Ministério da Educação
Universidade Tecnológica Federal do Paraná
Campus Apucarana

Engenharia De Computação

TCCO5A- Teoria da Computação

Mecanismo de reconhecimento de linguagens regulares com o Lema do Bombeamento

JOÃO VITOR NAKAHODO YOSHIDA

THIAGO CRISTOVÃO DE SOUZA

Apucarana-PR, 2023

JOÃO VITOR NAKAHODO YOSHIDA
THIAGO CRISTOVÃO DE SOUZA

Mecanismo de reconhecimento de linguagens regulares com o Lema do Bombeamento

Relatório Técnico do Trabalho Disciplinar apresentado como requisito parcial à obtenção de nota na disciplina de Teoria da Computação do Curso Superior de Engenharia de Computação da Universidade Tecnológica Federal do Paraná.

Orientador: Prof. Lucio Agostinho Rocha

Apucarana-PR, 2023

UTFPR
UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ

SUMÁRIO

1. INTRODUÇÃO

A Teoria da Computação é um ramo da matemática e da ciência da computação que examina a natureza e os limites da computação. Ela lida com problemas teóricos fundamentais, como a classificação de problemas como decidíveis ou indecidíveis, a análise de complexidade de algoritmos e a investigação de máquinas abstratas como as Máquinas de Turing. Compreender a Teoria da Computação é fundamental para o desenvolvimento de sistemas computacionais eficientes e confiáveis. Ao entender as limitações e os desafios teóricos, os engenheiros de software podem projetar algoritmos mais eficazes, otimizar o desempenho e antecipar possíveis obstáculos no desenvolvimento de software.

Ao longo do semestre, adquirimos conhecimentos teóricos e práticos sobre os princípios e técnicas dessa metodologia. Este relatório apresenta os detalhes e os resultados do nosso projeto final de Teoria da Computação, cujo objetivo foi desenvolver um programa que utilizasse os conceitos do lema do bombeamento para reconhecer se uma dada linguagem é regular ou não. Foi utilizada a linguagem de programação Python e exploramos diversas funcionalidades para criar uma solução eficiente e completa.

2. LEMA DO BOMBEAMENTO

O princípio do bombeamento afirma que, para qualquer linguagem regular, existe um comprimento mínimo, chamado comprimento de bombeamento (representado por *p*), tal que qualquer cadeia *w* pertencente à linguagem, com um comprimento maior ou igual a *p*, pode ser decomposta em três partes, denotadas por *x*, *y* e *z*, de acordo com as seguintes condições:

1.$|xy| \leq p$ (O comprimento total de xy não ultrapassa p)
2. $|y| \geq 1$ (A subcadeia y tem pelo menos um caractere)
3. para todo $i \geq 0$, $x\ y^i\ z \in L$ (Para qualquer número inteiro não negativo i, a concatenação de x, y repetida i vezes e z ainda pertence à linguagem)

Se para um dado valor de *p* a linguagem não conseguir atender esses 3 critérios, podemos afirmar que ela é uma linguagem não regular. Essencialmente, isso implica que para linguagens regulares, é possível "bombardear" a subcadeia y, removendo-a ou repetindo-a arbitrariamente, e ainda assim obter uma cadeia que permanece na linguagem.

O Lema do Bombeamento proporciona uma metodologia sistemática para verificar a regularidade de linguagens formais. Ele destaca a limitação das linguagens regulares na Teoria da Computação, demonstrando que certas

linguagens não podem ser reconhecidas por máquinas de estados finitos.

3. DESENVOLVIMENTO

Para o projeto foram desenvolvidos dois algoritmos de reconhecimento de linguagens, um para linguagens regulares e outro para não regulares a partir da linguagem de programação Python, por meio da IDE PyCharm. Os códigos podem ser acessados pelo link: https://github.com/JoaoVitorNY/Teoria-da-Computacao.git

O primeiro algoritmo implementa uma simulação do Lema do Bombeamento para a linguagem L1={a^nb^n | n≥0}, para verificar se a linguagem é regular, além disso, é preciso saber que o algoritmo divide a palavra “w” em três subpalavras “xyz” ou “uvz”.

