GCC224 - Introdução aos Algoritmos

Roteiro de Estudos Orientado (REO) - 13/07/2020 à 26/07/2020

1. Conteúdo abordado

Nessas duas semanas de estudo remoto, vamos abordar os seguintes conteúdos:

- Ordenação pelo Shell Sort;
- Ordenação pelo Quick Sort;
- Ordenação pelo Merge Sort.

2. Conhecimentos Necessários e Importância do Assunto

Na quinzena anterior, começamos a estudar algoritmos clássicos de ordenação, vendo os algoritmos mais intuitivos. Nesta quinzena veremos outros algoritmos clássicos, dessa vez menos intuitivos e mais práticos, ou seja, que desempenham melhor em condições reais de uso.

É importante conhecer os algoritmos clássicos de ordenação por diversos motivos. Quem está se preparando para produzir soluções algorítmicas para problemas precisa ter claro que existem muitas formas de se resolver qualquer problema. Existe uma tendência natural das pessoas a se perguntar qual das soluções é a melhor, entretanto essa é uma pergunta muitas vezes subjetiva. Aquilo que é melhor para uns (ou numa determinada situação) não é melhor para outros (ou noutra situação). Aprender os algoritmos clássicos de ordenação é uma primeira visão que na solução de problemas, melhorar algum aspecto geralmente significa piorar outro. Adicionalmente, os estudantes de computação ao aprender algoritmos começam a descobrir que existem grandes similaridades entre problemas aparentemente não relacionados; os problemas podem ser agrupados em categorias. Conhecer soluções de destaque, criadas por terceiros proporciona ferramentas para resolver outros problemas da mesma categoria. Fazendo uma analogia com Arte, é importante que o aprendiz de pintor conheça as grandes obras de artistas renomados.

O algoritmo *Shellsort* é uma generalização do *Insertion sort* que adiciona um parâmetro de configuração conhecido como salto (gap). Essa configuração extra permite adaptar o algoritmo a conjuntos de dados para os quais temos alguma informação da sua distribuição.

O algoritmo *Quicksort* é completamente novo, um pouco configurável, mas que desempenha muito bem em condições aleatórias, ou seja, em que praticamente não existe informação sobre a distribuição dos dados. O Quicksort é a

implementação de ordenação mais comum em bibliotecas de programação da maioria das linguagens de programação, algumas vezes aparecendo numa implementação híbrida que agrega outro método de ordenação. É um exemplo famoso de estratégia “dividir para conquistar” no projeto de algoritmos.

O algoritmo *Mergesort* tem um ótimo desempenho, fazendo uso de um recurso cuja disponibilidade precisa ser verificada: memória adicional. Ao contrário dos outros algoritmos, para ordenar *n* itens, este algoritmo precisa de espaço para guardar *2n* itens. É famoso pelo seu desempenho superior nos casos em a memória onde os itens estão armazenados precisa ser acessada sequencialmente, ou ao menos quando o acesso sequencial é mais rápido que o acesso aleatório, uma situação muito comum em dispositivos de memória secundária, como discos rígidos. Sua popularidade também aumentou com a redução dos custos de memória primária, o que tornou mais aceitável seu gasto extra de armazenamento.

3. Estratégias para Aprendizagem

Inicialmente, o estudante deverá assistir à videoaula disponibilizada na sala de aula virtual no Google Classroom, usando como material de apoio, o conjunto de slides disponível na seção Slides.

Há livros digitais na biblioteca digital da UFLA que abordam os conteúdos e estão descritos na seção de referências.

Na sala de aula virtual, há uma seção denominada **Mural** para que os estudantes postem suas dúvidas. Não coloquem código fonte de exercícios avaliativos no mural. É importante tirar dúvidas sobre um conteúdo antes de avançar para o conteúdo seguinte. Antes de submeter a solução de um exercício no Dredd, aconselha-se que você estude o conteúdo.

