Teoria da Computação

Computabilidade e complexidade computacional

Computabilidade e Complexidade

- **Computabilidade**: verifica a existência de algoritmos que resolva uma classe de linguagens – trata a possibilidade da sua construção

- **Complexidade**: trata da eficiência da computação (dos algoritmos) em computadores existentes

  - Complexidade temporal: tempo de processamento exigido

  - Complexidade espacial: espaço de armazenamento exigido

Custo computacional

Complexidade computacional

Complexidade computacional

- **Computabilidade**: verifica a existência de algoritmos que resolva uma classe de linguagens – trata a possibilidade da sua construção

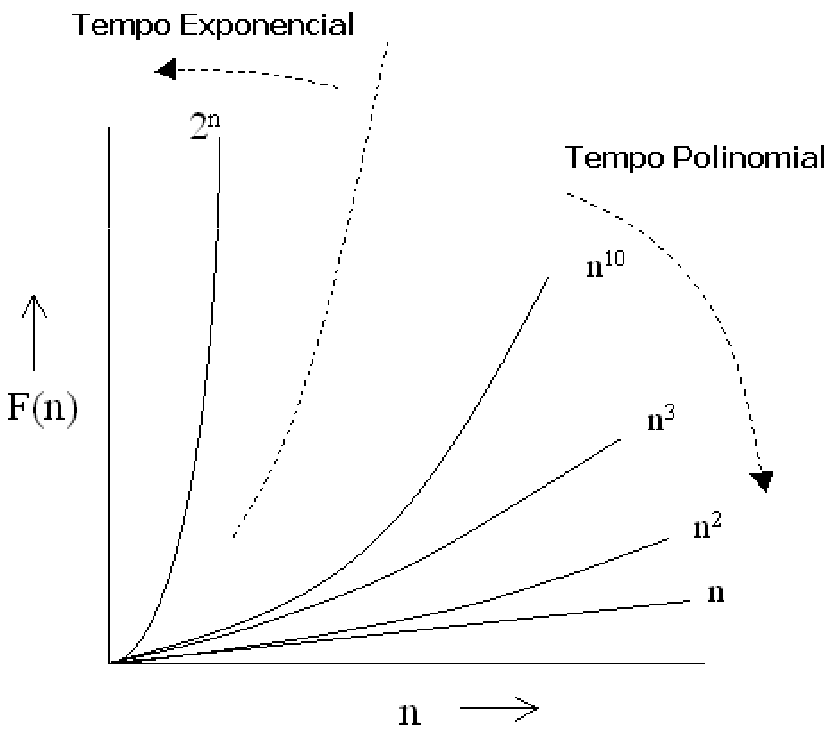
- **Complexidade**: trata da eficiência da computação (dos algoritmos) em computadores existentes
  - Complexidade temporal: tempo de processamento exigido
  - Complexidade espacial: espaço de armazenamento exigido

