Análise Teórica dos Algoritmos de Ordenação

Algoritmos de Complexidade Quadrática - O(n²)

Estes algoritmos são mais simples, mas sua eficiência cai drasticamente com o aumento dos dados. Seus melhores casos, no entanto, podem ser notavelmente rápidos.

1. Bubble Sort (Ordenação por Bolha)

Melhor Caso: O(n) — Ocorre quando a lista já está ordenada.

Caso Médio: O(n²)

Pior Caso: O(n²) — Ocorre quando a lista está em ordem inversa.

2. Selection Sort (Ordenação por Seleção)

Melhor Caso: O(n²)

Caso Médio: O(n²)

Pior Caso: O(n²) — O desempenho é consistente e não melhora para listas já ordenadas.

3. Insertion Sort (Ordenação por Inserção)

Melhor Caso: O(n) — Ocorre quando a lista já está ordenada.

Caso Médio: O(n²)

Pior Caso: O(n²) — Ocorre quando a lista está em ordem inversa.

Algoritmos de Complexidade Logarítmica Linear - O(n log n)

Esses algoritmos são muito mais eficientes e escaláveis para grandes volumes de dados. Seus piores e melhores casos costumam ser os mesmos, o que os torna previsíveis e confiáveis.

4. Merge Sort (Ordenação por Intercalação)

Melhor Caso: O(n log n)

Caso Médio: O(n log n)

Pior Caso: O(n log n) — Um dos algoritmos mais estáveis em termos de desempenho.

5. Quick Sort (Ordenação Rápida)

Melhor Caso: O(n log n) — Ocorre quando o pivô escolhido divide a lista em duas metades de tamanho quase igual.

Caso Médio: O(n log n)

Pior Caso: O(n²) — Ocorre em raras situações de péssima escolha de pivô.

6. Heap Sort (Ordenação por Monte)

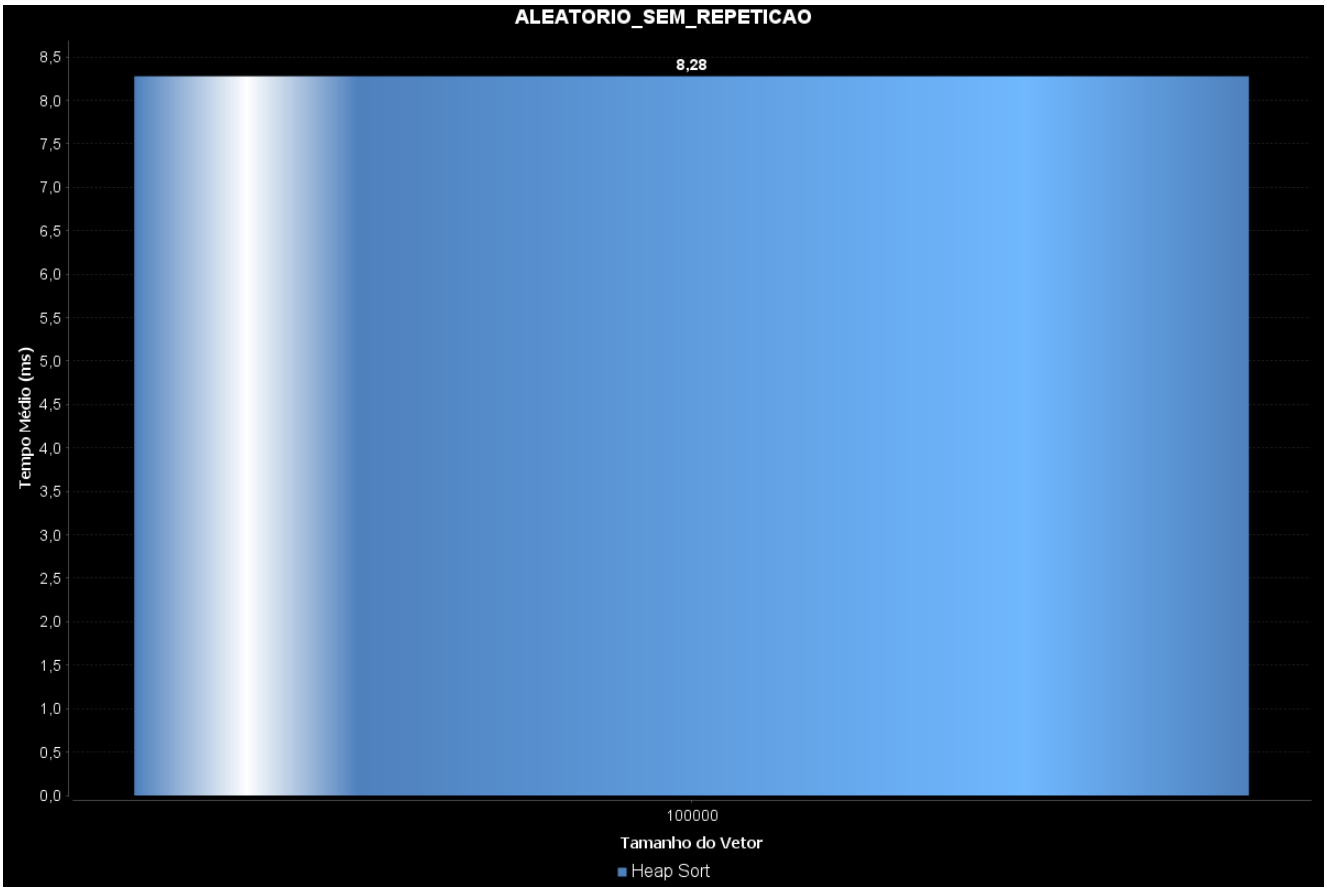
Melhor Caso: O(n log n)

