Hoje somos inundados por diversos dispositivos digitais executando tarefas de forma automatizada. Embora seja muito intuitivo uma tarefa é uma sequência ordenada de passos finitos para resolver um determinado problema, sistematizar esse processo é denominado de algoritmo. Um algoritmo é uma ferramenta utilizada em várias áreas do conhecimento, na matemática alguns algoritmos são responsáveis por padronizar uma operação matemática qualquer e essa padronização pode ser facilmente repetida sempre que necessário. Na ciência da computação ele é fundamental para a construção de sistemas, de forma pragmática, um aplicativo é o algoritmo que foi programado em uma determinada linguagem de programação e foi transformado em linguagem de máquina podendo ser executado em computador. Vamos entender o processo de criação de um algoritmo, algumas padronizações, sua complexidade e eficiência, bem como entender como os tipos abstratos de dados vetores e matrizes funcionar.

Podemos construir algoritmos para uma infinidade de coisas, na verdade, a todo momento usamos e colocamos em prática algoritmos, seja de forma intuitiva quando queremos descobrir como resolver um dado problema, seja de forma explicita por meio de ferramentas especializadas que aplica um algoritmo já determinado. Vejamos, se você é um chefe de cozinha e cria receitas novas, você está construindo algoritmos, se você compra seu café da manhã por meio de um aplicativo que seleciona uma refeição seja a partir de uma dieta especifica ou não, você está usando algoritmos. Compreender o seu conceito é base para a construção de conhecimento mais solido, afinal desenvolver algoritmos estimula o raciocínio lógico e a cognição das pessoas. E quanto mais especializado um algoritmo é, mais eficiente ele se torna.

A Teoria da Computação busca determinar quais problemas podem ser aplicados em um modelo de computação, esse modelo formalmente pode ser definida como a solução de um problema por meio de um algoritmo, assim chegamos a conclusão que um algoritmo para ser computado deve obedecer a arquitetura de Von Newmann. Isso significa que a partir de entradas um algoritmo pode ser processado e produzir saídas resultantes. Em uma calculadora a função soma, é resultado de um algoritmo que ao receber valores retorna o valor resultante da sua operação. É importante salientar que não existe um algoritmo único, cada pessoa pode ver de forma diferente como resolver um dado problema. Alguns algoritmos são de fato muito simples enquanto outros muito difíceis, o nível de complexidade entre algoritmos pode ser classificado como, Fáceis (P), Razoáveis (NP), Difíceis (NP-Completo).

Algoritmos são classificados pelo tipo de problema que eles podem resolver e o tempo usado para tal, um fato importante é que o tempo de processamento de um algoritmo pode mudar de acordo com o tamanho da entrada que será utilizada, formalmente o tempo de um algoritmo é expresso pelo polinômio O(n^^3). Podemos separa-los como, Classe P, aqui estão vários problemas naturais, como, estruturas de decisão de programação linear, cálculo de Máximo divisor comum e são problemas que podem ser resolvidos em tempo polinomial, ao generalizarmos P, obtemos a Classe NP que é o conjunto de todos os problemas que podem ser resolvidos por algoritmos não-

deterministicos em tempo polinomial usando uma máquina de Turing não-deterministica - em uma máquina de estados um autômato determinístico demonstra que para cada estado existe apenas uma transição possível, já em um autômato não determinístico pode existir estados que podem levar a outros estados. Os problemas NP mais difíceis são os da classe NP-completo, estes por sua vez são problemas de decisão do tipo "sim ou não" e ainda existem problemas de otimização chamados de NP-Difícil, contudo sua dificuldade não é menor que a dificuldade de decisão dos problemas NP-Completo. Outra características de problemas NP-Completo é que, se qualquer um deles puder ser resolvido em tempo polinomial então todos os outros problema NP-completo também terão uma solução com tempo polinomial.

O estudo da complexidade dos algoritmos permite observar o quanto de empenho existe na busca de algoritmos mais eficientes. Para alcançar essa eficiência é importante obedecer algumas premissas, algoritmos devem ser simples e não podem ter ambiguidade, as ações devem ser ordenadas e estabelecer todas as sequencias de ações em passos finitos. Ele deve ser capaz de ler e escrever dados, avaliar expressões algébricas, relacionais e lógicas, ser capaz de tomar decisões com base em expressões avaliadas e repetir esse conjunto de ações de acordo com a necessidade. Relembrando, um software é um algoritmo que foi programado, à vista disso, o requisito básico é que a especificação deve fornecer uma descrição precisa do procedimento computacional a ser seguido.

Os algoritmos são independentes de linguagem contudo a técnica de representação por meio de pseudo-código utiliza um conjunto restrito de palavras-chaves como, 1) leia e escreva, responsáveis pelo processo de entrada e saída; 2) se, então, senão, senão-se, são estruturas de controle e decisão; 3) para, faça, enquanto é uma estrutura de loop ou laço; e 4) fim-se; geralmente essas palavras estão em uma linguagem nativa ao desenvolvedor. Esse tipo de representação também não possui a rigidez do formalismo das linguagens de programação propriamente dita, permitindo ser fácil de interpretar e de codificar. Contudo, o pseudo-código permite aplicar tanto conceitos lógicos como, verdadeiro e falso, quanto conceitos matemáticos como, conjuntos numéricos, algebra entre outros. Um conceito fundamental é o da variável, elas permitem realizar as mais diversas operações com os mais diversos tipos de dados diferentes.

