Universidade Federal do Rio Grande do Norte
Centro de Ensino Superior do Seridó
Departamento de Computação e Tecnologia
Curso de Bacharelado em Sistemas de Informação

Atividade Avaliativa Prática

David do Nascimento Santos

Caicó, RN

14 de junho de 2022

David do Nascimento Santos

Atividade Avaliativa

Trabalho apresentado a disciplina de Estrutura de Dados do Departamento de Computação e Tecnologia da Universidade Federal do Rio Grande do Norte como requisito para a obtenção da nota da atividade avaliativa .

Professor

Professor Doutor João Paulo de Souza Medeiros

BSI – Bacharelado em Sistemas de Informação
DCT – Departamento de Computação e Tecnologia
CERES – Centro de Ensino Superior do Seridó
UFRN – Universidade Federal do Rio Grande do Norte

Caicó-RN

14 de junho de 2022

Atividade Avaliativa

Aluno: David do Nascimento Santos
Professor: Professor Doutor João Paulo de Souza Medeiros

Resumo

Este trabalho objetiva apresentar um relatório sobre análise de algoritmos associados ordenação, tendo como exemplo os algoritmos: insertion-sort, merge-sort e quicksort.Na análise dos algoritmos foram desenvolvidas três atividades, a primeira delas foi caracterizada pela criação de gráficos do tempo de execução com base no tamanho da entrada, a segunda atividade foi composta por uma análise analítica dos três algoritmos, e a ultima atividade se deu por uma comparacão sobre o desempenho de cada algoritmo em relação ao custo em tempo e memória.

Palavras-chave: Algoritmos de ordenação, Análise, Tempo de execução.

Practical Activity

Student: David do nascimento santos
Teacher: Professor Doctor João Paulo de Souza Medeiros

Abstract

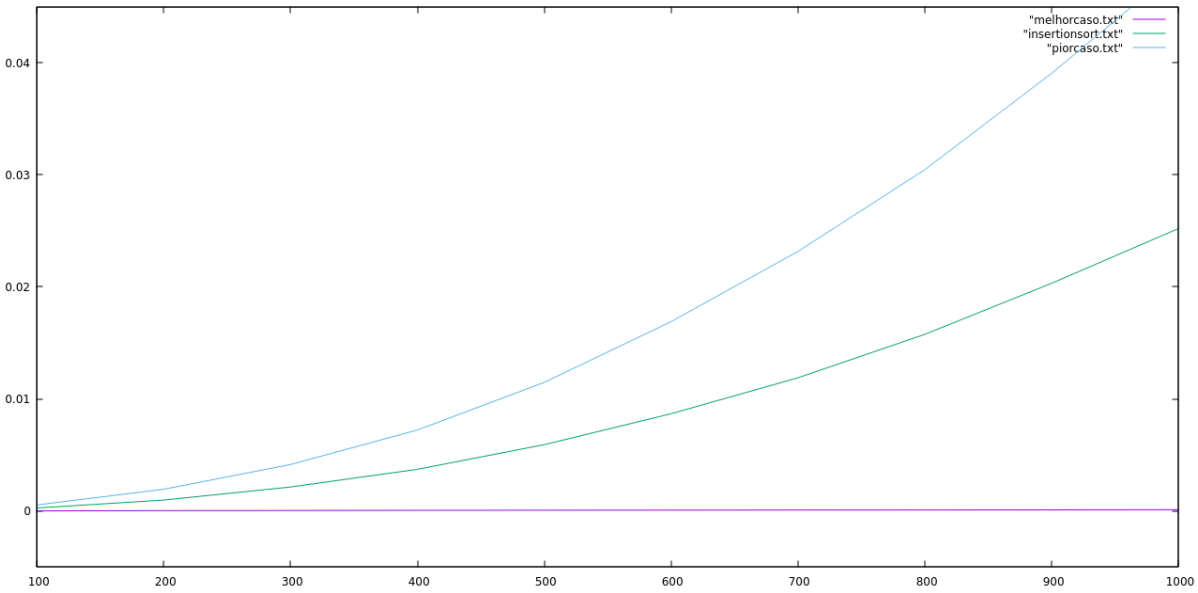
This work aims to present a report on algorithm analysis associated sorting, taking as an example the algorithms: insertion-sort, merge-sort and quicksort. In the analysis of the algorithms, three activities were developed, the The first one was characterized by the creation of runtime graphs based on in the size of the input, the second activity consisted of an analytical analysis of the three algorithms, and the last activity was carried out by comparing the performance of each algorithm relative to the cost in time and memory.

