MAC 414
Autômatos, Computabilidade e Complexidade
aula 20 — 30/11/2020

Que acontece se P = NP for resolvido?

Que acontece se P = NP for resolvido?

Alguém ganha 1 milhão de dólares.

Que acontece se P = NP for resolvido?

Alguém ganha 1 milhão de dólares.
Aí, depende:

- Se for provado que **P** $\neq$ **NP**, muda pouco em relação ao que se faz hoje.

Que acontece se P = NP for resolvido?

Alguém ganha 1 milhão de dólares.

Aí, depende:

- Se for provado que $\mathbf{P} \neq \mathbf{NP}$, muda pouco em relação ao que se faz hoje.
- Se for provado que $\mathbf{P} = \mathbf{NP}$, tem algumas possibilidades:

Que acontece se P = NP for resolvido?

Alguém ganha 1 milhão de dólares.

Aí, depende:

- Se for provado que **P** ≠ **NP**, muda pouco em relação ao que se faz hoje.
- Se for provado que **P** = **NP**, tem algumas possibilidades:
  - Se for mostrado um algoritmo de alta complexidade para um problema **NP**-completo, continua meio na mesma.

Que acontece se P = NP for resolvido?

Alguém ganha 1 milhão de dólares.

Aí, depende:

- Se for provado que **P** ≠ **NP**, muda pouco em relação ao que se faz hoje.
- Se for provado que **P** = **NP**, tem algumas possibilidades:
  - Se for mostrado um algoritmo de alta complexidade para um problema **NP**-completo, continua meio na mesma.

Que acontece se P = NP for resolvido?

Alguém ganha 1 milhão de dólares.
Aí, depende:

- Se for provado que **P** ≠ **NP**, muda pouco em relação ao que se faz hoje.
- Se for provado que **P** = **NP**, tem algumas possibilidades:
  - Se for mostrado um algoritmo de alta complexidade para um problema **NP**-completo, continua meio na mesma. Surgem duas frentes de trabalho: melhorar a complexidade ou provar um limite inferior.

Que acontece se P = NP for resolvido?

Alguém ganha 1 milhão de dólares.
Aí, depende:

- Se for provado que **P** ≠ **NP**, muda pouco em relação ao que se faz hoje.
- Se for provado que **P** = **NP**, tem algumas possibilidades:
  - Se for mostrado um algoritmo de alta complexidade para um problema **NP**-completo, continua meio na mesma. Surgem duas frentes de trabalho: melhorar a complexidade ou provar um limite inferior.
  - Se for mostrado um algoritmo prático para Programação Inteira, o mundo muda!

O que existe além de NP-completo?

- #-P-completo.

O que existe além de NP-completo?

- #-P-completo.
- PSPACE

O que existe além de NP-completo?

- #-P-completo.
- PSPACE
- Outras formas de avaliar algoritmos.

O que existe além de NP-completo?

- #-P-completo.
- PSPACE
- Outras formas de avaliar algoritmos.
- ...

Problemas de contagem

Problemas de contagem

Caminho mínimo: Dado um grafo dirigido acíclico, com pesos nas arestas, vértices s, t, encontrar um caminho mínimo de s a t.

Problemas de contagem

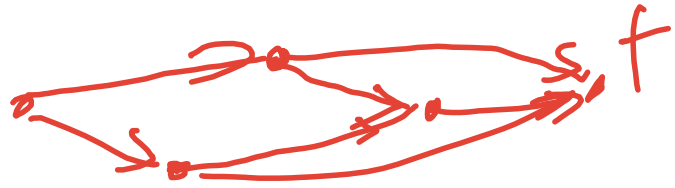
Caminho mínimo: Dado um grafo dirigido acíclico, com pesos nas arestas, vértices s, t, encontrar um caminho mínimo de s a t.

