Projeto e Análise de Algoritmos II (MC558)
Classes de Problemas

Prof. Dr. Ruben Interian

Resumo

Resumo

Revisão do conteúdo

Classes de problemas que vimos até o momento:

- $\mathcal{P}$: problemas que podem ser resolvidos por um **algoritmo determinístico** em tempo polinomial.
- $\mathcal{NP}$: problemas que podem ser resolvidos por um **algoritmo não-determinístico** em tempo polinomial.
- $\mathcal{NP}$-difícil: todo problema $\in \mathcal{NP}$ se reduz a cada um dos problemas $\mathcal{NP}$-difíceis em tempo polinomial.
- $\mathcal{NP}$-completo: $\mathcal{NP} \cap \mathcal{NP}$-difícil.

Objetivo

Objetivo:

- Estudar uma nova (e última) classe de problemas - os problemas **Indecidíveis**.

Observação

Hoje não veremos (quase) nenhum algoritmo.

Observação

Hoje não veremos (quase) nenhum algoritmo.

Os problemas que estudaremos não possuem algoritmos que os resolvam!

Resumo

Indecidibilidade: Introdução

- Os algoritmos parecem ser tão poderosos que poderíamos acreditar que todos os problemas acabarão "cedendo": que seria possível criar **pelo menos um algoritmo** para cada problema, mesmo seja ineficiente ou exponencial.

  Durante muito tempo acreditou-se que este era o caso.

Indecidibilidade: Introdução

- Os algoritmos parecem ser tão poderosos que poderíamos acreditar que todos os problemas acabarão "cedendo": que seria possível criar **pelo menos um algoritmo** para cada problema, mesmo seja ineficiente ou exponencial.

  Durante muito tempo acreditou-se que este era o caso.

- No entanto, é possível mostrar a existência de limitações fundamentais no poder da computação e dos algoritmos. Não somos "onipotentes".

Indecidibilidade: Introdução

Palavra **indecidível**: algo que não conseguimos decidir, ou responder corretamente.

Lembrando: Estamos trabalhando com a versão de decisão dos problemas.

Pergunta: Haverá problemas para os quais não conseguimos criar um algoritmo que possa chegar corretamente à resposta `SIM` ou `NÃO`, em todos os casos?

Indecidibilidade: Introdução

Palavra **indecidível**: algo que não conseguimos decidir, ou responder corretamente.

Lembrando: Estamos trabalhando com a versão de decisão dos problemas.

Pergunta: Haverá problemas para os quais não conseguimos criar um algoritmo que possa chegar corretamente à resposta `SIM` ou `NÃO`, em todos os casos?

Problema indecidível

Um problema de decisão P é indecidível se não existe um algoritmo que resolve P, ou seja, se não existe um algoritmo que irá retornar corretamente `SIM` ou `NÃO` para qualquer entrada.

Indecidibilidade: Introdução

Que tipo de problemas são indecidíveis?
São problemas esotéricos, relevantes apenas para cientistas de Teoria da Computação?

- Não! Diversos problemas comuns que as pessoas **querem** ou **precisam resolver** são computacionalmente indecidíveis!

Indecidibilidade: Problema da Parada

Exemplo de problema útil que muitos gostariam de resolver:

- Dado o código de um programa (algoritmo) em alguma linguagem de programação, **decidir** se este algoritmo **termina ou não termina**.

  Exemplo simples de algoritmo que não termina (*loop infinito*):

```
while (true) continue;
```

Indecidibilidade: Problema da Parada

Exemplo de problema útil que muitos gostariam de resolver:

- Dado o código de um programa (algoritmo) em alguma linguagem de programação, **decidir** se este algoritmo **termina ou não termina**.

  Exemplo simples de algoritmo que não termina (*loop infinito*):

  ```
  while (true) continue;
  ```

- Para muitos algoritmos, parece ser mais difícil classificar cada um deles em "algoritmo que termina" / "algoritmo que não termina".