Keywords: Sorting algorithms, Analysis, Runtime.

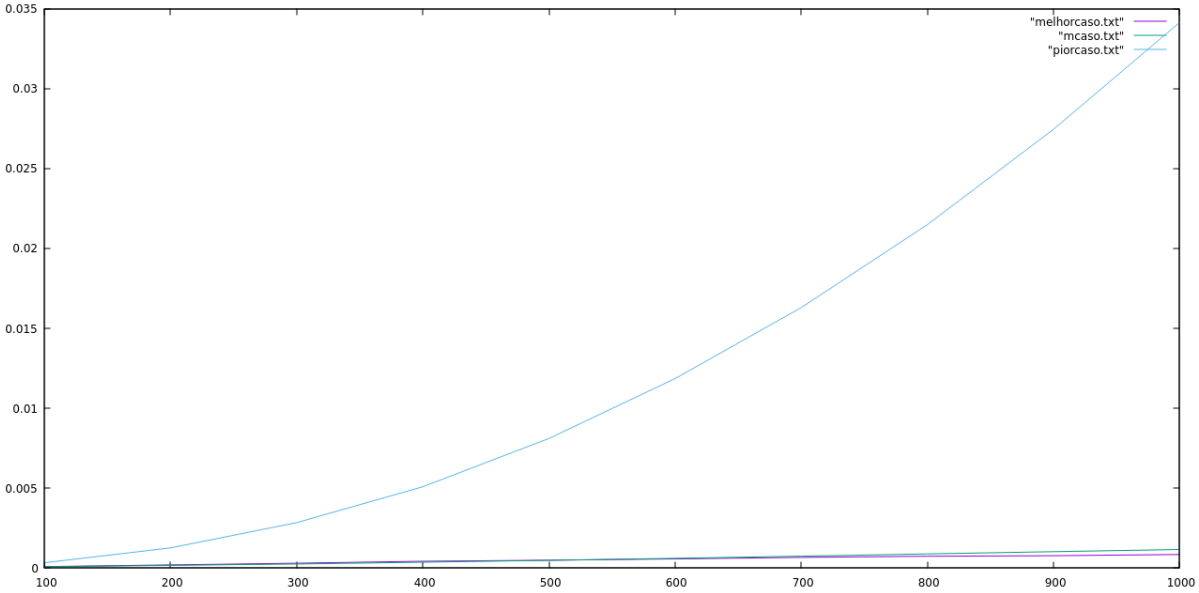
Sumário

1 Graficos

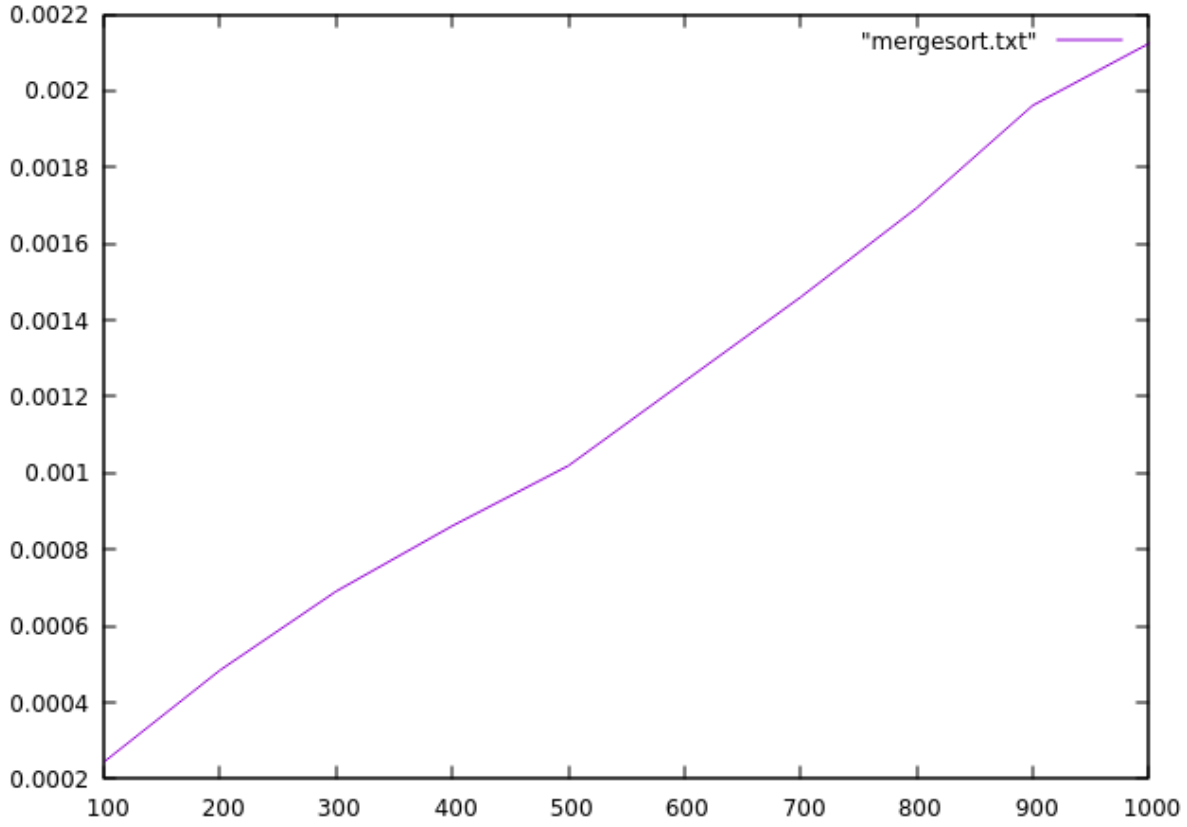
1.1 Insertion sort



1.2 Quick sort



1.3 Merge sort



2 Comparação em relação ao tempo

Pela análise dos gráficos no capítulo 1 , é possível observar que o algoritmo Insertion-sort é menos eficiente do que os algoritmos Merge-sort e Quick-sort.

se tomarmos o tempo "medio"como base, o Insertion-sort levou mais tempo para realizar a ordenação finalizando com um tempo de aproximadamente 0.025 segundos no seu ultimo teste.

