DCA0204, Módulo 8
Grafos

Daniel Aloise

baseado em slides do prof. Leo Liberti, École Polytechnique, França

DCA, UFRN

Sumário

- Definições
- Operações em grafos
- Problemas combinatórios em grafos
- Problemas fáceis e difíceis
- Modelagem para um método de solução genérico

Definições

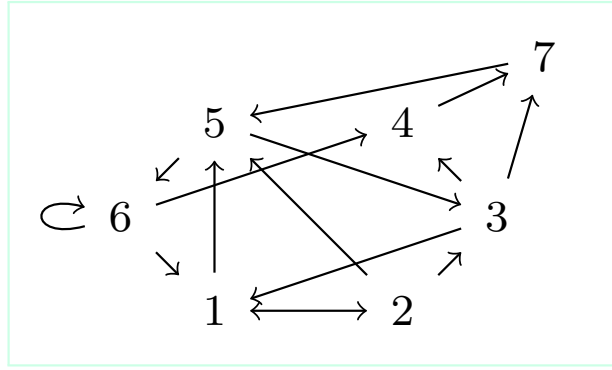
Motivação

A estrutura de dados final

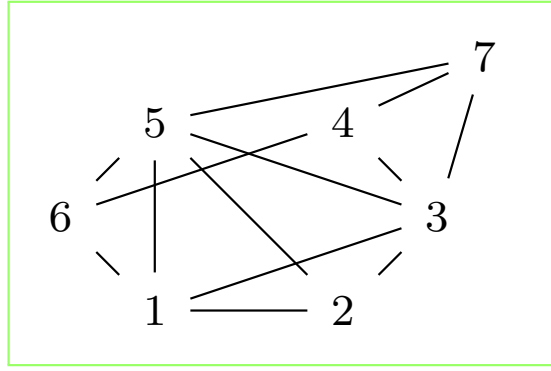
Toda vez que você vir arcos conectando caixas, círculos ou pontos pretos em um curso, você pode pensar em grafos e dígrafos!

Grafos e dígrafos

- Dígrafo $G = (V, A)$: relação A sob o conjunto V
  - V: conjunto de **nós**
  - A: conjunto de **arcos** (u, v) com $u, v \in V$



- Grafo $G = (V, E)$: relação simétrica E sob o conjunto V
  - V: conjunto de **vértices**
  - E: conjunto de **arestas** $\{u, v\}$ com $u, v \in V$



- (Dí)grafos **simples**: a relação é *irreflexiva*
  (I.e., *para todo* $v \in V$, v não está relacionado a ele mesmo)

Observações

- Apresentaremos principalmente resultados para grafos **não-dirigidos**
- A **maioria** dos resultados se extende trivialmente para grafos **dirigidos** (*dígrafos*)

Exemplo

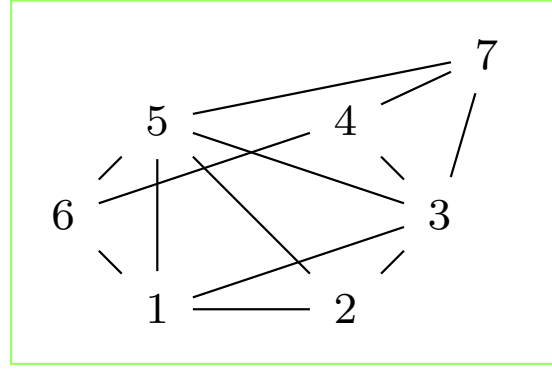
- Se G é um grafos, indicamos o seu conjunto de vértices por $V(G)$ e o seu conjunto de arcos por $E(G)$
- **Exemplo de extensão para dígrafos**:
  Se G é um dígrafo, indicamos o seu conjunto de nós por $V(G)$ e o seu conjunto de arcos por $A(G)$

Stars

Stars: conjunto de nós/vértices ou arcos/arestas adjacente a um dado nó

$\forall v \in V(G)$,

- Se G é não-dirigido,

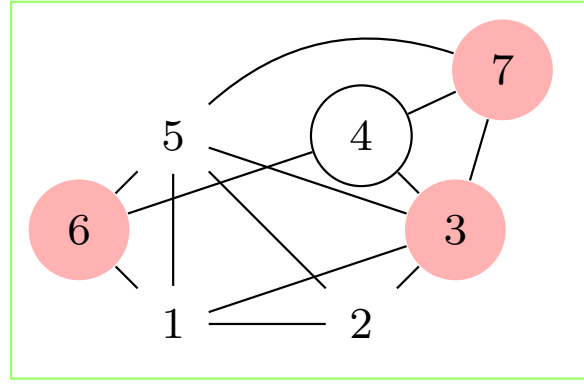


Stars

Stars: conjunto de nós/vértices ou arcos/arestas adjacente a um dado nó

$\forall v \in V(G)$,

- Se G é não-dirigido,
  - $N(v) = \{u \in V \mid \{u, v\} \in E(G)\}$

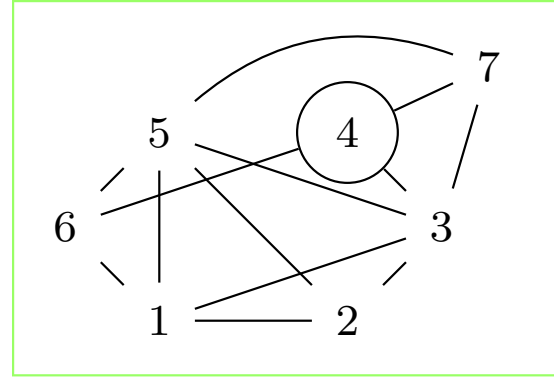


Stars

Stars: conjunto de nós/vértices ou arcos/arestas adjacente a um dado nó

$\forall v \in V(G)$,

- Se G é não-dirigido,
  - $N(v) = \{u \in V \mid \{u, v\} \in E(G)\}$
  - $\delta(v) = \{\{u, v\} \mid u \in N(v)\}$

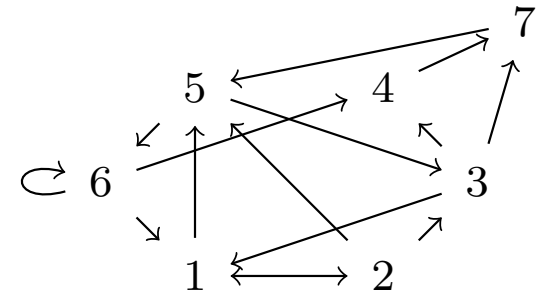


Stars

Stars: conjunto de nós/vértices ou arcos/arestas adjacente a um dado nó

$\forall v \in V(G)$,

- Se G é não-dirigido,
  - $N(v) = \{u \in V \mid \{u, v\} \in E(G)\}$
  - $\delta(v) = \{\{u, v\} \mid u \in N(v)\}$
- Se G é dirigido,

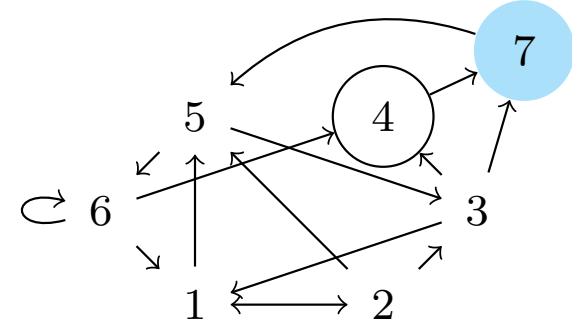


Stars

Stars: conjunto de nós/vértices ou arcos/arestas adjacente a um dado nó

$\forall v \in V(G)$,

- Se G é não-dirigido,
  - $N(v) = \{u \in V \mid \{u, v\} \in E(G)\}$
  - $\delta(v) = \{\{u, v\} \mid u \in N(v)\}$
- Se G é dirigido,
  - $N^+(v) = \{u \in V \mid (v, u) \in E(G)\}$

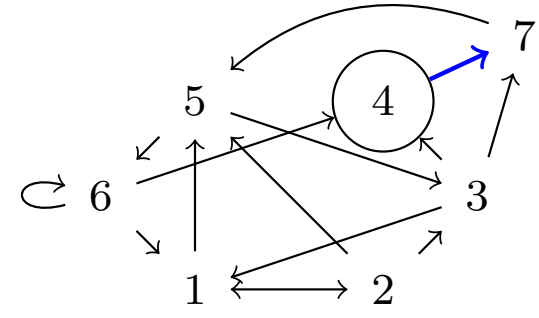


Stars

Stars: conjunto de nós/vértices ou arcos/arestas adjacente a um dado nó

$\forall v \in V(G)$,

- Se G é não-dirigido,
  - $N(v) = \{u \in V \mid \{u, v\} \in E(G)\}$
  - $\delta(v) = \{\{u, v\} \mid u \in N(v)\}$
- Se G é dirigido,
  - $N^+(v) = \{u \in V \mid (v, u) \in E(G)\}$
  - $\delta^+(v) = \{(v, u) \mid u \in N^+(v)\}$

