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 A3

Florianópolis

2023/2

1 Introdução

O presente documento tem por objetivo descrever o desenrolar da atividade 3 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 ponderado como argumento e retorne no console o fluxo máximo, utilizando o algoritmo de Edmonds-Karp.

O segundo é criar um programa que recebe um arquivo de grafo bipartido, não-dirigido e não-ponderado e no final informa qual o valor do emparelhamento máximo e quais arestas pertencem a ele, utilizando o algoritmo de Hopcroft-Karp.

A terceira e última questão é criar um programa que recebe um grafo não-dirigido e não-ponderado como argumento e ao final é informado a coloração mínima e qual número cromático foi utilizado em cada vértice.

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 e da segunda atividade com no máximo algumas pequenas correções, melhorias e foi adicionado alguns métodos a mais a fim de facilitar o desenvolvimento de alguns algoritmos. Além disso, como feito antes na atividade 2, 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, que utilizou-se do algoritmo de Edmonds-Karp disponível na apostila da disciplina. Essencialmente o algoritmo de Edmonds-Karp é uma abordagem eficiente para encontrar o fluxo máximo em um grafo direcionado e ponderado. Desenvolvido com base no algoritmo Ford-Fulkerson, o método utiliza uma busca em largura (BFS) para descobrir caminhos aumentadores no grafo residual, caminhos estes que permitem aumentar o fluxo de maneira iterativa. A capacidade residual das arestas é fundamental para o funcionamento do algoritmo, representando a quantidade de fluxo que ainda pode ser alocada em uma determinada aresta.

Entendido o funcionamento em alto nível, é interessante destacar quais estruturas de dados foram utilizadas. Primeiramente o grafo utilizado advém de uma matriz de adjacência assim como o grafo residual eles utilizam a mesma classe que está descrita no grafo.py. Além

disso, é utilizada uma fila para que a BFS seja feita, também uma lista de pais para controlar os antecessores de cada vértice no grafo, assim como uma lista de visitados para controlar vértices que já foram visitados.

Em resumo, a abordagem da questão 1, que empregou o algoritmo de Edmonds-Karp, destacou-se pela sua eficiência na busca do fluxo máximo em grafos direcionados e ponderados. Originado do algoritmo Ford-Fulkerson, sua simplicidade conceitual o torna aplicável em diversas situações práticas, como em redes de transporte e comunicação. A busca em largura (BFS) desempenhou um papel fundamental ao identificar caminhos aumentadores no grafo residual, possibilitando incrementos iterativos no fluxo. A consideração da capacidade residual das arestas e a escolha cuidadosa de estruturas de dados, como a matriz de adjacência, filas e listas, demonstraram a importância de uma implementação eficaz. Em conjunto, o algoritmo de Edmonds-Karp apresenta-se como uma ferramenta valiosa para a otimização de fluxos em grafos, fornecendo uma base sólida para análises de redes e aprimoramento de recursos em diversos contextos.

2.2 Questão 2

No tópico presente, comenta-se acerca da questão 2, utilizando-se do algoritmo de Hopcroft-Karp disponível na apostila da disciplina.
O algoritmo de Hopcroft-Karp é um algoritmo eficiente para encontrar o emparelhamento máximo bipartido em um grafo não direcionado. Ele é particularmente útil em situações em que os vértices do grafo podem ser divididos em dois conjuntos disjuntos, e o objetivo é encontrar o maior conjunto possível de arestas não sobrepostas que conectam os dois conjuntos.

É importante destacar nesse momento as estruturas de dados utilizadas: dicionários, para facilitar a indexação de None no algoritmo BFS e DFS; fila, utilizada no algoritmo BFS; lista, para a construção da saída do console; e, por último, pilha das chamadas recursivas das funções.

Neste tópico, discutimos a implementação da questão 2, utilizando o algoritmo de Hopcroft-Karp para encontrar emparelhamentos máximos em grafos bipartidos não direcionados. Destacamos a eficiência deste algoritmo em situações onde os vértices do grafo podem ser divididos em conjuntos distintos. Além disso, enfatizamos a importância das estruturas de dados empregadas, como dicionários para indexação, filas para o algoritmo BFS, listas para construção da saída no console e pilhas nas chamadas recursivas de funções. Essas escolhas cuidadosas de estruturas de dados não apenas facilitam a implementação, mas também contribuem para a eficácia e eficiência do algoritmo, proporcionando uma compreensão mais abrangente de sua aplicação em cenários práticos.

