

ALGORITMOS EM GRAFOS

COLORAÇÃO

Prof. Alexei Machado

PUC MINAS

CIÊNCIA DA COMPUTAÇÃO

Coloração de grafos

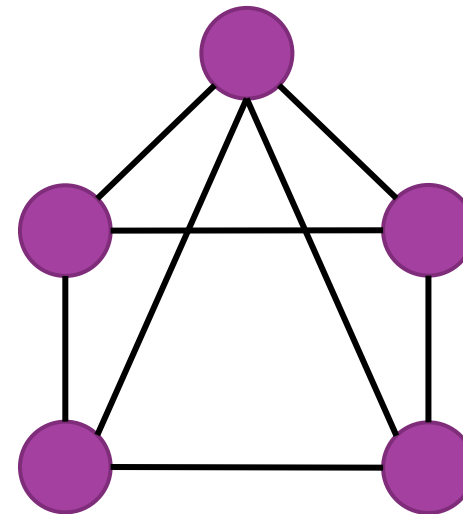
2

- Francis Guthrie (1852) percebeu que era possível colorir o mapa da Inglaterra usando apenas 4 cores
- Mas... Seriam 4 cores suficientes para colorir qualquer decomposição do plano em regiões?
- Em 1976, usando grafos, Haken e Appel mostram que a resposta era afirmativa

Coloração de grafos

3

- Dado um grafo G , como colorir seus vértices de maneira que vértices adjacentes não tenham as mesmas cores?
- Qual o menor número possível para esta **coloração**?



Coloração de grafos

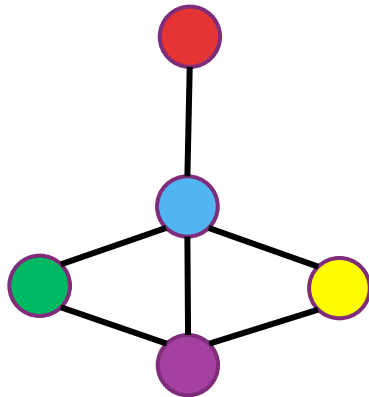
4

- Dado um grafo G conexo simples, uma **coloração** de G é uma atribuição de cores aos vértices de G de maneira que cores diferentes são atribuídas a vértices adjacentes
- Se existe uma coloração para um grafo G que utiliza K cores, então G é um grafo **K -colorido**

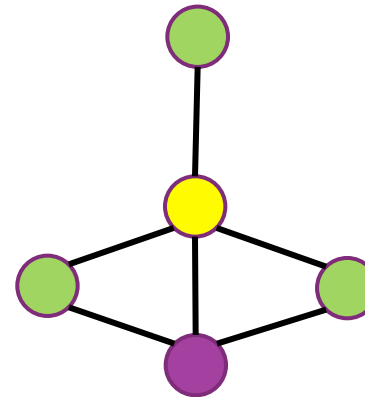
Número cromático

5

- O **número cromático** de um grafo G , $X(G)$, é o **menor** número K para o qual G é K -colorido



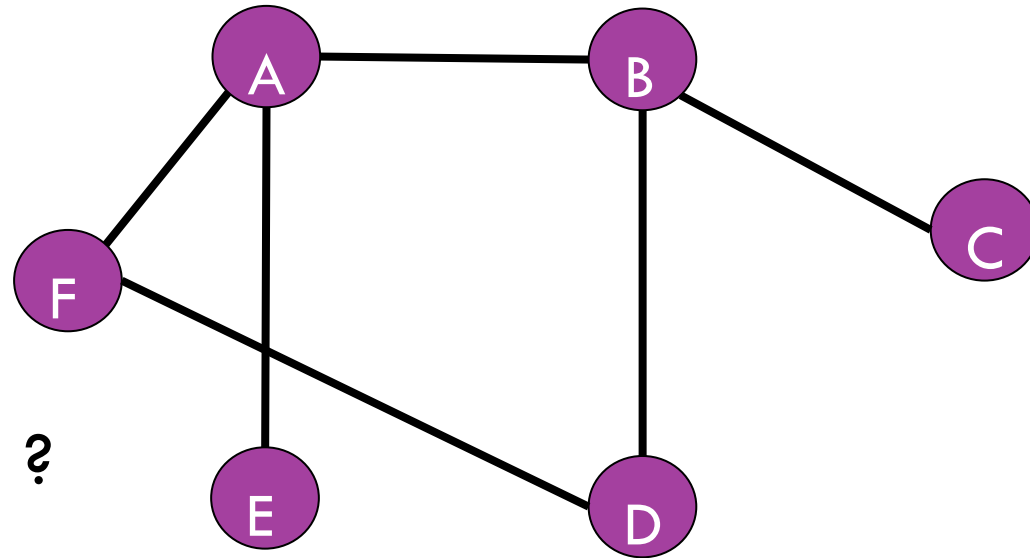
5-colorido



$X(G) = 3$

Exercício

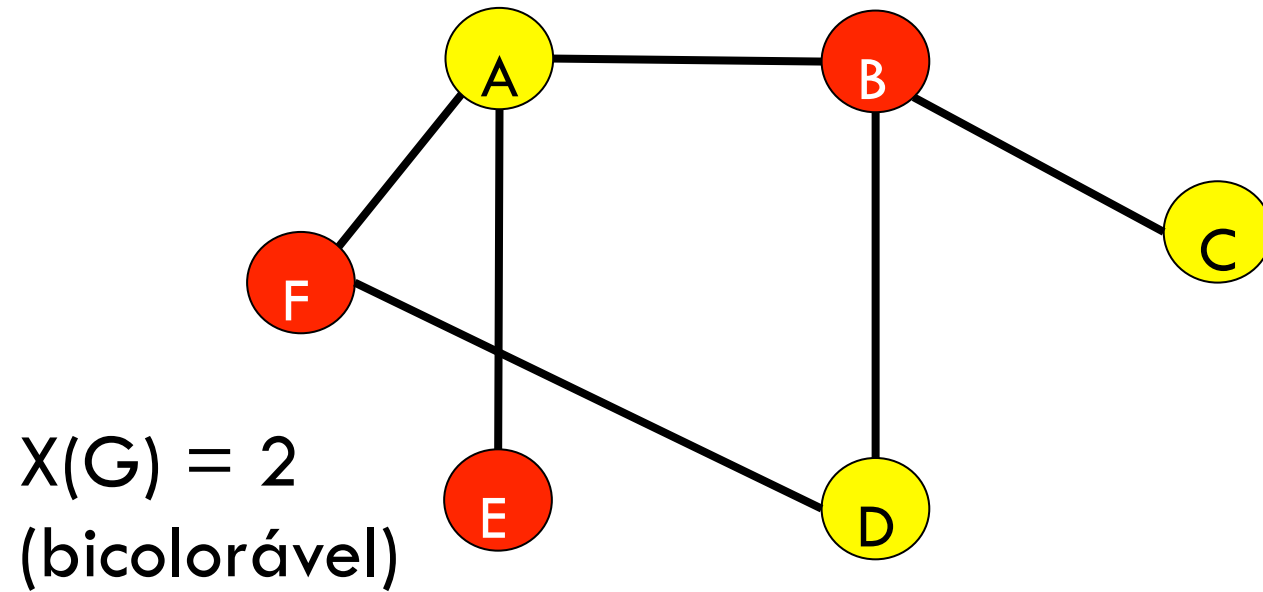
6



$X(G) = ?$

Exercício

7



Coloração de grafos

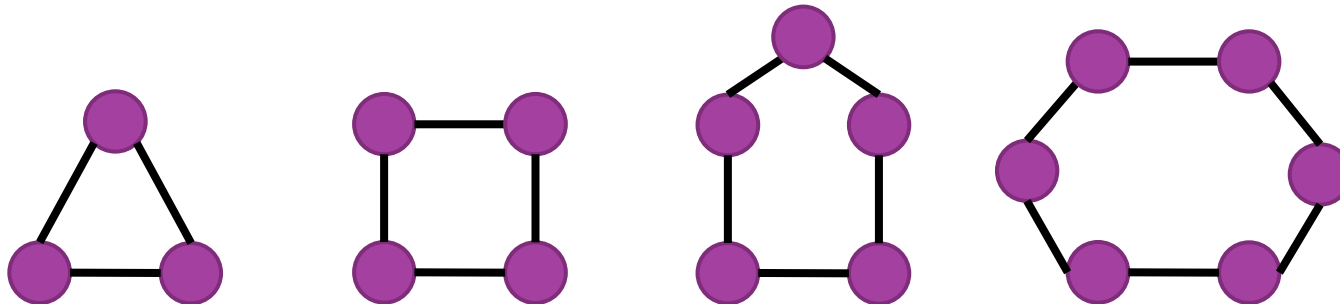
8

- Coloração se dá em grafos conexos simples:
 - Desconsiderar grafos desconexos. As cores utilizadas em um componente não têm efeito sobre as do outro componente.
 - Arestas paralelas não afetam a coloração.
 - Grafo não pode ter loops.

Coloração de circuitos

9

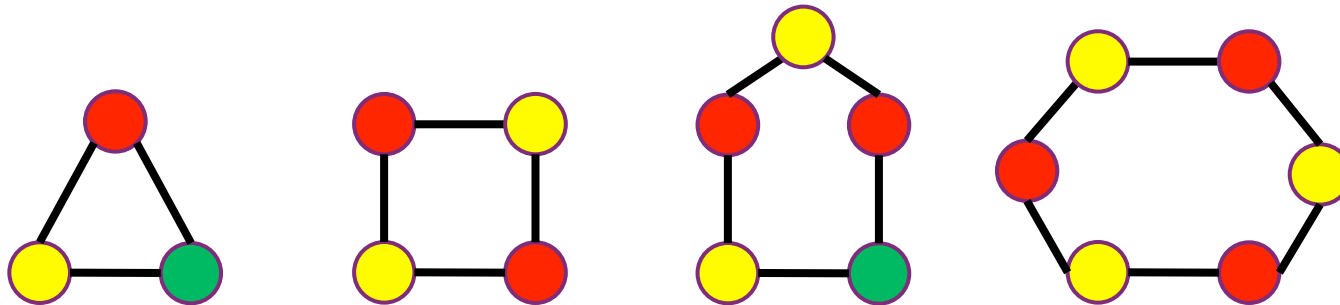
- TEOREMA: Um grafo consistindo simplesmente de um circuito com $n \geq 3$ vértices é 2-cromático se n é par e 3-cromático se n é ímpar



Coloração de circuitos

10

- TEOREMA: Um grafo consistindo simplesmente de um circuito com $n \geq 3$ vértices é 2-cromático se n é par e 3-cromático se n é ímpar



Coloração de circuitos

11

- TEOREMA: Um grafo simples G com pelo menos uma aresta é 2-cromático se, e somente se, G não contiver circuitos de tamanho ímpar

Graus e coloração

12

- Se d é o maior grau dos vértices de um grafo simples G , então $X(G) \leq d+1$
- Se d é o maior grau dos vértices de um grafo simples G , tal que G não contém um grafo circuito com um número ímpar de vértices e nem um grafo completo, de $d+1$ vértices, então $X(G) \leq d$

Coloração de grafos

13

- Infelizmente, o problema de determinar a coloração de um grafo é *NP-Completo*
- Na prática, tenta-se resolver o problema com a utilização de *heurísticas*

Algoritmo guloso (coloração sequencial)

14

$i=0$

enquanto houver vértice incolor faça

 seja v um vértice incolor

 se uma cor i não é usada por nenhum vizinho de v

 então atribua cor i a v

 senão faça $i=i+1$ e atribua cor i a v

Algoritmo guloso (coloração sequencial)

$i=0$

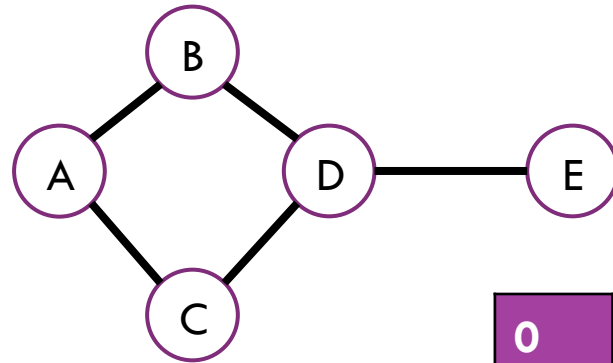
enquanto houver vértice incolor faça

seja v um vértice incolor

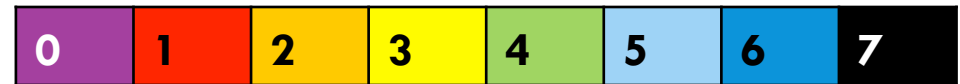
se uma cor i não é usada por nenhum vizinho de v

então atribua cor i a v

senão faça $i=i+1$ e atribua cor i a v



Cores disponíveis



Algoritmo guloso (coloração sequencial)

