UDESC

Complexidade de Algoritmos

Prof. Rafael Alceste Berri
rafaelberri@gmail.com

Prof. Diego Buchinger
diego.buchinger@outlook.com

Prof. Cristiano Damiani Vasconcellos
cristiano.vasconcellos@udesc.br

UDESC

Estudo da Tratabilidade de Problemas Computacionais

UDESC

Limite de complexidade de um problema

- Limite superior: melhor algoritmo
  - Ex: classificação O(n log n).
- Limite inferior: melhor complexidade (não dá para melhorar mais).
  - Teóricos calculam
- Sendo limite superior = limite inferior
  - Problema fechado (complexidade mínima conhecida)
    - Algoritmo solução conhecido em limite inferior.
  - Ex: Classificação -> fechado, complexidade mínima de O(n log n).

Problemas tratáveis e intratáveis

Problemas tratáveis: resolvidos por algoritmos deterministas que executam em **tempo polinomial** (limite superior).

Problemas intratáveis: **não se conhece algoritmos** deterministas que os resolvam em tempo polinomial (limite superior não é polinomial).

$$1 < \log\log n < \log n < n^{\varepsilon} < n^{c} < n^{\log n} < c^{n}$$

Problemas tratáveis e intratáveis

Problemas tratáveis: resolvidos por algoritmos deterministas que executam em **tempo polinomial**.

Problemas intratáveis: **não se conhece algoritmos** deterministas que os resolvam em tempo polinomial.

$$1 < \log\log n < \log n < n^{\varepsilon} < n^{c} < n^{\log n} < c^{n}$$

UDESC

Categorias de Problemas

Problemas de Otimização: Cada solução possível tem um valor associado e desejamos encontrar a solução com **melhor valor**.

Problemas de Decisão: Problemas que tem resposta **sim ou não**.

Problemas de Decisão são possivelmente “mais fáceis” do que problemas de Otimização, mas com certeza “não mais difíceis”!

Exemplo:

- Qual é o menor caminho entre os vértices *a* e *b* de um grafo?
- Existe um caminho de no máximo k arestas entre *a* e *b*?

Algoritmos Não Deterministas

Capaz de escolher **uma entre várias alternativas possíveis** a cada passo. A alternativa escolhida será sempre a alternativa que leva a conclusão esperada, caso essa alternativa exista.

```
int pesq(Estr *v, int n, int ch){
   int i;
   for (i = 0; i < n; i++)
      if (v[i].chave == ch) // decisão
         return i;
   return -1;
}
```

Máquina de Turing Determinista

Autômato/Máquina de Turing não determinista

```
int pesq(Estr *v, int n, int ch){
   int i;
   i = magicaND(0, n - 1);
   if (v[i].chave == ch)
      return i;
   return -1;
}
```

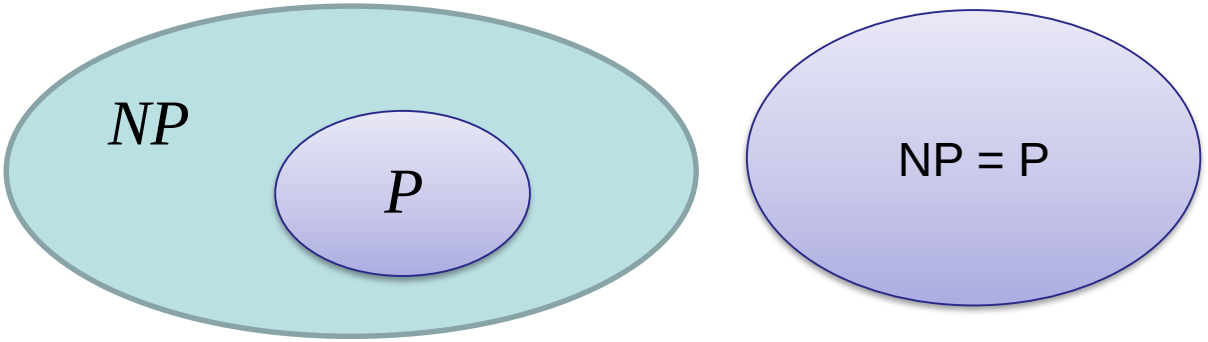
Classes de Problemas *P* e *NP*

Classe de Problemas *P*: Problemas que podem ser resolvido (por algoritmos deterministas) em tempo polinomial.

