ANÁLISE DO TEMPO DE EXECUÇÃO DE ALGORITMOS DE ORDENAÇÃO EM DIFERENTES LINGUAGENS

Leomar Camargo de Souza
Métodos Quantitativos em Computação
Pós-Graduação em Informática
Universidade de Brasília

INTRODUÇÃO

INTRODUÇÃO

- O uso de linguagens de programação
  - Desenvolvimento de *software*;
    - Leitura e escrita de arquivos;
    - Problemas de ordenação;
    - Acesso a banco de dados;
    - Soluções de bioinformática;
  - Sistemas embarcados;
  - Pesquisa científica;
  - Geralmente o fator **tempo** de execução é considerado;

INTRODUÇÃO

- *The Computer Language Benchmarks Game*
  - Comparação de programas em diferentes linguagens de programação;
    - Árvores binárias;
      - Alocação e desalocação de diversas árvores binárias;
    - Complemento reverso;
      - Conversão de sequência de DNA em seu complemento reverso;
    - Sequência FASTA
      - Representação de sequências de nucleotídeos em bioinformática;
  - Dados de performance disponibilizados para os pesquisadores;

INTRODUÇÃO

- Problema de pesquisa
  - Qual é a melhor linguagem de programação para ordenação de dados em grande escala?

INTRODUÇÃO

- Problema de pesquisa
  - Qual é a **melhor linguagem de programação** para ordenação de dados em grande escala?

MELHOR = MENOR TEMPO GASTO

METODOLOGIA

METODOLOGIA

- Linguagens de programação escolhidas:
  - C#
    - Versão 7;
    - Visual Studio 2017 Community;
  - Java
    - Versão 8;
    - Eclipse Java Neon;
  - Python
    - Versão 3.6;
    - Pycharm;

METODOLOGIA

- Linguagens de programação escolhidas:
  - **C#**
    - **Versão 7;**
    - **Visual Studio 2017 Community;**
  - Java
    - Versão 8;
    - Eclipse Java Neon;
  - Python
    - Versão 3.6;
    - Pycharm;

Linguagem escolhida para a comparação

METODOLOGIA

- Algoritmos de ordenação escolhidos:

ALGORITMO	COMPLEXIDADE
Heap Sort	Θ (n log n)
Merge Sort	Θ (n log n)
Quick Sort	O (n^2)
Radix Sort	Θ (nk)

METODOLOGIA

- Algoritmos de ordenação escolhidos:

ALGORITMO	COMPLEXIDADE
Heap Sort	$\Theta (n \log n)$
Merge Sort	$\Theta (n \log n)$
Quick Sort	$O (n^2)$
Radix Sort	$\Theta (nk)$

METODOLOGIA

- Algoritmos de ordenação escolhidos:

ALGORITMO	COMPLEXIDADE
Heap Sort	$\Theta (n \log n)$
Merge Sort	$\Theta (n \log n)$
Quick Sort	$O (n^2)$
Radix Sort	$\Theta (nk)$