Indecidibilidade: Problema da Parada

Exemplo: Este algoritmo **termina** ou **não termina**?

f_curiosa (n)

1: **enquanto** $n \neq 1$ **faça**
2: **se** n é par **então** $n = n/2$
3: **senão** $n = 3n + 1$

Indecidibilidade: Problema da Parada

Exemplo: Este algoritmo **termina** ou **não termina**?

f_curiosa (n)

enquanto $n \neq 1$ **faça**
    se n é par **então** $n = n/2$
    senão $n = 3n + 1$

- Não parece tão fácil saber se esse programa finaliza para qualquer n.

Indecidibilidade: Problema da Parada

Exemplo: Este algoritmo **termina** ou **não termina**?

f_curiosa (n)

1: **enquanto** $n \neq 1$ **faça**
2: **se** n é par **então** $n = n/2$
3: **senão** $n = 3n + 1$

- Não parece tão fácil saber se esse programa finaliza para qualquer n.

Quem quiser ler mais sobre o assunto $\rightarrow$ pesquisar "conjectura de Collatz".

Indecidibilidade: Problema da Parada

Suponha que você implementou um algoritmo H (em alguma linguagem de programação). Dado um programa P e uma entrada x:

- H retorna `SIM` se P para (finaliza) com a entrada x.
- H retorna `NÃO` se P não para (não finaliza) com a entrada x.

Indecidibilidade: Problema da Parada

Suponha que você implementou um algoritmo H (em alguma linguagem de programação). Dado um programa P e uma entrada x:

- H retorna `SIM` se P para (finaliza) com a entrada x.
- H retorna `NÃO` se P não para (não finaliza) com a entrada x.

Veja que esse programa seria **extremamente útil**! (e.g., para uso em **compiladores**)

Indecidibilidade: Problema da Parada

Suponha que você implementou um algoritmo H (em alguma linguagem de programação). Dado um programa P e uma entrada x:

- H retorna `SIM` se P para (finaliza) com a entrada x.
- H retorna `NÃO` se P não para (não finaliza) com a entrada x.

Veja que esse programa seria **extremamente útil**! (e.g., para uso em **compiladores**)

Usando H, é possível escrever o algoritmo **Halts** que testa (na linha 1) se um programa P finaliza quando a sua própria codificação for passada na entrada.

Halts (P)

1: **enquanto** $H(P, P) = \texttt{SIM}$ **faça**
2: `Imprime`("Halts")
3: **finalizar**

Indecidibilidade: Problema da Parada

Halts (P)

1: **enquanto** $H(P, P) = \texttt{SIM}$ **faça**
2: $\quad$ `Imprime`("Halts")
3: **finalizar**

E se passarmos **Halts** como entrada para ele mesmo? **Halts(Halts)** finalizaria? Ou não?. . .

Indecidibilidade: Problema da Parada

Halts (P)

1: **enquanto** $H(P, P) = \texttt{SIM}$ **faça**
2: $\quad$ `Imprime`("Halts")
3: **finalizar**

E se passarmos **Halts** como entrada para ele mesmo? **Halts(Halts)** finalizaria? Ou não?... Vamos ver:

- Se **H(Halts, Halts)** retorna `SIM` $\Rightarrow$

Indecidibilidade: Problema da Parada

Halts (P)

1: **enquanto** $H(P, P) = \texttt{SIM}$ **faça**
2: $\quad$ `Imprime`("Halts")
3: **finalizar**

E se passarmos **Halts** como entrada para ele mesmo? **Halts(Halts)** finalizaria? Ou não?... Vamos ver:

- Se **H(Halts, Halts)** retorna `SIM` $\Rightarrow$ **Halts(Halts)** não finaliza.

