Uma introdução à complexidade parametrizada

Vinicius Fernandes dos Santos - CEFET-MG

Uéverton dos Santos Souza - UFF/CEFET-RJ

34º JAI - Jornadas de Atualização em Informática
XXXV Congresso da Sociedade Brasileira de Computação
Recife, Julho 20–23, 2015

Objetivos

- Mundo ideal: conseguimos resolver todos os problemas de forma eficiente

Objetivos

- Mundo ideal: conseguimos resolver todos os problemas de forma eficiente (tempo polinomial).

Objetivos

- Mundo ideal: conseguimos resolver todos os problemas de forma eficiente (tempo polinomial).
- Mundo real: precisamos resolver problemas, mesmo os mais difíceis

Objetivos

- Mundo ideal: conseguimos resolver todos os problemas de forma eficiente (tempo polinomial).
- Mundo real: precisamos resolver problemas, mesmo os mais difíceis (NP-difíceis).

Objetivos

- Mundo ideal: conseguimos resolver todos os problemas de forma eficiente (tempo polinomial).
- Mundo real: precisamos resolver problemas, mesmo os mais difíceis (NP-difíceis).
- Em alguns casos, só conhecemos algoritmos exponenciais

Objetivos

- Mundo ideal: conseguimos resolver todos os problemas de forma eficiente (tempo polinomial).
- Mundo real: precisamos resolver problemas, mesmo os mais difíceis (NP-difíceis).
- Em alguns casos, só conhecemos algoritmos exponenciais (complexidades com variáveis no expoente).

Objetivos

- Mundo ideal: conseguimos resolver todos os problemas de forma eficiente (tempo polinomial).
- Mundo real: precisamos resolver problemas, mesmo os mais difíceis (NP-difíceis).
- Em alguns casos, só conhecemos algoritmos exponenciais (complexidades com variáveis no expoente).
- Que tal um algoritmo “pouco” exponencial?

Objetivos

- Mundo ideal: conseguimos resolver todos os problemas de forma eficiente (tempo polinomial).
- Mundo real: precisamos resolver problemas, mesmo os mais difíceis (NP-difíceis).
- Em alguns casos, só conhecemos algoritmos exponenciais (complexidades com variáveis no expoente).
- Que tal um algoritmo “pouco” exponencial?
  - Exponencial em algum valor tipicamente pequeno, ao invés do tamanho da entrada.

Objetivos

- Algoritmos parametrizados são mais uma ferramenta na caixa de ferramentas do resolvedor de problemas:
  - Algoritmos gulosos.

Objetivos

- Algoritmos parametrizados são mais uma ferramenta na caixa de ferramentas do resolvedor de problemas:
  - Algoritmos gulosos.
  - Divisão e conquista.

Objetivos

- Algoritmos parametrizados são mais uma ferramenta na caixa de ferramentas do resolvedor de problemas:
  - Algoritmos gulosos.
  - Divisão e conquista.
  - Programação dinâmica.

Objetivos

- Algoritmos parametrizados são mais uma ferramenta na caixa de ferramentas do resolvedor de problemas:
  - Algoritmos gulosos.
  - Divisão e conquista.
  - Programação dinâmica.
  - Backtracking.

Objetivos

- Algoritmos parametrizados são mais uma ferramenta na caixa de ferramentas do resolvedor de problemas:
  - Algoritmos gulosos.
  - Divisão e conquista.
  - Programação dinâmica.
  - Backtracking.
  - Branch & bound.

Objetivos

- Algoritmos parametrizados são mais uma ferramenta na caixa de ferramentas do resolvedor de problemas:
  - Algoritmos gulosos.
  - Divisão e conquista.
  - Programação dinâmica.
  - Backtracking.
  - Branch & bound.
  - Algoritmos aproximativos.

Objetivos

- Algoritmos parametrizados são mais uma ferramenta na caixa de ferramentas do resolvedor de problemas:
  - Algoritmos gulosos.
  - Divisão e conquista.
  - Programação dinâmica.
  - Backtracking.
  - Branch & bound.
  - Algoritmos aproximativos.
  - Algoritmos randomizados.

Objetivos

- Algoritmos parametrizados são mais uma ferramenta na caixa de ferramentas do resolvedor de problemas:
  - Algoritmos gulosos.
  - Divisão e conquista.
  - Programação dinâmica.
  - Backtracking.
  - Branch & bound.
  - Algoritmos aproximativos.
  - Algoritmos randomizados.
  - Algoritmos genéticos.

Objetivos

- Algoritmos parametrizados são mais uma ferramenta na caixa de ferramentas do resolvedor de problemas:
  - Algoritmos gulosos.
  - Divisão e conquista.
  - Programação dinâmica.
  - Backtracking.
  - Branch & bound.
  - Algoritmos aproximativos.
  - Algoritmos randomizados.
  - Algoritmos genéticos.
  - **Algoritmos parametrizados**.

Objetivos

- Algoritmos parametrizados são mais uma ferramenta na caixa de ferramentas do resolvedor de problemas:
  - Algoritmos gulosos.
  - Divisão e conquista.
  - Programação dinâmica.
  - Backtracking.
  - Branch & bound.
  - Algoritmos aproximativos.
  - Algoritmos randomizados.
  - Algoritmos genéticos.
  - **Algoritmos parametrizados**.

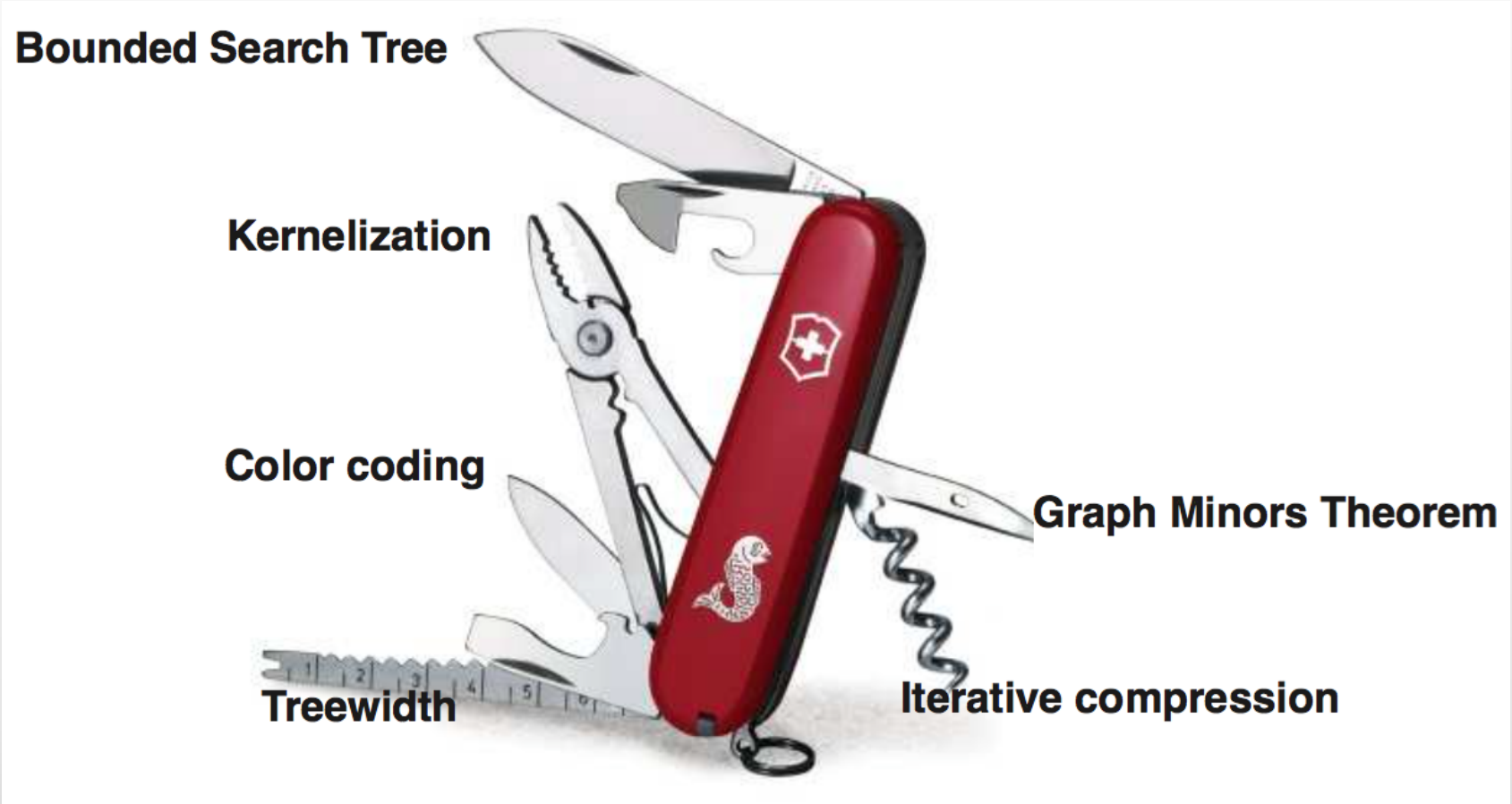
Tempo \ Resposta	Exato	Flexível
Exato		
Flexível		

Objetivos

- Algoritmos parametrizados são mais uma ferramenta na caixa de ferramentas do resolvedor de problemas:
  - Algoritmos gulosos.
  - Divisão e conquista.
  - Programação dinâmica.
  - Backtracking.
  - Branch & bound.
  - Algoritmos aproximativos.
  - Algoritmos randomizados.
  - Algoritmos genéticos.
  - **Algoritmos parametrizados**.

Tempo \ Resposta	Exato	Flexível
Exato	✓	
Flexível		

Complexidade parametrizada



Motivação teórica e pratica

- Interesse teórico:
  - Refinamento da dificuldade dos problemas.

Motivação teórica e pratica

- Interesse teórico:
  - Refinamento da dificuldade dos problemas.
  - Antes: Vários problemas eram NP-completos, mas alguns pareciam mais fáceis que outros.

Motivação teórica e pratica

- Interesse teórico:
  - Refinamento da dificuldade dos problemas.
  - Antes: Vários problemas eram NP-completos, mas alguns pareciam mais fáceis que outros.
  - Depois: Embora só conheçamos algoritmos exponenciais para problemas NP-difíceis, alguns são “menos exponenciais” que outros.

