*O código e senha dos moddles está em um .txt separado.*

**01/10 (Terça-Feira)**

Gerência de Configuração de Software (GCS):

Foi introduzida a GCS. Essa área auxilia na coordenação de um projeto ao auxiliar no controle e organização de mudanças e versões.

Tarefa no Moodle até Terça dia 08/10 às 19:00: Elencar os 18 mitos da GCS e elaborar sobre um deles. Um arquivo .txt com todos os mitos traduzidos e ajuda para acessar o livro está no GitHub.

Serão realizados seminários do dia 10/12 ao 18/02 sobre plataformas de GCS.

|  |  |
| --- | --- |
| Data | Tema |
| 10/12 | GitHub e GitLab |
| 17/12 | BitBucket e Bazaar |
| 04/02 | CVS e Selenium |
| 11/02 | Subversion e Mercurial |
| 18/02 | TFS e RedMine |

Banco de Dados (BD):

Foi revisada a matéria de BD. Essa área é útil para o armazenamento, organização e acesso de informações em imenso volume. Sistemas Gerenciadores de Banco de Dados (SGDBs) são usados como intermediários entre o usuário e o banco de dados, permitindo o emprego de estratégias mais avançadas para a definição, construção e manipulação de Bancos de Dados.

|  |
| --- |
| Vantagens de um SGDB |
| * Independência de dados |
| * Acesso eficiente |
| * Tempo reduzido no desenvolvimento de aplicações |
| * Segurança e integridade dos dados |
| * Administração de dados uniforme |
| * Acesso concorrente |
| * Recuperação contra crashes |

**02/10 (Quarta-Feira)**

Banco de Dados (BD):

Foi revisado o SQL, relembrando a Data Definition Language (DDL) e Data Manipulation Language (DML). Foi realizado um exercício (sem entrega) com a criação e manipulação de tabelas.

Um site com explicações e exemplos do SQL se encontra nesse link: [https://www.w3schools.com/sql/](https://www.w3schools.com/sql/default.asp)

Outros links úteis estão no Moodle da matéria.

Algoritmos e Estruturas de Dados (AE):

Foi introduzida a disciplina. Essa disciplina envolverá:

* Análise de algoritmos
* Tipos abstratos de dados
* Pilhas
* Filas
* Listas ordenadas
* Arvores (binárias)

Foi revisada a definição de algoritmos **(uma sequência finita de instruções não ambíguas para resolver um problema)**

Foi feita a análise matemática do custo de execução um algoritmo, onde cada instrução vale **1**, e instruções que repetem **n** vezes valem **n**, ou seja, um loop *for* com 3 instruções dentro vale **2**(inicialização) + **2n**(iteração) + **3n**(instruções dentro) ou seja, **2 + 5n**.

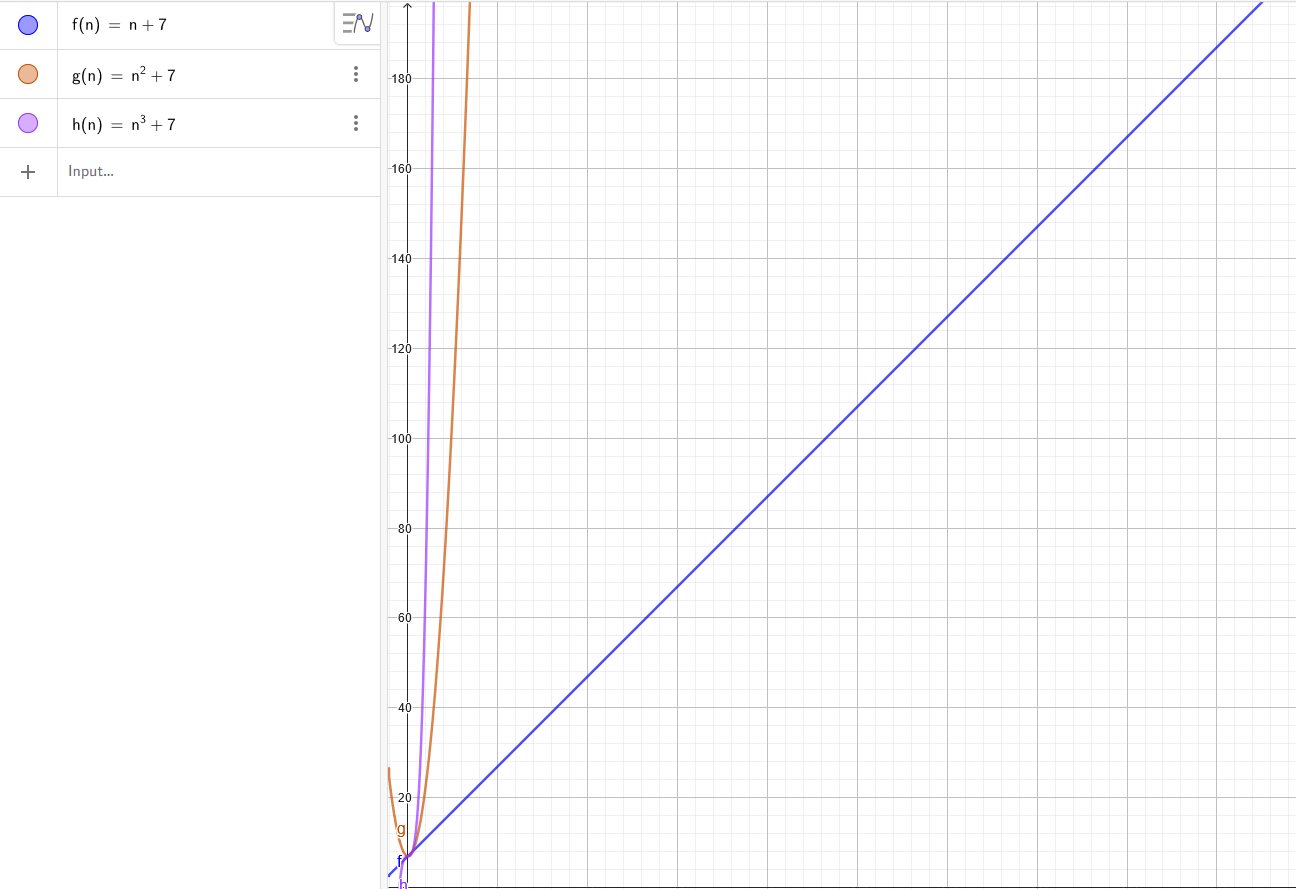
Também foi vista a possibilidade de melhores casos (execução mais rápida) e piores casos (execução mais lenta). Por exemplo, uma função que conta todas as vogais de uma String teria em seu melhor caso uma entrada vazia, e em seu pior caso com uma entrada composta exclusivamente por vogais, pois no último caso ela executaria quarto instruções (teste do for, incremento do for, teste da vogal, incremento da contagem) para cada letra.

