Universidade Federal do Rio Grande do Sul

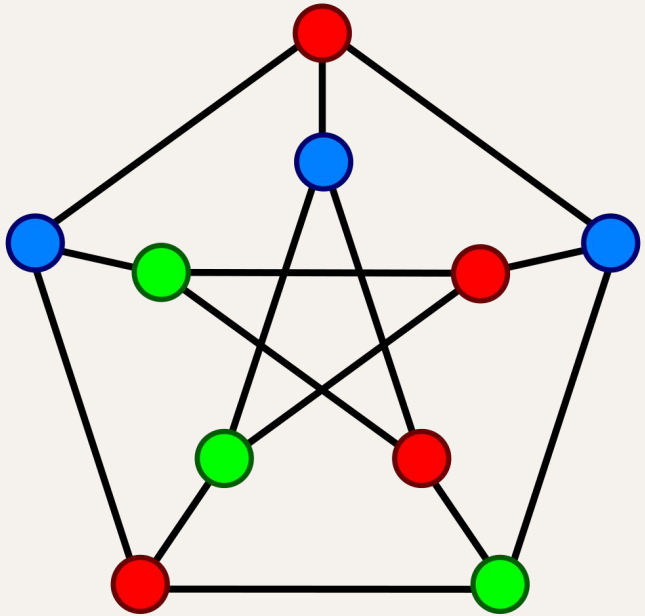
COLORAÇÃO DE GRAFOS

Pablo Diedrich Peralta
Pedro Afonso Tremea Serpa

Complexidade De Algoritmos

● Definição do problema

O problema de decisão da coloração de grafos consiste em, dado um grafo G e um número k, determinar se existe uma forma de colorir os vértices de G com no máximo k cores de tal forma que vértices conectados não tenham a mesma cor.

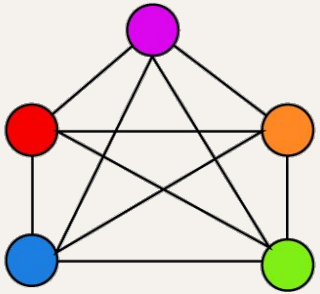


• Número cromático (k)

O número cromático, ou k, é o número de cores disponíveis para colorir o grafo.

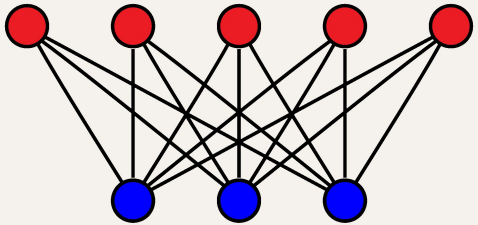
Alguns exemplos de números cromáticos de casos conhecidos:

grafo completo



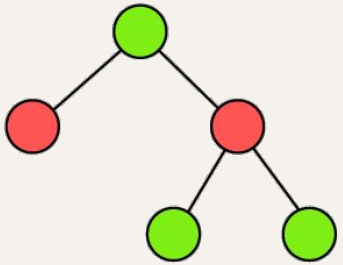
k = nº de vértices

grafo bipartido



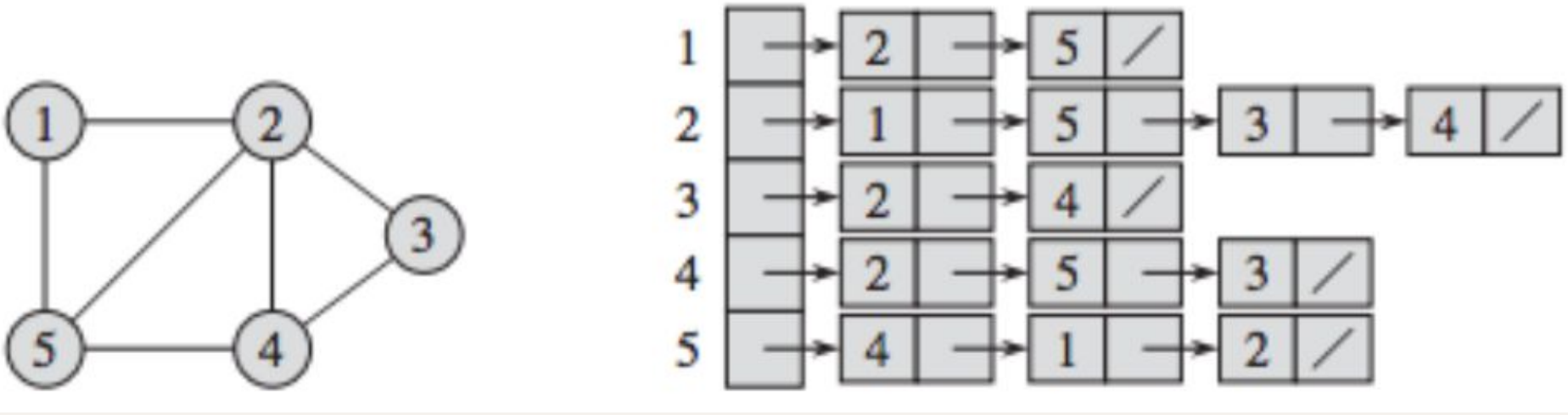
k = 2

grafo árvore



k = 2

- **Estrutura de dados do grafo no algoritmo**



- Coloração de grafos ∈ NP

O(2 * qtdArestas) = O(qtdArestas)

checkColoring.py

```python
def checkColoring(graph):
    for vertex in graph:
        for neighbor in graph[vertex]:
            if vertex[0] == neighbor[0] and vertex[1] == neighbor[1]:
                return False
    return True
```

Problema SAT

Também chamado de problema da satisfatibilidade booleana.

Dada uma fórmula booleana que contenha apenas variáveis e operadores do tipo NOT, AND e OR, é possível atribuir valores lógicos às variáveis a fim de tornar a fórmula verdadeira?

$$EX: \boxed{x_1 \bigvee \neg x_2}$$

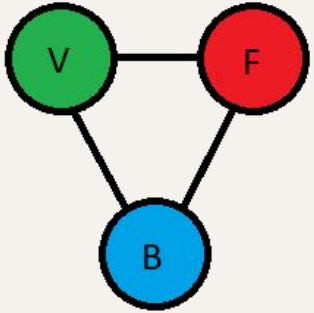
Problema 3-SAT

É um caso especial do problema SAT onde a fórmula booleana está na forma normal conjuntiva e cada cláusula contém 3 variáveis diferentes..

$$EX: \boxed{(x_1 \bigvee x_2 \bigvee x_3) \bigwedge (x_4 \bigvee x_5 \bigvee x_6)}$$

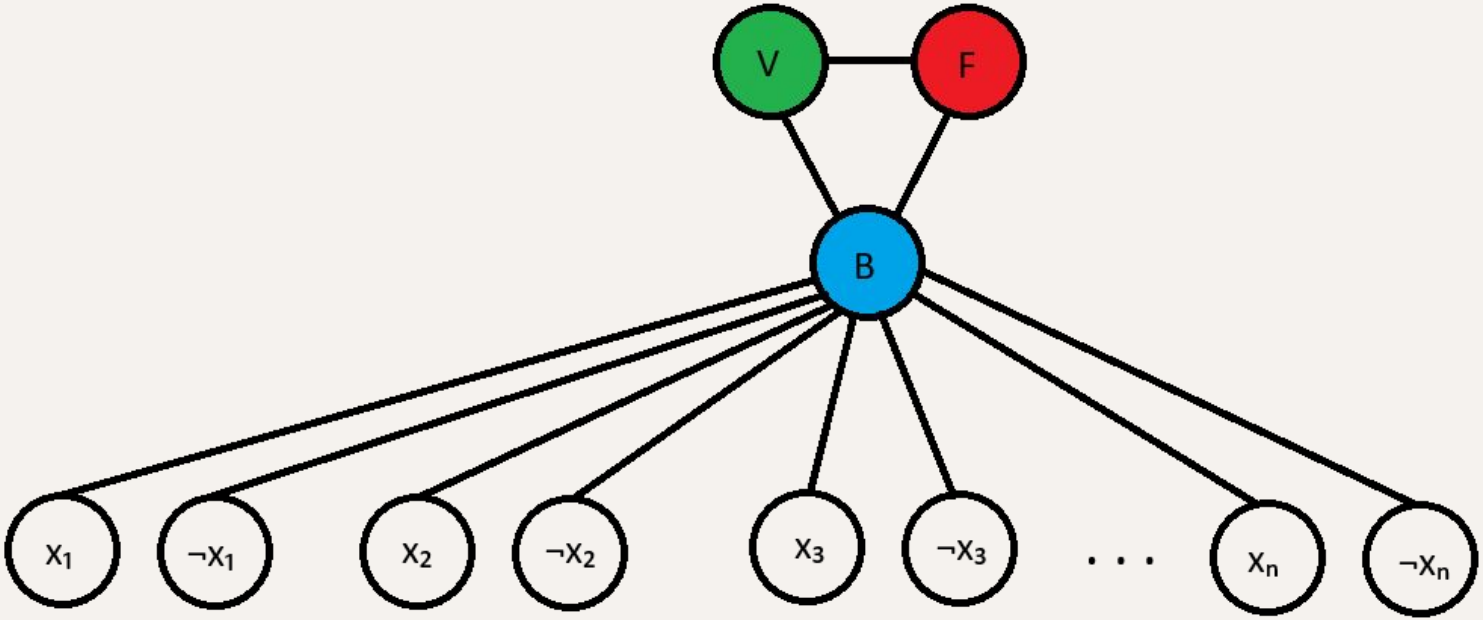
Redução de 3-sat para coloração de grafos

- Criar um grafo com vértices para os valores Verdadeiro (verde), Falso (vermelho), e um vértice extra para um cor adicional B (azul).
- Criar vértices para todas as variáveis presentes no problema sat e suas respectivas negações.



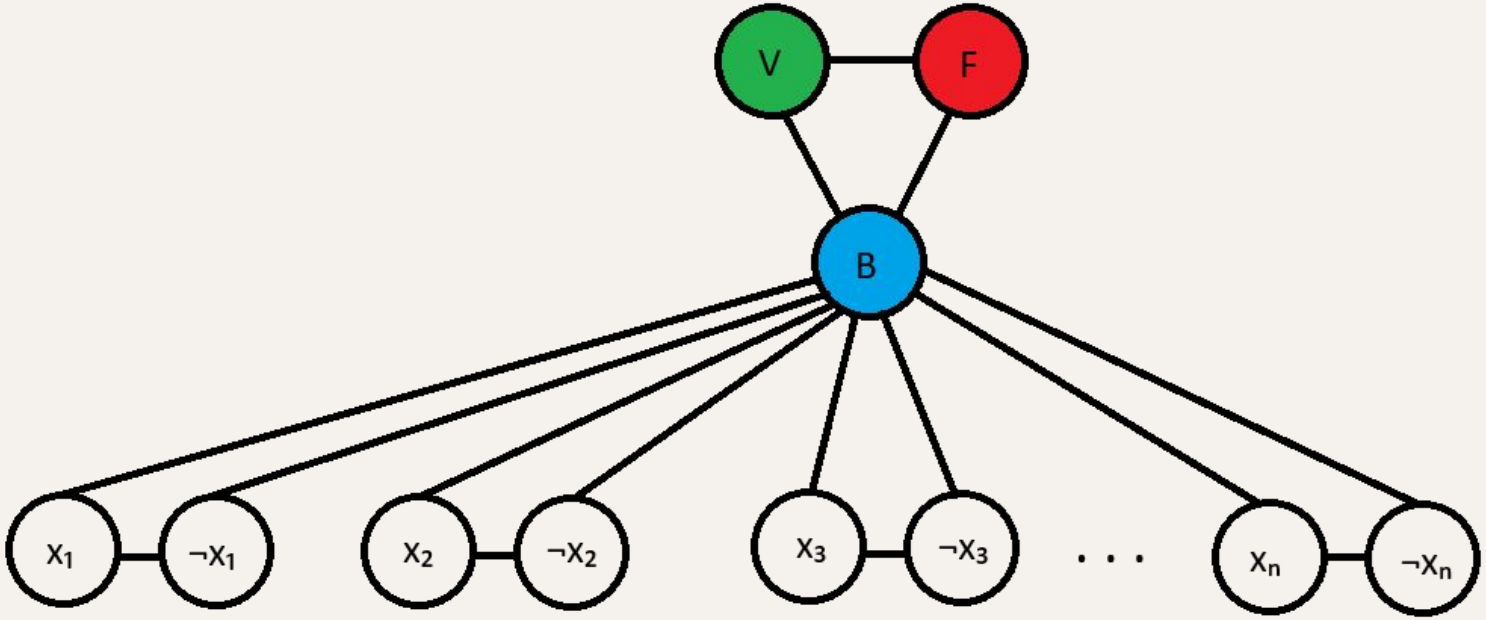
Redução de 3-sat para coloração de grafos

- Todas as variáveis são conectadas à cor Azul. Isso nos garante que todas só poderão receber valorações Verdadeiro (verde) ou Falso (vermelho).



Redução de 3-sat para coloração de grafos

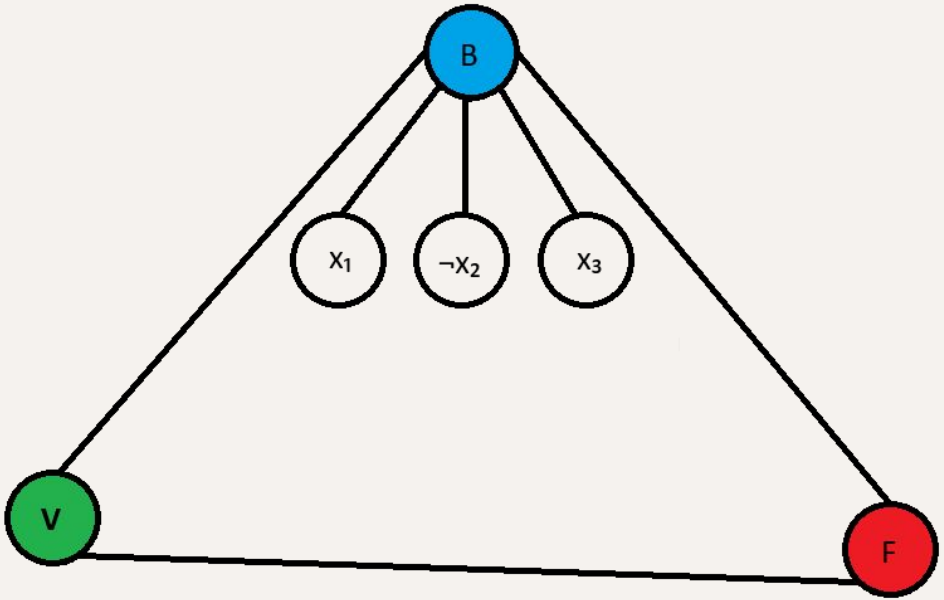
- Todas as variáveis são conectadas à cor Azul. Isso nos garante que todas só poderão receber valorações Verdadeiro (verde) ou Falso (vermelho).



Redução de 3-sat para coloração de grafos

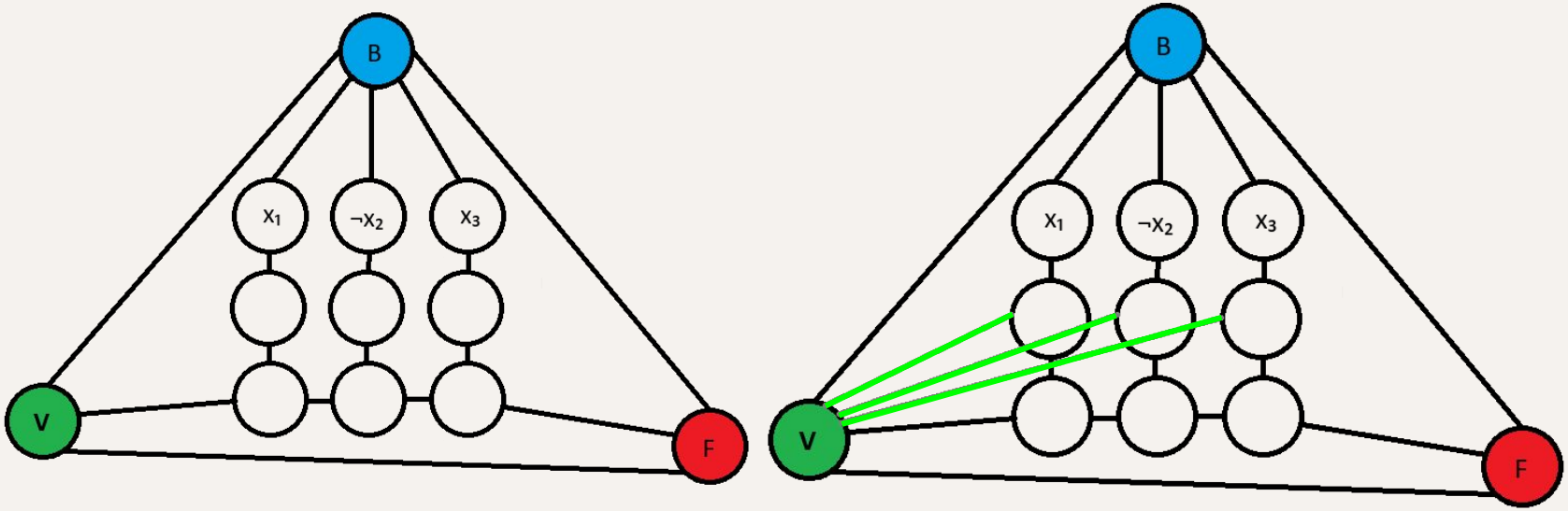
- Dispositivo para análise de cada cláusula.

$$EX: \boxed{(x_1 \bigvee \neg x_2 \bigvee x_3)}$$



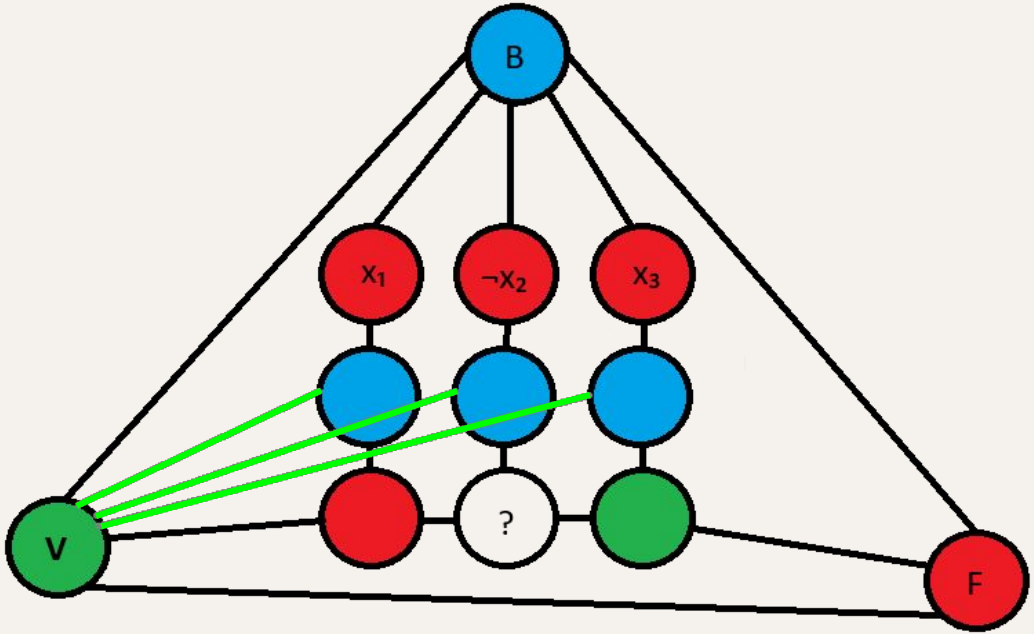
Redução de 3-sat para coloração de grafos

- Dispositivo para análise de cada cláusula.



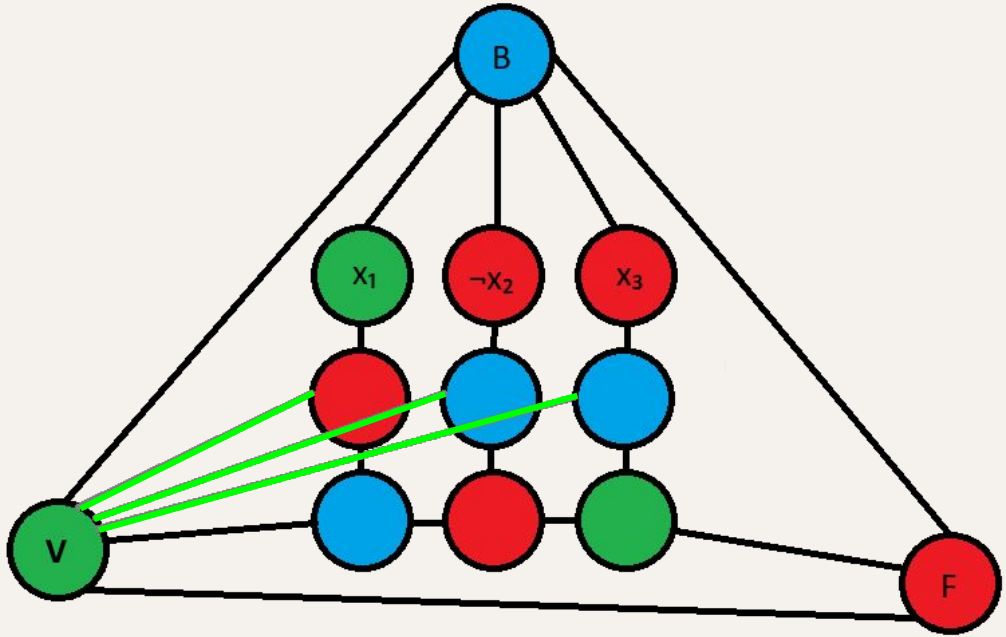
Redução de 3-sat para coloração de grafos

- O caso onde todos os literais são falsos resulta em uma impossibilidade de colorir o grafo de forma que satisfaça o problema da coloração.
- Traduz a insatisfatibilidade da fórmula $(x_1 \bigvee \neg x_2 \bigvee x_3)$ dada a valoração Falso para os três literais.



Redução de 3-sat para coloração de grafos

- Nos caso onde pelo menos um dos literais é verdadeiro, podemos colorir o grafo de forma que satisfaça o problema da coloração.
- Traduz a insatisfatibilidade da fórmula $(x_1 \bigvee \neg x_2 \bigvee x_3)$ dada pelo menos uma valoração Verdadeiro.



3sat-to-3color.py

```python
def sat3_to_3color(sat3):
    graph = {}
    variablesSat3 = readVariables3sat(sat3)

    addVertex(graph, ('B', 'Blue'))
    addVertex(graph, ('R', 'Red'))
    addVertex(graph, ('G', 'Green'))

    addEdge(graph, ('B', 'Blue'), ('R', 'Red'))
    addEdge(graph, ('B', 'Blue'), ('G', 'Green'))
    addEdge(graph, ('R', 'Red'), ('G', 'Green'))

    for element in variablesSat3:

        addVertex(graph, (element, 'Sem cor'))
        addVertex(graph, ('!'+element, 'Sem cor'))

        addEdge(graph, (element, 'Sem cor'), ('!'+element, 'Sem cor'))
        addEdge(graph, (element, 'Sem cor'), ('B', 'Blue'))
        addEdge(graph, ('!'+element, 'Sem cor'), ('B', 'Blue'))


    for i in range(len(sat3)):

        addVertex(graph, ('A'+str(i), 'Sem cor'))
        addVertex(graph, ('B'+str(i), 'Sem cor'))
        addVertex(graph, ('C'+str(i), 'Sem cor'))
        addVertex(graph, ('D'+str(i), 'Sem cor'))
        addVertex(graph, ('E'+str(i), 'Sem cor'))
        addVertex(graph, ('F'+str(i), 'Sem cor'))

        addEdge(graph, ('A'+str(i), 'Sem cor'), ('G', 'Green'))
        addEdge(graph, ('B'+str(i), 'Sem cor'), ('G', 'Green'))
        addEdge(graph, ('C'+str(i), 'Sem cor'), ('G', 'Green'))
        addEdge(graph, ('D'+str(i), 'Sem cor'), ('G', 'Green'))
        addEdge(graph, ('F'+str(i), 'Sem cor'), ('R','Red'))
        addEdge(graph, ('E'+str(i), 'Sem cor'), ('D'+str(i), 'Sem cor'))
        addEdge(graph, ('E'+str(i), 'Sem cor'), ('F'+str(i), 'Sem cor'))
        addEdge(graph, ('E'+str(i), 'Sem cor'), ('B'+str(i), 'Sem cor'))
        addEdge(graph, ('D'+str(i), 'Sem cor'), ('A'+str(i), 'Sem cor'))
        addEdge(graph, ('C'+str(i), 'Sem cor'), ('F'+str(i), 'Sem cor'))
        addEdge(graph, (sat3[i][0], 'Sem cor'), ('A'+str(i), 'Sem cor'))
        addEdge(graph, (sat3[i][1], 'Sem cor'), ('B'+str(i), 'Sem cor'))
        addEdge(graph, (sat3[i][2], 'Sem cor'), ('C'+str(i), 'Sem cor'))

    return graph
```

3sat-to-3color.py

```python
def addVertex(graph, vertex):
    if vertex not in graph:
        graph[vertex] = []


def addEdge(graph, vertex1, vertex2):
    if vertex1 in graph and vertex2 in graph:
        graph[vertex1].append(vertex2)
        graph[vertex2].append(vertex1)


def readVariables3sat(sat3):
    varInSat3 = []
    for i in range(len(sat3)):
        for j in range(len(sat3[i])):
            if sat3[i][j][0] == '!' and sat3[i][j][1:] not in varInSat3:
                varInSat3.append(sat3[i][j][1:])
            elif sat3[i][j][0] =/= '!' and sat3[i][j] not in varInSat3 :
                varInSat3.append(sat3[i][j])
    return varInSat3


def main():

    sat3 = [['x1', 'x4', '!x3'], ['x2', '!x5', 'x1']]
    print(sat3_to_3color(sat3))

if __name__ == "__main__":
    main()
```