Classe de Problemas *NP*: Problemas que podem ser resolvidos por algoritmos não deterministas em tempo polinomial (*polinomialmente verificável* ou *certificado*). Ou problemas que a solução pode ser verificada em tempo polinomial (**Decisão é Polinomial**).

Pergunta do milhão: P=NP ou P ≠NP?

Possíveis relações entre as classes:



Classes de Problemas *P* e *NP*

O status de muitos problemas **NP é desconhecido**:

- existe um algoritmo determinista polinomial para o problema?

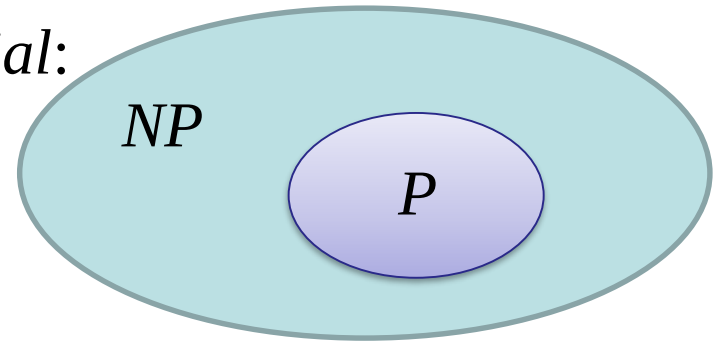
OBS: Máquina de Turing Não-Determinística "resolve" em tempo polinomial (usando sua mágica).

Investigar a complexidade relativa dos problemas da classe NP:

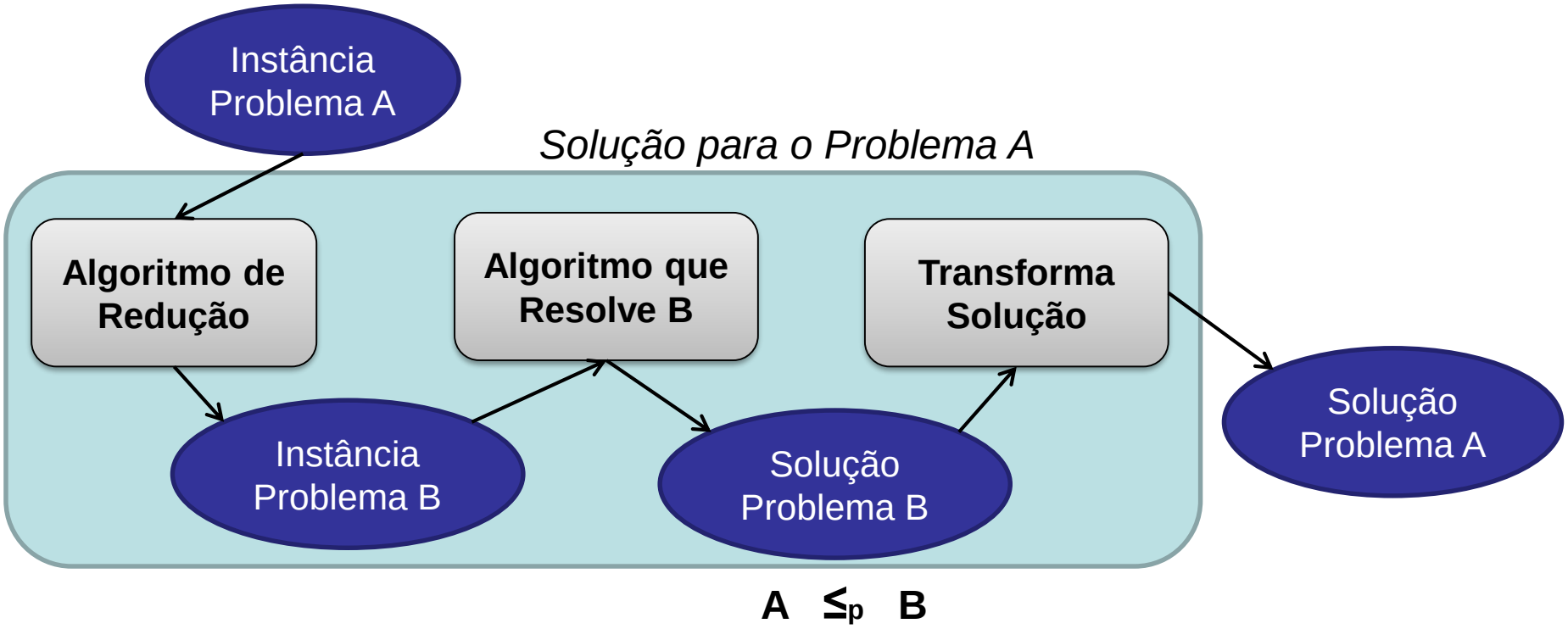
- Problema A é mais fácil ou mais difícil do que B?

Recorremos à ideia de *redução polinomial*:

- Mostra que A não é mais difícil que B ou que A é polinomialmente redutível ao problema B.



Redução de Problemas



Se o "**Algoritmo de Redução**", o "**Algoritmo que Resolve B**" e a "**Transformação de solução**" forem polinomiais, então podemos concluir algo sobre a solução do Problema A?

Redução de Problemas

Conclusões provenientes da redução:

Se Y é polinomialmente redutível a X então Y não é mais difícil do que X.

$$Y \leq_p X$$

Cenário 1: sabe-se que X está na classe P.
Logo, Y também deve estar na classe P.

Cenário 2: não se sabe se X está ou não em P,
mas sabe-se que Y não está em P.
Como Y não é mais difícil que X, então X deve estar fora de P.

Exercícios

1. Se um problema NP for resolvido em tempo polinomial então P = NP? Justifique sua resposta.
2. Encontre um problema que, até hoje, não tenha uma solução polinomial para ele, ou seja, com limite superior entre $n^{\log n}$ $até$ c^n. Descreva brevemente o problema (desafios envolvidos). Existe alguma heurística/algoritmo aproximativo capaz de apresentar alguma solução razoável (não necessariamente a melhor solução) em tempo polinomial para ele? Apresente a solução se disponível.

Envie respostas para rafaelberri@gmail.com Assunto: "TC-CAL13"
Anexar: resultados/código fonte

Classes de Problemas *NP-Hard*

Se podemos determinar que um problema não é mais difícil do que outro, podemos separar os problemas **mais difíceis** dos mais fáceis em NP!

Assim surge a classe dos problemas mais difíceis

A classe de problemas **NP-*Hard*** ou **NP-Difícil**!

"Um problema A é NP-Difícil se todos os problemas em NP **não são mais difíceis** do que A"