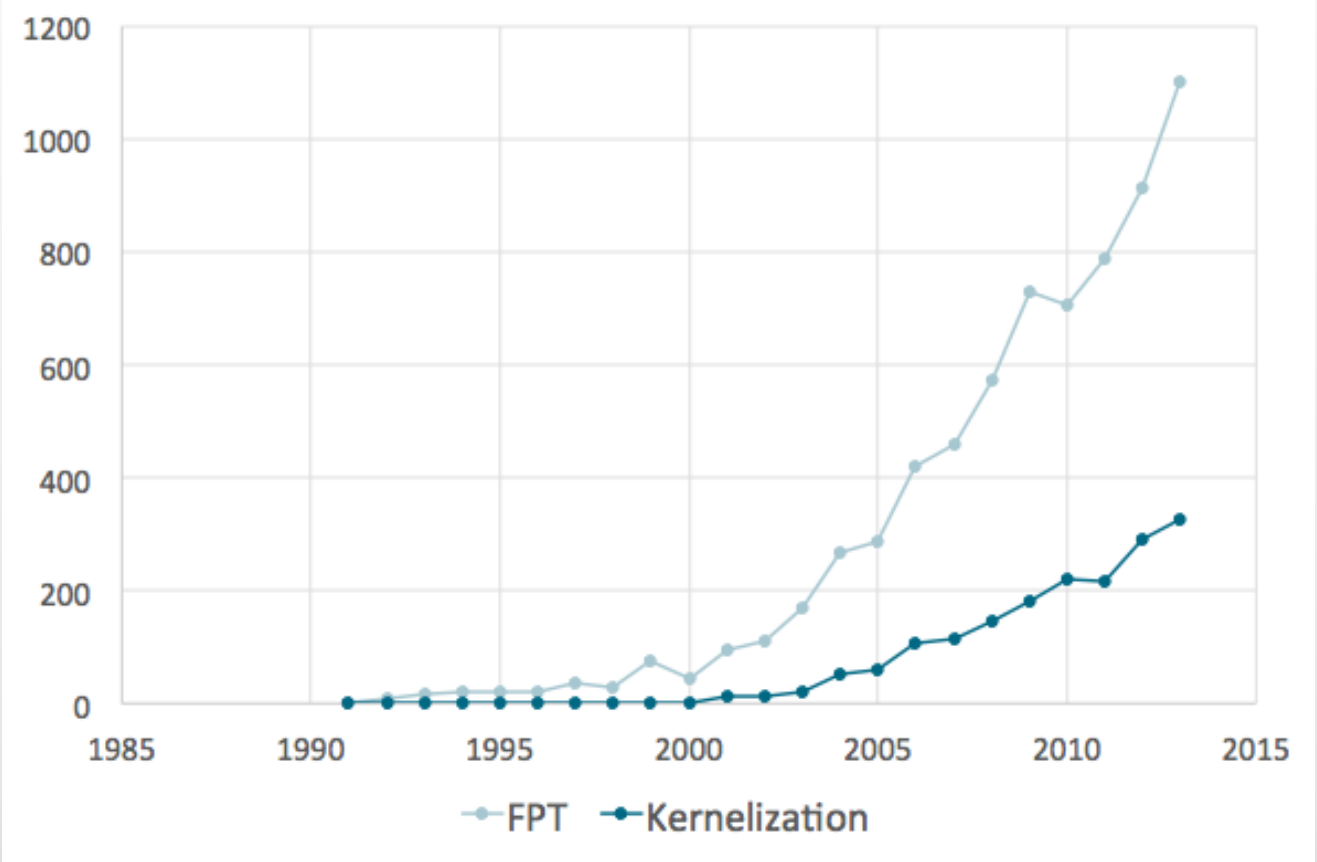
Motivação teórica e pratica

- Interesse teórico:
  - Refinamento da dificuldade dos problemas.
  - Antes: Vários problemas eram NP-completos, mas alguns pareciam mais fáceis que outros.
  - Depois: Embora só conheçamos algoritmos exponenciais para problemas NP-difíceis, alguns são “menos exponenciais” que outros.
  - Muito espaço para novas pesquisas.

Motivação teórica e pratica

- Interesse teórico:
  - Refinamento da dificuldade dos problemas.
  - Antes: Vários problemas eram NP-completos, mas alguns pareciam mais fáceis que outros.
  - Depois: Embora só conheçamos algoritmos exponenciais para problemas NP-difíceis, alguns são “menos exponenciais” que outros.
  - Muito espaço para novas pesquisas.
- Interesse prático: quando problemas reais precisam ser resolvidos exatamente, qualquer melhoria é significativa, principalmente em algoritmos exponenciais.

Artigos sobre complexidade parametrizada



Árvores de altura limitada

- Muitos problemas possuem uma solução simples de força bruta.

Árvores de altura limitada

- Muitos problemas possuem uma solução simples de força bruta.
- Algoritmos de força bruta podem tipicamente ser implementados recursivamente.

Árvores de altura limitada

- Muitos problemas possuem uma solução simples de força bruta.
- Algoritmos de força bruta podem tipicamente ser implementados recursivamente.
- Consideremos um algoritmo de força bruta para clique máxima.

Árvores de altura limitada

- Muitos problemas possuem uma solução simples de força bruta.
- Algoritmos de força bruta podem tipicamente ser implementados recursivamente.
- Consideremos um algoritmo de força bruta para clique máxima.
- Qual a complexidade do algoritmo?

Deleção de triângulos

Deleção de triângulos

Instância: Um grafo $G = (V, E)$.

Parâmetro: Um inteiro positivo k.

Questão: É possível deletar todos os triângulos de G removendo no máximo k arestas?

- Ideia mais ingênua: testar todos os $\binom{|E(G)|}{k}$ conjuntos de k arestas.

Deleção de triângulos

Deleção de triângulos

Instância: Um grafo $G = (V, E)$.

Parâmetro: Um inteiro positivo k.

Questão: É possível deletar todos os triângulos de G removendo no máximo k arestas?

- Ideia mais ingênua: testar todos os $\binom{|E(G)|}{k}$ conjuntos de k arestas.
- Complexidade $O(|E(G)|^k)$.

Deleção de triângulos

Deleção de triângulos

Instância: Um grafo $G = (V, E)$.

Parâmetro: Um inteiro positivo k.

Questão: É possível deletar todos os triângulos de G removendo no máximo k arestas?

- Ideia mais ingênua: testar todos os $\binom{|E(G)|}{k}$ conjuntos de k arestas.
- Complexidade $O(|E(G)|^k)$.
- Observação interessante: para cada triângulo do grafo, ao menos uma de suas arestas deve ser removida.

Deleção de triângulos

Deleção de triângulos

Instância: Um grafo $G = (V, E)$.

Parâmetro: Um inteiro positivo k.

Questão: É possível deletar todos os triângulos de G removendo no máximo k arestas?

- Ideia mais ingênua: testar todos os $\binom{|E(G)|}{k}$ conjuntos de k arestas.
- Complexidade $O(|E(G)|^k)$.
- Observação interessante: para cada triângulo do grafo, ao menos uma de suas arestas deve ser removida.

Deleção de triângulos

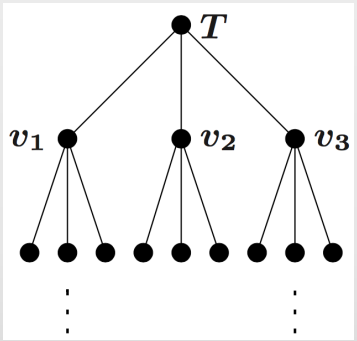
Deleção de triângulos

Instância: Um grafo $G = (V, E)$.

Parâmetro: Um inteiro positivo k.

Questão: É possível deletar todos os triângulos de G removendo no máximo k arestas?

- Ideia mais ingênua: testar todos os $\binom{|E(G)|}{k}$ conjuntos de k arestas.
- Complexidade $O(|E(G)|^k)$.
- Observação interessante: para cada triângulo do grafo, ao menos uma de suas arestas deve ser removida.



Deleção de triângulos

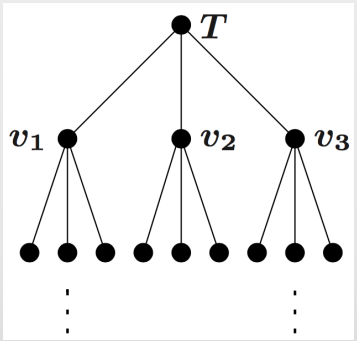
Deleção de triângulos

Instância: Um grafo $G = (V, E)$.

Parâmetro: Um inteiro positivo k.

Questão: É possível deletar todos os triângulos de G removendo no máximo k arestas?

- Ideia mais ingênua: testar todos os $\binom{|E(G)|}{k}$ conjuntos de k arestas.
- Complexidade $O(|E(G)|^k)$.
- Observação interessante: para cada triângulo do grafo, ao menos uma de suas arestas deve ser removida.



Complexidade: $O(n^{O(1)}3^k)$.

Cobertura de vértices

Cobertura por Vértices(k)
Instância: Um grafo $G = (V, E)$.
Parâmetro: Um inteiro positivo k.
Questão: G possui um conjunto de vértices I, tal que $|I| \leq k$ e toda aresta de G possui pelo menos um de seus extremos em I?

Cobertura de vértices

Cobertura por Vértices(k)
Instância: Um grafo $G = (V, E)$.
Parâmetro: Um inteiro positivo k.
Questão: G possui um conjunto de vértices I, tal que $|I| \leq k$ e toda aresta de G possui pelo menos um de seus extremos em I?

- Como resolver usando força bruta?

