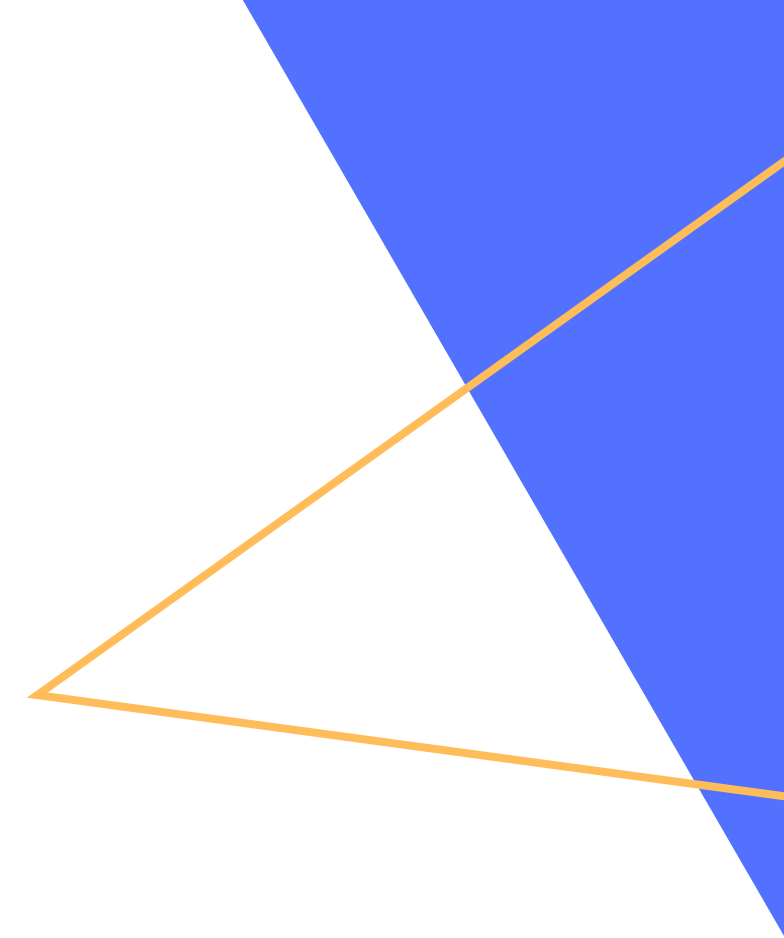



Teoria da Complexidade e Análise de Tempo de Algoritmos

PROFS. PÂMELA E DANIEL





Grupo

- Lucas Santos
- Lucas Souto
- Rodrigo Nunes

Algoritmo

- Quick Sort

Linguagens

- Python
- Java

Descrição do Algoritmo

Funcionamento - Divisão e conquista

- Escolhe um elemento como pivô
- Particiona a lista em dois subconjuntos:
 - Menores que o pivô
 - Maiores que o pivô
- Ordena recursivamente os subconjuntos

Pseudocódigo

Representa funcionamento básico do algoritmo:

```
QUICKSORT(A, baixo, alto)
  se baixo < alto
    pivo ← PARTITION(A, baixo, alto)
    QUICKSORT(A, baixo, pivo-1)
    QUICKSORT(A, pivo+1, alto)
```

```
PARTITION(A, baixo, alto)
  pivo ← A[alto]
  i ← baixo - 1
  para j ← baixo até alto-1
    se  $A[j] \leq \text{pivo}$ 
      i ← i + 1
      trocar A[i] e A[j]
  trocar A[i+1] e A[alto]
  retornar i+1
```



Escolha do Pivô

Python

Pivô é o elemento central da lista

- Melhor desempenho em listas parcialmente ordenadas

Java

Pivô é o último elemento da lista

- Simples de implementar, mas pode levar ao pior caso em listas ordenadas ou inversamente ordenadas



