

Felipe Augusto Ferreira de Castro **Matrícula:** 11711BCC033

Sarah Hanna VB Silva **Matrícula:** 11621BCC021

Renata Cristina Gomes da Silva **Matrícula:** 11721BCC012

Trabalho de Analise de Algoritmos

Universidade Federal de Uberlândia

2021

Sumário

Sumário	1
1	1
1.1	2
1.2	3
1.3	3
1.3.1	3
1.3.2	6
1.4	9

1 Coloração de Grafos

1.1 Pseudo-código

O problema de coloração de grafos é um problema bastante discutido na literatura da área e possui vários algoritmos para solucioná-lo. Portanto, nesta seção apresentaremos o algoritmo usado em nosso trabalho para encontrar a disposição de cores e posteriormente discutiremos a eficiência do algoritmo. A seguir o algoritmo utilizado apresentado em pseudo-código.

Algorithm 1: Coloração de Grafos

Input: grafo

Output: Lista com as cores de cada vertice

declare uma variavel flag;

faça flag = verdadeiro;

atribua a cor 0 para todos os vertices do grafo;

inicie pelo primeiro vértice do grafo;

while *não atribuido uma cor a todos os vertices* **do**

while *para todos os vertices adjacentes* **do**

 olhe a cor do vertice adjacente;

if *possui a mesma cor que o vertice adjacente* **then**

 faça flag = falso;

 pare o laço;

else

end

end

if *flag = verdadeiro* **then**

 | proximo vertice;

else

 faça flag = verdadeiro;

 some 1 a cor deste vertice;

end

end

Explicando de maneira mais informal o algoritmo se resume em alguns passos:

- Para cada vértice v do grafo G vamos olhar as cores de seus vertices adjacentes;
- vamos avançando na lista de cores até encontrar uma cor, a qual não foi atribuída a nenhum vértice adjacente a v ;

- atribuímos a cor encontrada ao vértice v ;

1.2 Estrutura de dados

Visto o algoritmo para coloração apresentado na seção anterior foi decidido estruturar o grafo de maneira a facilitar encontrar os vértices adjacentes de cada vértice v do grafo G . Desta forma, cada vértice é uma estrutura que possui duas informações:

- valor da cor atribuída ao vértice;
- uma lista de identificadores dos vértices adjacentes;

A identificação do vértice adjacente v_d é feita com um indexador da posição de v_d na lista de vértices do grafo G .

A Estrutura do grafo é constituída de duas informações:

- Lista de vértices presentes no grafo(o indexador do vértice nessa lista é o identificador do vértice);
- Quantidade de arestas presentes no grafo;

Utilizando-se desta estrutura, encontrar os vértices adjacentes de um vértice v se resume a apenas percorrer uma lista, excluía a necessidade de verificar se o vértice v_d é adjacente a v .

1.3 Desenvolvimento do Trabalho

O programa foi desenvolvido na linguagem Python devido a facilidade de encontrar ferramentas prontas para manipular estruturas de dados e o conhecimento prévio que os autores deste trabalho tinham sobre a linguagem. Além disso, o ambiente de desenvolvimento usado foi o Visual Studio Code.

1.3.1 Grafos

Quanto aos grafos, foram utilizados 5 grafos, no formato DIMACS, os quais serão apresentados a seguir.

1. Grafo 1

vertices: 10 , arestas: 15,

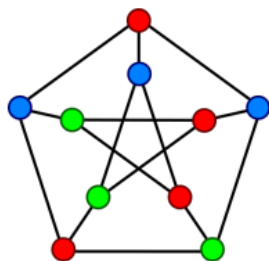
a 0 1,

a 0 2,

a 0 3,

a 1 4,

a 1 8,
a 2 6,
a 2 7,
a 3 5,
a 3 9,
a 4 5,
a 4 7,
a 5 6,
a 6 8,
a 7 9,
a 8 9;

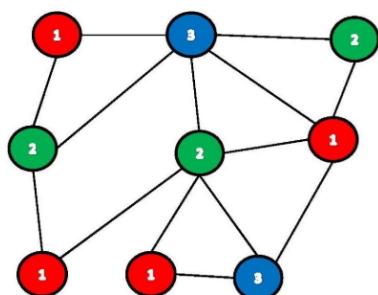


Fonte: O enunciado deste trabalho

2. Grafo 2

vertices: 9, arestas: 14,

a 0 3,
a 0 1,
a 1 2,
a 1 3,
a 2 4,
a 3 4,
a 3 6,
a 3 7,
a 4 5,
a 4 7,
a 4 8,
a 5 8,
a 6 7,
a 7 8;