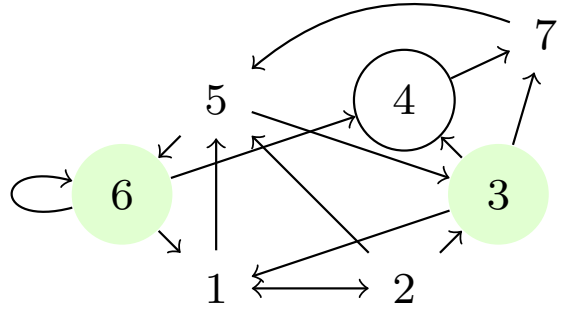


Stars

Stars: conjunto de nós/vértices ou arcos/arestas adjacente a um dado nó

$\forall v \in V(G)$,

- Se G é não-dirigido,
  - $N(v) = \{u \in V \mid \{u, v\} \in E(G)\}$
  - $\delta(v) = \{\{u, v\} \mid u \in N(v)\}$
- Se G é dirigido,
  - $N^+(v) = \{u \in V \mid (v, u) \in E(G)\}$
  - $\delta^+(v) = \{(v, u) \mid u \in N^+(v)\}$
  - $N^-(v) = \{u \in V \mid (u, v) \in E(G)\}$

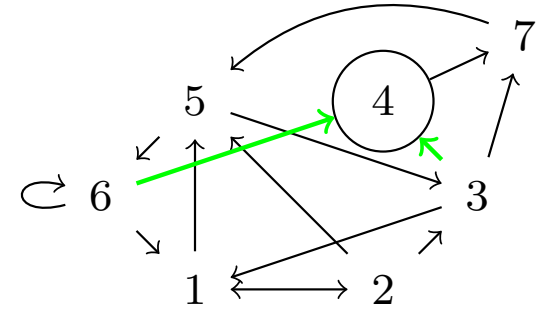


Stars

Stars: conjunto de nós/vértices ou arcos/arestas adjacente a um dado nó

$\forall v \in V(G)$,

- Se G é não-dirigido,
  - $N(v) = \{u \in V \mid \{u, v\} \in E(G)\}$
  - $\delta(v) = \{\{u, v\} \mid u \in N(v)\}$
- Se G é dirigido,
  - $N^+(v) = \{u \in V \mid (v, u) \in E(G)\}$
  - $\delta^+(v) = \{(v, u) \mid u \in N^+(v)\}$
  - $N^-(v) = \{u \in V \mid (u, v) \in E(G)\}$
  - $\delta^-(v) = \{(u, v) \mid u \in N^-(v)\}$



Stars

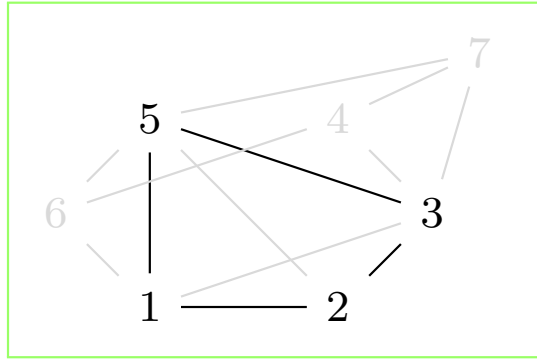
Stars: conjunto de nós/vértices ou arcos/arestas adjacente a um dado nó

$\forall v \in V(G)$,

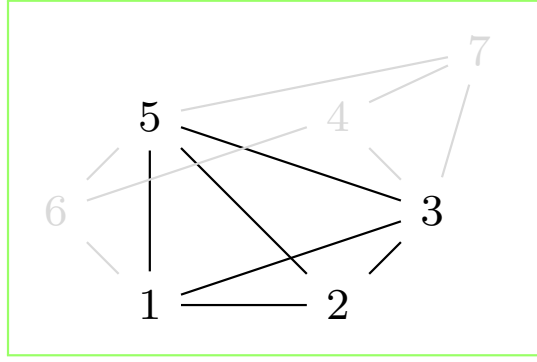
- Se G é não-dirigido,
  - $N(v) = \{u \in V \mid \{u, v\} \in E(G)\}$
  - $\delta(v) = \{\{u, v\} \mid u \in N(v)\}$
- Se G é dirigido,
  - $N^+(v) = \{u \in V \mid (v, u) \in E(G)\}$
  - $\delta^+(v) = \{(v, u) \mid u \in N^+(v)\}$
  - $N^-(v) = \{u \in V \mid (u, v) \in E(G)\}$
  - $\delta^-(v) = \{(u, v) \mid u \in N^-(v)\}$
- $|N(v)|$ =**grau**, $|N^+(v)|$ =**grau de saída**, $|N^-(v)|$ =**grau de entrada** of v

Subgrafos

- Um grafo $H = (U, F)$ é um **subgrafo** de $G = (V, E)$ se $U \subseteq V$, $F \subseteq E$ e $\forall \{u, v\} \in F \ (u, v \in U)$



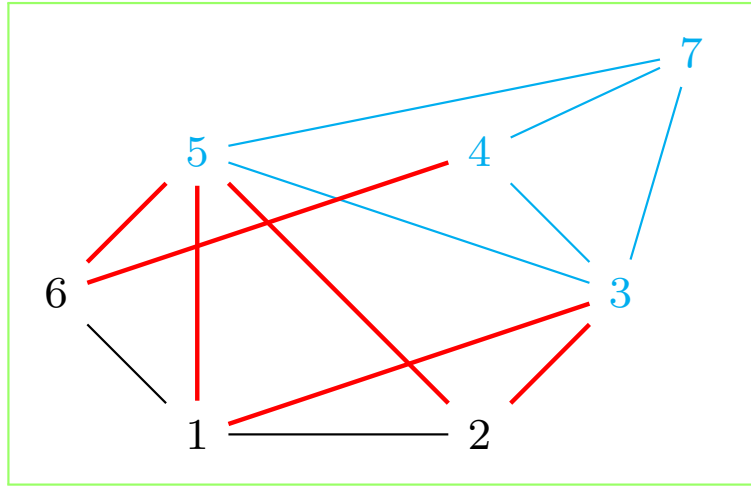
- Um subgrafo $H = (U, F)$ de $G = (V, E)$ é **gerador** se $U = V$
- Um subgrafo $H = (U, F)$ de $G = (V, E)$ é **induzido** por U se $\forall u, v \in U \ (\{u, v\} \in E \rightarrow \{u, v\} \in F)$



Notação de subgrafo induzido: $H = G[U]$

Corte

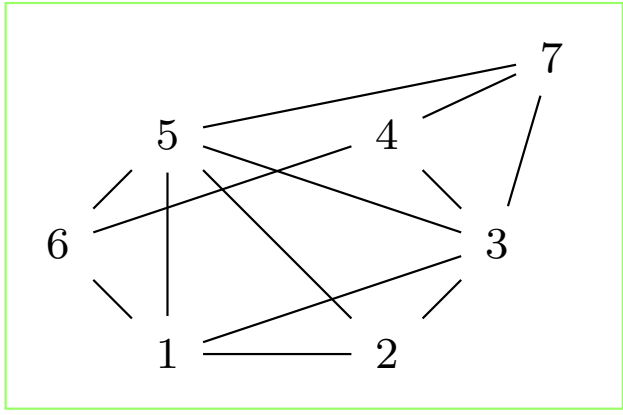
- Seja $H = (U, F)$ um subgrafo de $G = (V, E)$ (i.e. $U \subsetneq V$)
- O **corte** $\delta(H) = \left(\bigcup_{u \in U} \delta(u) \right) \setminus F$
  é o conjunto de arestas "separando" U e $V \setminus U$
- Ex. seja $U = \{1, 2, 6\}$ e $H = G[U]$, então $\delta(H)$ é mostrado pelas arestas vermelhas abaixo



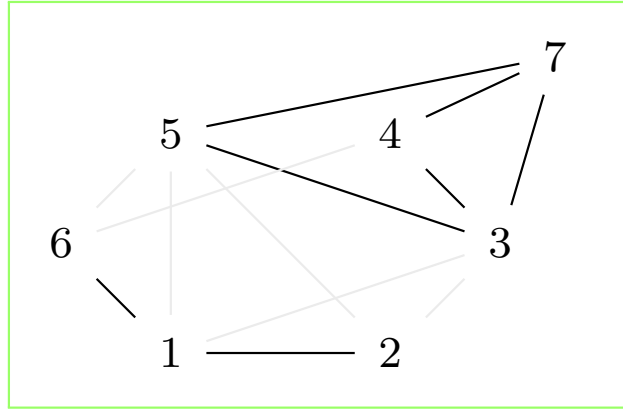
- Difinições similares são válidas para **cortes dirigidos**
- Se G é dirigido, $\delta(U) = \delta(V \setminus U)$ para todo $U \subseteq V(G)$

Conectividade

- Um grafo é **conexo** se não existem cortes (não-triviais) vazios.