Cobertura de vértices

- Uma observação importante: para cada aresta, pelo menos uma das extremidades deve estar na solução.

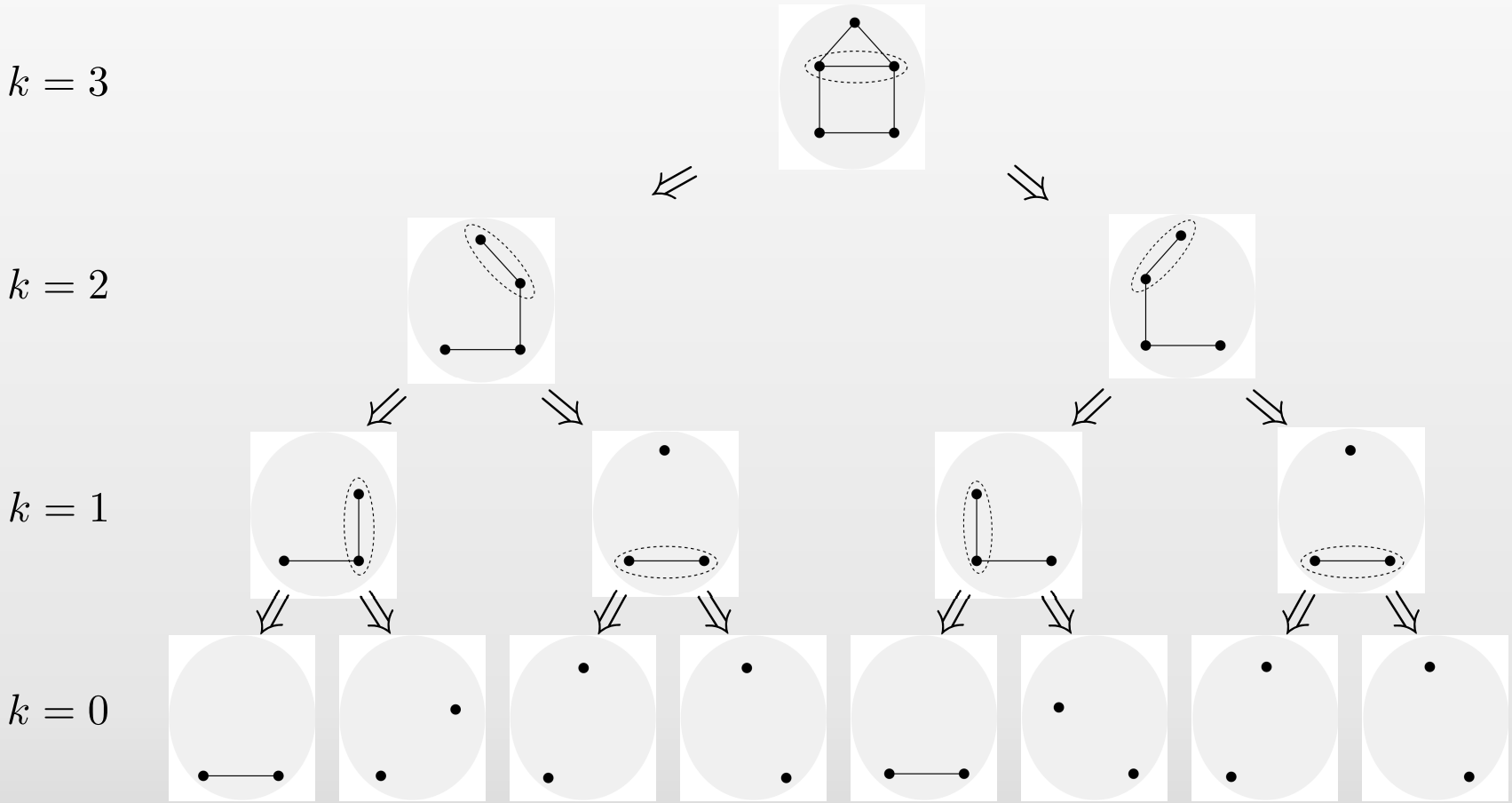
Cobertura de vértices

Algorithm 1: Primeiro algoritmo para cobertura por vértices.

```
1  CoberturaDeVertices (G, k):
2  se G não possui arestas então
3  |  retorna SIM
4  fim
5  se k = 0 então
6  |  retorna NÃO
7  fim
8  Escolha uma aresta uv arbitrariamente
9  se CoberturaDeVertices (G − {u}, k − 1) retornar SIM então
10 |  retorna SIM
11 fim
12 se CoberturaDeVertices (G − {v}, k − 1) retornar SIM então
13 |  retorna SIM
14 fim
15 retorna NÃO
```

Complexidade: $O(n^{O(1)}2^k)$

Cobertura de vértices



Cobertura de vértices

- Podemos utilizar conhecimento sobre o problema: se um vértice tem pelo menos uma aresta incidente a ele, então ou o vértice faz parte da cobertura, ou todos os seus vizinhos fazem parte da cobertura.

Cobertura de vértices

Algorithm 2: Segundo algoritmo para cobertura por vértices.

```
1  CoberturaDeVertices (G, k):
2  se Δ(G) < 2 então
3  |  se |E(G)| ≤ k então
4  |  |  retorna SIM
5  |  senão
6  |  |  retorna NÃO
7  |  fim
8  fim
9  se k = 0 então
10 |  retorna NÃO
11 fim
12 Seja v um vértice de grau Δ(G)
13 se CoberturaDeVertices (G − {v}, k − 1) retornar SIM então
14 |  retorna SIM
15 fim
16 se Δ(G) ≤ k então
17 |  se CoberturaDeVertices (G − {N(v)}, k − Δ(G)) retornar SIM
   |  então
18 |  |  retorna SIM
19 |  fim
20 fim
21 retorna NÃO
```

Complexidade: $O(n^{O(1)}1,6181^k)$

Cobertura de vértices

- Podemos utilizar **ainda mais** conhecimento sobre o problema: se o grau máximo é 2, então é possível resolver o problema em tempo polinomial.

Cobertura de vértices

Algorithm 3: Terceiro algoritmo para cobertura por vértices.

```
1  CoberturaDeVertices (G, k):
2  se Δ(G) ≤ 2 então
3      se Se existe cobertura de tamanho ≤ k então
4          retorna SIM
5      senão
6          retorna NÃO
7      fim
8  fim
9  se k = 0 então
10     retorna NÃO
11 fim
12 Seja v um vértice de grau Δ(G)
13 se CoberturaDeVertices (G − {v}, k − 1) retornar SIM então
14     retorna SIM
15 fim
16 se Δ(G) ≤ k então
17     se CoberturaDeVertices (G − {N(v)}, k − Δ(G)) retornar SIM
       então
18         retorna SIM
19     fim
20 fim
21 retorna NÃO
```

Complexidade: $O(n^{O(1)}1,4656^k)$

Cobertura de vértices

Algoritmo	$k = 20$	$k = 30$	$k = 40$
Algoritmo 1	1048576	1073741824	1099511627776
Algoritmo 2	15139	1862776	229199843
Algoritmo 3	2091	95601	4371377

Table: Número de chamadas à função, em cada uma das versões do algoritmo.