Reuniões no Meet para explicações adicionais e discussão de dúvidas serão divulgadas no mural da sala virtual.

De acordo com as dúvidas que vão surgindo, será disponibilizado material de apoio na sala virtual.

No ensino remoto, há maior dificuldade de entendimento do problema que o estudante está apresentando, por isso, a postagem de uma dúvida deve ser realizada de forma mais clara. Assim, o estudante deverá verificar se alguém já perguntou algo similar à sua dúvida e se a resposta obtida esclarece seu questionamento. Caso sua dúvida não tenha sido esclarecida, deve colocar na sua pergunta uma descrição dos passos que realizou para chegar no problema ou dúvida. Por exemplo, se ao executar o programa o resultado está incorreto, mas já fez vários testes de mesa e ainda assim não conseguiu solucionar o problema, anexe na sua pergunta o código ou trecho do código com problema, explique o que tentou fazer para resolver, pois facilitará que o professor entenda sua dificuldade e possa ajudá-lo.

As dúvidas serão respondidas pelos professores por meio de textos, vídeos curtos ou com agendamento de uma conversa via GoogleMeet.

Uma ferramenta que poderá ser utilizada para tirar dúvida com o monitor ou com algum professor é o editor compartilhado repl.it. Ele permite editar código de forma compartilhada, compilar e executar. Neste caso, será necessário que o estudante agende um horário com o monitor/professor para atendimento individual.

As listas de exercícios serão disponibilizadas no Dredd e com prazo final de entrega para 26/07/2020. Não é recomendável resolver os exercícios no último dia, pois o prazo não será prorrogado e a lista ficará indisponível. **A senha da lista de treino e da lista avaliativa é "reo2020".**

As notas referentes às listas desta quarta quinzena serão disponibilizadas até o dia 10/08/2020.

4. Forma de Avaliação

Nesta quarta quinzena será disponibilizada no Dredd uma lista de exercícios valendo 10% da nota. Cada exercício será avaliado quanto à corretude dos programas implementados. Cada exercício da lista avaliativa poderá ser submetido no Dredd 5 vezes. Em cada tentativa, o Dredd indicará um valor de 0 a 100 que sugere a corretude do programa implementado. Fora do Dredd, o estudante poderá testar e corrigir seu programa tantas vezes julgar necessário. Será realizada a análise de todos os programas submetidos visando detectar a existência de plágio. **Questões que forem identificadas como plágio terão a nota zerada e podem levar à criação de processo disciplinar na PRG.**

Fazer indentação adequada e dar nomes significativos para variáveis e subprogramas fazem parte de boas práticas de programação. Isso será um critério de avaliação da lista. Serão descontados pontos por indentação errada, nomes ruins, projeto ruim de passagem de parâmetros, e outras coisas excessivamente em desacordo com soluções planejadas. Tenha atenção ao enunciado para ter certeza sobre o que está sendo pedido. A falha em atender o enunciado tem sido a causa mais importante de perda de pontos.

5. Referências

Fundamentos da Programação de Computadores: algoritmos, Pascal e C/C++ - Ascencio, Ana Fernanda Gomes; Campos, Edilene Aparecida Veneruchi de. 3a edição;(Disponível na Biblioteca Virtual). Cap 10.

C++: Como programar - 5ª Edição - Deitel, Harvey M.; Deitel, Paul J. (Disponível na Biblioteca Virtual). Cap 10.

Conceitos de computação com o essencial de C++, 3ª Edição - HORSTMANN, Cay .(Disponível na Minha Biblioteca). Cap 15.

Programação de Computadores com C/C++. José Augusto N.Z. Manzano. 2014. (Disponível na Minha Biblioteca). Cap 6, Cap 7.

Projeto de algoritmos: com implementações em Pascal e C. N. Ziviani, Terceira Edição, Cengage Learning, 2010.(Disponível na Biblioteca Virtual) Cap 4.