Indecidibilidade: Problema da Parada

Halts (P)

1: **enquanto** $H(P, P) = \texttt{SIM}$ **faça**
2: `Imprime`("Halts")
3: **finalizar**

E se passarmos **Halts** como entrada para ele mesmo? **Halts(Halts)** finalizaria? Ou não?... Vamos ver:

- Se **H(Halts, Halts)** retorna `SIM` $\Rightarrow$ **Halts(Halts)** não finaliza.
- Se **H(Halts, Halts)** retorna `NÃO` $\Rightarrow$

Indecidibilidade: Problema da Parada

Halts (P)

1: **enquanto** $H(P,P) = \texttt{SIM}$ **faça**
2: $\quad$ `Imprime`("Halts")
3: **finalizar**

E se passarmos **Halts** como entrada para ele mesmo? **Halts(Halts)** finalizaria? Ou não?. . . Vamos ver:

- Se **H(Halts, Halts)** retorna `SIM` $\Rightarrow$ **Halts(Halts)** não finaliza.
- Se **H(Halts, Halts)** retorna `NÃO` $\Rightarrow$ **Halts(Halts)** finaliza.

Indecidibilidade: Problema da Parada

Halts (P)

1: **enquanto** $H(P, P) = \texttt{SIM}$ **faça**
2: `Imprime`("Halts")
3: **finalizar**

E se passarmos **Halts** como entrada para ele mesmo? **Halts(Halts)** finalizaria? Ou não?... Vamos ver:

- Se **H(Halts, Halts)** retorna `SIM` $\Rightarrow$ **Halts(Halts)** não finaliza.
- Se **H(Halts, Halts)** retorna `NÃO` $\Rightarrow$ **Halts(Halts)** finaliza.

O que há de errado aqui???

Indecidibilidade: Problema da Parada

Definição: Problema da Parada

O **Problema da Parada** (**HALT**) é o problema de, dado um algoritmo (Máquina de Turing) e uma entrada finita, decidir se o algoritmo retorna alguma saída, ou não (executa infinitamente ou trava).

Observação:

Ou seja, o **Problema da Parada** é o problema de decisão que seria resolvido pelo hipotético algoritmo H definido acima.

Indecidibilidade: Problema da Parada

Teorema

O **Problema da Parada** é indecidível.

Indecidibilidade: Problema da Parada

Teorema

O **Problema da Parada** é indecidível.

Prova: (por **contradição**)

Suponha que existe um algoritmo H que resolve o **Problema da Parada**, ou seja, dado um conjunto de instruções de uma Máquina de Turing M e uma entrada x, decide se a Máquina de Turing aceita x. Usando H, podemos definir **Halts**, como nos dois slides anteriores, chegando a uma contradição. ∎

Indecidibilidade: Problema da Parada

O **Problema da Parada** é indecidível.

Quais são as **consequências** desse resultado?

Indecidibilidade: Problema da Parada

O **Problema da Parada** é indecidível.

Quais são as **consequências** desse resultado?

- **Ninguém jamais** será capaz de criar um algoritmo para o Problema de Parada.

Indecidibilidade: Problema da Parada

O **Problema da Parada** é indecidível.

😱

Quais são as **consequências** desse resultado?

- **Ninguém jamais** será capaz de criar um algoritmo para o Problema de Parada.
- **Não está tudo perdido**: casos particulares (por exemplo, para algoritmos simples) podem ser resolvidos.

Indecidibilidade: Problema da Parada

O **Problema da Parada** é indecidível.

😱

Quais são as **consequências** desse resultado?

- **Ninguém jamais** será capaz de criar um algoritmo para o Problema de Parada.
- **Não está tudo perdido**: casos particulares (por exemplo, para algoritmos simples) podem ser resolvidos.
- Porém, **não faz sentido investir esforços** para resolver esse problema para o caso mais geral. É importante saber quando um problema é indecidível!

Indecidibilidade: Problema da Parada

Outras características do **Problema da Parada (HALT)**:

- **HALT** é $\mathcal{NP}$−difícil.
  - Redução **SAT** $\propto_{\text{poli}}$ **HALT**:
    Dada a instância do **SAT**, defina um algoritmo (MT) que testa todas as atribuições possíveis às variáveis. Se não for encontrada uma atribuição válida, o algoritmo entra em loop infinito. O algoritmo finaliza se e somente se a instância é `SIM` para **SAT**.

