Universidade Federal do Ceará - Campus Quixadá
Estrutura de Dados Avançada
Avaliação Parcial 03

Nome: Lucas Ferreira Lima Matrícula:418399
Nome: Felipe de Sousa Santos Matrícula:414235

[Problema 1] Colorindo um grafo com duas cores.

Para a resolução do problema foi utilizado uma classe *Graph* que possui como atributos privados um inteiro ***n*** e uma matriz ***g***, que guardam o número de vértices e a matriz de adjacência respectivamente.

Como público as funções de construtor e destrutor, ***addEdge()*** que recebe dois inteiros (x e y) que representam as ligações dos vértices e são alocados como 1 na matriz. Em seguida a função booleana ***isBipartite()***, que não recebe nenhum valor como parâmetro e retorna um valor verdadeiro ou falso. Por meio de uma lista `queue <int> number` ele percorre todo o grafo, o valor do vértice é inserido na fila então ele procura, se o encontrar retorna uma cor do vértice que é salva no vetor `int color[n]` , logo em seguida ele remove o vértice da lista e pega o vértice seguinte, assim colorindo todos os vértices.

A função ***isBipartite()*** utiliza uma abordagem que verifica se o grafo é bipartido ou não usando algoritmo de retrocesso *m* problema de coloração, utilizando a busca em largura (BFS), da seguinte forma:

1. Atribuir a cor 1 ao vértice de origem.
2. Pinte todos os vizinhos com a cor 2.
3. Pinte todos os vizinhos do vizinho com a cor 1.
4. Atribuir cor a todos os vértices de forma que satisfaça todas as restrições do problema de coloração de maneira *m* onde $m = 2$.
5. Ao atribuir cores, se encontrarmos um vizinho que é colorido com a mesma cor do vértice atual, então o grafo não pode ser colorido com 2 vértices, ou seja, o grafo não é bipartido.

A implementação foi feita com matriz de adjacência, portanto, possui um armazenamento de $O(|V|^2)$. As operações de inserção e remoção tem tempo $O(1)$. Já a operação de percorrer a vizinhança tem tempo $O(|V|)$.

A função ***isBipartite()*** possui uma complexidade linear de $O(n+m)$ que representa o valor máximo entre as variáveis m e n. Sua implementação se assemelha a busca em largura, pois parte de um vértice origem e vai em busca dos mais antigos primeiro, as arestas são exploradas em ordem crescente dos vértices adjacentes.

[Problema 2] Despedida no Vila Hall.

Para a resolução do problema foi utilizado uma classe Grafo que possui como atributos as seguintes funções, ***Grafo(int V)***, o construtor, ***adicionarAresta(int v1, int v2)***, recebe dois inteiros e liga as arestas, ***busca(int x)*** que recebe um inteiro e realiza a busca dos vértices e por último a auxiliar ***buscaAux()***, que chama a função principal várias vezes.

A implementação foi feita com lista de adjacência, portanto, possui um armazenamento de $O(|V|+|E|)$. As operações de inserção e remoção tem tempo $O(1)$ e $O(d(v))$, respectivamente. Já a operação de percorrer a vizinhança tem tempo $O(d(v))$.

Bibliografia

Apêndice B.4 e Capítulo 22: Thomas H. CORMEN et al.Introduction to algorithms.

https://pt.wikipedia.org/wiki/Grafo_bipartido

https://pt.wikipedia.org/wiki/Colora%C3%A7%C3%A3o_de_grafos