Trabalho Final GCC-108 - Teoria da Computação

Prof.: Douglas H. S. Abreu

Nome: Gustavo Ribeiro da Fonseca Santos, Layra Vilas Boas Ferreira

Turma: 10A

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

- 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

---

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).

Exercício 1)

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

A máquina de turing que computa a soma de 2 números binários, deve receber os dois números separados por um símbolo que indica um espaço em branco "□". Com isso, devemos ter os seguintes estados:

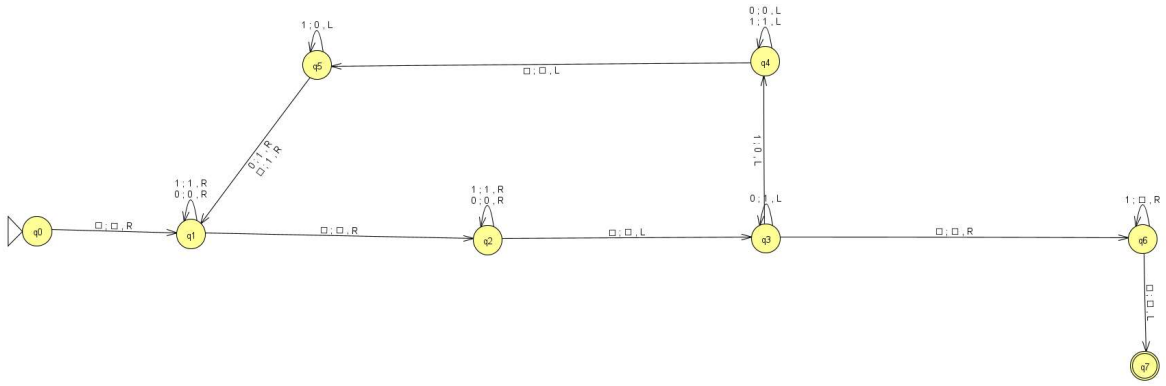
- *q0:* estado inicial
- *q1:*
- *q2:*

- *q3:*
- *q4:*
- *q5:*
- *q6:*
- *q7:* estado final

Inicialmente rodamos a fita até encontrar o separador, após isso acessamos o último valor do segundo número, e desse número subtraimos 1. Com isso ela troca 0's por 1's e 1's por 0's, adicionando 1 no primeiro número. Ao final, o segundo número tem seus valores trocados por □ e então a operação é aceita.

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
```

--

M = (Q,Σ,Γ,δ,q_0):

Q = {q1, q2, q3, q4, q5, q6, q7}

Σ = {0, 1}

Γ = {0, 1, □}

q_0 = {q1}

-

δ = (q0, □)->(q1, □, R)

-

δ = (q1, 1)->(q1, 1, R)

δ = (q1, 0)->(q1, 0, R)

δ = (q1, □)->(q2, □, R)

-

δ = (q2, 1)->(q2, 1, R)

δ = (q2, 0)->(q2, 0, R)

δ = (q2, □)->(q3, □, L)

-

δ = (q3, 0)->(q3, 1, L)

δ = (q3, 1)->(q4, 0, L)

δ = (q3, □)->(q6, □, R)

-

δ = (q4, 0)->(q4, 0, L)

δ = (q4, 1)->(q4, 1, L)

δ = (q4, □)->(q5, □, L)

-

δ = (q5, 1)->(q5, 0, L)

δ = (q5, 0)->(q1, 1, R)

δ = (q5, □)->(q1, 1, R)

-

δ = (q6, 1)->(q6, □, R)

δ = (q6, □)->(q7, □, L)

Exercício 4)

Faça a conversão de M em R(M)

δ = (q0, □)->(q1, □, R) = 101110110111011

δ = (q1, 1)->(q1, 1, R) = 11011011011011

δ = (q1, 0)->(q1, 0, R) = 110101101011

δ = (q1, □)->(q2, □, R) = 11011101110111011

δ = (q2, 1)->(q2, 1, R) = 1110110111011011

δ = (q2, 0)->(q2, 0, R) = 11101011101011

δ = (q2, □)->(q3, □, L) = 111011101111011101

δ = (q3, 0)->(q3, 1, L) = 1111010111101101

δ = (q3, 1)->(q4, 0, L) = 11110110111110101

δ = (q3, □)->(q6, □, R) = 1111011101111110111011

δ = (q4, 0)->(q4, 0, L) = 11111010111110101

δ = (q4, 1)->(q4, 1, L) = 1111101101111101101

δ = (q4, □)->(q5, □, L) = 11111011101111111011101

δ = (q5, 1)->(q5, 0, L) = 1111110110111110101

δ = (q5, 0)->(q1, 1, R) = 11111101011011011

δ = (q5, □)->(q1, 1, R) = 1111110111011011011

δ = (q6, 1)->(q6, □, R) = 111111101101111110111011

δ = (q6, □)->(q7, □, L) = 1111111011101111111011101

R(M) =
000101110110111011001101101101101100110101101011001101110111011101100111011011101 1

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.

In [1]:

