Trabalho Prático 2

Desemprego

Matheus Guimarães Couto de Melo Afonso
Matrícula: 2021039450

Departamento de Ciência da Computação – Universidade Federal de Minas Gerais (UFMG)
Belo Horizonte – MG – Brasil

matheusgcma@ufmg.br

1. Introdução

O trabalho prático 2 consiste em um problema no qual, dado um conjunto de pessoas P (com u pessoas) e um conjunto de cargos C (com j cargos), além de uma lista com e relações pessoa-cargo, deve ser encontrado o número máximo de pares únicos entre pessoas e cargos. O problema foi resolvido de duas maneiras diferentes:

1. Por meio de um algoritmo guloso, que não encontra a solução ótima;
2. Por meio de um algoritmo não-guloso, que é capaz de encontrar a solução ótima.

2. Modelagem

O problema foi modelado utilizando um grafo não-ponderado e não-direcionado, de forma que uma aresta (a, b) com $a \in U$ e $b \in C$ pertence ao grafo caso o usuário a seja qualificado para exercer o cargo b.

O grafo gerado pela entrada é um grafo bipartido, uma vez que não existe nenhuma aresta conectando duas pessoas ou dois cargos, apenas uma pessoa a um cargo. Dessa forma, o problema consiste em encontrar o pareamento máximo em um grafo bipartido.

Visto que a entrada do problema é dada por meio de strings que representam as pessoas e os cargos, foi necessário realizar um mapeamento de cada uma das pessoas e cargos para um determinado número inteiro, que varia entre 0 e $u - 1$ para as pessoas e entre 0 e $j - 1$ para os cargos.

2.1. Algoritmo Guloso

O algoritmo guloso utilizado consiste em iterar pela lista de adjacências de cada pessoa e atribuir a ela o primeiro cargo da lista que ainda não tenha sido atribuído a

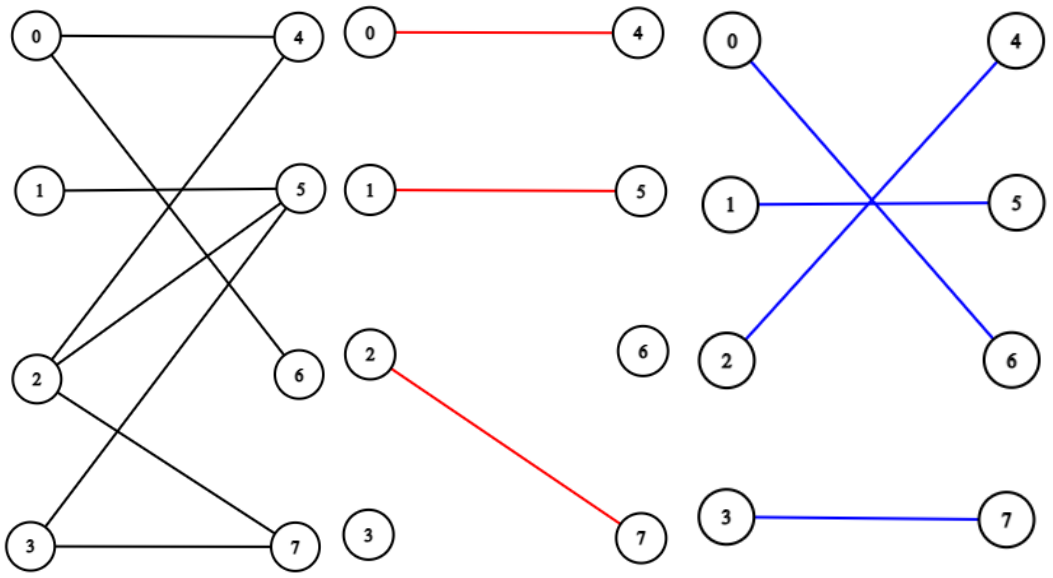
nenhum outro usuário. Caso não seja possível atribuir nenhum dos cargos para os quais o usuários tenha qualificação, ele é ignorado e o novo usuário é explorado.

O algoritmo guloso constrói um `map<int, int>` que mapeia o index de uma pessoa para o index de um cargo. Contudo, uma vez que é pedido apenas o número de casamentos, a solução retorna o tamanho desse map.

2.2. Solução Exata

A solução exata explora o conceito de caminho aumentante (*augmenting path*). Um caminho aumentante é um caminho entre uma pessoa sem cargo e um cargo sem ocupante, de forma que seja possível navegar pelos casamentos previamente feitos. A utilização desse conceito é importante pois permite desfazer uma decisão anterior em prol de achar uma solução mais adequada para o problema.

As imagens abaixo representam as diferenças entre soluções encontradas pelo algoritmo guloso e pelo algoritmo exato. A primeira imagem representa o grafo de entrada, a segunda representa a solução encontrada pelo algoritmo guloso e a terceira representa a solução encontrada pelo algoritmo exato.



Por meio da exploração dos caminhos aumentantes, o algoritmo ótimo é capaz de encontrar o caminho 3 - 7 - 2 - 4 - 0 - 6 e reverter os pares (0-4) e (2-7) previamente encontrados para aumentar o número total de casamentos, pareando os nós (0-6), (2-4) e (3-7).

A implementação da solução exata utiliza uma *depth-first search* (DFS) para verificar se ainda existe um caminho aumentante no grafo.

3. Análise de Complexidade

O grafo é armazenado em uma lista de adjacências, em oposição a uma matriz. A complexidade de espaço de uma lista de adjacências é $O(|V| \log |E|)$. Além disso, são utilizados dois `map<string, int>` para manter o mapeamento dos índices das pessoas e cargos, o que utiliza um espaço adicional $O(|V|)$. A complexidade de espaço total portanto é dominada por $O(|V| \log |E|)$.

3.1. Algoritmo Guloso

O algoritmo guloso utilizado tem complexidade de tempo dada por $O(|V| + |E|)$. Ele itera pelos vértices relativos às pessoas $O(|V|)$ e para cada uma das pessoas itera pela lista de adjacências correspondente. Ao final, ele terá passado por cada uma das arestas do grafo uma única vez $O(|E|)$.

3.2. Solução Exata

A complexidade de tempo da solução exata é $O(|V| * |E|)$. Isso ocorre pois, para cada vértice do grafo, a busca em profundidade percorre todas as arestas no pior caso.