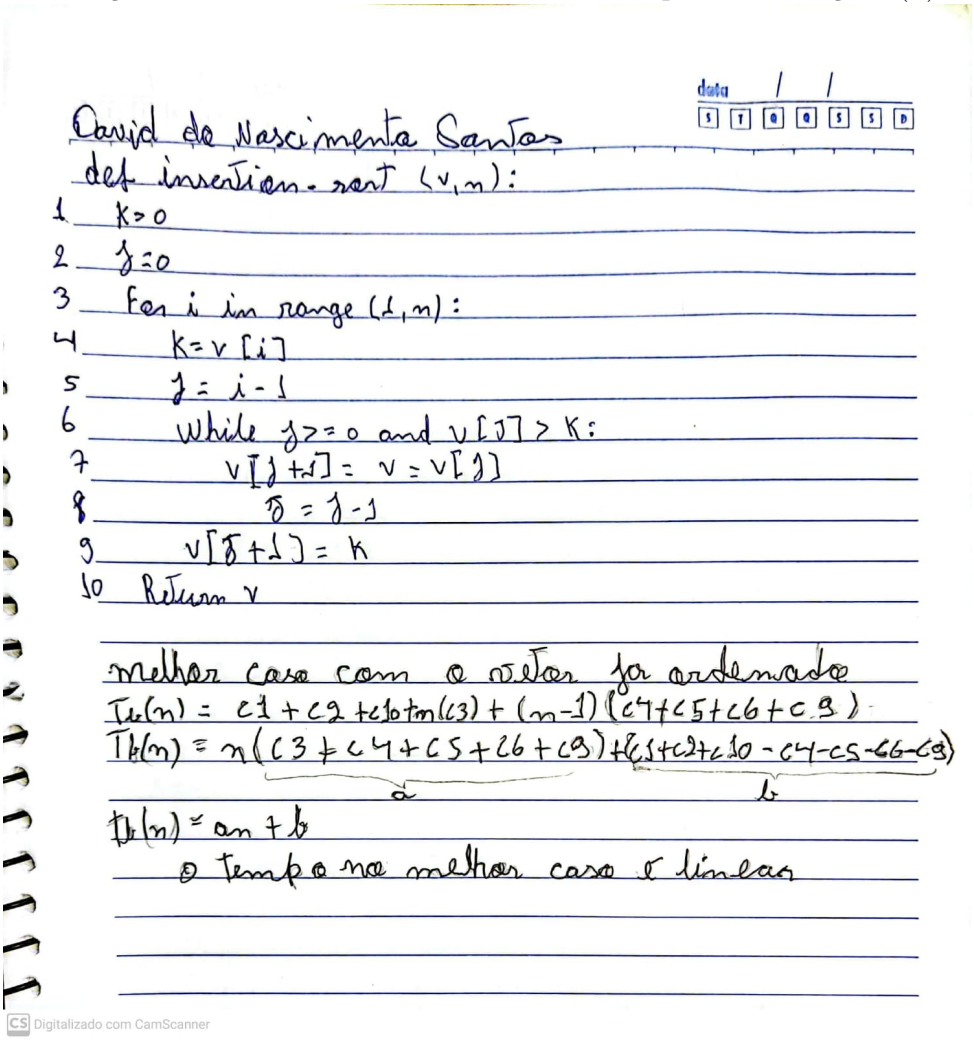
Tamanho do vetor	Insertion sort	Merge sort	Quick sort
200	0.000984	0.000484	0.000173
500	0.005927	0.001018	0.000492
1000	0.025207	0.002123	0.001171

