Trabalho Final GCC-108 - Teoria da Computação

Prof.: Douglas H. S. Abreu

Nome: Gustavo Costa Daguer e Vitor André de Oliveira Tenório

Turma: 10A

Link do repositório GitHub (https://github.com/VitorAndre/Trabalho_Final_GCC108)

O código se encontra após as observações abaixo e no GitHub com o nome de soma.py. Os exercícios estão em um PDF chamado TrabalhoFinal_GustavoDaguer_VitorTenorio.pdf.

● O trabalho deve ser feito em grupos de no máximo 2 componentes

● Trabalhos entregues após a data limite não serão aceitos

● Data limite de entrega: 29 de Abril de 2022 : 23h59m

● Enviar o trabalho para o campus virtual, do seguinte modo: Notebook exportado em PDF contendo o código e também o link do repositório GitHub para acesso aos arquivos. A Documentação deve estar no readme

● O trabalho deve ser desenvolvido no modelo Notebook utilizando a linguagem Python

---

In [ ]:
```
class Soma:
    def __init__(self):
        self.fita = ['B']
        self.posCabecote = 0 #posição do cabeçote

    def soma(self, num1, num2):
        for i in list(num1):
            self.fita.append(i)

        self.fita.append('B')

        for i in list(num2):
            self.fita.append(i)

        self.fita.append('B')
        self.q0()

    def q0(self):
        if (self.fita[self.posCabecote] == 'B'):
            self.posCabecote += 1
            self.q0()
        elif self.fita[self.posCabecote] == '1' or self.fita[self.posCabeco
            self.posCabecote += 1
            self.q1()

    def q1(self):
```

```
        if self.fita[self.posCabecote] == 'B':
            self.posCabecote += 1
            self.q2()
        else:
            self.posCabecote += 1
            self.q1()

    def q2(self):
        if (self.fita[self.posCabecote] == 'B'):
            self.posCabecote -= 1
            self.q3()
        elif(self.fita[self.posCabecote] == '0' or self.fita[self.posCabeco
            self.posCabecote += 1
            self.q2()

    def q3(self):
        if (self.fita[self.posCabecote] == '0'):
            self.fita[self.posCabecote] = 'B'
            self.posCabecote -= 1
            self.q4()
        elif self.fita[self.posCabecote] == '1':
            self.fita[self.posCabecote] = 'B'
            self.posCabecote -= 1
            self.q6()
        elif self.fita[self.posCabecote] == 'B':
            self.posCabecote -= 1
            self.q9()

    def q4(self):
        if (self.fita[self.posCabecote] == 'B'):
            self.posCabecote -= 1
            self.q5()
        elif(self.fita[self.posCabecote] == '0' or self.fita[self.posCabeco
            self.posCabecote -= 1
            self.q4()

    def q5(self):
        if (self.fita[self.posCabecote] == 'X' or self.fita[self.posCabecot
            self.posCabecote -= 1
            self.q5()
        elif(self.fita[self.posCabecote] == '0'):
            self.fita[self.posCabecote] = 'X'
            self.posCabecote += 1
            self.q1()
        elif self.fita[self.posCabecote] == '1':
            self.fita[self.posCabecote] = 'Y'
            self.posCabecote += 1
            self.q1()

    def q6(self):
        if (self.fita[self.posCabecote] == 'B'):
            self.posCabecote -= 1
            self.q7()
        elif(self.fita[self.posCabecote] == '0' or self.fita[self.posCabeco
            self.posCabecote -= 1
            self.q6()

    def q7(self):
        if (self.fita[self.posCabecote] == '1'):
            self.fita[self.posCabecote] = 'X'
            self.posCabecote -= 1
            self.q8()
        elif self.fita[self.posCabecote] == '0':
            self.fita[self.posCabecote] = 'Y'
```

```
            self.posCabecote += 1
            self.q1()
        elif(self.fita[self.posCabecote] == 'X' or self.fita[self.posCabeco
            self.posCabecote -= 1
            self.q7()

    def q8(self):
        if (self.fita[self.posCabecote] == '1'):
            self.fita[self.posCabecote] = '0'
            self.posCabecote -= 1
            self.q8()
        elif(self.fita[self.posCabecote] == 'B'):
            self.fita.insert(0, 'B')
            self.posCabecote += 1
            self.fita[self.posCabecote] = '1'
            self.posCabecote += 1
            self.q1()
        elif self.fita[self.posCabecote] == '0':
            self.fita[self.posCabecote] = '1'
            self.posCabecote += 1
            self.q1()

    def q9(self):
        if (self.fita[self.posCabecote] == 'B'):
            self.q10()
        elif(self.fita[self.posCabecote] == '0' or self.fita[self.posCabeco
            self.posCabecote -= 1
            self.q9()
        elif(self.fita[self.posCabecote] == 'X'):
            self.fita[self.posCabecote] = '0'
            self.posCabecote -= 1
            self.q9()
        elif self.fita[self.posCabecote] == 'Y':
            self.fita[self.posCabecote] = '1'
            self.posCabecote -= 1
            self.q9()

    def q10(self):
        print(''.join(self.fita))


m = Soma()
num1 = input("Primeiro numero: ")
num2 = input("Segundo numero: ")
if (len(num2) > len(num1)):
    num1, num2 = num2, num1
m.soma(num1, num2)
```