Resultados Experimentais

Entradas

100, 10.000, 1.000.000 elementos

Execuções

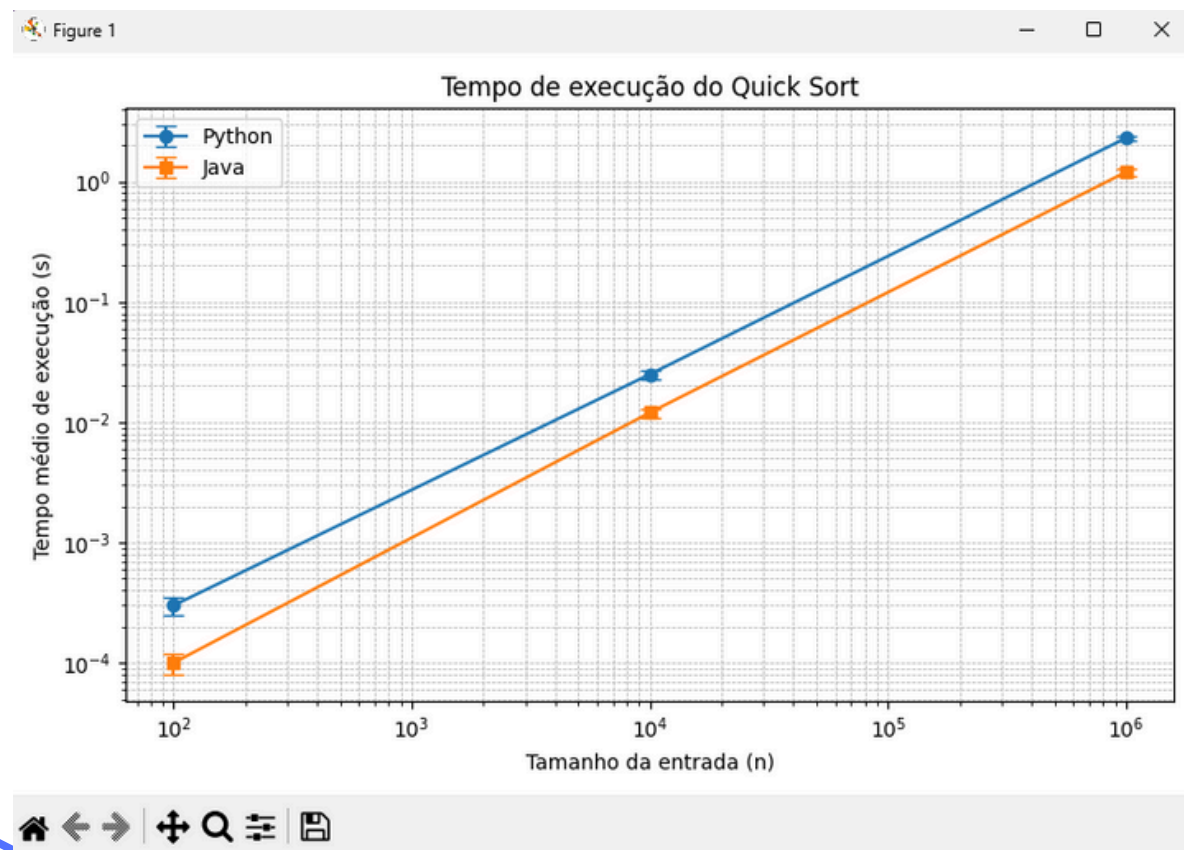