3 Analise analítica do tempo de execução

3.1 Insertion sort

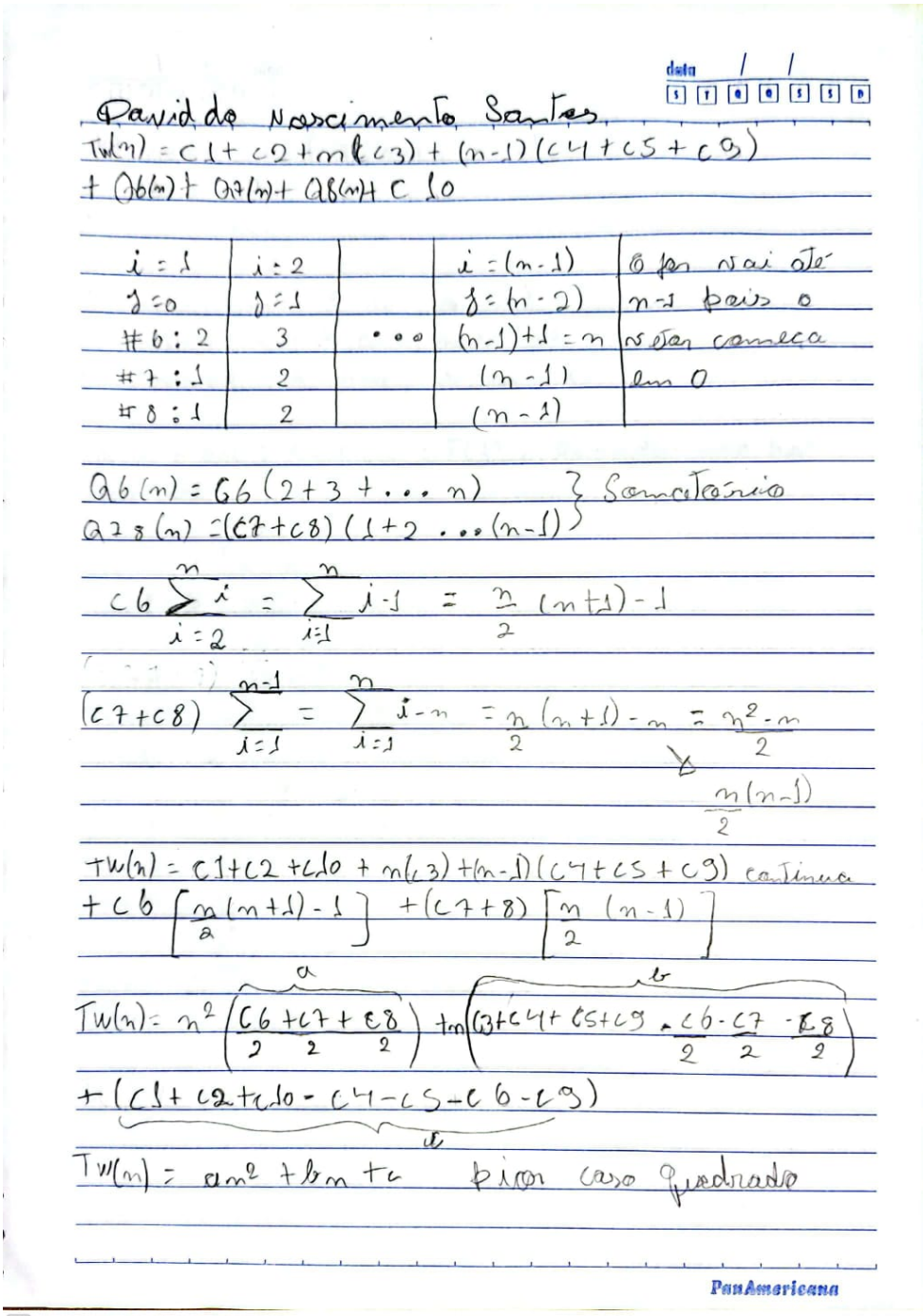
3.1.1 Melhor caso

A análise do tempo de execução do algoritmo Insertion-sort permitiu observar que o tempo de execução no melhor caso é linear, sendo dado por uma função f(n) = an + b



3.1.2 Pior caso

A análise do tempo de execução do algoritmo Insertion-sort permitiu observar que o tempo de execução no pior caso é quadrático, sendo dado por uma função f(n) = an² + bn + c.



David do Nascimento Santos

$T_w(n) = c1 + c2 + n(c3) + (n-1)(c4 + c5 + c9) + Q6(n) + Q7(n) + Q8(n) + c10$

i = 1	i = 2		i = (n-1)	O for vai até
j = 0	j = 1		j = (n-2)	n-1 pois o
#6 : 2	3	...	(n-1)+1 = n	vetor começa
#7 : 1	2		(n-1)	em 0
#8 : 1	2		(n-1)	

