

Aquecendo os motores

1

EDUARDO FREIRE NAKAMURA

Instituto de Computação
Universidade Federal do Amazonas
nakamura@icomp.ufam.edu.br

¹Este material utiliza conteúdo das aulas fornecidas pelo Prof. Vilar da Câmara Neto (disponível em <http://prof.vilarneto.com>).

²Este material utiliza conteúdo das aulas fornecidas pelo Prof. Marco Cristo (marco.cristo@gmail.com).

³Permissão de uso fornecida pelos autores.

⁴As figuras utilizadas neste material são de domínio público, disponíveis na Internet sem informações de direitos autorais.

Fundamentos de teoria da computação

2

- A disciplina está dividida em 3 grandes partes
 - Matemática discreta
 - ▣ Prova de teoremas; relações e funções; conjuntos enumeráveis;
 - ▣ Indução matemática.
 - Linguagens formais e autômatos
 - ▣ Tratada descrição de linguagens formais e da construção de “máquinas” para processá-las
 - ▣ Exemplos: Compiladores, processadores de texto, protocolos de comunicação, jogos
 - Máquinas universais e computabilidade
 - ▣ Estuda problemas que podem ser resolvidos por um computador, mesmo que este tenha memória e velocidade infinitas

Linguagens formais

3

- Nós usamos linguagem natural
- Computadores esperam que os comandos sejam dados de maneira simples e direta



```
leia um valor  
imprima um valor  
se uma condição então execute um comando  
enquanto uma condição for verdadeira faça tal coisa
```

Linguagens formais

4

- Definidas por meio de regras sem ambiguidade

programa → *Lista de comandos*

comando → $\begin{cases} \text{leia} \\ \text{imprima} \\ \text{faça} \\ \dots \end{cases}$

leia → "leia" variável

imprima → "imprima" expressão

faça → "faça" variável "←" expressão

variável → ...

expressão → ...

Linguagens formais

5

- Com o conjunto de regras, é possível analisar um programa e quebrá-lo em seus blocos

leia x

comando **leia**, variável x

leia y

comando **leia**, variável y

faça $z \leftarrow x + y$

comando **faça**, variável z , expressão $x + y$

imprima z

comando **imprima**, expressão z

Linguagens formais

6

- ... ou detectar erros

leia *x*

leia *y*

faça *z* \leftarrow **imprima**



*O comando **faça** requer uma expressão, mas foi encontrado o comando “**imprima**”*

Outro exemplo

7

expressão → {

- literal**
- expressão "+"** expressão
- expressão "-"** expressão
- expressão "*" expressão**
- expressão "/" expressão**
- "_"** expressão
- (" expressão ")**

literal é uma sequência de “0” a “9”, possivelmente com um ponto decimal no meio

Como formalizar esta definição?

Questões comuns em linguagens formais

8

- Como formalizar uma linguagem?
 - Como definir formalmente um conjunto de regras?
 - Como definir sequências de dígitos, letras, repetições, etc.?
- É possível descobrir se as regras possuem ambiguidade?
- Existe uma classificação para as linguagens? Há linguagens “mais simples” ou “mais complexas”?
- Como construir reconhecedores para as linguagens? Existem reconhecedores eficientes para as linguagens “mais complexas”?

Máquinas universais e computabilidade

9

- Estudo de modelos matemáticos que representam computadores
- Tais modelos não se preocupam com a eficiência dos programas, apenas com o que **podem** ou **não fazer**
- Qual é o melhor computador concebível?
 - Velocidade infinita
 - Memória infinita
- O que ele pode fazer? Há problemas que ele não pode resolver?



Computabilidade

10

- **Computabilidade** não se preocupa com a eficiência!
 - “Se alguma coisa pode ser feita por um computador, então depois descobrimos como fazer melhor!”
 - “Se alguma coisa não pode ser feita, então não adianta gastar tempo...”



Pode ser feita ou não?
É computável?

Ilustrando três problemas

11

- Busca
- Caixeiro Viajante
- Correspondência de Post

Problema de busca

12

- Encontre um número em uma lista de números

- Entradas: 45

| | |
|---|----|
| 0 | 12 |
| 1 | 35 |
| 2 | 44 |
| 3 | 02 |
| 4 | 17 |
| 5 | 45 |
| 6 | 66 |

- Saída: posição 5

Problema do caixeiro viajante

13

- Dada uma lista de cidades e as distâncias entre elas, encontre o menor caminho para que se possa visitar cada cidade exatamente uma vez