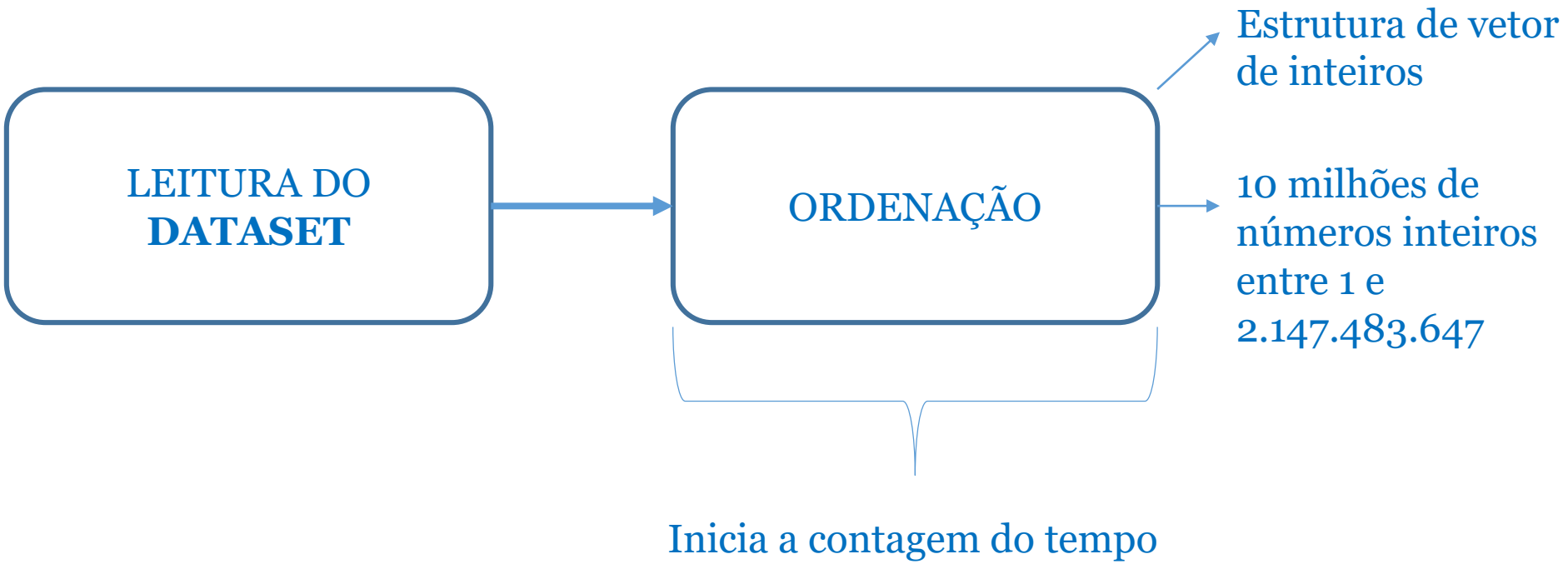
METODOLOGIA

- Algoritmos de ordenação escolhidos:

ALGORITMO	COMPLEXIDADE
Heap Sort	$\Theta (n \log n)$
Merge Sort	$\Theta (n \log n)$
Quick Sort	$O (n^2)$
Radix Sort	$\Theta (nk)$

