AED I Turma I Março/2023 Prof. Arlindo Flavio da Conceição
Larissa Adelino Ângelo Rosa RA 163879

Resumo sobre o livro "Algoritmos em Linguagem C", de Paulo Feofiloff

Este livro possui a introdução de algoritmos e estruturas de dados (além de alguns conceitos de C), que é exatamente o conteúdo de AED I. Além desse conteúdo ser útil para desenvolver o raciocínio lógico e também criar ferramentas para diversas áreas, ele é essencial para a computação. Profissionais da computação devem entender como os dados são processados para poder buscar a melhor forma de lidar com eles.

A divisão dos capítulos, utilizada no prefácio, é pertinente para falar de cada tema do livro:

- **Capítulo 1 (Documentação e invariantes):** A boa documentação diz o que a função faz, não como ela faz. E invariante refere-se aos valores de variáveis no início de cada processo de interação, que de acordo com o autor é o único tipo de comentário que vale a pena ser feito no corpo de uma função.

- **Capítulo 2 (Algoritmos recursivos):** A recursão divide o problema em problemas menores antes de aplicar o algoritmo de solução; sua solução volta ao problema original e o algoritmo de solução é aplicado nesse problema com partes já resolvidas. Isso facilita a resolução do problema, deixando ele menos complicado antes de achar a solução final.

- **Capítulo 3 (Vetores):** Vetores são estruturas que servem para armazenar uma sequência de dados (do mesmo tipo) em posições consecutivas na memória. O livro aborda a busca, remoção e inserção de dados nos vetores, para ilustrar outros conceitos tratados no livro, como recursão, por exemplo.

- **Capítulo 4, 5 e 6 (Listas encadeadas, Filas e Pilhas): Lista** encadeada é uma representação de objetos (registros/structs) na memória do computador, mas que diferente dos vetores, não precisam estar em posições consecutivas na memória. É utilizado o termo "célula" (como numa planilha) para referir-se ao local onde cada objeto fica guardado. Cada célula guarda a posição da próxima (e a última guarda NULL), isso determina a ordem da lista. É abordado também: lista com e sem cabeça, remoção, inserção, busca e outros tipos de listas encadeadas. O conceito de **fila** é o mesmo utilizado no nosso dia a dia. É uma lista dinâmica com elementos inseridos em ordem cronológica em que o elemento que entrou primeiro na fila é o primeiro a sair dela. Como é difícil prever o tamanho a ser reservado para a fila, ela pode ser implementada de maneira circular. Também há a implementação usando lista encadeada, que utiliza ponteiros. **Pilha** é uma sequência dinâmica de elementos (eles podem ser inseridos e removidos) do mesmo tipo. Na remoção sempre é removido o último da sequência e na inserção sempre é inserido no fim da sequência (LIFO: Last-In-First-Out). Em C ela pode ser implementada utilizando vetor ou lista encadeada.

- **Capítulo 7 (Busca em vetor ordenado)**: Vetores podem ser ordenados de forma crescente ou decrescente, de acordo com seus índices. Este capítulo aborda os tipos de busca dos valores do vetor. A busca sequencial é lenta e simples usando o while e uma variável contadora para ser comparada a todos os índices. A busca binária é mais eficiente, pois vai dividindo pela metade (mais de uma vez, se preciso) os intervalos de busca, não sendo necessário percorrer todos os índices. Na forma recursiva a função invoca a si mesma quantas vezes for necessário.

- **Capítulos 8, 9, 10 e 11 (Algoritmos de ordenação):** Servem para colocar componentes de um vetor numa ordem crescente para solucionar problemas. Os dois mais simples são o algoritmo de inserção e o algoritmo de seleção. Num algoritmo de ordenação estável os componentes repetidos vem sempre um antes do outro. O algoritmo **Mergesort** ordena os elementos de uma forma mais sofisticada, usando intercalação de vetores alternados de

forma recursiva. Na versão iterativa deste algoritmo cada iteração intercala dois blocos de elementos. O algoritmo **Heapsort** não precisa de um vetor auxiliar e utiliza uma estrutura de dados conhecida como heap (que possui max-heap, min-heap, tem a estrutura de árvore; e pode ter elementos inseridos e trocados de posição com funções auxiliares). O Heapsort usa essas funções auxiliares para colocar o vetor em ordem crescente, primeiro transformando-o em um max-heap e depois rearranjando suas posições na ordem crescente. O algoritmo **Quicksort** coloca os valores pequenos de um lado e grandes do outro (de forma mais ou menos equilibrada), utilizando um pivô.

