Relatório de Análise e Projeto de Algoritmos - Lista 02

Aluno - Balthazar Paixão

Repositório Github

Códigos da lista

Temas:

```
- AVL Tree
- Hash Table
- Graph
```

Exercício 1 - AVL Tree

Faça um programa que leia um arquivo texto (em .txt) e imprima, em ordem alfabética, as palavras e a suas frequências no texto, além de indicadores de performance dos algoritmos implementados.

A leitura do arquivo deverá desprezar espaços em branco e sinais de pontuação, que serão considerados separadores de palavras. Além disso, a leitura deverá converter todas as letras maiúsculas em minúsculas.

A pesquisa e inserção das palavras do texto deverão ser implementadas com as seguintes estruturas:

1. Pesquisa Binária (utilizando um vetor dinâmico para armazenar as palavras).
2. Árvore Binária de Pesquisa sem balanceamento.
3. Árvore Binária de Pesquisa com balanceamento (Árvore AVL).

Resposta: O codigo para a resolução do exercício se encontra no arquivo 01-avl_tree.py. Para rodar o código basta executar o comando `python3 01-avl_tree.py lorem_ipsum.txt method` no terminalsendo o parametro method um entre as 3 opções `binary_search`, `avl_tree_unbalanced` e `avl_tree_balanced`. Não está funcionando.

Exercício 2 - Hash Table

2.1. Observando cada um dos caracteres que formam a palavra P E S Q U I S A. Crie uma tabela de dispersão (hash table) que armazene cada um destes caracteres, utilizando a seguinte função:

h(char) = (ordem de char no alfabeto)%7

Desenhe o esquema de como ficaram a hash table e as listas encadeadas.

Resposta: O codigo para a resolução do exercício se encontra no arquivo 01-hash_table.py. Para rodar o código basta executar o comando `python3 02-hash_table.py` no terminal. Dessa forma ele irá mostrar como ficou a tabela hash.

2.2. Implemente uma classe Hash respeitando o seguinte:

- a) Utilize uma estrutura de tabela Hash para armazenar os elementos que devem ser inseridos e consultados com complexidade O(1).
- b) A estrutura criada deve permitir o armazenamento de elementos de qualquer tipo. Elementos esses identificados por uma chave do tipo string.

Resposta: Não implementado.

Exercício 3 - Graph

3.1. Faça os itens a seguir:

- a) Desenhe um grafo que tenha o conjunto de vértice V={1,2,3,4,5}, o conjunto de arestas E={a1,a2,a3,a4,a5,a6} e a função g(a1)={1,2}, g(a2={1,3}, g(a3)={3,4}, g(a4)={3,4}, g(a5)={4,5} e g(a6)={5,5}.
- b) Encontre no grafo do item (a): i) dois vértices não adjacentes; ii) um laço; iii) o grau do vértice 3.
- c) Encontre um isomorfismo do grafo apresentado no item (a)
- d) Apresente a matriz e lista de adjacência do grafo do item (a)

Resposta: O codigo para a resolução do exercício se encontra no arquivo 01-graph.py. Para rodar o código basta executar o comando `python3 01-graph.py` no terminal. Dessa forma ele irá apresentar os resultados das questões e salvar a imagem do grafo em um arquivo na pasta `./imgs/`.

3.2.

Um grafo G é planar se existe uma representação (desenho, imersão) de G no plano de modo que as áreas se encontrem somente nos vértices, isto é, de modo que as arestas não se cruzem. Um grafo planar divide o plano em regiões chamadas faces. Existe sempre uma única face chamada externa ou infinita, que não está limitada (i.e, tem área infinita). Demonstre a Fórmula de Euler: Em um grafo conexo planar com f faces, n vértices e m arestas, vale f=m-n+2.

Resposta: Não feito.

3.3

Explique o que são grafos eulerianos e hamiltonianos. Dê exemplos e propriedades de cada um.

Resposta: Grafos eulerianos são grafos que possuem um ciclo euleriano, ou seja, um ciclo que passa por todas as arestas do grafo. Enquanto grafos hamiltonianos são grafos que possuem um ciclo hamiltoniano, ou seja, um ciclo que passa por todos os vértices do grafo. Algumas propriedades são que todo grafo euleriano é hamiltoniano, mas nem todo grafo hamiltoniano é euleriano. Além disso, todo grafo euleriano possui todos os vértices com grau par, enquanto que um grafo hamiltoniano pode possuir vértices com grau ímpar.

3.4

Quando trabalhamos com grafos admitimos que algumas operações básicas já estejam implementadas. Então nossa primeira tarefa é implementar as seguintes operações básicas tanto para uma representação por matriz de adjacências quanto para lista de adjacências:

- a) Definição do Grafo
- b) Adição de Arestas
- c) Determinação de vizinhos de um dado vértice
- d) Teste de vizinhança entre par de vértices
- e) Remoção de arestas

Resposta: O codigo para a resolução do exercício se encontra no arquivo 04-graph.py. Para rodar o código basta executar o comando `python3 04-graph.py` no terminal. Dessa forma ele irá apresentar um exemplo de execução das funções implementadas.

3.5

O processo de busca é comum em diversas áreas de computação. Um algoritmo clássico para realizar buscas em estruturas de dados representadas por grafos é o chamado Algoritmo de Busca em Profundidade. Implemente tal algoritmo.

Resposta: O codigo para a resolução do exercício se encontra no arquivo 05-graph.py. Para rodar o código basta executar o comando `python3 05-graph.py` no terminal. Dessa forma ele irá apresentar um exemplo de execução das funções implementadas.