De início, o usuário deve passar três parâmetros para que o algoritmo funcione, primeiro temos o comprimento do bombeamento “p” que posteriormente será utilizado para criar uma palavra com base na linguagem L1, onde “n” assume valor de “p”. Segundo temos o “j” que seria a quantidade de elementos de “v” que serão submetidos ao bombeamento e por último é passado o “k”, a quantidade de vezes que o bombeamento será aplicado.

Figura 1: Entrada dos valores

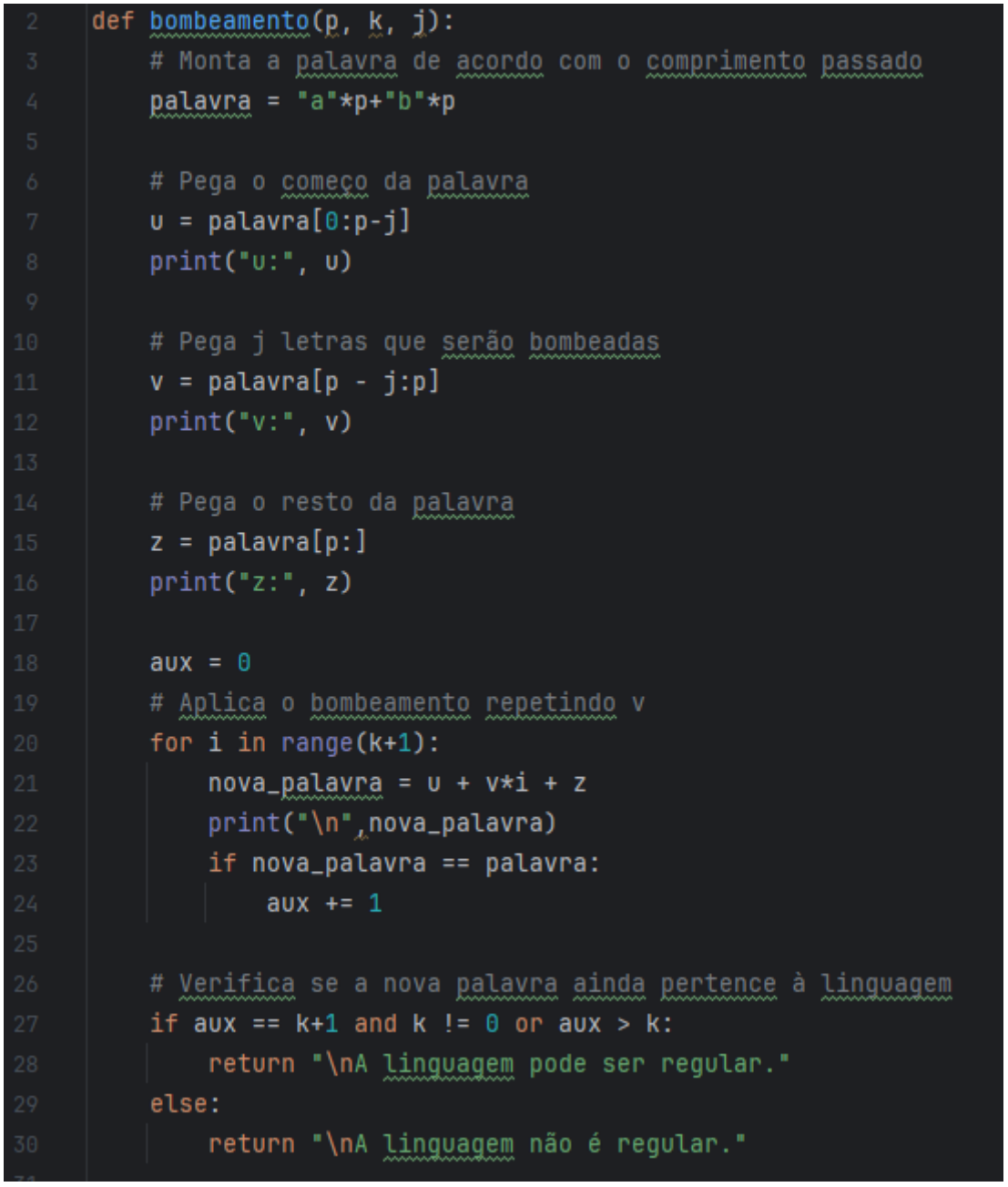
```
41  # Condições: |uv| <= p - o comprimento da parte que pode ser "bombeada" é limitado pelo comprimento de bombeamento.
42  # Condições: |v| > 0 - a parte v não pode ser vazia.
43  # Tamanho de v não pode ser maior que p (comprimento do bombeamento)
44  while(1):
45      # Tratamento para palavra vazia
46      if p == 0:
47          j = int(input("Digite o tamanho da divisão de v: "))
48          break
49
50      j = int(input("Digite o tamanho da divisão de v: ")) # Esse valor significa a quantidade de elementos de 'v' que serão submetidos ao bombeamento
51      if j > p or j <= 0:
52          continue
53      else:
54          break
55
56  while(1):
57      k = int(input("Digite a quantidade de bombeamento k: "))
58      if k < 0:
59          continue
60      else:
61          break
```

