

### Atividade I

1 - Grafo) Para a primeira questão, que foi a elaboração da classe que representaria o grafo, utilizou-se duas estruturas de dados, uma delas sendo uma matriz de adjacências e outra uma lista com os rótulos dos vértices, nesta última estrutura, o índice da lista representa o vértice e o valor na posição o seu rótulo, além disso, também utilizou-se uma variável contendo o número de arestas do grafo. A estrutura de matriz foi escolhida pois ela tem complexidade  $O(1)$  para todas as seguintes operações exigidas: "qtdVertices", "qtdArestas", "rotulo", "haAresta" e "peso", para as demais operações, "grau" e "vizinhos", a complexidade é  $O(|V|)$ , onde  $|V|$  é o número de vértices, apesar de estas duas últimas operações terem complexidade maior, elas não são muito inferiores às suas versões realizadas em uma lista de adjacências, já que mesmo nesta, a custo da operação no pior caso também é  $O(|V|)$ .

2 – Algoritmo de Busca em largura) A segunda questão não exigiu muita criatividade quanto ao uso de estruturas de dados, visto que o acesso as estruturas  $C_v$  (visitados),  $D_v$  (distâncias) e  $A_v$  (antecessores) é feito sempre através de índices, bastou utilizar o número do grafo como índice, assim, a complexidade para todas as operações realizadas neste algoritmo é  $O(1)$ , além disso, também foi utilizada uma pilha, que é própria do algoritmo, cuja complexidade de inserção e remoção também é  $O(1)$ .

3 – Algoritmo de Hierholzer) Neste algoritmo foi utilizada uma matriz indicar que arestas já haviam sido visitadas, esta opção consome muita memória, porém tem complexidade  $O(1)$  tanto para alterações quanto para consultas a arestas específicas, portanto é muito eficiente. Para representar o ciclo, foi utilizada uma lista.

4 – Algoritmo de Dijkstra) Nesta questão, utilizou-se uma estrutura de heap binária para mais facilmente encontrar o vértice cuja distância da origem é a menor, como estrutura auxiliar, também foi utilizada uma lista que mapeia de  $D_v$  (lista de distâncias) para a lista contendo a heap, de forma que seja possível manter a coerência entre elas. A complexidade da remoção do vertice de menor distância é  $O(\log n)$ , que é mais eficiente do que  $O(n)$  que seria a complexidade caso se utiliza-se uma lista simples.

5 – Algoritmo de Floyd-Warshall) Este algoritmo não requer nenhuma estrutura além da matriz que é própria dele, ainda assim, foi possível fazer a seguinte otimização: Considerando que nunca é feita uma referência a matriz  $d^{(k-2)}$  não há necessidade de se manter uma referência a todas as matrizes já criadas, portanto foi possível utilizar apenas duas matrizes, uma delas mantém a matriz  $d^{(k)}$  e a outra a matriz  $d^{(k-1)}$ , após uma iteração do primeiro laço, a matriz  $d^{(k-1)}$  recebe o valor da matriz  $d^{(k)}$  e a matriz  $d^{(k)}$  é reinicializada.