Connected

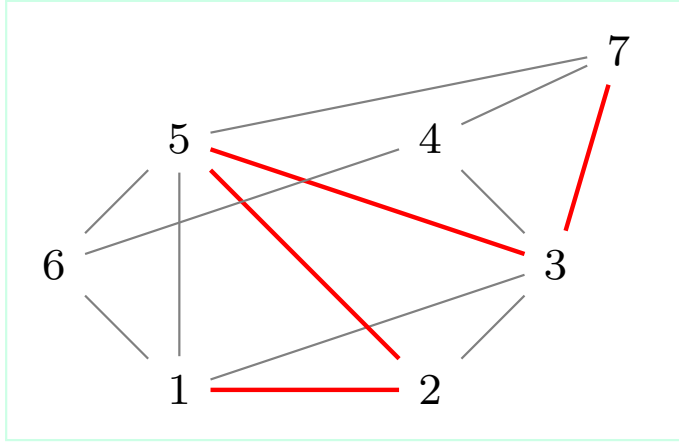


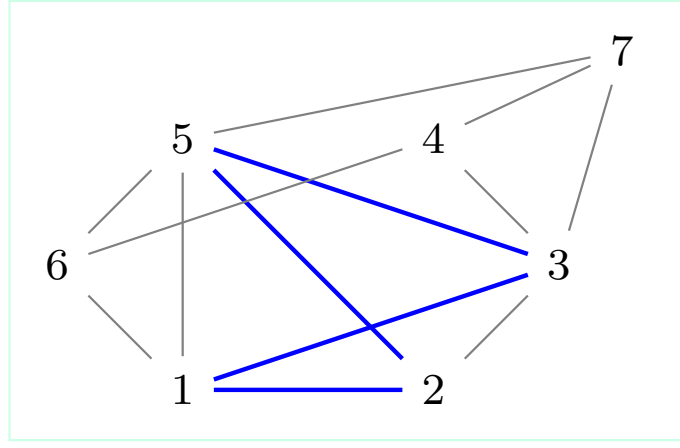
Desconexo: $\delta(\{1,2,6\}) = \varnothing$

- Cada subgrafo conexo maximal de um grafo é uma **componente conexa**
- A maioria dos algoritmos em grafos assume que o grafo de entrada é conexo, senão eles são aplicados para cada componente conexa

Caminhos e ciclos

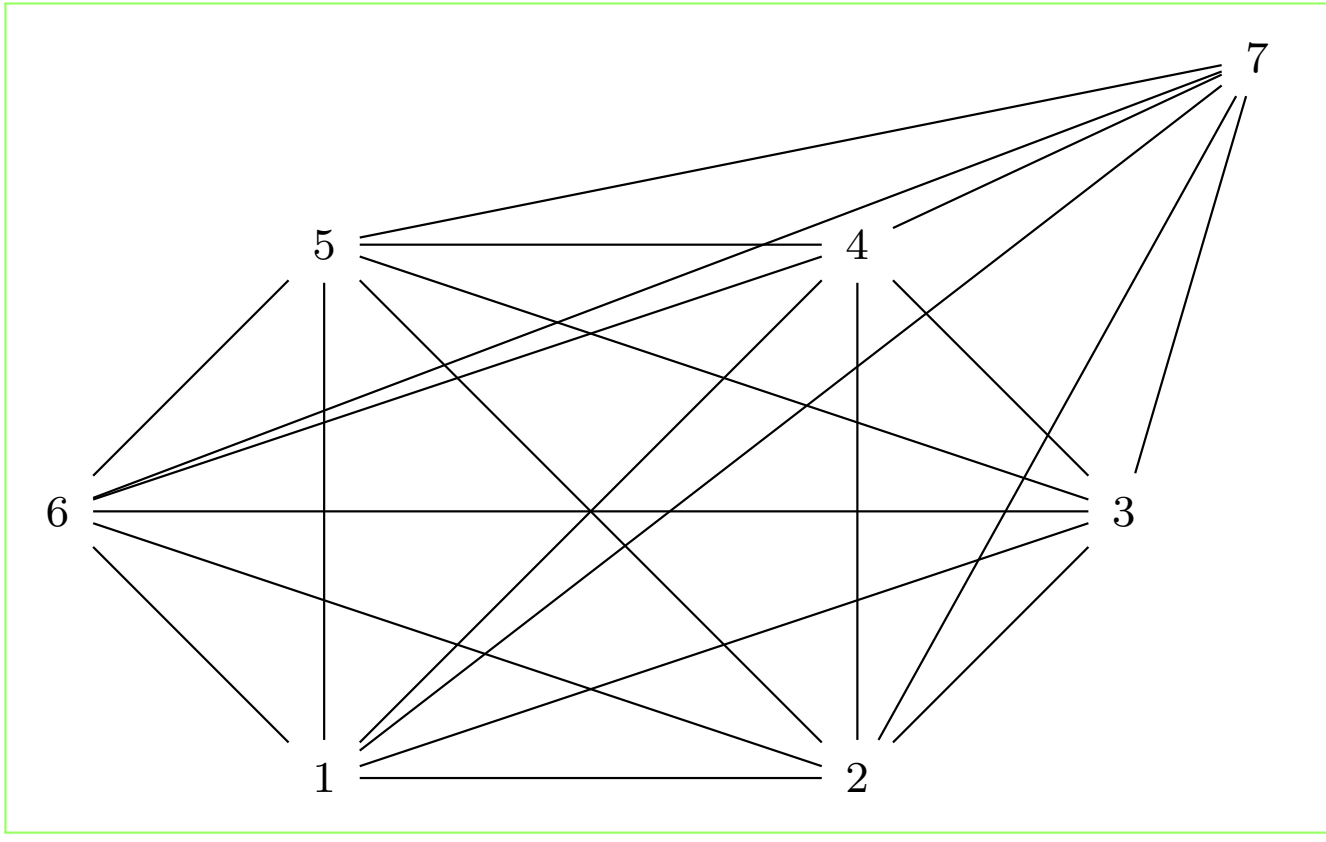
- Seja G um grafo e $u, v \in V(G)$
- Um **caminho simples** P de u até v em G é um subgrafo conexo de G s.t.:
  1. cada vértice w em P diferente de u, v tem $|N(w)| = 2$
  2. se $u \neq v$ então $|N(u)| = |N(v)| = 1$
  3. se $u = v$ então ou $E(P) = \varnothing$ ou $|N(u)| = |N(v)| = 2$
- Um caminho de u até v é denotado $u \to v$
- Se P é um caminho $u \to v$, então u, v são denominadas **extremidades** do caminho
- Um **ciclo simples** é um caminho simples com extremidades iguais





Grafo completo

- O **grafo completo** K_n em n vértices tem todas as possíveis arestas

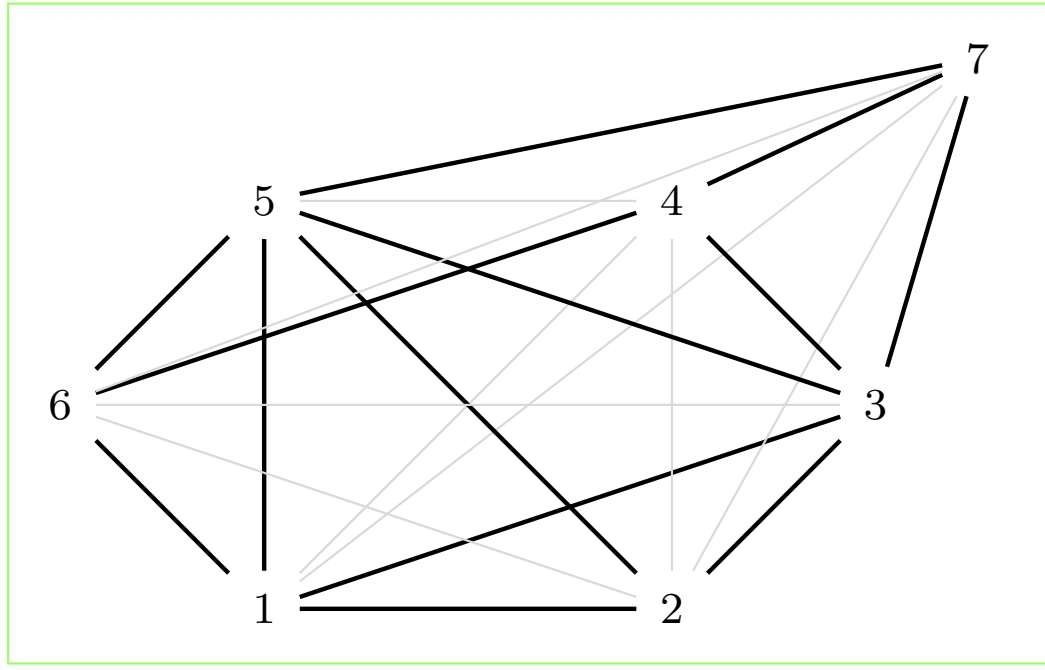


- K_n é também chamado n**-clique**; um grafo completo em um conjunto U de vértices é denotado por $K(U)$

Grafo complementar

- Dado $G = (V, F)$ com $|V| = n$, o **complemento** de G é $\bar{G} = (V, E(K_n) \setminus F)$

G

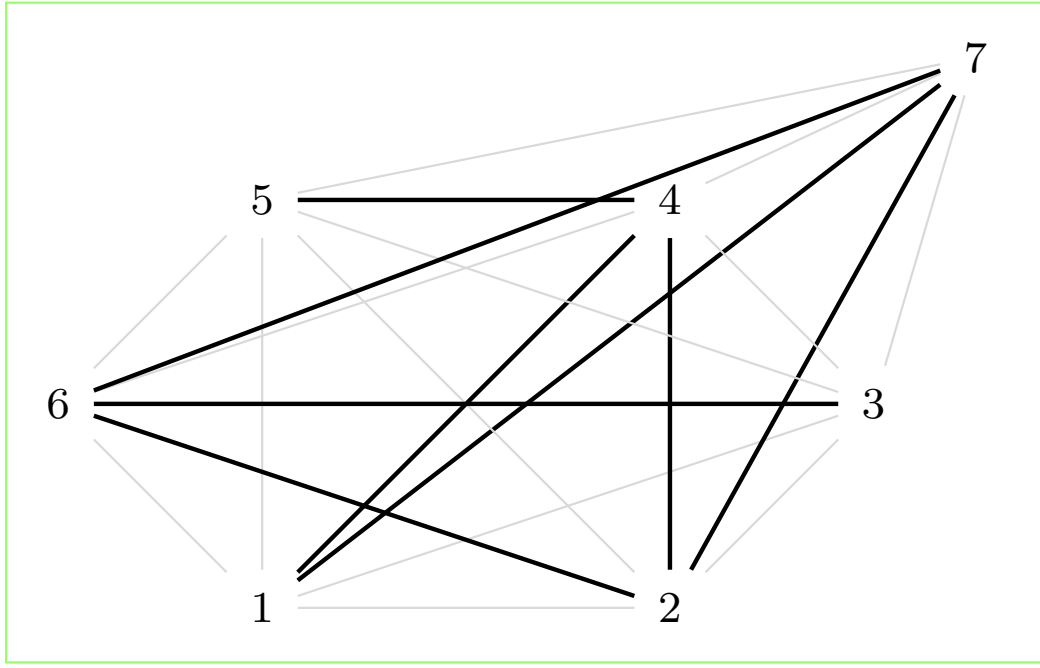


- O complemento de K_n é o **grafo vazio** $(V, \varnothing)$ em n vértices

Grafo complementar

- Dado $G = (V, F)$ com $|V| = n$, o **complemento** de G é $\bar{G} = (V, E(K_n) \setminus F)$

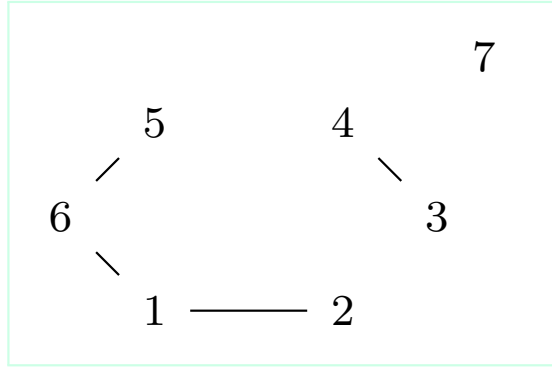
$\bar{G}$



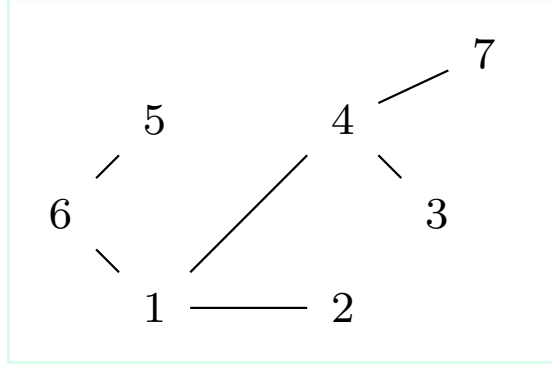
- O complemento de K_n é o **grafo vazio** $(V, \varnothing)$ em n vértices

Florestas e árvores

- Uma **floresta** é um grafo sem ciclos



- Uma **árvore** é uma floresta conexa



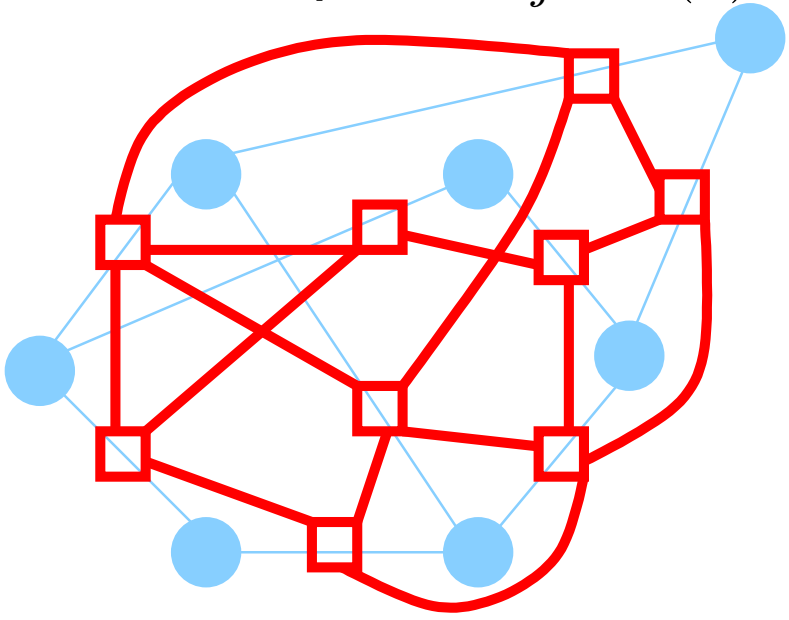
- Se uma árvore é um subgrafo gerador de outro grafo G, ela é denominada uma **árvore geradora**

Grafo linha

- Dado um grafo $G = (V, E)$ onde $E = \{e_1, \ldots, e_m\}$
- O **grafo linha** de G é

$$L(G) = (E, \{\{e_i, e_j\} \mid e_i \cap e_j \neq \varnothing\})$$

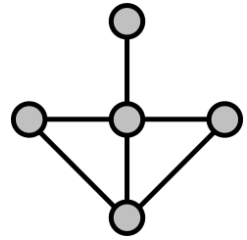
- Todo vértice de $L(G)$ é uma aresta de G
- Dois vértices e_i, e_j de $L(G)$ são adjacentes se existe $v \in V$ tal que $e_i, e_j \in \delta(v)$



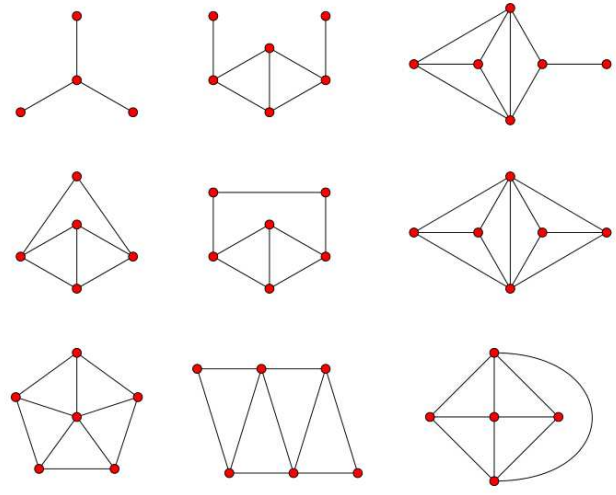
Propriedade: o grau $|N_{L(G)}(e)|$ de um vértice $e = \{u, v\}$ de $L(G)$ é $|N_G(u)| + |N_G(v)| - 2$.

Grafo linha

- Tente encontrar o grafo original do grafo linha abaixo



- Nove grafos minimais de *Beineke*



- Um grafo é um grafo linha sse ele não possui um destes nove grafos como subgrafo induzido.

Operações em grafos

Adição e remoção

- Adicione um vértice v:

  atualize $V \leftarrow V \cup \{v\}$

- Adicione um arco $e = \{u, v\}$:

  adicione vértices u, v, atualize $E \leftarrow E \cup \{e\}$

- Remova uma aresta $e = \{u, v\}$:

  atualize $E \leftarrow E \setminus \{e\}$

- Remova um vértice v:

  atualize $V \leftarrow V \setminus \{v\}$ e $E \leftarrow E \setminus \delta(v)$

Problemas combinatórios em grafos

O problema do subgrafo

- Seja $\mathbb{G}$ a classe de todos os grafos
- Para um conjunto de proposições válidas $P(G)$ (para $G \in \mathbb{G}$), um problema típico de decisão em teoria dos grafos é:

  PROBLEMA DE SUBGRAFO (SP_P). Dado um grafo G, ele tem um subgrafo H com propriedade P?

- **Problema de decisão**: questão do tipo SIM/NÃO para qualquer instância do problema (i.e., a entrada)
- Requer que problemas de decisão sempre forneçam um **certificado** (uma prova que certifica a resposta)
- Ex. se $P(H) \equiv (H$ é um ciclo$)$ o certificado é o ciclo
- **NP** é a classe de problemas de decisão cujos certificados para instâncias SIM podem ser verificados em tempo polinomial com respeito ao tamanho da instância.

Problemas de otimização em grafos

- Para a maioria dos problemas de decisão em grafos existe um *problema de otimização* correspondente
- Seja $\mu : \mathbb{G} \to \mathbb{R}$ uma função de "medida" para grafos
- Ex. μ pode ser o número de vértices, ou de arestas

  PROBLEMA DE SUBGRAFO, *versão de otimização* ($\text{SP}_{P,\mu}$).
  Dado um grafo G, ele tem um subgrafo H com propriedade P tendo uma medida mínima/máxima μ?

- Dada uma propriedade P e uma medida do grafo μ, o **problema de otimização** considera todas as instâncias possíveis.

Problemas fáceis

- Seja **P** a classe de problemas de decisão ou de otimização que podem ser resolvidos em tempo polinomial.
- Chamamos os problemas em **P** de “fáceis”
  - ÁRVORE GERADORA MÍNIMA

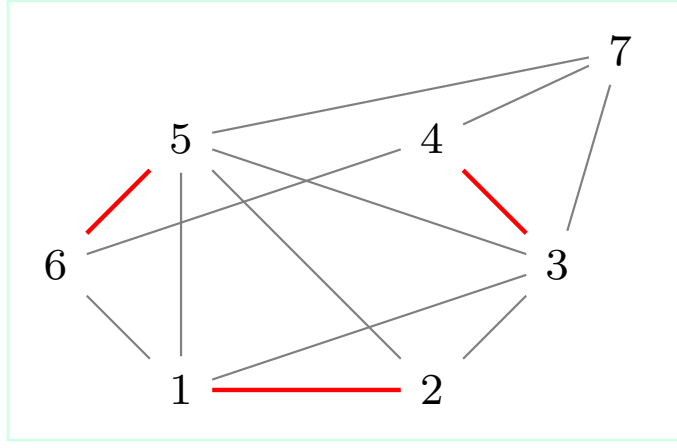
    Vista no Módulo 6
  - PROBLEMA DO CAMINHO MAIS CURTO de um vértice v para todos os outros vértices

    será visto no Módulo 9
  - PROBLEMA DO MATCHING MÁXIMO (MATCHING)

    Matching: subgrafo dado por um conjunto de arestas mutualmente não-adjacentes

    Um matching máximo M,

    $\mu(M) = |E(M)|$

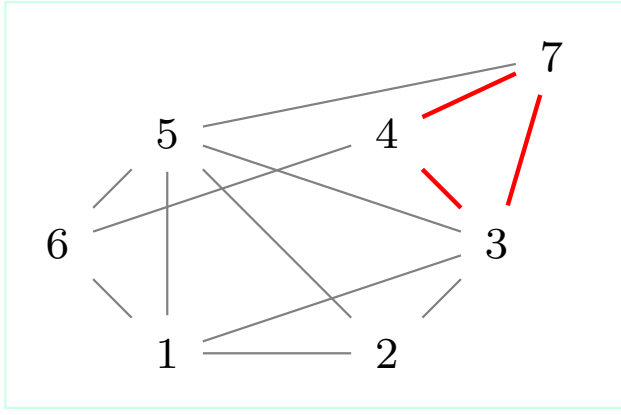


Problemas (mais) difíceis

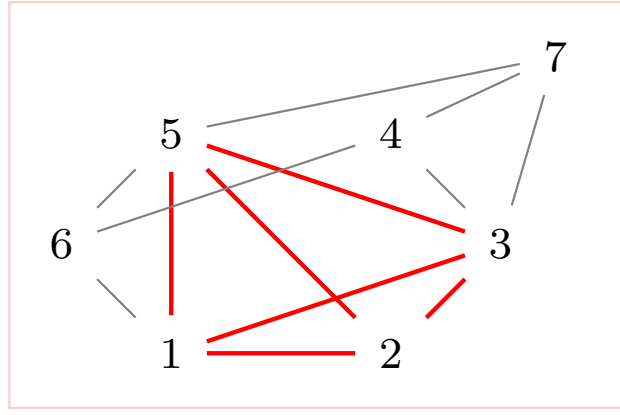
Clique máximo

PROBLEMA DE CLIQUE (CLIQUE). Dado um grafo G, qual é o maior n tal que G tem um K_n como um subgrafo?

- In CLIQUE, $\mu(H) = |V(H)|$ and $P(H) \equiv H = K(V(H))$



Um clique em G



O maior clique em G

- Exemplo de aplicação: Redes sociais
  - Pessoas em uma clique podem indicar um grupo de colegas em uma sala de aula ou uma célula terrorista

Clique e NP-completude

- A versão de decisão da CLIQUE é:

  PROBLEMA DA k-CLIQUE (k-CLIQUE). Dado um grafo G e um inteiro $k > 0$, G tem K_k como um subgrafo?

- Considere o resultado a seguir (que não iremos provar)

  Thm.

  [Karp 1972] Se CLIQUE $\in$ **P** então **P** $=$ **NP**

- Qualquer problema de decisão para o qual tal resultado é válido é chamado **NP**-completo

- Não se sabe se problemas **NP**-completos podem ser resolvidos em tempo polinomial; o palpite geral é que NÃO

Resolvendo problemas NP-completos

- Essencialmente, provar a **NP**-completude de um problema é igual a dizer que “ele é realmente difícil”

  Se ele fosse fácil, qualquer problema em **NP** seria fácil, o que é improvável: portanto ele é provavelmente difícil mesmo

- Métodos de solução para problemas **NP**-completos incluem:
  - algoritmos **exatos** mas com complexidade **exponencial** de pior caso
  - **heurísticas**

    sempre que oferecem uma resposta SIM elas fornecem um certificado, mas não há garantia de que elas forneçam uma resposta NÃO em um horizonte de tempo finito

- Um problema de otimização é dito **NP-difícil** se caso ele seja resolvido em tempo polinomial $\Rightarrow$ **P** $=$ **NP**.

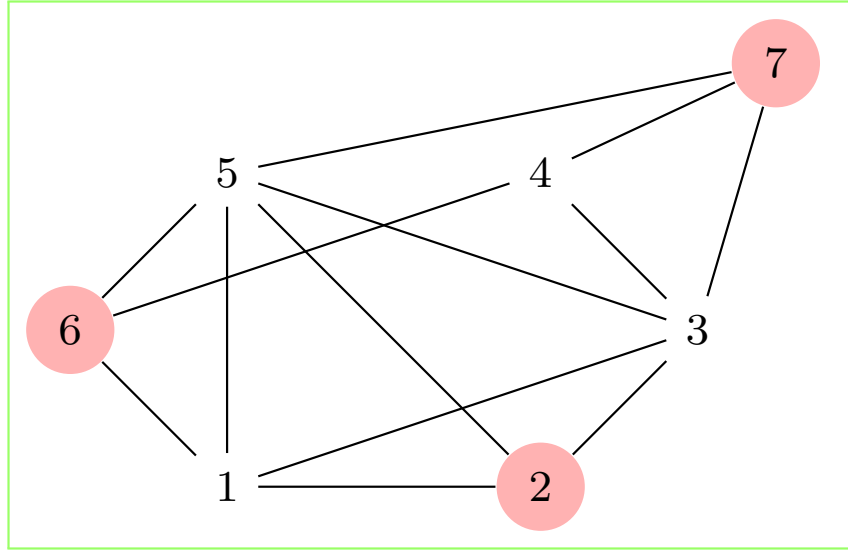
Conjuntos Independentes

- Um conjunto independente **CI** de um grafo $G = (V, E)$ é um subconjunto $U \subseteq V$ tal que $\forall u, v \in U \ (\{u, v\} \notin E)$

Thm.

U é um CI em G se e somente se $\bar{G}[U]$ é uma clique

um CI em G



- Problema de decisão: k-CI

Dados G e $k \in \mathbb{N}$, existe um CI $U \subseteq V(G)$ de tamanho k?

- Problema de otimização: CI

Dado G, encontre o CI de G de tamanho máximo

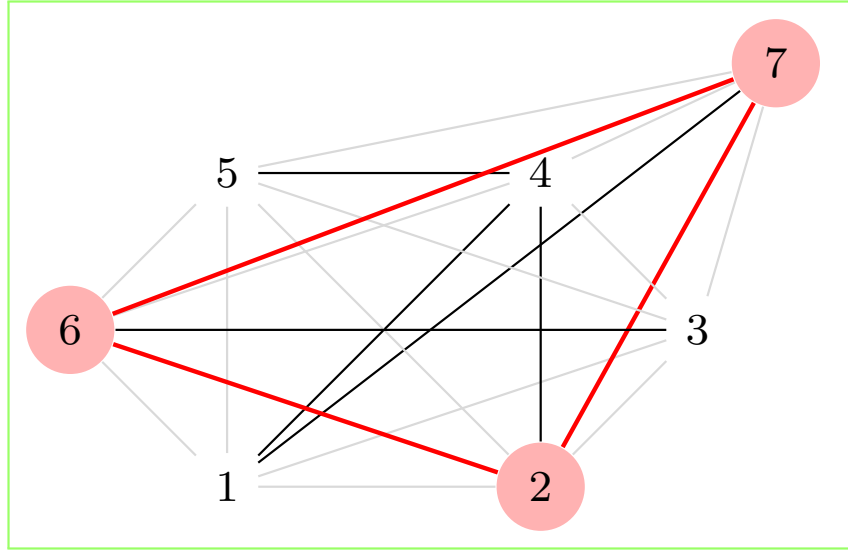
Conjuntos Independentes

- Um conjunto independente **CI** de um grafo $G = (V, E)$ é um subconjunto $U \subseteq V$ tal que $\forall u, v \in U \ (\{u, v\} \notin E)$

Thm.

U é um CI em G se e somente se $\bar{G}[U]$ é uma clique

$\bar{G}[U]$ *é uma clique*



- Problema de decisão: k-CI

Dados G e $k \in \mathbb{N}$, existe um CI $U \subseteq V(G)$ de tamanho k?

- Problema de otimização: CI

Dado G, encontre o CI de G de tamanho máximo

NP-completude de k-CI

Thm.

k-CI é NP-completo

Proof

Considere uma instância (G, k) de k-CLIQUE

O grafo complementar $\bar{G}$ pode ser obtido em tempo polinomial $(*)$

É fácil mostrar que $\bar{\bar{G}} = G$ $(**)$

Devido a $(**)$ e o thm. anterior,

(G, k) é uma instância SIM de k-CLIQUE sse $(\bar{G}, k)$ é uma instância SIM de k-CI

Por $(*)$, se k-CI $\in$ **P** então k-CLIQUE $\in$ **P**

NP-completude de k-CLIQUE, k-CI $\in$ **P** implica **P** $=$ **NP**

Portanto, k-CI é **NP**-completo

- Como demonstrar que um problema $\mathcal{P}$ é **NP**-completo:
  - Pegue outro problema sabidamente **NP**-completo $\mathcal{Q}$ "similar" a $\mathcal{P}$
  - Transforme em tempo polinom. uma instância de $\mathcal{Q}$ em uma instância de $\mathcal{P}$
  - Mostre que a transformação preserva a resposta SIM/NÃO

Heurística CI

- O método **guloso** a seguir encontra um CI *maximal*

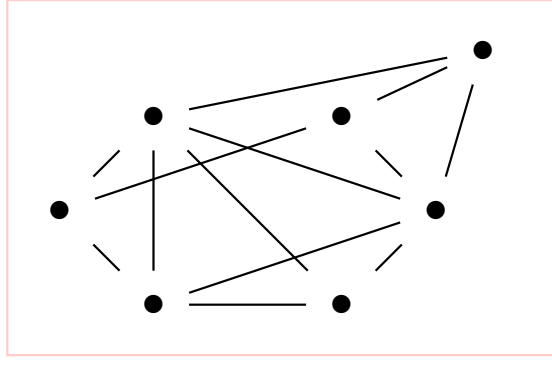
1: $U = \varnothing$;
2: ordene V por valores crescentes de $|N(v)|$;

3: **while** $V \neq \varnothing$ **do**
4: $\quad v = \min V$;
5: $\quad U \leftarrow U \cup \{v\}$;
6: $\quad V \leftarrow V \setminus (\{v\} \cup N(v))$
7: **end while**

- Pior caso: $O(n)$ (dado por um grafo sem arestas)

sequência de graus

$(3, 3, 3, 3, 4, 5, 5)$



Heurística CI

- O método **guloso** a seguir encontra um CI *maximal*

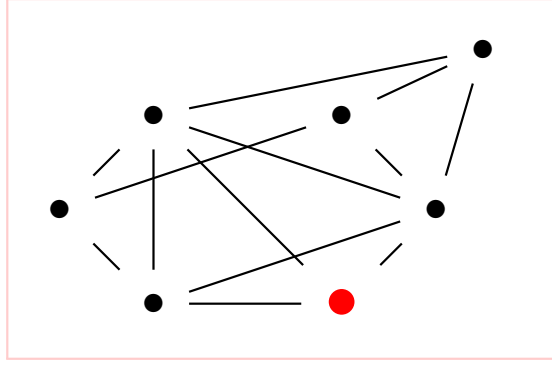
1: $U = \varnothing$;
2: ordene V por valores crescentes de $|N(v)|$;

3: **while** $V \neq \varnothing$ **do**
4: $\quad v = \min V$;
5: $\quad U \leftarrow U \cup \{v\}$;
6: $\quad V \leftarrow V \setminus (\{v\} \cup N(v))$
7: **end while**

- Pior caso: $O(n)$ (dado por um grafo sem arestas)

selecione $\min V$

coloque-o em U



Heurística CI

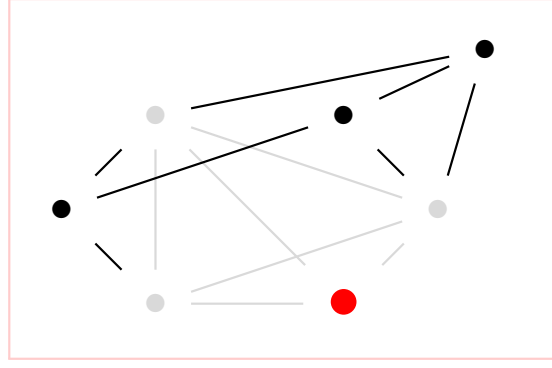
- O método **guloso** a seguir encontra um CI *maximal*

1: $U = \varnothing$;
2: ordene V por valores crescentes de $|N(v)|$;

3: **while** $V \neq \varnothing$ **do**
4: $\quad v = \min V$;
5: $\quad U \leftarrow U \cup \{v\}$;
6: $\quad V \leftarrow V \setminus (\{v\} \cup N(v))$
7: **end while**

- Pior caso: $O(n)$ (dado por um grafo sem arestas)

remova v e sua estrela de V



Heurística CI

- O método **guloso** a seguir encontra um CI *maximal*

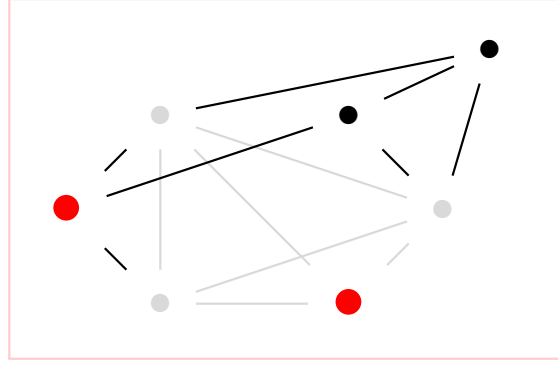
1: $U = \varnothing$;
2: ordene V por valores crescentes de $|N(v)|$;

3: **while** $V \neq \varnothing$ **do**
4: $\quad v = \min V$;
5: $\quad U \leftarrow U \cup \{v\}$;
6: $\quad V \leftarrow V \setminus (\{v\} \cup N(v))$
7: **end while**

- Pior caso: $O(n)$ (dado por um grafo sem arestas)

selecione $\min V$

coloque-o em U



Heurística CI

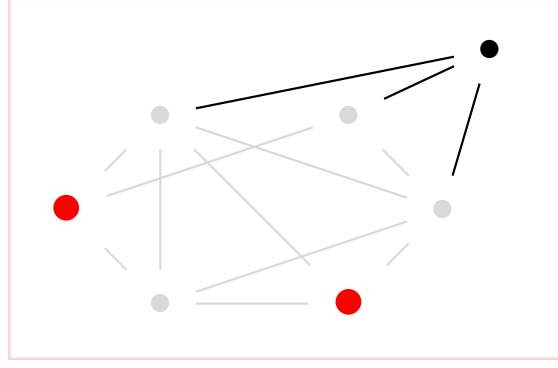
- O método **guloso** a seguir encontra um CI *maximal*

1: $U = \varnothing$;
2: ordene V por valores crescentes de $|N(v)|$;

3: **while** $V \neq \varnothing$ **do**
4: $\quad v = \min V$;
5: $\quad U \leftarrow U \cup \{v\}$;
6: $\quad V \leftarrow V \setminus (\{v\} \cup N(v))$
7: **end while**

- Pior caso: $O(n)$ (dado por um grafo sem arestas)

remova v e sua estrela de V



Heurística CI

- O método **guloso** a seguir encontra um CI *maximal*

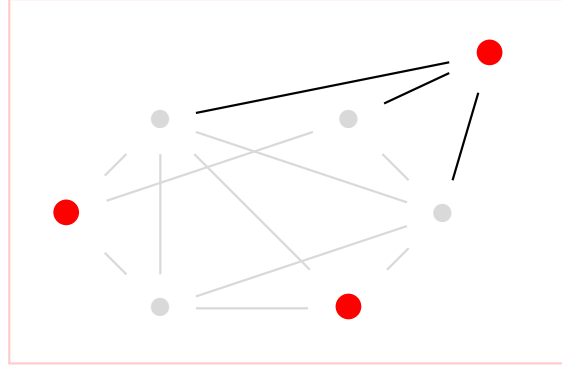
1: $U = \varnothing$;
2: ordene V por valores crescentes de $|N(v)|$;

3: **while** $V \neq \varnothing$ **do**
4: $\quad v = \min V$;
5: $\quad U \leftarrow U \cup \{v\}$;
6: $\quad V \leftarrow V \setminus (\{v\} \cup N(v))$
7: **end while**

- Pior caso: $O(n)$ (dado por um grafo sem arestas)

selecione $\min V$

coloque-o em U



Heurística CI

- O método **guloso** a seguir encontra um CI *maximal*

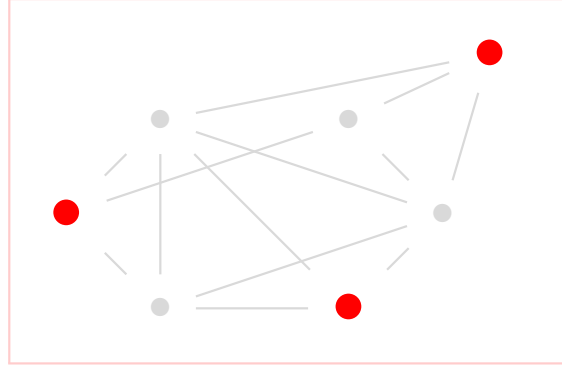
1: $U = \varnothing$;
2: ordene V por valores crescentes de $|N(v)|$;

3: **while** $V \neq \varnothing$ **do**
4: $\quad v = \min V$;
5: $\quad U \leftarrow U \cup \{v\}$;
6: $\quad V \leftarrow V \setminus (\{v\} \cup N(v))$
7: **end while**

- Pior caso: $O(n)$ (dado por um grafo sem arestas)

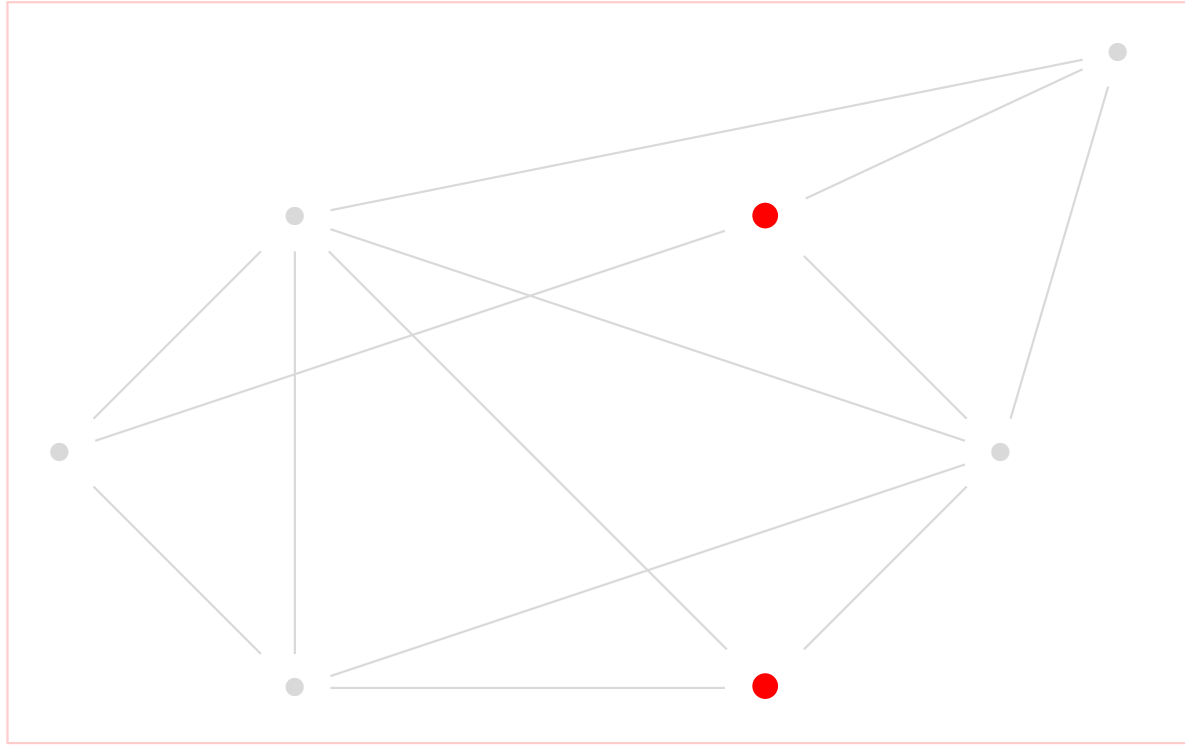
remova v e sua estrela de V

stop: CI maximal



Heurística falha

- O algoritmo acima pode falhar em encontrar um CI máximo
- Ao se escolher o segundo elemento de U, ao invés do vértice mais à esquerda, temos:



- O algoritmo para imediatamente com um CI de cardinalidade 2

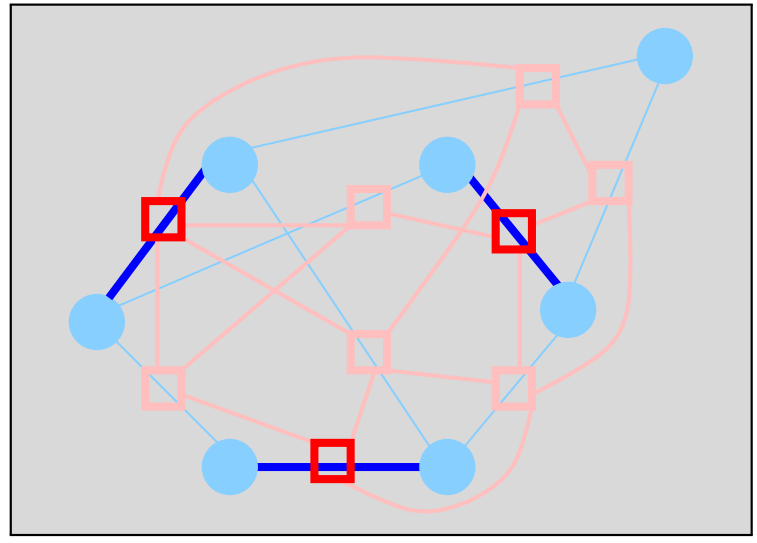
Um caso polinomial

- Nem todas as instâncias de um problema **NP**-completo são difíceis
- Seja P um problema de decisão e C$\subseteq$P um conjunto infinito de instâncias para as quais existe um algoritmo polinomial
- Então C$\in$**P**, e C é um **caso polinomial** de P
- Por exemplo, seja $\mathcal{L} = \{H \in \mathbb{G} \mid \exists G \in \mathbb{G}\ (H = L(G))\}$ a classe de grafos que são grafos linha de outro grafo

Thm.

Um matching máximo em G é um CI máximo em $L(G)$

Proof



- Uma vez que MATCHING$\in$**P** e podemos encontrar $L(G)$ em tempo polinom., $\text{CI}_{\mathcal{L}} \in$**P**

