Universidade Federal de Santa Catarina
Departamento de Informatica e Estatistica - INE
Ciências da Computacão - CCO
ATIVIDADE 1 Atividade Pratica A1 – Grafos (INE5413)

Prof. Rafael de Santanna

Estudantes:
Gabriel da Silva Cardoso (20100524)
Hans Buss Heidemann (19100528)

Representação

Decidimos estar utilizando Python para a implementação do trabalho tendo em vista a facilidade em se estar fazendo ações utilizando listas, bastante usadas em nossa aplicação.

Em relação a nossa implementação da classe Grafo, sendo ele um grafo não-dirigido e ponderado, criamos uma classe Graph que recebe como argumento um booleano que indica se o grafo é dirigido ou não. Além disso a classe possui como atributos duas listas, uma sendo a dos vertices e uma sendo a dos seus pesos.

Criamos também duas classes auxiliares, a classe Vertice e a Classe edges. A classe vertice possui o seu nome e uma lista de edges e a classe edge possui seu peso e seus dois vertices.

Além das funções pedidas implementamos também funções auxiliares para garantir uma melhor visibilidade durante a leitura do código, por exemplo as funções vertices_to_index() e queue_to_index().

Algoritmo 1 - Breadth-First Search

No algoritmo de busca Breadth-First Search utilizamos três estruturas para a sua implementação. No começo criamos uma lista para guardarmos os vertices que já foram visitados, inicializando com o vertice do index passado para a função. Além disso criamos uma fila com todos os vizinhos daquele determinado index. E por fim declaramos uma variavel depth para guardarmos em qual nivel estamos.

Algoritmo 2 - Hierholzer Algorithm

Neste algoritmo utilizamos basicamente as estruturas descritas nas funções pedidas para obtermos uma lista de edges que já foram visitados.

Utilizamos as funções implementadas pelas classes Graph, Vertice e Edge para fazermos a verificação da existe de um circlo Euleriano, sendo facil utilizando elas saber se existem edges não visitados e quais os vertices conectados a eles.

Algoritmo 3 - Dijikstra

No algoritmo de Dijikstra temos estruturas parecidas aos anteriores, temos uma lista que recebe quais nodos ainda não foram visitados. Também temos dois dicionarios, um D, para salvar o custo de visita a cada nodo, e um A, para salvar qual o caminho mais rapido encontrado até o momento para um nodo.

No dicionario D setamos todos os custo como sendo o maior permitido pelo sistema com sys.maxsize e setamos o custo inicial como 0.

Algoritmo 4 - FloydWarshall

Neste algoritmo criamos uma matrix de valores utilizando dicionarios ao inves de lista com o objetivo de diminuir a complexidade de memoria, além disso o algoritmo utiliza as mesmas estruturas utilizadas nos algoritmos anteriores.