Atividade Prática 1 - Grafos (INE 5413)

Gian Carlo F. Ferrari

Estruturas de Dados Utilizadas e Justificativas

1. **[Representação]**

Grafos são classes Python, com três atributos: vértices, arestas e pesos. As estruturas de dados básicas são: frozenset para os vértices, frozenset com elementos de tipo frozenset para as arestas, e um dicionário para a função de pesos. Aqui o objetivo foi aproximar a implementação da definição matemática vista em aula. Representações como lista de adjacências e a matriz de adjacência são construídas no inicializador (__init__) de um objeto Grafo.

O inicializador também aloca listas para que os métodos de consulta (grau(), rotulo() etc.) tenham tempo O(1), devido a serem apenas acesso à arrays em memória. De certa forma, usa-se mais memória (e aumenta-se a complexidade do inicializador) para que os métodos em si sejam rápidos.

2. **[Buscas]**

Aqui as estruturas de dados são as mesmas das vistas em aula e presentes nas notas, pois apenas foi-se implementado o algoritmo de busca em largura. Aqui no entanto deve-se notar que as estruturas operam sobre índices corrigidos por -1, pois o enunciado indica que a contagem dos índices dos vértices começa em 1. Estas estruturas estão presentes como listas python.

3. **[Ciclo Euleriano]**

Devido a dificuldades ao implementar o algoritmo de Hierholzer como disponível nas notas, um outro algoritmo foi implementado. O algoritmo em questão tem dois métodos: um inicial, que verifica a paridade dos graus dos vértices e chama um método recursivo caso seja possível haver ciclo euleriano. O método recursivo utiliza-se de um vértice de origem, um vértice atual e de duas listas: uma para o caminho em termos de vértices visitados sucessivamente, e outra que anota quais arestas já foram visitadas.

O método parte do vértice origem O, que inicialmente é também o vértice visitado V, e então percorre as arestas, verificando se existe alguma não visitada que contém V. Se encontra, esta aresta é adicionada numa lista “visited”, e o V é adicionado numa lista “path”. O método recursivo é então

chamado neste momento, porém com o vértice N que é vizinho de V por esta aresta.

Se todas as arestas foram visitadas, ou se o vértice V da chamada atual não está em nenhuma das não visitadas, o método faz um backtracking, retirando V da lista de caminhos e retornando. O que se tem no final é um caminho partindo do vértice O, representado na lista “path”, que tem todas as arestas visitadas (visited = arestas). Caso não encontre um caminho, todas as chamadas recursivas vão retornar com visited != arestas e portanto o “path” "resultante será vazio.

4. **[Bellman-Ford ou Dijkstra]**

Aqui foi-se implementado o algoritmo de Bellman-Ford como visto em aula e presente nas notas, devido à sua corretude e disponibilidade. Dessa forma, as estruturas de dados são análogas, considerando no entanto que a indexação é feita pelo índice dos vértices novamente corrigidos por -1. Para as distâncias iniciais, usou-se float(‘inf’) do python, e as estruturas estão como listas.

5. **[Floyd-Warshall]**

Analogamente, foi implementado o algoritmo visto em aula e presente nas notas, também devido à sua corretude e disponibilidade, com estruturas de dados semelhantes (novamente listas python). Aqui também há correção dos índices dos vértices, e valores infinitos foram representados como float(‘inf’).