UNIVERSIDADE FEDERAL DE SANTA CATARINA

CENTRO TECNOLÓGICO (CTC)

DEPARTAMENTO DE INFORMÁTICA E ESTATÍSTICA (INE)

Augusto de Hollanda Vieira Guerner	**22102192**
Eduardo Gwoszdz de Lazari	**22100612**
Micael Angelo Sabadin Presotto	**22104063**

INE5413 Grafos

Atividade Prática A2

Florianópolis

2023/2

1 Introdução

O presente documento tem por objetivo descrever o desenrolar da atividade 2 da matéria de Grafos (INE5413), sobretudo as estruturas de dados utilizadas para a implementação dos algoritmos. Antes, no entanto, será comentado brevemente sobre cada questão da atividade.

A primeira é produzir um programa que receba um grafo dirigido e não-ponderado como argumento e retorne no console as componentes conexas dele, uma em cada linha.

O segundo é criar um algoritmo que retorne a ordenação topológica de um grafo dirigido não ponderado. Após executá-lo, mostrar a ordenação na tela usando os rótulos dos vértices ao invés de seus identificadores.

A terceira e última questão é fazer ou o Algoritmo Kruskal ou o Prim para obter árvores mínimas geradores de grafos não-dirigido e ponderado.

Nessa mesma sequência apresentada as atividades, o desenvolvimento vai seguir para descrever cada uma das soluções. Ou seja, primeiro se comenta sobre a solução da questão 1, depois da 2 e, por fim, da 3.

2 Desenvolvimento

Antes de prosseguir para cada questão, vale comentar que o código do Grafo é idêntico ao da primeira atividade com no máximo algumas pequenas correções ou melhorias. Além disso, como feito antes na atividade 1, o ponto de entrada de cada questão está no arquivo com o mesmo número dela. Por exemplo, se a questão é a 1, então o ponto de entrada ou arquivo main é chamado de 1.py. No geral, os pontos de entrada de cada questão são muito simples e parecidos. Lida-se primeiro com o parâmetro de programa, em seguida com a criação do grafo e depois da chamada do algoritmo que soluciona o problema da questão.

2.1 Questão 1

Neste tópico é abordado como foi desenvolvida a questão 1 do trabalho. O primeiro passo foi escolher um algoritmo para solucionar o problema e depois implementá-lo, pensando cuidadosamente em quais estruturas de dados utilizar.

O algoritmo escolhido foi o que está disponível na apostila da disciplina, precisamente no capítulo 7.1, algoritmos 15, 16 e 17. Essencialmente o algoritmo faz uma busca por profundidade no grafo analisado; depois obtém o grafo transposto, isto é, o grafo em que suas arestas tem direção oposta à arestas do grafo original; e, por fim, faz-se novamente uma busca por profundidade, no entanto agora o algoritmo é adaptado para que os vértices a serem analisado primeiros sejam aqueles com maior valor de tempo final obtido na primeira busca por profundidade. Isso é traduzido em código como sendo essencialmente como uma ordenação de uma lista de tuplas, onde o primeiro elemento da tupla é o vértice e o segundo é o valor do tempo final do respectivo vértice.

Entendido o funcionamento em alto nível, é interessante destacar quais estruturas de dados foram utilizadas. Para os retornos das buscas por profundidade, os listas foram imprescindíveis, sendo eles o listas de visitados, o de tempo inicial, o de final e o de

antecessores. Há também o uso indireto de pilha com a chamadas recursivas das funções DFS's e também o uso de dicionários (poderia ser muito bem outra estrutura de dados) na função converter_para_componentes, a qual converte o vetor A (antecessores) em um vetor de componentes (cada linha sendo uma ilha ou componente conexa grafo).

Resumidamente, nesta primeira etapa, utilizou-se o algoritmo da apostila, capítulo 7.1, adaptado para uma implementação real, no caso em python. Já as estruturas de dados utilizadas para o desenvolvimento foram listas, pilhas, obtidas pela recursão da busca por profundidade, e dicionários, usados para converter o vetor de antecessores em um vetor de componentes conexas.

2.2 Questão 2

No tópico presente, comenta-se acerca da questão 2, em qual algoritmo o código foi baseado, como funciona e quais estruturas de dados foram utilizadas.

Igualmente a questão 1, o código foi inteiramente baseado na apostila da disciplina. Para tanto, usou-se como inspiração o capítulo 7.2, precisamente os algoritmos 18 e 19. Feito a questão, a implementação da dois ficou muito simples, uma vez que é literalmente o algoritmo de busca por profundidade com adição de uma lista denominada O. Ela seria a lista que armazena a ordenação topológica do grafo. Insere-se no início dela um vértice todo fim da função DFS_visit_OT, ou seja, após um esse mesmo vértice visitar todos os seus vizinhos, ele mesmo é inserido na lista O. Reforçando, a ordenação topológica é uma ordenação linear em que todo vértice aparece antes de seus descendentes.

Descrito sucintamente o funcionamento do algoritmo de ordenação topológica, parte-se para uma breve análise das estruturas de dados utilizadas nele. Como na questão 1, afinal o algoritmo é essencialmente o mesmo como uma ligeira mudança, as listas são indispensáveis. Tem-se a lista de visitados, de tempo incial, de final e da ordenação topológica (perceba que não é mais de antecessores). Ademais, tem-se a pilha obtida pela recursão das funções.

Assim, de acordo com supracitado, usufruiu-se novamente da apostila da disciplina para que se resolvesse a questão 2 da atividade 2. No mais, referentemente às estruturas de dados para a implementação, as listas, para armazenar as informações necessárias para o bom funcionamento da busca por profundidade, como os vértices visitados, os tempos iniciais e finais e a própria ordenação topológica, e as pilhas, novamente obtidas pela recursão das chamadas de funções, foram as escolhidas.

2.3 Questão 3

Antes de concluir, esta seção falará sobre o desenvolvimento da questão 3. Para isso, primeiro se destaca qual referência foi utilizada para a implementação e como o algoritmo funciona; depois quais estruturas de dados foram usadas.

Não diferentemente das outras, está também tirou sua inspiração da apostila, o algoritmo 21, o de Kruskal, do capítulo 8.2. Ele, o Kruskal, além de não ter nada em comum com os demais algoritmos implementados, é interativo, ou seja, não se usa recursão na sua implementação. O funcionamento dele é simples: primeiro se faz o setup do algoritmo

criando o conjunto de arestas da árvore geradora mínima; a lista que armazena as árvores a que pertence cada vértice; a lista de arestas ordenada por peso; e o próprio peso da árvore geradora. Feito isso, entra-se num for até que n - 1 arestas sejam adicionadas na árvore geradora chamada no algoritmo de A. Obviamente, tem-se uma verificação garantido que o vértice adicionado não forme um circuito na árvore geradora, pois, do contrário, isso geraria uma árvore geradora não mínima. Após adicionar uma nova aresta à árvore, atualiza-se a lista de árvores a que cada vértice pertence, para garantir a corretude de futuras inserções de vértices.

Além disso, as estruturas de dados utilizadas foram lista, conjunto e dicionário. A lista foi utilizada para armazenar conjuntos de árvores, uma para cada vértice; o conjunto foi utilizado para armazenar tanto as árvores geradas enquanto o algoritmo rodava quanto para armazenar a árvore mínima geradora, a saída do programa; por fim, o dicionário foi utilizado para fazer um mapeamento de aresta para seu peso. Como se nota, não teve-se o uso de pilhas com recursão, pois, como dito antes, o algoritmo Kruskal é interativo, aliás o número de iterações do for principal é n - 1, onde n é o número de vértices.

Conclui-se que, conforme exposto, teve-se novamente o uso da apostila como fonte de referência. Além do mais, de modo diferente aos algoritmos anteriores, as estruturas de dados utilizadas foram listas, conjuntos e dicionários.

3 Conclusão

Portanto, a partir da análise minuciosa feita em cada seção, percebe-se que a atividade 2 além de retomar conhecimentos já abordados na atividade 1, a saber, grafo e buscas, também trabalhou conceitos aprendidos recentemente, como componentes conexas, ordenação topológica e árvore geradora mínima, e reforçou o conteúdo de estruturas de dados, por exemplo, listas e pilhas. No que se refere a este último ponto a respeito das estruturas, teve-se o uso de listas, pilhas e dicionários na primeira questão; na segunda, apenas listas e pilhas; e na terceira e última listas, conjuntos e dicionários. Vale destacar que as pilhas são consequências da recursão utilizada nos algoritmos da questão 1 e 2.