```
def mtu(rm, w):
    N = 18 # Tamanho da fita
    maquina = open(rm).read()
    maquina = maquina.split('000')[1]
    w = open(w).read().split(',')
    w = '□'.join([w[0], w[1]])
```

```
    transicoes = {}
    estado = str(1) # 1 = q0 estado inicial
    fita = ['□'] * N # preenche a fita com '□' N vezes
    cabecote = N // 2  # cabecote é posicionada no meio da fita para começar a preen

    i = 0
    for elemento in w: # Coloca os elementos de w na fita
        fita[cabecote+i] = elemento #começa a partir da posição do cabeçote e vai pa
        i+=1

    for iterator in maquina.split('00'): # Separa as transições, quando juntas fica
        estadoPartida, leitura, estadoDestino, escrita, direcao  = iterator.split('0

        if(leitura == '11'):
            leitura = leitura.replace('11', '1')
        elif(leitura == '1'):
            leitura = leitura.replace('1', '0')
        elif(leitura == '111'):
            leitura = leitura.replace('111', '□')

        if(escrita == '11'):
            escrita = escrita.replace('11', '1')
        elif(escrita == '1'):
            escrita = escrita.replace('1', '0')
        elif(escrita == '111'):
            escrita = escrita.replace('111', '□')

        transicoes[estadoPartida,leitura] = (estadoDestino, escrita, direcao)

    cabecote -= 1 # Aponta para branco '□'
    leitura = fita[cabecote]

    while transicoes.get((estado, leitura)):
        acao = transicoes.get((estado, leitura)) #pega estado para onde vai e o que

        estadoDestino, escrita, direcao = acao #acao possui 3 posicoes
        fita[cabecote] = escrita #escreve na fita
        cabecote = cabecote + ({True: 1, False: -1} [direcao == '11']) #+1 se for di
        leitura = fita[cabecote]

        estado = estadoDestino

    print("Resultado: ", fita)
```

In [2]:

```
rm = 'rm.txt'
w = 'w.csv'