3sat-to-3color.py

```python
def sat3_to_3color(sat3):
    graph = {}
    variablesSat3 = readVariables3sat(sat3)

    addVertex(graph, ('B', 'Blue'))
    addVertex(graph, ('R', 'Red'))
    addVertex(graph, ('G', 'Green'))

    addEdge(graph, ('B', 'Blue'), ('R', 'Red'))
    addEdge(graph, ('B', 'Blue'), ('G', 'Green'))
    addEdge(graph, ('R', 'Red'), ('G', 'Green'))

    for element in variablesSat3:

        addVertex(graph, (element, 'Sem cor'))
        addVertex(graph, ('!'+element, 'Sem cor'))

        addEdge(graph, (element, 'Sem cor'), ('!'+element, 'Sem cor'))
        addEdge(graph, (element, 'Sem cor'), ('B', 'Blue'))
        addEdge(graph, ('!'+element, 'Sem cor'), ('B', 'Blue'))


    for i in range(len(sat3)):

        addVertex(graph, ('A'+str(i), 'Sem cor'))
        addVertex(graph, ('B'+str(i), 'Sem cor'))
        addVertex(graph, ('C'+str(i), 'Sem cor'))
        addVertex(graph, ('D'+str(i), 'Sem cor'))
        addVertex(graph, ('E'+str(i), 'Sem cor'))
        addVertex(graph, ('F'+str(i), 'Sem cor'))

        addEdge(graph, ('A'+str(i), 'Sem cor'), ('G', 'Green'))
        addEdge(graph, ('B'+str(i), 'Sem cor'), ('G', 'Green'))
        addEdge(graph, ('C'+str(i), 'Sem cor'), ('G', 'Green'))
        addEdge(graph, ('D'+str(i), 'Sem cor'), ('G', 'Green'))
        addEdge(graph, ('F'+str(i), 'Sem cor'), ('R','Red'))
        addEdge(graph, ('E'+str(i), 'Sem cor'), ('D'+str(i), 'Sem cor'))
        addEdge(graph, ('E'+str(i), 'Sem cor'), ('F'+str(i), 'Sem cor'))
        addEdge(graph, ('E'+str(i), 'Sem cor'), ('B'+str(i), 'Sem cor'))
        addEdge(graph, ('D'+str(i), 'Sem cor'), ('A'+str(i), 'Sem cor'))
        addEdge(graph, ('C'+str(i), 'Sem cor'), ('F'+str(i), 'Sem cor'))
        addEdge(graph, (sat3[i][0], 'Sem cor'), ('A'+str(i), 'Sem cor'))
        addEdge(graph, (sat3[i][1], 'Sem cor'), ('B'+str(i), 'Sem cor'))
        addEdge(graph, (sat3[i][2], 'Sem cor'), ('C'+str(i), 'Sem cor'))

    return graph
```

O(qtdVariaveis)

O(qtdCláusulas)

3sat-to-3color.py

```python
def addVertex(graph, vertex):
    if vertex not in graph:
        graph[vertex] = []


def addEdge(graph, vertex1, vertex2):
    if vertex1 in graph and vertex2 in graph:
        graph[vertex1].append(vertex2)
        graph[vertex2].append(vertex1)


def readVariables3sat(sat3):
    varInSat3 = []
    for i in range(len(sat3)):
        for j in range(len(sat3[i])):
            if sat3[i][j][0] == '!' and sat3[i][j][1:] not in varInSat3:
                varInSat3.append(sat3[i][j][1:])
            elif sat3[i][j][0] != '!' and sat3[i][j] not in varInSat3 :
                varInSat3.append(sat3[i][j])
    return varInSat3


def main():

    sat3 = [['x1', 'x4', '!x3'], ['x2', '!x5', 'x1']]
    print(sat3_to_3color(sat3))

if __name__ == "__main__":
    main()
```

O(1)

O(1)

O(qtdClausulas)

O(max(qtdCláusulas,qtdVariaveis))

- **Referências Bibliográficas**

Kleinberg, J., & Tardos, E. (2006). *Algorithm design*. Pearson. pp. 485-490. ISBN 0321295358

Cormen, Thomas H.. Algoritmos :teoria e prática. Rio de Janeiro: Campus, c2002. ISBN 8535209263.