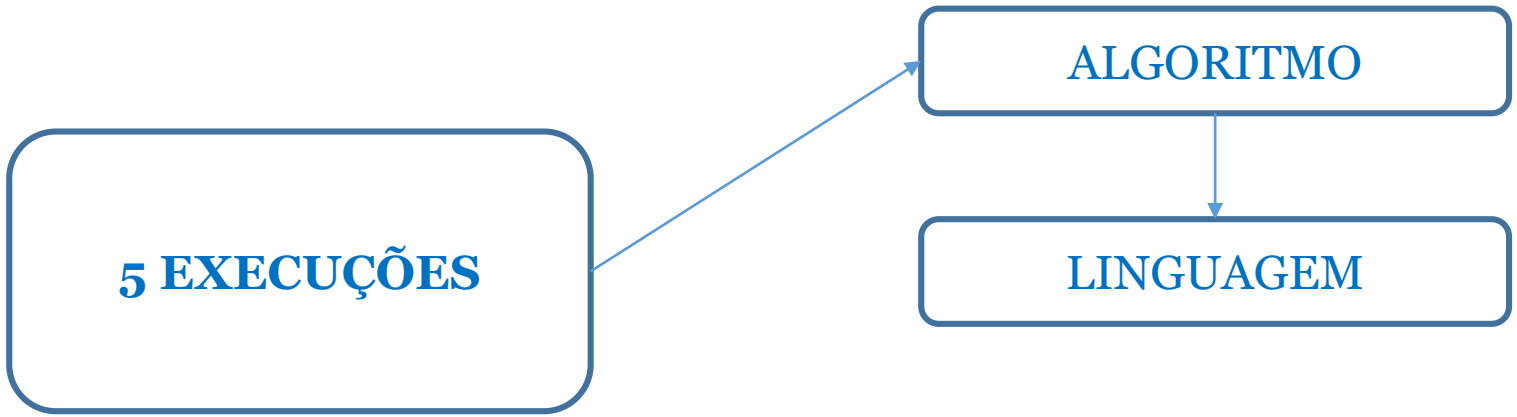
METODOLOGIA

1. Definição da pesquisa;
2. Implementação dos algoritmos;



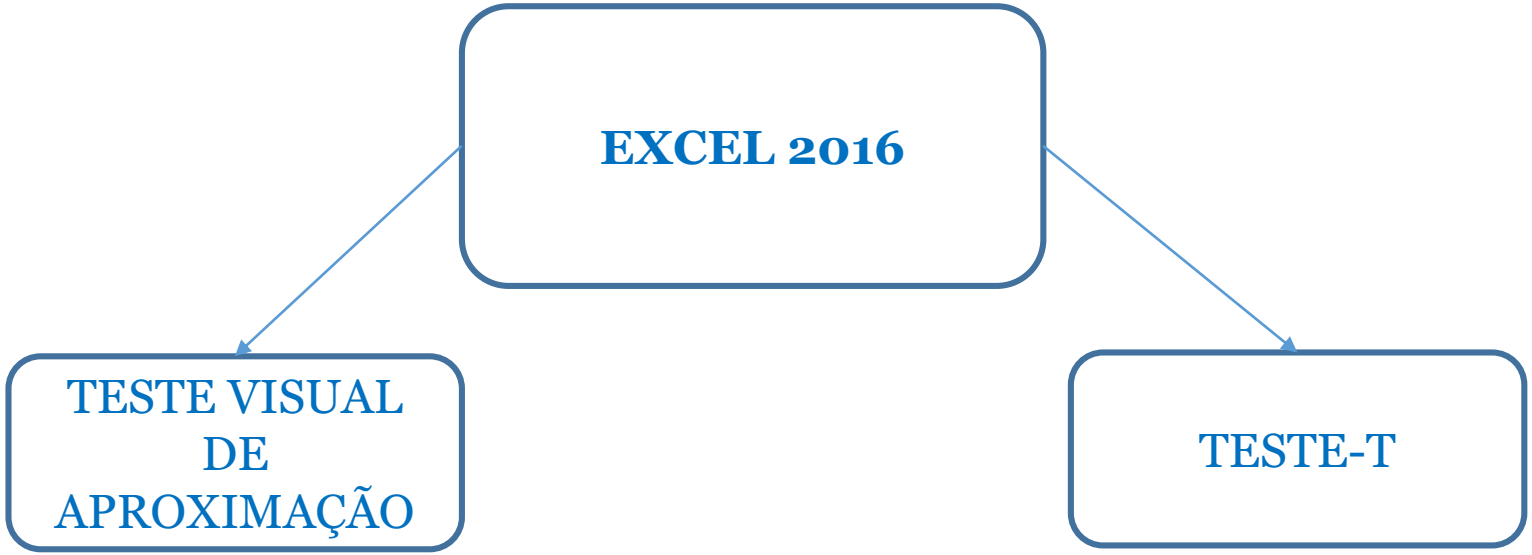
METODOLOGIA

1. Definição da pesquisa;
2. Implementação dos algoritmos;
3. Coleta dos dados;



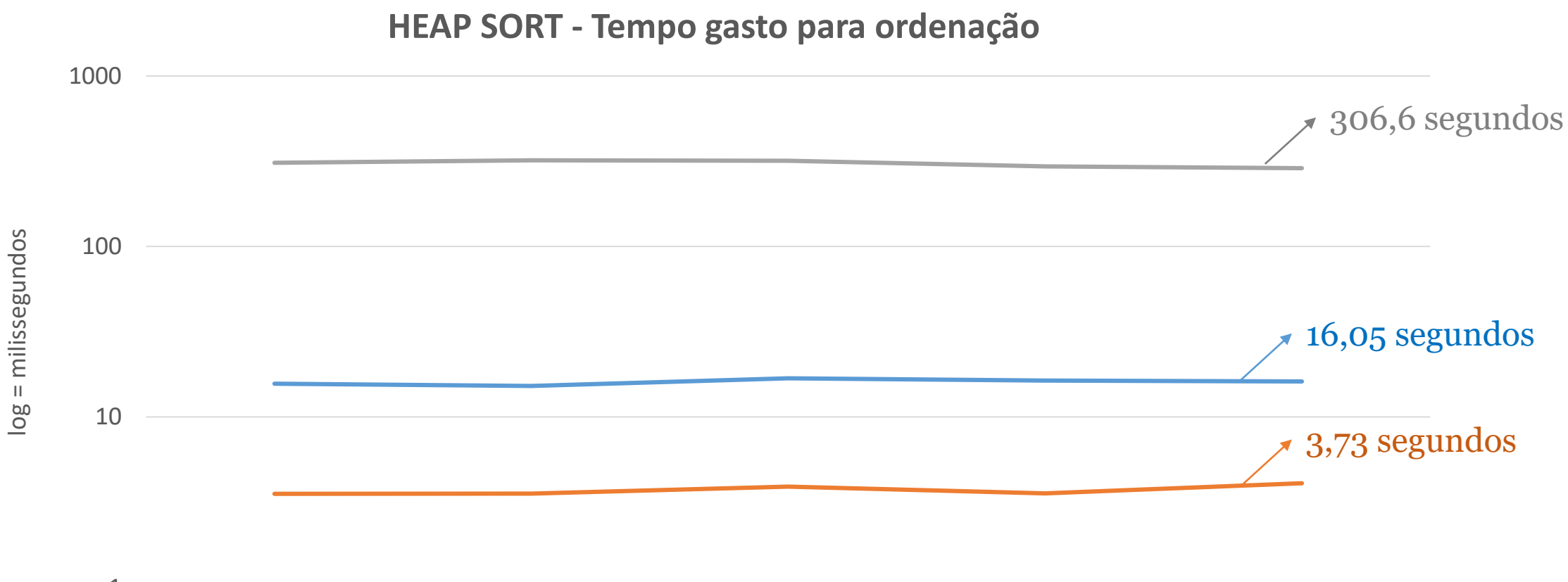
METODOLOGIA

1. Definição da pesquisa;
2. Implementação dos algoritmos;
3. Coleta dos dados;
4. Análise dos dados;



RESULTADOS

HEAP SORT

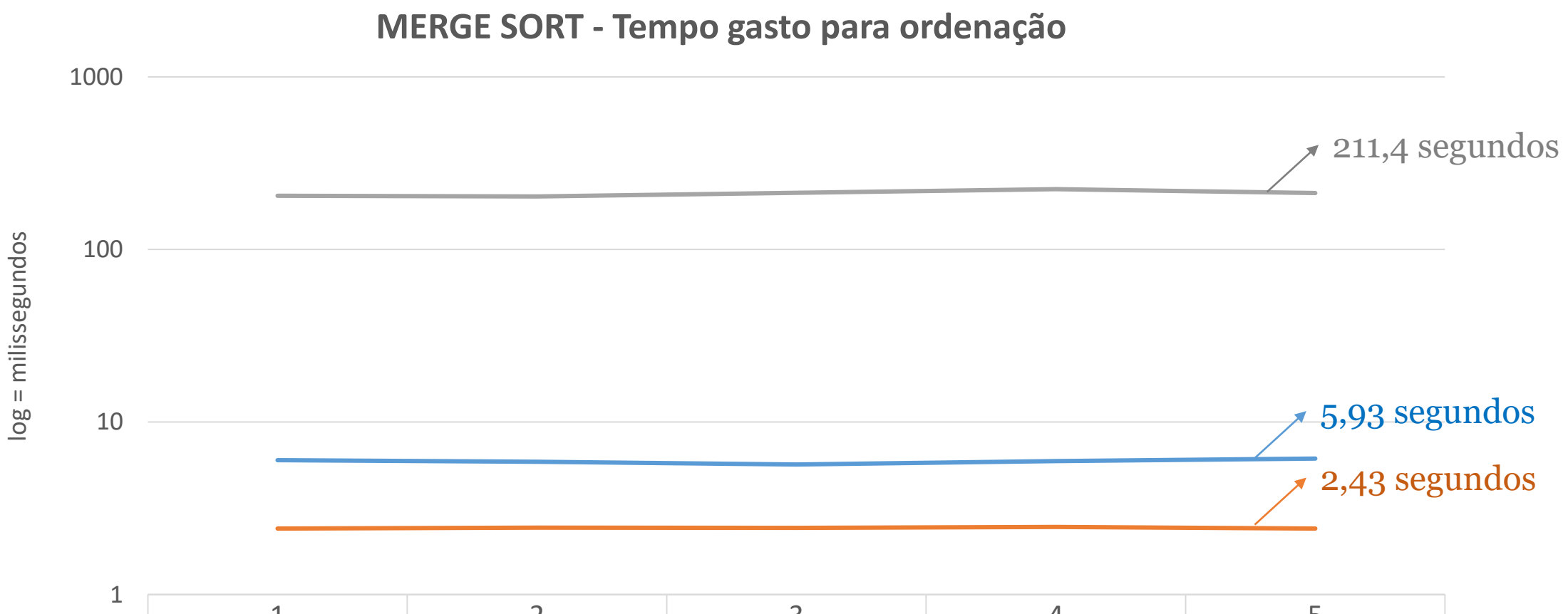


	1	2	3	4	5
C#	15,6870	15,2120	16,8420	16,3390	16,1510
Java	3,5400	3,5490	3,9000	3,5650	4,0870
Python	310,3590	320,4980	318,8120	295,0900	288,4740

HEAP SORT

- **H_0**: O tempo de ordenação utilizando C# é **mais significativo** que o tempo de ordenação utilizando Java?
  - Teste-T:
    - Índice de confiança a 90%: [(11,81),(15,11)];
    - A implementação em C# é de fato mais significativa;
- **H_0**: O tempo de ordenação utilizando C# é **mais significativo** que o tempo de ordenação utilizando Python?
  - Teste-T:
    - Índice de confiança a 90%: [(-304,84),(-2.258,51)];
    - A implementação em C# é de fato mais significativa;

MERGE SORT



	1	2	3	4	5
C#	6,0180	5,8790	5,6800	5,9440	6,1440
Java	2,4150	2,4430	2,4390	2,4620	2,4140
Python	205,1320	202,8800	213,1700	223,8930	212,3590

MERGE SORT

- **H_0**: O tempo de ordenação utilizando C# é **mais significativo** que o tempo de ordenação utilizando Java?
  - Teste-T:
    - Índice de confiança a 90%: [(3,32),(3,77)];
    - A implementação em C# é de fato mais significativa;
- **H_0**: O tempo de ordenação utilizando C# é **mais significativo** que o tempo de ordenação utilizando Python?
  - Teste-T:
    - Índice de confiança a 90%: [(-213,42),(-993,26)];
    - A implementação em C# é de fato mais significativa;

