Teoria dos GRAFOS

Aula 04: Coloração de Grafos

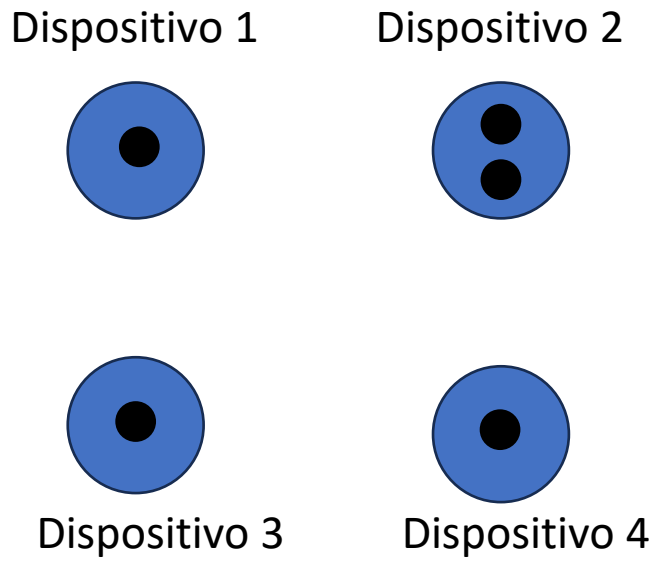
Prof. José Alberto S. Torres

Coloração de Grafos

- A coloração de grafos atraiu a atenção de muitos pesquisadores pela simples razão de que existem **inúmeras aplicações que podem ser modeladas usando colorações de grafos**.
- Colorir um grafo significa **atribuir uma cor a vértices ou arestas**.
- No caso da coloração de arestas, estamos interessados em **atribuir cores de forma que as arestas incidentes com o mesmo vértice tenham cores diferentes**.
- Se um grafo tiver **m arestas**, podemos **usar m cores diferentes para estabelecer uma coloração de aresta válida**.
- O truque é **encontrar o m, que é o número mínimo de cores necessário**.

Situação Exemplo

- Imagine uma **coleção de 4 dispositivos de armazenamento** e, por alguma razão, em determinado momento é necessário **mover cinco blocos de dados entre esses dispositivos**.
- Na situação final, deve-se:
  - Mover um bloco do 1 para o 3
  - Mover um bloco do 2 para o 1
  - Mover um bloco do 2 para o 4
  - Mover um bloco do 4 para o 3
  - Mover um bloco do 3 para o 2

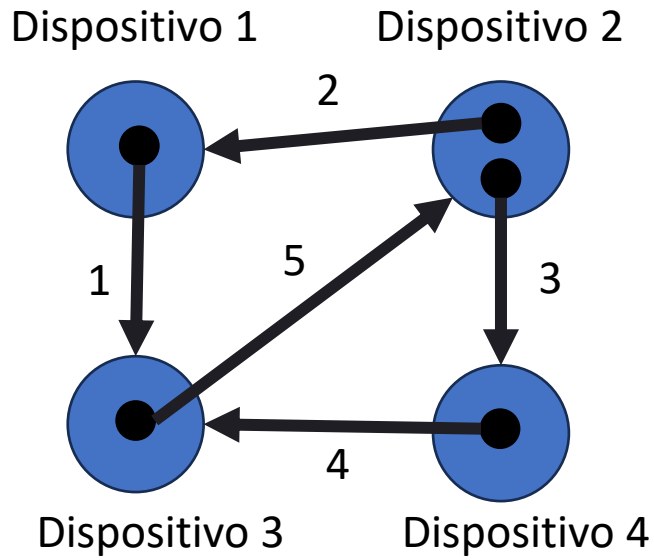


Situação Exemplo

- Há algumas questões a se considerar neste exemplo.
  - Um dispositivo **pode estar envolvido em apenas uma migração por vez**, ou seja, se o bloco b está sendo movido de i para j, então nem i nem j podem estar envolvidos na migração de qualquer outro bloco de dados.
  - Assumimos que **todos os dispositivos estão conectados entre si**. Em outras palavras, é possível migrar dados diretamente de qualquer dispositivo de armazenamento para qualquer outro.
  - Finalmente, assumimos que, se b deve ser movido para j, então j tem espaço suficiente para armazenar b, ou seja, não é necessário primeiro disponibilizar espaço em j, por exemplo, migrando outro bloco b ’ de j para, digamos, o dispositivo k.

Situação Exemplo

- Podemos **mover cada bloco um de cada vez**, o que levará **cinco unidades de tempo**.

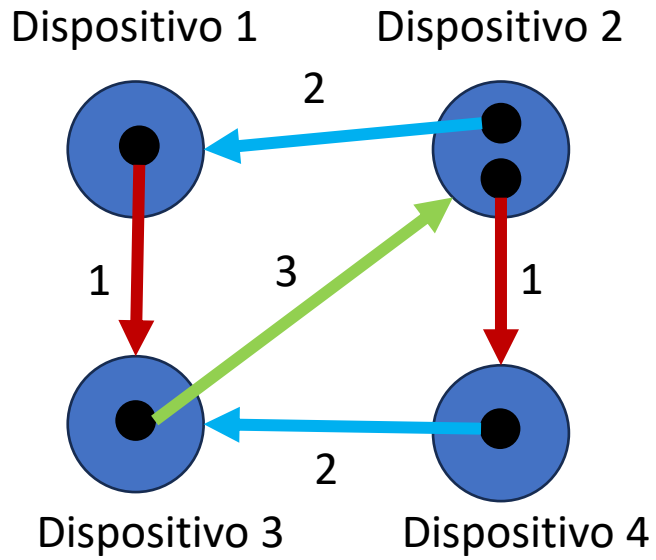


Situação Exemplo – **uso de cores**

- A regra do problema ("um dispositivo não pode fazer duas coisas ao mesmo tempo") é traduzida para uma regra de coloração: "**Todos os arcos ou arestas incidentes a um vértice precisam ter uma cor diferente.**”
- Isso significa que **todas as setas que "saem" ou "chegam" em um mesmo dispositivo devem ser pintadas com cores** (agendadas em tempos) diferentes.

Situação Exemplo – **uso de cores**

- A figura ao lado apresenta um escalonamento mais eficiente, na qual várias migrações ocorrem simultaneamente.
- A situação esboçada corresponde a uma coloração mínima de arestas com três cores diferentes.
- Especificamente, podemos completar a migração **em três unidades de tempo, em vez de cinco**.



Coloração de Arestas

- Formalmente, as **colorações de arestas** são definidas da seguinte forma:
  - Considere um grafo G **conectado e sem loop**.
  - G é **k-arestas coloríveis** se existe um **particionamento de E(G)** em **k conjuntos disjuntos E1,..., Ek** tal que **nenhuma aresta do mesmo Ei seja incidente com o mesmo vértice**.
  - Cada Ei é um subconjunto de E, o que significa que $\forall i: E_i \subseteq E$
  - Esses conjuntos juntos constituem E, isto é, $E_1 \cup E_2 \cup \cdots \cup E_k = E$.
  - Nenhum dos dois conjuntos tem um elemento em comum, o que pode ser escrito matematicamente como $\forall i \neq j: E_i \cap E_j = \emptyset$

Coloração de vértices

- Em essência, o problema se resume a encontrar **uma coloração dos vértices de um grafo** (simples e conexo) tal que **não haja dois vértices adjacentes com a mesma cor**.
- Considere um grafo conexo simples G. G é colorível em k-vértices se houver um particionamento de V(G) em k conjuntos disjuntos V1,..., Vk tal que não haja dois vértices do mesmo Vi adjacentes.

Formulação matemática

- A formulação matemática de que não há dois vértices adjacentes de um mesmo Vi é:

$$\forall Vi, \forall x, y \in Vi: \nexists e \in E(G): e = \langle x, y \rangle$$

- onde ∄ deve ser lido como “não existe” e onde onde cada conjunto Vi representa uma cor.
- Em outras palavras, para todos os pares de vértices distintos em Vi, "Dois vértices ligados por uma aresta (vizinhos) não podem ter a mesma cor."

Coloração de Vértices

- Desta forma, **o problema de coloração de vértices para um dado grafo G é encontrar o mínimo k para o qual G é k-vértice colorível**.
- Esse mínimo k é chamado de **número cromático de G**, denotado como χ(G).

Em resumo...

- Uma **coloração válida** de um grafo é, na verdade, um **particionamento do conjunto de vértices V** do grafo.
- Pense que cada "pedaço" ou subconjunto (V1, V2, ..., Vk) é o conjunto de todos os vértices que recebem a mesma cor...

  V1 = Conjunto de todos os vértices coloridos de **Vermelho**.

  V2 = Conjunto de todos os vértices coloridos de **Azul**.

  V3 = Conjunto de todos os vértices coloridos de **Verde**

Em resumo...

- Imagine um grafo com 5 vértices: {v1, v2, v3, v4, v5}. Após colorir, obtemos o seguinte resultado:
  - v1 e v4 são **Azuis**.
  - v2 e v5 são **Verdes**.
  - v3 é **Vermelho**.
- Isso criou a seguinte partição do conjunto de vértices:
  - V_Azul = {v1, v4}
  - V_Verde = {v2, v5}
  - V_Vermelho = {v3}

Em resumo...

- **Cada grupo é um subconjunto do original?** Sim, todos os vértices pertenciam ao conjunto original.
- **A união dos grupos forma o conjunto original?** Sim, {v1, v4} ∪ {v2, v5} ∪ {v3} = {v1, v2, v3, v4, v5}. Todos os vértices foram coloridos.
- **Os grupos têm elementos em comum?** Não. Nenhum vértice tem mais de uma cor.

O Problema do **Agendamento de Aulas**

- Imagine que você é o responsável por montar o horário de uma faculdade.
  - **O Desafio:** Você tem uma lista de aulas (Cálculo, Física, Literatura, etc.) e precisa decidir em qual horário cada uma acontecerá (Segunda 8h, Segunda 10h, Terça 8h, etc.).
  - **A Restrição Principal:** Se um grupo de alunos (por exemplo, a turma de Engenharia do 1º semestre) precisa cursar tanto Cálculo quanto Física, essas duas aulas **não podem** acontecer no mesmo horário. Elas entram em conflito.
  - **O Objetivo:** Criar um horário funcional usando o **menor número possível de blocos de tempo** (slots), para que o dia não se estenda desnecessariamente.

O Problema do **Agendamento de Aulas**

- Como você utilizaria coloração de grafos para modelar e resolver esse problema?

O Problema do **Agendamento de Aulas**

- Podemos modelar da seguinte forma:
  - **Vértices (os pontos):** Cada aula se torna um vértice no nosso grafo. Se temos 50 aulas, teremos 50 vértices.
  - **Arestas (as linhas):** Nós desenhamos uma linha (aresta) entre duas aulas se, e somente se, elas têm um conflito. Ou seja, se existe pelo menos um grupo de alunos que precisa assistir a ambas.
  - **Cores:** Cada horário (slot de tempo) será representado por uma cor. Por exemplo, "Segunda às 8h" pode ser a cor Vermelha, "Segunda às 10h" a cor Azul, e assim por diante.
- Ao final, temos um grafo onde os vértices são as aulas e as arestas conectam as aulas que não podem acontecer ao mesmo tempo.

O Problema do **Agendamento de Aulas**

- A regra da coloração de grafos é que **dois vértices conectados por uma aresta não podem ter a mesma cor**.
- Isso se encaixa perfeitamente no problema: **se duas aulas têm conflito, elas precisam de horários diferentes – ou cores diferentes**.
- Portanto, o problema de **encontrar o número mínimo de horários** é exatamente o mesmo que **encontrar o número mínimo de cores para colorir o grafo**.
- Esse número mínimo é chamado de **Número Cromático**, ou **$\chi(G)$**.
- O número cromático **$\chi(G)$ é limitado pelo grau máximo do grafo $\Delta(G)$.**
- Se o vértice com mais arestas tem 8 conflitos, sabemos que precisaremos de no máximo 9 cores ($\chi(G) \leq \Delta(G) + 1$), o que já ajuda a limitar o problema.

Por que χ(G) horários são suficientes?

- Por definição, o número cromático **χ(G) é a quantidade de cores necessárias para uma coloração válida**.
- Numa coloração válida, **todas as aulas que recebem a mesma cor (por exemplo, a cor "Vermelho") não estão conectadas por arestas**.
- Traduzindo de volta: **todas as aulas no grupo "Vermelho" não têm conflitos entre si.**
- Portanto, **podemos com segurança agendar todas as aulas "Vermelhas" no mesmo horário**. O mesmo vale para as "Azuis", "Verdes", etc. Se temos χ(G) cores, precisamos de apenas χ(G) horários.
- **Isso prova que χ(G) é um número de horários possível e suficiente**

IESB
Centro Universitário

O Problema do **Agendamento de Aulas**

- Na prática, **encontrar o valor de χ(G) para um grafo muito grande** (como o de uma universidade real) **é um problema computacionalmente "difícil”** já que é preciso testar muitas combinações.

Atividade 1

De acordo com o apresentado, modele o seguinte problema usando coloração de grafos – desenhe o grafo colorido.

Aulas a serem agendadas (6 no total):

1. **M**atemática Básica
2. **Á**lgebra Linear
3. **C**álculo I
4. **F**ísica I
5. **Q**uímica Geral
6. **P**rogramação I

Grupos de Alunos e suas Matérias Obrigatórias:

- **Turma de Engenharia:** Precisa cursar Cálculo I, Física I e Álgebra Linear.
- **Turma de Ciência da Computação:** Precisa cursar Matemática Básica, Programação I e Álgebra Linear.
- **Turma de Química:** Precisa cursar Química Geral e Física I

IESB Centro Universitário

Resolução - **Vértices**

Lista de Vértices

Matemática Básica

Álgebra Linear

Cálculo I

Física I

Química Geral

Programação I

Resolução - **Lista de Conflitos**

A partir dos grupos, identificamos as aulas que **não podem** ocorrer no mesmo horário:

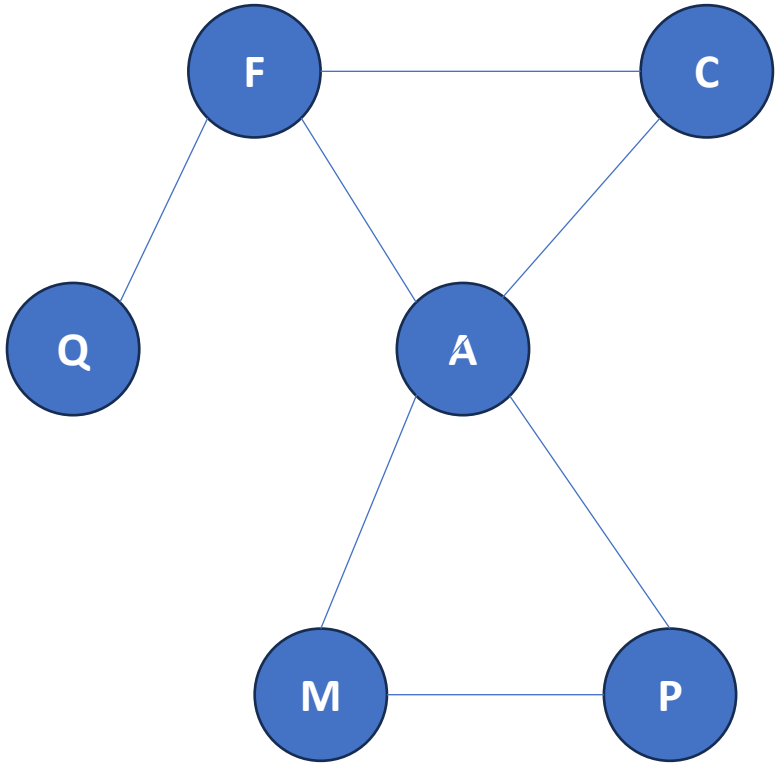
- Da Engenharia: Cálculo (C) conflita com Física (F), Cálculo (C) conflita com Álgebra (A), e Física (F)conflita com Álgebra (A).
- Da C. da Computação: Matemática (M) conflita com Programação (P), Matemática (M) conflita com Álgebra (A), e Programação (P) conflita com Álgebra (A).
- Da Química: Química (Q) conflita com Física (F).
- **Resumo dos Conflitos:** C-F, C-A, F-A, M-P, M-A, P-A, Q-F.

IESB
Centro Universitário

Construir o Grafo

1. Criamos 6 vértices, um para cada matéria (M, A, C, F, Q, P).
2. Desenhamos as arestas (linhas) de acordo com a lista de conflitos:

- Os vértices **A, C, F** formam um triângulo (todos conectados entre si).
- Os vértices **A, M, P** também formam um triângulo.
- O vértice **A** é central, conectando os dois "grupos".
- O vértice **Q** está conectado apenas ao **F**.

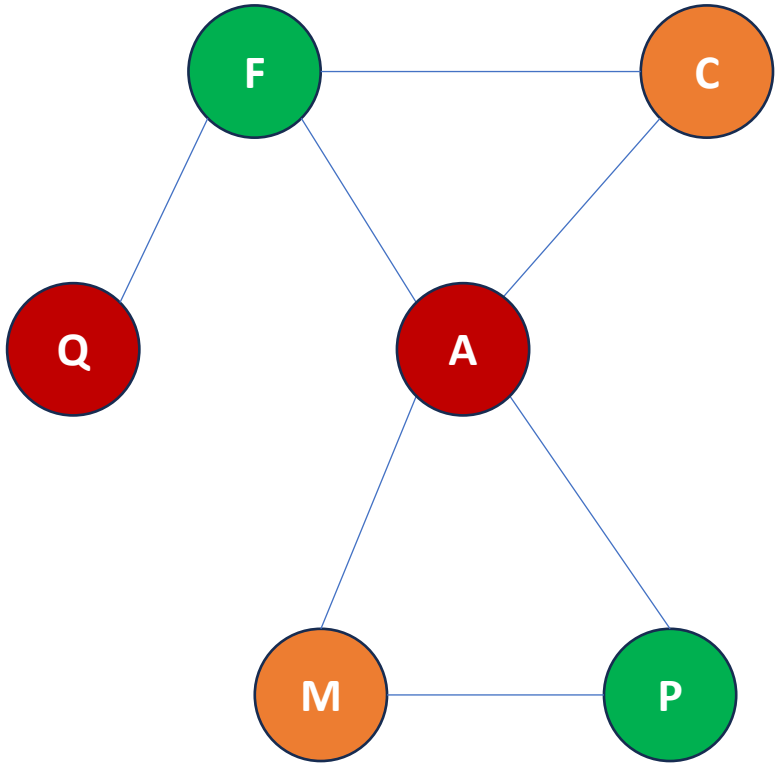


Colorir o Grafo

- Temos "triângulos" no grafo (ex: A-C-F).
- Vértices que formam um **triângulo** precisam, obrigatoriamente, de **3 cores distintas.**
- Portanto, já sabemos que o número mínimo de horários será **pelo menos 3**. $\chi(G) \geq 3$.

Tentativa com 3 cores

- **Cor 1 (Vermelho):** Vamos começar com a aula que tem mais conflitos, Álgebra (A). Vamos pintá-la de Vermelho.
- **Cor 2 (**Laranja**):** Cálculo (C) é vizinho de A (Vermelho), então não pode ser Vermelho. Vamos pintá-lo de Laranja. Matemática (M) também é vizinha de A (Vermelho). Podemos pintá-la de Laranja também, já que M e C não são vizinhos.
- **Cor 3 (Verde):** Física (F) é vizinha de A (Vermelho) e C (Azul). Precisa de uma terceira cor. Vamos pintá-la de Verde. Programação (P) é vizinha de A (Vermelho) e M (Azul). Também precisa ser Verde.
- A Sobra: Falta a Química (Q). Ela é vizinha apenas de F (Verde). Portanto, ela não pode ser Verde, mas pode ser Vermelho ou Laranja. Vamos atribuir a cor Vermelho.



Backtracking

- É uma das **técnicas de resolução de problemas** mais fundamentais na computação, sendo utilizada para coloração de grafos.
- Imagine que você precisa resolver um problema que envolve tomar uma série de decisões para chegar a uma solução final.
- O backtracking explora as possibilidades de forma incremental, uma decisão de cada vez.
- A regra de ouro do backtracking é: **avance por um caminho enquanto ele parecer promissor. Assim que ele se mostrar inviável, volte atrás (retroceda) exatamente o suficiente para tentar a próxima opção.**

Backtracking

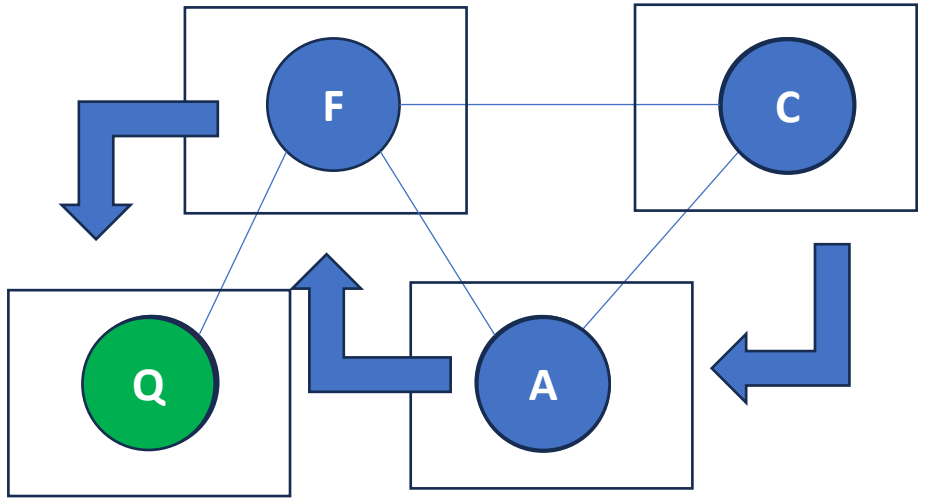
- **Ponto de Partida**: Você está no início do problema, com o grafo G e uma lista ordenada de vértices para colorir (ex: ['A', 'F', 'M', 'P', 'C', 'Q']). Nenhuma cor foi atribuída ainda.
- **Primeira Decisão**: Você chega ao primeiro vértice da lista ('A') e precisa decidir qual cor usar (digamos que estamos tentando com k=3 cores: Cor 1, Cor 2, Cor 3). Você escolhe o caminho da **Cor 1** para o vértice 'A'.
- **Exploração**: Você avança para o próximo vértice da lista ('F'). Ele te leva a uma nova decisão. Você escolhe a primeira cor válida para 'F' que não entre em conflito com seus vizinhos já coloridos (neste caso, 'A' é Cor 1, então você escolhe a **Cor 2**). Você continua avançando para os próximos vértices, sempre escolhendo a primeira cor válida disponível.
- **Beco Sem Saída (Falha):** De repente, você chega a um vértice (digamos, 'P') e descobre que **nenhuma cor é válida**. Todos os seus vizinhos já usaram todas as k=3 cores disponíveis. Você não pode colorir 'P' sem violar as regras. Isso significa que a sequência de decisões de cores que você tomou até agora estava errada.
- **O Retrocesso (Backtracking):** O que você faz? Você não apaga todas as cores e recomeça do zero. Você **retrocede** para o vértice anterior que você coloriu (digamos, 'M') e **desfaz a cor dele**, tratando-o como não colorido novamente.
- **Nova Tentativa:** Naquele vértice 'M', você sabe que a cor que você tentou antes (ex: Cor 2) te levou a um beco sem saída. Agora, você tenta a **próxima cor válida disponível** para 'M' (ex: Cor 3). Se não houver outra opção válida para 'M', você retrocede mais um passo, para o vértice anterior a ele.
- **Repetir:** Você repete esse processo: explora a nova sequência de cores a partir da sua nova decisão. Se essa nova escolha permitir que todos os vértices sejam coloridos, **solução encontrada**. Se levar a outro beco sem saída, você retrocede novamente para a última decisão e tenta a próxima alternativa, até que todas as possibilidades tenham sido exploradas de forma inteligente.

IESB
Centro Universitário

Backtracking

- K = 2 = VERMELHO e VERDE
- V = [Q, F, A, C]

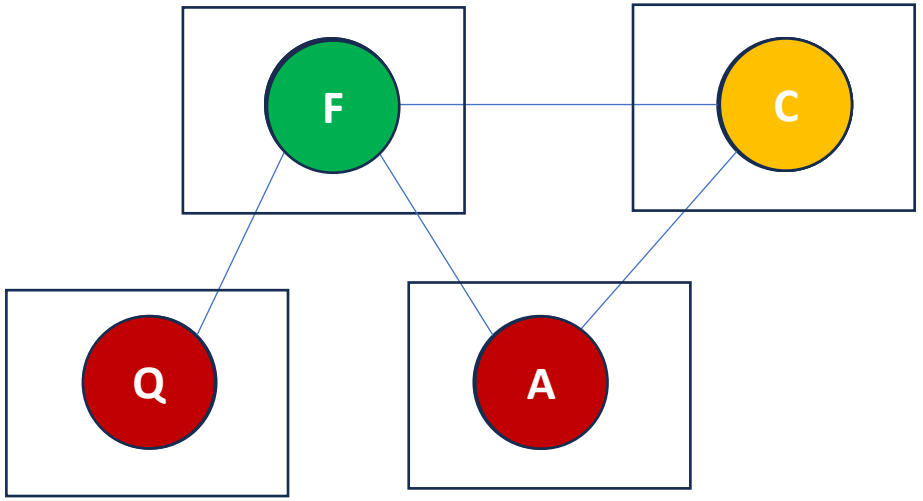
Sem opções porque já tentamos vermelho...



O algoritmo agora tentará explorar o caminho novamente com Q = verde. Isso levará novamente a uma falha. Ao retornar a Q, não haverá novos caminhos a seguir,

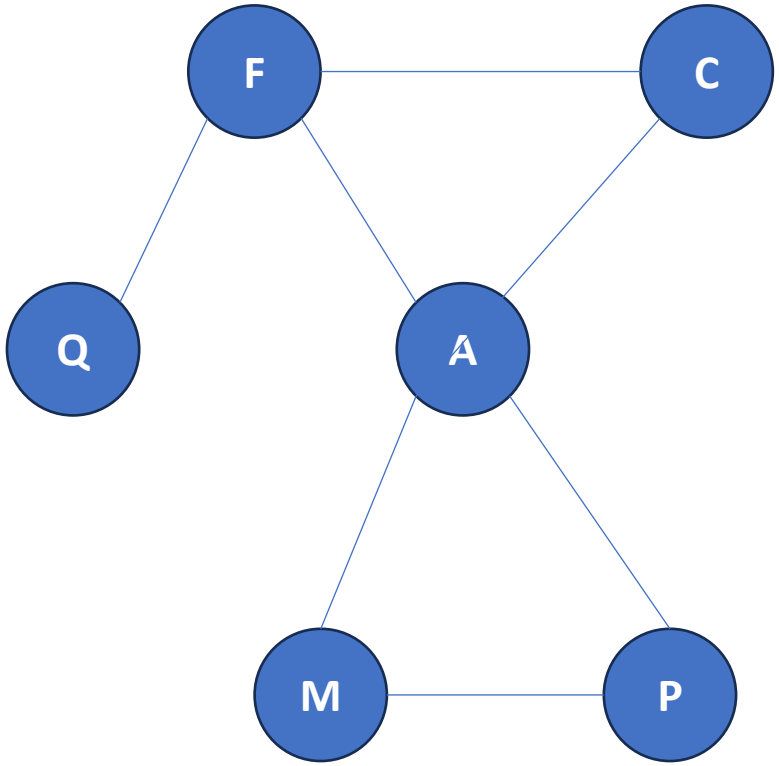
Backtracking

- K = 3 = VERMELHO, VERDE E LARANJA
- V = [Q, F, A, C]



Atividade 2

- Construa o código para o problema do conflito de aulas, de acordo com o Grafo construído na atividade anterior.
- Utiliza a lógica do algoritmo de backtracking.
- A saída deve reportar a quantidade de cores, bem como a cor de cada vértice.



Atividade 2

```python
if __name__ == "__main__":

    aulas = ['M', 'A', 'C', 'F', 'Q', 'P']
    g = GrafoEsparso(labels=aulas)


    # Os conflitos são as nossas arestas
    g.adicionar_aresta('C', 'F')
    g.adicionar_aresta('C', 'A')
    g.adicionar_aresta('F', 'A')
    g.adicionar_aresta('M', 'P')
    g.adicionar_aresta('M', 'A')
    g.adicionar_aresta('P', 'A')
    g.adicionar_aresta('Q', 'F')

    for v_origin, v_dest in g.get_arestas():
        print(f"- Aula {v_origin} tem conflito com: {v_dest}")
    print("-" * 30)

    numero_minimo_horarios, cores_atribuidas = g.colorir_grafo()
    print(f"Número mínimo de horários necessários: {numero_minimo_horarios}")
    print(cores_atribuidas)
```

```
e-Pessoal/IESB/Grafos/CodigoPythonGrafos/aula_04/grafo.py
Aresta adicionada entre C e F
Aresta adicionada entre C e A
Aresta adicionada entre F e A
Aresta adicionada entre M e P
Aresta adicionada entre M e A
Aresta adicionada entre P e A
Aresta adicionada entre Q e F
- Aula M tem conflito com: P
- Aula M tem conflito com: A
- Aula A tem conflito com: C
- Aula A tem conflito com: F
- Aula A tem conflito com: P
- Aula C tem conflito com: F
- Aula F tem conflito com: Q
------------------------------
Tentando colorir a aula M com 1 horários...
Tentando colorir a aula A com 1 horários...
Tentando colorir a aula M com 2 horários...
Tentando colorir a aula A com 2 horários...
Tentando colorir a aula C com 2 horários...
Tentando colorir a aula F com 2 horários...
Tentando colorir a aula A com 2 horários...
Tentando colorir a aula C com 2 horários...
Tentando colorir a aula F com 2 horários...
Tentando colorir a aula M com 3 horários...
Tentando colorir a aula A com 3 horários...
Tentando colorir a aula C com 3 horários...
Tentando colorir a aula F com 3 horários...
Tentando colorir a aula Q com 3 horários...
Tentando colorir a aula P com 3 horários...
Número mínimo de horários necessários (Número Cromático χ(G)): 3

{'M': 1, 'A': 2, 'C': 1, 'F': 3, 'Q': 1, 'P': 3}
```