Fonte: Autoria Própria (2023).

As entradas possuem tratamento de erro de acordo com as condições do lema do bombeamento.

Para a lógica do algoritmo, quando passado o valor do comprimento do bombeamento “p”, a função bombeamento é responsável por gerar a palavra assumindo a^nb^n como a^pb^p para p≥0. A partir da palavra gerada, é feita a divisão da mesma em três subpalavras, “u” recebe as letras da posição 0 até p-j da string, “v” recebe de p-j até p e “z” recebe o restante.

Figura 2: Divisão da palavra



```
def bombeamento(p, k, j):
    # Monta a palavra de acordo com o comprimento passado
    palavra = "a"*p+"b"*p

    # Pega o começo da palavra
    u = palavra[0:p-j]
    print("u:", u)

    # Pega j letras que serão bombeadas
    v = palavra[p - j:p]
    print("v:", v)

    # Pega o resto da palavra
    z = palavra[p:]
    print("z:", z)

    aux = 0
    # Aplica o bombeamento repetindo v
    for i in range(k+1):
        nova_palavra = u + v*i + z
        print("\n",nova_palavra)
        if nova_palavra == palavra:
            aux += 1

    # Verifica se a nova palavra ainda pertence à linguagem
    if aux == k+1 and k != 0 or aux > k:
        return "\nA linguagem pode ser regular."
    else:
        return "\nA linguagem não é regular."
```

Fonte: Autoria Própria (2023).

Após a divisão da palavra, é aplicado o bombeamento repetindo “v” até “k” vezes. Depois é verificado se as novas palavras geradas pertencem à linguagem L1, caso uma das novas palavras não pertença à linguagem, dizemos que a linguagem não é regular.

O segundo algoritmo implementa uma simulação do Lema do Bombeamento para a linguagem L2={a^nb^m | m, n≥0} para verificar se a linguagem é regular. Nesse algoritmo, ao invés de passar a palavra, o usuário passa a divisão de uvz.

De início, o usuário deve passar sete parâmetros para que o algoritmo

funcione, o primeiro termo (símbolo de u), índice do primeiro termo (potência de u), segundo termo (símbolo de v), índice do segundo termo (potência de v), terceiro termo (símbolo de z), índice do terceiro termo (potência de z) e o valor que será usado para o símbolo p.

Após receber os dados do usuário, o código realiza uma série de verificações, desde se os símbolos inseridos são aceitos pela linguagem, até o tratamento de cada possível caso para os índices dos termos inseridos. Após verificar as entradas, o algoritmo começa a realizar os teste para cada critério do lema do bombeamento:

1- se índice do termo u + índice do termo v ≤ p

2 - se índice do termo v > 0

3 - se para cada valor de i, u v^i z faz parte da linguagem

Após serem feitos esses testes, o programa gera a palavra utilizando o valor de p inserido pelo usuário, e verifica se essa palavra faz parte da linguagem.

4. CASO DE TESTE

Para o caso de teste do primeiro algoritmo com linguagem L1 = {a^nb^n | n≥0}

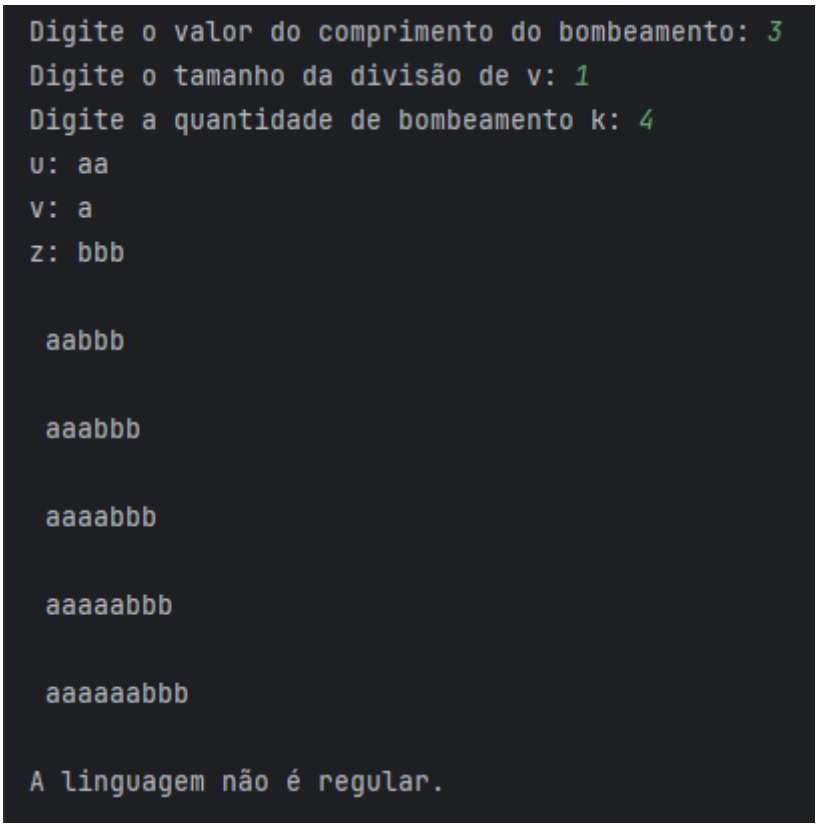
Figura 3: Casos de teste da L1

```
# Casos de teste
"""
Caso 1:
    p = 0
    j = qualquer
    k = qualquer
    resultado = palavra vazia é reconhecida pela linguagem

Caso 2:
    p > 0
    1 <= j <= p
    k = 1
    resultado = linguagem não é regular

Caso 3:
    p > 0
    1 <= j <= p
    k > 1
    resultado = linguagem não é regular
"""
```

Fonte: Autoria Própria (2023).

Realizando um teste para p = 3, j = 1, k = 4, obtemos:

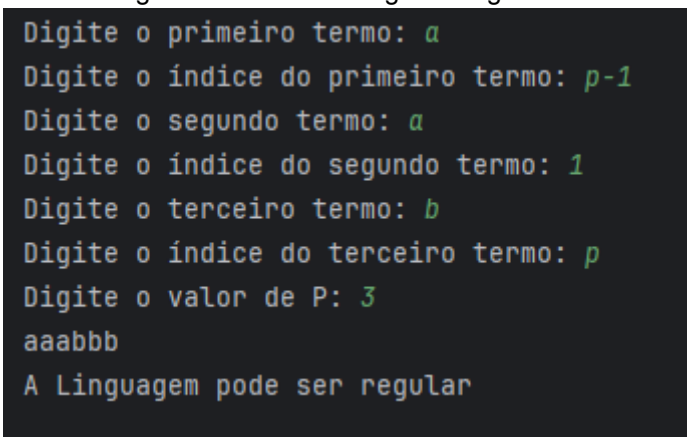
Figura 4: Teste do Caso 3



Fonte: Autoria Própria (2023).

