EDA - Projeto Final

Ezequiel dos Santos Melo, Gustavo Fernandes de Barros

4 de julho de 2023

1 [Problema 1] Colorindo os vértices do grafo com duas cores

Meus primeiros pensamentos sobre a resolução foram de certa forma intuitivos, visto que, tratando-se somente de duas cores, talvez fosse possível obter a resposta testando, ou seja, colorindo cada vértice do grafo para verificar se o mesmo pode ser colorido com duas cores ou não.

Após pesquisar sobre o problema da χ-coloração de vértices, descobri uma relação direta entre a veracidade desta propriedade para duas cores e grafos bipartidos, onde há um teorema no qual um grafo pode ser colorido com duas cores **se e somente se** ele for bipartido. Desta forma, é possível aproveitar-se deste teorema para resolver a questão.

Uma forma algorítmica de resolver essa questão e descobrir se um grafo é bipartido é de fato verificando se é possível colorir o mesmo com duas cores de forma que vértices adjacentes tenham cores distintas, e para isso usei uma abordagem semelhante a busca em largura, percorrendo todos os vértices do grafo, colorindo-os e verificando a cor de seus vizinhos. Caso haja em algum momento vizinhos de cores semelhantes, logo conclui-se que o grafo não é bipartido, portanto não é possível que o mesmo seja colorido com duas cores.

```
bool valid_coloring(Graph& graph) {
    vector<char> colors(graph.get_number_of_vertices(), ' ');
    colors[0] = 'R';

    queue<int> queue;
    queue.push(0);

    while(!queue.empty()) {
        int current = queue.front();
        queue.pop();

        for(auto edge : graph.neighbors(current)) {
            if(colors[edge.get_destiny()] == ' ') {
                if(colors[current] == 'R') colors[edge.get_destiny()] = 'B';

                else colors[edge.get_destiny()] = 'R';

                queue.push(edge.get_destiny());
            }

            else if(colors[edge.get_destiny()] == colors[current]) {
                return false;
            }
        }
    }

    return true;
}
```

Provavelmente esse algoritmo não tem uma complexidade de tempo e espaço muito eficientes, visto que no pior caso todos os vértices são visitados, além de que a cada vértice temos de visitar todos os seus vizinhos também, o que pode resultar em uma complexidade de tempo $O(V^2)$.

Em relação ao espaço, usa-se um vetor para mapear a cada vértice uma cor e uma fila para armazenar os vértices que têm de ser visitados, onde ambos têm tamanho V, fazendo o algoritmo possuir complexidade de espaço O(V).

2 [Problema 2] Seis graus de Kevin Bacon

3 [Problema 3] Vias de mão dupla

Referências