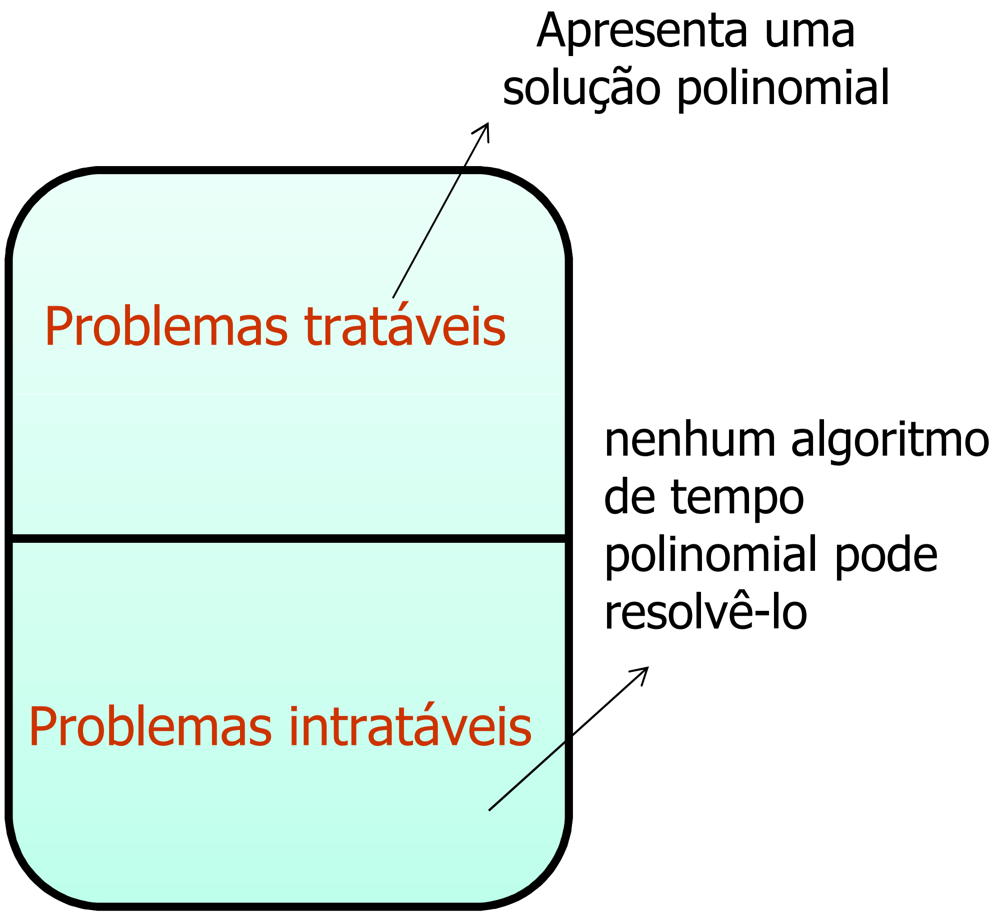
Complexidade computacional

- O conjunto das linguagens decidíveis pode ser dividida em classes de complexidade, que caracterizam os limites dos recursos computacionais usados para as decidir.

- Uma **classe de complexidade** é especificada por um modelo de computação – por exemplo, uma MT com computação determinística ou não determinística. Questões a serem consideradas:
  - recursos: tempo ou espaço
  - limite: uma função de $\mathbb{N}$ em $\mathbb{N}$.

Universo de problemas





Classes de problemas tratáveis e intratáveis

- Se um algoritmo apresentar uma complexidade polinominal ele é dito tratável, caso contrário, é intratável

  - Um problema tratável pode ser solucionado por um computador em um tempo aceitável → pode ser verificado a partir de um algoritmo de ordem polinomial
  - Algoritmos não polinomiais podem levar séculos, mesmo para entradas de tamanho reduzido.

Classificação dos problemas

- A seguir, classificaremos os problemas segundo o tempo necessário para:
    1. encontrar uma solução
    2. verificar se uma resposta fornecida é realmente uma solução para o problema.

- As classes mais importantes nesses dois quesitos são denominadas **P** e **NP**

Classes de problemas P e NP

- **Classe P**

- Classe de problemas que compreende precisamente aqueles problemas que admitem **algoritmo polinomial**. (= classe de problemas que podem ser resolvidos por uma MT determinística em tempo polinomial ao tamanho da entrada)

  - Exemplos: algoritmos de ordenação, Problema da Conectividade, entre outros.

Classes de problemas P e NP

- **Classe NP**

- A classe NP consiste nos problemas que são "verificáveis" em tempo polinomial.

- Para estes problemas são conhecidos algoritmos não-determinísticos polinomiais, ou seja, o algoritmo gera uma solução candidata ao problema e verifica sua viabilidade em tempo polinomial.

- Alguns exemplos:

  - problema do caminho hamiltoniano, cliques em grafos, conjunto independente em grafos e problema da mochila

Classes de problemas P e NP

- **Classe NP**

- Para que um problema p esteja na Classe NP é preciso que:
  - Seja possível verificar, em tempo polinomial, se uma candidata à solução de p seja de fato uma solução. Ou seja, a verificação/certificação/decisão da propriedade que faz dela uma solução para p tem que ser feita em tempo polinomial
    - resolver o problema de decisão subjacente já conhecido.

Classes de problemas P e NP

- **Classe NP – exemplos**

  - Caixeiro Viajante: Problema de decisão: Dada uma sequência de vértices do grafo, ela é um ciclo hamiltoniano?

  - Coloração de Grafos: Problema de decisão: dados G e um inteiro positivo k, existe uma coloração de G usando k cores?

- Se esses problemas de decisão forem polinomiais, então seus problemas originais, para os quais não se conhece solução polinomial, estão em NP.