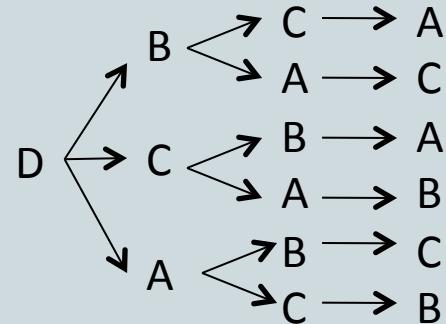
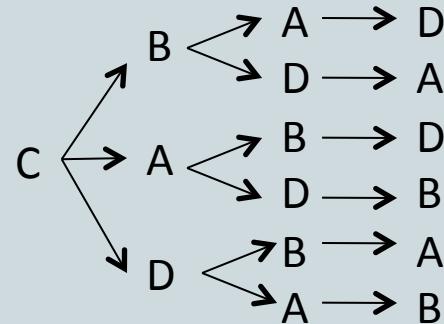
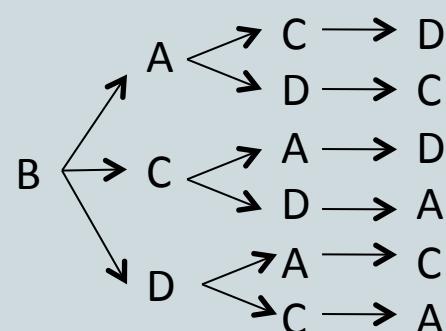
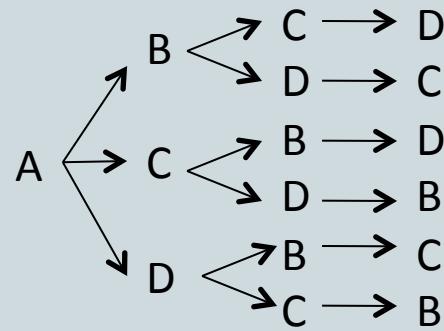
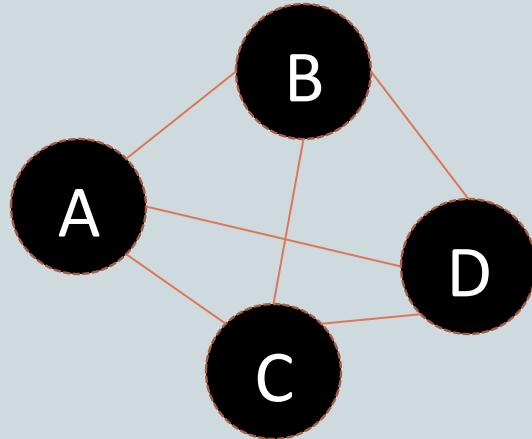


Problema do caixeiro viajante

14

- Quatro cidades

$$n! = 4! = 4 \times 3 \times 2 \times 1 = 24 \text{ rotas possíveis}$$



Problema do caixeiro viajante

15

- Trinta cidades
 - $30 \text{ cidades} = 30! = 2,65 \times 10^{32}$
 - Máquina capaz de 1 trilhão de adições / segundo:
 - ✖ 252.333.390.232.297 anos

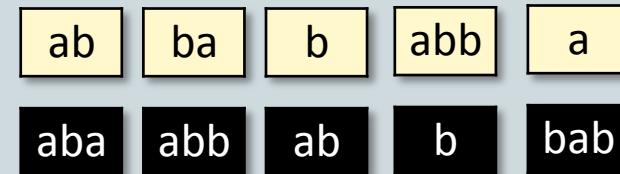
Problema da correspondência de Post

16

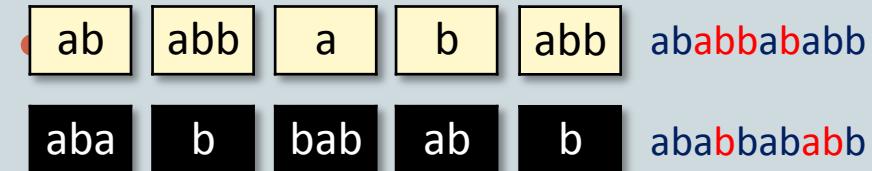
- Imagine um dominó com duas metades m_1 e m_2
- m_1 e m_2 são as palavras extraídas de dois alfabetos A_1 e A_2
- Arranjos usando m_1 e m_2
 - A string formada pelas peças de m_1 é idêntica à formada pelas peças de m_2
 - Repetições são permitidas

- Entrada:

- $m_1 = \{ab, ba, b, abb, a\}$
- $m_2 = \{aba, abb, ab, b, bab\}$
- Dominós:



- Saída:



Teoria da computação e os três problemas

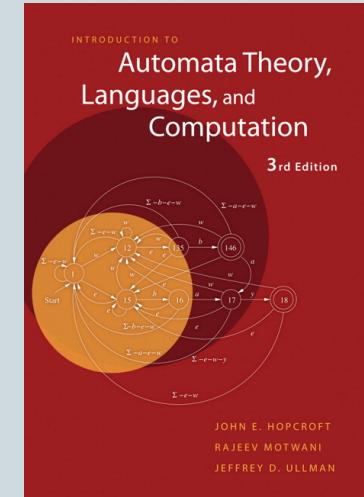
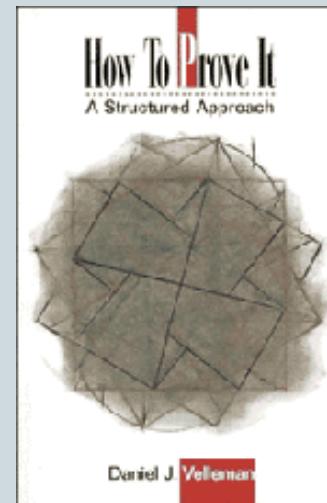
17

- Teoria da Computação
 - É computável? (tem solução?)
- Teoria da Complexidade
 - Quantos recursos (tempo e espaço) são necessários?
- Exemplos
 - **Busca** → Computável com Custo Polinomial
 - **Caixeiro viajante** → Computável com Custo Fatorial
 - **Correspondência de Post** → Não Computável

Bibliografia

18

- Roteiro
 - Matemática discreta
 - Teoria da Computação
 - ▣ Máquinas de Estado Finito
 - ▣ Autômatos com Pilha
 - ▣ Máquinas de Turing
 - ▣ Decidibilidade



Bibliografia

19

- Vieira, N.J. Introdução aos Fundamentos da Computação: Linguagens e Máquinas, Pioneira Thomson Learning (atualmente, CENGAGE), 2006.
- Hopcroft, J.E., Motwani, R., Ullman, J.D. Introduction to Automata Theory, Languages, and Computation, 3rd ed., Addison-Wesley, 2006.
- Velleman, D. J. How to Prove It: A Structured Approach

