Tarefa 1 — Introdução e Fundamentos Matemáticos e Computação e Representação

Teoria da Computação — PPComp

Eduardo Rigamonte

Novembro 2021

Problema 1 (*35pts*)

Parte I

Implemente os algoritmos de multiplicação tradicional (aquele que aprendemos na escola) e também o método de multiplicação de de Karatsuba (Algoritmo 0.4 do ITCS) como funções na linguagem Clojure. Suas funções receberão dois vetores com os dígitos dos número que devem ser multiplicados no esquema de dígito menos significante primeiro, i.e, o dígito de ordem 0 estará na posição de índice 0 do vetor. Suas funções devem, igualmente, retornar o resultado da multiplicação como um vetor contento os dígitos do resultado no esquema de dígito menos significante primeiro.

Por exemplo, $534826477703723139155890 9 \times 3047882377329360763487308$ seria calculado pelas chamadas de função abaixo.

```
(karatsuba-multiply
 [9 0 9 8 5 5 1 9 3 1 3 2 7 3 0 7 7 7 4 6 2 8 4 3 5]
 [8 0 3 7 8 4 3 6 7 0 6 3 9 2 3 7 7 3 2 8 8 7 4 0 3])
```

O resultado da execução das funções seria:

```
[2 7 9 6 2 8 5 5 3 5 4 6 2 8 1 6 4 5 6 4 9 0 2 9 4 4 8 0 0 5 1 0 4 0 2 1
   ↪ 3 2 2 3 6 9
1 8 8 0 0 3 6 1]
```

Parte II

Meça o tempo médio de execução das duas funções para números com quantidades crescentes de dígitos. Plote o gráfico com os tempos de execução das duas funções e determine a partir de que ponto – supondo que seja o caso – a multiplicação de Karatsuba se torna mais rápida que a multiplicação padrão.

Solução Parte I

```
(ns teocomp-trab1.core
  (:gen-class))

(defn custom-subvec
  "Retorna um subvetor forando ser um []"
  [X s e]
  (into [] (subvec X s e)))

(defn custom-reverse
  "Retorna o vetor ao contrario"
  [X]
  (into [] (reverse X)))

(defn num-to-vec
  "separa os digitos de um numero em um vetor com cada digito separado"
  [num]
  (cond
    (= num 0) (list '0)
    :else (loop [n num res []]
    (if (zero? n)
      res
      (recur (quot n 10) (cons (mod n 10) res))))))

(defn zero-left-pad
  "Funcao responsavel por completar com zeros a esquerda o vetor X ate o
     ↪ tamanho n"
  [X n]
  (cond
    (>= (count X) n) (into [] X)
    :else (reduce conj (vec (repeat (- n (count X)) 0)) X)))

(defn zero-right-pad
  "Funcao responsavel por completar com zeros a direita o vetor X ate o
     ↪ tamanho n"
  [X n]
  (cond
    (>= (count X) n) (into [] X)
    :else (reduce conj X (vec (repeat (- n (count X)) 0)))))

(defn add
  "Soma dois vetores de digitos"
  ([X Y] (add (into [] (zero-right-pad X (max (count X) (count Y))))
```