Para o caso de teste do segundo algoritmo com linguagem L2 = {a^nb^m ∣ m, n≥0}, usando valores de entrada a^p-1, a^1 b^p / p = 3, obtemos:

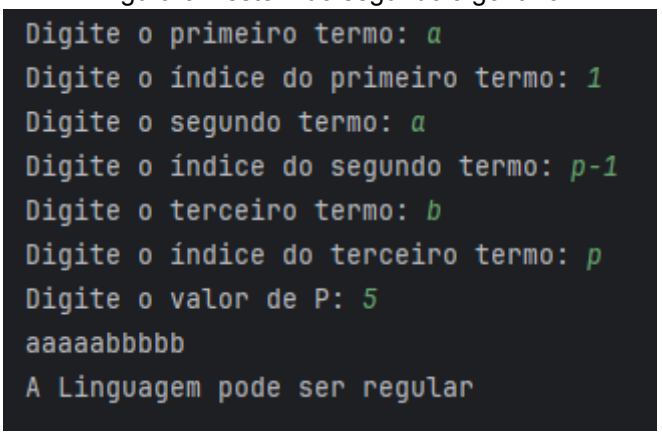
Figura 5: Teste 1 do segundo algoritmo



Fonte: Autoria Própria (2023).

Usando valores de entrada a^1 a^p-1 b^p / p = 5, obtemos:

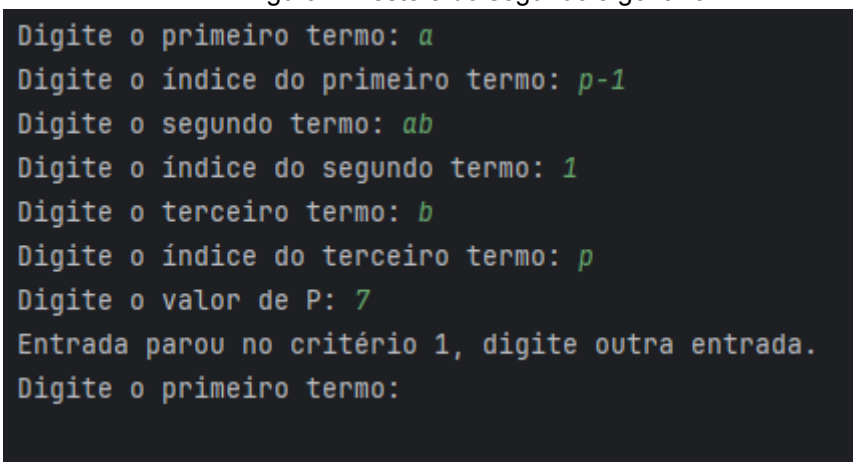
Figura 6: Teste 2 do segundo algoritmo



Fonte: Autoria Própria (2023).

No último teste, usando valores de entrada a^p-1 ab^1 b^p / p = 7, obtemos:

Figura 7: Teste 3 do segundo algoritmo



Fonte: Autoria Própria (2023).

5. CONCLUSÃO

Neste relatório técnico, apresentamos os detalhes e resultados do nosso projeto final de Teoria da Computação. Este trabalho nos permitiu aprimorar nossas habilidades em programação, modelagem de sistemas e trabalho em equipe. Através do desenvolvimento desse projeto, fomos capazes de aplicar os conhecimentos adquiridos ao longo do curso, enfrentar desafios e buscar soluções criativas.

Em resumo, o projeto final de Teoria da computação foi uma experiência enriquecedora que nos proporcionou a oportunidade de consolidar nosso aprendizado e demonstrar nossas habilidades no desenvolvimento de sistemas de reconhecimento de linguagens.

6. REFERÊNCIAS

TCCO5A - 2s2023, 2023. Disponível em: https://classroom.google.com/u/4/c/NTg5MTE5MTk1OTY1. Acesso em: 04 dezembro 2023