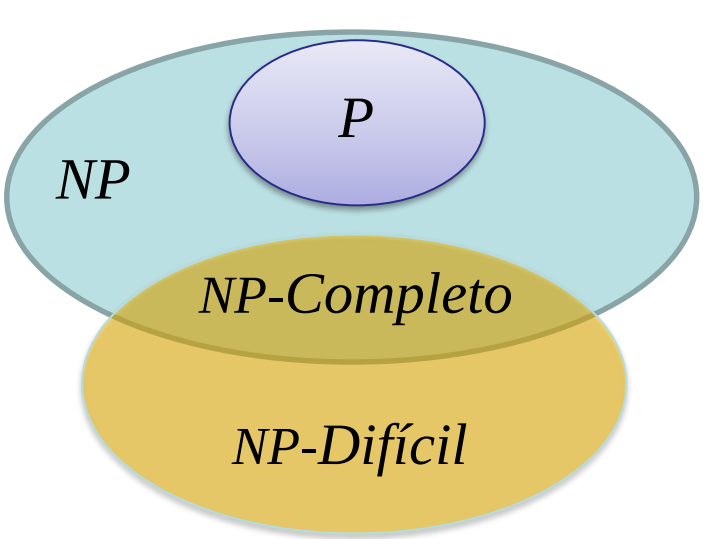
"Um problema NP-Difícil é **tão difícil quanto** qualquer problema em NP"

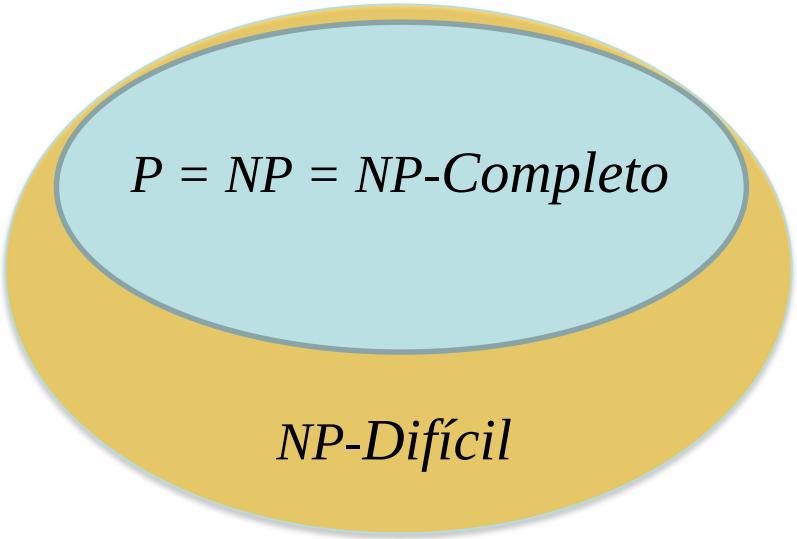
Classes de Problemas *NP-Hard*

Na classe NP-Difícil podemos encontrar problemas:

- Indecidíveis: ex. problema da parada e equações diofantinas;
- Decidíveis: podem ser resolvidos por um **algoritmo determinista em tempo polinomial** – um problema NP-Difícil que está em NP é dito **NP-Completo**.

Duas possíveis relações considerando P vs. NP





Redução de Problemas

Relação entre Redução e Problemas NP-Completos:

Uma vez conhecido um problema NP-Completo, podemos usar ***reduções polinomiais*** para provar que algum problema X também é NP-Completo.

"se *Y* é um problema NP-completo e *Y* não é mais difícil que um problema *X* (redução) então *X* também é NP-completo"