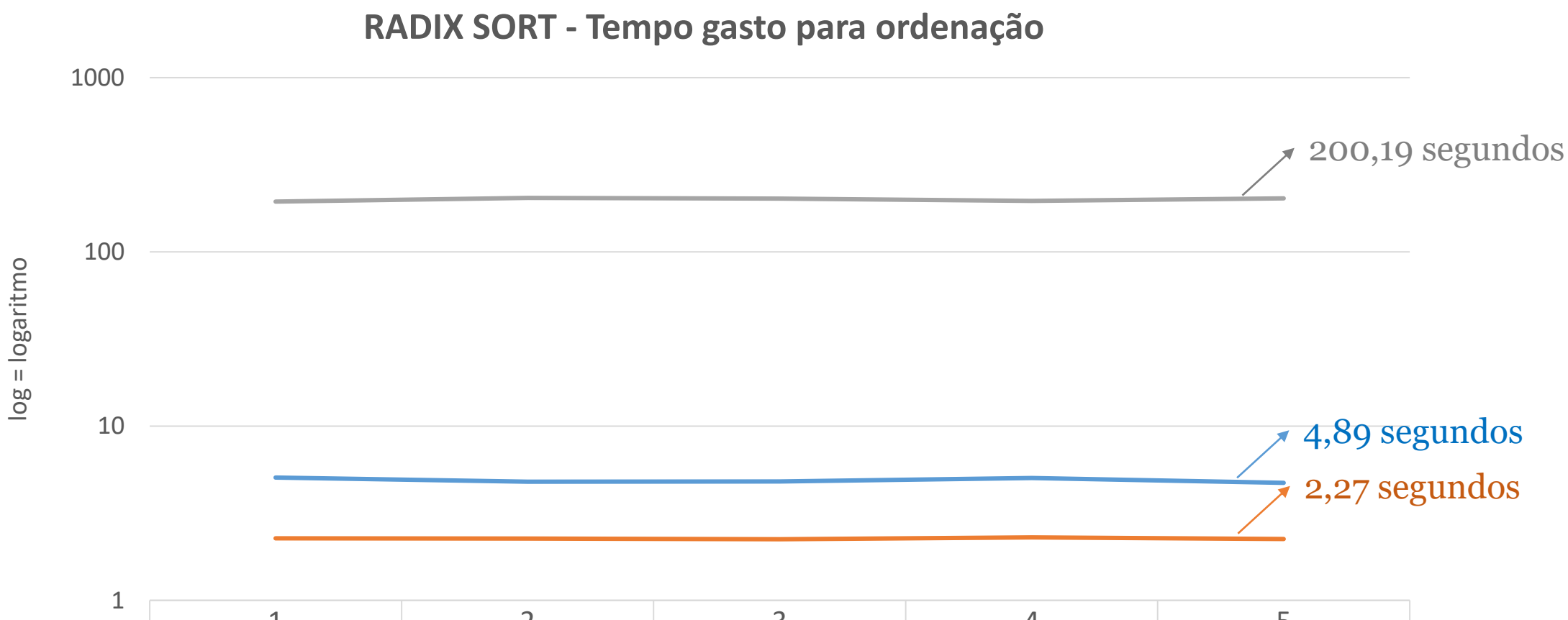
QUICK SORT

	1	2	3	4	5
C#	3,9330	3,5390	3,6080	3,5360	3,5020
Java	1,3800	1,4970	1,3720	1,3660	1,3720
Python	118,2450	110,2770	111,4870	113,6280	114,4030

QUICK SORT

- **H_0**: O tempo de ordenação utilizando C# é **mais significativo** que o tempo de ordenação utilizando Java?
  - Teste-T:
    - Índice de confiança a 90%: [(2,03),(2,40)];
    - A implementação em C# é de fato mais significativa;
- **H_0**: O tempo de ordenação utilizando C# é **mais significativo** que o tempo de ordenação utilizando Python?
  - Teste-T:
    - Índice de confiança a 90%: [(-112,78),(-258,22)];
    - A implementação em C# é de fato mais significativa;