$Q6(n) = C6(2 + 3 + \dots n)$ } Somatório

$Q7\,8(n) = (C7 + C8)(1 + 2 \dots (n-1))$

$$c6 \sum_{i=2}^{n} i = \sum_{i=1}^{n} i - 1 = \frac{n}{2}(n+1) - 1$$

$$(c7 + c8) \sum_{i=1}^{n-1} = \sum_{i=1}^{n} i - n = \frac{n}{2}(n+1) - n = \frac{n^2 - n}{2}$$

$$\frac{n(n-1)}{2}$$

$T_w(n) = c1 + c2 + c10 + n(c3) + (n-1)(c4 + c5 + c9)$ continua

$+ c6\left[\frac{n}{2}(n+1) - 1\right] + (c7 + 8)\left[\frac{n}{2}(n-1)\right]$

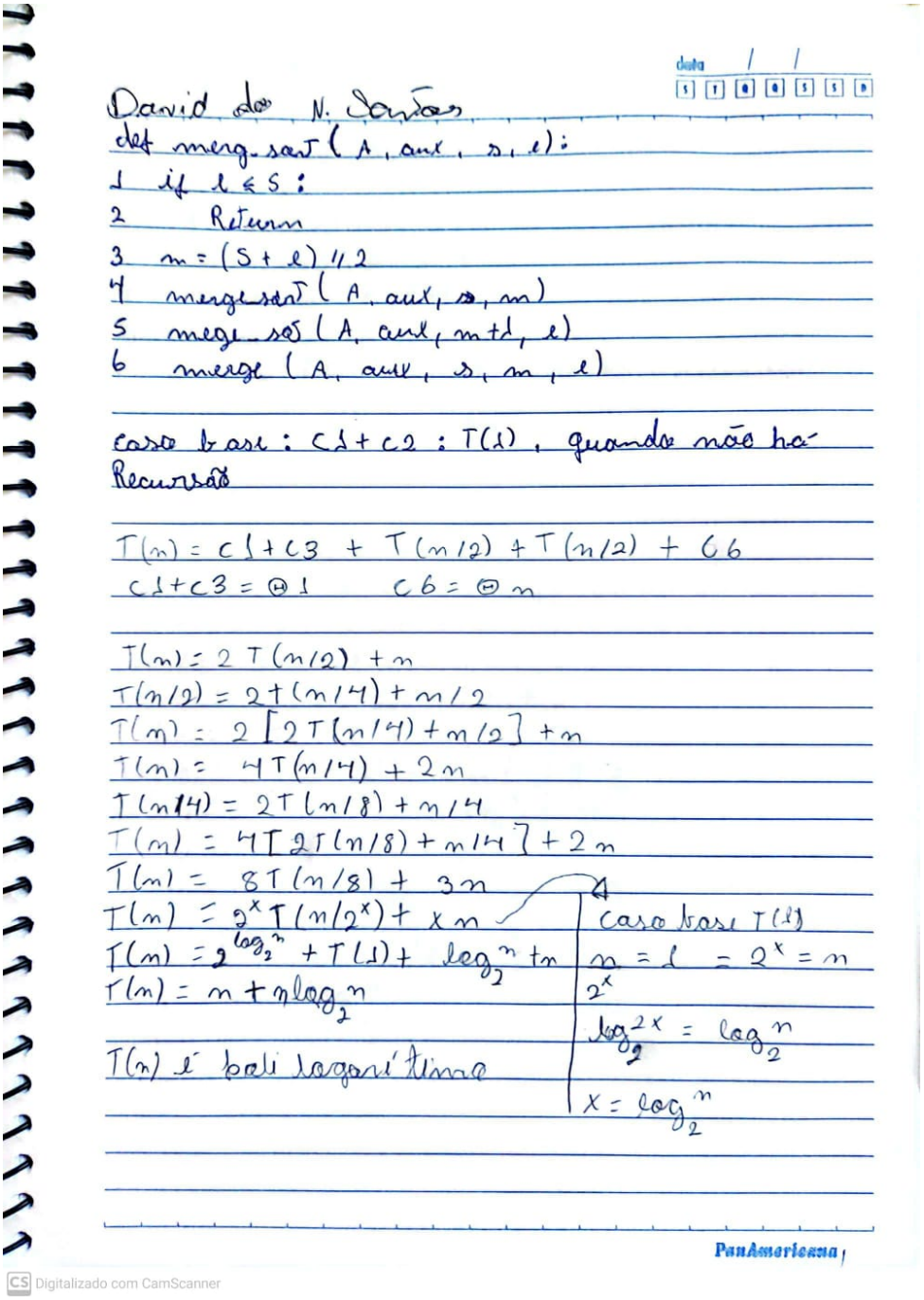
$$T_w(n) = n^2\underbrace{\left(\frac{c6}{2} + \frac{c7}{2} + \frac{c8}{2}\right)}_{a} + n\underbrace{\left(c3 + c4 + c5 + c9 - \frac{c6}{2} - \frac{c7}{2} - \frac{c8}{2}\right)}_{b} + \underbrace{(c1 + c2 + c10 - c4 - c5 - c6 - c9)}_{c}$$