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.

Para auxiliar no desenvolvimento da questão utilizou-se do algoritmo disponível na apostila, porém não era apenas isso para essa questão.

O algoritmo de Lawler para coloração de vértices em grafos é baseado em uma abordagem de busca local. Inicialmente, cada vértice recebe uma cor, e o algoritmo itera sobre os vértices, ajustando as cores dos vértices e de seus vizinhos para resolver conflitos de coloração. Esses ajustes podem envolver a troca de cores entre vértices adjacentes ou a escolha de novas cores. O processo continua até que não haja mais violações das regras de coloração, onde a condição de parada pode ser definida por um número máximo de iterações ou pela ausência de conflitos. A ordem em que os vértices são processados pode influenciar o desempenho do algoritmo, e a implementação pode incorporar heurísticas adicionais para otimização. O resultado final é uma coloração dos vértices sem conflitos, respeitando as regras de coloração estabelecidas.

Para a coloração foi utilizado um algoritmo de backtracking, ou seja, foram definidas possibilidades de cores de 1 até coloração mínima para cada vértice. Após isso, o algoritmo tentava colorir o vértice com uma das possibilidades, verificando seus vizinhos. Se nenhuma das cores puder ser aplicada, é necessário voltar ao vértice anterior e mudar sua cor, para assim, poder prosseguir a tentativa pela solução.

As estruturas de dados utilizadas foram tuplas para obter os conjuntos independentes maximais, listas para listar as possibilidades das cores, grafos nas iterações pelo conjunto potência e matrizes para construir a matriz de adjacência.

Em conclusão, a abordagem adotada para a questão 3 baseou-se no algoritmo de Lawler para coloração de vértices em grafos, destacando-se pela sua fundação em busca local. A implementação envolveu o uso de um algoritmo de backtracking para a coloração, explorando possibilidades de cores para cada vértice e ajustando conforme as regras estabelecidas. A ordem de processamento dos vértices e a incorporação de heurísticas adicionais foram considerações importantes para otimizar o desempenho do algoritmo. As estruturas de dados, como tuplas para conjuntos independentes maximais, listas para possibilidades de cores, grafos nas iterações pelo conjunto potência e matrizes para a construção da matriz de adjacência, desempenharam papéis cruciais na execução eficiente do algoritmo. O resultado final foi a obtenção de uma coloração de vértices sem conflitos, respeitando as regras de coloração, demonstrando a eficácia da abordagem escolhida para a resolução da questão 3.

3 Conclusão

Neste documento, abordou-se o desenvolvimento da atividade 3 da disciplina de Grafos (INE5413), explorando as implementações dos algoritmos de Edmonds-Karp, Hopcroft-Karp e Lawler para resolver problemas específicos. O algoritmo de Edmonds-Karp destacou-se na busca eficiente pelo fluxo máximo em grafos dirigidos e ponderados, utilizando uma abordagem baseada em busca em largura (BFS) e considerando a capacidade residual das arestas. Em seguida, o algoritmo de Hopcroft-Karp foi apresentado como uma solução eficaz para encontrar emparelhamentos máximos em grafos bipartidos não direcionados, enfatizando a importância das estruturas de dados como dicionários, filas e listas. Por fim, a implementação da questão 3 envolvendo o algoritmo de Lawler para coloração de vértices em

grafos foi discutida, destacando sua abordagem de busca local e a necessidade de ajustes nas cores dos vértices e seus vizinhos para resolver conflitos. Este algoritmo, baseado em heurísticas e estratégias de otimização, oferece uma solução para a coloração de vértices respeitando as regras estabelecidas. Em resumo, as estruturas de dados utilizadas em cada questão foram cuidadosamente escolhidas para garantir eficiência e eficácia nas implementações, evidenciando a aplicabilidade desses algoritmos em diversas situações práticas.