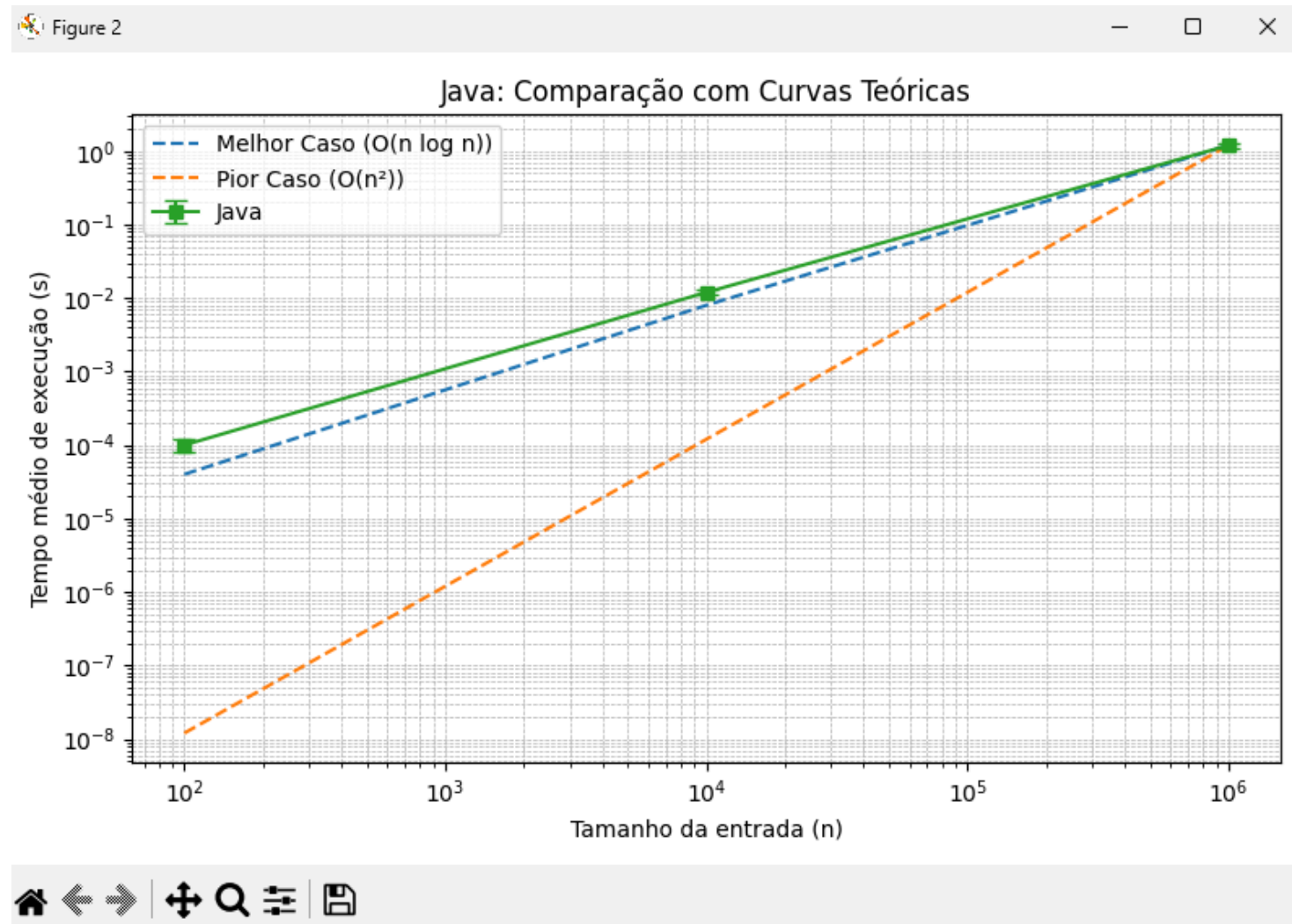
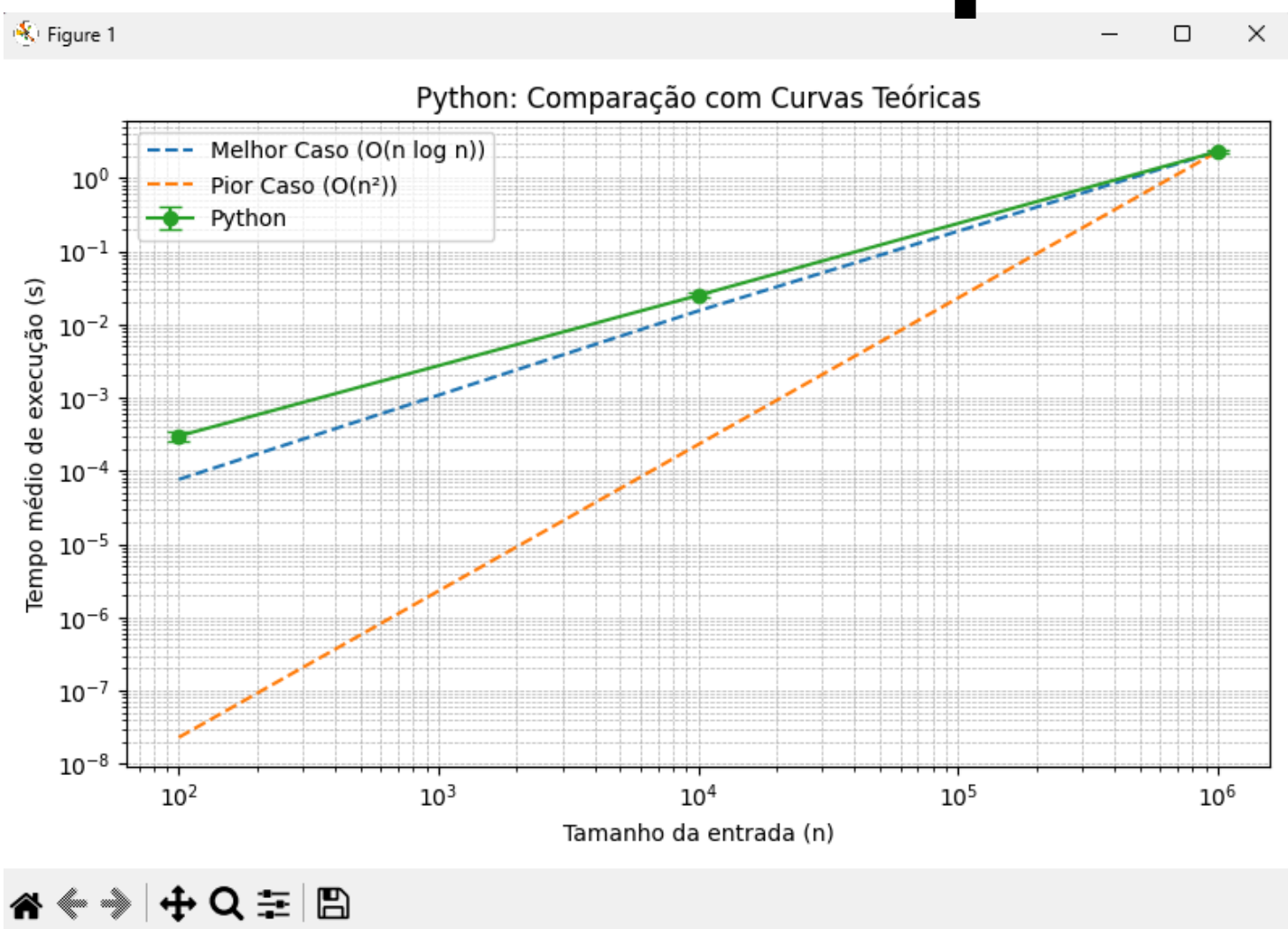
30 por teste