$T_w(n) = an^2 + bn + c$ pior caso quadrado

PanAmericana

3.2 Merge sort

A análise do tempo de execução do algoritmo Merge-sort permitiu observar que em ambos os casos, o seu tempo de execução será sempre de complexidade polilogatima.



David do N. Santos

```
def merg_sort ( A, aux, s, e):
1 if e ≤ s :
2     Return
3 m = (s + e) // 2
4 merge_sort ( A, aux, s, m)
5 merge_sort ( A, aux, m+1, e)
6 merge ( A, aux, s, m, e)
```

caso base : c1 + c2 : T(1), quando não há Recursão

$T(n) = c1 + c3 + T(n/2) + T(n/2) + c6$

$c1 + c3 = \Theta 1 \qquad c6 = \Theta n$

$T(n) = 2T(n/2) + n$

$T(n/2) = 2T(n/4) + n/2$

$T(n) = 2[2T(n/4) + n/2] + n$

$T(n) = 4T(n/4) + 2n$

$T(n/4) = 2T(n/8) + n/4$

$T(n) = 4T[2T(n/8) + n/4] + 2n$

$T(n) = 8T(n/8) + 3n$

$T(n) = 2^x T(n/2^x) + xn$

$T(n) = 2^{\log_2 n} + T(1) + \log_2 n \cdot n$

$T(n) = n + n\log_2 n$

T(n) é bali logarítima

caso base T(1)