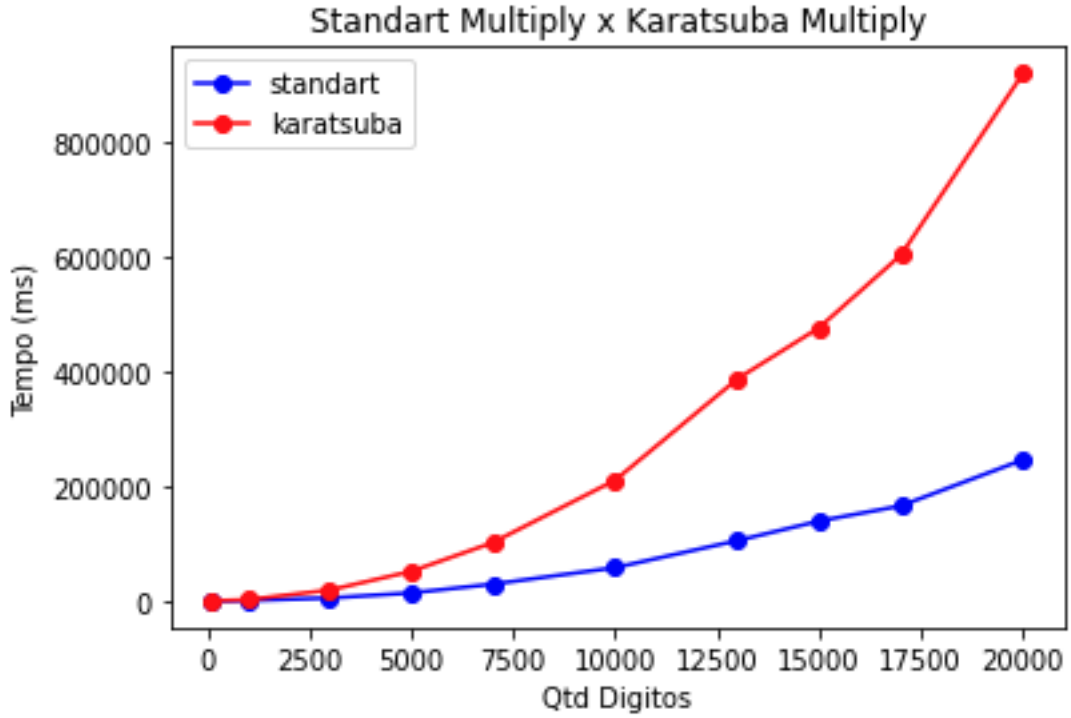
```
              (into [] (zero-right-pad Y (max (count X) (count Y))))
              0
              []))
  ([X Y carry acc]
   (cond
     (and (= carry 0) (and (= (count X) 0) (= (count Y) 0))) acc
     (and (> carry 0) (and (= (count X) 0) (= (count Y) 0))) (conj acc
        ↪ carry)
     :else (recur (rest X) (rest Y) (quot (+ (+ (first X) (first Y)) carry
        ↪ ) 10) (conj acc (mod (+ (+ (first X) (first Y)) carry) 10)))))
        ↪ )

(defn standard-multiply-one-digit
  "Multiplica um vetor por um digito pelo metodo standart"
  ([X y] (standard-multiply-one-digit X y 0 []))
  ([X y carry acc]
  (cond
    (and (= carry 0) (= (count X) 0)) acc
    (and (> carry 0) (= (count X) 0)) (conj acc carry)
    :else (recur (rest X)
                 y
                 (quot (+ (* (first X) y) carry) 10)
                 (conj acc (mod (+ (* (first X) y) carry) 10)))
    )))

(defn standard-multiply
  "Multiplica um vetor por outro vetor pelo metodo standart"
  [X Y]
  (reduce add (for [i (range (count Y))]
              (into [] (concat (zero-left-pad [] i) (
                 ↪ standard-multiply-one-digit X (nth Y i) 0 []))))))

(defn karatsuba-multiply
  "Multiplica um vetor por outro vetor pelo metodo karatsuba"
  ([X Y] (karatsuba-multiply (into [] (zero-right-pad X (max (count X) (
     ↪ count Y))))
                             (into [] (zero-right-pad Y (max (count X) (
                                ↪ count Y)))) (max (count X) (count Y))))
  ([X Y n]
   (cond
     (<= n 4) (standard-multiply X Y)
     :else (let [q-ceil (Math/ceil (/ n 2))
                 q-floor (Math/floor (/ n 2))
                 A (custom-subvec X q-ceil n)
```

```
                B (custom-subvec X 0 q-ceil)
                C (custom-subvec Y q-ceil n)
                D (custom-subvec Y 0 q-ceil)]
            (add (add (standard-multiply (karatsuba-multiply A C q-floor)
               ↪ (into [] (conj (zero-left-pad [] (* 2 q-ceil)) 1)))
                     (karatsuba-multiply B D q-ceil))
                 (standard-multiply (add (standard-multiply A D) (
                    ↪ standard-multiply B C)) (conj (zero-left-pad []
                    ↪ q-ceil) 1)))))))

(defn random-numbers
  ([n] (random-numbers n []))
  ([n acc]
  (cond
     (= n 0) acc
     :else (recur (- n 1) (conj acc (rand-int 10))))))

(defn test-multiply
  "Gera␣dois␣numeros␣aleatrios␣de␣n␣digitos␣e␣mede␣o␣tempo␣de␣cada␣um␣dos␣
     ↪ metodos␣de␣multiplicao"
  [n]
  (def A (random-numbers n))
  (def B (random-numbers n))
  (println "N:␣" n)
  (println "standard-multiply>>>␣" (time (standard-multiply A B)))
  (println "karatsuba-multiply>>␣" (time (karatsuba-multiply A B)))
  (println "----------------------------------------")
 )

(test-multiply 10)
(test-multiply 100)
(test-multiply 1000)
(test-multiply 3000)
(test-multiply 5000)
(test-multiply 7000)
(test-multiply 10000)
(test-multiply 13000)
(test-multiply 15000)
(test-multiply 17000)
(test-multiply 20000)
```

Solução Parte II

Para os algoritmos implementados, não foi encontrado um ponto em que o karatsuba

seja mais performático que o standart.



