

Grafos - Aula 12

Roteiro

- Paradigma guloso
- Coloração
- Número cromático
- Algoritmo guloso
- Teorema das 4 cores

Projetando Algoritmos



- Dado um problema P , como projetar um algoritmo que produza a solução?
 - ex. maior clique de um grafo
- Técnicas para *mecanizar* a construção de algoritmos

Problema Central da Computação!

- Ainda faltam princípios e técnicas fundamentais
 - projeto de algoritmos ainda baseado em arte
- **Algumas técnicas:** força bruta, técnica gulosa, programação dinâmica

Força Bruta



- Buscar pela solução no espaço de possíveis soluções para o problema
 - enumerar e testar possíveis soluções, retornar a melhor encontrada
- Ex. encontrar um clique de tamanho k em um grafo
 - enumerar todas combinações com k vértices, verificar se há arestas entre todos os vértices

Problema ?

- Algoritmo tem alta complexidade (espaço de soluções cresce rapidamente)
 - inadequado para problemas grandes
- Geralmente é fácil mecanizar usando força bruta

Algoritmo Guloso

- **Ideia:** algoritmo constrói solução de forma iterativa, tomando decisões ótimas a cada passo para otimizar algum objetivo global
 - cada iteração resolve um problema “pequeno” e “local” de forma ótima

Exemplos vistos em aula!

- Dijkstra: processo iterativo, a cada passo escolhe vértice de menor distância
- A*: processo iterativo, escolhe melhor estimativa de distância
- Kruskal: processo iterativo, a cada passo escolhe aresta de menor peso

Algoritmo Guloso

Vantagens

- Geralmente fácil de construir
- Baixa complexidade
- Heurísticas (intuição) ajudam na optimalidade

Desvantagens

- Geralmente não obtém solução ótima
- Garantir (provar) optimalidade é difícil
- Podem gerar soluções ruins