$n = 1 = 2^x = n$

2^x

$\log_2 2^x = \log_2 n$

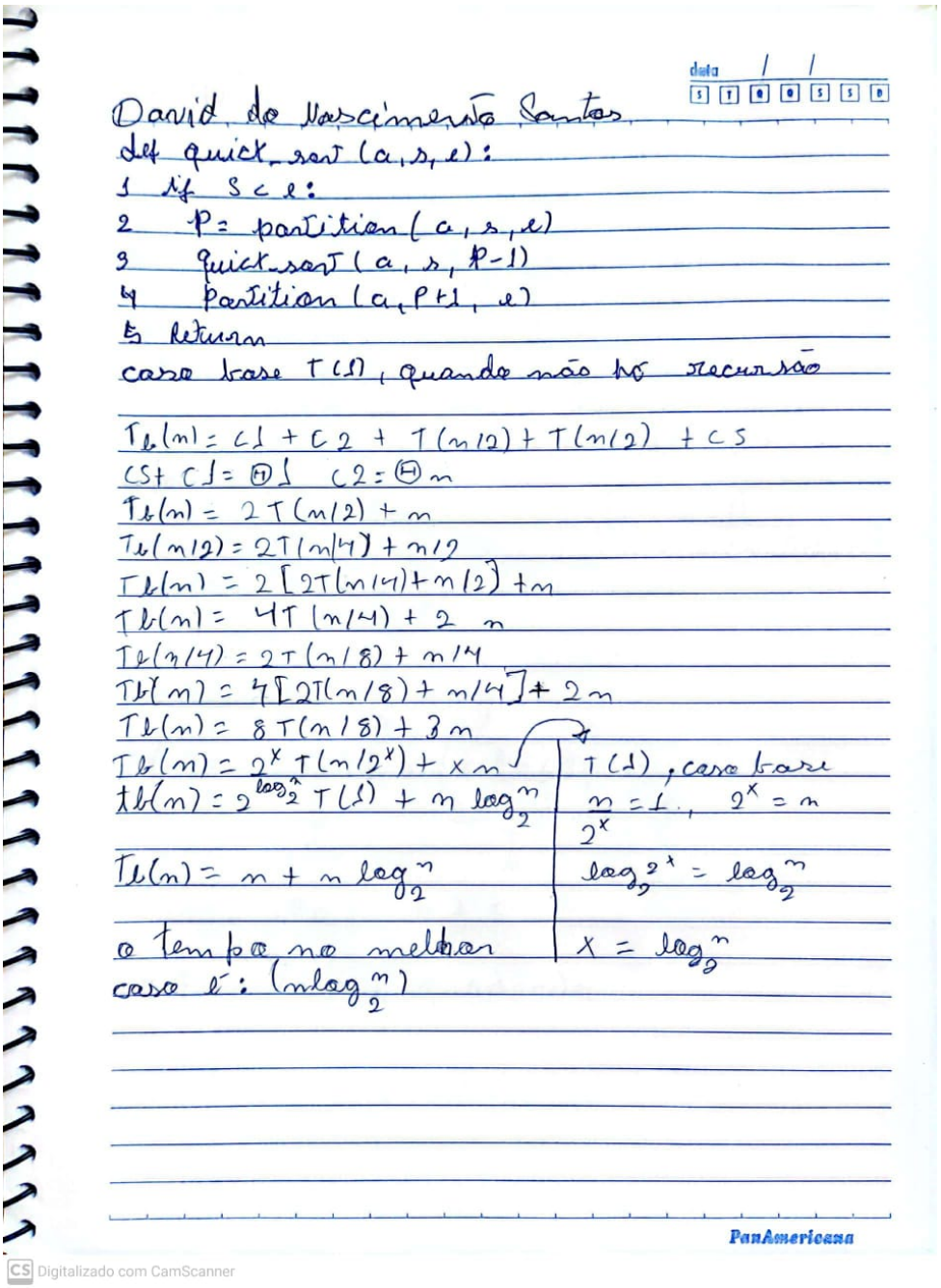
$x = \log_2 n$

PanAmericana

3.3 Quick sort

3.3.1 Melhor caso

A análise do tempo de execução do algoritmo Quick-sort permitiu observar que o tempo de execução no melhor caso é de complexidade polilogaritima .



David do Nascimento Santos

```
def quick_sort(a, s, e):
1 if s < e:
2   p = partition(a, s, e)
3   quick_sort(a, s, p-1)
4   partition(a, p+1, e)
5 return
```

caso base T(1), quando não há recursão

$T_b(n) = c1 + c2 + T(n/2) + T(n/2) + c5$

$c5 + c1 = \Theta 1 \quad c2 = \Theta n$

$T_b(n) = 2T(n/2) + n$

$T_b(n/2) = 2T(n/4) + n/2$

$T_b(n) = 2[2T(n/4) + n/2] + n$

$T_b(n) = 4T(n/4) + 2n$

$T_b(n/4) = 2T(n/8) + n/4$

$T_b(n) = 4[2T(n/8) + n/4] + 2n$

$T_b(n) = 8T(n/8) + 3n$

$T_b(n) = 2^x T(n/2^x) + xn$

$T_b(n) = 2^{\log_2^n} T(1) + n \log_2^n$

$T_b(n) = n + n \log_2^n$

o tempo no melhor caso é: $(n \log_2^n)$

T(1), caso base

$\frac{n}{2^x} = 1, \quad 2^x = n$

$\log_2 2^x = \log_2^n$

$x = \log_2^n$

PanAmericana

3.3.2 Pior caso

A análise do tempo de execução do algoritmo Quick-sort permitiu observar que o tempo de execução no pior caso é quadrático, sendo dado por uma função $f(n) = an^2 + bn + c$.