Introdução

Este trabalho propõe a utilização de operações da aritmética computacional por meio de uma Máquina de Turing. A máquina que foi desenvolvida recebe como entrada dois números em binário e gera como saída o resultado da adição desses números.

Números binários e adição em números binários

Os números binários são utilizados para representar dados em um meio digital, como por exemplo, a representação no meio analógico com presença ou ausência de carga elétrica e no meio digital por meio de zeros e uns. Essa representação com dois símbolos utiliza-se

da mesma técnica do telégrafo, que transmitia mensagens por código Morse, sendo os símbolos curto e longo análogos ao zero e um (1).

Utilizando-se a notação binária é possível representar uma faixa de valores diferentes de acordo com a quantidade de bits. Por exemplo, com dois bits pode-se representar quatro valores distintos, sendo eles 00, 01, 10 e 11. Ou seja, com n bits, podemos representar 2n valores distintos.

Para a notação de números inteiros usando a base binária de zeros e uns, podemos representar os números utilizando as seguintes representações: de binário puro, de binários em sinal magnitude e a representação em complemento de 2 (1).

Tomando como base a representação de números inteiros na base binária pura, que também é a representação utilizada neste trabalho, pode-se observar na Tabela 1, que com quatro bits temos as seguintes possibilidades para números inteiros.

Binário	Decimal	Binário	Decimal
0000	0	1000	8
0001	1	1001	9
0010	2	1010	10
0011	3	1011	11
0100	4	1100	12
0101	5	1101	13
0110	6	1110	14
0111	7	1111	15

Tabela 1. Comparação da base binário de 4 bits com base decimal

As operações matemáticas de adição e subtração feitas na base binária seguem as mesmas regras da base decimal, contando com a diferença que temos apenas dois dígitos. Para a adição de dois números, temos quatro possibilidades de valores, sendo elas: 0 + 0, 0 + 1, 1 + 0, e 1 + 1. As três primeiras têm os mesmos resultados de uma operação em decimal, já para a operação de 1 + 1 temos como resultado zero, gerando um “vai um” para a coluna da esquerda (1).

Máquina de Turing

Turing descreve um computador digital como sendo formado por: uma unidade de armazenamento, uma unidade de execução e uma unidade de controle. A unidade de armazenamento é formada por uma fita, dividida em células, com um cabeçote apontando para a célula atual, a qual pode ser lida/escrita de acordo com a unidade de execução. Por sua vez, a unidade de execução tem como objetivo fazer a leitura do caractere representado na célula atual, analisar o que deve ser feito e alterar quando necessário. Já

a unidade de controle faz as movimentações do cabeçote de acordo com o que a unidade de execução deseja, movendo o cabeçote para esquerda ou direita (2).

Código da soma binária:

Exercício 1)

Descreva com suas palavras uma estratégia para o desenvolvimento de uma maquina de Turing que compute a soma de 2 numeros binário.

Exercício 2)

Faça o esboço por meio de desenho da máquina de Turing proposta.

