Relatório de Desenvolvimento da Lista de Exercício 1

Bruno Da Fonseca Chevitarese (20231BSI0082)

Marcos Vinícius Souza dos Santos (20222BSI0156)

Outubro 2024

Sumário

1 Apresentação

Este documento tem por objetivo contemplar a segunda lista de exercícios propostos na Disciplina Linguagens Formais e Autômatos, lecionada pelo professor Doutor Jefferson de Oliveira Andrade, ofertada no Instituto de Ensino, Ciências e Tecnologia do Espírito Santo (Ifes), *campus* Serra. As seções seguintes contêm as respostas para os problemas propostos de 1 a 4.

Cabe destacar que todos os problemas que necessitam de uma implementação da solução, possuem o mesmo padrão de diretórios e projetos. Portanto, os *scripts* da solução do problema "x" encontram-se no caminho "/problemas/prblx/". Ademais, os códigos podem estar fragmentados em múltiplos arquivos por questões de organização e legibilidade.

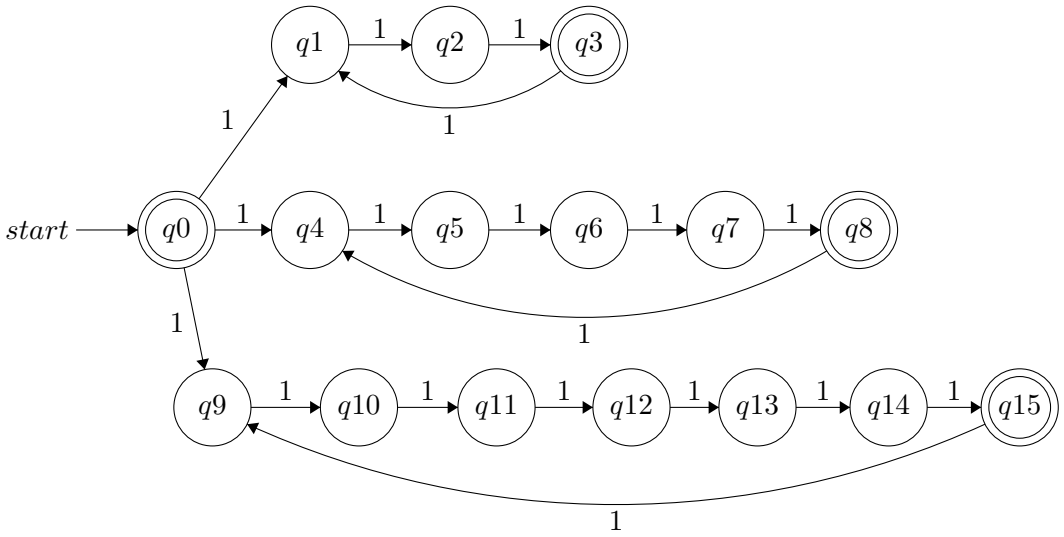
2 Múltiplos de 3, 5 ou 7

2.1 Enunciado

Construa um NFA_ε que reconheça a linguagem $L_1 = \{\, 1^n \mid n \text{ é múltiplo de } 3, 5 \text{ ou } 7 \,\}$.