Problema 2 (*10pts*)

Use os quantificadores lógicos $\forall$ e $\exists$, bem como os operadores lógicos $\wedge$, $\vee$ e $\neg$, os operadores aritméticos $+$ e $\times$, e os operadores relacionais $=$, $>$ e $<$ para escrever o seguinte:

1. Uma expressão $\phi(n, m)$ tal que, para quaisquer números naturais n e m, $\phi(n, m)$ é verdadeiro se e somente se n e m são primos entre si.

2. Uma expressão $\phi(n)$ tal que para todo número natural n, $\phi(n)$ seja verdadeiro se e somente se n for uma potência de cinco.

Solução 2.1

Considerando que para dois números serem primos entre si o MDC deve ser igual a 1, encontra-se a seguinte expressão:

$$\phi(m, n) \equiv \forall_{a,b} \in \mathbb{N} : (MDC(a, b) = 1)$$

Solução 2.2

Considerando que para descobrir se um numero é potencia de outro a divisão de $\frac{\log a}{\log 5} \in \mathbb{N}$, encontra-se a seguinte expressão:

$$\phi(n) \equiv \exists_{a,b} \in \mathbb{N} : (\log 5 \times b - \log a = 0)$$

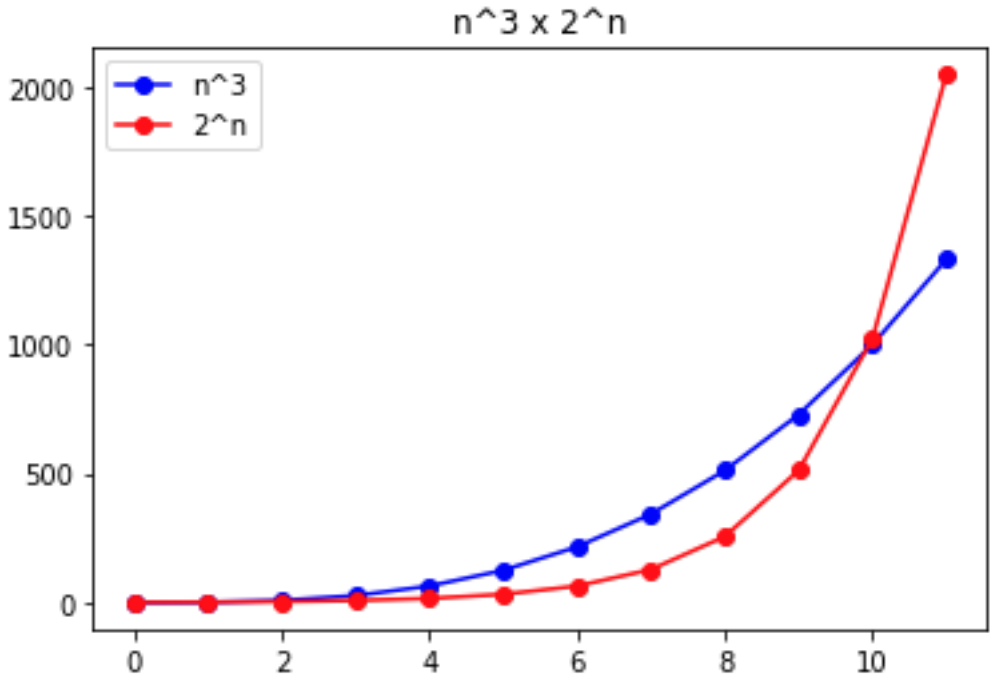
Problema 3 (*15pts*)

Seja $n \in \mathbb{N}$. Para cada um dos seguintes pares de conjuntos (S, T), prove ou refute a seguinte afirmação: existe uma função f, *injetora*, mapeando S em T.

1. Seja $n > 10$. $S = \{0,1\}^n$ e $T = [n] \times [n] \times [n]$.
2. Seja $n > 10$. S é o conjunto de todas as funções mapeando $\{0,1\}^n$ para $\{0, 1\}$. $T = \{0,1\}^{n^3}$
3. Seja $n > 100$. $S = \{k \in [n] : k \text{ é primo } \}$, $T = \{0,1\}^{\lceil \log n - 1 \rceil}$.

