UNIVERSIDADE FEDERAL DE SANTA CATARINA

CAMPUS Trindade

DEPARTAMENTO INE

CURSO Ciências da Computação

Carlos Eduardo Vitorino Gomes (23150560)

Eduardo Cunha Cabral (23150561)

Enrico Caliolo (23150562)

Grafos: AT1

Introdução e Dificuldades Encontradas

Fizemos nosso trabalho utilizando a IDE VSCode e Python em diferentes versões, dentre elas, 3.12.3 e 3.6.15. Localmente, em diferentes computadores e sistemas operacionais, todos os arquivos rodam corretamente, porém, ao testar esses mesmos arquivos no Moodle (alterando apenas o método com que a entrada é recebida), o desafio de número 3 apresentou problemas, quando o teste referente ao arquivo ContemCicloEuleriano2.net é executado, o erro "[invalid literal for int() with base 10" é retornado, importante notar que fizemos diversos testes com esse arquivo no VSCode e em nenhum momento recebemos esse erro, nesses casos, o código executa sem problemas assim como todos os outros códigos que fizemos para essa atividade.

1) GRAFO NÃO-DIRIGIDO E PONDERADO

O grafo é representado como uma lista de adjacências. Armazenamos seus vértices e suas arestas em hash tables específicas. Por ser não-dirigido, se o vértice *v* tem um vizinho *u*, o vértice *u* também tem um vizinho *v*, porém há apenas uma aresta *(v,u)* ou *(u,v)*. O arquivo de entrada define a ordem da aresta, que não impactará nos algoritmos, pois eles lidam com isso.

2) BUSCA EM LARGURA

Para a busca em largura, usamos uma hash table para manter as informações específicas de cada vértice, como distância, se o vértice é conhecido e qual seu antecessor. Utilizamos essa estrutura, pois é mais confiável usar o index do vértice como o valor de hash do que usar o index de uma lista.

3) CICLO EULERIANO

Para o ciclo euleriano, usamos uma hash table que armazena quais arestas já foram visitadas em determinado momento. Utilizamos essa estrutura pela mesma razão que utilizamos na pergunta número 2 (Busca em Largura).

4) CAMINHO MÍNIMO

Optamos por utilizar o algoritmo de Dijkstra, pois nessa situação, como não há pesos de distância negativos, acreditamos que o algoritmo escolhido seja de melhor eficácia e de mais fácil implementação. Usamos a mesma estrutura de dados vista na pergunta número 2 (Busca em Largura), com os mesmos atributos.

5) FLOYD-WARSHALL

Para o algoritmo de Floyd-Warshall, usamos uma lista de listas para armazenar os custos dos caminhos entre os vértices, já que temos os índices de antemão, não houve necessidade de hash tables.