$i=0$

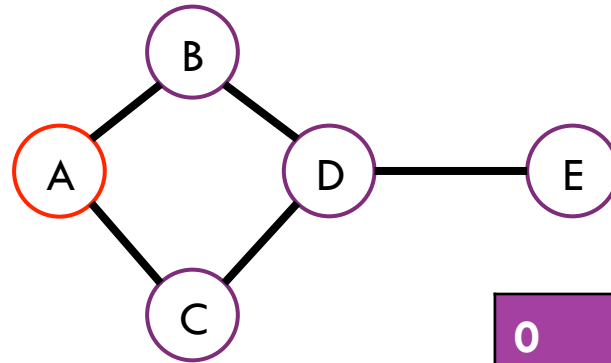
enquanto houver vértice incolor faça

seja v um vértice incolor

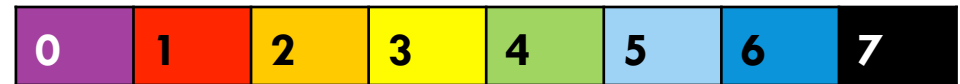
se uma cor i não é usada por nenhum vizinho de v

então atribua cor i a v

senão faça $i=i+1$ e atribua cor i a v



Cores disponíveis



Algoritmo guloso (coloração sequencial)

$i=0$

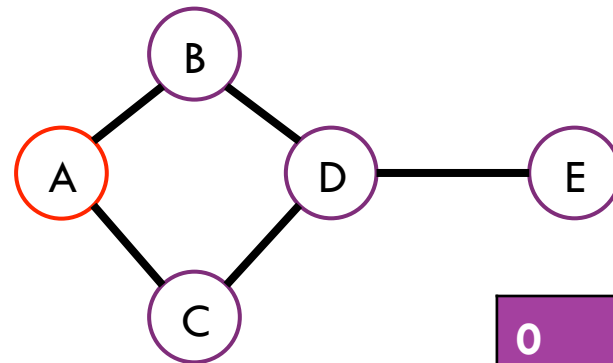
enquanto houver vértice incolor faça

seja v um vértice incolor

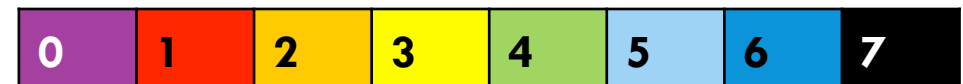
se uma cor i não é usada por nenhum vizinho de v

então atribua cor i a v

senão faça $i=i+1$ e atribua cor i a v



Cores disponíveis



Algoritmo guloso (coloração sequencial)

$i=0$

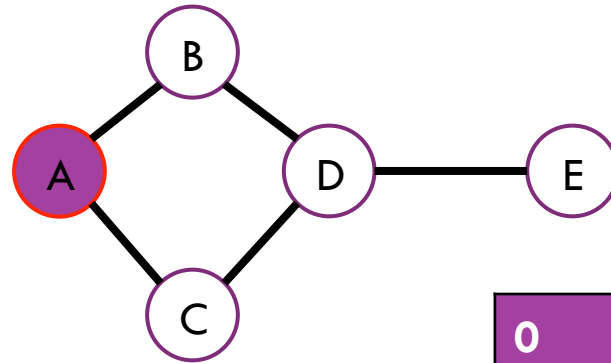
enquanto houver vértice incolor faça

seja v um vértice incolor

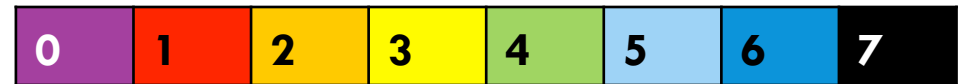
se uma cor i não é usada por nenhum vizinho de v

então atribua cor i a v

senão faça $i=i+1$ e atribua cor i a v



Cores disponíveis



Algoritmo guloso (coloração sequencial)

$i=0$

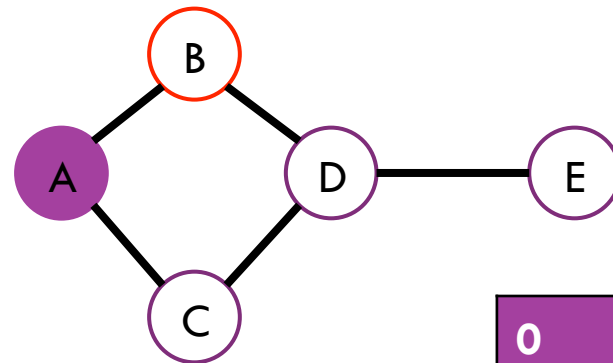
enquanto houver vértice incolor faça

seja v um vértice incolor

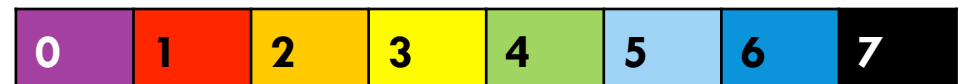
se uma cor i não é usada por nenhum vizinho de v

então atribua cor i a v

senão faça $i=i+1$ e atribua cor i a v



Cores disponíveis



Algoritmo guloso (coloração sequencial)

$i=0$

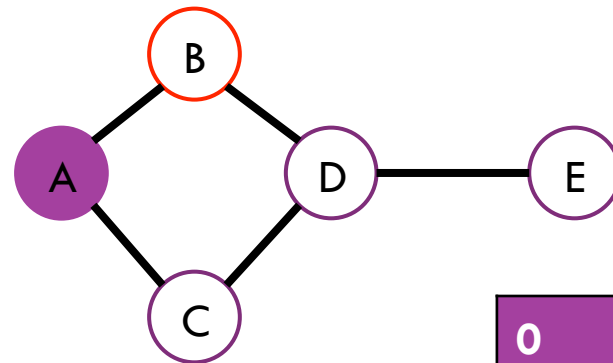
enquanto houver vértice incolor faça

seja v um vértice incolor

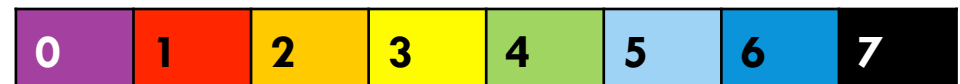
se uma cor i não é usada por nenhum vizinho de v

então atribua cor i a v

senão faça $i=i+1$ e atribua cor i a v



Cores disponíveis



Algoritmo guloso (coloração sequencial)

$i=0$

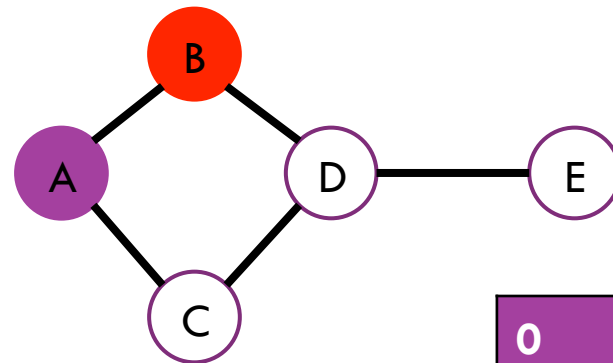
enquanto houver vértice incolor faça

seja v um vértice incolor

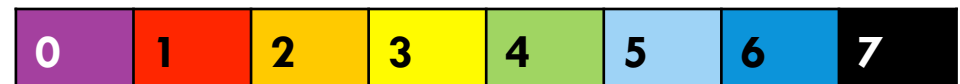
se uma cor i não é usada por nenhum vizinho de v

então atribua cor i a v

senão faça $i=i+1$ e atribua cor i a v



Cores disponíveis



Algoritmo guloso (coloração sequencial)

$i=0$

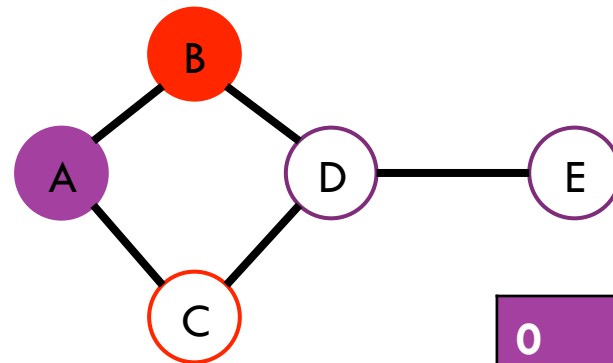
enquanto houver vértice incolor faça

seja v um vértice incolor

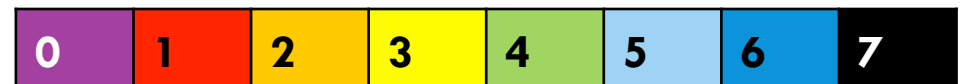
se uma cor i não é usada por nenhum vizinho de v

então atribua cor i a v

senão faça $i=i+1$ e atribua cor i a v



Cores disponíveis



Algoritmo guloso (coloração sequencial)

$i=0$

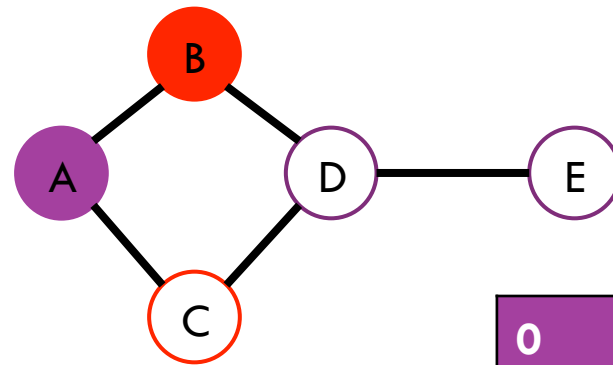
enquanto houver vértice incolor faça

seja v um vértice incolor

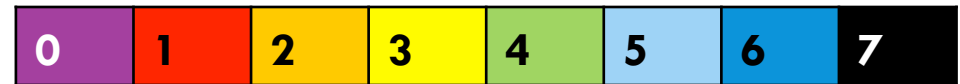
se uma cor i não é usada por nenhum vizinho de v

então atribua cor i a v

senão faça $i=i+1$ e atribua cor i a v



Cores disponíveis



Algoritmo guloso (coloração sequencial)

$i=0$

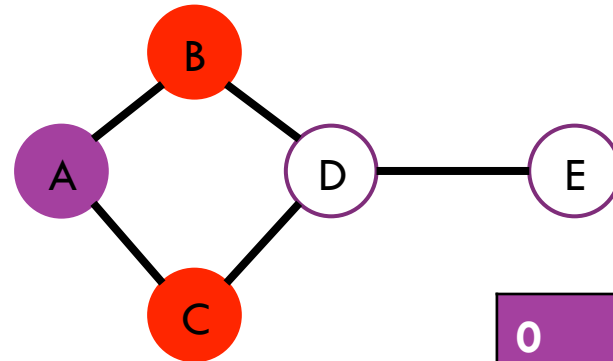
enquanto houver vértice incolor faça

seja v um vértice incolor

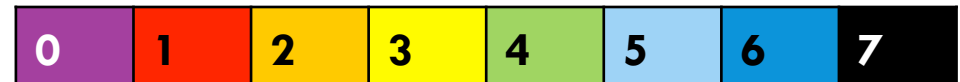
se uma cor i não é usada por nenhum vizinho de v

então atribua cor i a v

senão faça $i=i+1$ e atribua cor i a v



Cores disponíveis



Algoritmo guloso (coloração sequencial)

$i=0$

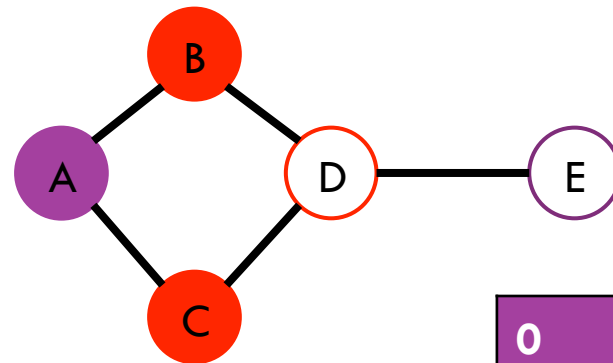
enquanto houver vértice incolor faça

seja v um vértice incolor

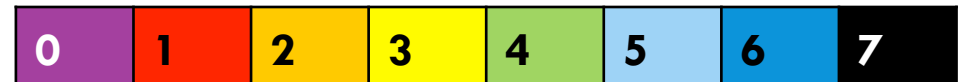
se uma cor i não é usada por nenhum vizinho de v

então atribua cor i a v

senão faça $i=i+1$ e atribua cor i a v



Cores disponíveis



Algoritmo guloso (coloração sequencial)

$i=0$

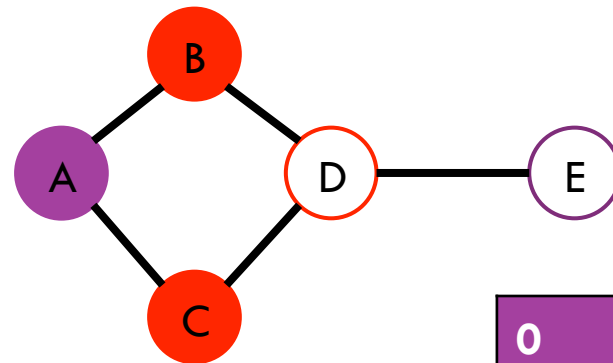
enquanto houver vértice incolor faça

seja v um vértice incolor

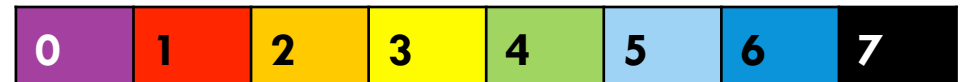
se uma cor i não é usada por nenhum vizinho de v

então atribua cor i a v

senão faça $i=i+1$ e atribua cor i a v



Cores disponíveis



Algoritmo guloso (coloração sequencial)

$i=0$

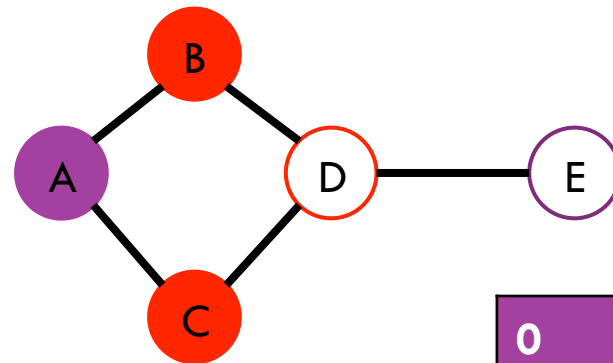
enquanto houver vértice incolor faça

seja v um vértice incolor

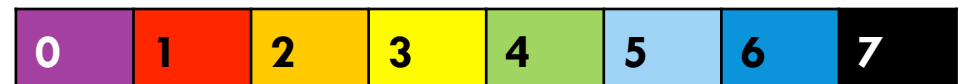
se uma cor i não é usada por nenhum vizinho de v

então atribua cor i a v

senão faça $i=i+1$ e atribua cor i a v



Cores disponíveis



Algoritmo guloso (coloração sequencial)

$i=0$

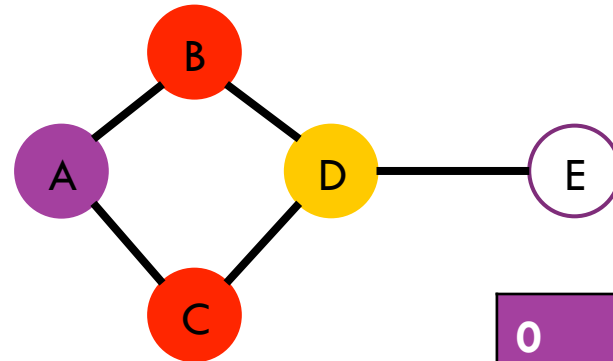
enquanto houver vértice incolor faça

seja v um vértice incolor

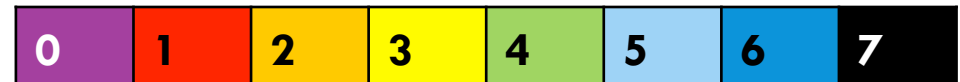
se uma cor i não é usada por nenhum vizinho de v

então atribua cor i a v

senão faça $i=i+1$ e atribua cor i a v



Cores disponíveis



Algoritmo guloso (coloração sequencial)

$i=0$

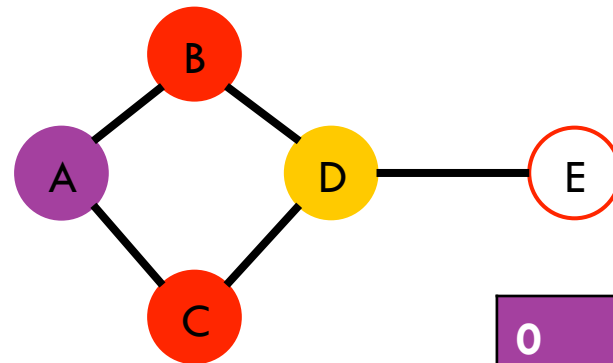
enquanto houver vértice incolor faça

seja v um vértice incolor

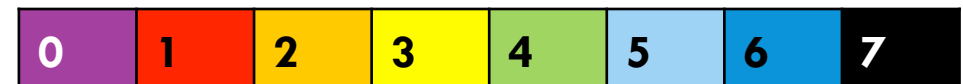
se uma cor i não é usada por nenhum vizinho de v

então atribua cor i a v

senão faça $i=i+1$ e atribua cor i a v



Cores disponíveis



Algoritmo guloso (coloração sequencial)

$i=0$

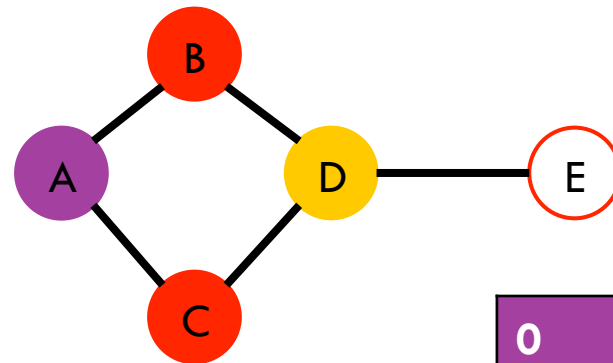
enquanto houver vértice incolor faça

seja v um vértice incolor

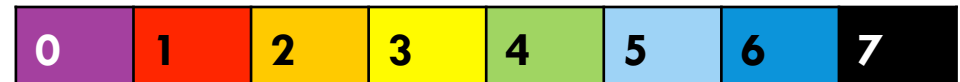
se uma cor i não é usada por nenhum vizinho de v

então atribua cor i a v

senão faça $i=i+1$ e atribua cor i a v



Cores disponíveis



Algoritmo guloso (coloração sequencial)

$i=0$

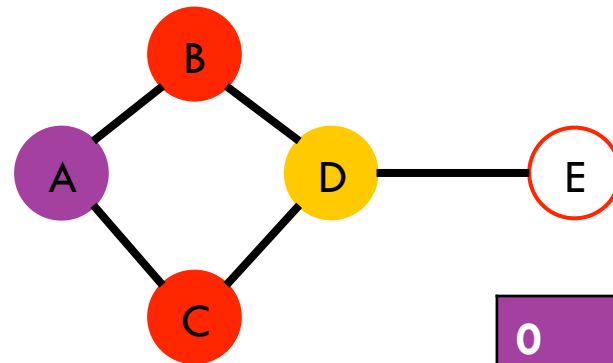
enquanto houver vértice incolor faça

seja v um vértice incolor

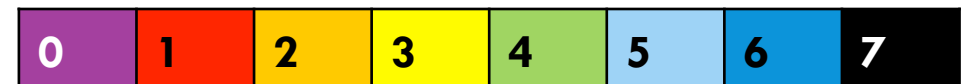
se uma cor i não é usada por nenhum vizinho de v

então atribua cor i a v

senão faça $i=i+1$ e atribua cor i a v



Cores disponíveis



Algoritmo guloso (coloração sequencial)

$i=0$

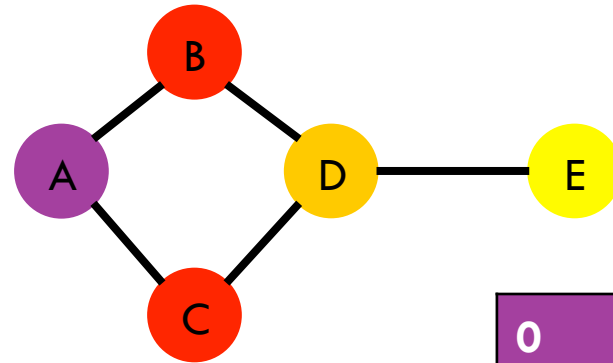
enquanto houver vértice incolor faça

seja v um vértice incolor

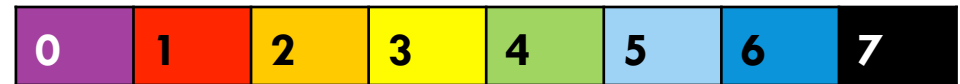
se uma cor i não é usada por nenhum vizinho de v

então atribua cor i a v

senão faça $i=i+1$ e atribua cor i a v



Cores disponíveis



Algoritmo guloso (coloração sequencial)

$i=0$

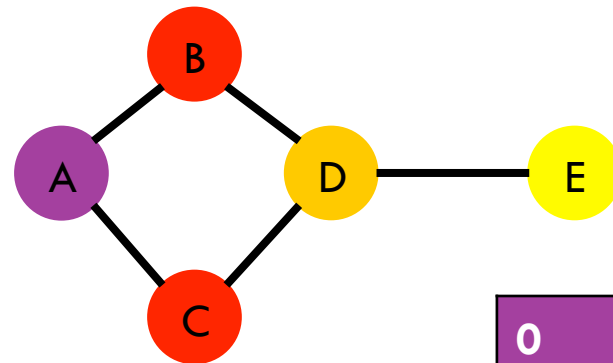
enquanto houver vértice incolor faça

seja v um vértice incolor

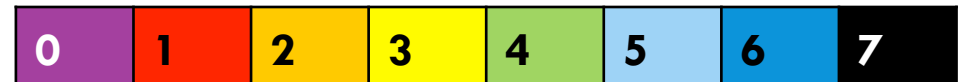
se uma cor i não é usada por nenhum vizinho de v

então atribua cor i a v

senão faça $i=i+1$ e atribua cor i a v



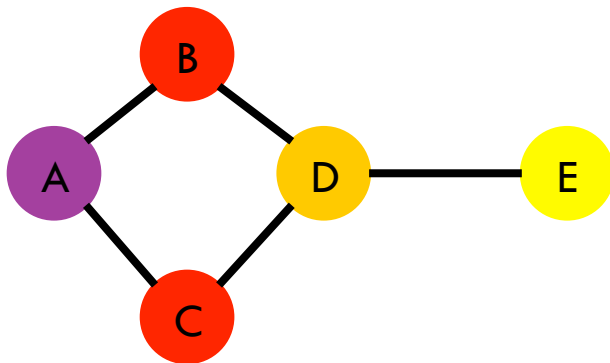
Cores disponíveis



Algoritmo guloso

34

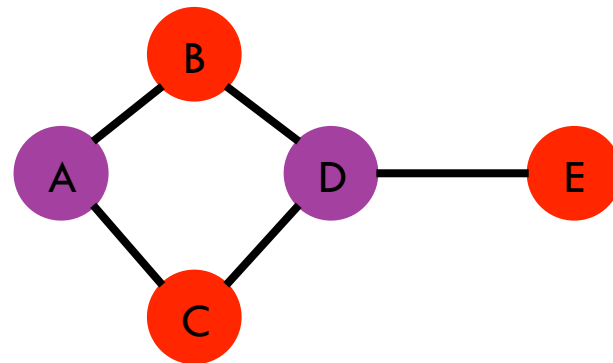
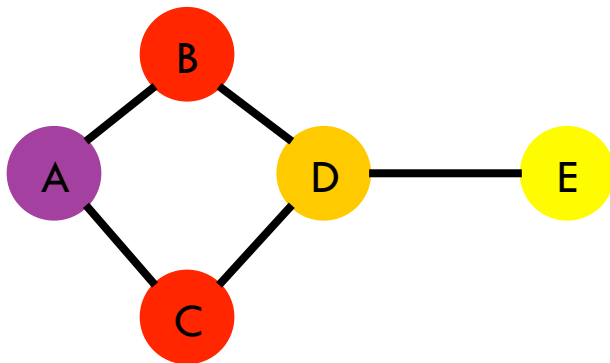
- Obtém a melhor solução?



Algoritmo guloso

35

□ Obtém a melhor solução?



Algoritmo ingênuo (*naive*)

36

$i=0$

atribuir cor i ao primeiro vértice.

percorrer sequencialmente os vértices restantes

para cada vértice v visitado

k = primeira cor já utilizada e que não pertence a nenhum dos vértices adjacentes a v

atribuir cor k a v

se os vértices adjacentes já usam todas as cores,

então $i = i+1$

atribuir cor i ao vértice v

Algoritmo ingênuo (*naive*)

$i=0$

atribuir cor i ao primeiro vértice.

percorrer sequencialmente os vértices restantes

para cada vértice v visitado

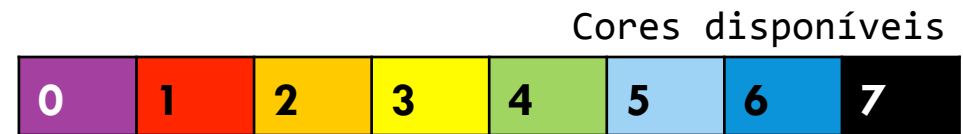
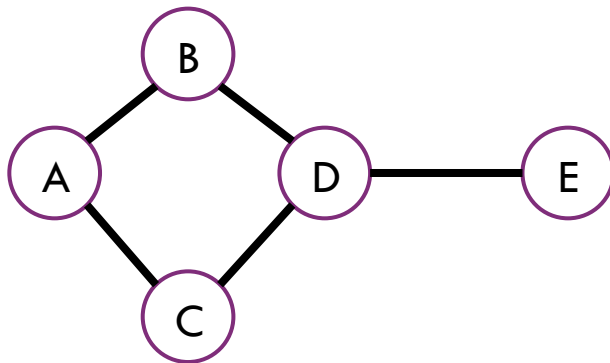
k = primeira cor já utilizada e que não pertence a nenhum dos vértices adjacentes a v

atribuir cor k a v

se os vértices adjacentes já usam todas as cores,

então $i = i+1$

atribuir cor i ao vértice v



Algoritmo ingênuo (*naive*)

$i=0$

atribuir cor i ao primeiro vértice.

percorrer sequencialmente os vértices restantes

para cada vértice v visitado

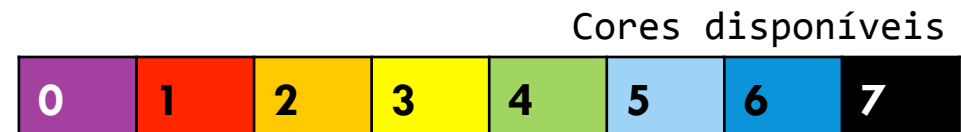
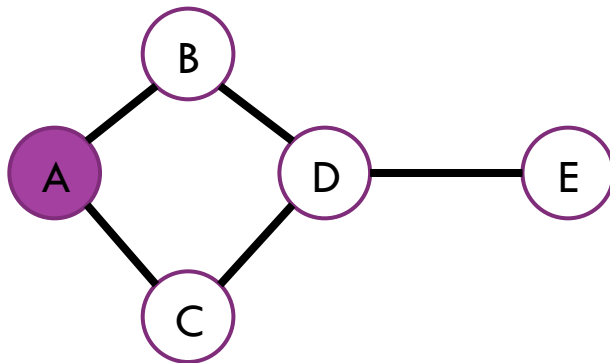
k = primeira cor já utilizada e que não pertence a nenhum dos vértices adjacentes a v

atribuir cor k a v

se os vértices adjacentes já usam todas as cores,

então $i = i+1$

atribuir cor i ao vértice v



Algoritmo ingênuo (*naive*)

$i=0$

atribuir cor i ao primeiro vértice.

percorrer sequencialmente os vértices restantes

para cada vértice v visitado

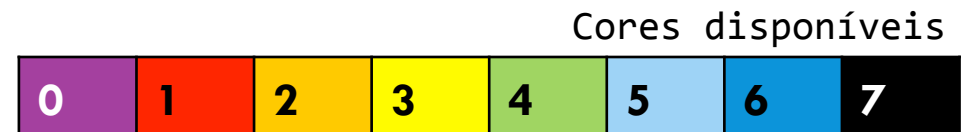
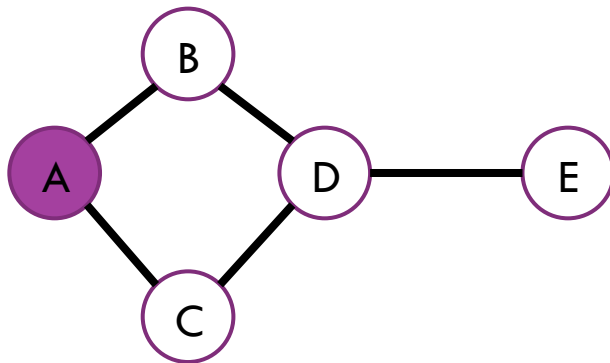
k = primeira cor já utilizada e que não pertence a nenhum dos vértices adjacentes a v

atribuir cor k a v

se os vértices adjacentes já usam todas as cores,

então $i = i+1$

atribuir cor i ao vértice v



Algoritmo ingênuo (*naive*)

$i=0$

atribuir cor i ao primeiro vértice.

percorrer sequencialmente os vértices restantes

para cada vértice v visitado

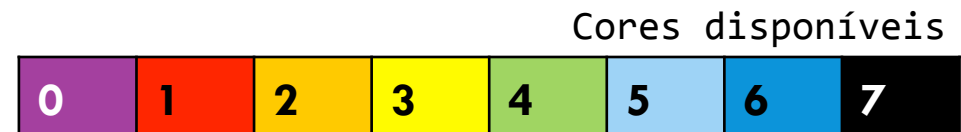
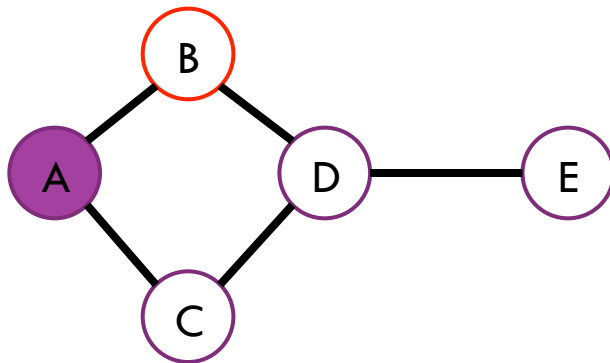
k = primeira cor já utilizada e que não pertence a nenhum dos vértices adjacentes a v

atribuir cor k a v

se os vértices adjacentes já usam todas as cores,

então $i = i+1$

atribuir cor i ao vértice v



Algoritmo ingênuo (*naive*)

$i=0$

atribuir cor i ao primeiro vértice.

percorrer sequencialmente os vértices restantes

para cada vértice v visitado

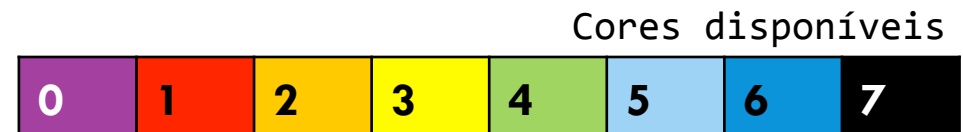
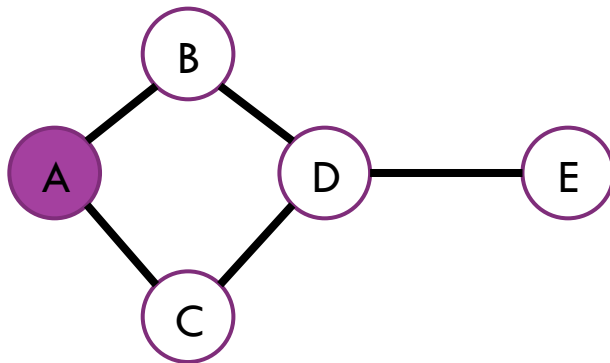
k = primeira cor já utilizada e que não pertence a nenhum dos vértices adjacentes a v

atribuir cor k a v

se os vértices adjacentes já usam todas as cores,

então $i = i+1$

atribuir cor i ao vértice v



Algoritmo ingênuo (*naive*)

$i=0$

atribuir cor i ao primeiro vértice.

percorrer sequencialmente os vértices restantes

para cada vértice v visitado

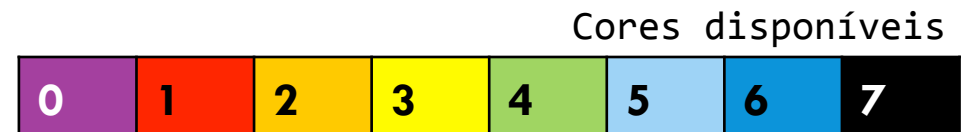
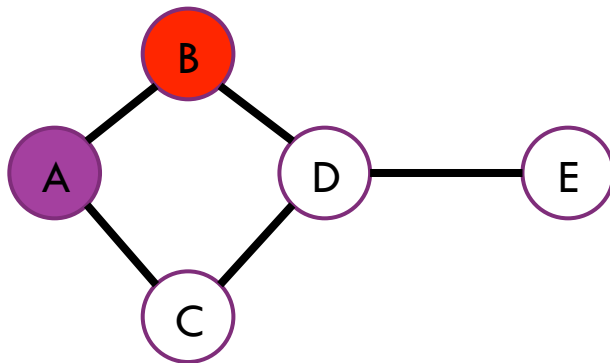
k = primeira cor já utilizada e que não pertence a nenhum dos vértices adjacentes a v

atribuir cor k a v

se os vértices adjacentes já usam todas as cores,

então $i = i+1$

atribuir cor i ao vértice v



Algoritmo ingênuo (*naive*)

$i=0$

atribuir cor i ao primeiro vértice.

percorrer sequencialmente os vértices restantes

para cada vértice v visitado

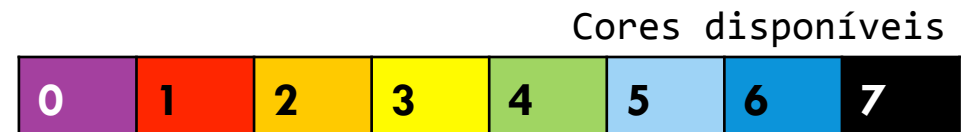
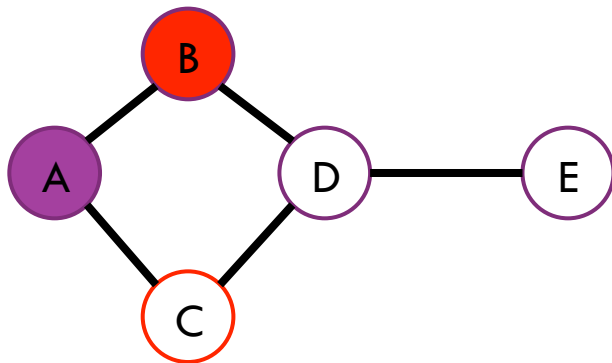
k = primeira cor já utilizada e que não pertence a nenhum dos vértices adjacentes a v

atribuir cor k a v

se os vértices adjacentes já usam todas as cores,

então $i = i+1$

atribuir cor i ao vértice v



Algoritmo ingênuo (*naive*)

$i=0$

atribuir cor i ao primeiro vértice.

percorrer sequencialmente os vértices restantes

para cada vértice v visitado

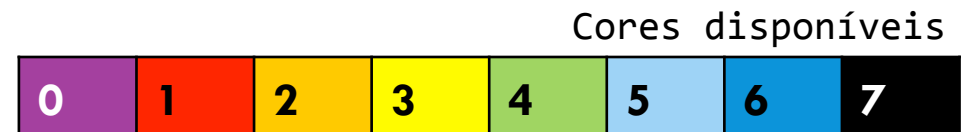
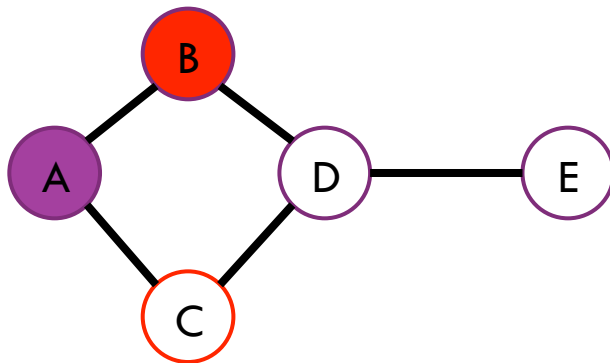
k = primeira cor já utilizada e que não pertence a nenhum dos vértices adjacentes a v

atribuir cor k a v

se os vértices adjacentes já usam todas as cores,

então $i = i+1$

atribuir cor i ao vértice v



Algoritmo ingênuo (*naive*)

$i=0$

atribuir cor i ao primeiro vértice.

percorrer sequencialmente os vértices restantes

para cada vértice v visitado

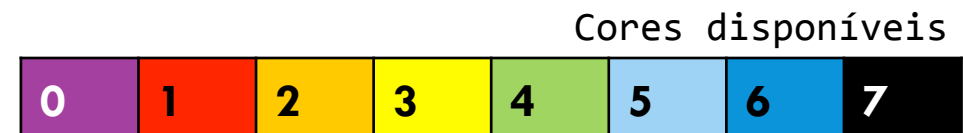
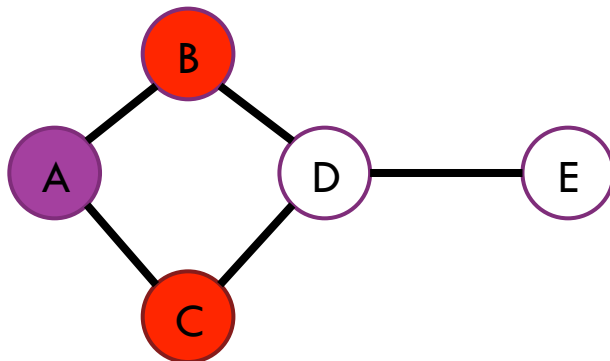
k = primeira cor já utilizada e que não pertence a nenhum dos vértices adjacentes a v

atribuir cor k a v

se os vértices adjacentes já usam todas as cores,

então $i = i+1$

atribuir cor i ao vértice v



Algoritmo ingênuo (*naive*)

$i=0$

atribuir cor i ao primeiro vértice.

percorrer sequencialmente os vértices restantes

para cada vértice v visitado

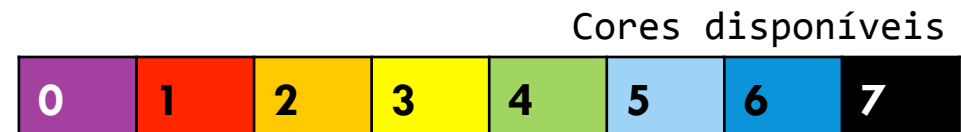
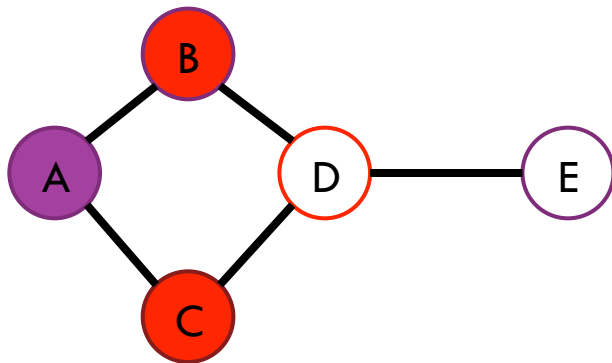
k = primeira cor já utilizada e que não pertence a nenhum dos vértices adjacentes a v

atribuir cor k a v

se os vértices adjacentes já usam todas as cores,

então $i = i+1$

atribuir cor i ao vértice v



Algoritmo ingênuo (*naive*)

$i=0$

atribuir cor i ao primeiro vértice.

percorrer sequencialmente os vértices restantes

para cada vértice v visitado

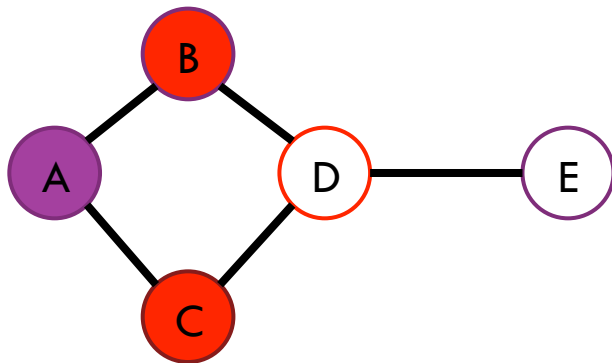
k = primeira cor já utilizada e que não pertence a nenhum dos vértices adjacentes a v

atribuir cor k a v

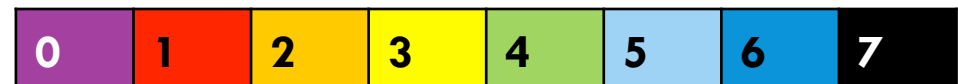
se os vértices adjacentes já usam todas as cores,

então $i = i+1$

atribuir cor i ao vértice v



Cores disponíveis



Algoritmo ingênuo (*naive*)

$i=0$

atribuir cor i ao primeiro vértice.

percorrer sequencialmente os vértices restantes

para cada vértice v visitado

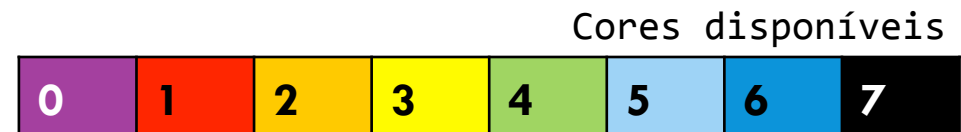
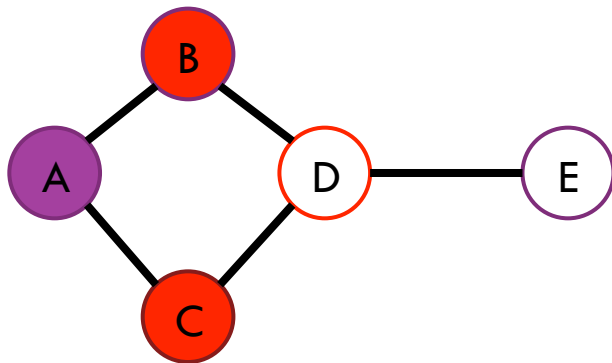
k = primeira cor já utilizada e que não pertence a nenhum dos vértices adjacentes a v

atribuir cor k a v

se os vértices adjacentes já usam todas as cores,

então $i = i+1$

atribuir cor i ao vértice v



Algoritmo ingênuo (*naive*)

$i=0$

atribuir cor i ao primeiro vértice.

percorrer sequencialmente os vértices restantes

para cada vértice v visitado

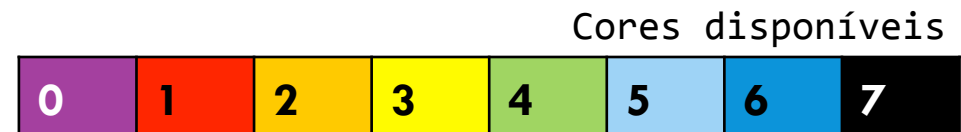
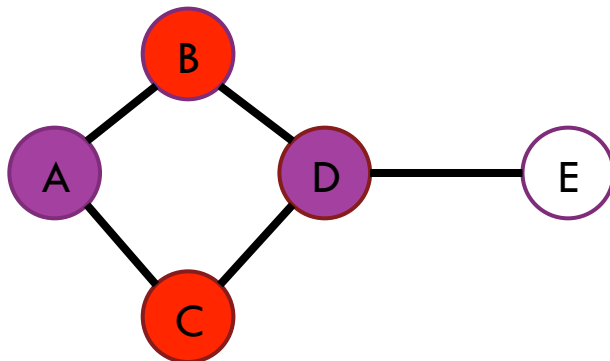
k = primeira cor já utilizada e que não pertence a nenhum dos vértices adjacentes a v

atribuir cor k a v

se os vértices adjacentes já usam todas as cores,

então $i = i+1$

atribuir cor i ao vértice v



Algoritmo ingênuo (*naive*)

$i=0$

atribuir cor i ao primeiro vértice.

percorrer sequencialmente os vértices restantes

para cada vértice v visitado

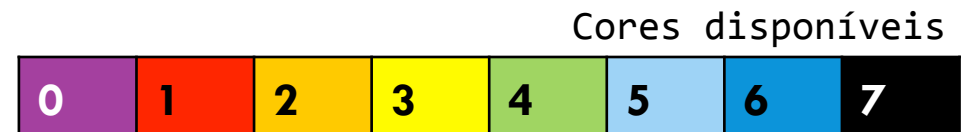
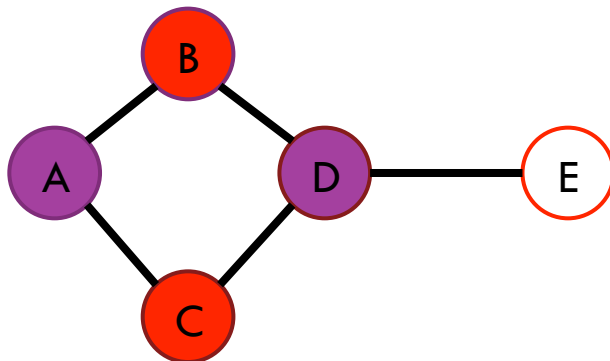
k = primeira cor já utilizada e que não pertence a nenhum dos vértices adjacentes a v

atribuir cor k a v

se os vértices adjacentes já usam todas as cores,

então $i = i+1$

atribuir cor i ao vértice v



Algoritmo ingênuo (*naive*)

$i=0$

atribuir cor i ao primeiro vértice.

percorrer sequencialmente os vértices restantes

para cada vértice v visitado

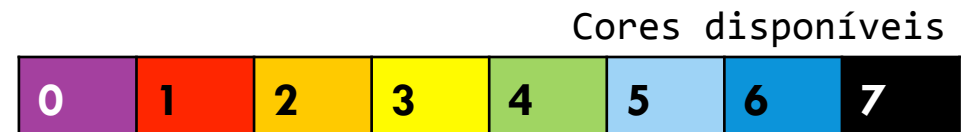
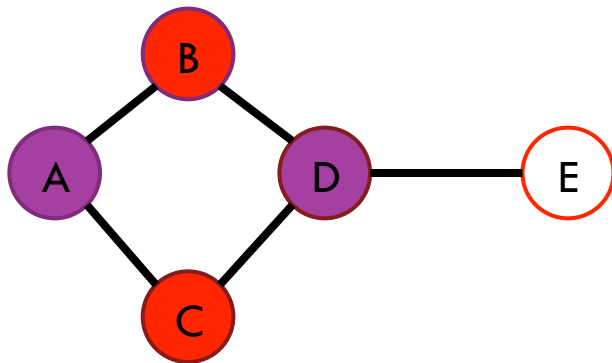
k = primeira cor já utilizada e que não pertence a nenhum dos vértices adjacentes a v

atribuir cor k a v

se os vértices adjacentes já usam todas as cores,

então $i = i+1$

atribuir cor i ao vértice v



Algoritmo ingênuo (*naive*)

$i=0$

atribuir cor i ao primeiro vértice.

percorrer sequencialmente os vértices restantes

para cada vértice v visitado

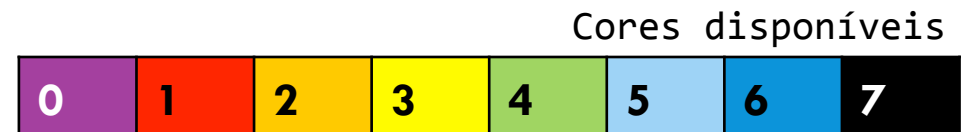
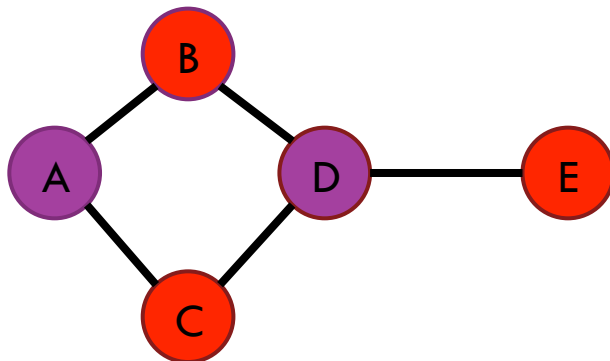
k = primeira cor já utilizada e que não pertence a nenhum dos vértices adjacentes a v

atribuir cor k a v

se os vértices adjacentes já usam todas as cores,

então $i = i+1$

atribuir cor i ao vértice v



Algoritmo ingênuo (*naive*)

$i=0$

atribuir cor i ao primeiro vértice.

percorrer sequencialmente os vértices restantes

para cada vértice v visitado

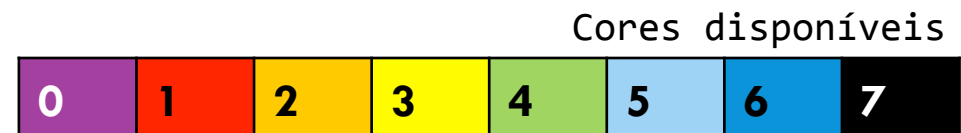
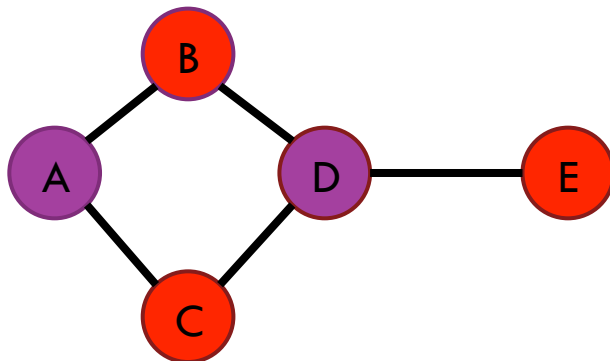
k = primeira cor já utilizada e que não pertence a nenhum dos vértices adjacentes a v

atribuir cor k a v

se os vértices adjacentes já usam todas as cores,

então $i = i+1$

atribuir cor i ao vértice v



Algoritmo ingênuo (*naive*)

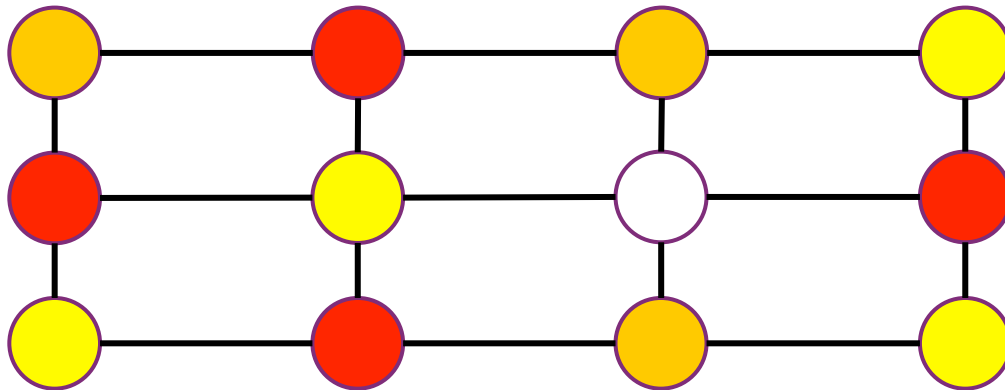
54

- Quanto maior o grau de um vértice, mais difícil sua coloração
- Algoritmo poderia percorrer vértices em ordem decrescente de grau

Coloração de grafos

55

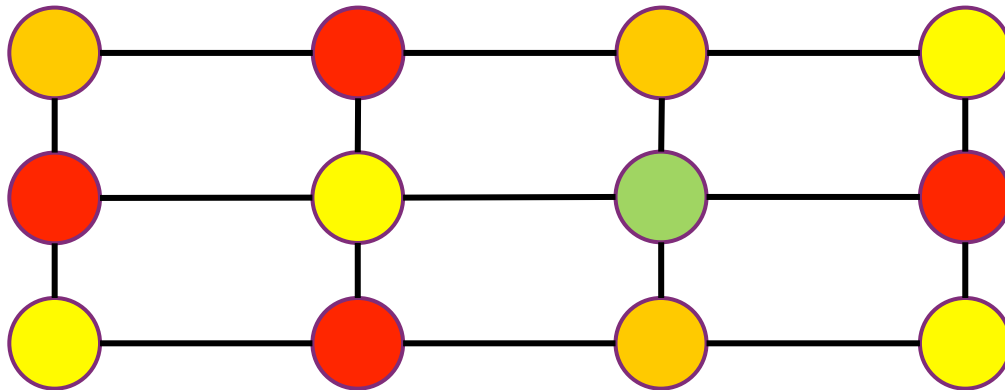
- Muitas outras heurísticas podem ser tentadas, por exemplo, troca de cores em componentes



Coloração de grafos

56

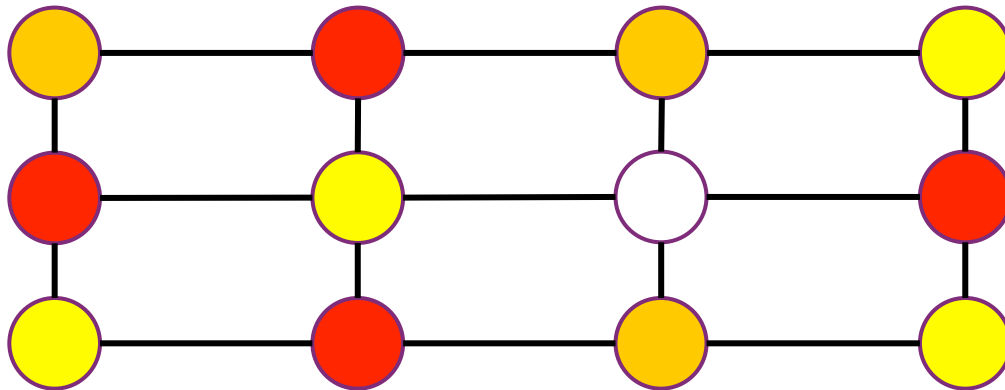
- Muitas outras heurísticas podem ser tentadas, por exemplo, troca de cores em componentes



Coloração de grafos

57

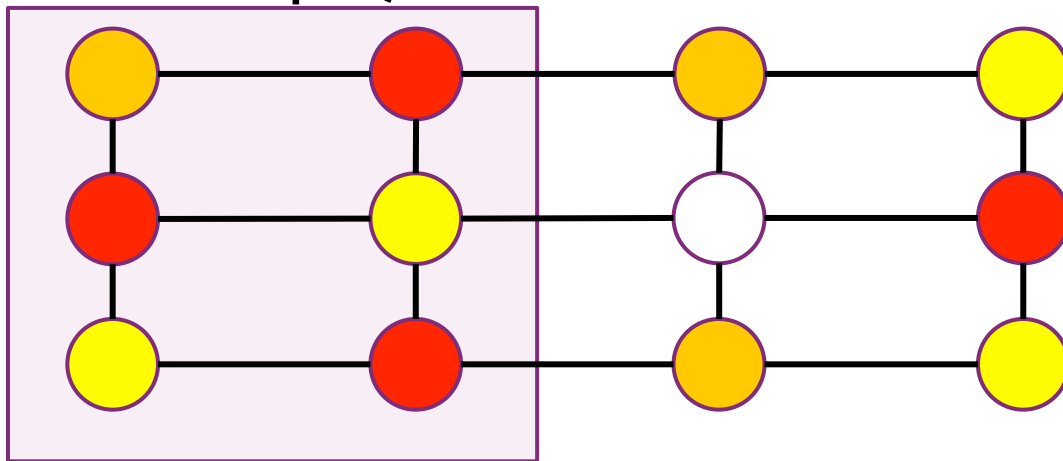
- Muitas outras heurísticas podem ser tentadas, por exemplo, troca de cores em componentes



Coloração de grafos

58

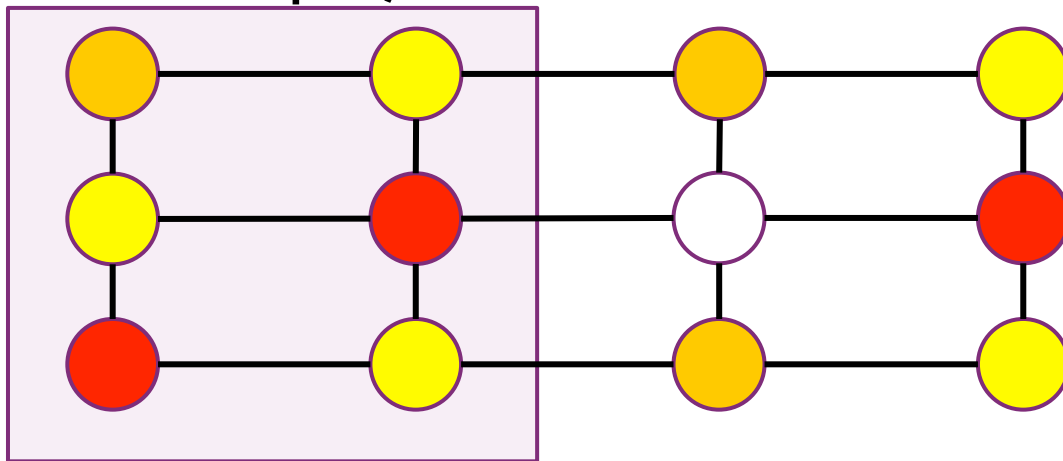
- Muitas outras heurísticas podem ser tentadas, por exemplo, troca de cores em componentes



Coloração de grafos

59

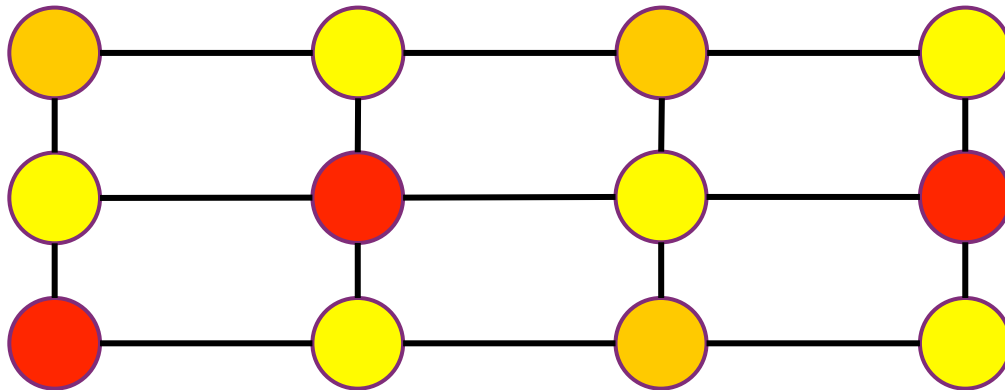
- Muitas outras heurísticas podem ser tentadas, por exemplo, troca de cores em componentes



Coloração de grafos

60

- Muitas outras heurísticas podem ser tentadas, por exemplo, troca de cores em componentes



Coloração de arestas

61

- Uma **coloração de arestas** de um grafo simples G é uma atribuição de cores às arestas de G de maneira que cores diferentes são atribuídas a arestas adjacentes.

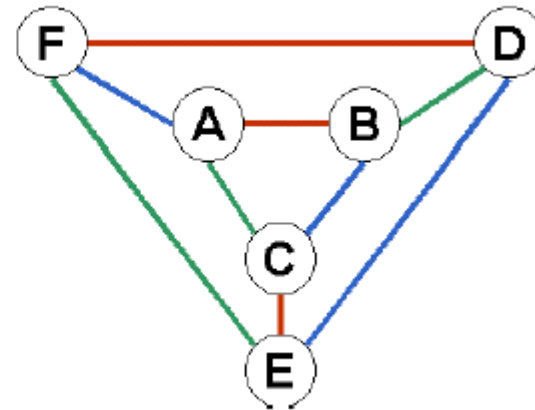
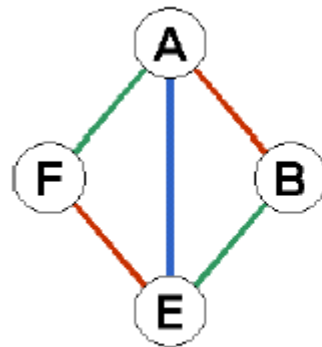
Coloração de arestas

62

- Se existe uma **coloração de arestas** para um grafo G que utiliza K cores, então, G é um grafo K -colorido de arestas.
- O **índice cromático** de um grafo G , $\chi'(G)$, é o menor número K para qual G é K -colorido de arestas.

Coloração de arestas

63



Coloração de arestas

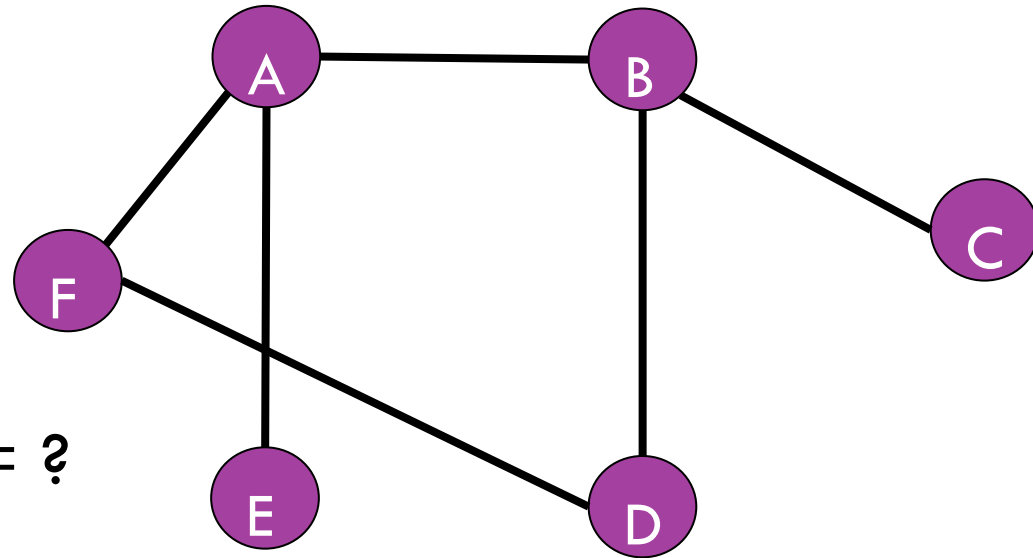
64

- TEOREMA: Se G é um grafo simples cujo vértice de maior grau tem grau $\delta(G)$, então

$$\delta(G) \leq \chi'(G) \leq \delta(G)+1$$

Exercício

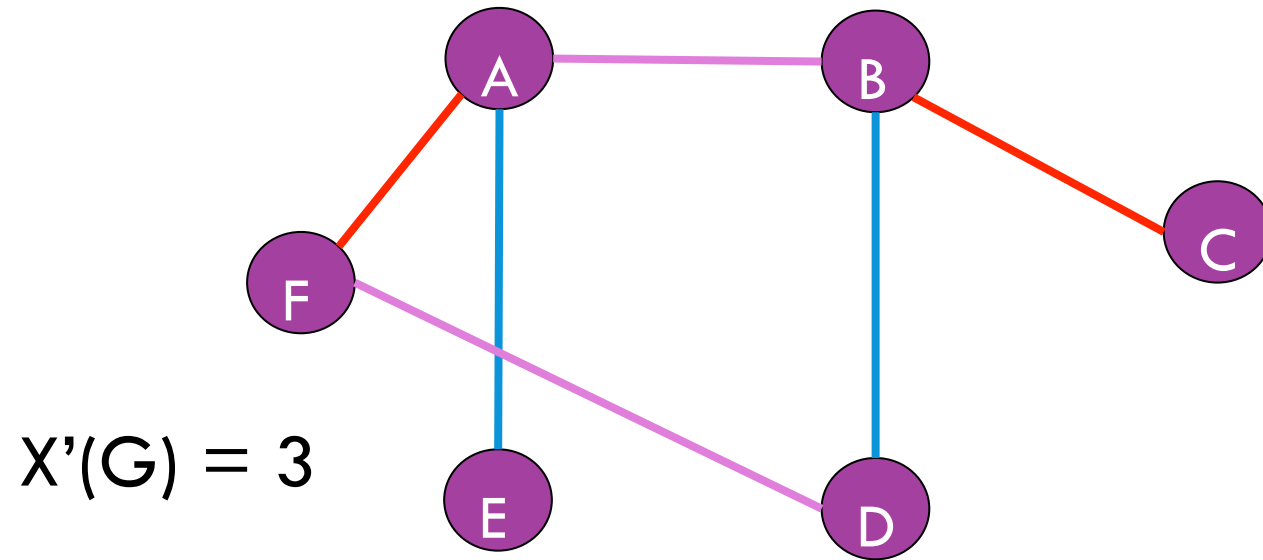
65



$X'(G) = ?$

Exercício

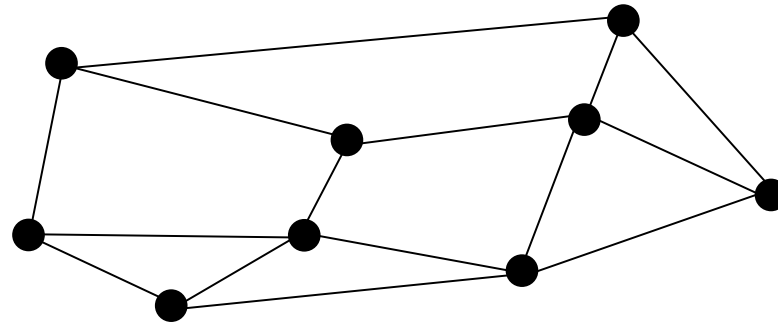
66



Coloração de grafos planares

67

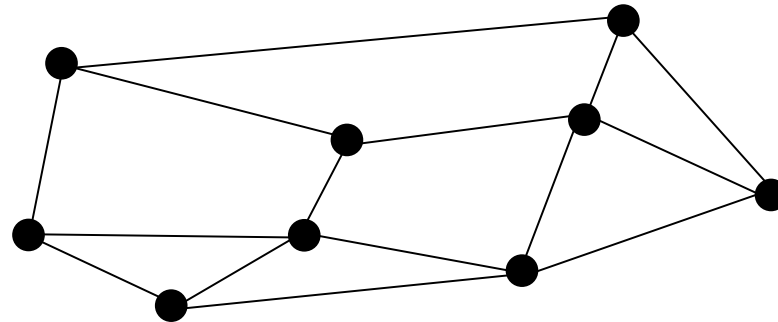
- Quantas cores são necessárias para colorir um grafo planar?



Coloração de grafos planares

68

- Quantas cores são necessárias para colorir um grafo planar?



- Coloração de **faces** do grafo

Dualidade

69

- Dado um grafo G planar, o grafo G^* , chamado **dual** de G , é construído da seguinte forma:
 - para cada face f de G , G^* tem um vértice

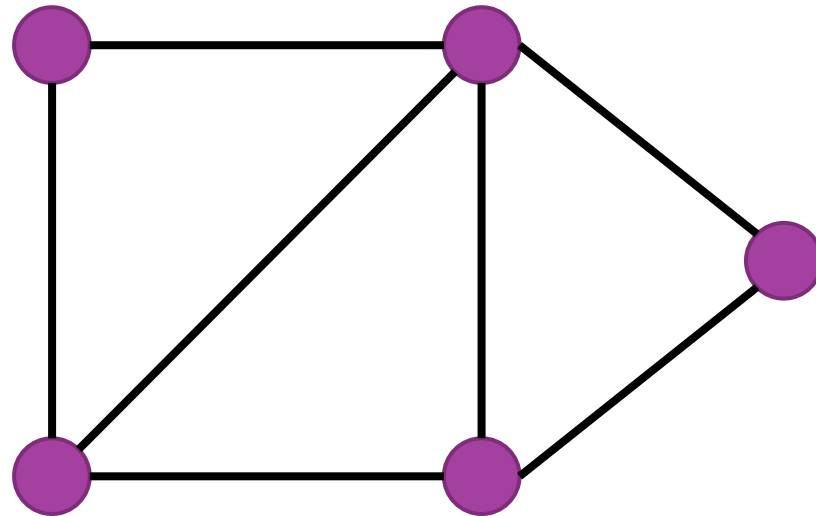
Dualidade

70

- una os vértices de G^* da seguinte forma
 - se 2 regiões f_i e f_k são adjacentes (possuem alguma aresta em comum) coloque uma aresta entre v_i e v_k interceptando a aresta em comum
 - se existirem mais de uma aresta em comum entre f_i e f_k coloque uma aresta entre v_i e v_k para cada aresta em comum
 - se uma aresta está inteiramente em uma região, f_i , coloque um loop no vértice v_i .

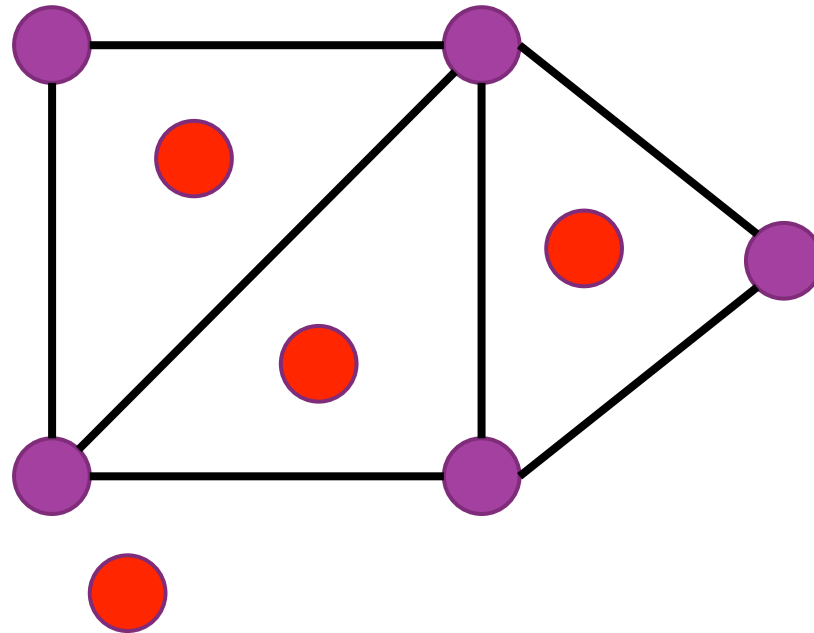
Dualidade

71



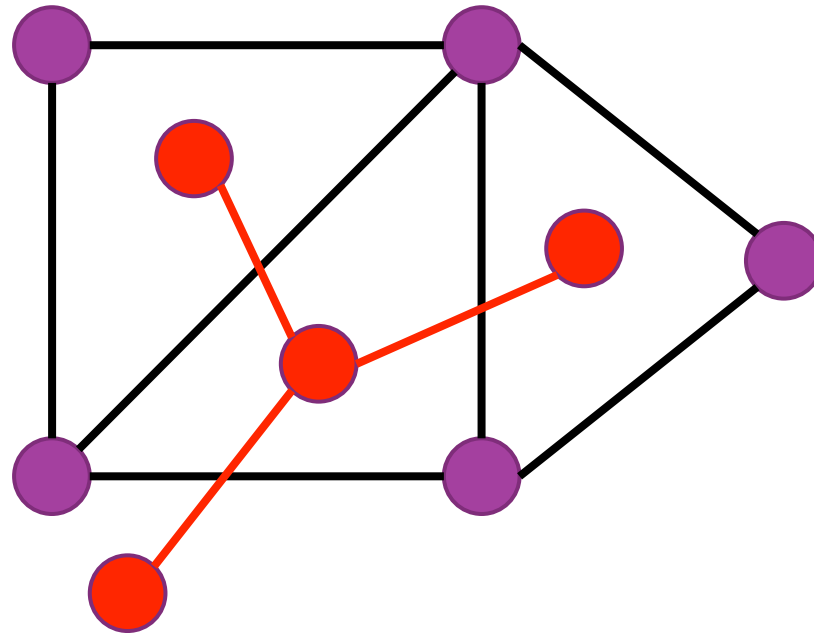
Dualidade

72



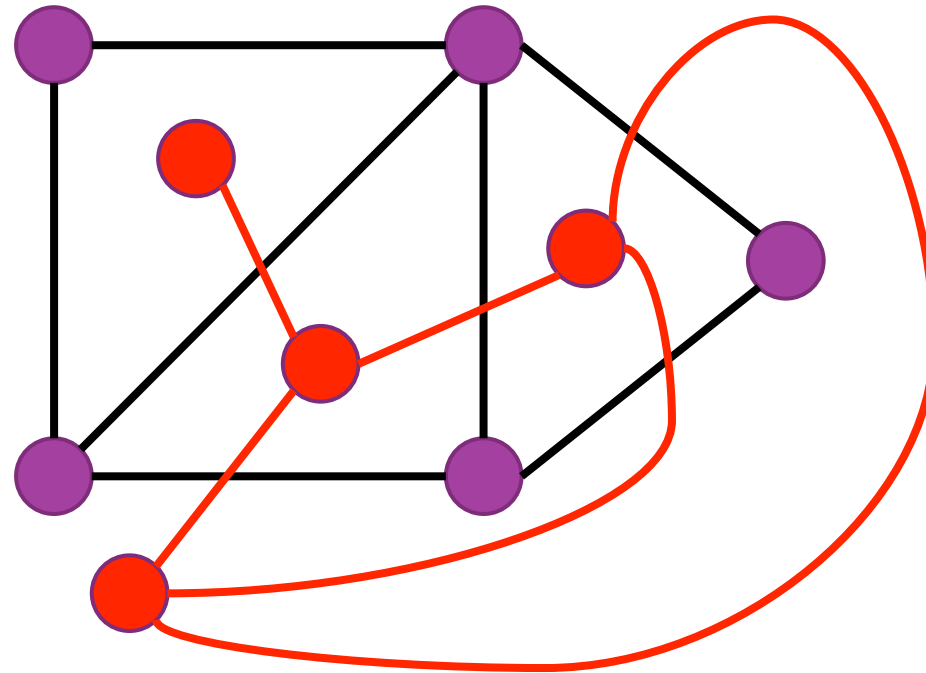
Dualidade

73



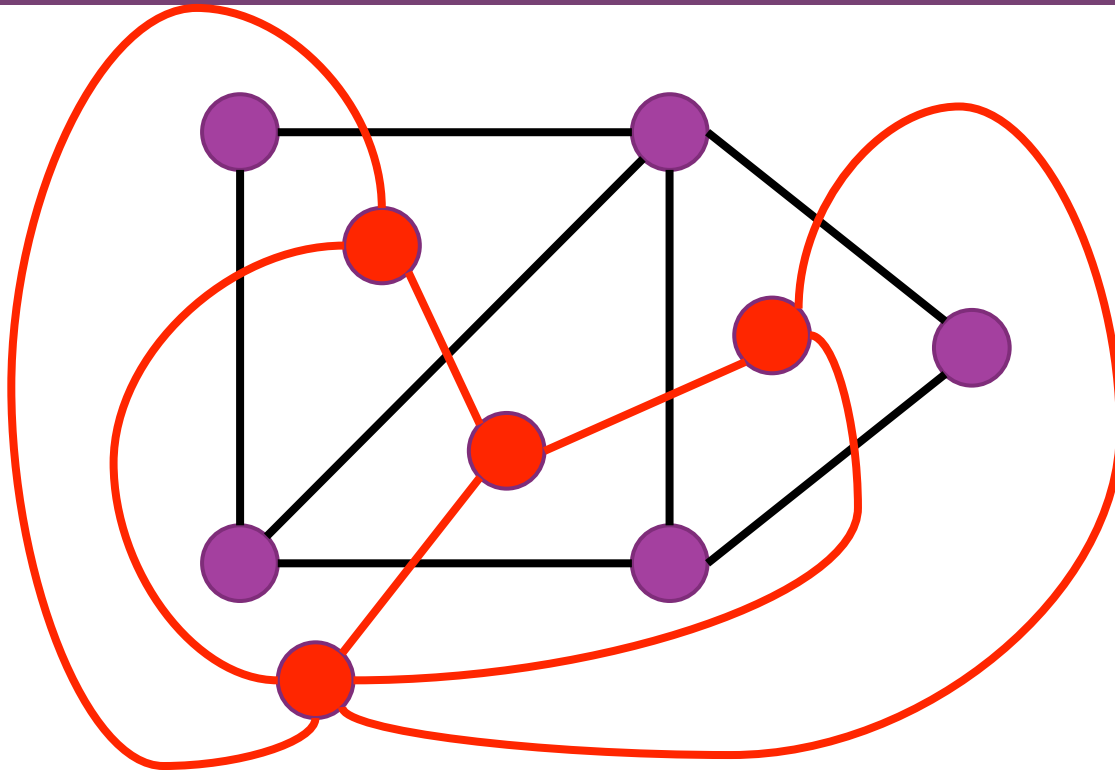
Dualidade

74



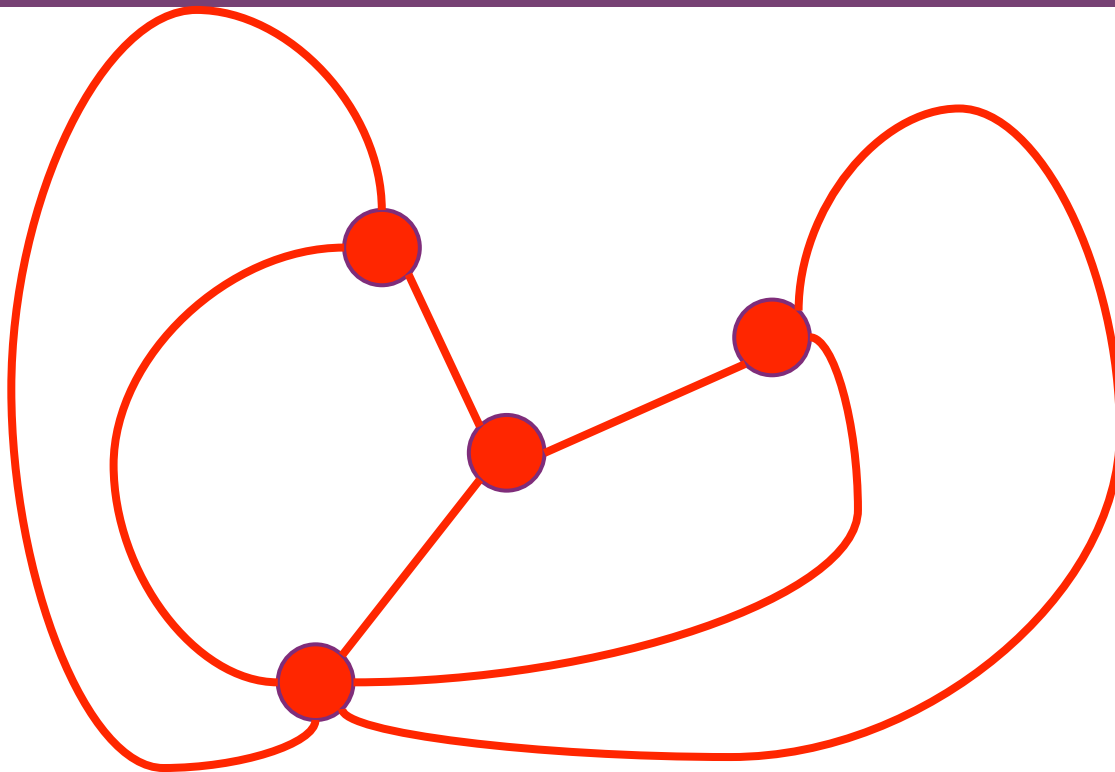
Dualidade

75



Dualidade

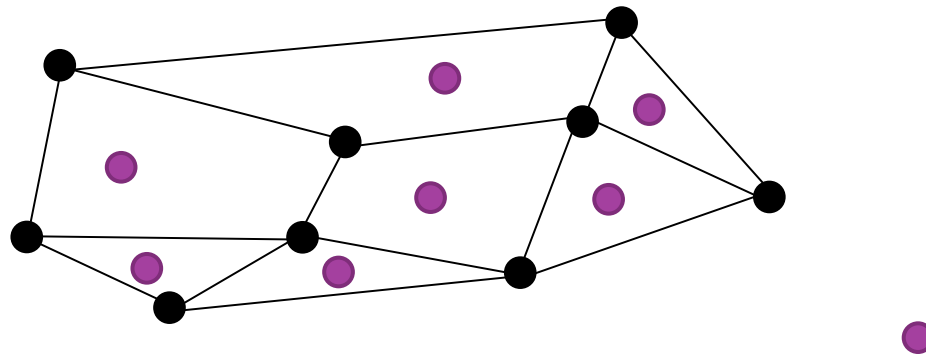
76



Coloração de grafos planares

77

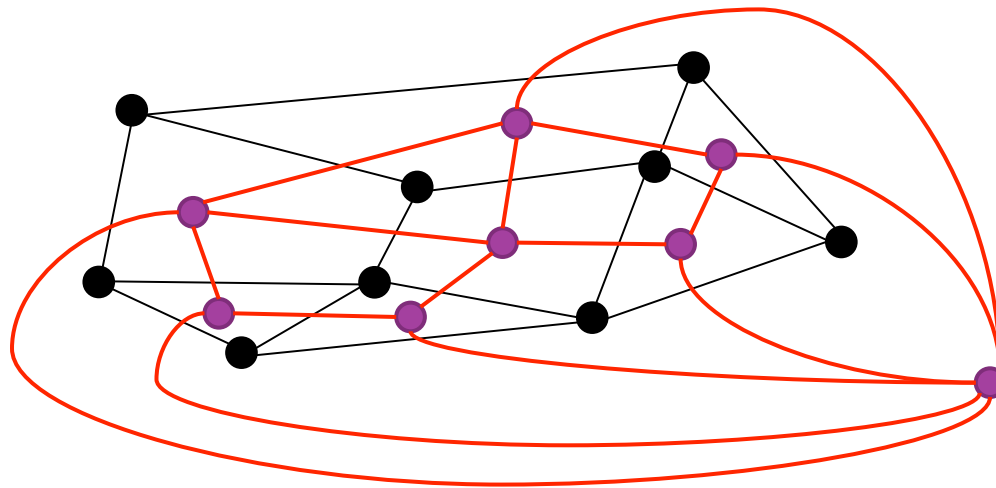
- Considerando o dual de G (G^*), a coloração de faces de G equivale à coloração de vértices de G^* .



Coloração de grafos planares

78

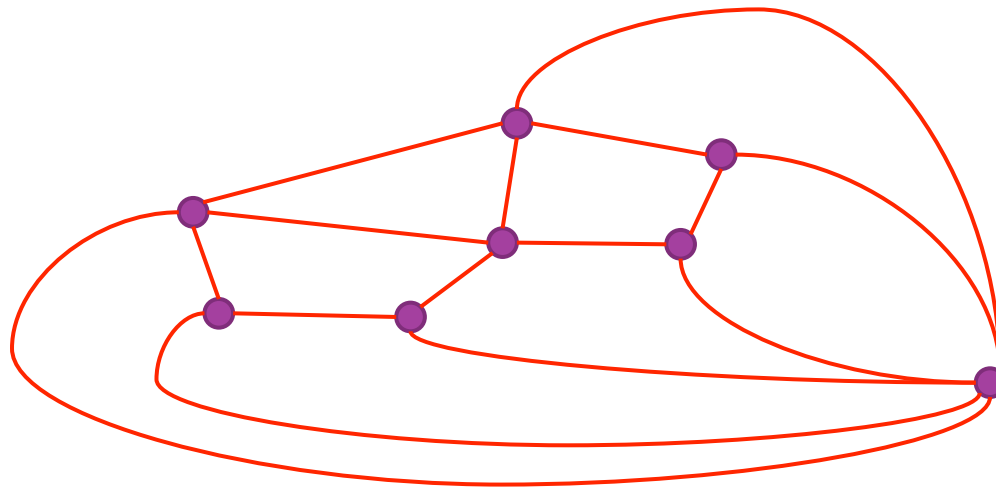
- Considerando o dual de G (G^*), a coloração de faces de G equivale à coloração de vértices de G^* .



Coloração de grafos planares

79

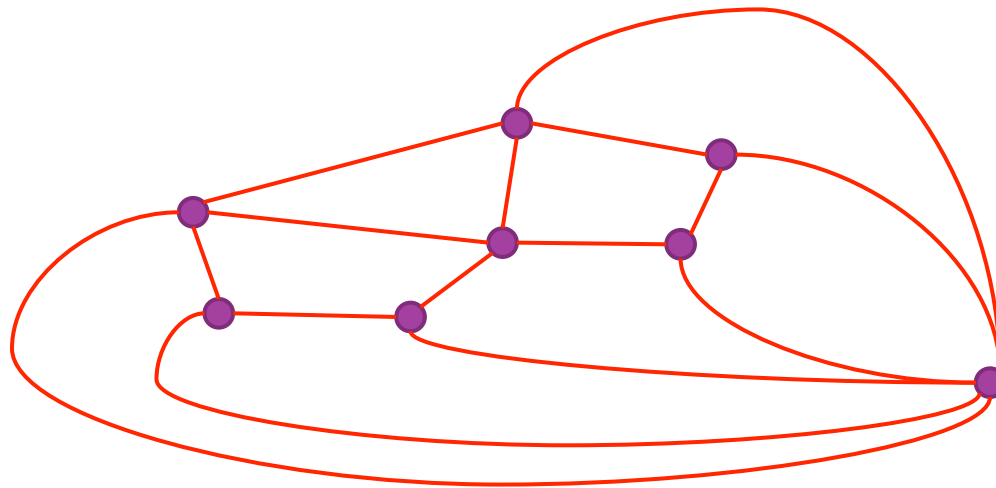
- Considerando o dual de G (G^*), a coloração de faces de G equivale à coloração de vértices de G^* .



Coloração de grafos planares

80

- G é K -cromático de faces se, e somente se, G^* for K -cromático de vértices



Coloração de grafos planares

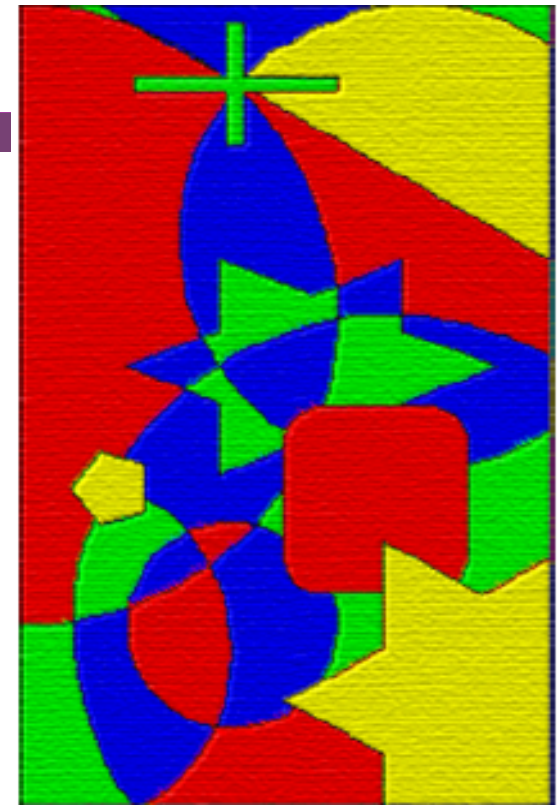
81

- TEOREMA DAS QUATRO CORES: Dado um mapa plano, dividido em regiões, quatro cores são suficientes para colorir, de forma a que regiões vizinhas não partilhem a mesma cor. (Appel e Haken, 1976)

Teorema das quatro cores

82

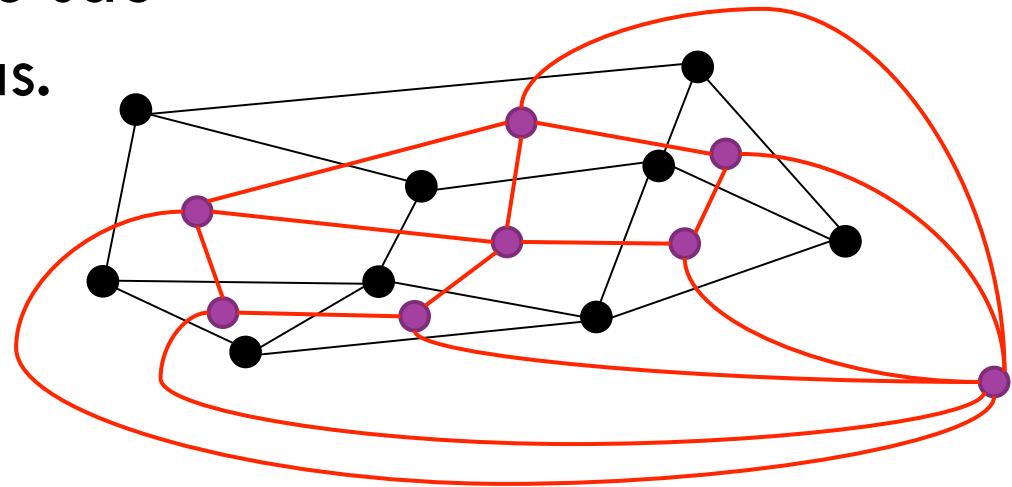
- Condição: as regiões que só se tocam num ponto não são consideradas vizinhas.



Teorema das quatro cores

83

- Condição: as regiões que só se tocam num ponto não são consideradas vizinhas.



Teorema das quatro cores

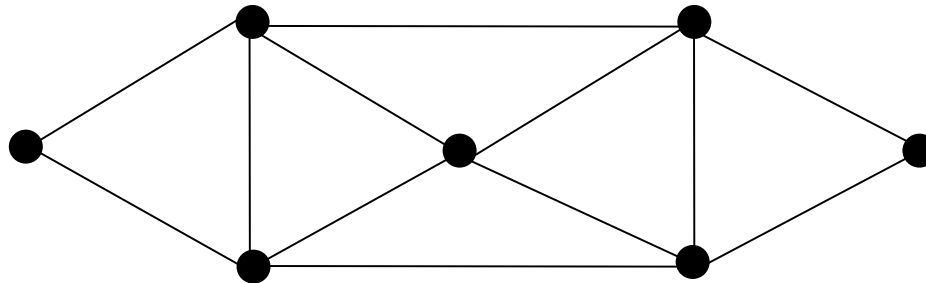
84

- Considerando tal condição, o teorema foi demonstrado pela primeira vez em 1976 por Appel e Haken, utilizando um computador IBM 360.
- Em 1994 foi produzida uma prova simplificada por Raul Seymour, Neil Robertson, Daniel Sanders e Robin Thomas, mas continua a ser impossível demonstrar o teorema sem recorrer a um computador.

Coloração de grafos planares

85

- TEOREMA: Um grafo planar G pode ter as faces coloridas com 2 cores se, e somente se, G for Euleriano.



Coloração de grafos planares

86

- TEOREMA: Um grafo planar G pode ter as faces coloridas com 2 cores se, e somente se, G for Euleriano.

