

QUICKSORT

Estudos de Casos: Variações de Pivô

Algoritmo e Estrutura de Dados I

Docente: Michel Pires da Silva



Engenharia de Computação

Álvaro Augusto José Silva
Arthur de Oliveira Mendonça
Arthur Santana de Mesquita
Júlia de Moura Souza
Luiz Fernando dos Santos Queiroz

Divinópolis - MG

2025

SUMÁRIO

1 Algoritmo Quick Sort

1.1 Origem

1.2 O Algoritmo

2 Manipulação

2.1 Abordagem de Ordenação

2.2 Custo Computacional

3 Aplicabilidade

3.1 Vantagens e Desvantagens

3.2 Exemplos de aplicações

4 Análise Comportamental

4.1 Elementos Crescentes

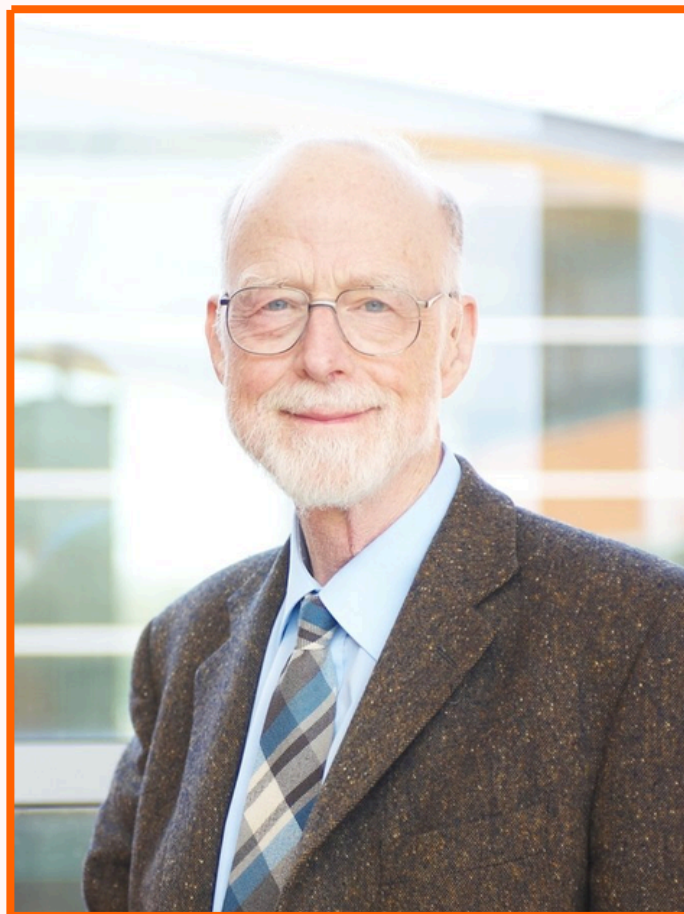
4.2 Elementos Decrescentes

4.3 Elementos Aleatórios

5 Conclusão

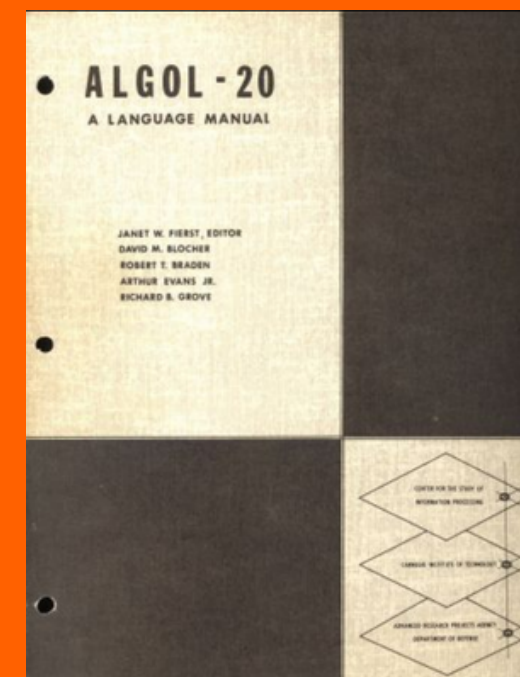
1 ALGORITMO QUICK SORT

1.1 Origem

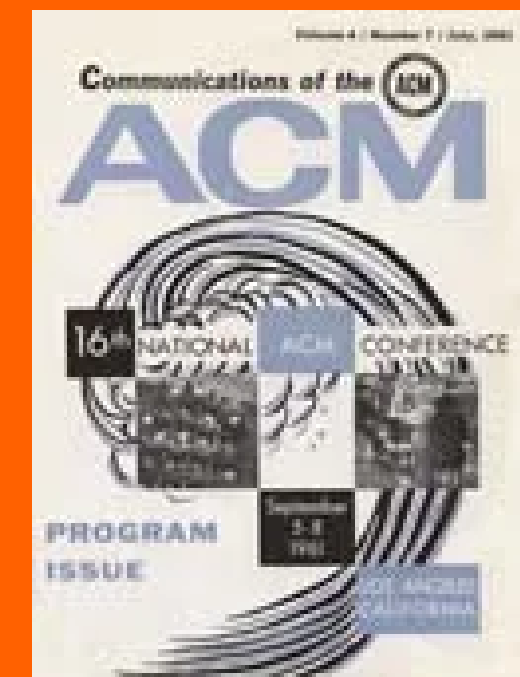


Sir Charles Antony Richard Hoare
(Tony Hoare)

- Rotinas de ordenação para um compilador (Tradutor)
- Eficiência prática (Quicksort não era recursivo)



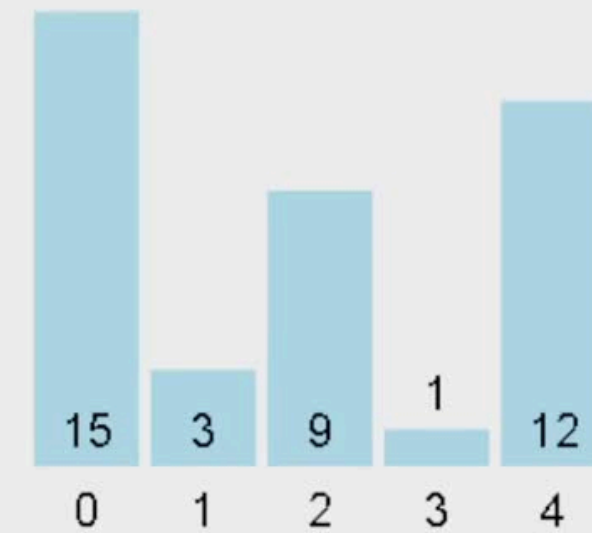
Linguagem Algorítmica
(ALGOL)



Communications of ACM
(ALGOL)

1 ALGORITMO QUICK SORT 1.1 0 Algoritmo

- Divide and Conquer (Recursividade)
- Ordenação **In-Place**
- Elemento de referência (Pivô)



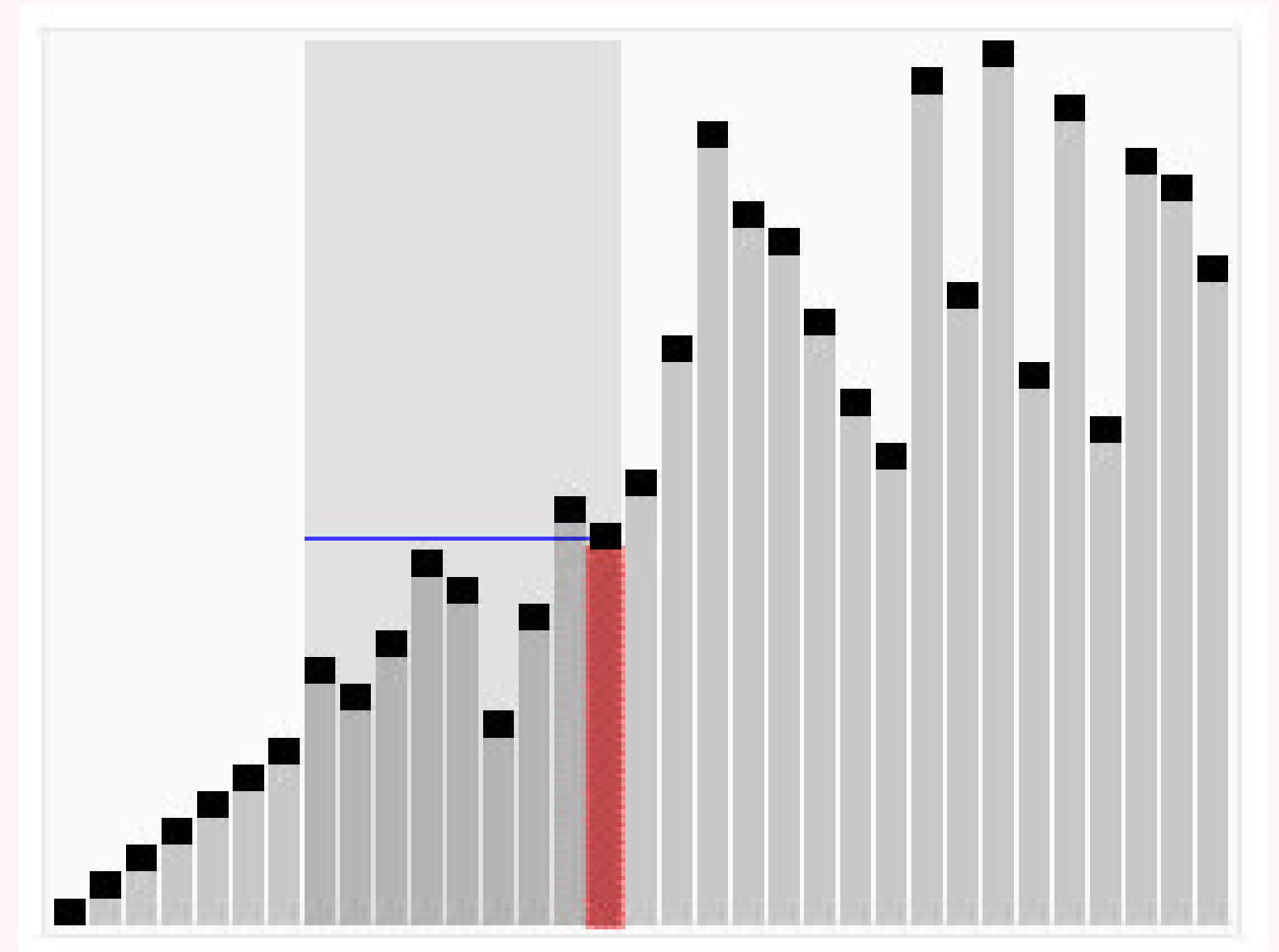
2 MANIPULAÇÃO

2.1 Abordagem de Ordenação

```
1: algoritmo Quicksort(V, início, fim)
2: if início  $\geq$  fim then
3:    $p \leftarrow$  Particiona(V, início, fim)
4:   Quicksort(V, início, p - 1)
5:   Quicksort(V, p + 1, fim)
6: end if
```

```
7: função Particiona(V, início, fim)
8:   pivo  $\leftarrow$  V[fim]
9:    $i \leftarrow$  início - 1
10:  for j  $\leftarrow$  início to fim - 1 do
11:    if V[j]  $\leq$  pivo then
12:       $i \leftarrow i + 1$ 
13:      Trocar(V[i], V[j])
14:    end if
15:  end for
16:  Trocar(V[i + 1], V[fim])
17:  retorna i + 1
```

```
18: procedimento Trocar(ref a, ref b)
19:   temp  $\leftarrow$  a
20:   a  $\leftarrow$  b
21:   b  $\leftarrow$  temp
```



2 MANIPULAÇÃO

2.1 Abordagem de Ordenação

```
1: algoritmo QuickSort_MedianaDeTres(vetor, inicio, fim)
2: if  $inicio < fim$  then
3:   pivô  $\leftarrow$  MedianaDeTres(vetor, inicio, fim)
4:   pos_pivô  $\leftarrow$  Particionar(vetor, inicio, fim, pivô)
5:   QuickSort_MedianaDeTres(vetor, inicio, pos_pivô - 1)
6:   QuickSort_MedianaDeTres(vetor, pos_pivô + 1, fim)
7: end if
```

```
8: função MedianaDeTres(vetor, inicio, fim)
9: meio  $\leftarrow \lfloor (inicio + fim)/2 \rfloor$ 
10: {Ordena os três elementos: inicio, meio, fim}
11: if  $vetor[inicio] > vetor[meio]$  then
12:   Trocar(vetor[inicio], vetor[meio])
13: end if
14: if  $vetor[meio] > vetor[fim]$  then
15:   Trocar(vetor[meio], vetor[fim])
16: end if
17: if  $vetor[inicio] > vetor[meio]$  then
18:   Trocar(vetor[inicio], vetor[meio])
19: end if
20: {A mediana está agora em vetor[meio]}
21: retorna vetor[meio]
```

```
22: função Particionar(vetor, inicio, fim, pivô)
23:  $i \leftarrow inicio$ 
24:  $j \leftarrow fim$ 
25: while  $i \leq j$  do
26:   while  $vetor[i] < pivô$  do
27:      $i \leftarrow i + 1$ 
28:   end while
29:   while  $vetor[j] > pivô$  do
30:      $j \leftarrow j - 1$ 
31:   end while
32:   if  $i \leq j$  then
33:     Trocar(vetor[i], vetor[j])
34:      $i \leftarrow i + 1$ 
35:      $j \leftarrow j - 1$ 
36:   end if
37: end while
38: retorna  $i$ 
```

```
39: procedimento Trocar(ref a, ref b)
40: temp  $\leftarrow a$ 
41:  $a \leftarrow b$ 
42:  $b \leftarrow temp$ 
```

