

Análise e Desenvolvimento de Sistemas

**ALGORITMOS E ESTRUTURA DE DADOS EM PYTHON**

RELATÓRIO DE AULAS PRÁTICAS

Nome: : Igor Pereira da Silva

RA: 2537577

Polo de matrícula: São Paulo – Anchieta

Local da realização da Aula Prática: UNIP - Anchieta

Ano da postagem: 2025

Professor: Fabio de Assis

## TÍTULO DA ATIVIDADE (ROTEIRO OU AULA): IMPLEMENTANDO ALGORITMOS EM PYTHON

**ORIENTAÇÕES**:

Cada aluno (ou equipe) deve produzir um relatório curto (1 a 2 páginas) contendo:

### Resumo Teórico:

* Explicar, com palavras próprias, o que é lógica de programação e por que ela é importante.

Lógica de programação é basicamente a forma de pensar que você usa para resolver problemas passo a passo, como se estivesse dando instruções para um amigo fazer alguma coisa. Quando a gente programa, a gente precisa organizar essas instruções de um jeito que o computador entenda e consiga executar sem erro. Ela é importante porque, sem essa forma de pensar clara e organizada, o programa pode não funcionar direito, ficar confuso ou travar. Além disso, a lógica ajuda a gente a pensar melhor na vida também, porque ensina a dividir problemas grandes em partes menores, encontrar soluções e ser mais organizado.

* Mencionar brevemente o que é pseudocódigo e fluxograma e como ajudam na organização de ideias.

O **pseudocódigo** é uma maneira de descrever a lógica de um algoritmo usando uma linguagem simples e próxima do português, sem se preocupar com a sintaxe específica de uma linguagem de programação. Ele serve para **planejar passo a passo** como um problema será resolvido, tornando mais fácil entender e organizar as ideias antes de programar. O **fluxograma** é uma representação gráfica do algoritmo, usando símbolos padronizados para indicar ações, decisões e fluxos do processo. Ele ajuda a visualizar a sequência de passos e as decisões envolvidas na resolução de um problema. Ambos são ferramentas que **organizam ideias,** permitindo que o raciocínio seja estruturado de forma clara e lógica, facilitando a implementação de algoritmos e a comunicação entre pessoas envolvidas no desenvolvimento.

* Citar as vantagens de usar Python para aprender programação.

**Vantagens de usar Python para aprender programação:**Python é uma linguagem de programação amplamente recomendada para iniciantes porque possui uma sintaxe simples e clara, parecida com a linguagem natural. Isso permite que os estudantes se concentrem na lógica do programa sem se perder em detalhes complexos de sintaxe. Além disso, Python possui uma grande quantidade de bibliotecas e recursos que facilitam desde cálculos matemáticos até desenvolvimento web e manipulação de dados, tornando o aprendizado mais prático e motivador.

* Definição de algoritmo, variável, condicional, laço;

**Definição de conceitos fundamentais:**

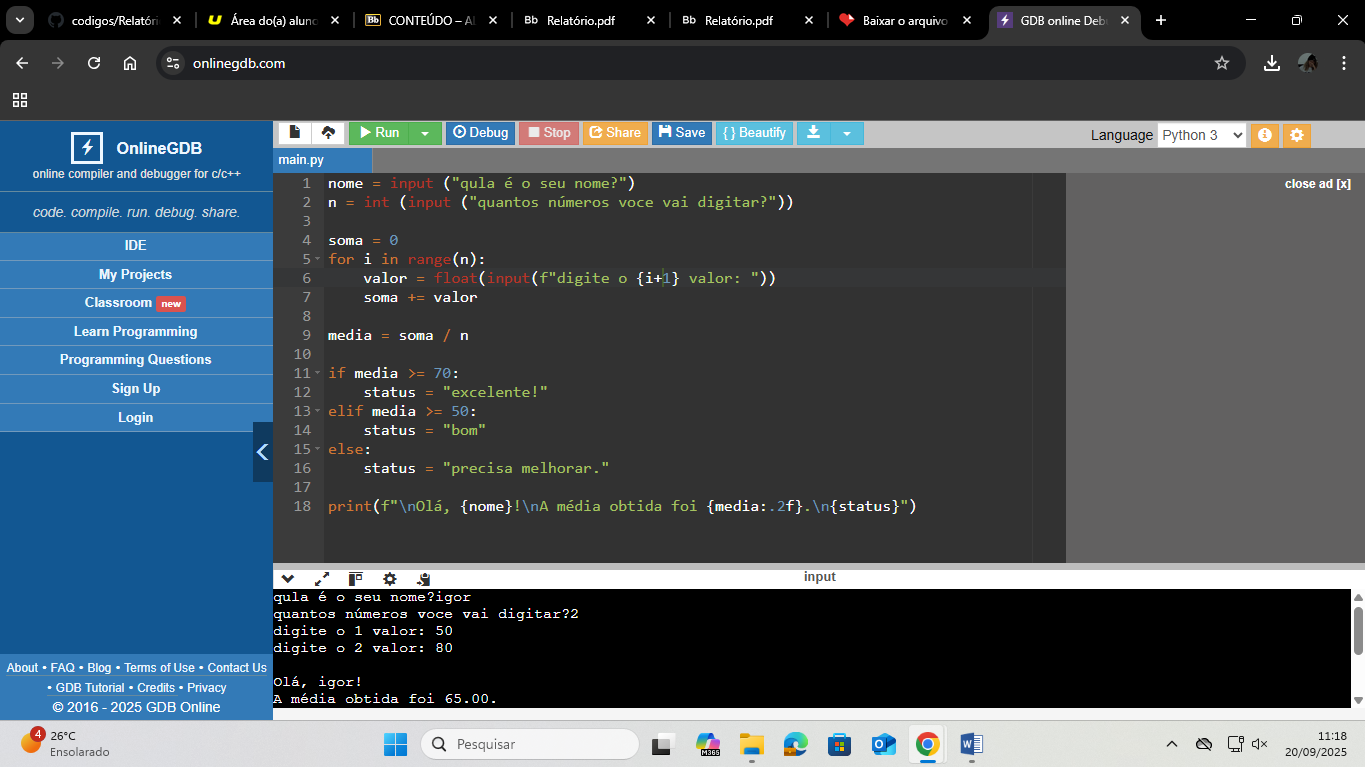
* **Algoritmo:** É uma sequência de passos lógicos e bem definidos que devem ser seguidos para resolver um problema ou realizar uma tarefa. Por exemplo, uma receita de bolo é um algoritmo, pois indica cada passo necessário para o resultado final.
* **Variável:** É um espaço na memória do computador que armazena um valor que pode ser alterado durante a execução do programa. Por exemplo, idade = 20 define uma variável chamada idade com o valor 20.
* **Condicional:** Estrutura que permite que o programa tome decisões com base em condições. Exemplo em Python:

**Definição de conceitos fundamentais:**

* **Algoritmo:** É uma sequência de passos lógicos e bem definidos que devem ser seguidos para resolver um problema ou realizar uma tarefa. Por exemplo, uma receita de bolo é um algoritmo, pois indica cada passo necessário para o resultado final.
* **Variável:** É um espaço na memória do computador que armazena um valor que pode ser alterado durante a execução do programa. Por exemplo, idade = 20 define uma variável chamada idade com o valor 20.
* **Condicional:** Estrutura que permite que o programa tome decisões com base em condições. Exemplo em Python:
* **Laço (ou loop):** Estrutura que repete um bloco de código enquanto uma condição for verdadeira, economizando esforço e evitando a repetição manual de comandos.

### Código-Fonte Comentado:

* Inserir o *código*-*fonte completo* da atividade proposta.



* Comentar as principais linhas, ressaltando o uso de print(), input(), variáveis etc.

input(): para ler dados do usuário (sempre como string).

int() e float(): para converter a entrada de texto em números inteiros e reais.

Variáveis: para armazenar informações como nome, n, soma, media, valor, status.

for: para repetir a entrada dos valores n vezes.

if, elif, else: para tomar decisões com base na média calculada.

print(): para exibir mensagens e resultados ao usuário.

F-strings (f""): para inserir variáveis diretamente dentro das strings.

**REFERÊNCIAS:**

CURSO EM VÍDEO. *Curso de Python*. Disponível em: https://www.cursoemvideo.com/course/python-3-mundo-1/. Acesso em: 20 set. 2025.

PYTHON SOFTWARE FOUNDATION. *Python documentation*. Disponível em: https://docs.python.org/3/. Acesso em: 20 set. 2025.

## TÍTULO DA ATIVIDADE (ROTEIRO OU AULA): ESTRUTURAS DE DADOS LINEARES EM PYTHON: LISTAS, PILHAS, FILAS E EFICIÊNCIA. NOTAÇÃO BIG-O

**ORIENTAÇÕES**:

Cada aluno deve produzir um relatório sucinto (2 a 3 páginas) contendo:

# Resumo Teórico

* Definição de listas, pilhas, filas e introdução à notação Big-O.

As listas em Python são estruturas de dados que armazenam elementos de forma sequencial, permitindo acesso por índice. São mutáveis, ou seja, podem ser alteradas após a criação. Internamente, são implementadas como vetores dinâmicos, o que permite acesso rápido a elementos (O(1)) e inserções/remoções com custo variável.  
A pilha é uma estrutura de dados do tipo **LIFO** (Last In, First Out), onde o último elemento inserido é o primeiro a ser removido. Operações principais:

* push: insere no topo.
* pop: remove do topo.
* peek: visualiza o topo.

**Complexidade:** Inserções e remoções ocorrem em tempo constante, O(1).

Filas (queue):

A fila é uma estrutura do tipo **FIFO** (First In, First Out), onde o primeiro elemento a entrar é o primeiro a sair. Operações principais:

* enqueue: insere no final.
* dequeue: remove do início.

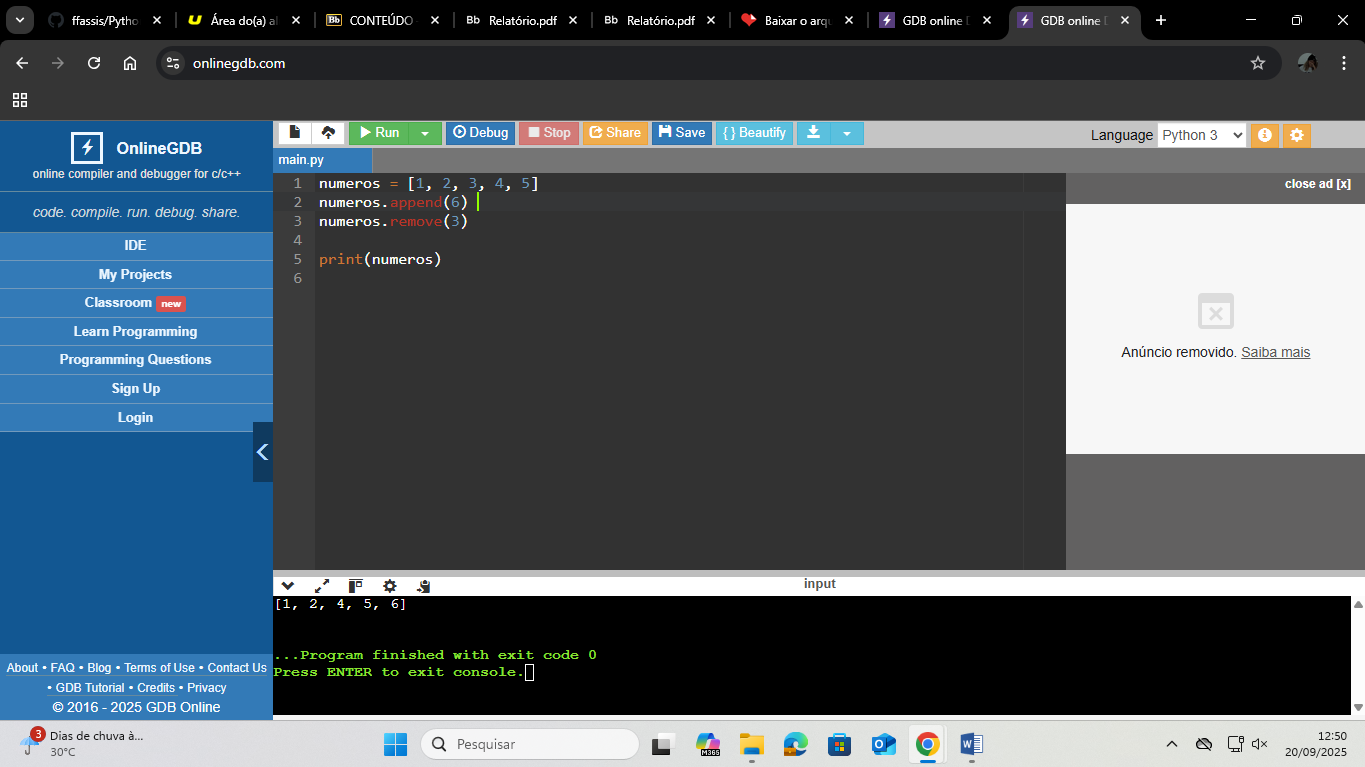
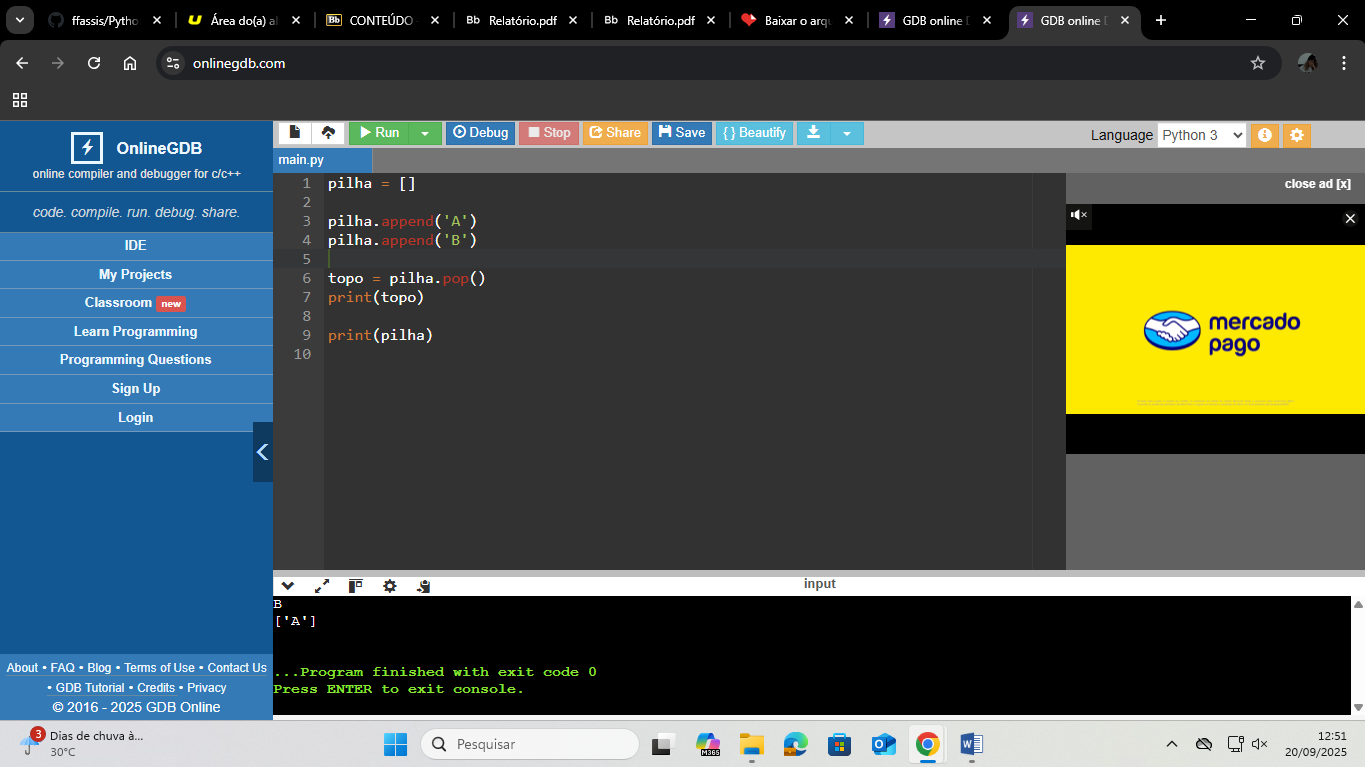
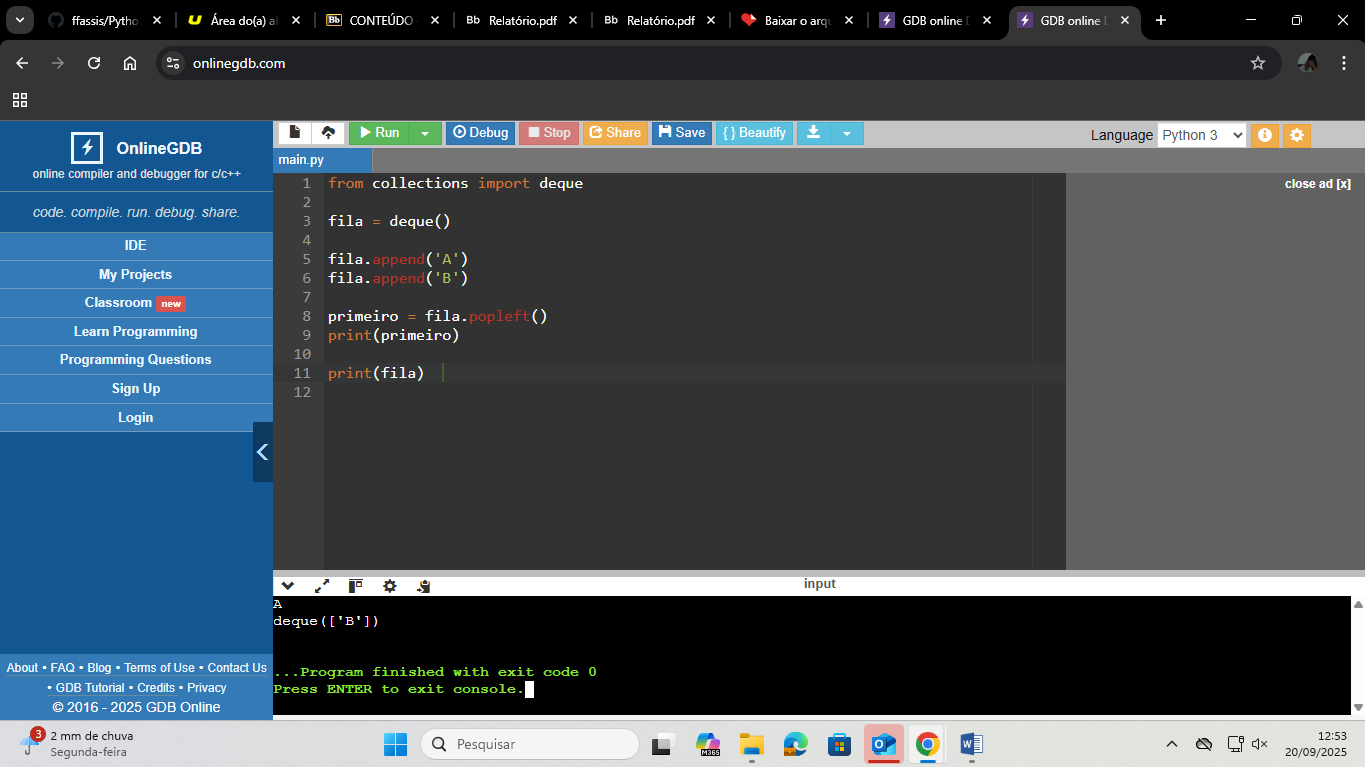
Em Python, usa-se collections.deque para implementar filas com eficiência (O(1) em inserções e remoções nas extremidades).