Fonte: <https://coloringbee.blogspot.com/2018/09/coloring-graph-example.html>

3. Grafo 3

vertices: 8, arestas: 12,

a 0 5,

a 0 6,

a 0 7,

a 1 4,

a 1 6,

a 1 7,

a 2 4,

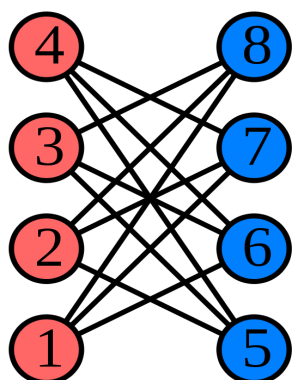
a 2 5,

a 2 7,

a 3 4,

a 3 5,

a 3 6;



Fonte: https://handwiki.org/wiki/Grundy_number

4. Grafo 4

vertices: 11, arestas: 13 a 0 1,

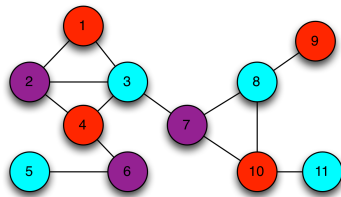
a 0 2,

a 1 2,

a 1 3,

a 2 3,

a 2 6,
a 3 5,
a 4 5,
a 6 7,
a 6 9,
a 7 8,
a 7 9,
a 9 10;



Fonte: <https://blog.cryptographyengineering.com/2014/11/27/zero-knowledge-proof/>

5. Grafo 5

vértices: 7393, arestas: 25569.

Devido a quantidade de vértices e arestas não é viável apresentar aqui a estrutura do grafo. Visto isso, vamos disponibiliza-lo no seguinte link: <https://github.com/felipe-2705/Trabalho-AA/blob/master/grafos/graf5.dimacs>

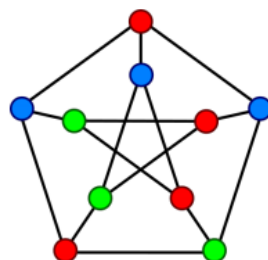
Fonte: <https://lcs.ios.ac.cn/~caisw/graphs.html>

1.3.2 Execução

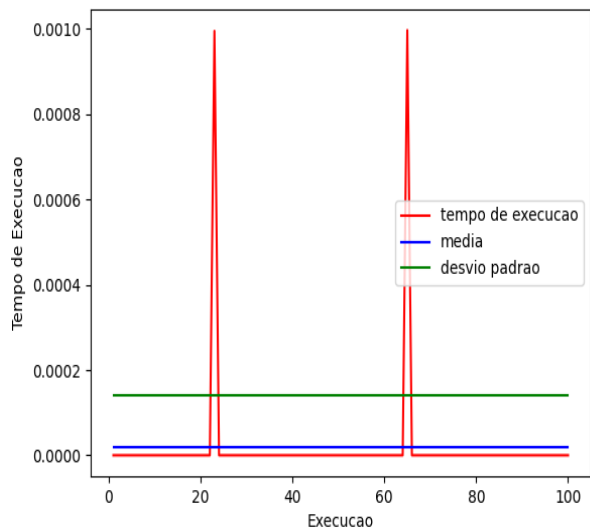
Aqui mostraremos o resulta das colorações para os grafos de 1 a 4 e compararemos com os resultados já conhecidos para cada um dos casos. Além de mostrar o tempo de execução para 100 execuções para cada grafo e suas medias. As bibliotecas utilizada para tal foram *statistics* e *Matplotlib*

1. Grafo 1

```
vertice[ 0 ] : vermelho
vertice[ 1 ] : azul
vertice[ 2 ] : azul
vertice[ 3 ] : azul
vertice[ 4 ] : vermelho
vertice[ 5 ] : verde
vertice[ 6 ] : vermelho
vertice[ 7 ] : verde
vertice[ 8 ] : verde
vertice[ 9 ] : vermelho
```