Cadeia mais próxima

Achar uma cadeia de caracteres que seja "próxima" de todas as cadeias de um conjunto.

Cadeia mais próxima parametrizada pela distância

Instância: Um conjunto S de k cadeias de caracteres, de comprimento ℓ.

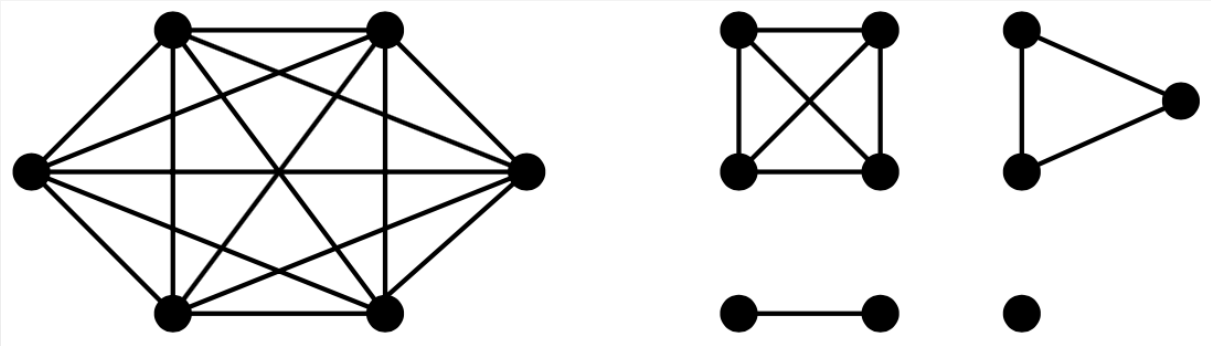
Parâmetro: Um inteiro positivo d.

Questão: Existe uma cadeia s tal que a distância de s para qualquer elemento de S é no máximo d?

Algoritmo parametrizado de complexidade $O(|S|(d+1)^d)$

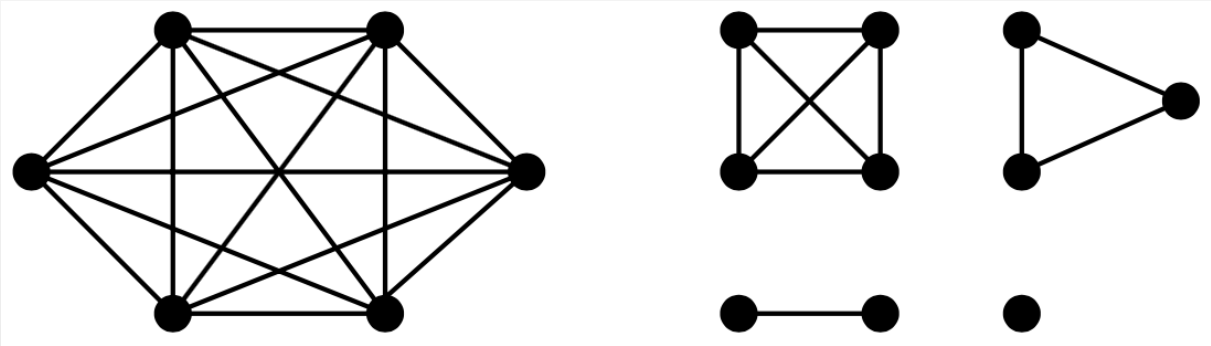
Cluster editing

- Problema consiste em adicionar adicionar ou remover arestas, de forma que cada componente conexa se torne um grafo completo.

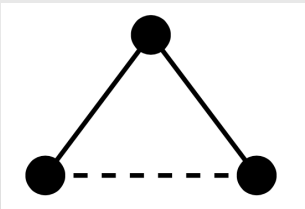


Cluster editing

- Problema consiste em adicionar adicionar ou remover arestas, de forma que cada componente conexa se torne um grafo completo.



- Usando raciocínio análogo à deleção de triângulos, o que precisamos fazer é evitar impedir a existência deste subgrafo:



Kernelização

Kernelização

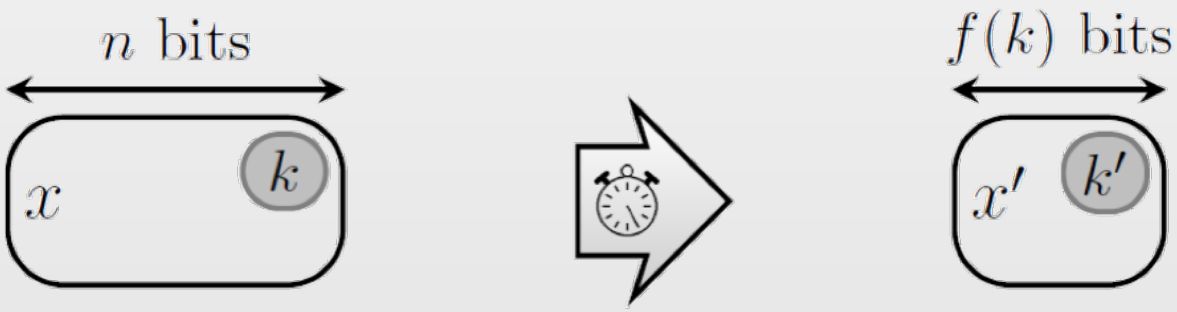
- Pré-processamento é utilizado em computação nas mais diversas áreas.

Kernelização

- Pré-processamento é utilizado em computação nas mais diversas áreas.
- Exemplo: força bruta para clique máxima.

Kernelização

- Pré-processamento é utilizado em computação nas mais diversas áreas.
- Exemplo: força bruta para clique máxima.
- Kernelização consiste em um conjunto de regras cuja aplicação garantem redução do tamanho da instância.



Kernelização

- **Definição:** Um algoritmo de kernelização para um problema Π consiste em um algoritmo que transforma um instância (x, k) em uma instância (x', k') em tempo polinomial em $|x|$, de forma que a resposta para as instâncias (x, k) e (x', k') são as mesmas, e existe uma função g tal que $x' \leq g(k)$ e $k' \leq g(k)$.