RADIX SORT

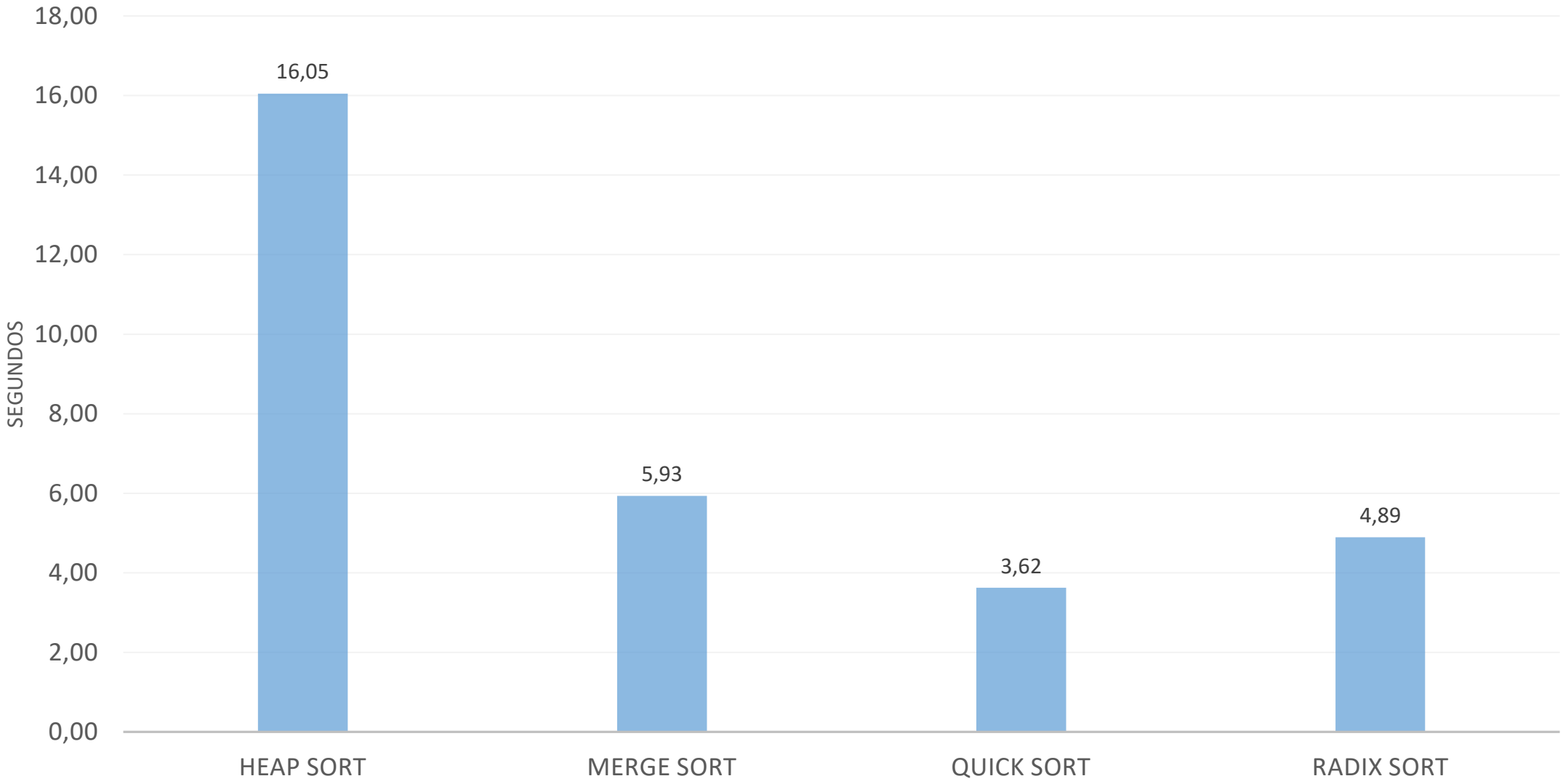


	1	2	3	4	5
C#	5,0770	4,7980	4,8110	5,0450	4,7350
Java	2,2710	2,2640	2,2460	2,3010	2,2530
Python	194,5640	204,5060	202,4150	196,6370	202,8250