Teorema das **quatro cores**

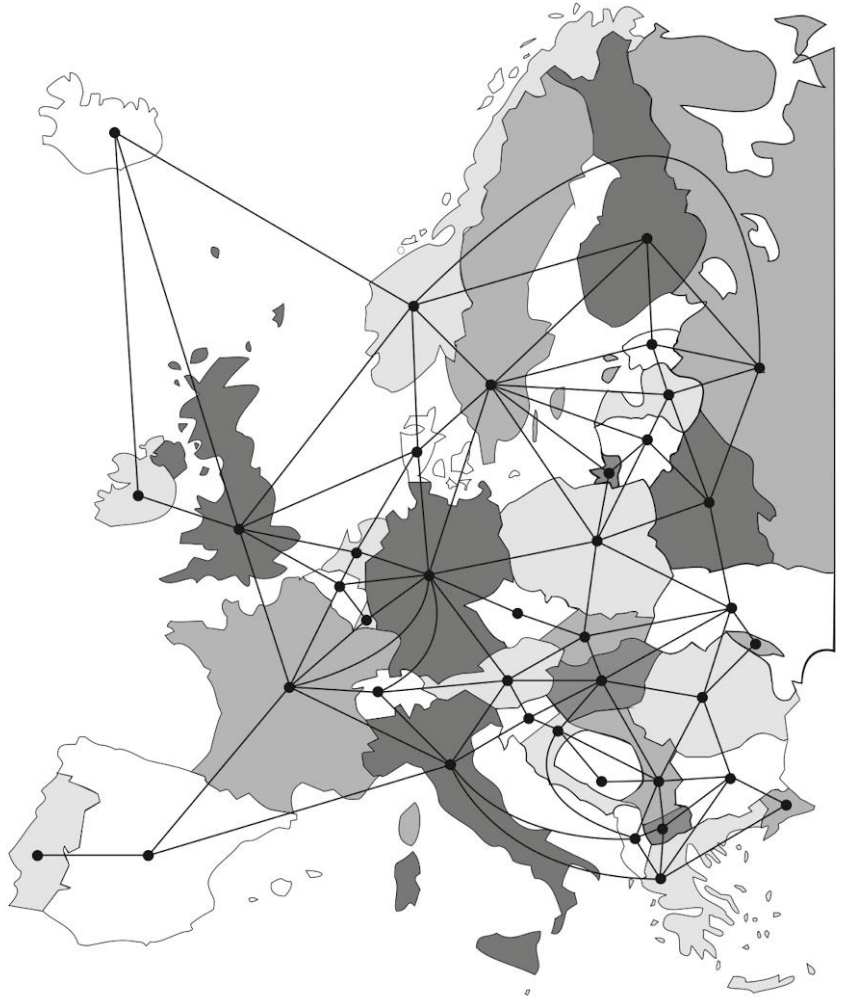
- Para qualquer mapa (ou grafo planar), você **nunca** precisará de mais de **4 cores** para colori-lo de forma que regiões vizinhas tenham cores diferentes. $\chi(G) \leq 4$.
- Por mais de um século, ninguém conseguiu provar isso matematicamente de forma tradicional.
- A prova que finalmente surgiu (de Appel e Haken) não foi uma dedução lógica elegante, mas através de uma prova de "força bruta”.

Teorema das **quatro cores**

- Eles reduziram o problema a cerca de 2.000 casos fundamentais.
- Escreveram um programa de computador para testar cada um desses casos, um por um.
- Após **1.200 horas de computação**, o computador confirmou que todos os casos funcionavam.

Teorema das **quatro cores**

- A comunidade matemática ficou desconfortável pois, se houvesse um bug no programam, não poderia ser verificada por um ser humano.
- Hoje, a prova é aceita, mas ainda não existe uma prova matemática para esse teorema.




DÚVIDAS