Coloração de vértices

- Problema de decisão

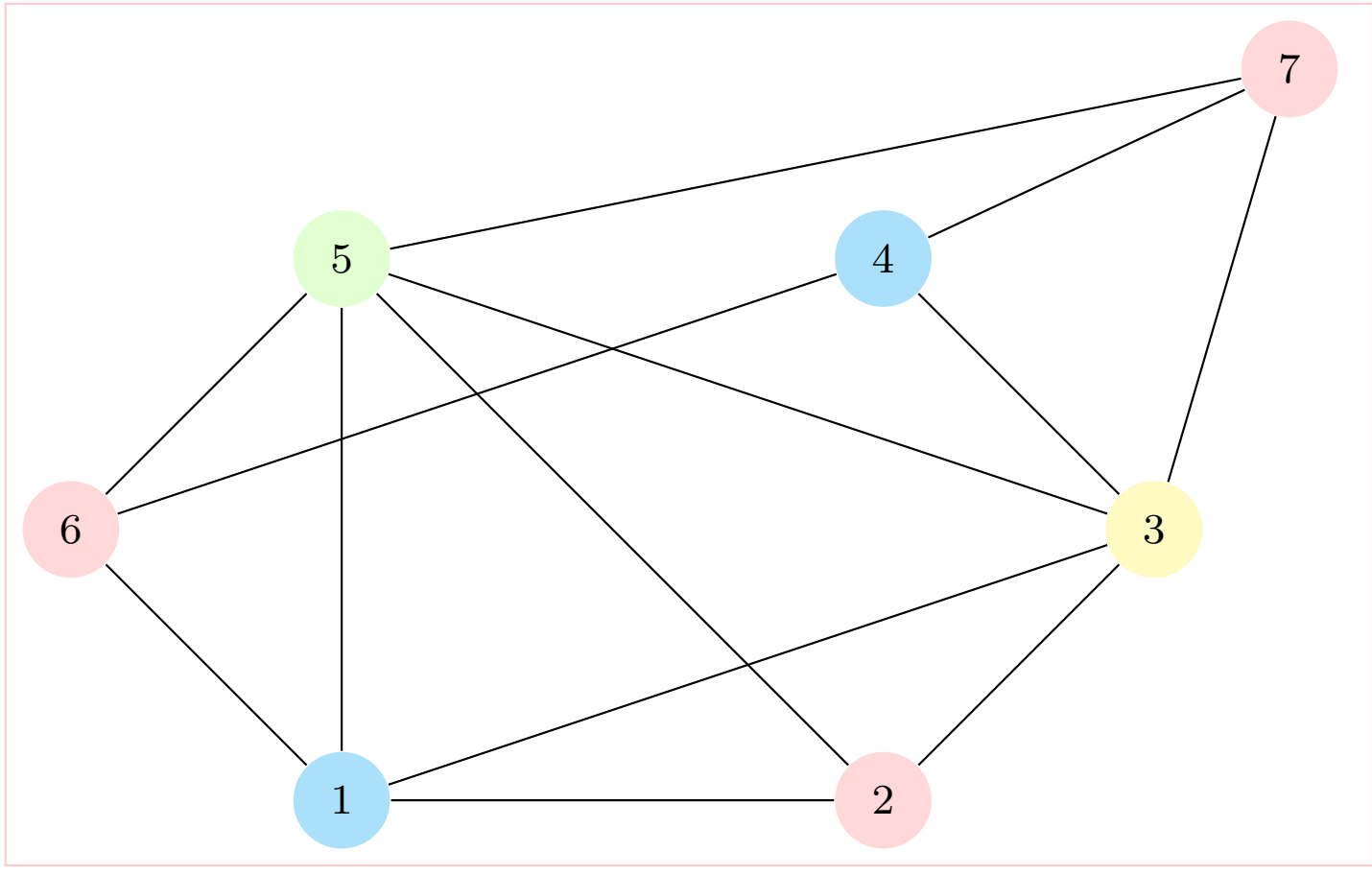
  PROBLEMA DE k-COLORAÇÃO. Dado um grafo $G = (V, E)$ e um inteiro $k > 0$, encontre uma função $c : V \to \{1, \ldots, k\}$ tal que $\forall \{u, v\} \in E \; (c(u) \neq c(v))$

- Problema de otimização

  PROBLEMA DE COLORAÇÃO. Dado um grafo $G = (V, E)$, encontre o $k \in \mathbb{N}$ mínimo tal que existe uma função $c : V \to \{1, \ldots, k\}$ with $\forall \{u, v\} \in E \; (c(u) \neq c(v))$

- Aplicações em scheduling e em redes sem fio.
- Sudoku!

Exemplo de coloração de vértices



Heurística de coloração de vértices

Thm.

Cada conjunto de cores $C_k = \{v \in V \mid c(v) = k\}$ é um CI

- Usa heurística CI como subpasso

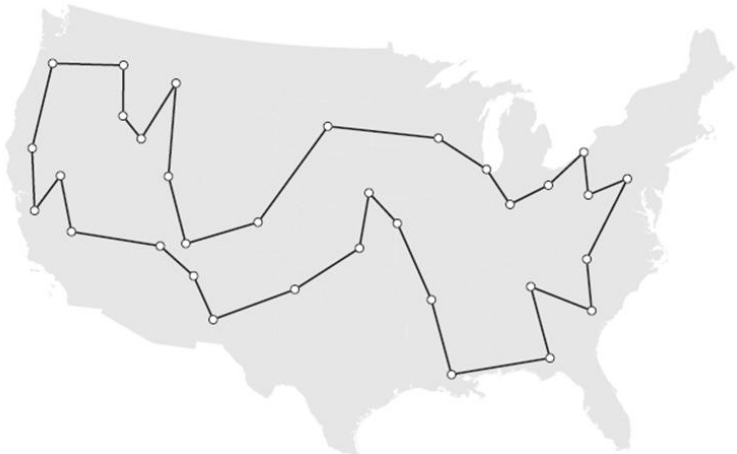
```
1: k = 1;
2: U = V;
3: while U ≠ ∅ do
4:    C_k = maximalCI(G[U]);
5:    U ← U ∖ C_k;
6:    k ← k + 1;
7: end while
```

- Pior caso: $O(n)$ (dado por um grafo completo ou um grafo sem arestas)

O Problema do Caixeiro Viajante

- A história da PO se confunde com a história do problema do caixeiro viajante.
- O problema é simples:

  Dado um conjunto finito de cidades juntamente com o custo de viagem entre cada par de cidades, encontre o modo mais barato de visitar todas as cidades uma única vez retornando ao ponto de partida.



Explosão Combinatória

- O número de soluções do problema do caixeiro viajante é $O(n-1)!$, onde n é o número de cidades.

n	$(n-1)!$	Tempo de CPU
5	24	insignificante
10	362.880	0.003 s
15	87 bilhões	20 min
20	1.2×10^{17}	73 anos
25	6.2×10^{23}	470 milhões de anos

Soluções exatas - milestones

1954 49 cidades

1971 64 cidades

1975 67 cidades

1977 120 cidades

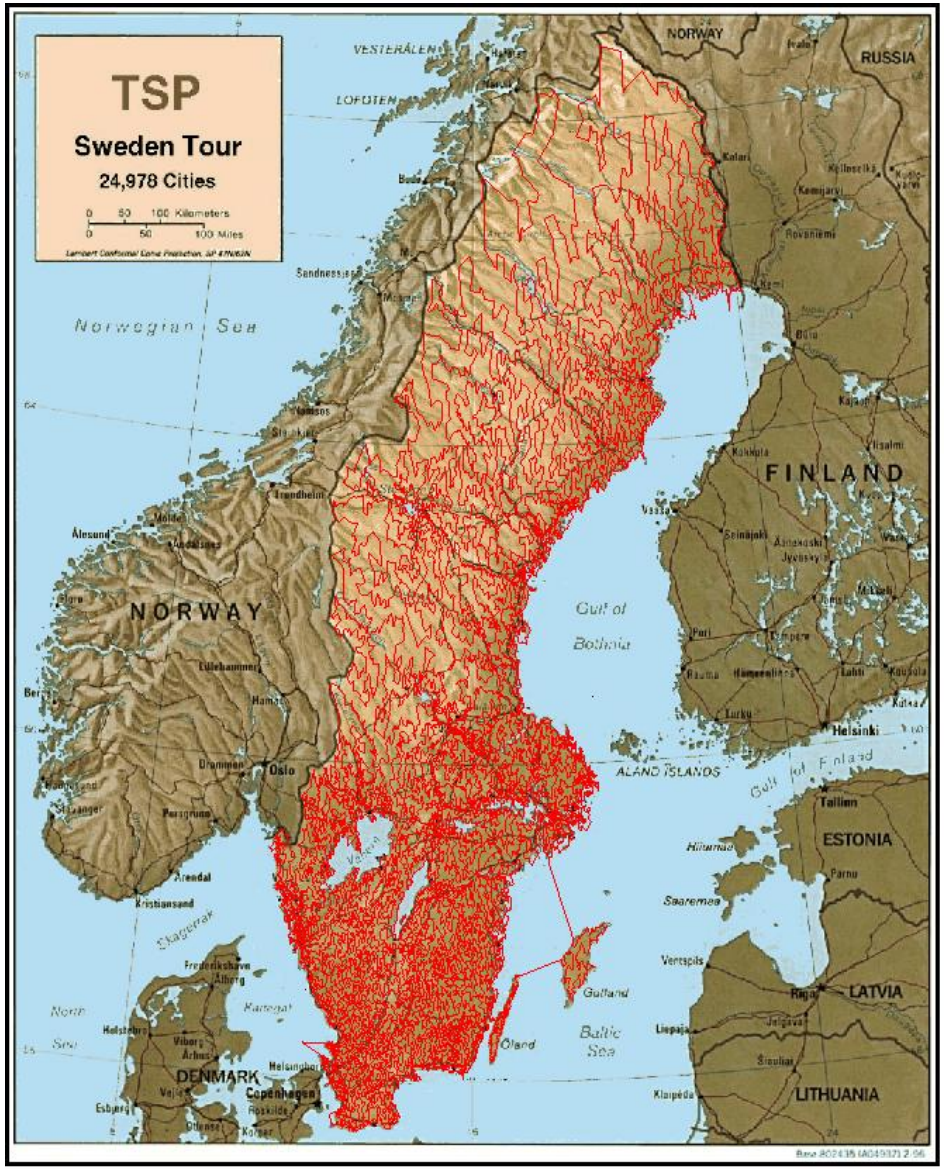
1980 318 cidades

1987 2392 cidades

1994 7397 cidades

1998 13509 cidades

2004 24978 cidades



Soluções heurísticas

Esta solução está no máximo a 0.0477% da solução ótima.

Algumas respostas...

Estes progressos foram possíveis devido ao:

- Avanço tecnológico dos computadores.
- Avanço da Pesquisa Operacional.

Fim do Módulo 8