Indecidibilidade: Problema da Parada

Outras características do **Problema da Parada** (**HALT**):

- **HALT** é $\mathcal{NP}-$difícil.
  - Redução **SAT** $\propto_{\text{poli}}$ **HALT**:
    Dada a instância do **SAT**, defina um algoritmo (MT) que testa todas as atribuições possíveis às variáveis. Se não for encontrada uma atribuição válida, o algoritmo entra em loop infinito. O algoritmo finaliza se e somente se a instância é `SIM` para **SAT**.
- **HALT** é $\mathcal{NP}-$completo?

Indecidibilidade: Problema da Parada

Outras características do **Problema da Parada (HALT)**:

- **HALT** é $\mathcal{NP}-$difícil.
  - Redução **SAT** $\propto_{\text{poli}}$ **HALT**:
    Dada a instância do **SAT**, defina um algoritmo (MT) que testa todas as atribuições possíveis às variáveis. Se não for encontrada uma atribuição válida, o algoritmo entra em loop infinito. O algoritmo finaliza se e somente se a instância é `SIM` para **SAT**.
- **HALT** é $\mathcal{NP}-$completo? – Não. Por quê?

Indecidibilidade: Reduções

Reduções para mostrar indecidibilidade:

- Vamos supor que temos dois problemas de decisão, A e B, e $A \propto B$. Se A é indecidível, há algo que podemos dizer sobre B?

Indecidibilidade: Reduções

Reduções para mostrar indecidibilidade:

- Vamos supor que temos dois problemas de decisão, A e B, e $A \propto B$.
  Se A é indecidível, há algo que podemos dizer sobre B?
- Se A é indecidível, então B é indecidível:
  Se pudéssemos resolver B, resolveríamos A!

Indecidibilidade: Problema da Veracidade

Definição: Problema da Veracidade

O **Problema da Veracidade** (**TRUTH**) é o problema de, dado um algoritmo e uma entrada finita, decidir se o algoritmo retorna `SIM`, ou não (ou seja, retorna `NÃO`, trava ou executa infinitamente).

Comparando com o Problema da Parada:

- **Problema da Veracidade**:
  Podemos construir um algoritmo que determine se outro algoritmo retorna `SIM`?
- **Problema da Parada**:
  Podemos construir um algoritmo que determine se outro algoritmo finaliza?

Indecidibilidade: Problema da Veracidade

Teorema

O **Problema da Veracidade** é indecidível.

Ideia da Prova: (por **contradição**)
Suponha que existe um algoritmo T que decide o Problema da Veracidade.

Indecidibilidade: Problema da Veracidade

Teorema

O **Problema da Veracidade** é indecidível.

Ideia da Prova: (por **contradição**)
Suponha que existe um algoritmo T que decide o Problema da Veracidade.
Defina H, um algoritmo que recebe na entrada um algoritmo M e a entrada w para M:

H (M, w)

1: $x \leftarrow T(M, w)$
2: **se** x **então devolva** `SIM`
3: **senão**
4: Defina um algoritmo $M'(w)$: { `return not(M(w))` }
5: **devolva** $T(M', w)$

Indecidibilidade: Problema da Veracidade

H (M, w)

$x \leftarrow T(M, w)$
se x **então devolva** `SIM`
senão
  Defina um algoritmo $M'(w) = \{$ `return not(M(w))` $\}$
  devolva $T(M', w)$

O algoritmo H resolve corretamente o Problema da Parada, que é indecidível, o que é uma **contradição**. Portanto, não existe T que decide o Problema da Veracidade. ∎

Indecidibilidade: Problema da Veracidade

Será que todos os problemas indecidíveis são para determinar propriedades de algoritmos (Se os algoritmos finalizam? O que eles retornam)?

Indecidibilidade: Problema da Veracidade

Será que todos os problemas indecidíveis são para determinar propriedades de algoritmos (Se os algoritmos finalizam? O que eles retornam)?

– Não! Mesmo problemas elementares sobre listas, conjuntos ou matrizes podem ser indecidíveis! Veremos exemplos a seguir.

Indecidibilidade: Problema da Correspondência de Post

Problema da Correspondência de Post, ou Problema do Dominó (**PCP**)

Seja $\mathcal{A}$ um alfabeto (com pelo menos dois símbolos), e sejam duas listas de n palavras sobre $\mathcal{A}$:

$$\alpha_1, \alpha_2, \ldots, \alpha_n \text{ e } \beta_1, \beta_2, \ldots, \beta_n.$$

Indecidibilidade: Problema da Correspondência de Post

Problema da Correspondência de Post, ou Problema do Dominó (**PCP**)

Seja $\mathcal{A}$ um alfabeto (com pelo menos dois símbolos), e sejam duas listas de n palavras sobre $\mathcal{A}$:

$$\alpha_1, \alpha_2, \ldots, \alpha_n \text{ e } \beta_1, \beta_2, \ldots, \beta_n.$$

Objetivo: decidir se existe uma sequência de $K \geq 1$ índices $(i_k) = i_1, \ldots, i_K$, $1 \leq i_k \leq n$, de forma que

$$\alpha_{i_1} \ldots \alpha_{i_K} = \beta_{i_1}, \ldots, \beta_{i_K}.$$

Indecidibilidade: Problema da Correspondência de Post

Problema da Correspondência de Post, ou Problema do Dominó (**PCP**)

Seja $\mathcal{A}$ um alfabeto (com pelo menos dois símbolos), e sejam duas listas de n palavras sobre $\mathcal{A}$:

$$\alpha_1, \alpha_2, \ldots, \alpha_n \text{ e } \beta_1, \beta_2, \ldots, \beta_n.$$

Objetivo: decidir se existe uma sequência de $K \geq 1$ índices $(i_k) = i_1, \ldots, i_K$, $1 \leq i_k \leq n$, de forma que

$$\alpha_{i_1} \ldots \alpha_{i_K} = \beta_{i_1}, \ldots, \beta_{i_K}.$$

Observação: queremos saber se há um tipo específico de **correspondência** entre os elementos de duas listas (nada a ver com a correspondência por cartas ou mensagens!).

Indecidibilidade: Problema da Correspondência de Post

Problema da Correspondência de Post: **formulação equivalente** (+ fácil de entender).

Temos uma coleção de dominós (peças de Dominó), onde cada peça possui duas strings (cadeias de caracteres), uma em cada face.

Uma peça do Dominó seria: $\left[\begin{matrix} ab \\ acc \end{matrix}\right]$.

Indecidibilidade: Problema da Correspondência de Post

Problema da Correspondência de Post: **formulação equivalente** (+ fácil de entender).

Temos uma coleção de dominós (peças de Dominó), onde cada peça possui duas strings (cadeias de caracteres), uma em cada face.

Uma peça do Dominó seria: $\begin{bmatrix} \color{blue}{ab} \\ \color{red}{acc} \end{bmatrix}$.

Uma coleção de peças seria: $\left\{ \begin{bmatrix} \color{blue}{b} \\ \color{red}{ca} \end{bmatrix}, \begin{bmatrix} \color{blue}{a} \\ \color{red}{ab} \end{bmatrix}, \begin{bmatrix} \color{blue}{ca} \\ \color{red}{a} \end{bmatrix}, \begin{bmatrix} \color{blue}{abc} \\ \color{red}{c} \end{bmatrix} \right\}$.

Indecidibilidade: Problema da Correspondência de Post

Problema da Correspondência de Post: **formulação equivalente** (+ fácil de entender).

Temos uma coleção de dominós (peças de Dominó), onde cada peça possui duas strings (cadeias de caracteres), uma em cada face.

Uma peça do Dominó seria: $\left[\frac{ab}{acc}\right]$.