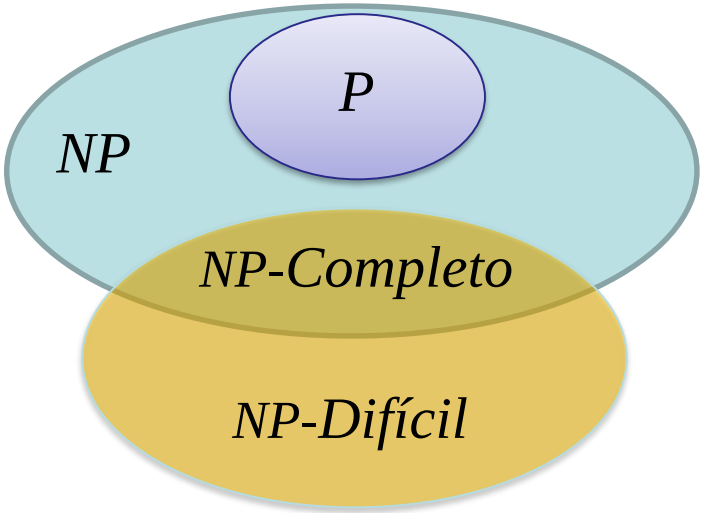
$$Y \leq_p X$$

Classes de Problemas *NP-Completo*

Se **um problema** NP-Completo pode ser resolvido em **tempo polinomial**, então **todo problema** NP-Completo pode ser resolvido em tempo polinomial e, portanto, **P = NP**

Acredita-se que a relação correta seja P ≠ NP

Por quê?



UDESC

NP-Completo

Um problema X é ***NP-Completo*** se:

1. O problema **deve ser NP**: $X \in NP$

   a) *Conseguir um algoritmo não determinista que resolva o problema em tempo polinomial*

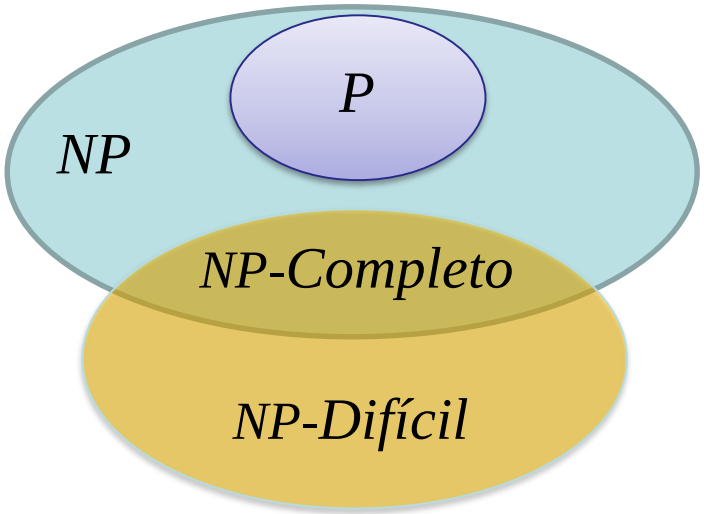
   b) *Conseguir um algoritmo determinista que verifica em tempo polinomial se uma resposta é verdadeira ou não* ***(certificado)***

2. Fazer a **redução** de um problema **NP-Completo** (Y) conhecido para o problema X: $Y \leq_p X$ para todo $Y \in NP$

Ex: NP-Completo é o Satisfatibilidade Booleana (SAT)

NP-Difícil ou NP-Hard

- São **polinomialmente reduzíveis** a NP-completos.
- **Não há algoritmo conhecido não determinístico** (tempo polinomial)
  - São NP? **não se sabe!**
- Possivelmente **mais difíceis** que NP-completos, provável que **NP-Difícil ⊄ NP**.
- *Classe nebulosa*



Exercícios

Verifique se as afirmações abaixo são **verdadeiras** ou **falsas**, justificando as **falsas**:

a) Se um problema NP-Difícil (ou NP-Hard) for resolvido em tempo polinomial então P = NP.

b) Se P = NP então todos os problemas considerados NP-Difícil (ou NP-Hard) podem ser resolvidos em tempo polinomial.

c) Podemos afirmar que os problemas NP-Completo possuem apenas soluções em tempo exponencial ou maior.

d) Considerando um problema P1 que tem uma solução em tempo polinomial conhecida, e um problema P2 que é NP-Completo, apresentando uma redução que pode ser executada em tempo polinomial de P1 a P2 ($P1 \leq_p P2$) estamos provando que P = NP.

e) Se $P \cap$ NP-Completo $\neq \emptyset$ então P = NP.

Envie respostas para rafaelberri@gmail.com Assunto: "TC-CAL14"
Anexar: resultados/código fonte