2.2 Solução Proposta

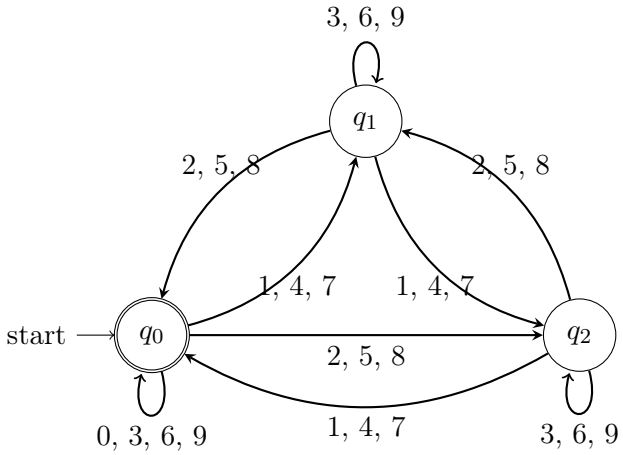
O problema consiste em verificar se o número de caracteres em uma cadeia é múltiplo de 3, 5 ou 7. Para resolver isso, o autômato é composto por três subautômatos, cada um responsável por avaliar a divisibilidade do comprimento da cadeia por um desses valores. Como se trata de um autômato não determinístico, ao ler o primeiro caractere no estado inicial, ele pode seguir simultaneamente três caminhos distintos, iniciando os fluxos de contagem para 3, 5 e 7. A cadeia será aceita se, ao final da leitura, o autômato alcançar um estado correspondente a um múltiplo de pelo menos um desses números.



3 O Problema do Autômato não Identificado

3.1 Enunciado

Descreva, em português, que linguagem é reconhecida pelo autômato M2 abaixo:



3.2 Solução

O Autômato exposto acima busca identificar números múltiplos de 3. O modo como ele faz isso será descrito abaixo.

O critério de divisibilidade por três estabelece que: dado um número x constituído por algarismos $ABC...$, com $ABC... \in \mathbb{N}$, $x \equiv 0(MOD3)$ se, e somente se, $A + B + C... \equiv 0(MOD3)$. Pelo princípio da compatibilidade com a translação e compatibilidade com a subtração, tem-se que $\forall x, y, z, w \in \mathbb{Z} \backslash x \equiv y(MODw) \wedge 0 < y < w \implies z + x \equiv y + z(MODw)$.

Isto posto, a partir de qualquer estado que o autômato esteja pode-se afirmar que o subsequente será igual para todos os dígitos que são côngruos no módulo por três. Assim, por exemplo, a partir do estado inicial os algarismos $1, 4$ e 7, os quais são côngruos entre si no módulo por 3, transitam para o mesmo estado.

Por fim, repare que cada estado do autômato representa os 3 menores resultados da divisão modular por 3, $q0 = 0, q1 = 1$ e $q2 = 2$. Assim, somente quando a soma do menor resultado dos módulos de cada algarismo do número fornecido for 0 é que o estado é passível de ser finalizado.

O automato acima possivelmente possui um erro. Note que sempre que a transição de estados com os valores $3, 6$ e 9 deveriam conter também o algarismo 0, no entanto isso não ocorreu. A consequência disso é que diversos números como 204, o qual é múltiplo de 3, não seja reconhecido.

4 Regex Crossword

4.1 Parte I

Capturas referentes aos puzzles resolvidos:

Figura 1: Tela inicial.

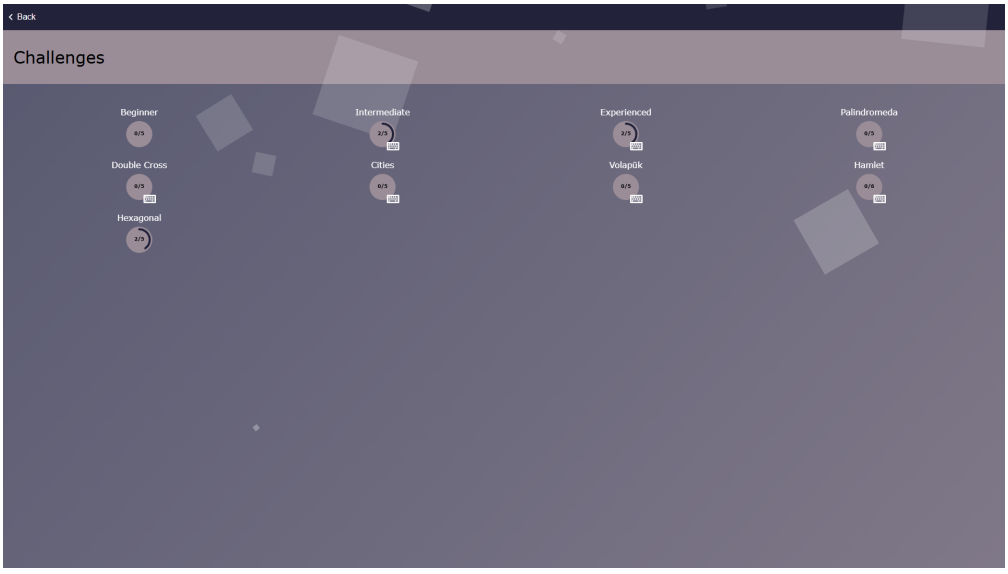


Figura 2: Desafios cumpridos.

4.2 Parte II

Puzzle criado por nós: LFMA - Marcos e Bruno

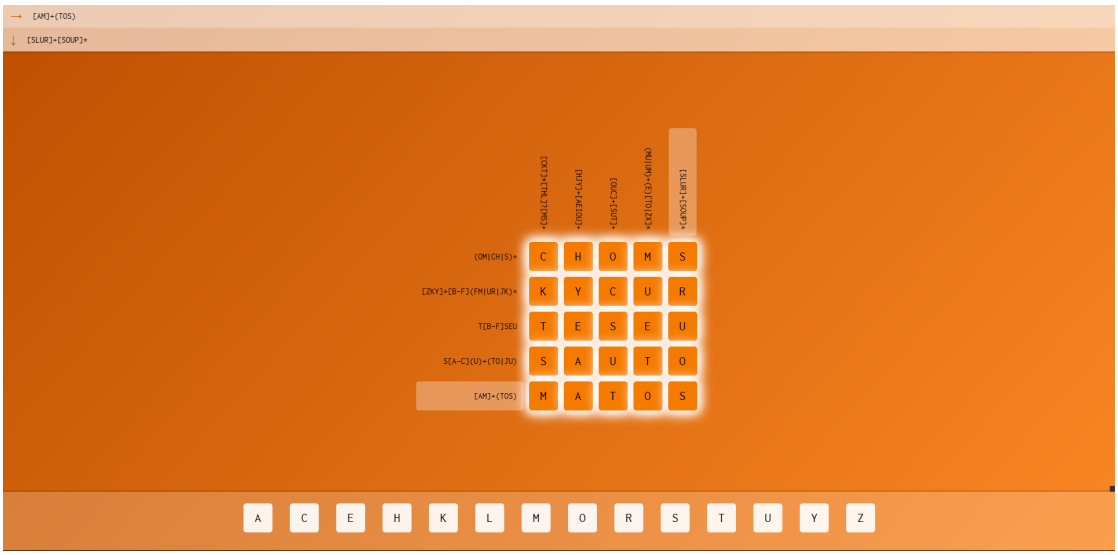


Figura 3: Puzzle criado pela dupla.

5 Regex Compiler

5.1 Enunciado

Escreva um programa que seja capaz de ler uma expressão regular pura e gerar o autômato finito com movimentos vazios correspondente.

5.2 Solução Proposta

Para maior facilidade de representar as abstrações, escolhemos o Java, uma linguagem orientada a objetos.

Para representar os autômatos, criamos Classes que representassem suas estruturas básicas. Cada 'estado' é um Elemento, que possui um nome e uma lista de Conexões. Conexão é uma Classe que possui como atributos um símbolo (que representa o caractere que precisa ser lido para realizar a transição) e um Elemento destino. No programa, os autômatos são representados pela classe Sentença, que possui um Elemento Inicial e um Elemento Final, junto a métodos para aplicar as operações de união, concatenação e estrela.

Para ler os caracteres, foi criada a função Parser, que possui métodos que leem a entrada, empilham os valores e as operações identificadas e as executa seguindo a precedência. Ao final, imprime o automato resultante, informando o estado inicial, os estados de aceitação e a função de transição.

Implementação completa pode ser encontrada no diretório /problemas/probl4.