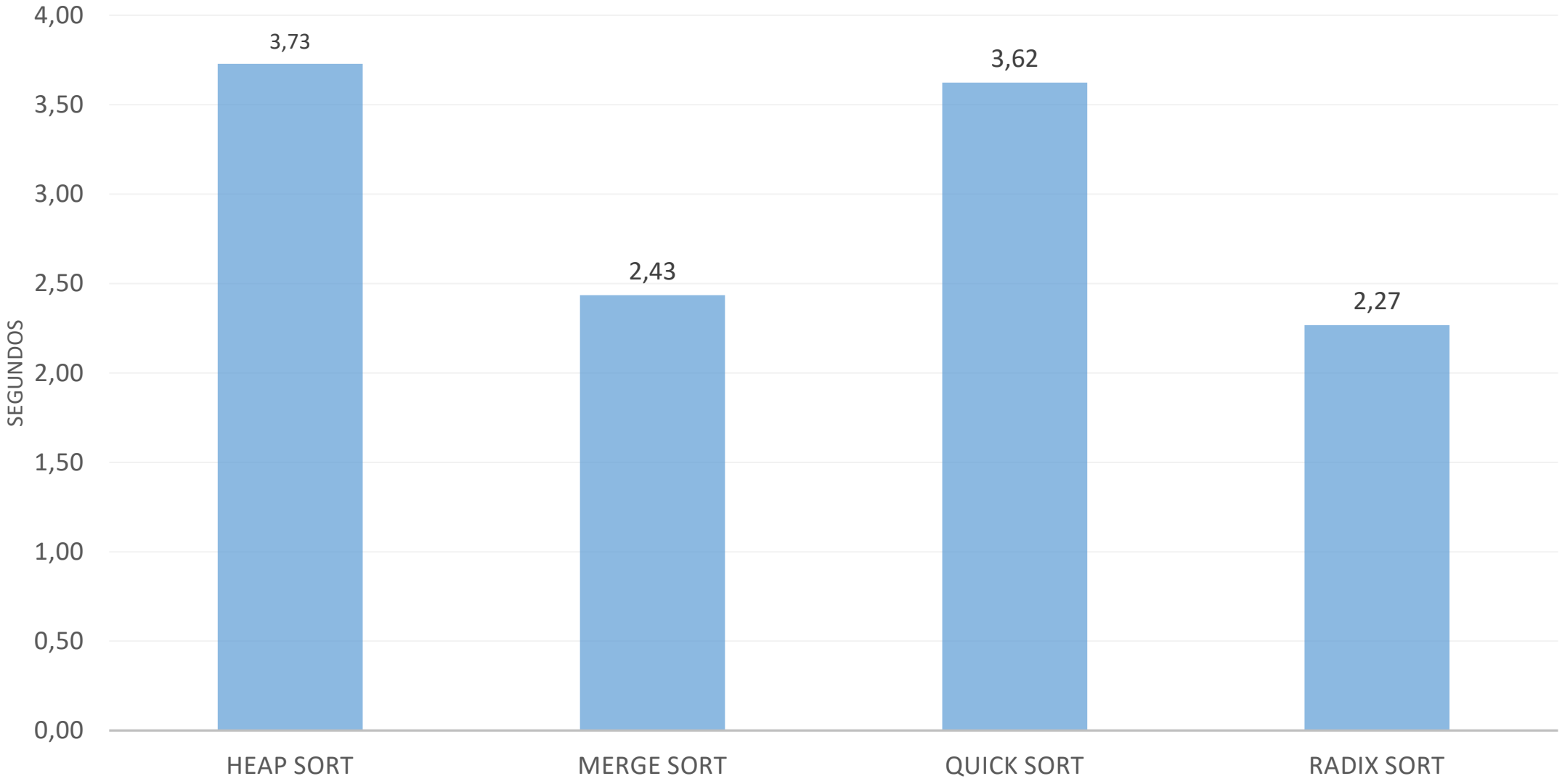
RADIX SORT

- **H_0**: O tempo de ordenação utilizando C# é **mais significativo** que o tempo de ordenação utilizando Java?
  - Teste-T:
    - Índice de confiança a 90%: [(2,49),(2,78)];
    - A implementação em C# é de fato mais significativa;
- **H_0**: O tempo de ordenação utilizando C# é **mais significativo** que o tempo de ordenação utilizando Python?
  - Teste-T:
    - Índice de confiança a 90%: [(-199,56),(-594,58)];
    - A implementação em C# é de fato mais significativa;

