Universidade Federal de Ouro Preto
PCC104 - Projeto e Análise de Algoritmos
Prova 1

Prof. Rodrigo Silva

October 1, 2023

Orientações

- É obrigatória a entrega do código fonte dos algoritmos implementados. Provas sem os códigos fonte não serão corrigidas e terão nota 0.
- A avaliação do código apresentado entra na avaliação das questões relacionadas.
- Simplificações feitas na análise de custo dos algoritmos devem ser indicadas e justificadas.

Questões

1. (1 pt) Quando devemos utilizar a estratégia de busca exautiva?
2. Considere a sua implementação do algoritmo *SelectionSort*
   (a) (2 pts) Apresente a expressão matemática que define o custo (em comparações).Explique o que representa e de onde saiu cada um dos termos da sua expressão.
   (b) (2 pts) Resolva a expressão para encontrar a classe de complexidade e determine a classe de complexidade em notação O ou Θ.
3. (1 pt) Qual a vantagem do algoritmo *SequentialSearch2* sobre uma implementação mais inocente de uma busca sequêncial? Explique.
4. Considere a sua implementação do algoritmo de busca em profundidade.
   (a) (2 pts) Defina a operação básica e apresente a expressão matemática que define o custo da sua implementação. Explique o que representa e de onde saiu cada um dos termos da sua expressão.
   (b) (2 pts) Resolva a expressão para encontrar a classe de complexidade e determine a classe de complexidade em notação O ou Θ.

1 Solução

1. -
2. -
3. -
4.

```python
def dfs(graph, start_vertex, visited=set()):
    visited.add(start_vertex)
    print(start_vertex, end=' ')

    for neighbor in graph[start_vertex]:
        if neighbor not in visited:
            dfs(graph, neighbor, visited)

# Example usage:
# Define an adjacency list representation of the graph
graph = {
    'A': ['B', 'C'],
    'B': ['A', 'D', 'E'],
    'C': ['A', 'F'],
    'D': ['B'],
    'E': ['B', 'F'],
    'F': ['C', 'E']
}

start_vertex = 'A'
print("DFS traversal starting from vertex 'A':")
dfs(graph, start_vertex)
```

Figure 1: Algoritmo DFS