Solução 3.1

Considerando que $\{0,1\}^n$ mapeia um conjunto de strings em binário de tamanho 2^n e que o contra domínio, representado por $|n| \times |n| \times |n|$, possui tamanho n^3, para n $>$ 10, o domínio passa a ser maior que o contradomínio, conforme mostrado na figura abaixo. Por definição, uma função injetora deve mapear todos os elementos do domínio no contradomínio e todos os elementos do contradomínio devem ser mapeados uma única vez, com o domínio com mais elementos que o contradomínio ou a relação deixa de ser uma função ou não é uma função injetora.



Solução 3.2

Considerando que o conjunto de todas as funções de $\{0,1\}^n$ mapeia um domínio de $\{0,1\}^{2^n}$ e um contradomínio $\{0,1\}^{n^3}$, esta relação entra no mesmo caso da relação do problema anterior, onde o domínio passa a ter muitos mais elementos que o contradomínio em $n > 10$ impossibilitando a existência de uma função injetora.

Solução 3.3

Sendo S o conjunto de números primos existentes entre 0 e n-1, para n = 1000, temos 168 números primos nessa faixa e o valor de $\lceil \log 999 \rceil = 3$. Neste exemplo, o contradomínio seria capaz de mapear apenas 2^3 elementos, o que é inferior a quantidade de números primos entre 0 e 999, sendo assim a existência de uma função injetora nesta relação não seria possível.

Problema 4 (*20pts*)

A codificação ASCII pode ser usada para codificar cadeias de n letras em inglês como cadeias binárias de $7n$ bits. Nesse exercício é pedido que você encontre uma codificação mais compacta para cadeias de letras minúsculas em inglês

1. Prove que existe um esquema de representação (E, D) para strings sobre o alfabeto de 26 letras $\{a, b, c, ..., x, y, z\}$ como cadeias binárias tal que para todo $n > 0$ e cadeia $x \in \{a, b, c, ..., x, y, z\}^n$, de tamanho n, a representação $E(x)$ é uma cadeia binária de tamanho máximo $4.8n + 1000$. Em outras palavras, prove que para todo n, existe uma função injetora $E : \{a, b, c, ..., x, y, z\}^n \to \{0,1\}^{\lceil 4.8n+1000 \rceil}$

2. Prove que não existe esquema de representação sobre o alfabeto $\{a, b, c, ..., x, y, z\}$ como cadeias binárias tal que para toda cadeia $x \in \{a, b, c, ..., x, y, z\}^n$, de tamanho n, a representação $E(x)$ seja uma cadeia binária de tamanho $\lceil 4.6n + 1000 \rceil$. Em outras palavras, prove que para algum $n > 0$ tal que não existe função injetora $E : \{a, b, c, ..., x, y, z\}^n \to \{0,1\}^{\lceil 4.6n+1000 \rceil}$

Solução 4.1

Premissas:

- A função NtB(x, y) retorna o numero binário correspondente ao numeral passado, sendo x o número e y a quantidade de casas.
- A função len(x) retorna o tamanho da palavra.
- A função LrN(x) retorna um número entre 0 e 25 correspondente a posição da letra no alfabeto.
- As funções first(x) e rest(x) retornam o primeiro caracter da palavra e o restante da palavra, respectivamente

- O simbolo || foi utilizado com o significado de concatenação de strings

$$E(x) = \begin{cases} 1||NtB(LrN(x), 4) & len(x) = 1 \wedge LrN(x) < 16 \\ 01||NtB(LrN(x) - 16, 3) & len(x) = 1 \wedge LrN(x) < 24 \\ 00||NtB(LrN(x) - 24, 1) & len(x) = 1 \wedge LrN(x) < 26 \\ E(first(x))||E(rest(x)) & len(x) > 1 \end{cases}$$

A função E(x) será mais eficiente caso a função LrN(x) retornar o número correspondente a cada letra de modo a priorizar para cair no caso três e dois as letras que são mais recorrentes.

Problema 5 (*20pts*)

1. Foi demonstrado que os números naturais podem ser representados como cadeias binárias. Prove que o inverso também se aplica, i.e., que existe uma função injetora $StN : \{0,1\}^* \to \mathbb{N}$

2. O Teorema de Canto provou que não existe uma função injetora $RtN : \mathbb{R} \to \mathbb{N}$. Demonstre que o Teorema de Canto implica no teorema abaixo:

   Teorema: Não existe uma função injetora $RtS : \mathbb{R} \to \{0,1\}^*$.

Solução 5.1

Considerando f(x) sendo o digito referente a posição x do numero em binário, encontra-se a seguinte função responsável por mapear todos os números binários em números decimais.

$$StN(f) = \sum_{i=0}^{n-1} 2^i \times f((n-1) - i)$$