Kernelização

- **Definição:** Um algoritmo de kernelização para um problema Π consiste em um algoritmo que transforma um instância (x, k) em uma instância (x', k') em tempo polinomial em $|x|$, de forma que a resposta para as instâncias (x, k) e (x', k') são as mesmas, e existe uma função g tal que $x' \leq g(k)$ e $k' \leq g(k)$.
- Dizemos que (x', k') é o núcleo (**kernel**) da instância (x, k).

Kernelização

- **Definição:** Um algoritmo de kernelização para um problema Π consiste em um algoritmo que transforma um instância (x, k) em uma instância (x', k') em tempo polinomial em $|x|$, de forma que a resposta para as instâncias (x, k) e (x', k') são as mesmas, e existe uma função g tal que $x' \leq g(k)$ e $k' \leq g(k)$.
- Dizemos que (x', k') é o núcleo (**kernel**) da instância (x, k).
- Por que estudar kernelização?

Kernelização

- **Definição:** Um algoritmo de kernelização para um problema Π consiste em um algoritmo que transforma um instância (x, k) em uma instância (x', k') em tempo polinomial em $|x|$, de forma que a resposta para as instâncias (x, k) e (x', k') são as mesmas, e existe uma função g tal que $x' \leq g(k)$ e $k' \leq g(k)$.
- Dizemos que (x', k') é o núcleo (**kernel**) da instância (x, k).
- Por que estudar kernelização?

> ***Teorema***
>
> *Um problema Π admite um algoritmo de parâmetro fixo se e somente se ele admite um núcleo.*

Kernelização

- Algoritmos de redução a um núcleo se baseiam na aplicação de **regras de redução**.

Kernelização

- Algoritmos de redução a um núcleo se baseiam na aplicação de **regras de redução**.
- Em geral, regras de redução são aplicadas em uma ordem pré-determinada.

Kernelização

- Algoritmos de redução a um núcleo se baseiam na aplicação de **regras de redução**.
- Em geral, regras de redução são aplicadas em uma ordem pré-determinada.
- Algumas regras de redução podem já concluir a resposta para instância.

Kernelização

- Algoritmos de redução a um núcleo se baseiam na aplicação de **regras de redução**.
- Em geral, regras de redução são aplicadas em uma ordem pré-determinada.
- Algumas regras de redução podem já concluir a resposta para instância.
- Quando nenhuma regra de redução puder ser aplicada, temos em mãos um núcleo.

Cobertura de vértices

Regra de redução

Se (G, k) é uma instância do problema de cobertura de vértices parametrizado e o grafo G contém um vértice isolado v, então remova v de G, obtendo uma nova instância $(G - v, k)$.

Cobertura de vértices

Regra de redução

Se (G, k) é uma instância do problema de cobertura de vértices parametrizado e o grafo G contém um vértice isolado v, então remova v de G, obtendo uma nova instância $(G - v, k)$.

Regra de redução

Se (G, k) é uma instância do problema de cobertura de vértices parametrizado e o grafo G contém um vértice v com $d(v) \geq k + 1$, então remova v de G e reduza uma unidade de k, obtendo uma nova instância $(G - v, k - 1)$.

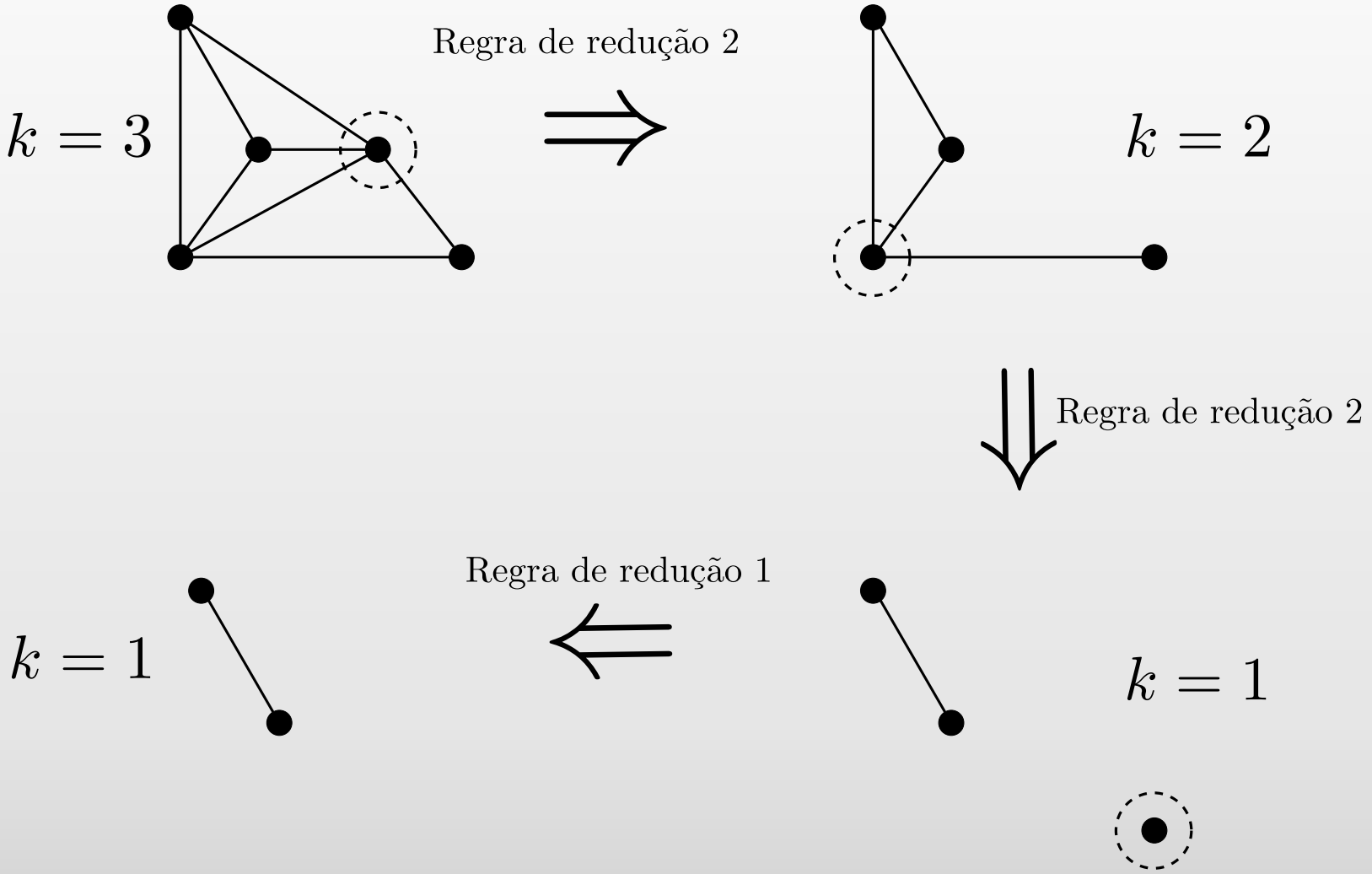
Cobertura de vértices

Regra de redução

Se $I = (G, k)$ é uma instância do problema de cobertura de vértices parametrizado tal que as Regras de redução 1 e 2 não podem ser mais aplicadas e alguma das condições abaixo é satisfeita, então I é uma instância Não.