2 MANIPULAÇÃO

2.3 Custo Computacional

Custo de Armazenamento: $O(n)$

- Pior Caso (Pivô Extremo): $O(n^2)$
 - Médio Caso (Comum): $O(n \cdot \log(n))$
 - Melhor Caso (Pivô Médio): $O(n \cdot \log(n))$
-

3 APLICAÇÕES

3.1 Vantagens e Desvantagens

VANTAGENS

- *In-Place*
- Ordenação por Prioridade
- Complexidade de Tempo Consistente
- Implementação
- Paralelização

DESVANTAGENS

- Recursividade
- Instabilidade
- Sensível à escolha do pivô

3 APLICAÇÕES

3.2 Exemplos de Aplicações

COMPILADORES

- Durante a ordenação de identificadores

BANCO DE DADOS

- Em bancos de dados que executam múltiplas tarefas simultaneamente

SISTEMAS EMBARCADOS

- Ideal para ambientes com RAM restrita, como microprocessadores

BIBLIOTECAS DE DIFERENTES LINGUAGENS

- Oferece alta eficiência prática na ordenação de arrays e listas em memória.

4 ANÁLISE COMPORTAMENTAL

4.1 Ratings x Timestamp

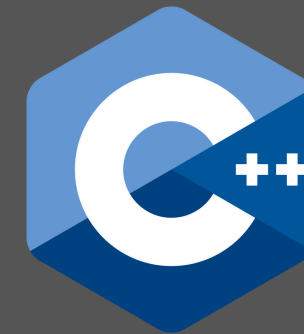
4.2 Lista Estática

4.3 Lista Dinâmica

4.4 Pilha

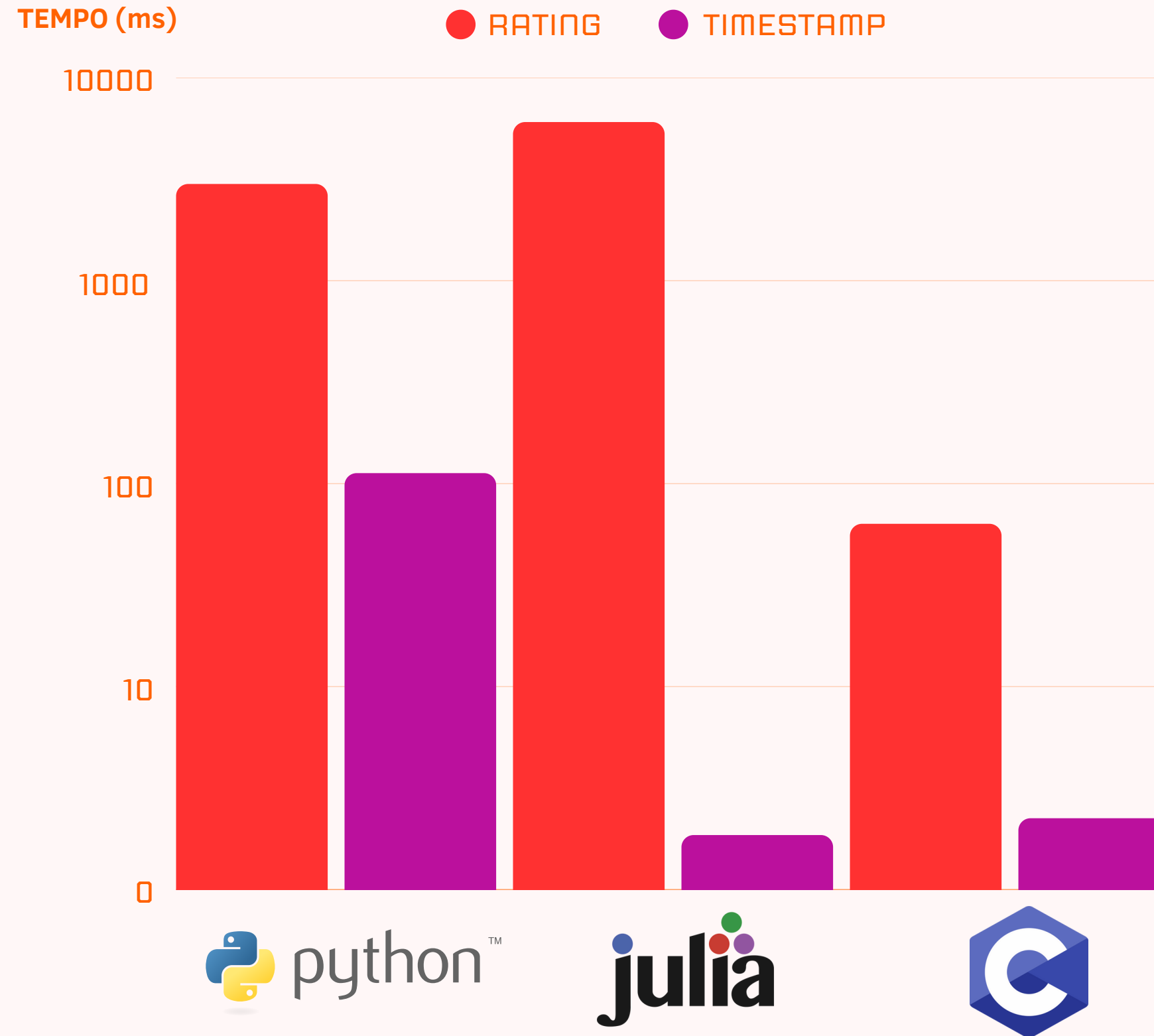
4.5 Fila

LINGUAGENS UTILIZADAS:



4 ANÁLISE COMPORTAMENTAL

4.1 RATING X TIMESTAMP



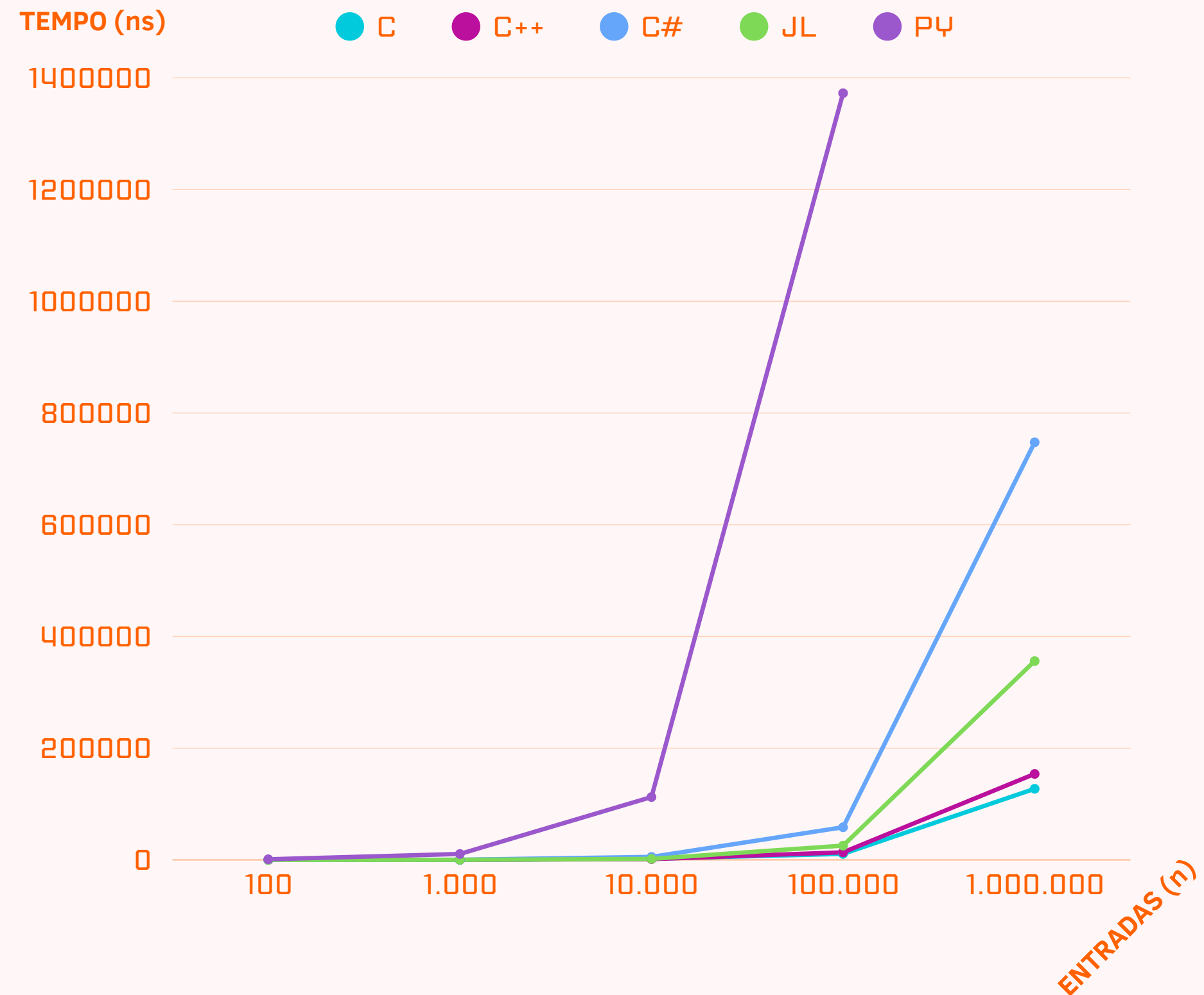
4 ANÁLISE COMPORTAMENTAL

4.2 LISTA ESTÁTICA

Tamanho	Python	Julia	C#	C	C++
100	0,001200	0,0000882	0,000039	0,000045	0,000063
1.000	0,010700	0,0002041	0,000387	0,000100	0,000237
10.000	0,112600	0,0018548	0,005492	0,002246	0,001479
100.000	1,372500	0,0256452	0,058507	0,010848	0,013609
1.000.000	15,691050	0,3560414	0,747585	0,127346	0,153992

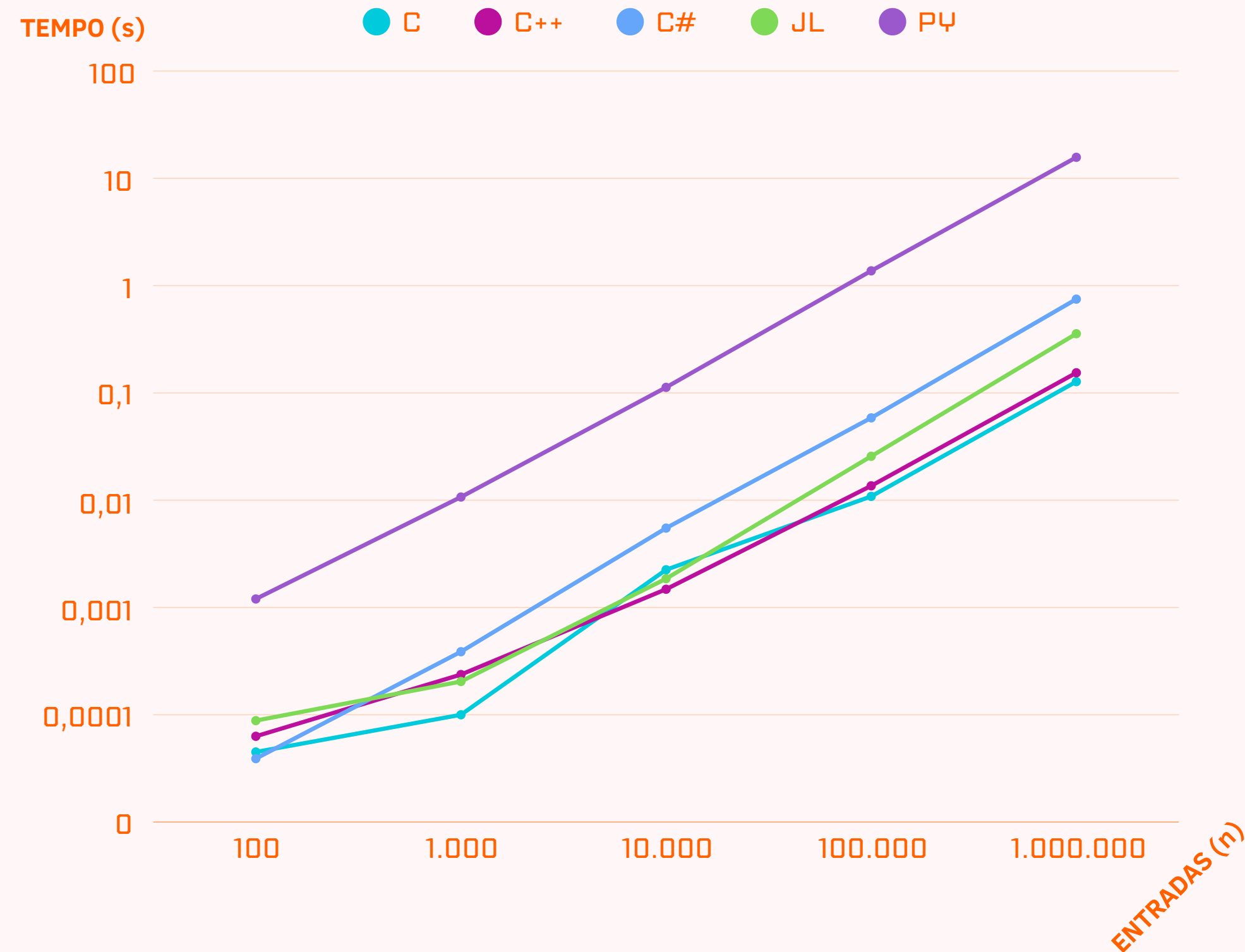
4 ANÁLISE COMPORTAMENTAL

4.2 LISTA ESTÁTICA



4 ANÁLISE COMPORTAMENTAL

4.2 LISTA ESTÁTICA (ESCALA LOGARÍTMICA)



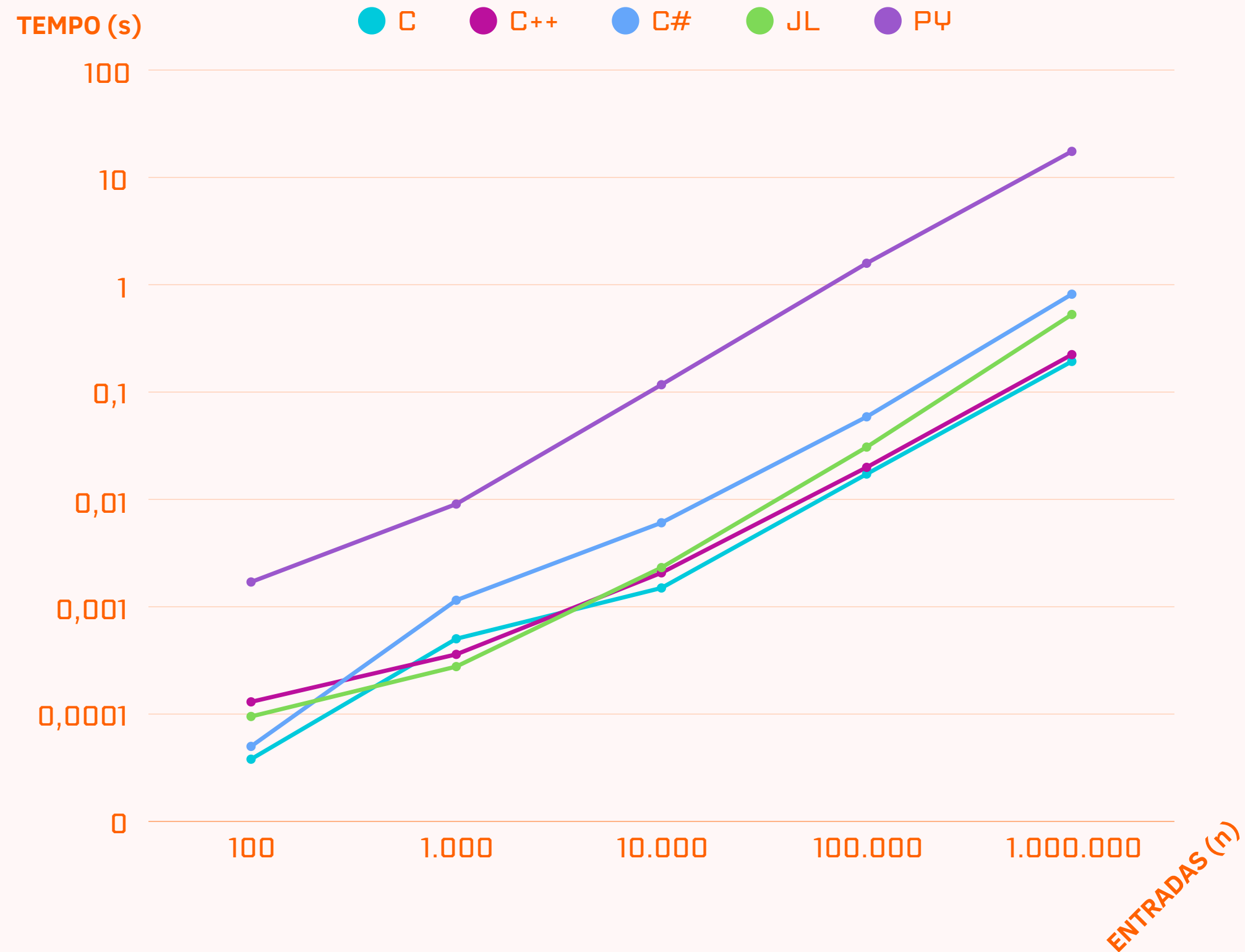
4 ANÁLISE COMPORTAMENTAL

4.3 LISTA DINÂMICA

Tamanho	Python	Julia	C#	C	C++
100	0,00170	0,000095	0,00005	0,000038	0,00013
1.000	0,00905	0,000277	0,00115	0,000503	0,00036
10.000	0,11710	0,002316	0,00606	0,001496	0,00207
100.000	1,58545	0,030721	0,05881	0,017228	0,01989
1.000.000	17,47410	0,528310	0,81629	0,192995	0,22331