Podemos observar que para este grafo o algoritmo atingiu uma solução ótima, utilizando 3 cores para colorir o grafo 1; Os tempos de execução são mostrados a seguir:

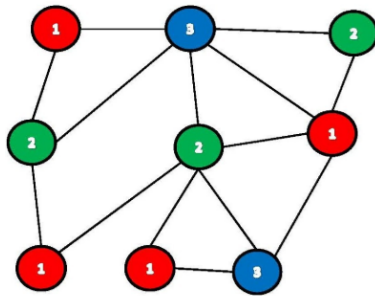


2. Grafo 2

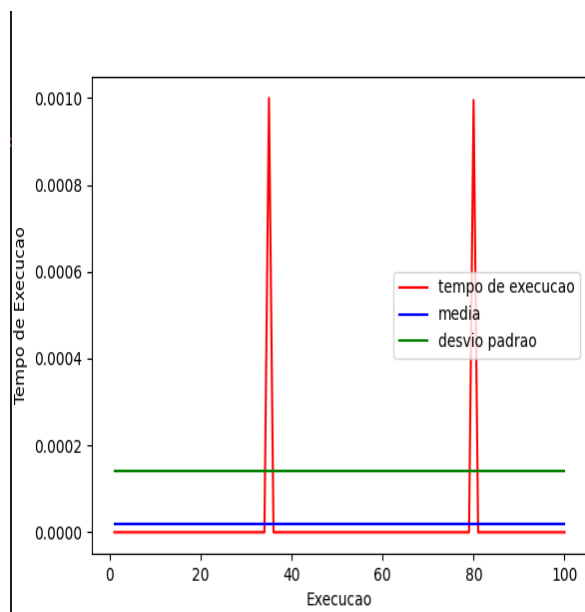
```

vertice[ 0 ] : vermelho
vertice[ 1 ] : azul
vertice[ 2 ] : vermelho
vertice[ 3 ] : verde
vertice[ 4 ] : azul
vertice[ 5 ] : vermelho
vertice[ 6 ] : vermelho
vertice[ 7 ] : Amarelo
vertice[ 8 ] : verde

```

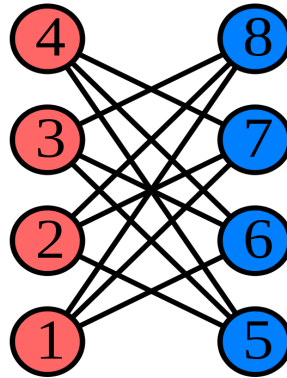


Para este grafo a coloração atingida pelo algoritmo não foi a solução ótima, utilizando 4 cores para colorir o grafo, sendo o ideal 5 cores; Os tempos de execução são mostrados a seguir:

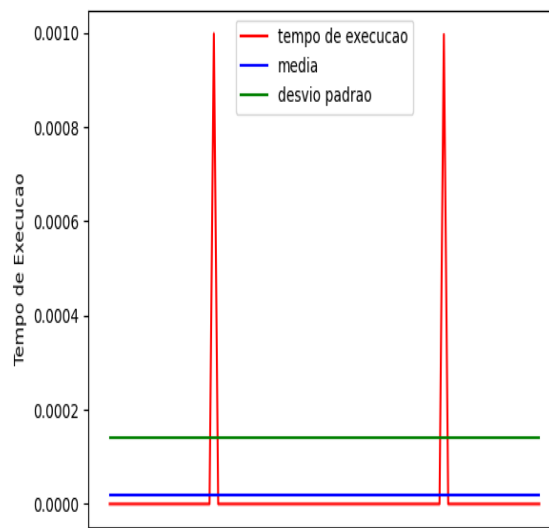


3. Grafo 3

```
vertice[ 0 ] : vermelho  
vertice[ 1 ] : vermelho  
vertice[ 2 ] : vermelho  
vertice[ 3 ] : vermelho  
vertice[ 4 ] : azul  
vertice[ 5 ] : azul  
vertice[ 6 ] : azul  
vertice[ 7 ] : azul
```

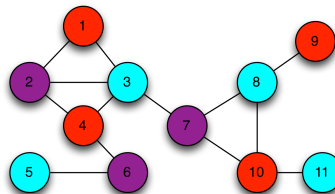


Para o Grafo bipartido a solução ótima foi atingida de 2 cores; Os tempos de execução são mostrados a seguir:

