PUC Minas

Trabalho individual 3 - Valor 10 pontos

INFORMAÇÕES DOCENTE						
CURSO: ENGENHARIA DE SOFTWARE	**DISCIPLINA:** FUNDAMENTOS DE PROJETO E ANÁLISE DE ALGORITMOS	**TURNO**	MANHÃ	TARDE	NOITE	**PERÍODO/SALA:** 5º
					x	
PROFESSOR (A): João Paulo Carneiro Aramuni						

Enunciado do projeto: **Implementação do Algoritmo para Caminho Hamiltoniano em Python**

Objetivo:

- Desenvolver um programa em Python que implemente o algoritmo para encontrar um Caminho Hamiltoniano em um grafo orientado ou não orientado. O projeto deverá ser entregue por meio de um link para o repositório do GitHub no CANVAS.

Sobre o algoritmo:

- Um Caminho Hamiltoniano em um grafo é um caminho que visita cada vértice exatamente uma vez. Encontrar esse caminho é um problema clássico em teoria dos grafos e está associado a problemas de alta complexidade computacional, como o Problema do Caixeiro Viajante. Este projeto tem como objetivo implementar uma abordagem para determinar se um Caminho Hamiltoniano existe em um grafo e, em caso afirmativo, encontrá-lo.

Requisitos do projeto:

1. Código Python:
   - O programa deverá conter a implementação do algoritmo para encontrar um Caminho Hamiltoniano em um arquivo chamado main.py.

2. Documentação no README.md:
   - O repositório deverá incluir um arquivo README.md que explique como rodar o projeto e também a lógica do algoritmo implementado.
   - O README deverá ser estruturado conforme o exemplo fornecido neste *repo*: *https://github.com/joaopauloaramuni/fundamentos-de-projeto-e-analise-de-algoritmos/tree/main/PROJETOS*
   - O README deverá conter:
     - Descrição do projeto: Explicação do algoritmo e da lógica de como ele foi implementado (linha a linha).

- Como executar o projeto: Instruções para rodar o código no ambiente local.
- Relatório técnico contendo as análises sobre o algoritmo.

3. Relatório técnico incorporado ao README:
   - Análise da complexidade computacional:
     - Classes P, NP, NP-Completo e NP-Difícil:
       - 1. Explique em quais classes de complexidade (P, NP, NP-Completo e NP-Difícil) o problema do Caminho Hamiltoniano se enquadra.
       - 2. Justifique sua resposta com base nas características dessas classes e na relação com o Problema do Caixeiro Viajante.
       - Exemplos:
         - AULA 02_Introdução à teoria da complexidade.pdf
         - *https://github.com/joaopauloaramuni/fundamentos-de-projeto-e-analise-de-algoritmos/tree/main/PDF*

     - Análise da complexidade assintótica de tempo:
       - 1. Determine a complexidade temporal do algoritmo.
       - 2. Explique como essa complexidade foi determinada, indicando o método utilizado (contagem de operações, expansão de recorrência, Teorema Mestre, etc.).

     - Aplicação do Teorema Mestre:
       - Verifique se é possível aplicar o Teorema Mestre ao problema ou ao algoritmo fornecido.
       - Justifique sua resposta explicando os critérios de aplicabilidade do Teorema Mestre.

     - Análise dos casos de complexidade:
       - 1. Explique as diferenças entre os casos de complexidade: pior caso, caso médio e melhor caso para o algoritmo utilizado.
       - 2. Analise como essas diferenças impactam o desempenho do algoritmo.

4. Entrega:
   - O projeto deverá ser enviado por meio de um repositório no GitHub, com o link postado no sistema CANVAS. Certifique-se de que o repositório esteja público ou acessível (antes de realizar a entrega, faça um teste em uma aba anônima do navegador).

- Exemplo de link a ser entregue no CANVAS:
  - https://github.com/exemploaluno/trabalho_individual_2_FPAA

Critérios de avaliação:

1. Implementação do algoritmo (50%):
   - O código está correto e eficiente?
   - A lógica do algoritmo para encontrar o Caminho Hamiltoniano foi seguida adequadamente?
2. Documentação no README.md (50%):
   - O README segue o padrão especificado?
   - O relatório técnico está claro e apresenta uma análise detalhada?
   - A análise da complexidade assintótica, bem como a definição da classe P, NP, NP-Completo e NP-Difícil, está correta e bem explicada?

Dicas para o desenvolvimento:

- Comece implementando um algoritmo básico de backtracking para encontrar Caminho Hamiltoniano.
- Teste o algoritmo com grafos pequenos para verificar se funciona corretamente.
- Consulte o material e leia a aula sobre complexidade assintótica e Teorema Mestre para enriquecer o relatório.
  - AULA 02_Introdução à teoria da complexidade.pdf
  - *https://github.com/joaopauloaramuni/fundamentos-de-projeto-e-analise-de-algoritmos/tree/main/PDF*

Ponto extra (Opcional):

- Visualize o Caminho Hamiltoniano utilizando uma biblioteca Python de grafos:
  - Use bibliotecas como NetworkX e Matplotlib para desenhar o grafo e destacar o Caminho Hamiltoniano (se encontrado).
  - Requisitos:
    - 1. Desenhe o grafo original: Inclua todos os nós e arestas com etiquetas para identificação.
    - 2. Destaque o Caminho Hamiltoniano: As arestas do caminho encontrado devem ser destacadas com uma cor ou estilo diferente.
    - 3. Adicione a função de visualização ao código Python separadamente em outro arquivo .py (exemplo view.py).
- Exporte a imagem gerada como PNG e inclua no repositório do GitHub na pasta assets.
- Inclua no README.md as instruções para instalar as bibliotecas necessárias e um exemplo do grafo visualizado.