no "sendo" de algumas sentenças, sendo afetado, por exemplo, uma madeira sendo queimada. Você pode usar mais de uma categoria em uma sentença e optar por usar quais melhor se adequariam ao seu objetivo linguístico, exemplo: um cavalo branco está deitado descansando.

Na lógica aristotélica as declarações mais básicas são proposições, isto é, uma sentença completa (bem declarada categoricamente) que afirma algo. Nem todas as sentenças declaram algo verdadeiro ou falso sobre o mundo, no entanto, estão fora do escopo lógico. Uma proposição é composta de um sujeito, um predicado e um verbo conectivo – o que lógicos usualmente chamam de copula. Na frase "Homens são mortais", "Homens" é o sujeito, o "é" é a copula e "mortais" é o predicado. Simbolicamente isso pode ser expressado assim: S é P. Verifique que, simbolicamente, não há atribuição de conteúdo, isto é, pouco

importa se o sujeito são homens, sapos ou pássaros, e muito menos importa se eles são mortais ou imortais. Repito, simbolicamente, a importância está na construção e não no conteúdo próprio construído. Isso será importante quando chegarmos na lógica simbólica posteriormente.

No entanto, nem toda frase propositiva correta simbolicamente significa que ela é verdadeira. Se falarmos que os pássaros são imortais, é evidente que há um erro aí – não propriamente gramatical, mas evidentemente na forma de que as coisas são. Para Aristóteles, falar que é o que é e falar que não é o que não é, é verdadeiro. Isto é, se eu falar algo que é, de fato, na natureza ou que representa a realidade, é a verdade. E aqui faremos o primeiro paralelo com a computação: assim como os computadores trabalham de forma binária (verdadeiro ou falso), Aristóteles também, sobre sentenças propositivas.

Para Aristóteles, uma proposição não pode ser afirmativa ou negativa ao mesmo tempo. No entanto, tenha em mente sempre que os programas de computador os quais você irá escrever deverão sempre recorrer à binariedade da lógica aristotélica. Ou é, ou faz, ou não é, ou não faz. As crenças que você tem sobre seu projeto não irão modificá-lo ou fazer que ele faça o que você espera que ele faça, assim como crenças pessoais sobre o que acontecerá no futuro não determina se tais proposições são verdadeiras ou falsas. É necessário substituir sua mente achista com o pensamento binário do computador.

O entendimento que você deve ter até agora é que Aristóteles afirma que todas as proposições podem ser expressadas usando a fórmula de "Sujeito *copula* Predicado". Até as mais complexas podem ser resumidas nessa simples formulação. Para estudos filosóficos avançados e talvez para os conhecimentos posteriores aqui, reconhecer termos particulares e universais (globais) é também importante. As particulares referem-se só a um indivíduo, bem como antes eu havia falado sobre as substâncias primárias. De fato, tem essa semelhança, mas também inclui uma construção para a usabilidade da lógica. Os termos particulares equivalem às substâncias primárias no plano lógico. Os termos universais referem-se a muitos indivíduos ou Entes ao mesmo tempo e equivalem às substâncias secundárias

que foram antes apresentadas. Nem todo universal é substância secundária, entretanto, já que existe a diferenciação de acidentes., isto é, aquilo que é uma propriedade de um ser que não necessariamente é senhor de sua existência. Algo muito usado em exemplos de acidentes são as cores: elas são universais, mas não é uma substância. Cores são acidentes. Para simplificar tal diferenciação, substâncias são coisas que existem em si mesmas. A substância é o que porta os acidentes. Acidentes são aquilo que existe em outra coisa, e não em si mesma. A cor branca é universal pois é comum de muitos. Tendo diferenciado termos individuais de termos universais, continuemos.

Tal diferenciação foi de importante menção já que agora iremos iniciar os estudos das quatro diferentes formas de proposições categóricas, cujo nome é oriundo do fato que estabelecem relações entre duas categorias. Todas as proposições possuem quantidade, isto é, se referem à algo em seu todo ou em partes. Também há uma qualidade, a qual será positiva ou negativa, como afirma ou nega tal. Os adjetivos "todo", "nenhum" e "alguns" são os quantitativos que definem a quantidade da proposição. A qualidade é determinada de acordo com o verbo.

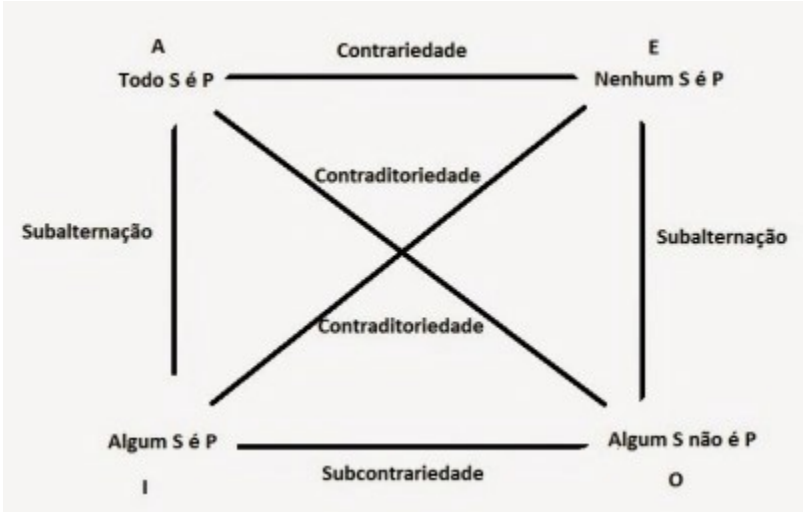
1. Afirmação Universal: Todo S é P (chamado de declaração A, do latim "**A**FFIRMO" ou Eu afirmo);
2. Negação Universal: Nenhum S é P (declaração E, do latim "N**E**GO");
3. Afirmação Particular: Algum S é P (ou alguns são) (chamado de declaração I, do latim "AFF**I**RMO");
4. Negação Particular: Algum S não é P (ou alguns não são) (chamados de declarações O, do latim "NEG**O**").

As proposições que usam o "algum" implica que, no mínimo um Ente do grupo do termo universal é P. Tenha em mente também que, se todo S é P, automaticamente algum S é P, isto é, uma afirmação universal não necessariamente classifica como falso uma afirmação particular, bem pelo contrário. Aristóteles trata proposições com um sujeito particular como uma proposição universal: "Sócrates é mortal", como se "todas as instâncias e partes de Sócrates são mortais". Na próxima parte desse capítulo

isso tudo ficará menos abstrato, tendo em vista que o Quadrado das Oposições bem ilustra como as afirmações e negações se relacionam entre si.

0.4. Quadrado das Oposições

Também chamado de Tábua das Oposições ou Quadrado Lógico. Foi desenvovido na filosofia medieval para ilustrar melhor a lógica aristotélica em relação às proposições.



Olhe as extremidades dessa figura. As vogais representam cada uma das afirmações ou negações que vimos anteriormente. Escolhi essa imagem em específico para representar porque ela também contém a afirmação/negação embaixo da vogal que representa ela.

As linhas diagonais representam contraditoriedade. Perceba que contraditoriedade e contrariedade são palavras parecidas mas diferem em significados. Se uma das extremidades da linha percebida for verdadeira, automaticamente a outra precisa ser falsa. Suponhamos que o A seja uma proposiçaõ que implica que todo sorvete é gelado. Se isso for verdade, a frase da outra extremidade é falsa. Isto é, a proposição que algum sorvete

não é gelado é automaticamente falsa. Perceba aqui a relação entre (A) Todo S é P e (O) Algum S não é P. Se por ventura descobrimos que há um sorvete o qual não é gelado, então a proposição O passa a ser a verdadeira e a A automaticamente entra em contradição e passa a ser falsa. Não tem como todo sorvete ser gelado e ter um que não é, portanto, a proposição que implica a existência desse um que não é é a verdadeira. Se a proposição "alguns sorvetes não são gelados" é verdadeira, é necessariamente falso a proposição "todos os sorvetes são gelados". Mesma lógica se aplica à outra linha diagonal que passa por E e I.

A linha horizontal do topo, as proposições contrárias, entre (A) e (E), não podem ser todas verdadeiras. Se todo S é P, então automaticamente a proposição que nenhum S é P é falsa, ou vice-e-versa, dependendo da veracidade real. Isto é, se há uma proposição que indica que todo super-herói pode voar, e outra que indica que nenhum super-herói pode voar, as duas não podem ser mutualmente verdadeiras, pois são contrárias entre si. Claramente, podem ser ambas falsas caso seja fato que alguns super-heróis voam.

Os subalternos, as linhas verticais, as (A) e (I) e (E) e (O), indicam que a veracidade de uma proposição universal, o "superalterno", requere a verdade de uma segunda proposição particular, o "subalterno". Isto é, se todos os membros de um determinado grupo possuem uma característica X, é evidente que qualquer indivíduo ou grupo menor que pertence a esse grupo maior tem também a característica X. Se todos os homens são mortais, é evidente que homens brancos, negros ou asiáticos são também mortais. Ou seja, se todo S é P, é necessariamente verdadeiro que alguns S sejam P, sejam esses segundos sujeitos pertencentes a um grupo menor, como exemplificado com a etnia dos homens. Perceba também que se é verdade que alguns S são P, não necessariamente todos os S são P, isto é, o caminho inverso (de baixo para cima) não funciona. Somente o topo para baixo é possível. Se o subalterno é falso, o superalterno é também falso.

A linha horizontal inferior, entre (I) e (O), que representa subcontrariedade, implica que duas proposições que são subcontrárias entre si não podem ser mutualmente falsas. Se todos

S são P (A), então é verdade que alguns S são P (I) e é falso que alguns S não são P (O). Isto é, nesse caso, somente a proposição I é verdadeiro. Se nenhum S é P (E), então é verdade que alguns S não são P (O) e é falso que alguns S são P (I). Isto é, nesse outro caso, só O é verdadeira. Se alguns S têm P e outros não têm, então ambas proposições são verdadeiras, então O e I seriam verdadeiros ao mesmo tempo.

Para exemplificar esse último caso, projetei um exemplo para cada um dos casos:

1. Caso 1: Todos S são P. O universal seria, por exemplo, que todos os triangulos têm tres lados. O primeiro particular (I) seria: alguns triângulos têm três lados (o que é verdadeiro). O segundo particular (O) seria: alguns triângulos não têm três lados (o que é falso). Nesse caso, somente uma (I) é verdadeira.
2. Caso 2: Nenhum S é P. O universal seria, por exemplo, que nenhum peixe é mamífero. O primeiro particular (I) seria: alguns peixes são mamíferos (o que é falso). O segundo particular (O) seria: alguns peixes não são mamíferos (o que é verdadeiro). Nesse caso, somente uma (O) é verdadeira.
3. Caso 3: Alguns S são P e alguns S não são P. Particular I seria que alguns alunos estudam de manhã (o que é verdadeiro) e o segundo que alguns alunos não estudam de manhã (o que também é verdadeiro). Nesse caso, ambas (I) e (O) são verdadeiras.

0.5. Leis do Pensamento

Também chamado de Tábua das Oposições ou Quadrado Lógico. Foi desenvovido na filosofia medieval, e é errôneo atribuir os créditos de tal tecnologia somente à Aristóteles. Nos últimos três séculos foi visto que a lógica basicamente se consistia de três axiomas, isto é, de afirmações verdadeiras sem necessidade de prova, algo "autoevidente". Tais axiomas consistiriam as leis do pensamento, cujo nome é auto-explicativo. No entanto, é algo

agregável ao seu futuro conhecimento computacional, tendo em vista que muda, no mínimo um pouco, para melhor, a estruturação do seu raciocínio.

1. Lei da Identidade[1]: Aristóteles escreve na Metafísica sobre algo ser propriamente o algo em questão. "O fato de uma coisa ser ela mesma é [a única] resposta a todas as perguntas do tipo: por que o homem é homem, ou por que o músico é músico". O que Aristóteles gostaria de dizer com isso? Incrivelmente, simplesmente que as coisas são o que são;

2. Lei (ou Princípio) da Não-Contradição[2]: É impossível algo ser e não ser ao mesmo tempo. O mesmo atributo não pode ao mesmo tempo pertencer e não pertencer ao mesmo sujeito. Declarações contraditórias não podem ser ao mesmo tempo verdadeiras. Representado simbolicamente como (não (A e não A) / $\neg(A \wedge \neg A)$[3]);

3. Lei do Terceiro Excluído[4]: Essa lei pode ser simplificadamente explicada como a ideia de que cada proposição deva ser verdadeira ou falsa, não ambas e não nenhuma. Simbolicamente, pode ser representada como (A é verdade ou A é falso / $A \vee \neg A$). Essa lei particular é um avanço frente aos sofistas da época, que acreditavam que a verdade poderia ser relativa ("O homem é a medida de todas as coisas[5]") e então trabalhavam unicamente a retórica de convencimento. Para Aristóteles, toda proposição é verdadeira ou falsa e não pode ser simultaneamente verdadeira e falsa, nem simultaneamente não verdadeira e não falsa. Isto é, a verdade existe, mesmo que, em alguns casos, não possamos conhecê-la inteiramente.

---

1 Metafísica, VII.17.1041a16-18, Ross.
2 Ibid, IV.3.1005b23-24 / Ibid., IV.3.1005b19-20. / Ibid., IV.6.1011b13-14.
3 Não se preocupe ainda com tais símbolos, eles serão aprofundados.
4 De Interpretatione, 9.18a28-29, Ackrill.
5 Protágoras de Abdera.

0.6. Lógica Modal Aristotélica

Na lógica categórica aristotélica, nos deparamos simplesmente com proposições de verdade simples: A, O, etc., sem considerar possibilidade ou necessidade.

- Necessário: □A. Uma proposição é necessária quando não poderia ser de outra forma. Não é contingente[6] e nem depende de condições favorecidas externamente. A necessidade está ligada à essência das coisas, e não aos acidentes (como antes mencionados) ou circunstâncias e intervenções externas, exemplo: Todo triângulo tem três lados. É necessário todo triângulo ter três lados, pois é parte de sua essência. Não é contingente porque um triângulo não poderia não ter três lados e o fato de um triângulo ter três lados não é algo externo, mas em si.

- Possível: ◇A. Algo é possível quando não há impedimento lógico ou natural, mas não é necessário e nem é garantido.

- Impossível: □¬A. Algo é impossível quando há impedimento lógico ou natural.

Regras da lógica modal aristotélica:

1. O necessário implica o possível: □A⟹◊A. Se todo triângulo tem em sua essência ter três lados, isso é necessário, mas também possíve: é realizável e não contraditório;

2. O impossível não pode ser necessário: ¬◊A⟸□¬A. Se algo viola a essência de algo, isto é, por exemplo, um triângulo ter quatro lados, não há como realizá-la;

6 Para Aristóteles, contingente é aquilo ambíguo, verdadeiro ou falso, mas que não é necessário nem impossível, não é determinado pela essência das coisas e depende das circunstâncias. Em suma, o contingente pode ou não ser, sem violar a lógica ou essência das coisas.

3. A possibilidade não implica necessidade: ◊A!⇒□A. Algo ser plausível de acontecer não implica que ele deve acontecer. Exemplo: É totalmente plausível que chova amanhã, mas se não chover, nenhuma regra lógica ou natural seria quebrada.

Outro ponto chave da filosofia de Aristóteles é o conceito de ato e de potênica, os quais não me aprofundarei aqui como fiz com a sua lógica, mas, em suma, uma semente é, em potência, uma árvore, e o mármore é, em potência, uma estátua. O ato é a realização da potência. O ato é a realização plena, a efetivação do que estava em potência.

0.7. Silogismo

Agora chegamos na parte mais importante da lógica aristotélica, o qual muito errôneamente estudantes pesquisam sobre sem antes ter certa base teórica, como a que foi apresentada aqui até agora. Silogismo é basicamente um argumento feito de três proposições categóricas: duas premissas (usualmente destacando a evidência), e uma conclusão (que logicamente segue as premissas). As proposições são compostas de três termos: um sujeito, um predicado e um termo transitório (ou middle term). O sujeito é o sujeito da conclusão. O predicado modifica o sujeito na conclusão, e o termo transitório liga o sujeito e o predicado nas premissas. O sujeito e o predicado aparecem em premissas diferentes, e o termo transitório aparece uma vez em cada premissa. A premissa com o predicado e o termo transitório é chamada de premissa maior, e a premissa com o sujeito e o termo transitório é chamada de premissa menor.

Em prática, o mais famoso certamente é: Todos os homens são mortais. Sócrates é homem. Sócrates é mortal. O termo transitório é H=homem, o sujeito é S=Sócrates e o predicado é P=mortal, podemos representar assim:

- Premissa maior: Todo H é P (Todo homem é mortal);
- Premissa menor: S é H (Sócrates é homem);

- Conclusão: S é P. (Sócrates é mortal).

Isso nos leva a conhecer o silogismo indutivo. O que é propriamente o silogismo? Isso é definitivamente fruto de muito debate acadêmico, mas o que é certo é que o silogismo seria uma forma de raciocinar que começa nas percepções sensoriais (paladar, oufato, tato, etc) de particulares e que termina sendo uma compreensão que possa ser expressa em uma proposição universal. Isto é, repetidamente recebemos exemplos através de nossos sensos (percepções sensoriais corporais e físicas, como visão, oufato, paladar etc já mencionados) que naturalmente nós construímos a um conceito abstrato universal de toda a categoria. Isto é, por exemplo, o fato de nunca termos visto o homem ou o sapo. Vimos um homem aqui e um sapo ali, mas não vimos o todo, é discutivelmente humanamente impossível fazê-lo. No entanto, nós juntamos os padrões e formamos um entendimento geral. A palavra silogismo significa literalmente dedução.

É esse, no entanto, o último tópico de Aristóteles o qual eu gostaria de acrescentar aqui: A diferença entre proposição indutiva e dedutiva. A dedução vai do universal ao particular, enquanto a indução acontece do particular ao universal.

Para simplificar, eis os perfeitos exemplos:

- Dedução: Clássico Silogismo de Sócrates: Homens são mortais, Sócrates é homem, no entanto, Sócrates é mortal.

- Intuição: Este cisne é branco, aquele cisne é branco, todos os cisnes que me lembro de ter visto são brancos, no entanto, todos os cisnes são brancos.

Note que é extremamente complicado de pensar em um silogismo (dedução) intuitiva que seja naturalmente verdadeira, o que naturalmente nos leva às limitações dessa lógica, e Aristóteles já previa a fragilidade disso. Basta, por exemplo, aparecer um cisne negro. Séculos mais tarde, claramente, essa crítica seria discutida e nos levaria à grandes avanços tanto na filosofia quanto na lógica geral. No entanto, para esse livro basta a visão de Aristóteles.

Aristóteles reconhecia uma importância em ambas questões. A dedução era logicamente interessante e necessária, pois era provada "cientificamente" (o bastante para a época), enquanto a indução fornece princípios. Essa distinção é estritamente necessária para o conhecimento verdadeiro, tendo em vista que ela toca no coração do conceito aristotélico de ciência. Para Aristóteles, o conhecimento verdadeiro exige a demonstração dedutiva a partir de universais evidentes.

Isso nos leva inevitalmente às figuras de silogismo, o que seria um desperdício eu ter ditado tanto sobre a lógica aristotélica e não mencioná-las, juntamente com os tais modos válidos de silogismo e as regras gerais de validade. As figuras de silogismo, por sua vez, referem-se à posição do termo transitório[7], que alteram diretamente a estrutura do raciocínio. Essas alternanças são as tais figuras de silogismo.

1. A primeira figura conhecemos, que é quando o termo transitório é sujeito na premissa maior e predicado na premissa menor. Exemplo é, novamente, o do Sócrates: Todo homem é mortal, Sócrates é mortal. Logo, Sócrates é mortal;

2. A segunda é quando o termo transitório aparece como predicado em ambas premissas. Exemplo é:
   - Nenhum peixe é mamífero (P-T);
   - Todo golfinho é mamífero (S-T);
   - Logo, nenhum golfinho é peixe (S-P).

3. A terceira é quando o termo médio aparece como sujeito em ambas premissas. Exemplo é:
   - Todo homem é mortal (T-P);
   - Todo homem é animal (M-S);
   - Logo, algum animal é mortal (S-P) – este seria, no mínimo o homem anteriormente afirmado.

---

7 O termo transitório também pode ser chamado de médio por alguns autores, mas creio que a compreensão geral se favorece usando o nome de termo transitório.

A primeira figura é a mais conhecida e usada, de longe, e têm quatro modos célebres muito utilizados – note que aqui o T substitui o M, já que os fins didáticos terminaram e M é mais usado como "termo médio" e não "termo transitório", apesar de serem sinônimos e eu pessoalmente gostar mais da didática de um:

- Barbara (AAA-1): Todo M é P; Todo S é M; Logo, Todo S é P. Exemplo:
  - Todos os homens são mortais (M-P);
  - Todo grego é homem (S-M);
  - Logo, todo grego é mortal (S-P).

- Celarent (EAE-1): Nenhum M é P; Todo S é M; Logo, Nenhum S é P. Exemplo:
  - Nenhum peixe é mamífero (M-P);
  - Todo salmão é peixe (S-M);
  - Logo, nenhum salmão é mamífero (S-P).

- Darii (AII-1): Todo M é P; Algum S é M; Logo, Algum S é P. Exemplo:
  - Todo homem é mortal (M-P);
  - Algum filósofo é homem (S-M);
  - Logo, algum filósofo é mortal (S-P).

- Ferio (EIO-1): Nenhum M é P; Algum S é M; Logo, Algum S não é P. Exemplo:
  - Nenhum triângulo é círculo (M-P);
  - Alguma figura geométrica é triângulo (S-M);
  - Logo, alguma figura geométrica não é círculo (S-P).

Para distinguir silogismos válidos dos inválidos, Aristóteles estabeleceu regras gerais:

1. O termo médio deve aparecer ao menos uma vez, de forma distribuída;

2. Se ambos os termos são afirmativos, a conclusão será afirmativa;
3. Nada se conclui de duas premissas negativas;
4. De duas premissas particulares nada se conclui.

0.8. História da Lógica Simbólica

A Lógica Simbólica começou, resumidamente, no século 19. George Boole, matemático britânico, filósofo e estudante de lógica decidiu criar uma nova "linguagem" artificial, com o intuito de se evitar dificuldades de vaguidão, equivocações, discursos ambíguos e confusão advinda da significância das emoções do propositor.

1. A primeira coisa a se fazer era entender os elementos dessa "nova linguagem";
2. A segunda coisa era traduzir a linguagem ordinária e usual à notação simbólica;
3. Finalmente, a terceira coisa a se fazer é avaliar argumentos e proposições nessa nova linguagem.

Há quem argumenta que a Lógica Simbólica é a forma lógica mais simples, tendo em vista do tempo que ela economiza nas argumentações.

0.9. Sintaxe e Semântica

Sintaxe é como a gramática em uma linguagem. Estas se referem às regras formais que nos diz o que conta como *well-formed formula (WFF)* ou fórmula bem-formada (FBF). *Se P é uma variável proposicional, então P é uma fórmula bem-formada (FBF).* Uma variável proposicional é um símbolo básico que representa uma proposição simples. Uma fórmula bem-formada é uma expressão construída corretamente segundo as regras lógicas da sintaxe. É o equivalente, na lógica tradicional, a uma frase gramaticalmente correta em uma língua. Exemplos:

- P é uma FBF (porque é uma proposição simples).
- ¬P é uma FBF (porque a negação de uma FBF é uma FBF).
- (P→Q) é uma FBF (porque combina duas FBFs com um conectivo).
- *"→PQ"* não é uma FBF (violação de sintaxe).

A sintaxe na Lógica Simbólica se preocupa, como o nome diz, com os símbolos e como eles podem ser combinados, e não com o que eles significam. No entanto, a Lógica Simbólica não irá se preocupar com a veracidade de uma proposição, e sim se sua sintaxe, semântica e lógica estão corretas.

Elementos de sintaxe:

- Variáveis de proposições (P, Q, R, ...): Estas equivalem a proposições que podem ser verdadeiras ou falsas. São o cerne, o núcleo dessa lógica. As relações dessas variáveis que serão estudadas através dos símbolos. São tratados como tijolos de uma construção.
- Conectivos lógicos: ¬ (não), ∧ (e), ∨ (ou), → (se... então), ↔ (se e somente se).
- Parênteses são usados aqui, assim como na matemática, para agrupar expressões e evitar ambiguidade.

Exemplo de uma sintaxe correta:
(P→Q) ∧ ¬R / Se P, então Q, e não-R.

Toda variável de proposição pode ser assinada com um valor de verdadeiro ou falso. Se P pode ser verdadeiro e Q pode ser falso, então:

Fórmula / Verdadeiro quando
¬P / P é falso
P ∧ Q / Ambos P e Q são verdadeiros
P ∨ Q / Pelo menos um é verdadeiro
P→Q / Falso somente quando P é verdadeiro e Q é falso
P↔Q / Ambos contém a mesma veracidade

Podemos combinar tais peculiaridades e formular tais tipos:

P→Q: "Se está chovendo, permaneço dentro de casa."
P ∧ R: "Está chovendo e eu fico dentro de casa."
¬P: "Não está chovendo."

P, Q ou R não significam nada em específico, eles são como espaços reservados para proposições que podem ser verdadeiras ou falsas.

0.9.1 Exemplos de Sintaxe e Semântica

Vamos exemplificar para tirar um pouco da abstração até agora mantida:

- Exemplo 1:
  - Sintaxe:
    - Considere ¬P (uma negação de uma variável proposicional):
      - "P" é uma FBF (por ser uma variável proposicional, cujas são já, em si, FBF);
      - A regra diz que, se P é FBF, então ¬P também é;
      - Logo, ¬P é FBF.
  - Semântica:
    - Se P é verdadeiro, ¬P é falso; se P é falso, ¬P é verdadeiro:
      - P = "Está chovendo."
      - ¬P = "Não está chovendo."

- Exemplo 2:
  - Sintaxe:
    - Considere P ∧ Q (P e Q):
      - P e Q são FBFs (por serem variáveis proposicionais, cujas são já, em si, FBFs);

        - A regra diz que, se P e Q são FBF, então (P ∧ Q) também é;
        - Logo, (P ∧ Q) é FBF.
    - Semântica:
        - Verdadeira somente se P e Q forem verdadeiros:
            - P = "Está chovendo."
            - Q = "O chão está molhado."
            - P ∧ Q = "Está chovendo *e* o chão está molhado."

- Exemplo 3:
    - Sintaxe:
        - Considere P ∨ ¬Q (P ou não-Q):
            - Q é FBF, logo, ¬Q é FBF;
            - P é FBF;
            - Logo, P ∨ ¬Q é FBF.
    - Semântica:
        - Verdadeira se P for verdadeiro ou Q for falsa (ou ambas):
            - P = "Está chovendo."
            - Q = "O céu está limpo."
            - P ∨ ¬Q = "Está chovendo ou o céu não está limpo."

- Exemplo 4:
    - Sintaxe:
        - Considere (P→Q) (se P então Q):
            - P e Q são FBFs, logo, (P→Q) é FBF.
    - Semântica:
        - Falsa somente se P for verdadeiro e Q for falso:
            - "Se está chovendo, então o chão está molhado."

- Exemplo 5:
    - Sintaxe:
        - Considere (P↔Q) (P é se e somente se Q):
            - P e Q são FBFs, logo, (P↔Q) é FBF.
    - Semântica:

        - Verdadeira quando ambos têm o mesmo valor de verdade:
            - "Está chovendo se e somente se o céu está nublado."

- Exemplo 6:
    - Sintaxe:
        - Considere $[(P \rightarrow Q) \land (\neg R \lor S)] \leftrightarrow \neg(P \land R)$ (o grupo de 'se P então Q' e 'não-R ou S' é equivalente à não-(P e R)):
            - Como esta é muito mais complexa do que as vistas anteriormente, aqui uma forma mais agradável de lê-la: "É verdadeiro que a proposição composta formada por 'se P então Q' e 'ou não R ou S' é logicamente equivalente à proposição 'não é o caso que P e R sejam ambos verdadeiros'.";
            - P, Q, R, S são FBF básicas;
            - $P \rightarrow Q$ é FBF;
            - $\neg R$ é FBF;
            - $(\neg R \lor S)$ é FBF;
            - $(P \rightarrow Q) \land (\neg R \lor S)$ é FBF;
            - $P \land R$ é FBF;
            - $\neg(P \land R)$ é FBF;
            - Logo, toda a expressão é uma FBF complexa e correta.
    - Semântica:
        - Essa expressão em específico afirma que: "A condição 'se P então Q, e (não R ou S)' é equivalente à negação de 'P e R'."
        - Ela é somente verdadeira quando a situação descrita do lado esquerdo e a do lado direito têm o mesmo valor de igualdade. Dica: separe sempre o lado esquerdo do direito, se algum dia você precisar se aprofundar em lógica simbólica, o que, segundo meu palpite, pode ser muito interessante para pavimentar melhor sua estrada até a Ciência da Computação.

0.10. Lógica Proposicional

Esta é uma das formas mais simples e possivelmente didátivas da lógica formal. Por sua vez, ela se concentra nos estudos dos conectivos vistos antes:

- Conectivos lógicos: ¬ (não), ∧ (e), ∨ (ou), → (se... então), ↔ (se e somente se).

É também estudado os valores de verdade das proposições quando unidos pelos tais conectivos lógicos. É notável que, em alguns momentos nos tópicos anteriores aqui estudados, houve um pouco de lógica proposicional. Isso se deve à extrema necessidade e ligação de um tema com o outro, seja com fins didáticos ou propriamente acadêmicos ou lógicos.

A validação de um argumento, por exemplo, depende dos conectivos lógicos usados para unir várias das proposições: se as proposições "se é cachorro, late", "Fred é cachorro" são verdadeiras, então (∴ – *portanto*) é verossímil afirmar que Fred late. Essa parte é parecida com o silogismo aristotélico.

Falamos disso para nos aprofundarmos agora na parte mais importante da lógica proposicional: a tabela verdade. Esta tabela pode demonstrar se uma proposição composta é verdadeira ou falsa baseando-se na verdade ou falsidade das proposições simples nela contidas.

p	q	~p	p . q	p v q	p w q	p → q	p ↔ q
V	V	F	V	V	F	V	V
V	F	F	F	V	V	F	F
F	V	V	F	V	V	V	F
F	F	V	F	F	F	V	V

Está é a forma completa da tabela verdade, mas iremos analisar parte por parte, para que sua compreensão possa ser máxima.

Primeiramente, em ambas primeiras colunas, vemos apenas as variáveis proposicionais, que podem conter diversos dos mais valores. Começando do básico:

q	~p
V	F
F	V

Esta é a forma mais básica de se ver uma negação, pois elas não tem duas proposições juntas. Pode ser lido assim: se P é verdadeiro, não-P é falso. Se P é falso, não-P é verdadeiro.

Se P indica que o céu é azul, não-P indica que o céu não é azul. Se P é verdadeira, não-P é automaticamente falsa.

p	q	p . q
V	V	V
V	F	F
F	V	F
F	F	F

Nosso próximo exemplo é sobre conjunção, isto é, soma. Nesse caso só pode ser verdadeiro se ambas proposições simples for verdadeira. Se qualquer uma das proposições simples for falsa, automaticamente a proposição inteira se torna falsa. Exemplo:

- P: Jabutis são quelônios. (V)
- Q: Gatos são felinos. (V)
- A = P ∨ Q: Jabutis são quelônios e gatos são felinos (V)

- P: Sapos são quelônios. (F)
- Q: Gatos são felinos. (V)
- A = P ∨ Q: Sapos são quelônios e gatos são felinos (F)

- P: Jabutis são quelônios. (V)
- Q: Sapos são felinos. (F)
- A = P ∨ Q: Jabutis são quelônios e sapos são felinos (F)

- P: Sapos são quelônios. (F)
- Q: Peixes são felinos. (F)
- A = P ∨ Q: Sapos são quelônios e peixes são felinos. (F)

p	q	p v q
V	V	V
V	F	V
F	V	V
F	F	F

Creio que esta seja a mais fácil de se aprender: a disjunção inclusiva, ou "ou". Ela requere que qualquer uma das duas proposições simples sejam verdadeiras, logo, só será falsa a proposição composta a qual for formada por duas proposições simples falsas.

Pensando mais claramente, ela quer que P seja verdadeira, caso não, que Q seja. Se P for falsa e Q for verdadeira, tudo bem. Se P for verdadeira e Q for falsa, tudo bem. Se ambas forem

verdadeiras, tudo bem. Só não estará tudo bem se ambas forem falsas.

- P: Jabutis são quelônios. (V)
- Q: Gatos são felinos. (V)
- A = P . Q: Jabutis são quelônios ou gatos são felinos (V)

- P: Sapos são quelônios. (F)
- Q: Gatos são felinos. (V)
- A = P . Q: Sapos são quelônios ou gatos são felinos (V)

- P: Jabutis são quelônios. (V)
- Q: Sapos são felinos. (F)
- A = P . Q: Jabutis são quelônios ou sapos são felinos (V)

- P: Sapos são quelônios. (F)
- Q: Peixes são felinos. (F)
- A = P . Q: Sapos são quelônios ou peixes são felinos. (F)

p	q	p w q
V	V	F
V	F	V
F	V	V
F	F	F

Esta se chama disjunção exclusiva. Esta só será verdadeira quando ambas proposições serem opostas, isto é, uma for verdadeira e a outra for falsa. Se ambas proposições simples forem verdadeiras ou se ambas proposições simples forem falsas, automaticamente a proposição composta será também falsa.

Os exemplos dessa parte são interessantes, e o meu preferido é:

- P: O homem está vivo. (V)
- Q: O homem está morto. (V)
- P w Q: O homem está vivo ou morto. (F)

Se alguém falar que ele está vivo e que está morto, ou que está morto e está vivo, entrará no princípio da não-contradição, já visto aqui anteriormente.

p	q	p → q
V	V	V
V	F	F
F	V	V
F	F	V

Isto é implicação. A implicação propõe uma condição: "Se P, então Q". Só será falsa se a primeira proposição simples for verdadeira e a segunda proposição simples for falsa, isto é, se é uma condição, a segunda não pode deixar de acontecer se a primeira acontecer. Nos casos restante, a proposição composta é verdadeira.

- P: Eu vou comer. (V)
- Q: Vou ficar forte. (V)
- A = P→Q: Se eu comer, vou ficar forte. (V)

Eis um exemplo para o único caso possível de falsidade na implicação:

- P: Eu vou comer. (F)
- Q: Vou ficar forte. (V)
- A = P→Q: Se eu não comer, vou ficar forte. (F)

p	q	p ↔ q
V	V	V
V	F	F
F	V	F
F	F	V

Esta se chama equivalência (se e somente se). Só será verdadeira quando ambas proposições simples forem idênticas na veracidade, seja para verdadeiro ou para o falso.

- P: Eu vou dormir.
- Q: Eu vou escovar os dentes.
- A = P↔Q: Eu vou dormir se e somente se eu escovar os dentes.

Eu posso ir dormir quando eu estiver escovado os dentes ou não ir dormir enquanto eu não tenha escovado os dentes, mas jamais ir dormir sem escová-los ou escová-los sem dormir, nesse caso. Claro, a realidade abrange fatores muito mais complexos, é necessário que sua cabeça se adapte à esse tipo de realidade didática para primeiro compreender o básico. É verdadeiro a proposição composta se as duas proposições simples forem verdadeiras ou se ambas proposições simples forem falsas. Tente pensar em outras duas proposições simples que possam servir para o exemplo de equivalência.

0.11. Lógica de Primeira Ordem (FOL)

Também chamada de lógica de predicados. É uma extensão das proposições, criada para expressar conceitos mais complexos sobre objetos, suas propriedades e relações entre eles. Na lógica proposicional, a qual estudamos, exemplo: "Se chove, o chão está molhado", lidamos apenas com proposições simples e conectivos. Não sabemos o que é "chuva" ou "chão". Tudo é tratado como átomos ou blocos inteiros inmodificáveis ou inqualificáveis. A FOL pretende explorar as estruturas internas das proposições, se permitindo falar sobre suas coisas e propriedades. Esta conta com alguns elementos básicos, que são estes que serão indicados nesse esquemas com fins didáticos:

Categoria	Função	Exemplos	Explicação
Constantes	Referem-se a **objetos específicos** do domínio (entidades particulares).	a, b, c, João, Maria	Cada constante representa um indivíduo concreto, como uma pessoa, cidade ou número específico. Ex: joao → refere-se exatamente a João, um indivíduo do domínio (sim, mesmo domínio visto e aprendido em matemática no seu ensino médio em matrizes).

Categoria	Função	Exemplos	Explicação
Variáveis	Representam **qualquer elemento** do domínio.	x, y, z	São usadas em fórmulas que expressam generalizações. Ex: Homem(x) → "x é homem" (sem dizer quem x é).
Predicados	Expressam **propriedades** ou **relações** entre objetos.	Homem(x), MaiorQue(x, y)	São como "funções booleanas": retornam verdadeiro ou falso. Ex: Homem(joao) → verdadeiro, se João for homem. MaiorQue(joao, maria) → verdadeiro se João for mais velho (ou maior) que Maria.
Funções	Produzem **um objeto** a partir de outros objetos.	PaiDe(x), Soma(x, y)	Diferem dos predicados porque o resultado **é um objeto**, não um valor lógico. Ex: PaiDe(joao) → "o pai de João" (um indivíduo). Soma(2,3) → 5.

Categoria	Função	Exemplos	Explicação
Conectivos lógicos	Ligam fórmulas para formar proposições compostas.	∧ (e), ∨ (ou), ¬ (não), → (se...então), ↔ (se e somente se)	Funcionam como na lógica proposicional. Ex: Homem(x) ∧ Pai(x) → "x é homem e x é pai". Homem(x) → Mortal(x) → "se x é homem, então x é mortal".
Quantificadores	Permitem **generalizar** sobre os elementos do domínio.	∀ (para todo), ∃ (existe)	Introduzem variáveis ligadas. Ex: ∀x (Homem(x) → Mortal(x)) → "todo homem é mortal". ∃x (Mulher(x) ∧ Médica(x)) → "existe uma mulher que é médica".

Exemplos mais caprichados:

- Todo homem é mortal: ∀x(Homem(x)→Mortal(x));
  - Leitura formal/tradicional: Para todo x, se x é homem, então x é mortal.

- Existe alguém que é filósofo: ∃x(Filosofo(x));
  - Leitura formal: Existe pelo menos um x que é filósofo.

- Se alguém é pai, ele tem um filho: ∀x(Pai(x)→∃y(Filho(y,x)))

- Leitura formal: Para todo x, se x é pai, então existe algum y tal que y é filho de x.

Fórmulas atômicas são os tais Pai(x), Homem(x) ou MaiorQue(x, y) que lhe foram apresentados. Esta é a aplicação de um predicado a um ou mais termos. Toda fórmula atômica é uma FBF (Fórmula Bem Formada). Se φ e ψ são FBFs, então:

- ¬φ é uma FBF
- (φ ∧ ψ), (φ ∨ ψ), (φ→ψ), (φ↔ψ) são FBFs

Se φ é uma FBF e x é uma variável, então:

- ∀xφ (para todo x, φ) e ∃xφ (existe x tal que) são FBFs

Exemplo: ∀x(Homem(x)→Mortal(x))

0.12. Sistemas de Prova

Um sistema de prova é um conjunto de regras formais que nos permite deduzir fórmulas novas a partir de outras, com base apenas na forma lógica. Ele nos diz como raciocinar corretamente dentro da lógica, sem precisar olhar e/ou analisar o "mundo real" tal qual ele é.

1. Sistema de axiomas (dedução de Hilbert): baseia-se em axiomas (sempre verdadeiros) fixos e imóveis e em regras de inferência simples, principalmente o *Monus Ponens*, o qual ainda veremos. Exemplo dos axiomas fixos:

    - A1: φ→(ψ→φ)→("Se φ é verdadeiro, então, se ψ for verdadeiro, φ continua verdadeiro, isto é, a verdade de φ não depende de ψ, exemplo: o fato do céu ser azul não interfere na veracidade de estar chovendo"). A utilidade desta é encontrada na vez de garantir que proposições verdadeiras permanecem verdadeiras mesmo dentro de implicações encadeadas.

- A2: $(\varphi \rightarrow (\psi \rightarrow \chi)) \rightarrow ((\varphi \rightarrow \psi) \rightarrow (\varphi \rightarrow \chi))$ ("Se de φ decorre que ψ implica χ, então, se de φ decorre ψ, também decorre χ."). Chamado axioma da transitividade lógica da implicação. Exemplo:

  - φ: “Estudo.”
  - ψ: “Aprendo.”
  - χ: “Passo na prova.”

  > "Se estudar implica que aprender leva a passar na prova, então, se estudar implica aprender, estudar também implica passar na prova."

- A3: $(\neg\psi \rightarrow \neg\varphi) \rightarrow (\varphi \rightarrow \psi)$ ("Se a negação de ψ implica a negação de φ, então φ implica ψ."). Chamado de princípio da contraposição, também muito usado em lógica: duas proposições são logicamente equivalentes às suas contrárias. Se negar ψ leva a negar φ, então se φ for verdadeiro, ψ também deve ser.

  - φ: “Está chovendo.”
  - ψ: “O chão está molhado.”

  > Se é verdade que **“se o chão não está molhado, então não está chovendo”**, então também é verdade que **“se está chovendo, o chão está molhado”**.

- Modus Ponens:
- Se φ e $\varphi \rightarrow \psi$ são teoremas, então ψ também é teorema. Isto é, Modus Ponens é uma regra lógica que lhe permite tirar conclusão a partir de duas premissas. A fórmula usualmente vemos com (ψ) no denominador e $\varphi, \varphi \rightarrow \psi$ no numerador. Claro, não são estas as palavras usadas em lógica, mas como não vamos nos aprofundar nisso aqui, usarei essas palavras com fins didáticos. No entanto, recomendo você se aprofundar

nisso caso seu fim seja a ser um bom profissional da Computação, que se sobressai perante os outros conforme seu conhecimento. Outra forma de expressar é:

$$(\varphi \rightarrow \psi), \varphi \vdash \psi$$

2. Cálculo Sequente (Gentzen): Também pouco intuitivo para humanos, como os que foram vistos anteriormente. No entanto, serve para treinar seu raciocínio lógico. Estes usam sequentes: $\Gamma \vdash \Delta$: "de todas as fórmulas em Γ, podemos provar pelo menos uma em Δ". Muito usado para estudos teóricos.

Um bom sistema de prova deve satisfazer duas condições fundamentais:

Propriedade	Significado
Correção (Soundness)	Tudo o que pode ser provado no sistema é logicamente verdadeiro (válido).
Completude (Completeness)	Tudo o que é logicamente verdadeiro pode ser provado no sistema.

Em 1930, Gödel fez o incrível feito de provar a Completude da LPO: "Toda fórmula verdadeira em todas as interpretações pode ser provada dentro do sistema formal".

0.12.1. Dedução Natural

> "Se todo homem é mortal e Sócrates é homem, então Sócrates é mortal."

Esta é possivelmente a máxima da lógica Ocidental. Vamos comprovar isso utilizando um pouco da dedução natural?

$$\forall x(Homem(x) \rightarrow Mortal(x)), Homem(socrates) \vdash Mortal(socrates)$$

Passo-a-passo:

1. De $\forall x(Homem(x) \rightarrow Mortal(x))$, pela eliminação do quantificador universal:
   $Homem(socrates) \rightarrow Mortal(socrates)$
2. É evidente que temos Homem(socrates) (premissa).
3. Ao aplicar o visto Modus Ponens, obtemos: Mortal(socrates).

0.12.2. Cálculo de Sequentes

$$P, P \rightarrow Q \vdash Q$$

As premissas são P e P→Q, conclusão: Q. Significado: "Se P e P→Q são verdadeiros, então Q também é verdadeiro".

Optei por dar um exemplo antes da forma geral, esta é: $\Gamma \vdash \Delta$.

- Γ = conjunto (ou lista) de fórmulas **premissas**.
- Δ = conjunto (ou lista) de fórmulas **conclusões possíveis**.
- "Se todas as fórmulas em Γ forem verdadeiras, então pelo menos uma fórmula em Δ é verdadeira."

0.12.3. Princípio da Resolução

Para provar que uma fórmula F é verdadeira, você assume o **contrário** (¬F) e mostra que isso **leva a uma contradição**. Exemplo, vamos provar: $(P \vee Q) \wedge (\neg P \vee R) \wedge (\neg Q \vee \neg R) \vdash ?$
Passos:

1. Negar a conclusão (supondo que a conclusão seja R): ¬R
2. Transformar tudo em cláusulas (FNC): Fazendo-o, fica: C1:P ∨ QC2:¬P ∨ RC3:¬Q ∨ ¬RC4:¬R

3. Aplicar resolução:

- Resolva C2:$\neg P \vee R$ com C4:$\neg R \rightarrow \neg P$
- Resolva C1:$P \vee Q$ com $\neg P \rightarrow Q$
- Resolva C3:$\neg Q \vee \neg R$ com $Q \rightarrow \neg R$
- Resolva $\neg R$ com C4:$\neg R \rightarrow$ cláusula vazia $\perp$

No fim, você encontrará uma contradição, eis o motivo:

- Começamos assumindo as quatro cláusulas iniciais.
- Usamos resolução entre $\neg P \vee R$ e $\neg R$ para deduzir que $\neg P$ deve ser verdadeiro.
- Combinamos isso com $P \vee Q$ para concluir que Q deve ser verdadeiro.
- Então combinamos Q com $\neg Q \vee \neg R$ para concluir que $\neg R$ é verdadeiro.
- Finalmente, $\neg R$ já estava entre as premissas, e combinando ambos chegamos à **cláusula vazia**, mostrando que há uma contradição.

Sim, entendo perfeitamente que o nível de abstração agora aumentou, e é por isso que vou me abster de maiores aprofundamentos, até porque ainda carece de meu conhecimento. No entanto, acabamos a parte teórica de lógica, ainda que eu recomende que você estude metateoremas e o teorema de Gödel. Como dito, isso tudo não é estudado nem aprofundado nas universidades, mas com certeza estaríamos formando outro nível de profissionais caso fosse. Sinceramente, não poder me aprofundar totalmente nesse assunto me causa um pouco de frustração, mas nada que me impeça de continuar com esse projeto. Na realidade, pouco ainda hei de falar sobre, já que, em minha concepção hiper-fundamentalista, é de suma importância

saber pelo menos do que se trata cada um desses assuntos. Ficará aqui na nota de rodapé[8] um pouco sobre cada um deles.

8 Metateoremas e o Teorema de Gödel: "meta" significa acima de ou sobre algo. É evidente que, se você já estudou filosofia, tal conceito lhe é familiar com a tal "metafísica" de Aristóteles. Na realidade, é interessante a história: os compiladores da obra de Aristóteles, após compilares a Física, se depararam com um compilado de livros que falavam sobre estudos das causas primeiras, e nomearam-na de Metafísica, também porque tais assuntos se encontraram acima de ou posterior à Física tradicional aristotélica. Um metateorema é um teorema sobre teoremas. Um metateorema fala de fora do sistema, analisando principalmente suas capacidades e limitações. Por exemplo, o Teorema da Completude de Gödel, que veremos adiante, é um metateorema sobre a lógica de primeira ordem, isto é, um estudo sobre limites e capacidades de tal. Enquanto teoremas em si falam sobre sistemas utilizando a própria linguagem de tal, um metateorema utiliza a linguagem da lógica e da matemática para falar sobre esse sistema. Um metateorema usualmente usa técnicas que não estão disponíveis dentro do sistema que está sendo analisado. Gödel afirma que um enunciado na lógica de primeira ordem é universalmente válido se, e somente se, ele é demonstrável.

O Primeiro Teorema da Incompletude de Gödel afirma que qualquer sistema formal axiomático que seja suficientemente poderoso para expressar a aritmética básica dos números naturais, se for consistente, então é incompleto. Isto é, existem enunciados (afirmações ou fórmulas bem formuladas) no sistema que são indecidíveis. Nem o enunciado nem sua negação podem ser provados a partir dos axiomas do sistema. Gödel fala sobre uma afirmação G, esta é: "está afirmação não pode ser provada dentro deste sistema". Se G for falsa, então ela pode ser provada. Mas um sistema que prova uma falsidade é inconsistente, ora uma contradição! Se G for verdadeira, então ela não pode ser provada (que é exatamente o que é afirmado). Portanto, existe aí uma afirmação verdadeira que o sistema é incapaz de provar.

Sendo extremamente e didático, basicamente, Gödel demonstra que nenhum sistema de regras matemático que seja suficientemente poderoso pode ser ao mesmo tempo completo e consistente. Isto é, qualquer conjuntos de regras matemáticas que seja bom o bastante para fazer contas básicas sempre terá perguntas que propriamente não consegue

0.13. Aplicando Lógica à Computação

Como vimos, a lógica é um problema filosófico (mesmo que hoje em dia se considere que ele está além da filosofia) que é oriundo da necessidade de estudar as condições essenciais para a constituição do conhecimento, e fixa as regras de seu funcionamento correto. A lógica tratará sempre de estudar as condições que são fundamentais que possibilitarão os tipos de discernimento e de estabelecer as regras (normas) de seu funcionamento correto. A lógica é, em suma, sobre o funcionamento do conhecimento[9]. A partir de agora iremos aplicar toda a nossa bagagem lógica à computação.

A lógica proposicional do qual gastamos muitas páginas discernindo sobre, será muito útil para sua formação, tendo em vista que ela é essencial para circuitos digitais, otimização de código, IA e verificação de programas.

---

responder, e você não pode consertar isso sem criar contradições, como demonstrado acima as novas contradições criadas.

Para melhor representar isso e finalizar de uma vez por todas toda a lógica aqui apresentada, e de forma entusiástica, vou apresentar-vos a Analogia do Jogo de Tabuleiro. Um jogo de tabuleiro é, por sua vez, complexo. As regras do jogo seriam como os axiomas matemáticos, os movimentos permitidos seriam as demonstrações matemáticas, as posições no tabuleiro seriam os teoremas matemáticos, etc. No entanto, sempre haverão posições no tabuleiro que são perfeitamente válidas, mas que não podem ser alcançadas seguindo as regras do jogo. Ou seja, existem verdades matemáticas que não podem ser provadas dentro do sistema. Se você tentar modificar o jogo para alcançar todas as posições, eventualmente você será obrigado a adicionar regras que se contradizem. Se você tentar provar tudo, vai acabar podendo provar coisas contraditórias, como 1 = 2.

Em suma, esta afirmação não pode ser demonstrada como verdadeira dentro deste sistema matemático. Espero que esse leve toque de Gödel tenha lhe aguçado o gosto pela lógica geral, mas que, claramente, esta parte em específico, serve apenas como "food for thought" para exercitar seu cérebro para problemas matemáticos mais complexos vistos dentro da Ciência da Computação. Abstenho-me de continuar sobre lógica agora, partiremos para a computação.

9 MONDIN, Battista. Introdução à Filosofia. Trad. João Bosco Penido Burnier. 7 ed. São Paulo: Paulus, 1980.

A lógica de predicados é a base da lógica matemática e da IA simbólica. Ela pode servir para definir de maneira formal as linguagens de programação e sistemas de prova (testes) automático. Outra coisa, caso não saibas, quando você consulta utilizando SQL, você está essencialmente utilizando lógica de predicados. A tabela de verdade também será essencial e muito em seu cotidiano programador.

Sistemas de dedução e as provas formais são para provar algo. A lógica não é somente apenas sobre verdade, mas principalmente sobre como provar algo. Existem, como vimos, as provas dedutivas, como o Modus Ponens, que é importante para formular bases de mecanismos de inferência em IA. Fundamental também para compiladores e verificadores de tipo, e essencial para provar correção de programas.

A álgebra boolena vem de George Boole (daí o nome em homenagem á Boole) transforma proposições em expressões algébricas com valores 0 e 1. Isto é útil de se entender para quem pretende se aprofundar em hardware ou em softwares de sistemas lógicos binários. $(P \land Q) \lor \neg R$ pode ser representado em circuitos digitais como as portas lógicas AND, OR e NOT.

Algo que vimos pouquíssimo, mas ainda o vimos indiretamente, é a teoria dos conjuntos. Exemplo:

- $A = \{1, 2, 3\}$, $B = \{2, 3, 4\}$
- $A \cap B = \{2, 3\}$

Isso é matemática básica que me recordo de ter aprendido no colégio. No entanto, caso você ainda não o saiba, recomendo pesquisar por conta própria em livros de autores renomados sobre o assunto para você não ficar para trás no desenvolvimento de software. Essa é útil para a modelagem matemática de dados e ela fundamentaliza a semântica formal de linguagens de programação. Também oferece suporte para bancos de dados relacionais e teoria da computabilidade.