Um algoritmo:

1. Faça uma ordenação topológica no grafo

Problemas de contagem

Caminho mínimo: Dado um grafo dirigido acíclico, com pesos nas arestas, vértices s, t, encontrar um caminho mínimo de s a t.



Um algoritmo:

1. Faça uma ordenação topológica no grafo
2. Voltando de t a s marque para cada vértice o comprimento do caminho mínimo dele a t, e informação para recuperar o caminho.

Problemas de contagem

Caminho mínimo: Dado um grafo dirigido acíclico, com pesos nas arestas, vértices s, t, encontrar um caminho mínimo de s a t.

Um algoritmo:

1. Faça uma ordenação topológica no grafo
2. Voltando de t a s marque para cada vértice o comprimento do caminho mínimo dele a t, e informação para recuperar o caminho.

Problemas de contagem

Caminho mínimo: Dado um grafo dirigido acíclico, com pesos nas arestas, vértices s, t, encontrar um caminho mínimo de s a t.

Um algoritmo:

1. Faça uma ordenação topológica no grafo
2. Voltando de t a s marque para cada vértice o comprimento do caminho mínimo dele a t, e informação para recuperar o caminho.

Número de caminhos mínimos: Dado um grafo dirigido acíclico, com pesos nas arestas, vértices s, t, encontrar o *número* de caminhos mínimos de s a t.

Problemas de contagem

Caminho mínimo: Dado um grafo dirigido acíclico, com pesos nas arestas, vértices s, t, encontrar um caminho mínimo de s a t.

Um algoritmo:

1. Faça uma ordenação topológica no grafo
2. Voltando de t a s marque para cada vértice o comprimento do caminho mínimo dele a t, e informação para recuperar o caminho.

Número de caminhos mínimos: Dado um grafo dirigido acíclico, com pesos nas arestas, vértices s, t, encontrar o *número* de caminhos mínimos de s a t.

Só adaptar o algoritmo acima.

Contagem de certificados

Contagem de certificados

Para um dado problema em **NP**, dada uma instância, encontrar o número de certificados.

Contagem de certificados

Para um dado problema em **NP**, dada uma instância, encontrar o número de certificados.

Ou: Dada uma MTND decidindo L, determinar, para cada $x \in L$, o número de caminhos de processamento que aceitam x.

Contagem de certificados

Para um dado problema em **NP**, dada uma instância, encontrar o número de certificados.

Ou: Dada uma MTND decidindo L, determinar, para cada $x \in L$, o número de caminhos de processamento que aceitam x.

- #-SAT: dada uma instância do SAT, encontrar o número de atribuições que satisfazem a fórmula.

Contagem de certificados

Para um dado problema em **NP**, dada uma instância, encontrar o número de certificados.

Ou: Dada uma MTND decidindo L, determinar, para cada $x \in L$, o número de caminhos de processamento que aceitam x.

- #-SAT: dada uma instância do SAT, encontrar o número de atribuições que satisfazem a fórmula.
- #-HAMILTONIANO: dado um grafo, encontrar o número de circuitos hamiltonianos.

Contagem de certificados

Para um dado problema em **NP**, dada uma instância, encontrar o número de certificados.

Ou: Dada uma MTND decidindo L, determinar, para cada $x \in L$, o número de caminhos de processamento que aceitam x.

- #-SAT: dada uma instância do SAT, encontrar o número de atribuições que satisfazem a fórmula.
- #-HAMILTONIANO: dado um grafo, encontrar o número de circuitos hamiltonianos.

Contagem de certificados

Para um dado problema em **NP**, dada uma instância, encontrar o número de certificados.

Ou: Dada uma MTND decidindo L, determinar, para cada $x \in L$, o número de caminhos de processamento que aceitam x.

- #-SAT: dada uma instância do SAT, encontrar o número de atribuições que satisfazem a fórmula.
- #-HAMILTONIANO: dado um grafo, encontrar o número de circuitos hamiltonianos.

Claro que se o original é **NP**-completo, a versão de contagem é **NP**-difícil.

Emparelhamentos

Emparelhamentos

Um emparelhamento perfeito em um grafo G é um conjunto M de arestas tais que

- Arestas de M não tem vértice em comum.

Emparelhamentos

Um emparelhamento perfeito em um grafo G é um conjunto M de arestas tais que

- Arestas de M não tem vértice em comum.
- Todo vértice de G é incidente a M.

Emparelhamentos

Um emparelhamento perfeito em um grafo G é um conjunto M de arestas tais que

- Arestas de M não tem vértice em comum.
- Todo vértice de G é incidente a M.

Emparelhamentos

Um emparelhamento perfeito em um grafo G é um conjunto M de arestas tais que

- Arestas de M não tem vértice em comum.
- Todo vértice de G é incidente a M.

Emparelhamento perfeito bipartido: dado um grafo bipartido, ele tem emparelhamento perfeito?

Emparelhamentos

Um emparelhamento perfeito em um grafo G é um conjunto M de arestas tais que

- Arestas de M não tem vértice em comum.
- Todo vértice de G é incidente a M.

Emparelhamento perfeito bipartido: dado um grafo bipartido, ele tem emparelhamento perfeito?

Problema clássico, um dos primeiros algoritmos polinomiais da teoria dos grafos (König, 1931).

Emparelhamentos

Um emparelhamento perfeito em um grafo G é um conjunto M de arestas tais que

- Arestas de M não tem vértice em comum.
- Todo vértice de G é incidente a M.

Emparelhamento perfeito bipartido: dado um grafo bipartido, ele tem emparelhamento perfeito?

Problema clássico, um dos primeiros algoritmos polinomiais da teoria dos grafos (König, 1931).

#-Emparelhamento: dado um grafo bipartido, determinar o número de emparelhamentos perfeitos.

Emparelhamentos

Um emparelhamento perfeito em um grafo G é um conjunto M de arestas tais que

- Arestas de M não tem vértice em comum.
- Todo vértice de G é incidente a M.

Emparelhamento perfeito bipartido: dado um grafo bipartido, ele tem emparelhamento perfeito?

Problema clássico, um dos primeiros algoritmos polinomiais da teoria dos grafos (König, 1931).

#-Emparelhamento: dado um grafo bipartido, determinar o número de emparelhamentos perfeitos.

Teorema (Valiant, 1979): #-Emparelhamento é **NP**-difícil.

Determinante X Permanente

Determinante X Permanente

Dada matriz quadrada $A_{n \times n}$ sobre um corpo.

Determinante X Permanente

Dada matriz quadrada $A_{n\times n}$ sobre um corpo.

Determinante:

$$\det(A) = \sum_{\pi\in S_n} \text{sinal}(\pi) a_{1,\pi(1)} a_{2,\pi(2)} \cdots a_{n,\pi(n)}.$$

$\text{sinal}(\pi) = 1$ se π é par, -1 se π é ímpar.

Determinante X Permanente

Dada matriz quadrada $A_{n\times n}$ sobre um corpo.

Determinante:

$$\det(A) = \sum_{\pi\in S_n} \text{sinal}(\pi) a_{1,\pi(1)} a_{2,\pi(2)} \cdots a_{n,\pi(n)}.$$

$$\text{sinal}(\pi) = 1 \text{se } \pi \text{ é par, -1 se } \pi \text{ é ímpar.}$$

Permanente:

$$\det(A) = \sum_{\pi\in S_n} a_{1\pi(1)} a_{2\pi(2)} \cdots a_{n\pi(n)}.$$

Determinante X Permanente

Dada matriz quadrada $A_{n\times n}$ sobre um corpo.

Determinante:

$$\det(A) = \sum_{\pi\in S_n} \text{sinal}(\pi) a_{1,\pi(1)} a_{2,\pi(2)} \cdots a_{n,\pi(n)}.$$

$$\text{sinal}(\pi) = 1 \text{se } \pi \text{ é par, -1 se } \pi \text{ é ímpar.}$$

Permanente:

$$\det(A) = \sum_{\pi\in S_n} a_{1\pi(1)} a_{2\pi(2)} \cdots a_{n\pi(n)}.$$

Fato: calcular determinante é polinomial.

Determinante X Permanente

Dada matriz quadrada $A_{n\times n}$ sobre um corpo.

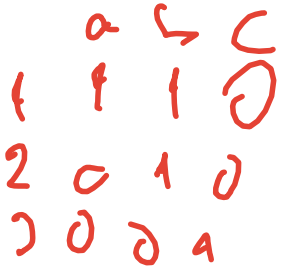
Determinante:

$$\det(A) = \sum_{\pi \in S_n} \text{sinal}(\pi) a_{1,\pi(1)} a_{2,\pi(2)} \cdots a_{n,\pi(n)}.$$

$$\text{sinal}(\pi) = 1 \text{se } \pi \text{ é par, -1 se } \pi \text{ é ímpar.}$$

Permanente:

#EMP é #P-completo



$$\det(A) = \sum_{\pi \in S_n} a_{1\pi(1)} a_{2\pi(2)} \cdots a_{n\pi(n)}.$$

Fato: calcular determinante é polinomial.

Fato: calcular permanente é **NP**-difícil, mesmo se A é uma matriz de 0's e 1's.

Complexidade de espaço

Complexidade de espaço

Dada uma MTND que termina sempre, sua complexidade de espaço é a função $f(n)$ que dá o máximo número de células da fita usadas em qualquer computação começando com entrada de comprimento n.

Complexidade de espaço

Dada uma MTND que termina sempre, sua complexidade de espaço é a função $f(n)$ que dá o máximo número de células da fita usadas em qualquer computação começando com entrada de comprimento n.

Dada uma função $f : \mathbb{N} \to \mathbb{N}$, definimos:

- $\mathrm{SPACE}(f(n)) = \{$linguagens decidiveis por uma MT com espaço $\mathcal{O}(f(n))\}$

Complexidade de espaço

Dada uma MTND que termina sempre, sua complexidade de espaço é a função $f(n)$ que dá o máximo número de células da fita usadas em qualquer computação começando com entrada de comprimento n.

Dada uma função $f : \mathbb{N} \to \mathbb{N}$, definimos:

- $\mathrm{SPACE}(f(n)) = \{$linguagens decidiveis por uma MT com espaço $\mathcal{O}(f(n))\}$
- $\mathrm{NSPACE}(f(n)) = \{$linguagens decidiveis por uma MTND com espaço $\mathcal{O}(f(n))\}$

Savitch

Savitch

Teorema de Savitch: Para toda função $f : \mathbb{N} \to \mathbb{N}$ tal que $f(n) \geq n$,

$$\text{NSPACE}(f(n)) \subseteq \text{SPACE}(f(n)^2).$$

Savitch

Teorema de Savitch: Para toda função $f : \mathbb{N} \to \mathbb{N}$ tal que $f(n) \geq n$,

$$\text{NSPACE}(f(n)) \subseteq \text{SPACE}(f(n)^2).$$

Dem.: (Idéia) Recursão, procurando o meio do caminho.

PSPACE

PSPACE

PSPACE é a classe de linguagens que podem ser decididas em espaço polinomial:

$$\text{PSPACE} = \bigcup_k \text{SPACE}(n^k).$$

PSPACE

PSPACE é a classe de linguagens que podem ser decididas em espaço polinomial:

$$\text{PSPACE} = \bigcup_k \text{SPACE}(n^k).$$

$$\mathbf{P} \subseteq \mathbf{NP} \subseteq \text{PSPACE} = \text{NSPACE} \subseteq \text{EXPTIME}$$

PSPACE

PSPACE é a classe de linguagens que podem ser decididas em espaço polinomial:

$$\text{PSPACE} = \bigcup_k \text{SPACE}(n^k).$$

$$\mathbf{P} \subseteq \mathbf{NP} \subseteq \text{PSPACE} = \text{NSPACE} \subseteq \text{EXPTIME}$$

Sabe-se: $\mathbf{P} \neq \text{EXPTIME}$.

PSPACE

PSPACE é a classe de linguagens que podem ser decididas em espaço polinomial:

$$\text{PSPACE} = \bigcup_k \text{SPACE}(n^k).$$

$\mathbf{P} \subseteq \mathbf{NP} \subseteq \text{PSPACE} = \text{NSPACE} \subseteq \text{EXPTIME}$

Sabe-se: $\mathbf{P} \neq \text{EXPTIME}$.

E existe coisa pior que EXPTIME!

ERs generalizadas

ERs generalizadas

Fixe um alfabeto Σ. Vamos estender as ERs sobre Σ com o símbolo $!$, e a semântica

$$\mathcal{L}(!E) = \Sigma^* \setminus \mathcal{L}(E).$$

ERs generalizadas

Fixe um alfabeto Σ. Vamos estender as ERs sobre Σ com o símbolo $!$, e a semântica

$$\mathcal{L}(!E) = \Sigma^* \setminus \mathcal{L}(E).$$

Claro que essas expressões denotam linguagens regulares.

ERs generalizadas

Fixe um alfabeto Σ. Vamos estender as ERs sobre Σ com o símbolo $!$, e a semântica

$$\mathcal{L}(!E) = \Sigma^* \setminus \mathcal{L}(E).$$

Claro que essas expressões denotam linguagens regulares.

Fato: Decidir se uma ER generalizada de comprimento n denota Σ^* requer espaço

$$\mathbf{torre}(2,n) = 2^{\overset{\underbrace{\quad}_{n \text{ vezes}}}{2^{\cdot^{\cdot^{\cdot^{2}}}}}}$$

Outra forma de avaliar algoritmos

Outra forma de avaliar algoritmos

Tempo médio.

Outra forma de avaliar algoritmos

Tempo médio.

Requer assumir uma distribuição de probabilidades sobre as instâncias.

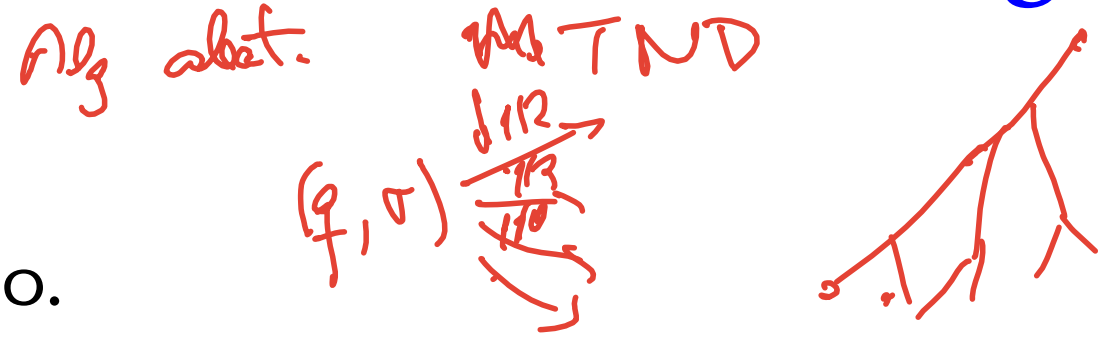
Outra forma de avaliar algoritmos

Tempo médio.

Requer assumir uma distribuição de probabilidades dobre as instâncias.

Difícil e um pouco suspeito (que distribuição?).

Outra forma de avaliar algoritmos



Tempo médio.

Requer assumir uma distribuição de probabilidades dobre as instâncias.

Difícil e um pouco suspeito (que distribuição?).

Versão mais útil e muito prática: algoritmos aleatorizados.