4 ANÁLISE COMPORTAMENTAL

4.3 LISTA DINÂMICA



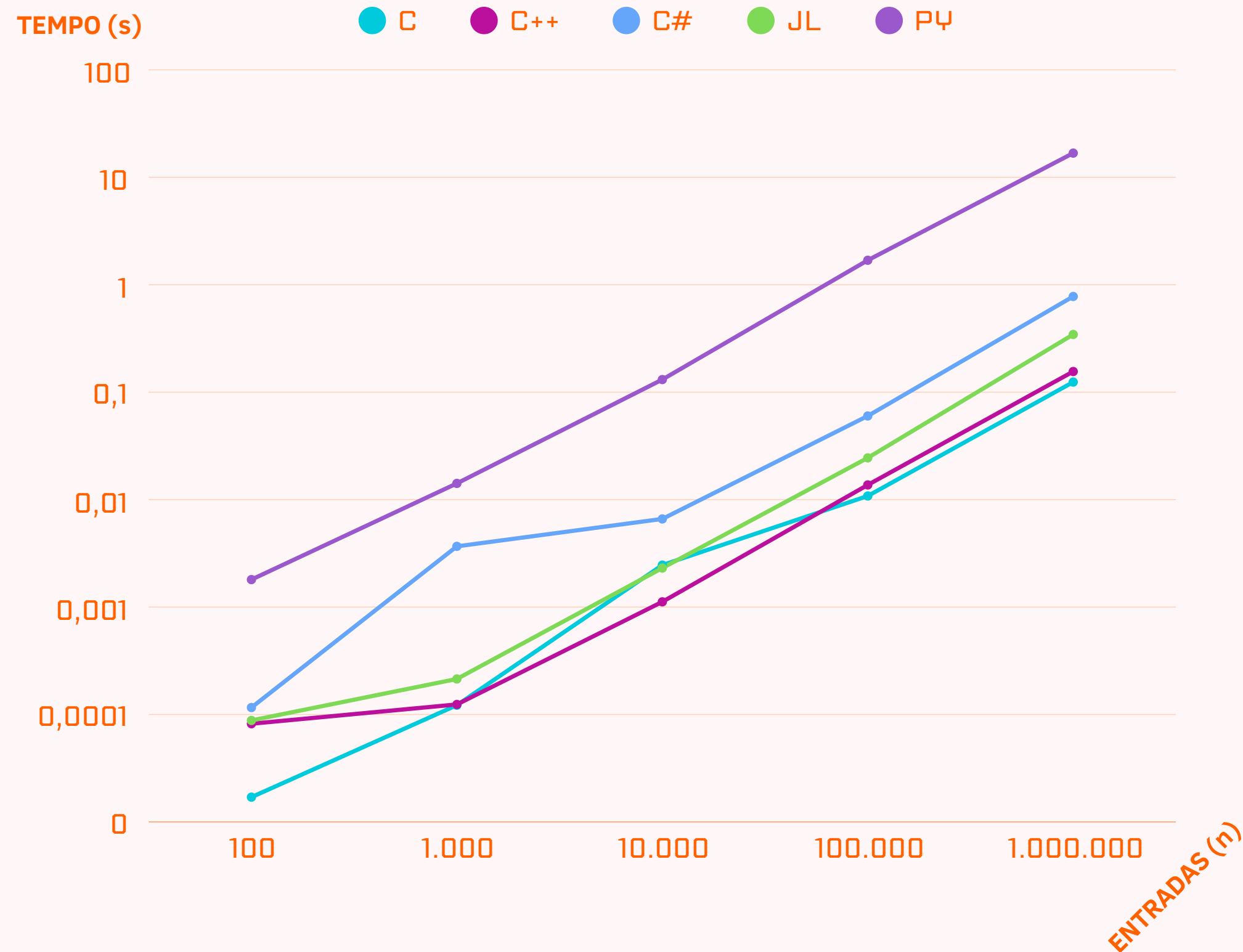
4 ANÁLISE COMPORTAMENTAL

4.4 PILHA ESTÁTICA

Tamanho	Python	Julia	C#	C	C++
100	0,001800	0,000088	0,000116	0,000017	0,000082
1.000	0,014150	0,000214	0,003671	0,000122	0,000124
10.000	0,131000	0,002305	0,006607	0,002457	0,001118
100.000	1,689400	0,024462	0,060055	0,010808	0,013696
1.000.000	16,785150	0,344365	0,777399	0,124157	0,155643

4 ANÁLISE COMPORTAMENTAL

4.4 PILHA ESTÁTICA



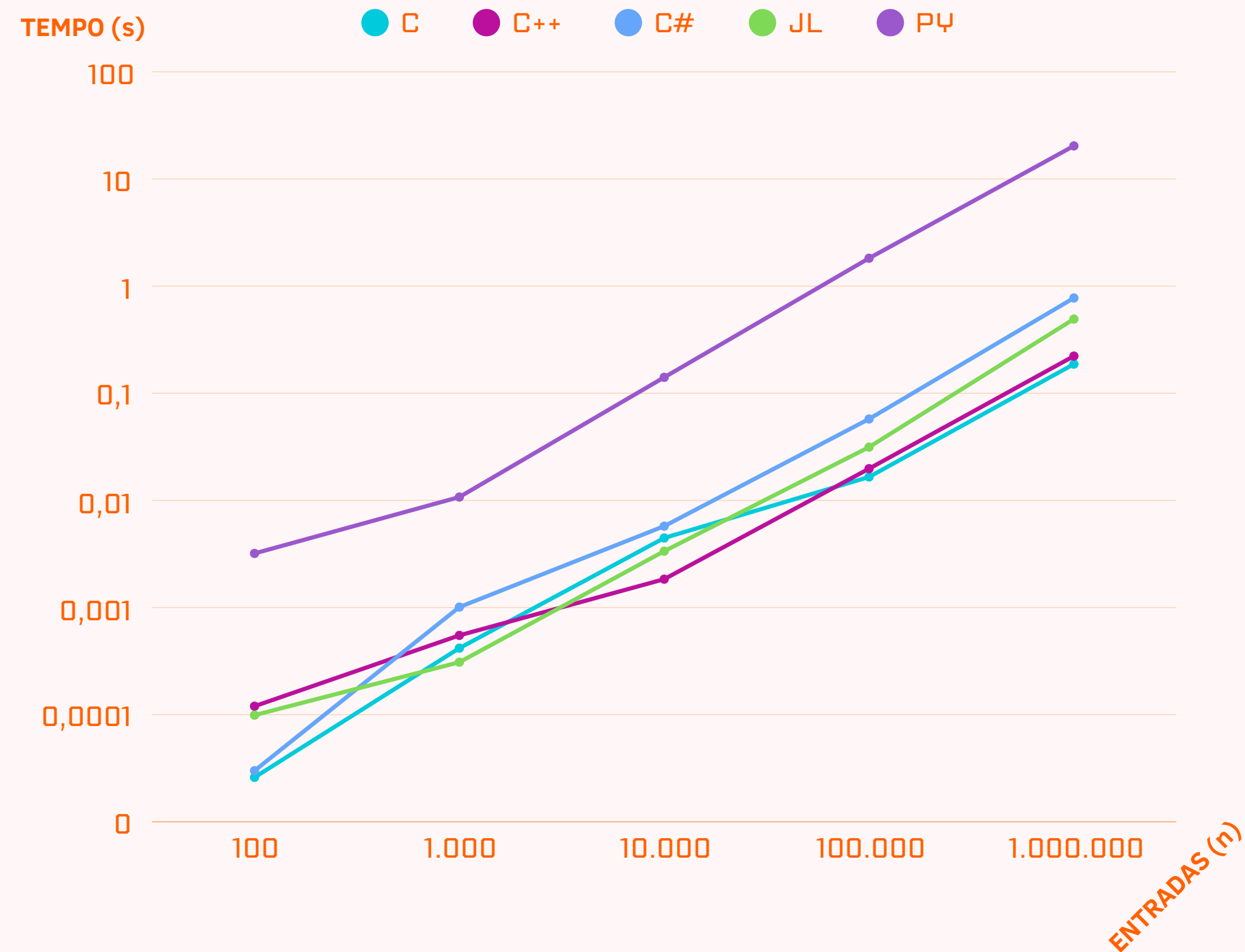
4 ANÁLISE COMPORTAMENTAL

4.4 PILHA DINÂMICA

Tamanho	Python	Julia	C#	C	C++
100	0,00320	0,000099	0,00003	0,000026	0,00012
1.000	0,01075	0,000309	0,00101	0,000418	0,00055
10.000	0,14085	0,003359	0,00575	0,004460	0,00184
100.000	1,82170	0,031429	0,05753	0,016554	0,01978
1.000.000	20,36265	0,493027	0,77498	0,187110	0,22237

4 ANÁLISE COMPORTAMENTAL

4.4 PILHA DINÂMICA



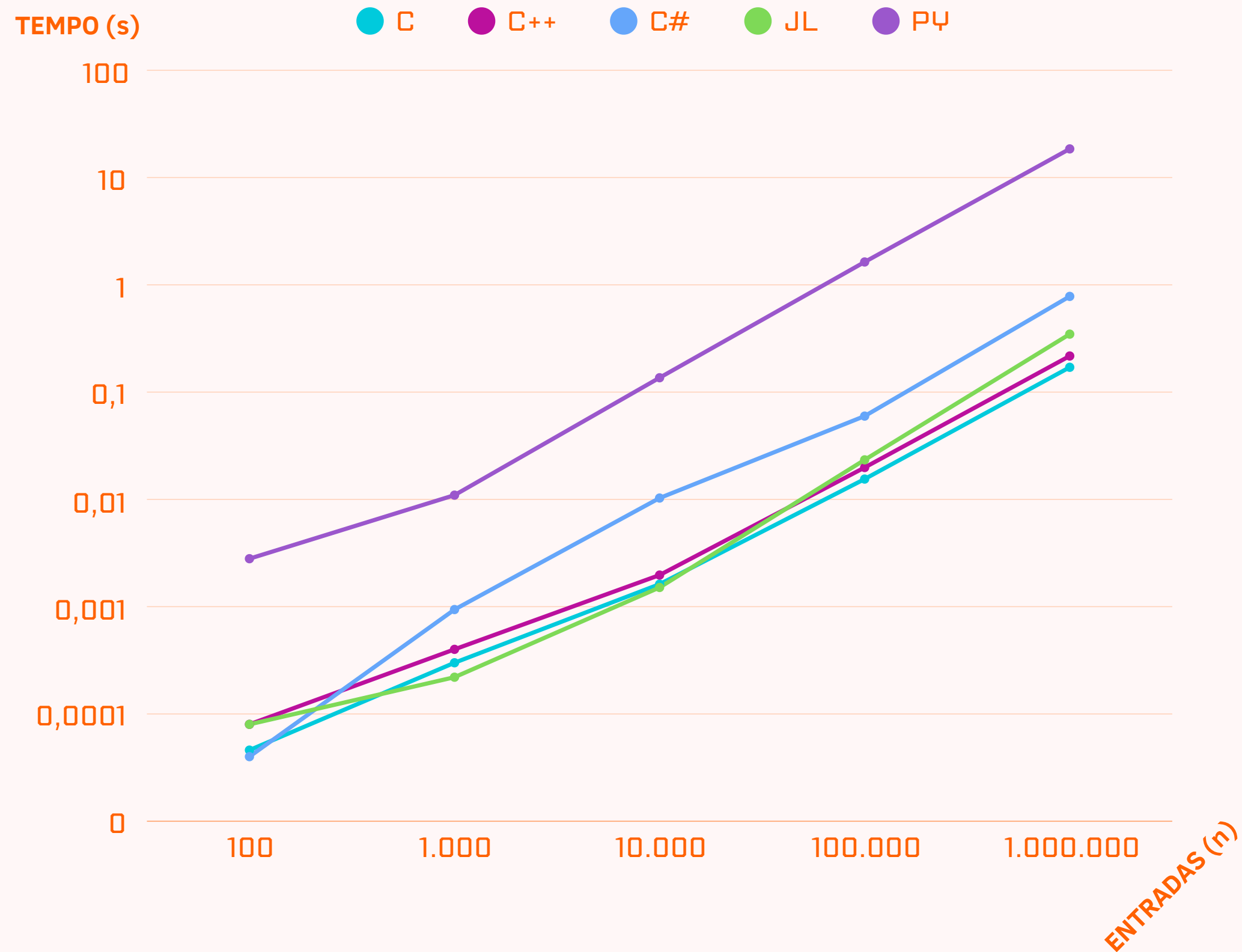
4 ANÁLISE COMPORTAMENTAL

4.5 FILA ESTÁTICA

Tamanho	Python	Julia	C#	C	C++
100	0,00280	0,00008	0,00004	0,000046	0,00008
1.000	0,01095	0,00022	0,00094	0,00030	0,00040
10.000	0,13615	0,00151	0,01029	0,00162	0,00197
100.000	1,63270	0,02336	0,05975	0,01547	0,01982
1.000.000	18,49560	0,34703	0,78017	0,17034	0,21705

4 ANÁLISE COMPORTAMENTAL

4.5 FILA ESTÁTICA



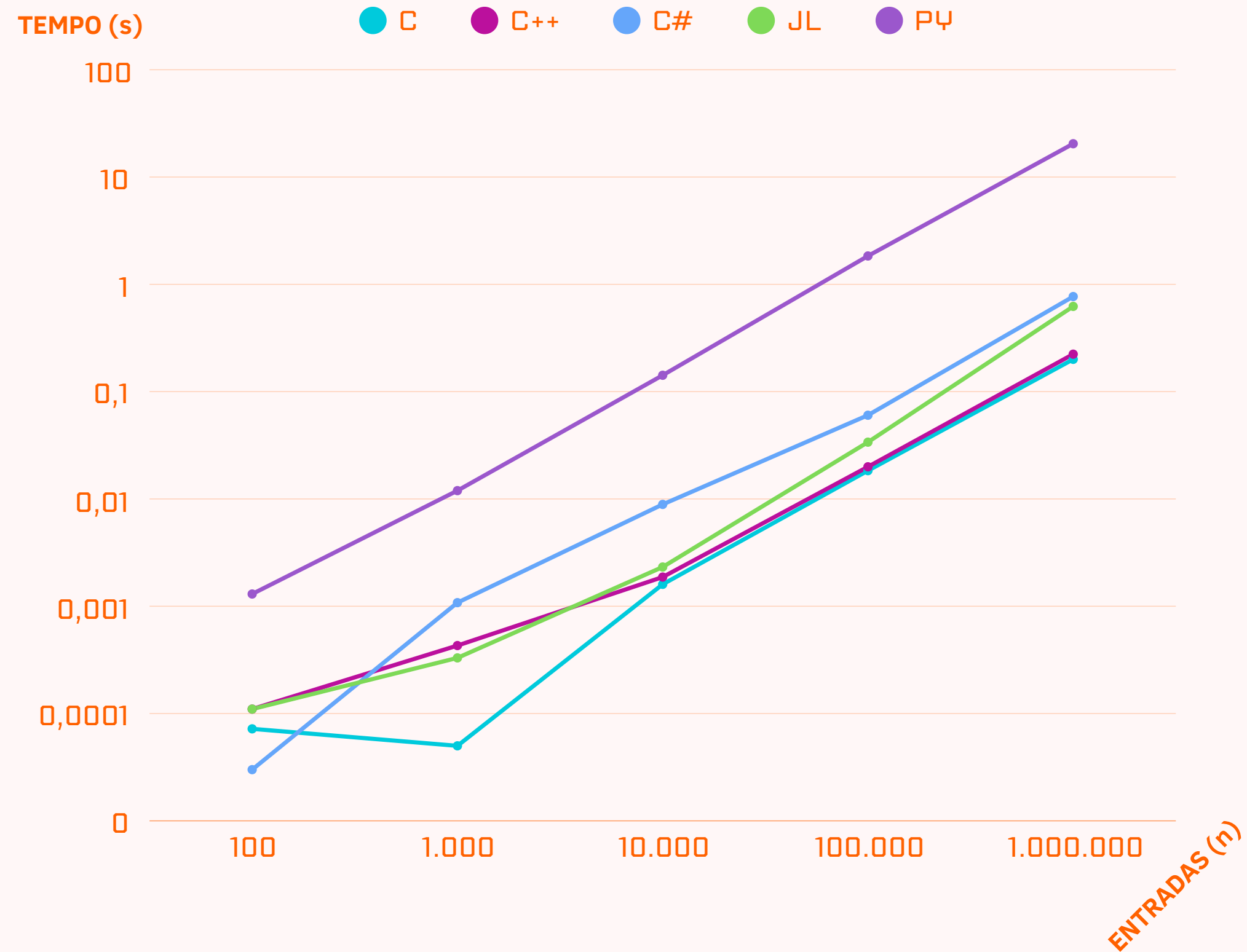
4 ANÁLISE COMPORTAMENTAL

4.5 FILA DINÂMICA

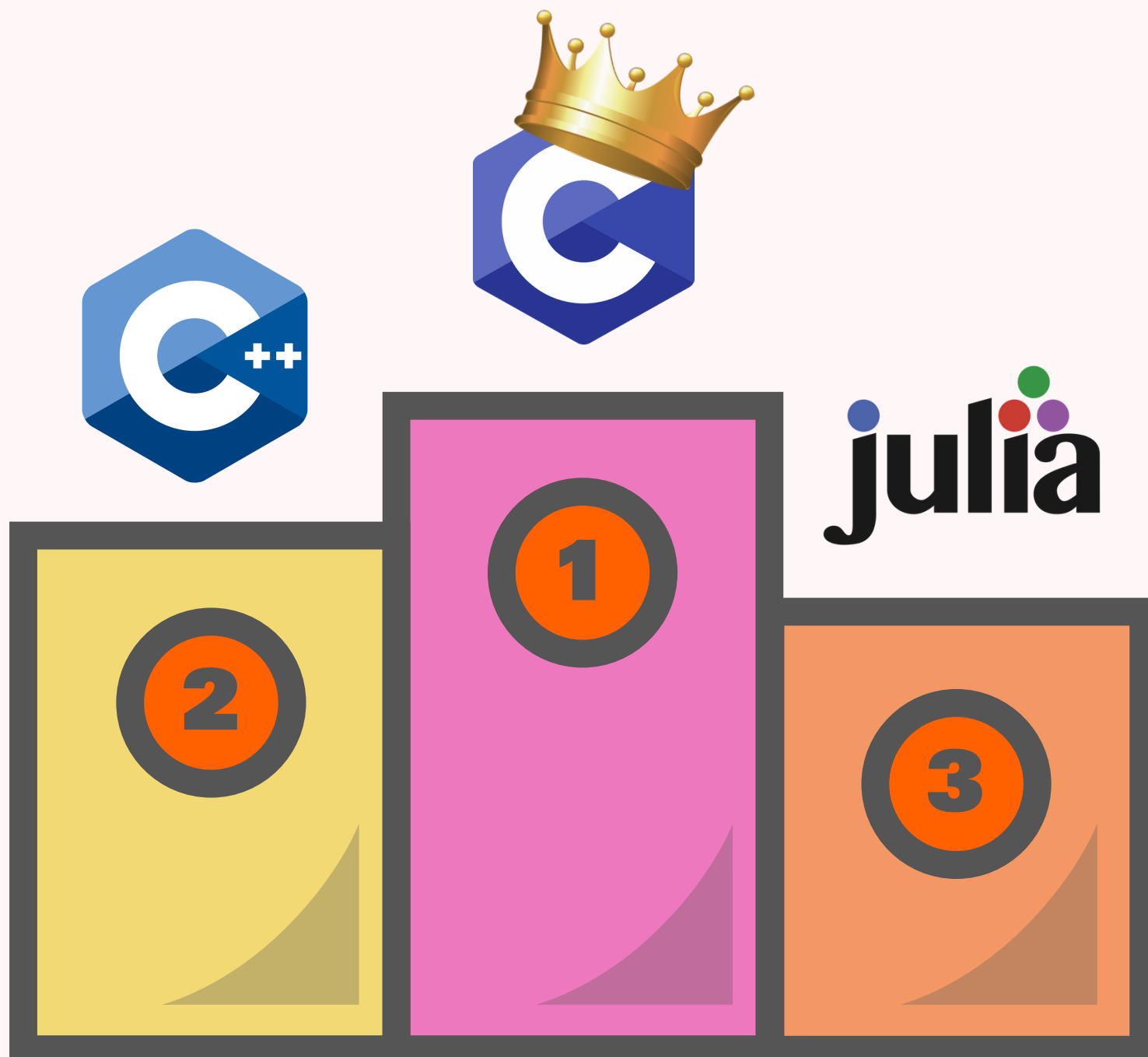
Tamanho	Python	Julia	C#	C	C++
100	0,00170	0,000095	0,00005	0,000038	0,00013
1.000	0,00905	0,000277	0,00115	0,000503	0,00036
10.000	0,11710	0,002316	0,00606	0,001496	0,00207
100.000	1,58545	0,030721	0,05881	0,017228	0,01989
1.000.000	17,47410	0,528310	0,81629	0,192995	0,22331

4 ANÁLISE COMPORTAMENTAL

4.5 FILA DINÂMICA



5 CONCLUSÃO



- **Importância da Base de Dados**
- **Estrutura mais compatível**
- **Linguagens mais eficientes.**
- **Futuras melhorias para o desenvolvimento da discussão.**

REFERÊNCIAS

- SILVA, A. A. Algoritmo Quick Sort. URL: <https://github.com/alvaroajs/ordenacaoAEDS>.
- HOARE, C. A. R. "Algorithm 64: Quicksort". Em: Communications of the ACM 4.7 (1961), p. 321.
- CORMEN, T. H. Algoritmos - Teoria e Pratica. MITPress, 1989.
- KNUTH, D. E. The Art of Computer Programming, Volume Sorting and Searching. 2nd. Addison-Wesley Professional, 1998.
- BENTLEY, J. L. and MCILROY, M. D. "Engineering a Sort Function". Em: Software: Practice and Experience 23.11 (1993), pp. 1249–1265.
- MUSSER, D. R. "Introspective Sorting and Selection Algorithms". Em: Software: Practice and Experience 27.8 (1997), pp. 983–993.
- SO/IEC. C Standard. <https://en.cppreference.com/w/c>.
- The Julia Language. Julia Documentation. <https://docs.julialang.org/>.
- Python Software Foundation. Python Documentation. <https://docs.python.org/3/>.
- Microsoft Corporation. C Language Specification. <https://learn.microsoft.com/en-us/dotnet/csharp/language-reference/specification>
- SO/IEC. C++ Standard. <https://en.cppreference.com/w/cpp>.
- TORVALDS, L. Linux Kernel - lib/sort.c. Código-fonte do Linux Kernel, implementado com Heapsort. 2024. URL: <https://github.com/torvalds/linux/blob/master/lib/sort.c>.

Obrigado !!!

Dúvidas ?