mtu(rm, w)
```

```
Resultado:  ['□', '□', '□', '□', '□', '□', '□', '□', '1', '1', '0', '0', '□', '□',
'□', '□', '□', '□']
```

Exercício 6)

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

Toda 'função que seria naturalmente considerada computável' pode ser computada por uma Máquina de Turing. Ou seja, tudo que pode ser efetivamente calculado é computável(produzida por uma MT).

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.

Sendo a entrada w uma linguagem recursiva, o teorema de Rice mostra que qualquer propriedade que seja satisfeita por algumas, mas não todas, linguagens recursivamente enumeráveis é indecidível.Turing provou que não existe nenhum algoritmo que sempre decida corretamente se, para um determinado programa e entrada arbitrários, o programa é interrompido quando executado com aquela entrada. A essência da prova de Turing é que qualquer algoritmo pode se contradizer e, portanto, não pode ser correto.Logo, se é um problema indecidivel não existe MT que possa resolver 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)

-

a) $\left[\frac{a}{aa}\right]\left[\frac{bb}{b}\right]\left[\frac{a}{bb}\right]$

$\downarrow$

$\left[\frac{a}{aa}\right]\left[\frac{a}{bb}\right]\left[\frac{bb}{b}\right]\left[\frac{bb}{b}\right]$

b) $\left[\frac{a}{ab}\right]\left[\frac{ba}{aba}\right]\left[\frac{b}{aba}\right]\left[\frac{bba}{b}\right]$

$\downarrow$

$\left[\frac{a}{ab}\right]\left[\frac{bba}{b}\right]\left[\frac{ba}{aba}\right]$

c) $\left[\frac{abb}{ab}\right]\left[\frac{aba}{ba}\right]\left[\frac{aab}{abab}\right]$

$\downarrow$

$$\left[\frac{abb}{ab}\right]\left[\frac{aba}{ba}\right]\left[\frac{aba}{ba}\right] \quad \text{INDECIDÍVEL}$$

d) $\left[\frac{ab}{aba}\right]\left[\frac{baa}{aa}\right]\left[\frac{aba}{baa}\right]$

$$\downarrow$$

$$\left[\frac{ab}{aba}\right]\left[\frac{aba}{baa}\right]\left[\frac{ab}{aba}\right] \quad \text{INDECIDÍVEL}$$

e) $\left[\frac{a}{aaa}\right]\left[\frac{aab}{b}\right]\left[\frac{abaaa}{ab}\right]$

$$\downarrow$$

$$\left[\frac{a}{aaa}\right]\left[\frac{aab}{b}\right]$$

f) $\left[\frac{ab}{bb}\right]\left[\frac{aa}{ba}\right]\left[\frac{ab}{abb}\right]\left[\frac{bb}{bab}\right]$

$$\left[\frac{ab}{abb}\right]\left[\frac{bb}{bab}\right]\left[\frac{ab}{bb}\right] \quad \text{INDECIDÍVEL}$$

CS Digitalizado com CamScanner

D)

a) Prove que a função é primitiva recursiva

$$f(x_1, \dots, x_n,) = \begin{matrix} u(x_1, \dots, x_n) \\ \mu z \end{matrix} [p(x_1, \dots, x_n, z)]$$

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

---

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

$pn(0)^{4+1} * pn(1)^{1+1} * pn(2)^{0+1} * pn(3)^{2+1} * pn(4)^1 + 1$

$2^{4+1} * 3^{1+1} * 5^{0+1} * 7^{2+1} * 11^{1+1} = 32 * 9 * 5 * 343 * 121 = 59764320$

E)

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

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

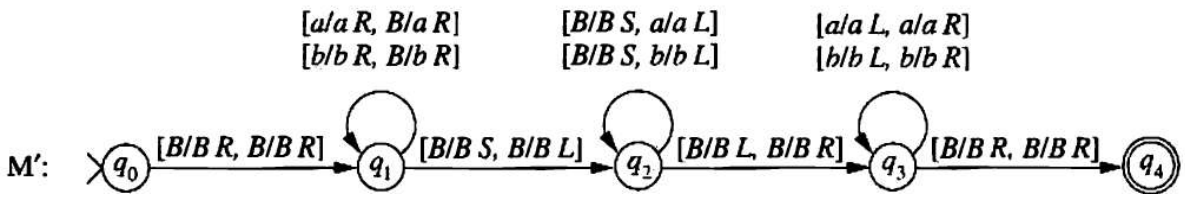
Complexidade de $f(x) = 3x^2$, dessa forma $O(f) = O(N^2)$

Complexidade de $g(x) = 5x^2$ dessa forma $O(g) = O(N^2)$

Logo, $f(x) \in O(f)$ e $g(x) \in O(f)$;

---

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



A complexidade de M' é tcM'(n) = 3(n + 1) + 1. Que representa a complexidade de uma função que aumenta linearmente e em proporção direta ao número de entradas.

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 [ ]: