INE5413 - Grafos

Relatório Atividade Prática 1

Alunas: Jéssica Regina dos Santos e Myllena da Conceição Correa

---

Neste relatório, descrevemos as estruturas de dados utilizadas na implementação da Atividade Prática 1.

1. Refere-se ao arquivo main2.py. Utilizamos uma matriz de adjacência para representar o grafo. Como o grafo é ponderado, o valor de cada célula representa o peso entre dois vértices conectados por uma aresta. A matriz de adjacência foi escolhida pela sua simplicidade, eficiência em armazenar e acessar arestas, e complexidade de acesso O(1). Os vértices do grafo foram armazenados em um dicionário, onde a chave é o índice do vértice e o valor é o rótulo correspondente. Criar um dicionário de vértices permitiu o mapeamento rápido entre o índice e o rótulo do vértice, com complexidade de busca O(1). Armazenamos as arestas do grafo em uma lista, onde cada elemento é uma tupla representando a conexão entre dois vértices. A lista de arestas facilita a iteração sobre as conexões entre vértices, sendo útil nos algoritmos que exigem percorrer todas as arestas.

2. Refere-se ao arquivo A1_2.py. Para o algoritmo de Busca em Largura, utilizamos uma fila para gerenciar os vértices a serem explorados. A fila é uma estrutura essencial já que o algoritmo segue a estratégia de exploração em níveis (onde os vértices vizinhos são visitados antes de seguir para os vértices seguintes). Durante a execução do algoritmo, as distâncias dos vértices a partir do vértice inicial são armazenadas em um dicionário, onde a chave é o vértice e o valor é a distância.

3. Refere-se ao arquivo A1_3.py. Para o algoritmo de Hierholzer, que é usado para encontrar um ciclo Euleriano em um grafo (ciclo que visita todas as arestas do grafo exatamente uma vez), utilizamos listas e dicionário. A lista de arestas “E” armazena todas as arestas do grafo como tuplas de vértices (foi eficiente para armazenar e percorrer as conexões do grafo ao longo da execução do algoritmo, permitindo verificar e marcar o estado de cada aresta durante o processo de busca). A lista “ciclo” é utilizada para armazenar o ciclo Euleriano construído durante a execução do algoritmo (foi eficiente para armazenar a sequência de vértices do ciclo Euleriano, pois permite fácil inserção de novos vértices e subsequentes subciclos, mantendo a ordem correta dos vértices no ciclo). A lista de vértices “V” contém todos os vértices do grafo e é utilizada para verificar possíveis conexões e continuar a busca de subciclos (facilitou a iteração sobre todos os vértices do grafo para verificar quais vizinhos ainda podem ser explorados). O dicionário “C” armazena o estado de cada aresta, onde a chave é uma aresta representada como tupla, e o valor indica se a aresta já foi visitada ou não.

4. Refere-se ao arquivo A1_4.py. As estruturas de dados utilizadas foram listas e dicionários novamente. A lista de arestas “E” armazena todas as arestas do grafo como tuplas de vértices (permite percorrer todas as conexões do grafo durante o processo de relaxamento, o que é necessário para calcular as menores distâncias entre os vértices.). A lista “V” contém todos os vértices do grafo, ela é usada para garantir que todas as operações de relaxamento sejam aplicadas a cada vértice. O dicionário “Dv” armazena a menor distância conhecida de cada vértice para o vértice de origem, as chaves são os vértices e os valores são as distâncias (permite a atualização e consulta da menor distância de cada vértice durante o processo de relaxamento). Já o dicionário “Av” guarda o antecessor de cada vértice no caminho de menor distância desde o vértice de origem. Ele é necessário

para reconstruir o caminho percorrido pelo algoritmo de Bellman-Ford, permitindo que o caminho seja reconstruído a partir de qualquer vértice, bastando seguir os antecessores até o vértice de origem.