Técnica fundamental para projetar algoritmos

Colorindo um Mapa

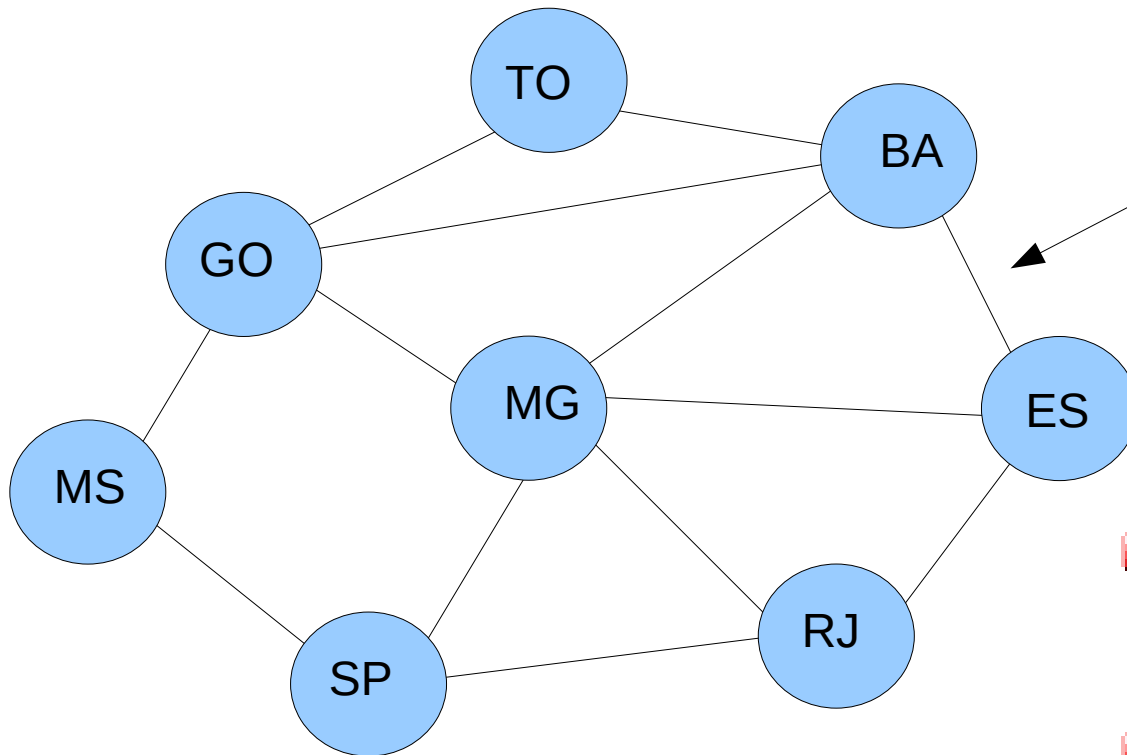


- Mapa de regiões (estados)
- Colorir o mapa
 - regiões vizinhas (com fronteira) **não** podem ter mesma cor

- **Problema 1:** Como colorir um mapa atendendo a restrição?
- **Problema 2:** Qual é o **menor** número de cores necessário?

Colorindo um Mapa

- Abstração via grafos
- Vértices: regiões (estados)
- Arestas: duas regiões são vizinhas

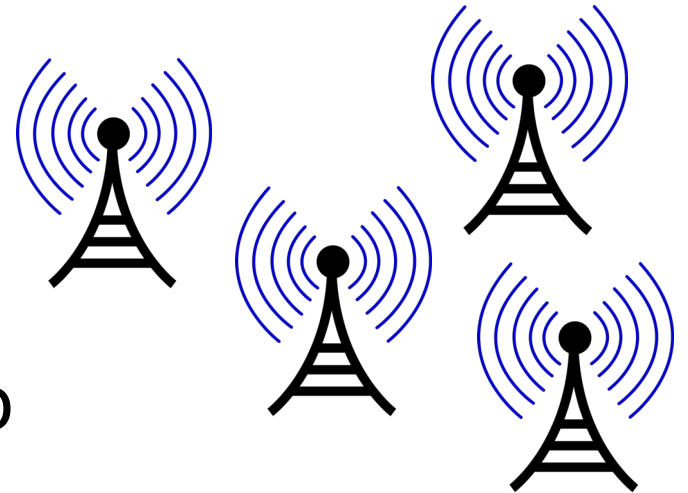


BA e ES são vizinhos

- Vértices vizinhos não podem ter mesma cor
- Número mínimo de cores?

Alocação de Frequências

- Rede telefonia celular
 - estações base (torre)
- Células vizinhas não podem usar mesma frequência de rádio
 - interferência!

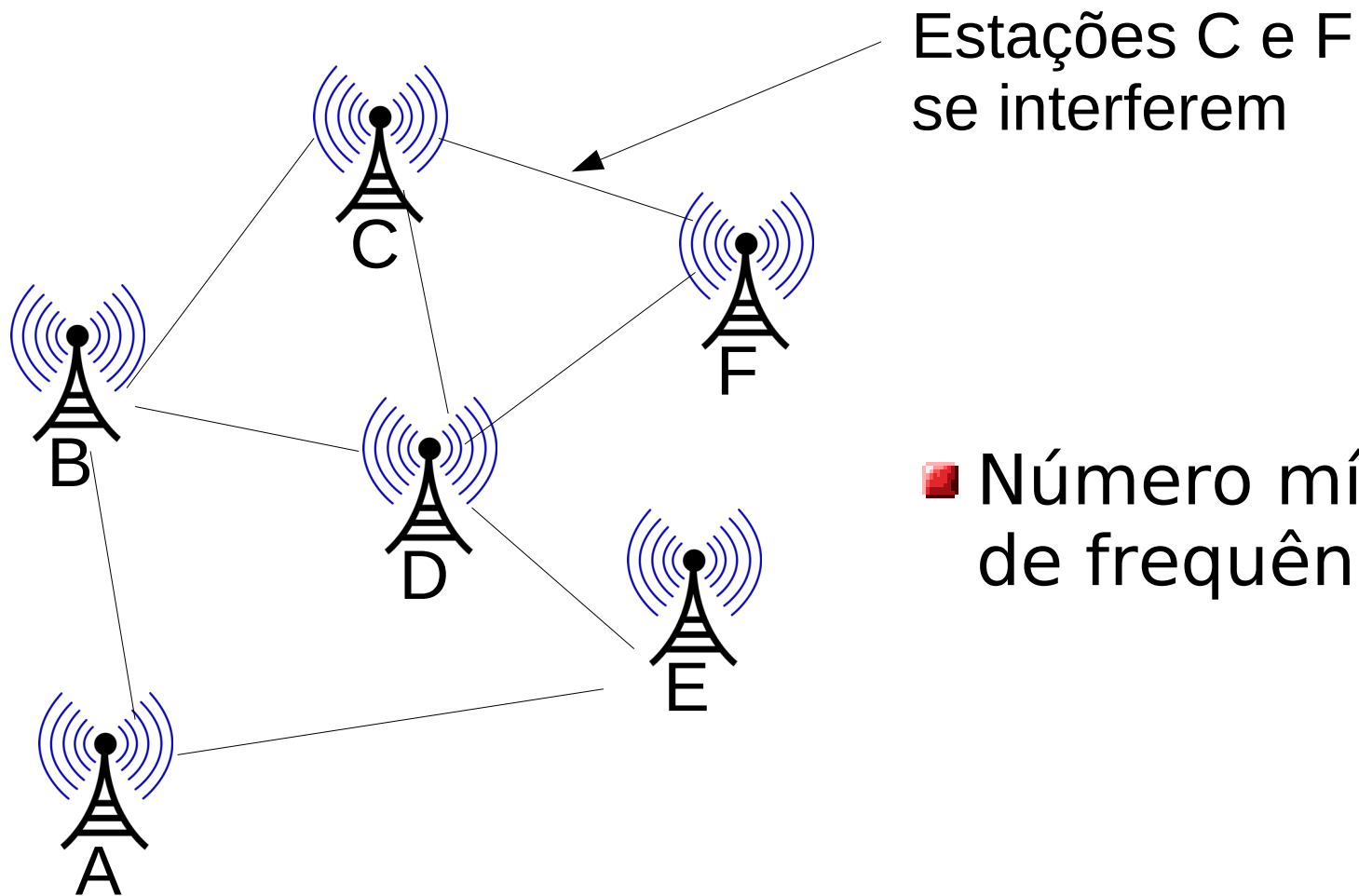


- **Problema 1:** Como alocar frequência às células?
- **Problema 2:** Qual é o menor número de frequências necessário?

Mesma abstração!

Alocação de Frequências

- Vértices: estações base
- Arestas: duas estações são vizinhas (interferem)

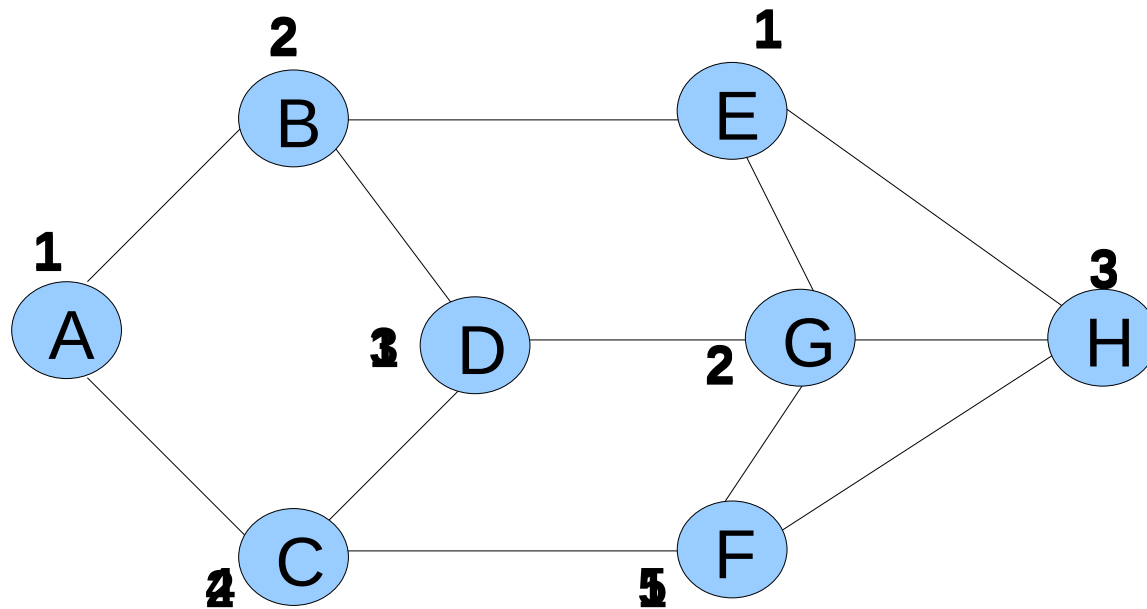


Coloração de Vértices

- Dado grafo $G = (V, E)$, colorir seus vértices
- Restrição: vértices vizinhos não podem ter a mesma cor
- *k-coloração*: coloração que utiliza exatamente k cores
 - dizemos que o grafo é *k-colorível*
 - todo grafo com n vértices é n -colorível
- **Número cromático**: menor número de cores necessário para colorir o grafo
 - grafo completo: número cromático é n

Exemplo

- Uma coloração qualquer?
- Número cromático?



- Coloração qualquer é fácil, coloração com o número cromático é mais difícil

Algoritmo para Coloração

- Algoritmo para colorir um grafo com o menor número de cores possível
- Aplicar a técnica gulosa
 - guloso em que aspecto?
- **Obs:** vértices de maior grau possuem mais restrições
- **Ideia:** guloso no grau do vértice
 - colorir primeiro os mais restritos

Algoritmo Guloso

- Guloso no grau dos vértices
 - em ordem decrescente

1. Colorir(G)
2. Ordenar vertices em ordem decrescente de grau
3. Define *conjunto* $C[i] = \emptyset$ para $i=1, \dots, n$
4. Para $j=1, \dots, n$ faça
5. Selecione r , a menor cor para colorir $v[j]$
 // menor r tal que nenhum vertice em $C[r]$
 // seja vizinho de $v[j]$
6. Incluir $v[j]$ em $C[r]$

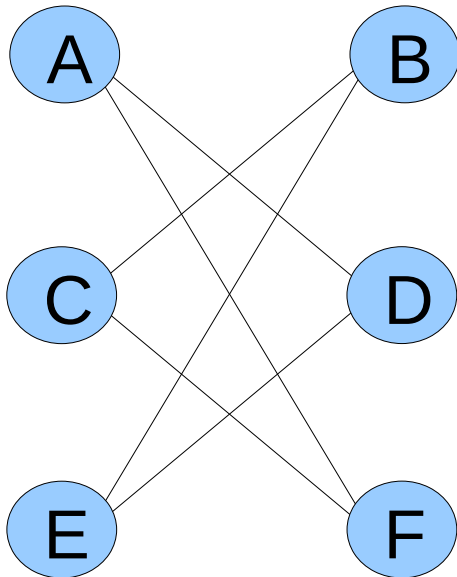
Observações:

- $v[j]$: j -ésimo vértice na ordenação decrescente por grau
- $C[i]$: conjunto de vértices que tem cor i
- $C[r] = \emptyset$: cor r não foi usada

Algoritmo Guloso

- Algoritmo gera uma coloração de G ?
 - sim, corretude é dada pelo próprio funcionamento
- Algoritmo obtém o número cromático?
 - não está claro, mas a resposta é não!

Contra-Exemplo



- Todos os vértices possuem mesmo grau
 - Ordenação 1: A,B,C,D,E,F
 - 3 cores
 - Ordenação 2: A,C,E,B,D,F
 - 2 cores
-
- Número de cores depende da ordenação dos vértices
 - Como saber qual a melhor ordenação para coloração?

Número Cromático

- Não se conhece algoritmo eficiente (tempo polinomial) para determinar o número cromático de um grafo qualquer
 - algoritmo guloso nem sempre usa o menor número de cores
- Determinar se um grafo é *k-colorível* é igualmente difícil, para $k > 2$
 - para $k = 2$ é bem fácil (*dever de casa*)