- $k < 0$, *ou*
- G *possui mais de* $k^2 + k$ *vértices, ou*
- G *possui mais de* k^2 *arestas.*

Cobertura de vértices



Cobertura de vértices

Teorema

O problema de cobertura de vértices admite um núcleo com no máximo $O(k^2)$ vértices e $O(k^2)$ arestas.

Conjunto de arestas de retroalimentação em torneios

- Um torneio é um grafo direcionado completo.

Conjunto de arestas de retroalimentação em torneios

- Um torneio é um grafo direcionado completo.
- Um conjunto de arestas de retroalimentação é um conjunto de arestas cuja remoção torna o grafo acíclico.

Conjunto de arestas de retroalimentação em torneios

- Um torneio é um grafo direcionado completo.
- Um conjunto de arestas de retroalimentação é um conjunto de arestas cuja remoção torna o grafo acíclico.
- Não podemos remover arestas do grafo, pois estaríamos alterando o problema.

Conjunto de arestas de retroalimentação em torneios

- Um torneio é um grafo direcionado completo.
- Um conjunto de arestas de retroalimentação é um conjunto de arestas cuja remoção torna o grafo acíclico.
- Não podemos remover arestas do grafo, pois estaríamos alterando o problema.

Observação

Se G é um grafo direcionado e F é um subconjunto de arestas de G tal que a reversão de F deixa o grafo acíclico, então F é um conjunto de retroalimentação.

Conjunto de arestas de retroalimentação em torneios

- Um torneio é um grafo direcionado completo.
- Um conjunto de arestas de retroalimentação é um conjunto de arestas cuja remoção torna o grafo acíclico.
- Não podemos remover arestas do grafo, pois estaríamos alterando o problema.

Observação

Se G é um grafo direcionado e F é um subconjunto de arestas de G tal que a reversão de F deixa o grafo acíclico, então F é um conjunto de retroalimentação.

Teorema

Se G é um grafo direcionado e F é um subconjunto de arestas de G então F é um conjunto de retroalimentação minimal se e somente se F é um conjunto minimal cuja reversão torna o grafo acíclico.

Conjunto de arestas de retroalimentação em torneios

Regra de redução

Se (G, k) é uma instância para o problema do conjunto de arestas de retroalimentação em torneio e f é uma aresta pertencente a pelo menos $k + 1$ triângulos, então podemos reverter f e diminuir k em uma unidade.

Conjunto de arestas de retroalimentação em torneios

Regra de redução

Se (G, k) é uma instância para o problema do conjunto de arestas de retroalimentação em torneio e f é uma aresta pertencente a pelo menos $k + 1$ triângulos, então podemos reverter f e diminuir k em uma unidade.

Regra de redução

Se (G, k) é uma instância para o problema do conjunto de arestas de retroalimentação em torneio e v é um vértice que não faz parte de nenhum triângulo, então podemos removê-lo de G.

Conjunto de arestas de retroalimentação em torneios

Regra de redução

Se (G, k) é uma instância para o problema do conjunto de arestas de retroalimentação em torneio e f é uma aresta pertencente a pelo menos $k + 1$ triângulos, então podemos reverter f e diminuir k em uma unidade.

Regra de redução

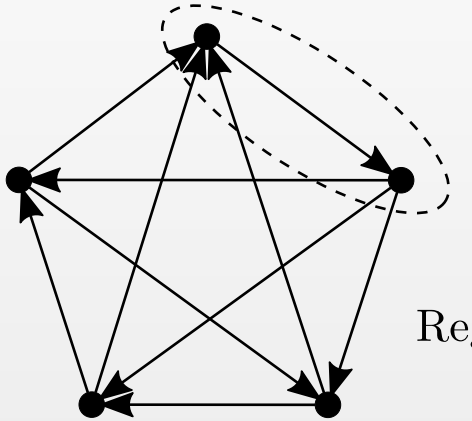
Se (G, k) é uma instância para o problema do conjunto de arestas de retroalimentação em torneio e v é um vértice que não faz parte de nenhum triângulo, então podemos removê-lo de G.

Regra de redução

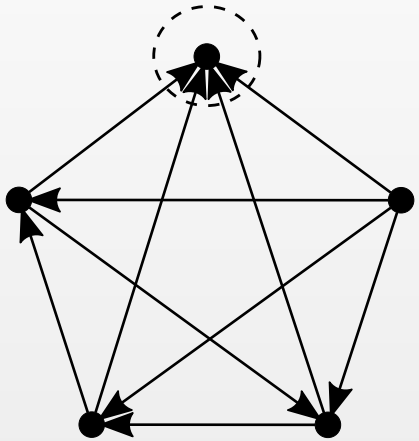
Se (G, k) é uma instância para o problema do conjunto de arestas de retroalimentação em torneio em que as regras anteriores não podem ser aplicadas, então, se G possui mais de $k(k + 2)$ vértices, então (G, k) é uma instância Não.