Espero que se recorde da lógica modal a qual falamos, aquelas que lida com possibilidades, necessidade e tempo. A modal usa operadores como $\Diamond$ (“é possível que”) e $\Box$ (“é necessário que”). A temporal usa operadores como G (sempre), F

(eventualmente) e X (no próximo caso/estado). Também útil para IA e verificação de sistemas reativos e concorrentes, como redes.

A teoria da computabilidade, por sua vez, estuda o que é impossível de computar ou o que é possível de ser computado, usando fundamentos lógicos. Os principais conceitos desta são Máquina de Turing, Funções Recursivas e Problemas Indecidíveis, todos esses serão abordados em breve. Essa teoria é importante porque define os limites da computação e o que pode ser resolvido ou não por um computador.

Lógica na verificação de programas é demasiadamente oriunda à Lógica de Hoare, muito usada para provar a correção de programas. {P} C {Q} onde P é a pré-condição, C é o comando/programa e Q é a pós condição. { x > 0 } x := x - 1 { x ≥ 0 } é um exemplo prático, que pode ser lido como:

- Pré-condição: { x > 0}: Antes da execução, sabemos que x é maior que zero.
- Condição: x := x – 1: O valor de x é atualizado para x – 1, ou seja, subtraímos 1 de x.
- Pós-condição: { x >= 0}: Depois da execução, queremos garantir que x seja maior ou igual a zero.

Este programa serviria para preservar a validade lógica da propriedade cuja "x não se torna negativo".

0.14. Lógica de Programação e Fundamentos da Computação

Como vimos, a lógica da computação ainda nos parece abstrata. Tudo ficará mais claro quando começarmos a escrever código e vermos de forma aplicada tudo o que estudamos até agora, na lógica da programação. Eu escolhi a linguagem C para iniciar-te porque ela me é a mais interessante, pois ela é poderosa e simples de se compreender, comparada com Assembly (cuja eu claramente lhe recomendo aprender também, caso tenhas tempo).

0.14.1. Fundamentos da Computação

No 0.14.1. lhe será apresentado os fundamentos da computação, desde o que é um Kernel até as possibilidades principais da linguagem de programação C, priorizando lhe ensinar Ciência da Computação da forma mais pura e livre de distrações possível.

0.14.1.1. Introdução à Computação (Conceitos e Partes Técnicas)

Irei começar explicando o básico, a diferença entre Hardware e Software. Evidentemente irei me aprofundar mais nos fundamentos, mas gostaria de diferenciar tais duas coisas antes de me aprofundar pouco em Hardware e muito mas em Software. Esta parte (0.14.1. e adiante) será uma transcrição em Português do curso gratuito de Harvard chamado CS50 (Computer Science 50). Computadores têm dois digitos, cujos podem ser representados pelo termo técnico o qual você provavelmente já ouviu sobre, o tal "bit". "bit" significa "binary digit" ou "digito binário". Nós, humanos, usamos o sistema decimal (de 0 até 9) enquanto computadores usam o sistema binário. Binário é ligado ou desligado, corrente ou sem corrente, 0s e 1s. No entanto, um "bit" é um zero (0) ou um um (1).

Computadores representam zeros e uns de forma física. Uma analogia possível é, para se representar o zero, você mantém uma lâmpada desligada. Quando for para representá-la acesa, você liga-a. Computadores de fato fazem isso, mas com milhões desses tipos. O nome desses são transistores. Ok, se de fato entendemos que uma "lâmpada" pode contar de 0 a 1, como podemos contar de 1 a 2 e assim respectivamente? Com mais "lâmpadas", claramente. Seu computador têm milhões destes "switches".

Decimal	Binário
0	0
1	1
2	10
3	11
4	100
5	101
6	110
7	111
8	1000
9	1001
10	1010
11	1011
12	1100
13	1101
14	1110
15	1111
16	10000
17	10001
18	10010
19	10011
20	10100

Na tabela acima, contamos de 0 até 20 em números binários. Antes de ler a explicação de como tal contagem funciona, recomendo que faça uma análise da tabela e tente identificar padrões e então tente você mesmo contar até 30 em binário.

O 0 é 0 e o 1 é 1. O 2 é 10 porque o 1 que ficou à direita passará à esquerda. O 3 é 11 porque outro 1 entrará agora no local que ficou o "0" anterior".

Quando um número passa de 0 a 1, ele não pode passar de 1 para dois, tendo como única alternativa avançar uma casa para a esquerda com o número "1". Pense na contagem binária como uma contagem a qual tem como objetivo passar todos os zeros para um, respeitando a ordem, transformando em 0 as casas das

quais antes eram 1 e foram para a esquerda. Tente agora contar de 0 até 30 em binário.

0.14.1.2. Aprofundamento à Kernel

A completa compreensão de como binários funcionam por si só não é completa, pois esta necessita de uma ponte para o kernel.

Como dizemos, temos os bits. Temos também os bytes, que são 8 bits cada. Exemplo: 01000001 = 65 em decimal = "A" em ASCII.

As instruções em binários são extremamente low-levels e não cabe a uma Iniciação tamanho aprofundamento, mas eis aqui um exemplo de instrução em binário:

```
10110000 01100001 ; MOV AL, 61h (em x86)
```

O processador vai ler as intruções binárias da memória, executar as operações em registradores e trabalhar diretamente com o binário.

A linguagem de programação Assembly é a que eu pretendia usar nesta iniciação, mas por ofícios eu preferi usar C. No entanto, ao ler um programa em Assembly, literalmente não há maneira mais eficiente de se reconhecer as funções dadas pelo código para seu computador, e tudo é facilmente traduzido para o binário.

Exemplo de código em Assembly:

```
section .data
    msg db 'Hello', 0

section .text
    mov eax, 4      ; sys_write
    mov ebx, 1      ; stdout
    mov ecx, msg    ; mensagem
    mov edx, 5      ; tamanho
    int 0x80        ; chamada do kernel
```

Código Fonte → Assembly → Objeto → Executável é o caminho que faz.

Em suma, o processador (hardware) é como se fosse um tradutor. Todo código, seja em Python ou Java, deve ser eventualmente reduzido em uma série de zeros e uns que o processador possa decodificar e executar. A ponte entre o homem e a máquina são as linguagens de programação. Algumas, como demonstradas anteriormente, permite-lhe uma melhor visualização do que está acontecendo em seu processador. Estas são chamadas de linguagem low-levels. A linguagem "high-level" mais fácil de ler o que está acontecendo em seu processador é a C, a qual nos adentraremos aqui nessas aulas.

A compilação é o processo de grande tradução, o qual nosso computador irá pegar nossa linguagem "humanizada" para linguagem "computadorizada" (somente zeros e uns). Nesse processo, o compilador analisa a sintaxe e semântica do código fonte, verifica a sua correção lógica e, em seguida, gera código assembly equivalente específico para a arquitetura do processador alvo. O código assembly é processado por um assembler, que realizará a tradução final para o código da máquina binário. O resultado é um arquivo objeto contendo instruções binárias, mas ainda não é completamente autônomo, pois pode conter referências a funções externas, bibliotecas do sistema ou outros módulos de código. O linker aparecerá para resolver justamente estas dependências, combinando múltiplos arquivos objeto e bibliotecas em um único executável. O kernel é o grande mediador: quando um usuário executa um programa através do terminal ou GUI (Interface Gráfica), irá se iniciar uma complexa interação com o kernel do sistema operacional.

O kernel é o componente central de um SO (Sistema Operacional) que agirá como intermediador entre o hardware do computador e os aplicativos de software. Ele é o primeiro programa que o computador carrega quando inicia. Imagine um grande prédio de uma respectiva empresa. Os funcionários em seus escritórios representam os programas dos usuários, cada um com sua respectiva tarefa em específico. O kernel seria como uma gerência: ele não realiza o trabalho diretamente, mas garante que

todos tenham os recursos necessários. Evita também conflitos e mantém a segurança do prédio inteiro.

O kernel é, portanto, uma "ponte", que liga os programas de seu computador (como os seus joguinhos) e o seu hardware físico.

0.14.1.3. Sistemas Operacionais e Arquivos

O SO (Windows, MacOS, GNU-Linux-based distros, FreeBSD etc.) é a administração entre os softwares e o hardware. Ele inclui o kernel, tanto que muitos confundem o Linux como sistema operacional. O Linux é somente um kernel, que vários SOs possuem. O SO gerencia o processador, a memória RAM, dispositivos de entrada e saída, oferece uma GUI, etc..

SOs possuem o kernel (já explicado), uma GUI (interface gráfica) e uma CLI (interface de linha de comando, como o terminal e o cmd), system calls (que são pontes entre os programas e o kernel e utilitários gerais.

Os arquivos organizam os dados no seu disco. O seu funcionamento e organização varia de acordo com o SO, principalmente no Windows. No entanto, todos focam em organizar os arquivos do sistema em diretórios.

0.14.1.4. História das Linguagens de Programação

Me é cômico ver toda a suposta preparação em lógica de programação que os estudantes têm – isso advém da ausência de conteúdo sobre Aristóteles em

0.14.1.5. Máquina de Turing

Me é cômico ver toda a suposta preparação em lógica de programação que os estudantes têm – isso advém da ausência de conteúdo sobre Aristóteles em

0.14.1.6. Funções Recursivas

Me é cômico ver toda a suposta preparação em lógica de programação que os estudantes têm – isso advém da ausência de conteúdo sobre Aristóteles em

0.14.1.7. Problemas Indecidíveis

Me é cômico ver toda a suposta preparação em lógica de programação que os estudantes têm – isso advém da ausência de conteúdo sobre Aristóteles em

0.14.2. Linguagem C

Me é cômico ver toda a suposta preparação em lógica de programação que os estudantes têm – isso advém da ausência de conteúdo sobre Aristóteles em

0.14.3. Estruturas de Controle

Me é cômico ver toda a suposta preparação em lógica de programação que os estudantes têm – isso advém da ausência de conteúdo sobre Aristóteles em

0.14.4. Estruturas de Dados

Me é cômico ver toda a suposta preparação em lógica de programação que os estudantes têm – isso advém da ausência de conteúdo sobre Aristóteles em

0.14.5. Modularização

Me é cômico ver toda a suposta preparação em lógica de programação que os estudantes têm – isso advém da ausência de conteúdo sobre Aristóteles em

0.14.6. Pilhas (stack)

Me é cômico ver toda a suposta preparação em lógica de programação que os estudantes têm – isso advém da ausência de conteúdo sobre Aristóteles em

0.14.7. Filas (queue)

Me é cômico ver toda a suposta preparação em lógica de programação que os estudantes têm – isso advém da ausência de conteúdo sobre Aristóteles em

0.14.8. Listas Encadeadas

Me é cômico ver toda a suposta preparação em lógica de programação que os estudantes têm – isso advém da ausência de conteúdo sobre Aristóteles em

0.14.9. Árvores

Me é cômico ver toda a suposta preparação em lógica de programação que os estudantes têm – isso advém da ausência de conteúdo sobre Aristóteles em

0.15. Lógica Fuzzy e Incerteza

Me é cômico ver toda a suposta preparação em lógica de programação que os estudantes têm – isso advém da ausência de conteúdo sobre Aristóteles em

0.16. Lógica Modal Computacional

Me é cômico ver toda a suposta preparação em lógica de programação que os estudantes têm – isso advém da ausência de conteúdo sobre Aristóteles em

0.17. Autômatos e Linguagens Formais

Me é cômico ver toda a suposta preparação em lógica de programação que os estudantes têm – isso advém da ausência de conteúdo sobre Aristóteles em

0.18. Teoria da Computabilidade

Me é cômico ver toda a suposta preparação em lógica de programação que os estudantes têm – isso advém da ausência de conteúdo sobre Aristóteles em

0.19. Problemas NP, NP-completos e NP-difíceis

Me é cômico ver toda a suposta preparação em lógica de programação que os estudantes têm – isso advém da ausência de conteúdo sobre Aristóteles em

0.20. Redução de Problemas e Complexidade

Me é cômico ver toda a suposta preparação em lógica de programação que os estudantes têm – isso advém da ausência de conteúdo sobre Aristóteles em

0.21. Álgebra Linear

Me é cômico ver toda a suposta preparação em lógica de programação que os estudantes têm – isso advém da ausência de conteúdo sobre Aristóteles em

0.21.1. Vetores

Me é cômico ver toda a suposta preparação em lógica de programação que os estudantes têm – isso advém da ausência de conteúdo sobre Aristóteles em

0.21.2. Matrizes

Me é cômico ver toda a suposta preparação em lógica de programação que os estudantes têm – isso advém da ausência de conteúdo sobre Aristóteles em

0.21.3. Determinantes

Me é cômico ver toda a suposta preparação em lógica de programação que os estudantes têm – isso advém da ausência de conteúdo sobre Aristóteles em

0.22. Teoria dos Números

Me é cômico ver toda a suposta preparação em lógica de programação que os estudantes têm – isso advém da ausência de conteúdo sobre Aristóteles em

0.22.1. Primalidade

Me é cômico ver toda a suposta preparação em lógica de programação que os estudantes têm – isso advém da ausência de conteúdo sobre Aristóteles em

0.22.2. Criptografia

Me é cômico ver toda a suposta preparação em lógica de programação que os estudantes têm – isso advém da ausência de conteúdo sobre Aristóteles em

0.22.3. Modulares

Me é cômico ver toda a suposta preparação em lógica de programação que os estudantes têm – isso advém da ausência de conteúdo sobre Aristóteles em

0.23. Probabilidade e Estatística (análise de algoritmos aleatórios)

Me é cômico ver toda a suposta preparação em lógica de programação que os estudantes têm – isso advém da ausência de conteúdo sobre Aristóteles em

0.24. Geometria Computacional

Me é cômico ver toda a suposta preparação em lógica de programação que os estudantes têm – isso advém da ausência de conteúdo sobre Aristóteles em

0.25. Teoria dos Grafos

Me é cômico ver toda a suposta preparação em lógica de programação que os estudantes têm – isso advém da ausência de conteúdo sobre Aristóteles em

0.25.1. Fluxo Máximo

Me é cômico ver toda a suposta preparação em lógica de programação que os estudantes têm – isso advém da ausência de conteúdo sobre Aristóteles em

0.25.2. Emparelhamento

Me é cômico ver toda a suposta preparação em lógica de programação que os estudantes têm – isso advém da ausência de conteúdo sobre Aristóteles em

0.25.3. Redes

Me é cômico ver toda a suposta preparação em lógica de programação que os estudantes têm – isso advém da ausência de conteúdo sobre Aristóteles em

Capítulo 1 – Fundamentos de Algoritmos

1.1. Complexidade de Tempo e Espaço

Me é cômico ver toda a suposta preparação em lógica de programação que os estudantes têm – isso advém da ausência de conteúdo sobre Aristóteles em

1.1.1. Notação Big O (ex.: O(1), O(log n), O(n), O(n log n), O(n^2))

Me é cômico ver toda a suposta preparação em lógica de programação que os estudantes têm – isso advém da ausência de conteúdo sobre Aristóteles em

1.1.2. Melhor, Média e Pior Caso

Me é cômico ver toda a suposta preparação em lógica de programação que os estudantes têm – isso advém da ausência de conteúdo sobre Aristóteles em

1.1.3. Complexidade de Tempo de Operações Comuns (ex.: acesso a lista, inserção em tabela hash)

Me é cômico ver toda a suposta preparação em lógica de programação que os estudantes têm – isso advém da ausência de conteúdo sobre Aristóteles em

1.1.4. Análise de Complexidade de Espaço

Me é cômico ver toda a suposta preparação em lógica de programação que os estudantes têm – isso advém da ausência de conteúdo sobre Aristóteles em

1.1.5. Análise Amortizada (ex.: redimensionamento de array dinâmico)

Me é cômico ver toda a suposta preparação em lógica de programação que os estudantes têm – isso advém da ausência de conteúdo sobre Aristóteles em

1.2. Fundamentos Matemáticos

Me é cômico ver toda a suposta preparação em lógica de programação que os estudantes têm – isso advém da ausência de conteúdo sobre Aristóteles em

1.2.1. Logaritmos e Expoentes (crucial para análise de complexidade)

Me é cômico ver toda a suposta preparação em lógica de programação que os estudantes têm – isso advém da ausência de conteúdo sobre Aristóteles em

1.2.2. Combinatória Básica (permutações, combinações)

Me é cômico ver toda a suposta preparação em lógica de programação que os estudantes têm – isso advém da ausência de conteúdo sobre Aristóteles em

1.2.3. Aritmética Modular (usada em hashing e teoria dos números)

Me é cômico ver toda a suposta preparação em lógica de programação que os estudantes têm – isso advém da ausência de conteúdo sobre Aristóteles em

1.2.4. Relações de Recorrência (usadas em dividir para conquistar)

Me é cômico ver toda a suposta preparação em lógica de programação que os estudantes têm – isso advém da ausência de conteúdo sobre Aristóteles em

1.2.5. Técnicas de Prova (indução, contradição, etc.)

Me é cômico ver toda a suposta preparação em lógica de programação que os estudantes têm – isso advém da ausência de conteúdo sobre Aristóteles em

1.3. Representação de Dados e Operações com Bits

Me é cômico ver toda a suposta preparação em lógica de programação que os estudantes têm – isso advém da ausência de conteúdo sobre Aristóteles em

1.3.1. Conversão entre Binário, Decimal e Hexadecimal

Me é cômico ver toda a suposta preparação em lógica de programação que os estudantes têm – isso advém da ausência de conteúdo sobre Aristóteles em

1.3.2. Manipulação de Bits (AND, OR, XOR, NOT, deslocamentos de bits (bit shifts))

Me é cômico ver toda a suposta preparação em lógica de programação que os estudantes têm – isso advém da ausência de conteúdo sobre Aristóteles em

1.3.3. Representação em Complemento de Dois

Me é cômico ver toda a suposta preparação em lógica de programação que os estudantes têm – isso advém da ausência de conteúdo sobre Aristóteles em

1.3.4. Overflow e Underflow de Inteiros

Me é cômico ver toda a suposta preparação em lógica de programação que os estudantes têm – isso advém da ausência de conteúdo sobre Aristóteles em

1.3.5. Máscaras e Bandeiras de Bits

Me é cômico ver toda a suposta preparação em lógica de programação que os estudantes têm – isso advém da ausência de conteúdo sobre Aristóteles em

1.4. Memória e Arquitetura de Computadores (Noções Básicas)

Me é cômico ver toda a suposta preparação em lógica de programação que os estudantes têm – isso advém da ausência de conteúdo sobre Aristóteles em

1.4.1. Hierarquia de Memória (RAM, Cache, Registradores)

Me é cômico ver toda a suposta preparação em lógica de programação que os estudantes têm – isso advém da ausência de conteúdo sobre Aristóteles em

1.4.2. Pilha vs Heap

Me é cômico ver toda a suposta preparação em lógica de programação que os estudantes têm – isso advém da ausência de conteúdo sobre Aristóteles em

1.4.3. Ponteiros e Referências

Me é cômico ver toda a suposta preparação em lógica de programação que os estudantes têm – isso advém da ausência de conteúdo sobre Aristóteles em

1.4.4. Pilha de Execução e Pilha de Recursão

Me é cômico ver toda a suposta preparação em lógica de programação que os estudantes têm – isso advém da ausência de conteúdo sobre Aristóteles em

1.4.5. Gargalos entre CPU e Memória

Me é cômico ver toda a suposta preparação em lógica de programação que os estudantes têm – isso advém da ausência de conteúdo sobre Aristóteles em

1.5. Estruturas de Dados Essenciais

Me é cômico ver toda a suposta preparação em lógica de programação que os estudantes têm – isso advém da ausência de conteúdo sobre Aristóteles em

1.5.1. Vetores e Strings

Me é cômico ver toda a suposta preparação em lógica de programação que os estudantes têm – isso advém da ausência de conteúdo sobre Aristóteles em

1.5.2. Listas Ligadas

Me é cômico ver toda a suposta preparação em lógica de programação que os estudantes têm – isso advém da ausência de conteúdo sobre Aristóteles em

1.5.3. Pilhas e Filas

Me é cômico ver toda a suposta preparação em lógica de programação que os estudantes têm – isso advém da ausência de conteúdo sobre Aristóteles em

1.5.4. Tabelas de Dispersão / Mapas Hash

Me é cômico ver toda a suposta preparação em lógica de programação que os estudantes têm – isso advém da ausência de conteúdo sobre Aristóteles em

1.5.5. Árvores e Árvores Binárias

Me é cômico ver toda a suposta preparação em lógica de programação que os estudantes têm – isso advém da ausência de conteúdo sobre Aristóteles em

1.5.6. Grafos (lista de adjacência vs matriz)

Me é cômico ver toda a suposta preparação em lógica de programação que os estudantes têm – isso advém da ausência de conteúdo sobre Aristóteles em

1.5.7. Heaps / Filas de Prioridade

Me é cômico ver toda a suposta preparação em lógica de programação que os estudantes têm – isso advém da ausência de conteúdo sobre Aristóteles em

1.5.8. Tries

Me é cômico ver toda a suposta preparação em lógica de programação que os estudantes têm – isso advém da ausência de conteúdo sobre Aristóteles em

1.5.9. Conjuntos Disjuntos (Union-Find)

Me é cômico ver toda a suposta preparação em lógica de programação que os estudantes têm – isso advém da ausência de conteúdo sobre Aristóteles em

1.6. Fundamentos da Linguagem de Programação

Me é cômico ver toda a suposta preparação em lógica de programação que os estudantes têm – isso advém da ausência de conteúdo sobre Aristóteles em

1.6.1. Laços (Loops) e Recursão

Me é cômico ver toda a suposta preparação em lógica de programação que os estudantes têm – isso advém da ausência de conteúdo sobre Aristóteles em

1.6.2. Pilha de Chamadas de Função

Me é cômico ver toda a suposta preparação em lógica de programação que os estudantes têm – isso advém da ausência de conteúdo sobre Aristóteles em

1.6.3. Gerenciamento de Memória (manual vs coleta de lixo)

Me é cômico ver toda a suposta preparação em lógica de programação que os estudantes têm – isso advém da ausência de conteúdo sobre Aristóteles em

1.6.4. Passagem por Valor vs Passagem por Referência

Me é cômico ver toda a suposta preparação em lógica de programação que os estudantes têm – isso advém da ausência de conteúdo sobre Aristóteles em

1.6.5. Tipos Imutáveis vs Mutáveis

Me é cômico ver toda a suposta preparação em lógica de programação que os estudantes têm – isso advém da ausência de conteúdo sobre Aristóteles em

1.6.6. Uso da Biblioteca Padrão (ex.: ordenação, heapq em Python)

Me é cômico ver toda a suposta preparação em lógica de programação que os estudantes têm – isso advém da ausência de conteúdo sobre Aristóteles em

1.7. Princípios de Projeto de Algoritmos

Me é cômico ver toda a suposta preparação em lógica de programação que os estudantes têm – isso advém da ausência de conteúdo sobre Aristóteles em

1.7.1. Força Bruta

Me é cômico ver toda a suposta preparação em lógica de programação que os estudantes têm – isso advém da ausência de conteúdo sobre Aristóteles em

1.7.2. Estratégia Gulosa (Greedy)

Me é cômico ver toda a suposta preparação em lógica de programação que os estudantes têm – isso advém da ausência de conteúdo sobre Aristóteles em

1.7.3. Dividir para Conquistar

Me é cômico ver toda a suposta preparação em lógica de programação que os estudantes têm – isso advém da ausência de conteúdo sobre Aristóteles em

1.7.4. Programação Dinâmica (Memoização vs Tabela)

Me é cômico ver toda a suposta preparação em lógica de programação que os estudantes têm – isso advém da ausência de conteúdo sobre Aristóteles em

1.7.5. Retrocesso (Backtracking)

Me é cômico ver toda a suposta preparação em lógica de programação que os estudantes têm – isso advém da ausência de conteúdo sobre Aristóteles em

1.7.6. Percursos em Grafos

Me é cômico ver toda a suposta preparação em lógica de programação que os estudantes têm – isso advém da ausência de conteúdo sobre Aristóteles em

1.7.7. Janela Deslizante / Dois Ponteiros

Me é cômico ver toda a suposta preparação em lógica de programação que os estudantes têm – isso advém da ausência de conteúdo sobre Aristóteles e

Capítulo 2 – Algoritmos Básicos

2.1. Busca Linear

Me é cômico ver toda a suposta preparação em lógica de programação que os estudantes têm – isso advém da ausência de conteúdo sobre Aristóteles em

2.2. Busca Binária

Me é cômico ver toda a suposta preparação em lógica de programação que os estudantes têm – isso advém da ausência de conteúdo sobre Aristóteles em

2.3. Técnica de Dois Ponteiros

Me é cômico ver toda a suposta preparação em lógica de programação que os estudantes têm – isso advém da ausência de conteúdo sobre Aristóteles em

2.4. Janela Deslizante

Me é cômico ver toda a suposta preparação em lógica de programação que os estudantes têm – isso advém da ausência de conteúdo sobre Aristóteles em

Capítulo 3 – Algoritmos de Ordenação

3.1. Ordenação por Bolha (Bubble Sort)

Me é cômico ver toda a suposta preparação em lógica de programação que os estudantes têm – isso advém da ausência de conteúdo sobre Aristóteles em

3.2. Ordenação por Inserção (Insertion Sort)

Me é cômico ver toda a suposta preparação em lógica de programação que os estudantes têm – isso advém da ausência de conteúdo sobre Aristóteles em

3.3. Ordenação por Seleção (Selection Sort)

Me é cômico ver toda a suposta preparação em lógica de programação que os estudantes têm – isso advém da ausência de conteúdo sobre Aristóteles em

3.4. Merge Sort

Me é cômico ver toda a suposta preparação em lógica de programação que os estudantes têm – isso advém da ausência de conteúdo sobre Aristóteles em

3.5. Quick Sort

Me é cômico ver toda a suposta preparação em lógica de programação que os estudantes têm – isso advém da ausência de conteúdo sobre Aristóteles em

3.6. Heap Sort

Me é cômico ver toda a suposta preparação em lógica de programação que os estudantes têm – isso advém da ausência de conteúdo sobre Aristóteles em

3.7. Counting Sort

Me é cômico ver toda a suposta preparação em lógica de programação que os estudantes têm – isso advém da ausência de conteúdo sobre Aristóteles em