Tempos Médios e Desvios Padrão

Tamanho da Entrada (n)	Python - Tempo Médio (s)	Python - Desvio Padrão (s)	Java - Tempo Médio (s)	Java - Desvio Padrão (s)
100	0.0003	0.00005	0.0001	0.00002
10.000	0.025	0.002	0.012	0.001
1.000.000	2.3	0.12	1.2	0.08



Resultados Experimentais



Percebe-se que no pior caso (em laranja) há uma discrepância, evidenciando sua classificação assintótica diferente ($O n^2$).

Discussão e Aplicabilidade

Complexidade

- Melhor caso: $O(n \log n)$
- Caso médio: $\Theta(n \log n)$
- Pior caso: $O(n^2)$

Aplicabilidade

- Ampla utilização em grandes volumes de dados
- Python: Não é in-place, maior uso de memória
- Java: In-place, mais eficiente em memória

Otimizações possíveis

- Escolha de pivô aleatório ou mediana para evitar o pior caso



Conclusão

Reflexão Final

- **Quick Sort pertence à classe P (tempo polinomial)**
- **É eficiente na prática, mas pode ser otimizado**

Resultados

- **Python e Java apresentaram tempos consistentes com a teoria**
- **Gráficos e tabelas mostram a relação entre tamanho da entrada e desempenho**

Próximos passos

- Explorar outras estratégias de escolha de pivô
- Comparar com outros algoritmos de ordenação