Conjunto de arestas de retroalimentação em torneios

$k = 2$



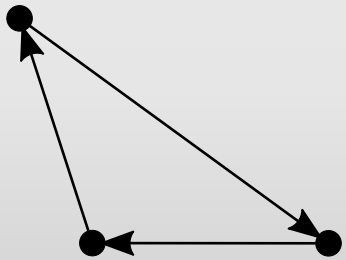
$\Rightarrow$ Regra de Redução 4



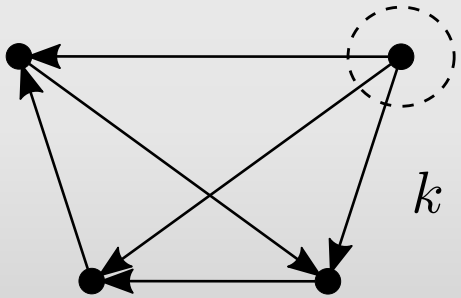
$k = 1$

$\Downarrow$ Regra de Redução 5

$k = 1$



$\Leftarrow$ Regra de Redução 5



$k = 1$

Conjunto de arestas de retroalimentação em torneios

Teorema

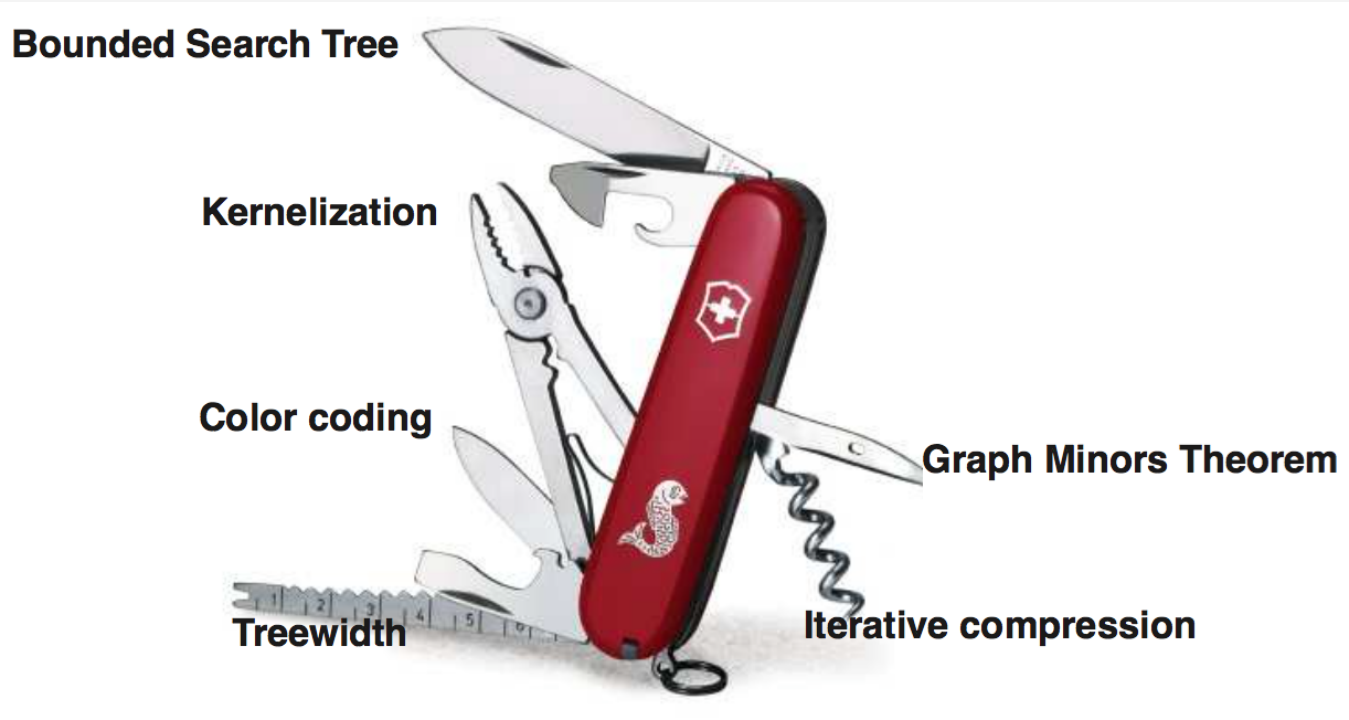
O problema do conjunto de arestas de retroalimentação em torneios possui um núcleo com no máximo $k^2 + 2k$ vértices.

Conclusão parcial

Temos algumas técnicas interessantes para mostrar a tratabilidade de problemas por parâmetro fixo.

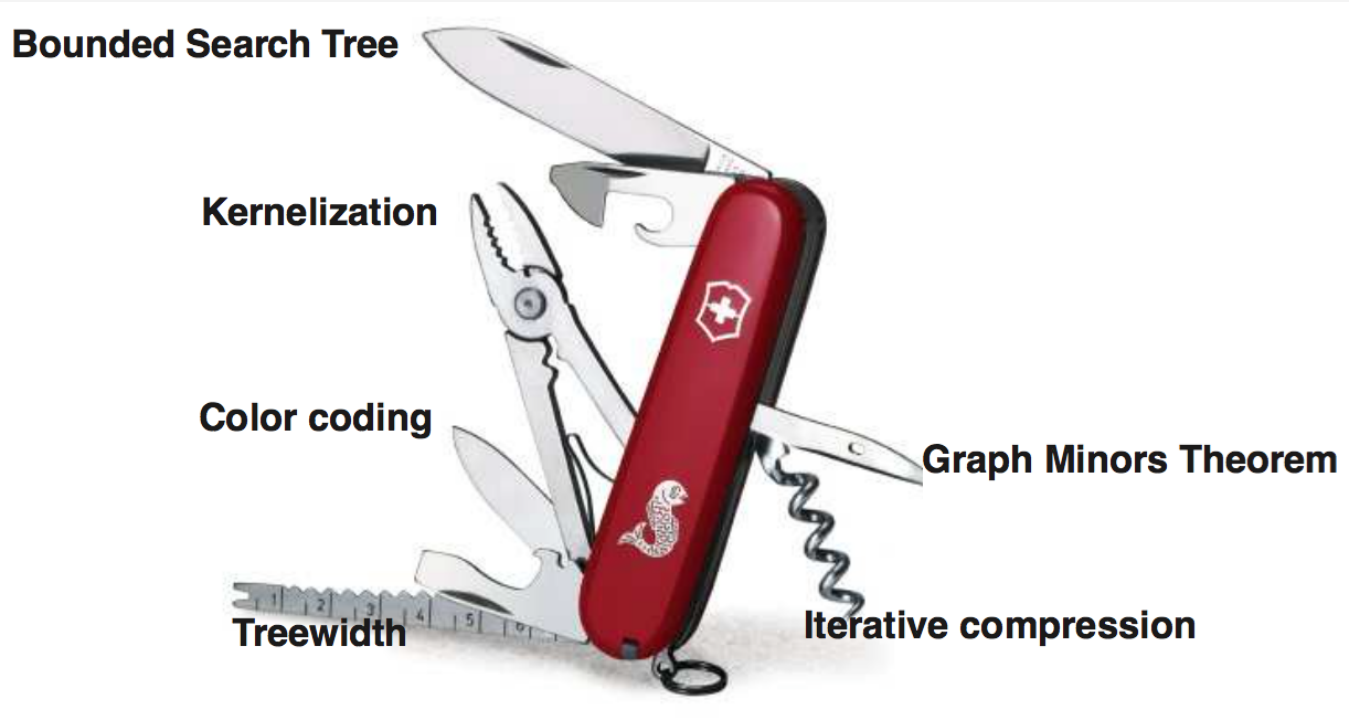
Conclusão parcial

Temos algumas técnicas interessantes para mostrar a tratabilidade de problemas por parâmetro fixo.



Conclusão parcial

Temos algumas técnicas interessantes para mostrar a tratabilidade de problemas por parâmetro fixo.



Mas será que todos os problemas são tratáveis por parâmetro fixo?