3.8. Radix Sort

Me é cômico ver toda a suposta preparação em lógica de programação que os estudantes têm – isso advém da ausência de conteúdo sobre Aristóteles em

Capítulo 4 – Recursão e Backtracking

4.1. Fatorial / Fibonacci (recursivo)

Me é cômico ver toda a suposta preparação em lógica de programação que os estudantes têm – isso advém da ausência de conteúdo sobre Aristóteles em

4.2. Template de Backtracking

Me é cômico ver toda a suposta preparação em lógica de programação que os estudantes têm – isso advém da ausência de conteúdo sobre Aristóteles em

4.2.1. Permutações

Me é cômico ver toda a suposta preparação em lógica de programação que os estudantes têm – isso advém da ausência de conteúdo sobre Aristóteles em

4.2.2. Combinações

Me é cômico ver toda a suposta preparação em lógica de programação que os estudantes têm – isso advém da ausência de conteúdo sobre Aristóteles em

4.2.3. Problema das N-Rainhas

Me é cômico ver toda a suposta preparação em lógica de programação que os estudantes têm – isso advém da ausência de conteúdo sobre Aristóteles em

4.2.4. Solucionador de Sudoku

Me é cômico ver toda a suposta preparação em lógica de programação que os estudantes têm – isso advém da ausência de conteúdo sobre Aristóteles em

4.2.5. Subconjuntos

Me é cômico ver toda a suposta preparação em lógica de programação que os estudantes têm – isso advém da ausência de conteúdo sobre Aristóteles em

Capítulo 5 – Dividir para Conquistar

5.1. Merge Sort

Me é cômico ver toda a suposta preparação em lógica de programação que os estudantes têm – isso advém da ausência de conteúdo sobre Aristóteles em

5.2. Quick Sort

Me é cômico ver toda a suposta preparação em lógica de programação que os estudantes têm – isso advém da ausência de conteúdo sobre Aristóteles em

5.3. Busca Binária

Me é cômico ver toda a suposta preparação em lógica de programação que os estudantes têm – isso advém da ausência de conteúdo sobre Aristóteles em

5.4. Subarray Máximo (Kadane via DyC)

Me é cômico ver toda a suposta preparação em lógica de programação que os estudantes têm – isso advém da ausência de conteúdo sobre Aristóteles em

Capítulo 6 – Algoritmos Gulosos

6.1. Seleção de Atividades

Me é cômico ver toda a suposta preparação em lógica de programação que os estudantes têm – isso advém da ausência de conteúdo sobre Aristóteles em

6.2. Agendamento de Intervalos

Me é cômico ver toda a suposta preparação em lógica de programação que os estudantes têm – isso advém da ausência de conteúdo sobre Aristóteles em

6.3. Mochila Fracionária (Fractional Knapsack)

Me é cômico ver toda a suposta preparação em lógica de programação que os estudantes têm – isso advém da ausência de conteúdo sobre Aristóteles em

6.4. Codificação de Huffman

Me é cômico ver toda a suposta preparação em lógica de programação que os estudantes têm – isso advém da ausência de conteúdo sobre Aristóteles em

6.5. Algoritmo de Dijkstra

Me é cômico ver toda a suposta preparação em lógica de programação que os estudantes têm – isso advém da ausência de conteúdo sobre Aristóteles em

Capítulo 7 – Programação Dinâmica

7.1. Fibonacci (com memoização)

Me é cômico ver toda a suposta preparação em lógica de programação que os estudantes têm – isso advém da ausência de conteúdo sobre Aristóteles em

7.2. Mochila 0/1

Me é cômico ver toda a suposta preparação em lógica de programação que os estudantes têm – isso advém da ausência de conteúdo sobre Aristóteles em

7.3. Mochila com Itens Infinitos

Me é cômico ver toda a suposta preparação em lógica de programação que os estudantes têm – isso advém da ausência de conteúdo sobre Aristóteles em

7.4. Subsequência Comum Mais Longa (LCS)

Me é cômico ver toda a suposta preparação em lógica de programação que os estudantes têm – isso advém da ausência de conteúdo sobre Aristóteles em

7.5. Subsequência Crescente Mais Longa (LIS)

Me é cômico ver toda a suposta preparação em lógica de programação que os estudantes têm – isso advém da ausência de conteúdo sobre Aristóteles em

7.6. Troco com Moedas (Coin Change)

Me é cômico ver toda a suposta preparação em lógica de programação que os estudantes têm – isso advém da ausência de conteúdo sobre Aristóteles em

7.7. Corte de Hastes (Rod Cutting)

Me é cômico ver toda a suposta preparação em lógica de programação que os estudantes têm – isso advém da ausência de conteúdo sobre Aristóteles em

7.8. Multiplicação de Cadeia de Matrizes

Me é cômico ver toda a suposta preparação em lógica de programação que os estudantes têm – isso advém da ausência de conteúdo sobre Aristóteles em

7.9. Distância de Edição (Edit Distance)

Me é cômico ver toda a suposta preparação em lógica de programação que os estudantes têm – isso advém da ausência de conteúdo sobre Aristóteles em

7.10. Particionamento Palindrômico

Me é cômico ver toda a suposta preparação em lógica de programação que os estudantes têm – isso advém da ausência de conteúdo sobre Aristóteles em

7.11. Correspondência com Curingas (Wildcard Matching)

Me é cômico ver toda a suposta preparação em lógica de programação que os estudantes têm – isso advém da ausência de conteúdo sobre Aristóteles em

Capítulo 8 – Algoritmos de Grafos

8.1. Busca em Largura (BFS)

Me é cômico ver toda a suposta preparação em lógica de programação que os estudantes têm – isso advém da ausência de conteúdo sobre Aristóteles em

8.2. Busca em Profundidade (DFS)

Me é cômico ver toda a suposta preparação em lógica de programação que os estudantes têm – isso advém da ausência de conteúdo sobre Aristóteles em

8.3. Ordenação Topológica

Me é cômico ver toda a suposta preparação em lógica de programação que os estudantes têm – isso advém da ausência de conteúdo sobre Aristóteles em

8.4. Algoritmo de Dijkstra

Me é cômico ver toda a suposta preparação em lógica de programação que os estudantes têm – isso advém da ausência de conteúdo sobre Aristóteles em

8.5. Algoritmo de Bellman-Ford

Me é cômico ver toda a suposta preparação em lógica de programação que os estudantes têm – isso advém da ausência de conteúdo sobre Aristóteles em

8.6. Algoritmo de Floyd-Warshall

Me é cômico ver toda a suposta preparação em lógica de programação que os estudantes têm – isso advém da ausência de conteúdo sobre Aristóteles em

8.7. Algoritmo A*

Me é cômico ver toda a suposta preparação em lógica de programação que os estudantes têm – isso advém da ausência de conteúdo sobre Aristóteles em

8.8. Union-Find (Conjuntos Disjuntos)

Me é cômico ver toda a suposta preparação em lógica de programação que os estudantes têm – isso advém da ausência de conteúdo sobre Aristóteles em

8.9. Algoritmo de Kruskal (Árvore Geradora Mínima)

Me é cômico ver toda a suposta preparação em lógica de programação que os estudantes têm – isso advém da ausência de conteúdo sobre Aristóteles em

8.10. Algoritmo de Prim (Árvore Geradora Mínima)

Me é cômico ver toda a suposta preparação em lógica de programação que os estudantes têm – isso advém da ausência de conteúdo sobre Aristóteles em

Capítulo 9 – Algoritmos em Árvores

9.1. DFS/BFS em Árvores

Me é cômico ver toda a suposta preparação em lógica de programação que os estudantes têm – isso advém da ausência de conteúdo sobre Aristóteles em

9.2. Percursos em Árvores Binárias (Pré-ordem, Em-ordem, Pós-ordem)

Me é cômico ver toda a suposta preparação em lógica de programação que os estudantes têm – isso advém da ausência de conteúdo sobre Aristóteles em

9.3. Menor Ancestral Comum (LCA)

Me é cômico ver toda a suposta preparação em lógica de programação que os estudantes têm – isso advém da ausência de conteúdo sobre Aristóteles em

9.4. Inserção/Remoção/Busca em Árvore Binária de Busca (BST)

Me é cômico ver toda a suposta preparação em lógica de programação que os estudantes têm – isso advém da ausência de conteúdo sobre Aristóteles em

9.5. Verificação de Balanceamento de Altura

Me é cômico ver toda a suposta preparação em lógica de programação que os estudantes têm – isso advém da ausência de conteúdo sobre Aristóteles em

9.6. Diâmetro de uma Árvore

Me é cômico ver toda a suposta preparação em lógica de programação que os estudantes têm – isso advém da ausência de conteúdo sobre Aristóteles em

9.7. Trie (Árvore de Prefixos)

Me é cômico ver toda a suposta preparação em lógica de programação que os estudantes têm – isso advém da ausência de conteúdo sobre Aristóteles em

Capítulo 10 – Estruturas de Dados Avançadas (Relacionadas a Algoritmos)

10.1. Árvore de Segmento (Segment Tree)

Me é cômico ver toda a suposta preparação em lógica de programação que os estudantes têm – isso advém da ausência de conteúdo sobre Aristóteles em

10.2. Árvore de Fenwick (Árvore Indexada Binária)

Me é cômico ver toda a suposta preparação em lógica de programação que os estudantes têm – isso advém da ausência de conteúdo sobre Aristóteles em

10.3. Union-Find com Compressão de Caminho

Me é cômico ver toda a suposta preparação em lógica de programação que os estudantes têm – isso advém da ausência de conteúdo sobre Aristóteles em

10.4. Trie com Armazenamento de Palavras

Me é cômico ver toda a suposta preparação em lógica de programação que os estudantes têm – isso advém da ausência de conteúdo sobre Aristóteles em

10.5. Fila de Prioridade / Operações com Heap

Me é cômico ver toda a suposta preparação em lógica de programação que os estudantes têm – isso advém da ausência de conteúdo sobre Aristóteles em

Capítulo 11 – Algoritmos Diversos de Alta Frequência

11.1. KMP (Knuth-Morris-Pratt) – Correspondência de Padrões

Me é cômico ver toda a suposta preparação em lógica de programação que os estudantes têm – isso advém da ausência de conteúdo sobre Aristóteles em

11.2. Algoritmo de Rabin-Karp

Me é cômico ver toda a suposta preparação em lógica de programação que os estudantes têm – isso advém da ausência de conteúdo sobre Aristóteles em

11.3. Algoritmo de Manacher (Maior Substring Palindrômica)

Me é cômico ver toda a suposta preparação em lógica de programação que os estudantes têm – isso advém da ausência de conteúdo sobre Aristóteles em

11.4. Amostragem Reservatória (Reservoir Sampling)

Me é cômico ver toda a suposta preparação em lógica de programação que os estudantes têm – isso advém da ausência de conteúdo sobre Aristóteles em

11.5. Máximo com Janela Deslizante (com Deque)

Me é cômico ver toda a suposta preparação em lógica de programação que os estudantes têm – isso advém da ausência de conteúdo sobre Aristóteles em

11.6. Top K Elementos (com Heap ou QuickSelect)

Me é cômico ver toda a suposta preparação em lógica de programação que os estudantes têm – isso advém da ausência de conteúdo sobre Aristóteles em

11.7. Detecção de Ciclo em Grafos (DFS + Union-Find)

Me é cômico ver toda a suposta preparação em lógica de programação que os estudantes têm – isso advém da ausência de conteúdo sobre Aristóteles em

Capítulo 12 - Aprofundando na Prática

12.1. Didática do Capítulo

Me é cômico ver toda a suposta preparação em lógica de programação que os estudantes têm – isso advém da ausência de conteúdo sobre Aristóteles em

12.2. Problemas Clássicos da Ciência da Computação

Me é cômico ver toda a suposta preparação em lógica de programação que os estudantes têm – isso advém da ausência de conteúdo sobre Aristóteles em

12.2.1. Soma de Elementos de um Vetor / Estrutura: Vetor, Array

Me é cômico ver toda a suposta preparação em lógica de programação que os estudantes têm – isso advém da ausência de conteúdo sobre Aristóteles em

12.2.2. Verificando Palíndromo / Estrutura: String, Array

Me é cômico ver toda a suposta preparação em lógica de programação que os estudantes têm – isso advém da ausência de conteúdo sobre Aristóteles em

12.2.3. Contagem de Frequências em uma String / Estrutura: Hash table / Array

Me é cômico ver toda a suposta preparação em lógica de programação que os estudantes têm – isso advém da ausência de conteúdo sobre Aristóteles em

12.2.5. Caminho Mínimo em Grafo Simples (Usando BFS (Busca em Largura) para grafos não ponderados) / Estrutura: Grafo

Me é cômico ver toda a suposta preparação em lógica de programação que os estudantes têm – isso advém da ausência de conteúdo sobre Aristóteles em

12.2.6. DFS em grafos / árvores (Percorrer todos os nós) / Estrutura: Grafo / Árvore

Me é cômico ver toda a suposta preparação em lógica de programação que os estudantes têm – isso advém da ausência de conteúdo sobre Aristóteles em

12.2.7. Problema da Mochila (Knapsack) – sem repetição / Estrutura: Programação Dinâmica (DP)

Me é cômico ver toda a suposta preparação em lógica de programação que os estudantes têm – isso advém da ausência de conteúdo sobre Aristóteles em