As variáveis são espaços reservados na memória RAM do computador para guardar informações que serão utilizadas durante o código do programa. Pode-se armazenar nestas variáveis diversos valores de diversos tipos e tamanhos, tais como números inteiros, números reais, caracteres, frases, enfim, diversas coisas. A diversidade de tipos de dados que uma variável pode assumir é muito grande e não conhecer o tipo de dado que está armazenado no momento de uma determinada operação pode ocasionar erros, por isso, existem alguns tipos básicos em todas as linguagens de programação como: int (valores inteiros), float (valores decimais), e string (representa cadeias de caracteres, seja números ou letras). O tipo de um dado define o conjunto de valores ao qual o dado pertence, bem como o conjunto de todas as operações que podem atuar

sobre qualquer valor daquele conjunto de valores. Uma variável possui nome, tipo e conteúdo e pode ser representada em pseudo-código assim: var nome : string; var media : float; Por ver, quando é necessário mostrar o valor de uma variável em um pseudo-código, use o comando "imprima (conjunto de variáveis)", assim para construir a mensagem "Caro <aluno> sua média é <media>" comando é " imprima 'Caro' + nome + 'sua média é' + str(media)". Veja que é possível construir textos interpolando com algumas variáveis utilizando a operação de concatenação, que usa o mesmo conceito e sinal simbólico da operação aritmética soma. Contudo essa operação só é permitida para o tipo String, qualquer variável de outro tipo, deverá ser convertida para string atravéz do comado str(variavel) que converte outros tipos para script. Em muitos casos precisamos armazenar um numero considerável de dados uma variável, como uma lista de pessoas que participam de um concurso. Para esse caso, podemos usar uma estrutura de dado denominado de Vetor ou Matriz.

Um vetor é um tipo especial de variável e possui uma estrutura uni-dimensional capaz de armazenar um conjunto de variáveis do mesmo tipo acessíveis com um único nome. Em computação um Vetor (Array) é constituído por dados agrupados continuamente na memória e, é acessados por referência a posição do dado na variável, sua representação é variável[índice], e o vetor é variável = []. Vamos representar dois vetores, o primeiro com o nome de 3 alunos e o seguinte com as médias de suas notas, alunos = ['Maria', 'Pedro', 'Marcos'], médias = [10, 7, 9] respectivamente. Vetores são tipos de variáveis de referência se diferenciando das variáveis comuns. Ao observar o vetor alunos, nota-se algumas características, como, alunos é um vetor de tamanho 3, possui apenas strings, e seu maior índice é 2. Isso porque na computação a contagem sempre inicia de 0, assim, Maria, Pedro e Marcos respectivamente tem índices 0,1 e 2. Ao executar o comando " imprima alunos, médias", um resultado praticamente inútil mostra a resposta - ['Maria', 'Pedro', 'Marcos'], [10, 7, 9], o que não traz sentido. Contudo o objetivo agora é apenas imprimir: "Maria possui média 10" assim, ao relembrar a representação do vetor, a reescrita do comando é " imprima alunos[0] + ' possui média ' + medias[0]". O uso de vetores aplicando alguma estrutura de controle já apresentada, poderia otimizar o ultimo comando imprima mostrado, e imprimir de uma vez só todos os alunos. Veja o pseudo-código usando a estrutura de laço "para" aplicado aos vetores alunos e médias, "para aluno,media em alunos,medias: imprima aluno + ' possui ' + media". O comando para vai verificar se existe um aluno o vetor alunos o simultaneamente o mesmo fará com a média, e sempre que encontrar uma combinação satisfatória ele executa o comando imprima, consequentemente ira percorres as listas até o fim, imprimindo uma mensagem para cada aluno. Haja visto o poder que um vetor possui, já que ele pode armazenar vários dados em uma mesma instância de variável, imagine, vetores dentro de vetores.

Em diversas circustância é necessário construir arranjos mais complexos, denominadas matrizes. Esta estrutura tem comportamento singular ao vetor com a diferença de ser n-dimensional, como mostra o vetor = [[[1],[2],[3]][[4],[5],[6]]]. Estas estruturas permitem movimentar fluxos na

memória estruturas mais volumosa de dados. Uma matriz possui um comportamento semelhante a uma planilha eletrônica, com células, linhas e colunas, desta forma uma matriz pode ser dica como variável quadrada 2x2 se o numero de linhas e colunas for 2, já a matriz vetor é 2x3. A busca de um valor na matriz considera que cada linha da matriz como é vetor, assim, para encontrar o valor 3 o vetor[0][2], e representa a linha 1 na coluna 2 – é importante lembrar que na computação a contagem inicia de 0. Não existe um limite de profundidade dentro de uma matriz, cada celula da matriz pode ter uma outra matriz. Hoje é muito comum utilizar matrizes, haja visto que os sistemas bigdata organizam dados em forma de matrizes, e as mais elaboradas consultas são continuamente otimizadas, isso pode demonstrar o quanto eficiente são as matrizes.

Podemos concluir que um algoritmo é uma ferramenta metodológica tanto para aprendizado da lógica de programação quanto é utilizada por desenvolvedores experientes enxergar de forma clara a solução de determinados problemas. O estudo de estruturas de dados como vetores e matrizes são determinantes para a construção de novos algoritmos mais otimizados.