### Notação Big-O:

A notação Big-O descreve o comportamento assintótico de algoritmos em relação ao tempo ou espaço. Exemplos comuns:

* O(1): constante
* O(n): linear
* O(log n): logarítmica
* O(n^2): quadrática

Ela ajuda a avaliar a eficiência das estruturas e algoritmos.

* Código-fonte comentado das soluções desenvolvidas.
* Reflexão sobre desafios encontrados e critérios para selecionar cada estrutura;

Durante a implementação, um dos principais desafios foi escolher a estrutura de dados ideal para cada problema. Por exemplo, listas são versáteis, mas operações de inserção/remoção no meio são custosas (O(n)). Pilhas são úteis em algoritmos de backtracking, e filas são ideais para problemas de atendimento e simulação.

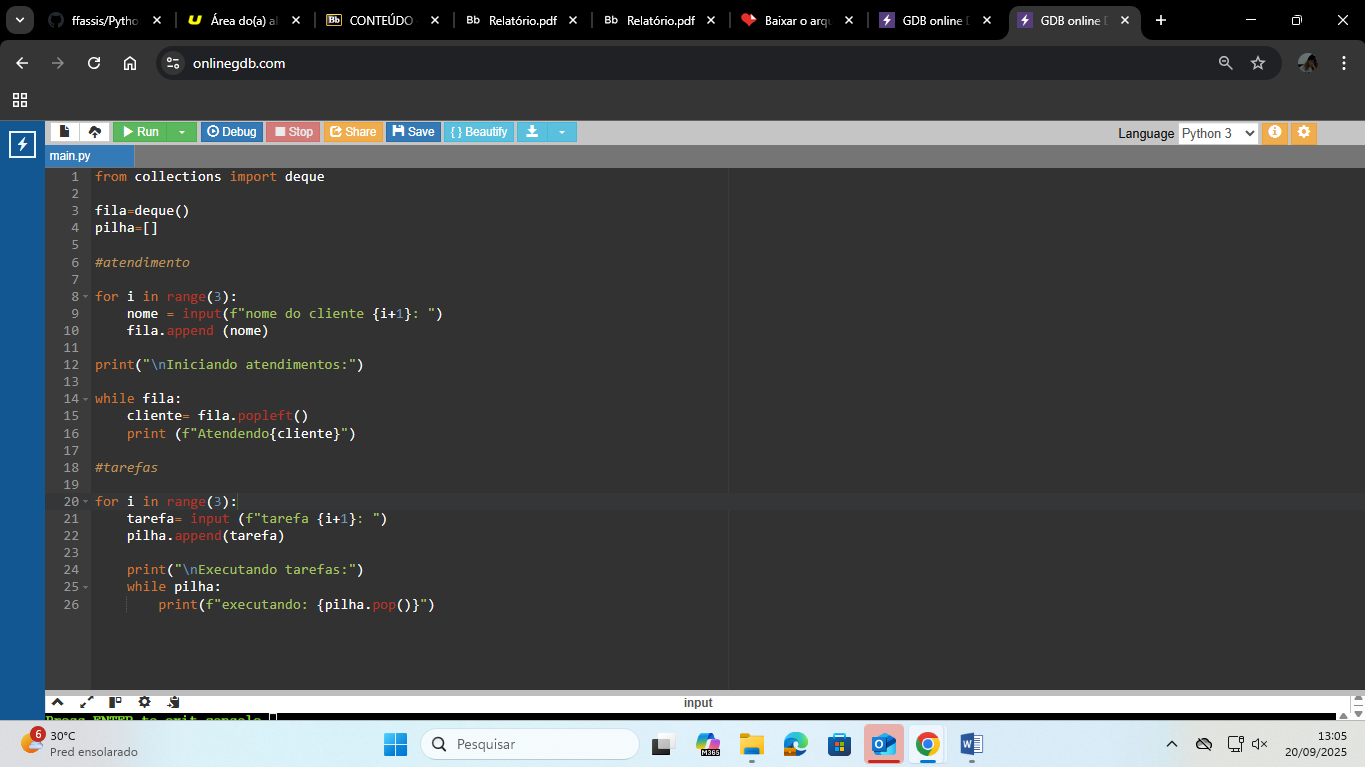
Outro desafio foi compreender a complexidade das operações. O uso da notação Big-O ajudou a prever o comportamento dos algoritmos em grande escala.

A escolha da estrutura foi guiada pelos seguintes critérios:

* **Ordem de acesso:** direto (lista), no topo (pilha), na frente (fila).
* **Complexidade das operações.**
* **Necessidade de ordenação ou reversão.**

# Códigos-Fontes Comentados

* Inserir o *código*-*fonte completo* da atividade proposta.



* Comentar as principais linhas, ressaltando o uso das estruturas de dados.

### Atendimento (Fila - FIFO)

* Utiliza deque para simular uma **fila de clientes.**
* A ordem de atendimento é a mesma ordem de chegada: o primeiro a entrar é o primeiro a ser atendido (**First In, First Out**).
* Métodos usados:
  + append() para enfileirar.
  + popleft() para desenfileirar.

### Tarefas (Pilha - LIFO)

* Utiliza uma lista (list) como **pilha de tarefas.**
* A ordem de execução é a **última tarefa adicionada é a primeira a ser executada (Last In, First Out**).
* Métodos usados:
  + append() para empilhar.
  + pop() para desempilhar.

**REFERÊNCIAS**:

ZIVIANI, Nivio. Projeto de Algoritmos com Implementações em Pascal e C. 3. ed. São Paulo: Cengage Learning, 2010.

GOODRICH, Michael T.; TAMASSIA, Roberto. Estruturas de Dados e Algoritmos em Python. Rio de Janeiro: LTC, 2016.

BEAZLEY, David. Python Essential Reference. 4. ed. Addison-Wesley, 2009.

PYTHON SOFTWARE FOUNDATION. The Python Standard Library — collections — deque. Disponível em: https://docs.python.org/3/library/collections.html#collections.deque. Acesso em: 20 set. 2025.

WIKIPEDIA. Big O notation. Disponível em: https://en.wikipedia.org/wiki/Big\_O\_notation. Acesso em: 20 set. 2025.

## TÍTULO DA ATIVIDADE (ROTEIRO OU AULA): ESTRUTURAS DE DADOS NÃO LINEARES - ÁRVORES E GRAFOS EM PYTHON

**ORIENTAÇÕES**:

Cada aluno deve produzir um relatório curto (2 a 3 páginas), contendo:

# Resumo Teórico:

* Definição de árvore e grafo, vantagens de cada estrutura.

### Definição de Árvore

Árvore é uma estrutura de dados hierárquica formada por nós chamados vértices (ou nodos), onde cada nó pode ter zero ou mais filhos, e há um nó especial chamado raiz. É usada para representar relações de hierarquia, como sistemas de arquivos, expressões matemáticas, ou árvores genealógicas.

### Definição de Grafo

Grafo é uma coleção de vértices (ou nós) conectados por arestas, que podem ser direcionadas ou não. Grafos são usados para representar redes complexas como redes sociais, mapas, rotas, circuitos, entre outros.

### Vantagens de Cada Estrutura

**Árvores:** facilitam a organização hierárquica, busca eficiente (exemplo: árvores binárias de busca), e estruturação de dados de forma ordenada.

**Grafos:** permitem modelar relações complexas entre objetos sem hierarquia fixa, com possibilidades de múltiplas conexões entre vértices.

* Comentário sobre a escolha de Python para ilustrar algoritmos de percursos;

Python é uma linguagem de alto nível, simples e expressiva, que possui diversas bibliotecas e estruturas integradas para manipulação de listas, filas e dicionários, facilitando a implementação de algoritmos de percursos em árvores e grafos. Além disso, sua sintaxe clara ajuda no entendimento conceitual dos algoritmos, o que é importante para aprendizado.

# Códigos Desenvolvidos:

* Inserir o *código*-*fonte completo* da atividade proposta.
* Comentar as principais linhas, ressaltando criação de vértices, filas, laços e condições de parada.

# Conclusão:

* Descrever se a aula ajudou no entendimento de árvores e grafos.

**REFERÊNCIAS**: O aluno deverá colocar o nome dos livros e *sites* utilizados para a realização da atividade. As regras para fazer referência ao material utilizado deverão ser de acordo com a ABNT.

## TÍTULO DA ATIVIDADE (ROTEIRO OU AULA): ALGORITMOS DE ORDENAÇÃO

### EM PYTHON (Bubble Sort, Selection Sort, Insertion Sort, Merge Sort e Quick Sort)

**ORIENTAÇÕES**:

Cada aluno deve produzir um relatório sintético (2 a 3 páginas) contendo:

# Resumo Teórico:

* Definir ordenação e justificar sua relevância em ciência da computação.
* Explicar diferenças conceituais entre algoritmos quadráticos e log- lineares.
* Comentar vantagens e limitações de cada método;

# Códigos Desenvolvidos:

* Inserir as implementações completas, destacando linhas decisivas (trocas, partições, fusões).
* Incluir tabela dos tempos obtidos nas medições.

**REFERÊNCIAS**: O aluno deverá colocar o nome dos livros e *sites* utilizados para a realização da atividade. As regras para fazer referência ao material utilizado deverão ser de acordo com a ABNT.

**CRITÉRIOS PARA AVALIAÇÃO**

|  |  |  |
| --- | --- | --- |
| **Critério** | **Peso** | **Descrição** |
| **Clareza do**  **Resumo Teórico** | 2,0 | Clareza e correlação correta entre conceitos. |
| **Organização e Comentários do**  **Código** | 3,0 | O código deve estar indentado corretamente, usar nomes de variáveis adequados e conter  comentários informativos (quando necessários). |
| **Funcionalidade do Código** | 3,0 | Execução correta e apresentação dos resultados de tempo. |
| **Criatividade e Aprimoramentos** | 2,0 | Introdução de visualização gráfica simples, pivô aleatório ou otimizações. |

## TÍTULO DA ATIVIDADE (ROTEIRO OU AULA): ALGORITMOS DE PESQUISA - BUSCA LINEAR E BUSCA BINÁRIA EM PYTHON

**ORIENTAÇÕES**:

Cada aluno deve produzir um relatório (2 a 3 páginas) contendo:

# Resumo Teórico:

* Explicar a diferença entre pesquisa exaustiva e pesquisa por divisão.
* Comparar custos de busca linear e binária em termos de complexidade e de requisitos de ordenação;

# Códigos Desenvolvidos:

* Inserir implementações completas de ambos os métodos.
* Apresentar tabela com tempos coletados para três tamanhos distintos de listas.

**REFERÊNCIAS**: O aluno deverá colocar o nome dos livros e *sites* utilizados para a realização da atividade. As regras para fazer referência ao material utilizado deverão ser de acordo com a ABNT.

**CRITÉRIOS PARA AVALIAÇÃO**

|  |  |  |
| --- | --- | --- |
| **Critério** | **Peso** | **Descrição** |
| **Clareza do Resumo**  **Teórico** | 2,0 | Clareza conceitual e uso correto de terminologia. |
| **Organização e**  **Comentários do Código** | 3,0 | O código deve estar indentado corretamente, usar nomes de variáveis adequados e conter comentários informativos (quando necessários). |
| **Funcionalidade do Código** | 3,0 | Execução correta e apresentação dos resultados de tempo. |
| **Criatividade e Aprimoramentos** | 2,0 | Execução correta e apresentação dos resultados de tempo. |

## TÍTULO DA ATIVIDADE (ROTEIRO OU AULA): TABELAS DE DISPERSÃO (HASH TABLES) E OS HEAPS EM PYTHON

**ORIENTAÇÕES**:

Cada aluno deve produzir um relatório (2 a 3 páginas) contendo:

# Resumo Teórico:

* Definir tabelas de dispersão, explicar colisões e tratamentos.
* Descrever heaps binários e justificar eficiência em filas de prioridade;

# Códigos Desenvolvidos:

* Inserir implementações das medições solicitadas, com observações sobre linhas-chave (cálculo de hash, heappush, heappop).
* Apresentar tabela dos tempos obtidos em cada experimento.

**REFERÊNCIAS**: O aluno deverá colocar o nome dos livros e *sites* utilizados para a realização da atividade. As regras para fazer referência ao material utilizado deverão ser de acordo com a ABNT.

**CRITÉRIOS PARA AVALIAÇÃO**

|  |  |  |
| --- | --- | --- |
| **Critério** | **Peso** | **Descrição** |
| **Clareza do Resumo Teórico** | 2,0 | Precisão conceitual e clareza de exposição. |
| **Organização e**  **Comentários do Código** | 3,0 | O código deve estar indentado corretamente, usar  nomes de variáveis adequados e conter comentários informativos (quando necessários). |
| **Funcionalidade do Código** | 3,0 | Execução sem erros, coleta e exibição confiável dos tempos. |
| **Criatividade e Aprimoramentos** | 2,0 | Implementação de heap de máx-prioridade, visualizações simples ou análise de fator de carga. |

## TÍTULO DA ATIVIDADE (ROTEIRO OU AULA): ALGORITMOS DE GRAFOS - DIJKSTRA, BELLMAN-FORD, KRUSKAL E PRIM EM PYTHON

**ORIENTAÇÕES**:

Cada aluno deve produzir um relatório (2 a 3 páginas) contendo:

# Resumo Teórico:

* Explicar diferenças entre caminhos mínimos de fonte única e árvores geradoras mínimas.
* Apontar condições de aplicabilidade (pesos negativos, denso × esparso);

# Códigos Desenvolvidos:

* Incluir implementações completas, indicando linhas de relaxamento e união-busca.
* Apresentar tabela de tempos e pesos totais das árvores.

**REFERÊNCIAS**: O aluno deverá colocar o nome dos livros e *sites* utilizados para a realização da atividade. As regras para fazer referência ao material utilizado deverão ser de acordo com a ABNT.

**CRITÉRIOS PARA AVALIAÇÃO**

|  |  |  |
| --- | --- | --- |
| **Critério** | **Peso** | **Descrição** |
| **Clareza do**  **Resumo Teórico** | 2,0 | Precisão conceitual e clareza de exposição. |
| **Organização e**  **Comentários do Código** | 3,0 | O código deve estar indentado corretamente, usar nomes de variáveis adequados e conter comentários informativos (quando necessários). |
| **Funcionalidade do Código** | 3,0 | Execução sem erros, resultados coerentes. |
| **Criatividade e Aprimoramentos** | 2,0 | Uso de visualizações, análise de ciclos negativos ou comparação com bibliotecas externas. |

## TÍTULO DA ATIVIDADE (ROTEIRO OU AULA): TÉCNICAS DE DIVISÃO E CONQUISTA E DE PROGRAMAÇÃO DINÂMICA EM PYTHON

**ORIENTAÇÕES**:

Cada aluno deve produzir um relatório (2 a 3 páginas) contendo:

# Resumo Teórico:

* A explanação das diferenças estruturais entre divisão e conquista e programação dinâmica.
* A justificativa dos ganhos obtidos com memoização ou tabulação nos problemas escolhidos;

# Descrição das Classes Criadas:

* A inclusão das três versões do algoritmo escolhido (recursiva simples, memoizada, bottom-up).
* A apresentação dos tempos medidos em tabela.

**REFERÊNCIAS**: O aluno deverá colocar o nome dos livros e *sites* utilizados para a realização da atividade. As regras para fazer referência ao material utilizado deverão ser de acordo com a ABNT.

**CRITÉRIOS PARA AVALIAÇÃO**

|  |  |  |
| --- | --- | --- |
| **Critério** | **Peso** | **Descrição** |
| **Clareza do Resumo**  **Teórico** | 2,0 | A inclusão das três versões do algoritmo escolhido (recursiva simples, memoizada, bottom-up). |
| **Organização e**  **Comentários do Código** | 3,0 | A apresentação dos tempos medidos em tabela. |
| **Funcionalidade do Código** | 3,0 | A inclusão das três versões do algoritmo escolhido (recursiva simples, memoizada, bottom-up). |
| **Criatividade e Aprimoramentos** | 2,0 | Análise gráfica, discussão sobre consumo de memória ou casos extremos.. |