Exercício 3)

Defina a MT como uma quíntupla M=(Q,Σ,Γ,δ,q_0):

```
Q = conjunto de estados (padrão q[0-9]+)

Σ = alfabeto de entrada

Γ = alfabeto da fita

δ = função de transição no formato (q_i,x)→(q_j,y,D); assim,
estando no estado q_i, lendo x, vai para o estado q_j,
escreve y e movimenta na direção de D. D será L para
esquerda ou R para direita.

q_0 = estado inicial
```

Exercício 4)

Faça a conversão de M em R(M)

Exercício 5)

Desenvolva uma função MTU que receba R(M) acrescido de uma entrada w, onde w é um arquivo csv que contem dois números binário. A saída da função MTU deve ser a computação de M para uma entrada w.

Exercício 6)

A) Explique a Tese de Chuch-Turing de forma sucinta

B) Dada uma máquina de Turing arbitrária M e uma string de entrada w, a computação de M com entrada w irá parar em menos de 100 transições? Descreva uma máquina de Turing que resolva esse problema de decisão.

C) Motre a solução para cada um dos seguintes sitemas de correspondência de Post:

a) (a, aa), (bb, b), (a, bb)

b) (a, ab), (ba, aba), (b, aba), (bba, b)

c) (abb, ab), (aba, ba), (aab,a bab)

d) (ab,aba), (baa, aa), (aba, baa)

e) (a, aaa), (aab, b), (abaaa, ab)

f) (ab, bb), (aa, ba), (ab, abb), (bb, bab)

D)

a) Prove que a função é primitiva recursiva

$$f(x_1, \ldots, x_n,) = \mu z^{u(x_1, \ldots, x_n)} [p(x_1, \ldots, x_n, z)]$$

sempre que *p* e *u* são recursivas primitivas

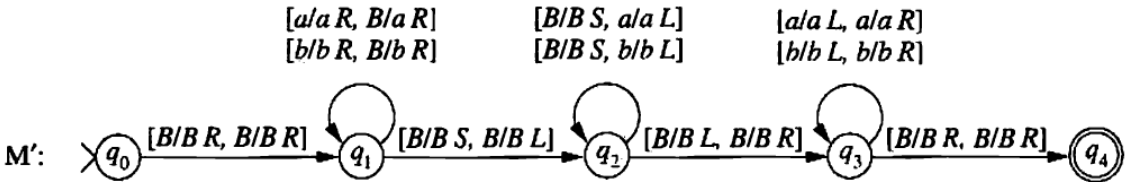
b) defina o valor "passo a passo" de *gn(4,1,0,2,1)* =

E)

a) Dado $f(x) = 3x^2 + 4x + 6$ e $g(x) = 5x^2$

Prove que $g(x) \in O(f)$ e $f(x) \in O(g)$

b) Qual é a complexidade e o "*big O"* de M'?



Referências

(1) *Ronald. J. Tocci, Neal. S. Widmer e Gregory L. Moss. 2011. Sistemas Digitais: Princípios e Aplicações. (11ª ed.). Pearson.*

(2) *Alan Turing. 1937. Computability and λ-definability. Journal of Symbolic Logic, 2, 4: 153–163.*

(3) *Sudkamp, T. A. 2006. Languages and machines: an introduction to the theory of computer science. 3rd Edition*

In [ ]:
```
# bla bla bla
x= "sala"
x
```

In [ ]:
```
# função MTU = {R(M)| R(M) aceite w}

def MTU ()
```

In [ ]:
```
# bla bla bla
```

Links úteis:

Link do site Jupyter

Link do site Anaconda

Link para ajuda com Markdown no Notebook

In [ ]:
```
import pandas as pd
exemplo = pd.read_csv('exemplo2.CSV')
exemplo
```

Out[ ]:

	1011000;1101

In [ ]:
```
txt = "000111010100111011001111001111111010101101000111111011111"

x = txt.split("000")
print(x)
y = txt.split("00")
print(y)
print(len(y[2]))
```

```
['', '1110101001110110011110011111111010101101', '1111111011111']
['', '01110101', '111011', '1111', '1111111010101101', '01111111011111']
6
```

$$\prod f(x) = x^6 + 3 + y \times \begin{pmatrix} aabb \\ bbaaa \end{pmatrix}$$

In [ ]:

In [ ]:

In [ ]: