UNIVERSIDADE FEDERAL DE SANTA CATARINA
CENTRO TECNOLÓGICO
CIÊNCIA DA COMPUTAÇÃO

Artur Luiz Rizzato Toru Soda

Relatório - Atividade Prática A2 - Grafos

Florianópolis
2024

1 INTRODUÇÃO

Este trabalho tem como objetivo aplicar os conhecimentos aprendidos na disciplina de Grafos até o momento, requisitando a implementação de um grafo e alguns algoritmos para serem aplicados sobre o mesmo para resolver problemas complexos.

1 EXERCÍCIO 1: COMPONENTES FORTEMENTE CONEXAS

Nesta tarefa, foi implementado o algoritmo CFC (Componentes Fortemente Conexas), que recebe como entrada um grafo dirigido não ponderado, retornando então, os componentes fortemente conexos, na forma de árvores representadas como os antecessores dos vértices. Assim, neste algoritmo é utilizado a busca em profundidade, onde primeiramente, faz-se a busca para descobrir os caminhos de todos os vértices para todos os outros. Depois, percorre-se os mesmos caminhos no grafo transposto. Importante notar, que durante a busca é criado 4 vetores importantes, um para controle dos vértices já visitados (C), outro para armazenar os tempos de chegada nos vértices (T), outro para os tempos de volta (F), e por fim o vetor para armazenar os antecessor (A).

Ademais, observando o algoritmo, é possível notar que é utilizado um busca por profundidade adaptado, isso foi implementado no trabalho, de forma que o algoritmo de busca (DFS) altere seus funcionamento caso o parâmetro *F_daptado* receba algum valor.

2 EXERCÍCIO 2: ORDENAÇÃO TOPOLÓGICA

Neste exercício, foi implementado o algoritmo de ordenação topológica, que recebendo como entrada um grafo dirigido, ele retorna seus vértices topologicamente organizados, ou seja, se existe um arco (u, v) pertencente a E, então u aparece antes de v na ordenação.

Assim, no início do algoritmo, é criado 2 vetores principais, um para o controle de vértices já visitados (C), e outro para armazenar os vértices topologicamente organizados (O), então utilizando uma busca em profundidade os vértices vão sendo inseridos no início do vetor O logo que o algoritmo termina de visitá-lo.

3 EXERCÍCIO 2: KRUSKAL OU PRIM

Neste exercício havia a opção de usar o algoritmo de Kruskal ou de Prim, pois os dois calculam árvores geradoras mínimas, entretanto neste trabalho foram implementados os dois.

Em Kruskal, recebendo como entrada o grafo não dirigido ponderado, retorna a árvore geradora na forma de um conjunto das arestas de menores pesos que visitam todos os vértices. Assim em seus inicio é criado 3 vetores principais, o conjunto para armazenar as arestas (A), um vetor em que seus valores são os conjuntos que cada vértice estão participando (S), e um vetor das arestas ordenadas em ordem crescente de peso (E). Então, o algoritmo passa por todas as arestas (u, v) de E, e se o conjunto de que ‘u’ pertence for diferente do conjunto que ‘v’ pertence, adicionamos a aresta (u, v) em A, e unimos os conjuntos de ‘u’ e ‘v’.

Agora em Prim, é retornada também a árvore geradora mínima, entretanto na forma da lista de antecessores. Dessa forma o algoritmo segue os seguintes passos: inicialmente é selecionado um vértice arbitrário (entretanto, no algoritmo implementado, ele sempre escolhe o primeiro, para facilitar). Depois 3 vetores importantes são criados, os de antecessores (A), os que possuem as chaves de cada vértice (que são os custos para cada vértice, K) e um para controlar os vértices que já fazem parte da solução (C). Em seguida, entra em um looping, onde cada iteração, seleciona-se o vértice ‘u’ com a menor chave (utilizando um árvore binária, para melhora da complexidade), assim indica, em C, que o vértice selecionado faz parte da solução. Por fim, passando por todos os vizinhos ‘v’ de ‘u’, caso ‘v’ não fazer parte da solução e o peso da aresta (u, v) for menor que a chave de ‘v’, então definimos ‘u’ como antecessor de ‘v’ e definimos que a chave de ‘v’ agora é o peso de (u, v).

4 CONCLUSÃO

Considerando todos os exercícios realizado, é possível concluir que o projeto proposto foi uma ótima prática para a aplicação de grafos e algoritmos aplicados nele, nos permitindo uma compreensão mais profunda das estruturas de dados e suas possibilidades, mostrando algoritmos de busca por componentes fortemente conexos (CFC), ordenação topológica e a busca por árvores geradoras mínimas, com os algoritmos de Kruskal e Prim.