- **Capítulo 12 (Algoritmos de enumeração):** Há problemas que precisam enumerar objetos (de um determinado tipo) para poderem ser resolvidos, comumente são grandes números de objetos e algoritmos que fazem isso consomem muito tempo. Os algoritmos de enumeração apresentados no livro utilizam a enumeração de subsequências em ordem lexicográfica (como a ordenação utilizada em dicionários). A ordem lexicográfica especial dá preferência às subsequências mais longas.

- **Capítulo 13 (Busca de palavras em um texto):** Usando como exemplo a busca de uma palavra em um texto com retorno de seu número de ocorrências: O algoritmo trivial faz comparações entre os dois vetores, mas pode consumir muito tempo. O **primeiro algoritmo de Boyer-Moore** tem duas fases: o pré-processamento da palavra e a procura de ocorrências; e o seu número de comparações é menor que o do algoritmo trivial. O **segundo algoritmo de Boyer-Moore** faz esse pré-processamento construindo uma tabela de deslocamento e depois procura as ocorrências da palavra. E o **terceiro algoritmo de Boyer-Moore** é uma junção dos outros dois, ele a cada passo o algoritmo escolhe o maior dos deslocamentos.

- **Capítulos 14 e 15 (Árvores binárias e Árvores de busca):** Árvore binária é um conjunto de registros (chamados de nós) onde cada um deles possui três campos: um para o conteúdo e dois para ponteiros (um para a esquerda e outro para a direita). Esses campos dos ponteiros podem ter endereços de outros nós ou NULL. Esses endereços criam uma hierarquia de pai e filho (como uma árvore genealógica), onde pode haver filho esquerdo e filho direito. Também é possível acrescentar o campo pai. São utilizados conceitos como subárvores e descendentes. O endereço de uma árvore binária pode ser o endereço de sua raiz ou NULL. Existe a varredura de nós a esquerda-raiz-direita, mas não é a única. A altura de um nó é a distância entre ele e seu descendente mais afastado. Quando as subárvores direita e esquerda têm aproximadamente a mesma altura, a árvore é considerada balanceada. É possível utilizar um algoritmo para calcular o nó seguinte. As **árvores binárias de busca** possuem um campo a mais, o chave, que se colocado corretamente permite que a varredura esquerda-raiz-direita veja as chaves na ordem crescente. Para a busca e inserção nesse tipo de árvore é utilizado o valor da chave. E na remoção outro nó assume papel de raiz.

  No prefácio é sugerido que a leitura comece pelo apêndice "A Leiaute", ele dá recomendações sobre espaços entre palavras e símbolos e também sobre a indentação das linhas de código. Do apêndice B ao K (Caracteres; Números: naturais e inteiros; Endereços e ponteiros; Registros e structs; Alocação dinâmica de memória; Strings; Entrada e saída; Números aleatórios; Miscelânea; Arquivos-interface de bibliotecas) há assuntos mais básicos de programação em C (para consulta de quem já aprendeu o conteúdo). Cada capítulo possui exercícios, algo que é indispensável no processo de aprendizagem de programação, e o último apêndice, "L Soluções de alguns exercícios", possui algumas soluções.

  No livro são abordados temas desde documentação e ferramentas da linguagem C, até conceitos mais específicos de estruturas de dados e como utilizá-las. Falando de forma geral, esses temas são estudados para aprendermos a resolver problemas de forma lógica e eficiente, e entendermos como os dados podem ser organizados num dispositivo.