**03/10 (Quinta-Feira)**

Algoritmos e Estruturas de Dados (AE):

Foi introduzido o conceito de comportamento assintótico. A ideia é que, com quantidades enormes de dados, variáveis constantes são inconsequentes e podem ser descartadas. Por exemplo:

|  |  |  |
| --- | --- | --- |
|  | **n = 10** | **n = 8000** |
| n + 7 | 17 | 8007 |
| n² + 7 | 107 | 64000007 |
| n³ + 7 | 1007 | 512000000007 |



Como a variável com o maior expoente é o fator determinante do custo apenas ela é usada.

A notação Big-O expande essa lógica ao assumir que, para qualquer algoritmo, ela sempre será maior ou igual ao custo, nos dando sempre o pior caso e, com ele, o limite de tempo de execução. Aplicando a notação:

|  |  |
| --- | --- |
| **Custo** | **Notação Big-o** |
| n + 7 | O(n) |
| n + n²+7 | O(n²) |
| n² + n³ + 7 | O(n³) |
| n + n² + n³ + 7 | O(n³) |

Foram mencionados outros tipos de análise assintótica:

* Notação Big-Ômega (Big-𝞨)
* Notação Big-Theta (Big-𝞡)
* Notação Pequeno-o
* Notação Pequeno-Ômega

Mais informaçõe sobre esses tipos de notações nos slides da aula.

Foram explicadas classes de complexidade de problemas comums:

* **O(1) Ordem constante:**

Independente do tamanho da entrada. Custo fixo;

* **O(log(n)) Ordem logarítmica:**

Típica de algoritmos que resolvem um problema

transformando-o em problemas menores;

* **O(n) ordem linear:**

Em geral, uma certa quantidade de operações é

realizada sobre cada elemento de entrada (um for loop);

* **O(n(log(n)) ordem log linear:**

Algoritmos que trabalham com particionamento dos dados; Resolvem um problema transformando-o em problemas menores, que são resolvidos de forma independente e depois unidos;

* **O(n²) ordem quadrática:**

Normalmente ocorre quando os dados sãoprocessados aos pares (for loop dentro de um for loop)

* **O(n³) ordem cúbica:**

É caracterizado pela presença de três estruturas de repetição (for

loop) aninhadas;

* **O(2n) ordem exponencial:**

Geralmente ocorre quando se usa uma solução de força bruta. Não são úteis do ponto de vista prático;

* **O(n!) ordem fatorial:**

Geralmente ocorre quando se usa uma solução de força bruta. Não são úteis do ponto de vista prático. Possui um comportamento muito pior que o exponencial.

|  |  |  |  |
| --- | --- | --- | --- |
| f(n) | n = 10 | n = 50 | n = 100 |
| n | 1.0E-05 segundos | 5,0E-05 segundos | 6,0E-05 segundos |
| n log n | 3,3E-05 segundos | 2,8E-04 segundos | 3,5E-04 segundos |
| n² | 1,0E-04 segundos | 2,5E-03 segundos | 3,6E-03 segundos |
| n³ | 1,0E-03 segundos | 0,13 segundos | 0,22 segundos |
| 2n | 1,0E-03 segundos | 35,7 anos | 365,6 séculos |
| 3n | 5,9E-02 segundos | 2,3E+08 seculos | 1,3E+13 séculos |

**04/10 (Sexta-Feira)**

Foi introduzido o Java, uma tecnologia para o desenvolvimento de Softwares Multiplataforma, através da Máquina Virtual do Java (jvm).

O código-fonte é salvo no formato .java, após compilado pelo Java Compiler (java), ele se torna o bytecode com formato .class, finalmente, a jvm é chamada para executar o programa.

JDK é o uma coleção de binários, bibliotecas etc para necessária para desenvolvedores de Java. Contido dele está o JRE (também baixável separadamente) que é uma coleção de APIs e contém a máquina virtual para a execução de programas Java.

IDEs, idealmente instalados depois do JDK, são programas para o desenvolvimento em java, que facilitam a criação e gerenciamento de projetos e a compilação e execução de programas. É recomendado o [NetBeans](https://netbeans.apache.org/front/main/download/index.html) ou o [IntelliJ IDEA](https://www.jetbrains.com/idea/) (Ctrl + Click para acessar o link).

Tarefa no Moodle até dia Quinta 17/10 às 23:59: Prática 1 do Java