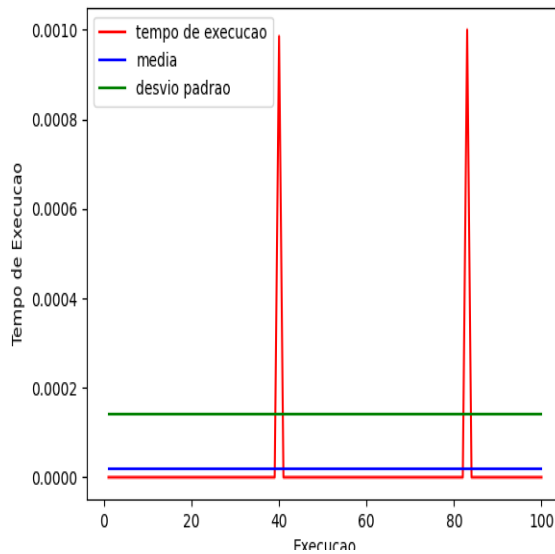


4. Grafo 4

```
vertice[ 0 ] : vermelho  
vertice[ 1 ] : azul  
vertice[ 2 ] : verde  
vertice[ 3 ] : vermelho  
vertice[ 4 ] : vermelho  
vertice[ 5 ] : azul  
vertice[ 6 ] : vermelho  
vertice[ 7 ] : azul  
vertice[ 8 ] : vermelho  
vertice[ 9 ] : verde  
vertice[ 10 ] : vermelho
```

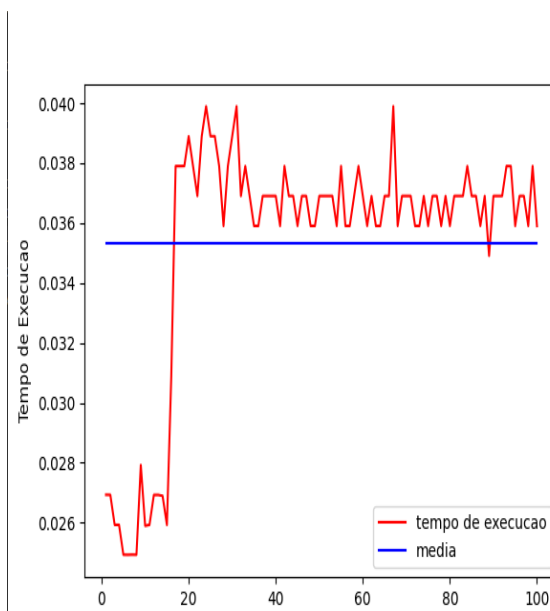


Para este caso também foi obtido um resultado ótimo de 3 cores para a coloração do grafo; Os tempos de execução são mostrados a seguir:



5. Grafo 5

Devido ao tamanho do grafo 5 não será apresentada uma comparação das coloração realizada, porém foram utilizada 3 cores. Não se sabe se este é a coloração ótima; Para os tempos de execução foi obtido um desvio padrão de 0.004048385768314759, neste caso o desvio padrão não foi incluído e os seguintes tempos:



1.4 conclusão

A seção anterior apresentou os resultados para os cinco gráficos descritos neste trabalho. Nesta seção iremos discutir estes resultados. Podemos observar que o algoritmo conseguiu atingir para os grafos 1,3,4 uma solução ótima e para o grafo 2 uma bastante próximo

do ótimo, visto que utilizou apenas uma cor a mais, o que é um bom resultado para o algoritmo. Quanto aos tempos de execução podemos ver que para os grafo 1 ao 4 permanece bastante próximo de 0 segundos, tendo pouquíssima variação como podemos observar pelo valor do desvio padrão. Por isso foi adicionado o grafo 5, cujo numero de vértices e arestas é bastante superior ao aos demais. O grafo 5 manteve tempo de execução entre 0.035 e 0.040 para as 100 execuções e um desvio padrão baixo, o que é um aumento abaixo do esperado, visto a diferença de grandeza entre o grafo 5 e os demais.

1.5 Vídeo

O vídeo apresentando uma execução desse trabalho encontra-se em : <https://web.microsoftstream.com/video/05dafa84-e1f3-44f9-b7fe-269058867a08>