Classes de problemas P e NP

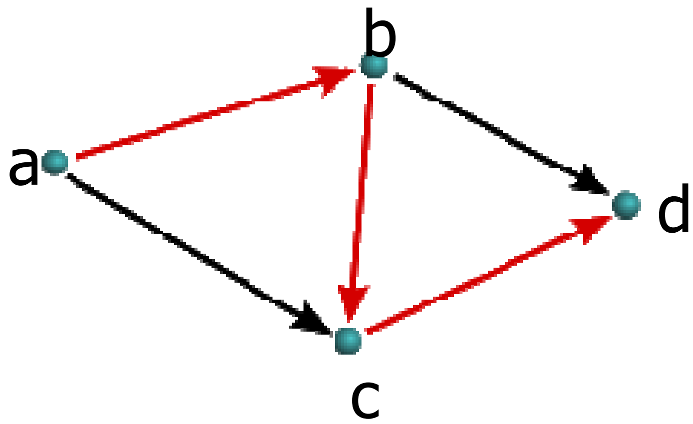
- **Classe NP**
- Observa-se que:
  - Não se exige uma solução polinomial para os problemas de NP; somente que uma certificação possa ser verificada em tempo polinomial;
  - Todo problema que está em P também está em NP, pois, se é possível achar uma solução em tempo polinomial, então é possível verificá-la em tempo polinomial também. Logo, P $\subseteq$ NP.
    - Para toda solução determinística pode ser construída uma não-determinística

Classes de problemas P e NP

- **Classe NP**
- Observa-se que:
  - Assim, os problemas que estão em NP e não estão em P são aqueles cuja certificação é polinomial, mas para os quais não se conhece solução polinomial.
  - **NP NÃO significa "Não-Polinomial", significa:** Classe de problemas que podem ser resolvidos por uma MT NÃO-Determinística, em tempo POLINOMIAL.

Classes de problemas NP

- Exemplo: ciclo Hamiltoniano está em NP?
  - **Ciclo hamiltoniano** é um caminho que permite passar por todos os vértices de um grafo G, não repetindo nenhum, ou, seja, passar por todos uma e uma só vez por cada. Ex: arestas em vermelho mostra o caminho

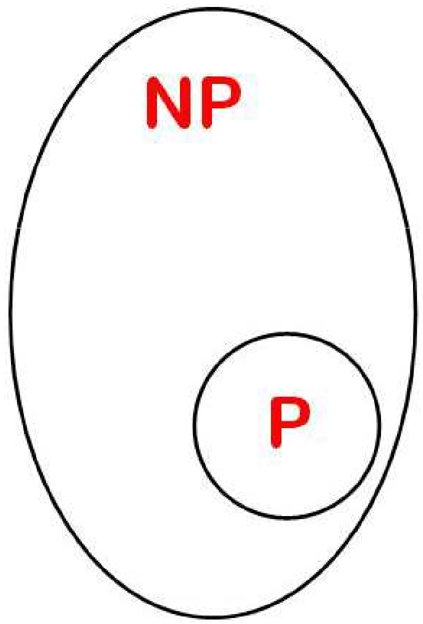


  - **Problema de Decisão subjacente:** uma sequência de vértices de G possui a propriedade de ser um ciclo hamiltoniano? (ou seja, ser um ciclo simples, contendo todos os vértices de G)

Classes de problemas P

- Exemplo de problemas da classe P: algoritmos de Ordenação

Método	Complexidade
Inserção	$O(n^2)$
Seleção	$O(n^2)$
Bolha	$O(n^2)$
Shellsort	$O(n \lg(n)^2)$
Quicksort	$O(n \lg(n))$
Heapsort	$O(n \lg(n))$



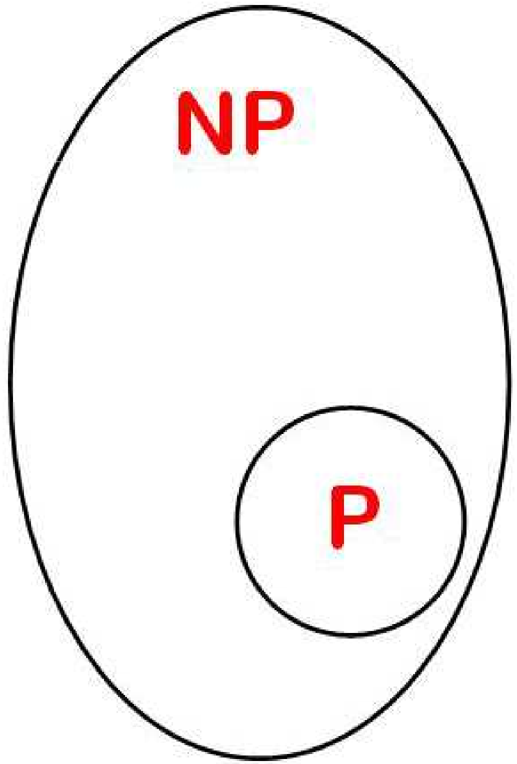
Classe de problemas P ou NP?

- Para demonstrar que um problema ***pertence à classe P*** basta mostrar um algoritmo polinomial que o resolva.

- Classe NP: devemos provar que não existe algoritmo (determinístico) polinomial para resolve-lo.

$P \subseteq NP$?

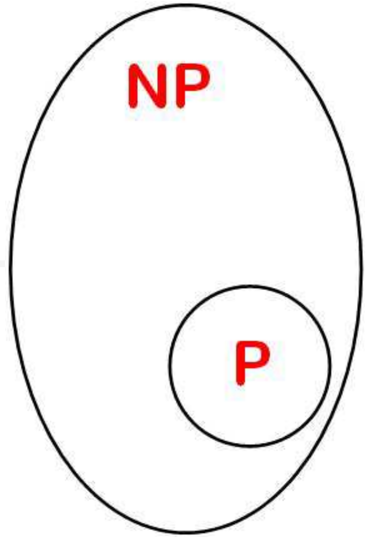
- SIM, pois algoritmos determinísticos são um caso especial dos não-determinísticos



Classe NP-Completo

- A classe NP-completo tem a propriedade de que se um problema NP- completo puder ser resolvido em tempo polinomial todos os problemas em NP tem solução polinomial.
- Para definir formalmente a classe NP-completo **precisamos da noção de Redução Polinomial**.

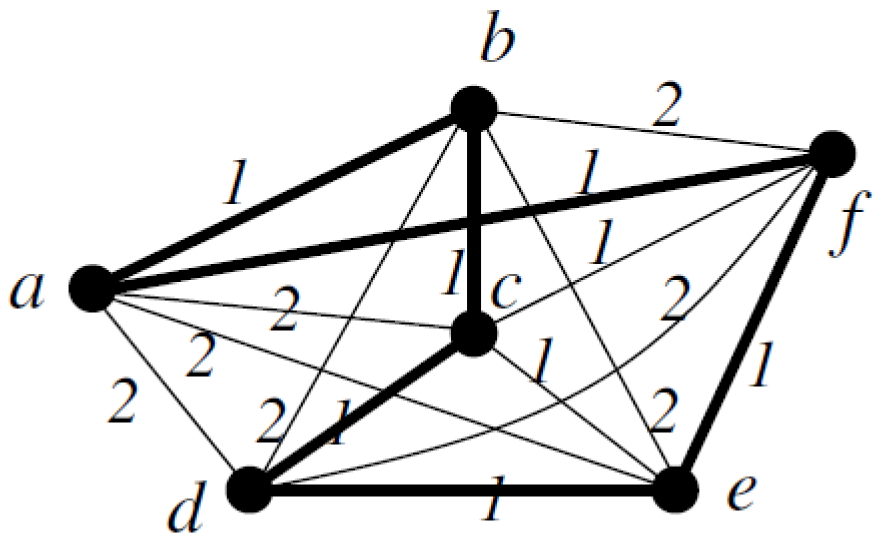
Classe NP-Completo



- NP-completo é um subconjunto de NP
  - Conjunto de todos os problemas de decisão os quais suas soluções podem ser verificadas em tempo polinomial;
  - Classe NP pode ser equivalentemente definida como o conjunto de problemas de decisão que podem ser solucionados em tempo polinomial em uma Máquina de Turing não determinística.

Exemplo: Problema do caixeiro viajante

- Este problema de decisão consiste em se determinar se há um ciclo que passa por todos os vértices apenas uma vez com **peso total no máximo um valor k**.
  - É um caso especial do ciclo hamiltoniano



Problemas exponenciais

- É desejável resolver instâncias grandes de problemas de otimização em tempo razoável.
- Os melhores algoritmos para problemas NP-completo têm comportamento de pior caso exponencial no tamanho da entrada.
- Para um algoritmo que execute em tempo proporcional a 2^N, não é garantido obter resposta para todos os problemas de tamanho N >=100.
- Independente da velocidade do computador, ninguém poderia esperar por um algoritmo que leva 2^{100} passos para terminar sua tarefa.
- Um supercomputador poderia resolver um problema de tamanho N=50 em 1 hora, ou N=51 em 2 horas, ou N=59 em um ano.

O que fazer para resolver problemas exponenciais?

- Usar algoritmos exponenciais “eficientes” aplicando técnicas de tentativa e erro.
- Usar algoritmos aproximados. Acham uma resposta que pode não ser a solução ótima, mas é garantido ser próxima dela.
- Exemplo: backtracking (tentativa e erro), programação dinâmica, branch-and-bound (ramificação e poda), etc.

Complexidade de algoritmos