**Coloração é problema difícil
(e vale 1 milhão de dólares)**

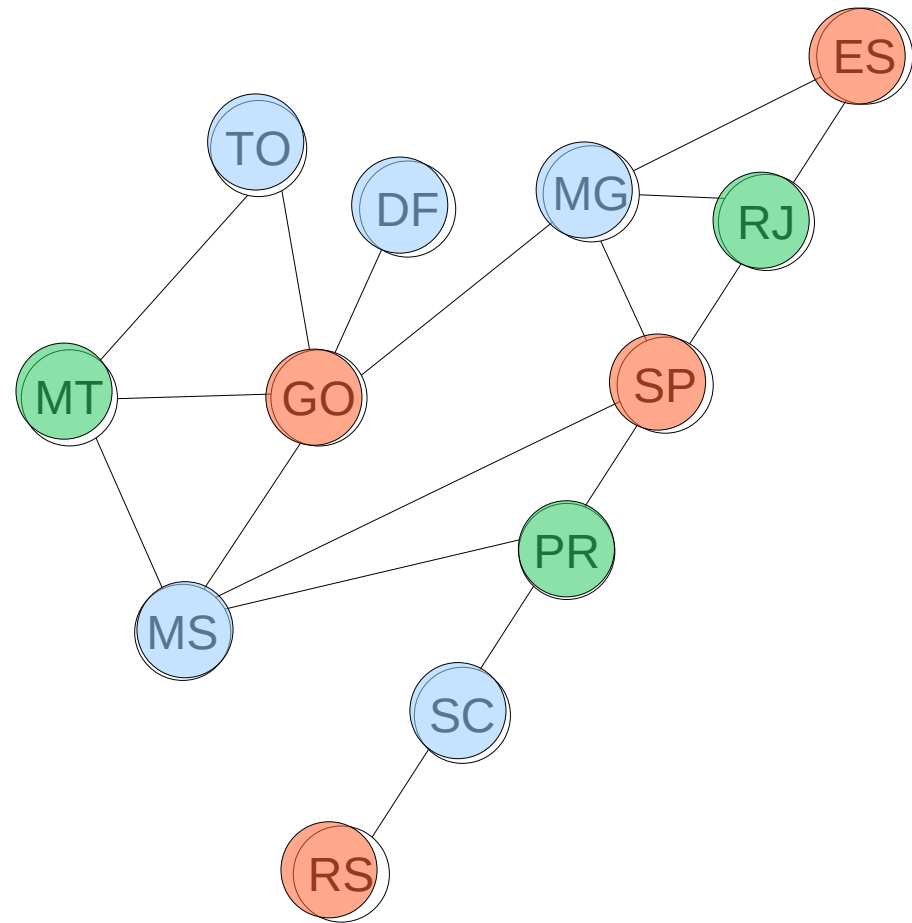
Coloração de Mapas

- Caso especial de coloração de grafos
- Grafo induzido pelo mapa é *planar*
 - restrição geométrica imposta pelas fronteiras
- Um grafo é planar se é possível desenhar o grafo sem cruzar as arestas

- **Problema:** Qual é o menor número de cores necessário para colorir qualquer mapa?

Exemplo

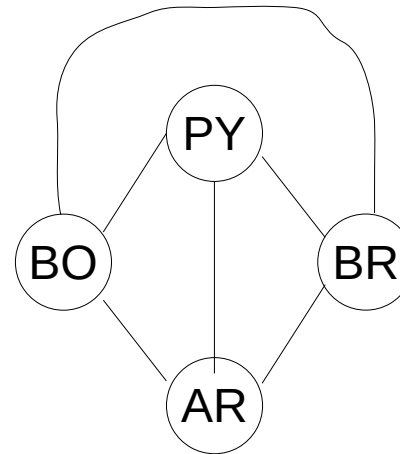
- Região Sul, Sudeste e Centro-Oeste



- Grafo é planar
- Número cromático?

Exemplo com 4 Cores

- Existe mapa que demanda quatro cores?



- Existe mapa que demanda cinco cores?

Não!

Teorema das 4 Cores

- Quatro cores são suficientes para colorir qualquer mapa
- Conjectura de De Morgan em 1852
- Várias provas erradas da conjectura!
- Provado somente em 1972 por Appel, Haken e um computador
 - prova por “força bruta” mostra que não há mapa para qual 5 cores seja necessário
 - Análise de 2000 casos, via computador
- Primeira grande prova com ajuda do computador
- Matemáticos não gostam: e se código tiver bug?
 - amplamente verificada e validada posteriormente