UNIVERSIDADE PRESBITERIANA MACKENZIE
Faculdade de Computação e Informática
Prof. Dr. Ivan Carlos Alcântara de Oliveira
Teoria dos Grafos
Atividade de Implementação (grupo do Projeto) – Coloração de Vértices

Nome do Integrante (ordem alfabética)	RA
Gabriel Gonzaga Chung	**10403025**
Igor Benites Moura	**10403462**
Rodrigo Machado de Assis Oliveira de Lima	**10401873**

Conteúdo

Tendo por base a implementação em Java, C++, C ou Python para matriz de adjacência ou Lista de Adjacência e o estudo realizado sobre coloração e os algoritmos de coloração por classe e sequencial, realizar a implementação de um dos dois algoritmos estudados na classe TGrafo. Utilizar como subsídio os algoritmos presentes no material da aula.

Os resultados de pelo menos dois dos seus testes devem ser colocados no relatório a seguir. Um dos testes deve ser com a atividade entregue no tema “Coloração de Vértices – Exercício”.

Além disso, ao final do relatório, colocar um Apêndice com o código fonte completo contendo todos os arquivos utilizados na atividade.

Não se esquecer de anexar todos os arquivos fonte compactados ao enviar pela área do Moodle.

Relatório

Obs: usamos o início da contagem dos vértices ‘0’ ao invés do ‘1’.

1) Teste do grafo do material disponibilizado em aula:

```
## Igor Benites Moura - 10403462
## Rodrigo Machado de Assis Oliveira de Lima - 10401873
## Gabriel Gonzaga Chung - 10403025

from TGrafoColoracao import Grafo

g = Grafo(5)

g.insereA(0, 1)
g.insereA(0, 4)
g.insereA(1, 2)
g.insereA(1, 3)
g.insereA(2, 3)
g.insereA(3, 4)

coloracao = g.coloracao_sequencial()
print(coloracao)
```

Resultado:

```
[{0, 2}, {1, 4}, {3}]
```

Ou seja, é possível colorir os vértices 0 e 2 de uma cor, 1 e 4 de outra e o 3 de outra

2) Teste do grafo da atividade "Coloração de Vértices – Exercício" :

```
## Igor Benites Moura - 10403462
## Rodrigo Machado de Assis Oliveira de Lima - 10401873
## Gabriel Gonzaga Chung - 10403025

from TGrafoColoracao import Grafo

g = Grafo(9)

g.insereA(0, 5)
g.insereA(0, 6)
g.insereA(0, 8)
g.insereA(1, 2)
g.insereA(1, 7)
g.insereA(2, 4)
g.insereA(2, 7)
g.insereA(3, 4)
g.insereA(3, 5)
g.insereA(3, 7)
g.insereA(4, 6)
g.insereA(5, 6)
g.insereA(5, 8)
g.insereA(6, 7)
g.insereA(6, 8)

coloracao = g.coloracao_sequencial()
print(coloracao)
```

Resultado:

```
[{0, 1, 3}, {2, 5}, {8, 4, 7}, {6}]
```

Ou seja, é possível colorir os vértices 0, 1 e 3 de uma cor, 2 e 5 de outra, 8, 4 e 7 de outra e o 6 de outra

Apêndice

Código fonte TGrafoColoracao.py:

```
## Igor Benites Moura - 10403462
## Rodrigo Machado de Assis Oliveira de Lima - 10401873
## Gabriel Gonzaga Chung - 10403025

class Grafo:
    TAM_MAX_DEFAULT = 100 # qtde de vértices máxima default
    # construtor da classe grafo
    def __init__(self, n=TAM_MAX_DEFAULT):
        self.n = n # número de vértices
        self.m = 0 # número de arestas
        # matriz de adjacência
        self.adj = [[0 for i in range(n)] for j in range(n)]

    # Insere uma aresta no Grafo tal que
    # v é adjacente a w
    def insereA(self, v, w):
        if self.adj[v][w] == 0:
            self.adj[v][w] = 1
            self.adj[w][v] = 1
            self.m+=1

    # remove uma aresta v->w do Grafo
    def removeA(self, v, w):
        if self.adj[v][w] == 1:
            self.adj[v][w] = 0
            self.adj[w][v] = 0
            self.m-=1 # atualiza qtd arestas


    # Algoritmo de Coloração Sequencial
    def coloracao_sequencial(self):
        C = [set() for _ in range(self.n)]
        k = 0
        for i in range(self.n):
            while True:
                if all(self.adj[i][j] == 0 or j not in C[k] for j in
range(self.n)):
```

```
                    C[k].add(i)
                    break
                else:
                    k += 1
            k = 0

        # Filtra os conjuntos vazios
        C = [subset for subset in C if subset]
        return C




    # Apresenta o Grafo contendo
    # número de vértices, arestas
    # e a matriz de adjacência obtida
    def show(self):
        print(f"\n n: {self.n:2d} ", end="")
        print(f"m: {self.m:2d}\n")
        for i in range(self.n):
            for w in range(self.n):
                if self.adj[i][w] != 0:
                    print(f"Adj[{i:2d},{w:2d}] = {self.adj[i][w]:2d} ",
end="")
                else:
                    print(f"Adj[{i:2d},{w:2d}] = ∞ ", end="")
            print("\n")
        print("\nfim da impressao do grafo." )

    # Apresenta o Grafo contendo
    # número de vértices, arestas
    # e a matriz de adjacência obtida
    # Apresentando apenas os valores 0 ou 1
    def showMin(self):
        print(f"\n n: {self.n:2d} ", end="")
        print(f"m: {self.m:2d}\n")
        for i in range(self.n):
            for w in range(self.n):
                if self.adj[i][w] != 0:
                    print(f" {self.adj[i][w]:2d} ", end="")
                else:
                    print(" ∞  ", end="")
```

```
            print("\n")
        print("\nfim da impressao do grafo." )
```

Código fonte testeColoracao.py:

```
## Igor Benites Moura - 10403462
## Rodrigo Machado de Assis Oliveira de Lima - 10401873
## Gabriel Gonzaga Chung - 10403025

from TGrafoColoracao import Grafo

g = Grafo(9)

g.insereA(0, 5)
g.insereA(0, 6)
g.insereA(0, 8)
g.insereA(1, 2)
g.insereA(1, 7)
g.insereA(2, 4)
g.insereA(2, 7)
g.insereA(3, 4)
g.insereA(3, 5)
g.insereA(3, 7)
g.insereA(4, 6)
g.insereA(5, 6)
g.insereA(5, 8)
g.insereA(6, 7)
g.insereA(6, 8)

coloracao = g.coloracao_sequencial()
print(coloracao)
```