12.2.8. Subsequência Comum Máxima (LCS) / Estrutura: DP

Me é cômico ver toda a suposta preparação em lógica de programação que os estudantes têm – isso advém da ausência de conteúdo sobre Aristóteles em

12.2.9. Merge Sort e Quick Sort com Recursão e DC / Estrutura: Array, Recursão

Me é cômico ver toda a suposta preparação em lógica de programação que os estudantes têm – isso advém da ausência de conteúdo sobre Aristóteles em

12.2.10. Busca em Árvore Binária (BST) / Estrutura: Árvores Binárias

Me é cômico ver toda a suposta preparação em lógica de programação que os estudantes têm – isso advém da ausência de conteúdo sobre Aristóteles em

12.2.11. Caminho Mínimo em Grafos Ponderados com Dijkstra e Bellman-Ford / Estrutura: Grafo com Pesos

Me é cômico ver toda a suposta preparação em lógica de programação que os estudantes têm – isso advém da ausência de conteúdo sobre Aristóteles em

12.2.12. Árvore de segmentação (Segment Tree) – Consultas e Atualizações de Intervalos / Estrutura: Árvores Segmentadas / Interval Trees

Me é cômico ver toda a suposta preparação em lógica de programação que os estudantes têm – isso advém da ausência de conteúdo sobre Aristóteles em

12.2.13. Problema da mochila com repetição ilimitada (Unbounded Knapsack) / Estrutura: DP 1D/2D

Me é cômico ver toda a suposta preparação em lógica de programação que os estudantes têm – isso advém da ausência de conteúdo sobre Aristóteles em

12.2.14. Árvore binária balanceada (AVL / Red-Black Tree) – Inserção e Remoção Mantendo Balanceamento / Estrutura: Árvores binárias balanceadas

Me é cômico ver toda a suposta preparação em lógica de programação que os estudantes têm – isso advém da ausência de conteúdo sobre Aristóteles em

12.2.15. Subconjunto de Soma Máxima / Partição de Conjunto (Subset Sum) / Estrutura: DP de tabela, backtracking

Me é cômico ver toda a suposta preparação em lógica de programação que os estudantes têm – isso advém da ausência de conteúdo sobre Aristóteles em

12.2.16. Travelling Salesman Problem (TSP) – Abordagem DP com Bitmask / Estrutura: DP + Grafos + Bitmasking

Me é cômico ver toda a suposta preparação em lógica de programação que os estudantes têm – isso advém da ausência de conteúdo sobre Aristóteles em

Capítulo 13 – Algoritmos Aleatórios e Probabilísticos

13.1. QuickSort Aleatório

Me é cômico ver toda a suposta preparação em lógica de programação que os estudantes têm – isso advém da ausência de conteúdo sobre Aristóteles em

13.2. Randomized Select

Me é cômico ver toda a suposta preparação em lógica de programação que os estudantes têm – isso advém da ausência de conteúdo sobre Aristóteles em

13.3. Algoritmos Monte Carlo

Me é cômico ver toda a suposta preparação em lógica de programação que os estudantes têm – isso advém da ausência de conteúdo sobre Aristóteles em

13.4. Algoritmos Las Vegas

Me é cômico ver toda a suposta preparação em lógica de programação que os estudantes têm – isso advém da ausência de conteúdo sobre Aristóteles em

13.5. Amostragem Aleatória e Hashing Universal

Me é cômico ver toda a suposta preparação em lógica de programação que os estudantes têm – isso advém da ausência de conteúdo sobre Aristóteles em

Capítulo 14 – Técnicas Avançadas em Grafos

14.1. Fluxo Máximo (Ford-Fulkerson, Edmonds-Karp, Dinic)

Me é cômico ver toda a suposta preparação em lógica de programação que os estudantes têm – isso advém da ausência de conteúdo sobre Aristóteles em

14.2. Emparelhamento Máximo em Grafos Bipartidos

Me é cômico ver toda a suposta preparação em lógica de programação que os estudantes têm – isso advém da ausência de conteúdo sobre Aristóteles em

14.3. Componentes Fortemente Conexos (Kosaraju, Tarjan)

Me é cômico ver toda a suposta preparação em lógica de programação que os estudantes têm – isso advém da ausência de conteúdo sobre Aristóteles em

14.4. Pontes e Articulações

Me é cômico ver toda a suposta preparação em lógica de programação que os estudantes têm – isso advém da ausência de conteúdo sobre Aristóteles em

14.5. Caminhos Mínimos Avançados (Johnson, A* com heurística)

Me é cômico ver toda a suposta preparação em lógica de programação que os estudantes têm – isso advém da ausência de conteúdo sobre Aristóteles em

14.6. Ciclo Euleriano e Hamiltoniano

Me é cômico ver toda a suposta preparação em lógica de programação que os estudantes têm – isso advém da ausência de conteúdo sobre Aristóteles em

Capítulo 15 – Estruturas de Dados Avançadas II

15.1. Skip Lists

Me é cômico ver toda a suposta preparação em lógica de programação que os estudantes têm – isso advém da ausência de conteúdo sobre Aristóteles em

15.2. Segment Tree Lazy Propagation

Me é cômico ver toda a suposta preparação em lógica de programação que os estudantes têm – isso advém da ausência de conteúdo sobre Aristóteles em

15.3. Persistent Data Structures

Me é cômico ver toda a suposta preparação em lógica de programação que os estudantes têm – isso advém da ausência de conteúdo sobre Aristóteles em

15.4. Suffix Array e Suffix Tree

Me é cômico ver toda a suposta preparação em lógica de programação que os estudantes têm – isso advém da ausência de conteúdo sobre Aristóteles em

15.5. RMQ (Range Minimum Query) e LCA Avançado

Me é cômico ver toda a suposta preparação em lógica de programação que os estudantes têm – isso advém da ausência de conteúdo sobre Aristóteles em

Capítulo 16 – Técnicas Avançadas de Programação Dinâmica

16.1. DP com Bitmask

Me é cômico ver toda a suposta preparação em lógica de programação que os estudantes têm – isso advém da ausência de conteúdo sobre Aristóteles em

16.2. DP em Grafos

Me é cômico ver toda a suposta preparação em lógica de programação que os estudantes têm – isso advém da ausência de conteúdo sobre Aristóteles em

16.3. Otimização de Estado em DP (1D/2D)

Me é cômico ver toda a suposta preparação em lógica de programação que os estudantes têm – isso advém da ausência de conteúdo sobre Aristóteles em

16.4. Problemas de Contagem (Paths, Partições)

Me é cômico ver toda a suposta preparação em lógica de programação que os estudantes têm – isso advém da ausência de conteúdo sobre Aristóteles em

16.5. DP em Subconjuntos / Máscaras de Bits

Me é cômico ver toda a suposta preparação em lógica de programação que os estudantes têm – isso advém da ausência de conteúdo sobre Aristóteles em

Capítulo 17 – Matemática Discreta Aplicada

17.1. Teoria dos Conjuntos

Me é cômico ver toda a suposta preparação em lógica de programação que os estudantes têm – isso advém da ausência de conteúdo sobre Aristóteles em

17.2. Funções e Relações

Me é cômico ver toda a suposta preparação em lógica de programação que os estudantes têm – isso advém da ausência de conteúdo sobre Aristóteles em

17.3. Grafos e Árvores (terminologia e propriedades)

Me é cômico ver toda a suposta preparação em lógica de programação que os estudantes têm – isso advém da ausência de conteúdo sobre Aristóteles em

17.4. Recorrências e Sequências

Me é cômico ver toda a suposta preparação em lógica de programação que os estudantes têm – isso advém da ausência de conteúdo sobre Aristóteles em

17.5. Principio da Inclusão-Exclusão

Me é cômico ver toda a suposta preparação em lógica de programação que os estudantes têm – isso advém da ausência de conteúdo sobre Aristóteles em

17.6. Análise Combinatória Avançada

Me é cômico ver toda a suposta preparação em lógica de programação que os estudantes têm – isso advém da ausência de conteúdo sobre Aristóteles em

Capítulo 18 – Problemas Clássicos Avançados

18.1. Problema do Caixeiro Viajante (TSP)

Me é cômico ver toda a suposta preparação em lógica de programação que os estudantes têm – isso advém da ausência de conteúdo sobre Aristóteles em

18.2. Problema do Fluxo Máximo

Me é cômico ver toda a suposta preparação em lógica de programação que os estudantes têm – isso advém da ausência de conteúdo sobre Aristóteles em

18.3. Problema da Mochila com Restrições

Me é cômico ver toda a suposta preparação em lógica de programação que os estudantes têm – isso advém da ausência de conteúdo sobre Aristóteles em

18.4. Problema de Partição de Conjuntos

Me é cômico ver toda a suposta preparação em lógica de programação que os estudantes têm – isso advém da ausência de conteúdo sobre Aristóteles em

18.5. Sudoku Generalizado

Me é cômico ver toda a suposta preparação em lógica de programação que os estudantes têm – isso advém da ausência de conteúdo sobre Aristóteles em

18.6. Problemas de Grafos Planar e Árvores Geradoras Mínimas

Me é cômico ver toda a suposta preparação em lógica de programação que os estudantes têm – isso advém da ausência de conteúdo sobre Aristóteles em

18.7. Problemas de Palíndromos e Strings

Me é cômico ver toda a suposta preparação em lógica de programação que os estudantes têm – isso advém da ausência de conteúdo sobre Aristóteles em

Capítulo 19 – À Prática: O Mercado de Trabalho

19.1. Panorama do Mercado de TI

Me é cômico ver toda a suposta preparação em lógica de programação que os estudantes têm – isso advém da ausência de conteúdo sobre Aristóteles em

19.2. "Perfis Profissionais" e Caminhos Possíveis

Me é cômico ver toda a suposta preparação em lógica de programação que os estudantes têm – isso advém da ausência de conteúdo sobre Aristóteles em

19.3. O que o Mercado Espera de um Programador

Me é cômico ver toda a suposta preparação em lógica de programação que os estudantes têm – isso advém da ausência de conteúdo sobre Aristóteles em

19.4. Ferramentas Essenciais do Programador

Me é cômico ver toda a suposta preparação em lógica de programação que os estudantes têm – isso advém da ausência de conteúdo sobre Aristóteles em

19.5. Construindo & Importância de um Portfólio

Me é cômico ver toda a suposta preparação em lógica de programação que os estudantes têm – isso advém da ausência de conteúdo sobre Aristóteles em

19.6. Projetos Reais e Experiência Prática

Me é cômico ver toda a suposta preparação em lógica de programação que os estudantes têm – isso advém da ausência de conteúdo sobre Aristóteles em

19.7. Versionamento e Trabalho em Equipe

Me é cômico ver toda a suposta preparação em lógica de programação que os estudantes têm – isso advém da ausência de conteúdo sobre Aristóteles em

19.8. Princípios de Engenharia de Software

Me é cômico ver toda a suposta preparação em lógica de programação que os estudantes têm – isso advém da ausência de conteúdo sobre Aristóteles em

19.9. Linguagens e Tecnologias Mais Pedidas

Me é cômico ver toda a suposta preparação em lógica de programação que os estudantes têm – isso advém da ausência de conteúdo sobre Aristóteles em

19.10. Bancos de Dados e Armazenamento

Me é cômico ver toda a suposta preparação em lógica de programação que os estudantes têm – isso advém da ausência de conteúdo sobre Aristóteles em

19.11. APIs e Comunicação entre Sistemas

Me é cômico ver toda a suposta preparação em lógica de programação que os estudantes têm – isso advém da ausência de conteúdo sobre Aristóteles em

19.12. Desenvolvimento Web e Mobile

Me é cômico ver toda a suposta preparação em lógica de programação que os estudantes têm – isso advém da ausência de conteúdo sobre Aristóteles em

19.13. DevOps e Cloud Computing

Me é cômico ver toda a suposta preparação em lógica de programação que os estudantes têm – isso advém da ausência de conteúdo sobre Aristóteles em

19.14. Segurança e OPSEC

Me é cômico ver toda a suposta preparação em lógica de programação que os estudantes têm – isso advém da ausência de conteúdo sobre Aristóteles em

19.15. Ciência de Dados e IA

Me é cômico ver toda a suposta preparação em lógica de programação que os estudantes têm – isso advém da ausência de conteúdo sobre Aristóteles em

19.16. Processo de Seleção

Me é cômico ver toda a suposta preparação em lógica de programação que os estudantes têm – isso advém da ausência de conteúdo sobre Aristóteles em

19.17. Portais, Freelance e Networking | Minhas Recomendações

Me é cômico ver toda a suposta preparação em lógica de programação que os estudantes têm – isso advém da ausência de conteúdo sobre Aristóteles em

19.18. Ética e Responsabilidade Profissional

Me é cômico ver toda a suposta preparação em lógica de programação que os estudantes têm – isso advém da ausência de conteúdo sobre Aristóteles em

19.19. Carreira Internacional e Trabalho Remoto | Minhas Recomendações

Me é cômico ver toda a suposta preparação em lógica de programação que os estudantes têm – isso advém da ausência de conteúdo sobre Aristóteles em

19.20. Crescimento e Aprendizado Contínuo / Minhas Recomendações

Me é cômico ver toda a suposta preparação em lógica de programação que os estudantes têm – isso advém da ausência de conteúdo sobre Aristóteles em