Uma coleção de peças seria: $\left\{ \left[\frac{b}{ca}\right], \left[\frac{a}{ab}\right], \left[\frac{ca}{a}\right], \left[\frac{abc}{c}\right] \right\}$.

Objetivo: encontrar uma sequência de peças (as repetições são permitidas), de modo que a sequência de letras na parte superior seja a mesma que a da parte inferior.

Indecidibilidade: Problema da Correspondência de Post

Problema da Correspondência de Post: **formulação equivalente** (+ fácil de entender).

Temos uma coleção de dominós (peças de Dominó), onde cada peça possui duas strings (cadeias de caracteres), uma em cada face.

Uma peça do Dominó seria: $\left[\frac{ab}{acc}\right]$.

Uma coleção de peças seria: $\left\{ \left[\frac{b}{ca}\right], \left[\frac{a}{ab}\right], \left[\frac{ca}{a}\right], \left[\frac{abc}{c}\right] \right\}$.

Objetivo: encontrar uma sequência de peças (as repetições são permitidas), de modo que a sequência de letras na parte superior seja a mesma que a da parte inferior.

Exemplo de solução: $\left[\frac{a}{ab}\right] \left[\frac{b}{ca}\right] \left[\frac{ca}{a}\right] \left[\frac{a}{ab}\right] \left[\frac{abc}{c}\right]$. Sequência de índices: $2, 1, 3, 2, 4$.

Indecidibilidade: Problema da Correspondência de Post

Teorema

O **Problema da Correspondência de Post** é indecidível.

Indecidibilidade: Problema da Correspondência de Post

Teorema

O **Problema da Correspondência de Post** é indecidível.

Ideia da Prova: Conceitualmente a prova é simples. Consiste em reduzir o Problema da Veracidade para o Problema da Correspondência: **TRUTH** $\propto$ **PCP**.

Indecidibilidade: Problema da Correspondência de Post

Teorema

O **Problema da Correspondência de Post** é indecidível.

Ideia da Prova: Conceitualmente a prova é simples. Consiste em reduzir o Problema da Veracidade para o Problema da Correspondência: **TRUTH** $\propto$ **PCP**.

- Há alguns detalhes técnicos complexos. A redução transforma uma execução de um algoritmo M (MT) para uma entrada w em uma instância P do **PCP**.
- É possível mostrar que, para qualquer algoritmo M e entrada w, podemos criar uma instância P do **PCP** que possui solução **se e somente se** a entrada w é instância `SIM` para M, ou seja, se existe uma execução de aceitação de w.
- Se for possível resolver o Problema da Correspondência de Post, seria possível determinar se a entrada w é (ou não é) `SIM` para M!

Indecidibilidade: Problema da Correspondência de Post

- A indecidibilidade do Problema da Correspondência de Post foi **muito importante**: ela mostrou que existem problemas indecidíveis envolvendo estruturas simples, tais como listas e matrizes.

  A partir deste resultado, foi **mais fácil** mostrar a indecidibilidade de problemas onde a entrada não envolve um algoritmo.

Indecidibilidade: Problema da Correspondência de Post

- A indecidibilidade do Problema da Correspondência de Post foi **muito importante**: ela mostrou que existem problemas indecidíveis envolvendo estruturas simples, tais como listas e matrizes.

  A partir deste resultado, foi **mais fácil** mostrar a indecidibilidade de problemas onde a entrada não envolve um algoritmo.

- Vamos analisar mais um exemplo de um problema muito simples de formular (e impossível de resolver) **envolvendo matrizes**.

Indecidibilidade: Problema da Mortalidade de Matrizes

Problema da Mortalidade de Matrizes,
(ou Problema Mortal de Matrizes, *Matrix Mortality Problem*)

Indecidibilidade: Problema da Mortalidade de Matrizes

Problema da Mortalidade de Matrizes,
(ou Problema Mortal de Matrizes, *Matrix Mortality Problem*)

Dado um conjunto de matrizes $A_1, A_2, \ldots, A_K$, *existe um produto dessas matrizes que seja igual à matriz nula?*

Indecidibilidade: Problema da Mortalidade de Matrizes

Problema da Mortalidade de Matrizes,
(ou Problema Mortal de Matrizes, *Matrix Mortality Problem*)

Dado um conjunto de matrizes $A_1, A_2, \ldots, A_K$, *existe um produto dessas matrizes que seja igual à matriz nula?*

Observações:

- As matrizes são $n \times n$ (só assim qualquer par de matrizes pode ser multiplicado).

Indecidibilidade: Problema da Mortalidade de Matrizes

Problema da Mortalidade de Matrizes,
(ou Problema Mortal de Matrizes, *Matrix Mortality Problem*)

Dado um conjunto de matrizes $A_1, A_2, \ldots, A_K$, *existe um produto dessas matrizes que seja igual à matriz nula?*

Observações:

- As matrizes são $n \times n$ (só assim qualquer par de matrizes pode ser multiplicado).
- Precisamos decidir se existe uma sequência de índices (repetições são permitidas) $i_1, \ldots, i_L$ de forma que o produto $A_{i_1}, \ldots, A_{i_L}$ seja uma matriz de zeros.

Indecidibilidade: Problema da Mortalidade de Matrizes

Teorema

O **Problema da Mortalidade de Matrizes** é indecidível.

Indecidibilidade: Problema da Mortalidade de Matrizes

Teorema

O **Problema da Mortalidade de Matrizes** é indecidível.

Ideia da Prova: A prova consiste em reduzir o Problema da Correspondência de Post para o Problema da Mortalidade de Matrizes: **PCP** $\propto$ **PMM**.

Indecidibilidade: Problema da Mortalidade de Matrizes

Teorema

O **Problema da Mortalidade de Matrizes** é indecidível.

Ideia da Prova: A prova consiste em reduzir o Problema da Correspondência de Post para o Problema da Mortalidade de Matrizes: **PCP** $\propto$ **PMM**.

Observação:

- Se considerarmos apenas matrizes 3×3 o problema continua indecidível.

Indecidibilidade: Problema da Mortalidade de Matrizes

Por que este problema é indecidível?

- Para ter uma noção da sua complexidade, considere apenas duas matrizes 3×3:

$$A = \begin{bmatrix} 0 & 1 & -1 \\ 0 & 1 & -1 \\ 1 & 0 & 1 \end{bmatrix}, \quad B = \begin{bmatrix} -1 & 0 & 1 \\ -1 & -1 & 0 \\ -1 & 1 & 1 \end{bmatrix}.$$

- Será que esse par de matrizes é **mortal** ou **imortal**?
  Há como multiplicar essas matrizes para chegar à matriz nula?

Indecidibilidade: Problema da Mortalidade de Matrizes

Por que este problema é indecidível?

- Para ter uma noção da sua complexidade, considere apenas duas matrizes 3×3:

$$A = \begin{bmatrix} 0 & 1 & -1 \\ 0 & 1 & -1 \\ 1 & 0 & 1 \end{bmatrix}, \quad B = \begin{bmatrix} -1 & 0 & 1 \\ -1 & -1 & 0 \\ -1 & 1 & 1 \end{bmatrix}.$$

- Será que esse par de matrizes é **mortal** ou **imortal**?
  Há como multiplicar essas matrizes para chegar à matriz nula?
- **Resposta**: `SIM`, esse par de matrizes é mortal: $AB^2A^3B^4A^2BAB^2A = 0$.
  Há 17 matrizes nesse produto, e **não há produto menor** que seja nulo.

Indecidibilidade: Problema da Mortalidade de Matrizes

Por que este problema é indecidível?

- Para ter uma noção da sua complexidade, considere apenas duas matrizes 3×3:

$$A = \begin{bmatrix} 0 & 1 & -1 \\ 0 & 1 & -1 \\ 1 & 0 & 1 \end{bmatrix}, \quad B = \begin{bmatrix} -1 & 0 & 1 \\ -1 & -1 & 0 \\ -1 & 1 & 1 \end{bmatrix}.$$

- Será que esse par de matrizes é **mortal** ou **imortal**?
  Há como multiplicar essas matrizes para chegar à matriz nula?
- **Resposta**: `SIM`, esse par de matrizes é mortal: $AB^2A^3B^4A^2BAB^2A = 0$.
  Há 17 matrizes nesse produto, e **não há produto menor** que seja nulo.
- Nenhum algoritmo finito consegue garantir se a minha instância é `SIM` ou `NÃO`.
  Precisaríamos de um *"algoritmo infinito"* para resolver o problema!

Indecidibilidade: outros exemplos

Outros exemplos de problemas indecidíveis:

Decidir se uma fórmula lógica de primeira ordem é válida ou não.

Ou seja, decidir se a fórmula é ou não válida para todas as possíveis realizações das variáveis ou predicados (problema conhecido como *Entscheidungsproblem*).

Indecidibilidade: outros exemplos

Outros exemplos de problemas indecidíveis:

Decidir se uma fórmula lógica de primeira ordem é válida ou não.

Ou seja, decidir se a fórmula é ou não válida para todas as possíveis realizações das variáveis ou predicados (problema conhecido como *Entscheidungsproblem*).

- **Exemplo**: $(\neg\forall x\ P(x)) \Rightarrow (\exists x\ \neg P(x))$, onde $P(x)$ é alguma relação ou propriedade (predicados) de x. Por exemplo, "x pertence ao conjunto P".

Indecidibilidade: outros exemplos

Outros exemplos de problemas indecidíveis:

Decidir se uma fórmula lógica de primeira ordem é válida ou não.

Ou seja, decidir se a fórmula é ou não válida para todas as possíveis realizações das variáveis ou predicados (problema conhecido como *Entscheidungsproblem*).

- **Exemplo**: $(\neg\forall x\ P(x)) \Rightarrow (\exists x\ \neg P(x))$, onde $P(x)$ é alguma relação ou propriedade (predicados) de x. Por exemplo, "x pertence ao conjunto P".

- Veja que o problema é "semelhante" ao **SAT** para a lógica proposicional.

  A lógica proposicional é decidível: o método das tabelas de verdade pode ser utilizado para determinar se uma fórmula proposicional arbitrária é válida.
  A lógica de primeira ordem é, em geral, indecidível.

Indecidibilidade: outros exemplos

Outros exemplos de problemas indecidíveis:

- **Ray Tracing**: em computação gráfica, para um conjunto finito de objetos em 3D, determinar se um raio que sai de uma determinada posição em uma direção atinge um determinado ponto. $\rightarrow$ Computability and Complexity of Ray Tracing

Indecidibilidade: outros exemplos

Outros exemplos de problemas indecidíveis:

- **Ray Tracing**: em computação gráfica, para um conjunto finito de objetos em 3D, determinar se um raio que sai de uma determinada posição em uma direção atinge um determinado ponto. $\rightarrow$ Computability and Complexity of Ray Tracing
- **Game of Life**: dado um estado inicial, determinar se um padrão irá aparecer no Jogo da Vida de Conway $\rightarrow$ Turing Machine Universality of the Game of Life

Indecidibilidade: outros exemplos

Problema:

Dada uma afirmação em linguagem natural, saber se a afirmação é verdadeira.

Indecidibilidade: outros exemplos

Problema:

Dada uma afirmação em linguagem natural, saber se a afirmação é verdadeira.

O que vocês acham, esse problema é decidível?

Resumo

Síntese

- Diversos problemas são **indecidíveis**: não existe nenhum algoritmo para resolver estes problemas.
- O primeiro problema identificado como indecidível foi o **Problema da Parada**, ou *Halting Problem*.
- Não vale a pena investir esforços em resolver um problema indecidível!

Material bibliográfico

Michael Sipser, "Introduction to the Theory of Computation" (2012).

Dúvidas

Dúvidas?