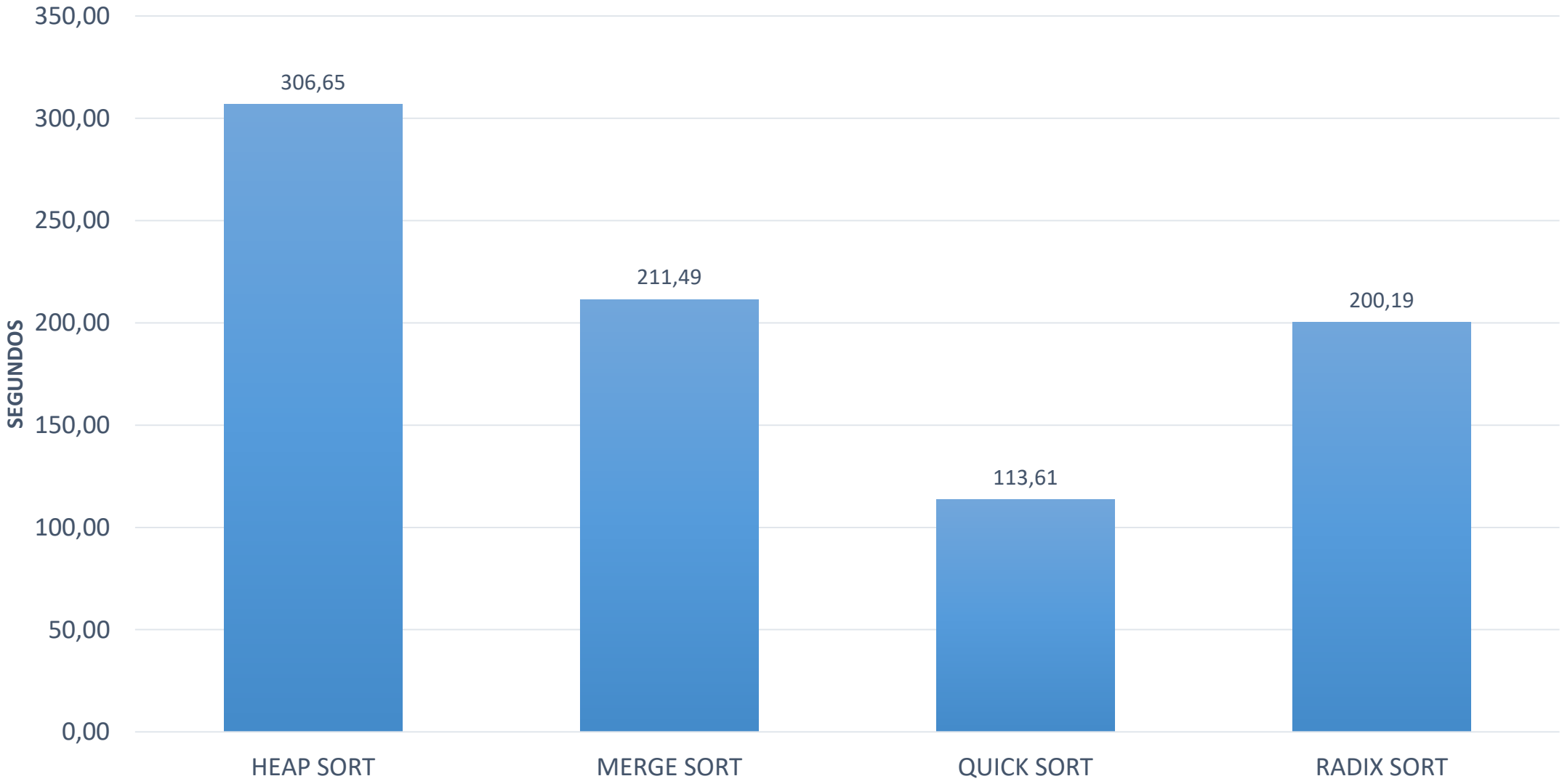
ALGORITMOS EM C#



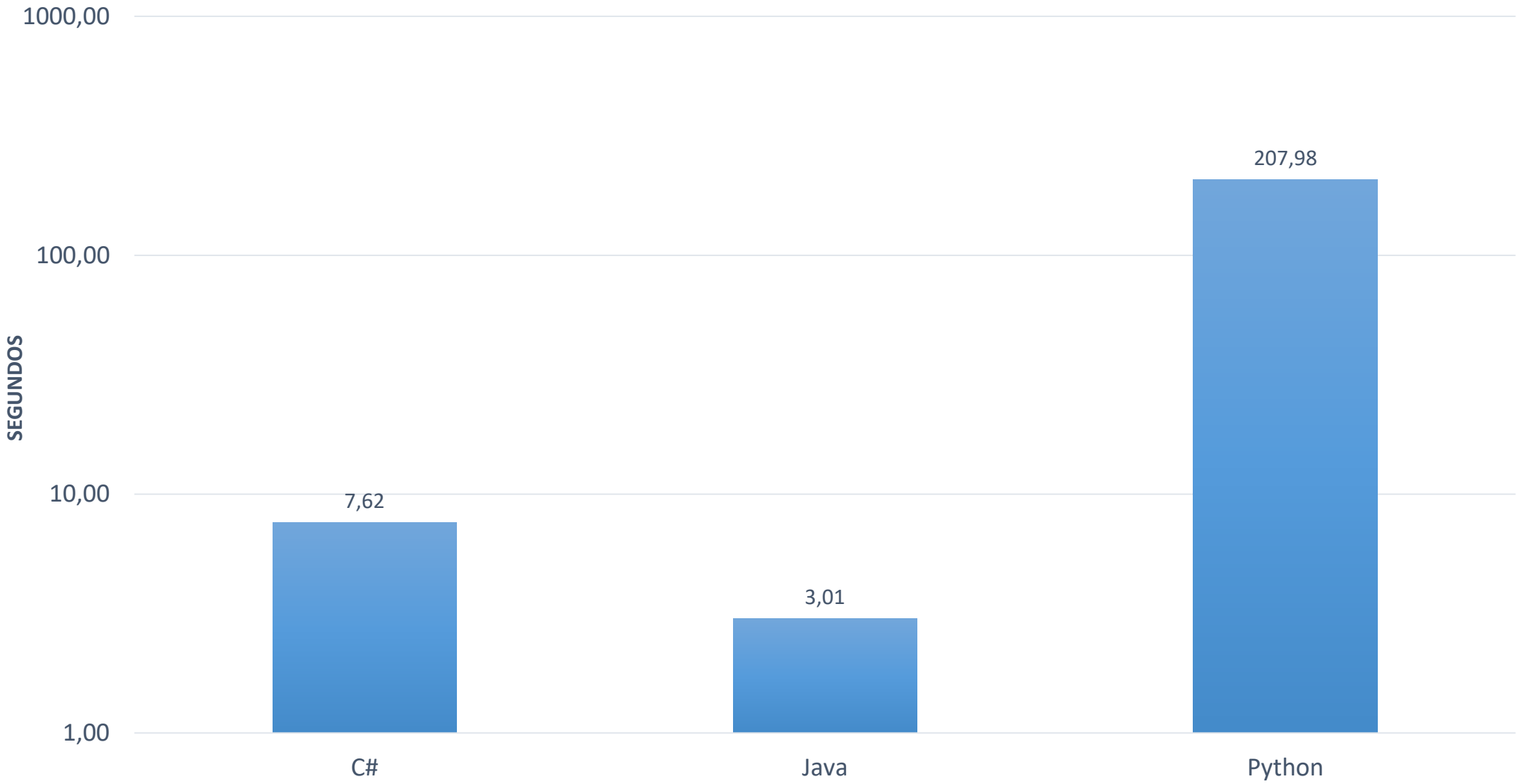
ALGORITMOS EM JAVA



ALGORITMOS EM PYTHON



ALGORITMOS – VISÃO GERAL



ALGORITMOS – VISÃO GERAL

- Implementação em Java
  - Merge Sort vs Radix Sort
    - Não houve significância;
      - Teste-T
        - Índice de confiança a 90%: [(-0,026),(0,362)];

CONCLUSÃO

CONCLUSÃO

- Java mostrou ser mais rápida na ordenação do conjunto de números;
  - Realizar o mesmo experimento na linguagem **c** e **c++**;
- Quick Sort foi o algoritmo que demandou menos tempo de ordenação;
  - O (n^2)
- Houve uma diferença considerável entre o tempo de ordenação utilizando Heap Sort e Merge Sort;
  - Θ (n log n);
- Trabalhos futuros com novos testes em outros fatores de computação;

ANÁLISE DO TEMPO DE EXECUÇÃO DE ALGORITMOS DE ORDENAÇÃO EM DIFERENTES LINGUAGENS

Leomar Camargo de Souza
Métodos Quantitativos em Computação
Pós-Graduação em Informática
Universidade de Brasília