Relatório AP3 EDA

Equipe:

Francisco Valdemi Leal Costa Junior- 485325
Roberto de Oliveira Coutinho - 499484

Problema 1: Colorindo um grafo com duas cores.

Introdução:

A questão pede para resolvermos o problema de pintar um dado grafo que foi fornecido por um arquivo TXT com duas cores, de forma que nenhum do vertices tenha a mesma cor dos seus vizinhos, após fazer isso é preciso retorna se foi possível ou não, caso tenha sido possível é preciso ainda retornar a cor de cada vértice.

Solução:

Para resolver o problema foi usado um algoritmo de busca em largura(BFS) para percorrer o grafo, então o primeiro vértice é pintado de vermelho e os seu vizinhos preto e assim em diante, mas se for verificado em algum vértice que seu vizinho tem a mesma cor então o algoritmo para e retorna falso, ou seja o grafo não pode ser pintado de duas cores(não é bipartido), caso o grafo seja bipartido o algoritmo será executado até o final, pintado todos os vértices e no fim printa a cor de todos os vértices.

Complexidade:

Como o grafo foi implementado usando uma matriz de adjacência n por n, em termo de memoria a solução usa $O(n^2)$ sendo n número de vértices.

Em tempo de execução a solução é $O(n^2)$.

Bibliografia:

https://pt.wikipedia.org/wiki/Matriz_de_adjac%C3%AAncia
https://pt.wikipedia.org/wiki/Busca_em_largura
https://www.geeksforgeeks.org/breadth-first-search-or-bfs-for-a-graph/

Problema 2: Despedida no Vila Hall.

Introdução:

A questão pede que o nosso código leia entradas onde a primeira entrada é sempre o nome do dono da empresa e as linhas seguintes são os nomes dos outros funcionários com seus respectivos chefes, onde todos os funcionários só possuem um chefe diretamente. Para a despedida no Vila Hall, caso um funcionário seja chamado o seu chefe direto não pode ser chamado e vice e versa, ou seja, um funcionário não pode ir a festa se seu chefe for, mas nada impede que o chefe do chefe vá. O nosso algoritmo deve retornar o número máximo de convidados e SIM caso a nossa lista de convidados seja única e NAO caso a lista não seja única(Ter o tamanho máximo, mas com pessoas diferentes).

Solução:

A nossa solução começa marcando todos os funcionários que não possuem subordinados, uma vez que para o maior número de pessoas na festa eles tem que ir já que a ida deles não tira outra pessoa da festa, com exceção dos próprios chefes dele, depois de pegarmos as pessoas que não possuem subordinados, percorremos todos os funcionários da empresa verificando se eles já estão marcados que não vão à festa porque algum subordinado dele já vai, caso eu chegue em um caso em que o subordinado dele ainda não foi marcado sobre não ir a festa e nem esse funcionário foi marcado que vai a festa, eu marco que esse funcionário vai a festa e marco que seu subordinado não irá à festa e assim eu sigo até ter marcado todos os vértices do meu grafo. Infelizmente não conseguimos fazer com que a função nos falasse se nossa lista é única ou não.

Obs: Professor esse código que você está recebendo possui duas funções no main.cpp fora a main, nossa melhor_caso e duas funções comentadas (DFS e DFSaux) que fizemos depois de fazer a melhor caso, tentamos fazer de outra maneira onde iriamos percorrer todos vértices do grafo comparando qual caso teria maior numero de convidados, tentamos dessa forma pq achávamos que a nossa melhor caso em algum caso iria da errado e para melhorar a complexidade, mas chegamos em um caso em que as duas funções deram uma diferença, debugando a função melhor_caso na mão achamos que ela estava certa na verdade e como não conseguimos debugar a DFS uma vez que ela é recursiva ficamos na dúvida já que nos casos do PDF as duas funções deram o resultado certo.
De toda forma, considere a nossa função melhor_caso professor, estou enviando essa DFS mais com o intuito de o senhor tirar a minha dúvida do por que ela está errada.

Complexidade:

Como o grafo foi implementado usando uma matriz de adjacência n por n, em termo de memoria a solução usa $O(n^2)$ sendo n número de vértices.

Em tempo de execução a solução é O(n + n * lv).
sendo n o numero de vertices e lv a lista de vizinhos.

Bibliografia:

https://pt.wikipedia.org/wiki/Matriz_de_adjac%C3%AAncia
https://www.geeksforgeeks.org/breadth-first-search-or-bfs-for-a-graph/