TP 06

Ana Neri

24 de Abril 2023

Exercícios práticos sobre estruturas de dados e grafos.

Exercício 1

Uma forma bastante eficiente de representar grafos é à custa de tabelas de Hash, particularmente se usarmos Python (via dicionários).

Implemente uma classe de Grafo_pesado em Python. Crie funções para adicionar elementos, consultar os pesos, devolver o grau de entrada e de saía.

Dica: As chaves do dicionário podem ser vistas como nós, os valores em si podem ser dicionários.

Exercício 2 Um grafo orientado e pesado (com pesos > 0) é descrito pela matriz:

$$\begin{bmatrix} 0 & 3 & 0 & 0 & 0 & 2 \\ 0 & 0 & 4 & 0 & 0 & 0 \\ 0 & 0 & 0 & 9 & 0 & 0 \\ 0 & 0 & 0 & 0 & 7 & 3 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 8 & 0 \end{bmatrix}$$

Crie uma função `matToList` que transforma representação em matriz de adjacência para uma representação em listas de adjacência.

De seguida crie uma função `listToVect` que transforma a representação de listas de adjacências para uma representação em vetores de adjacência.

Exercício 3 Para um grafo implementado com listas de adjacência escreva uma função que conta o número de arestas no grafo.

Analise a complexidade da sua função em função do tamanho da representação do grafo. É costume desdobrar este tamanho em dois itens: o número de vértices (V) e o número de arestas (E)

Exercício 4

As tabelas de hash podem evitar colisões com uma estratégia de open addressing.

Implemente uma tabela de Hash em Python que consiga gerir colisões e suporte as funções insere(), procura() e apaga(). Use linear probing (dica).

Sugestão de exercícios para alunos que não estão interessados no modo fácil: Use quadratic probing ou double hashing.