Caso Médio: O(n log n)

Pior Caso: O(n log n) — Assim como o Merge Sort, seu desempenho é muito previsível.

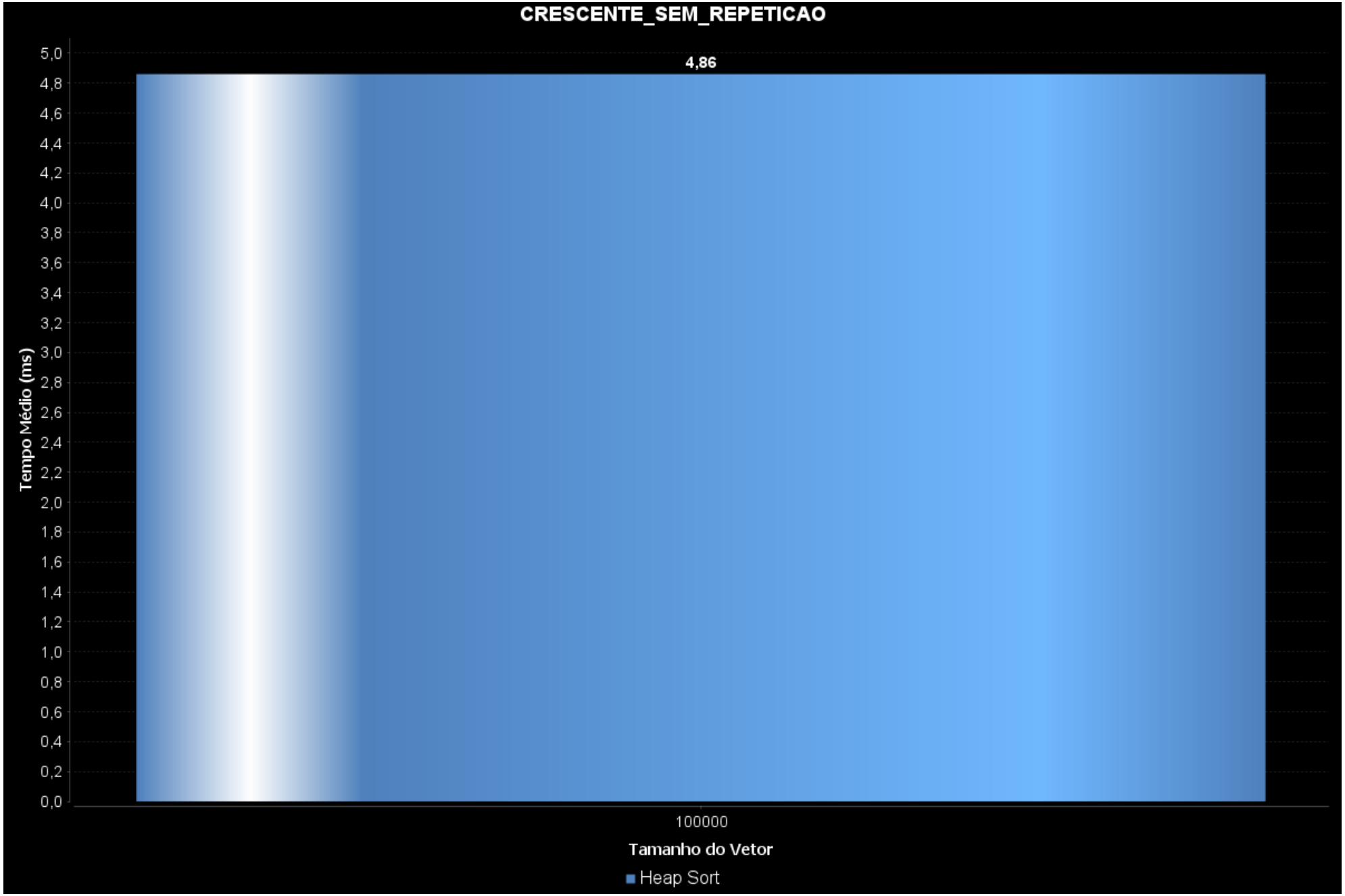
Relatório de Desempenho: ALEATORIO_SEM_REPETICAO

Algoritmo	100000
Heap Sort	8,28 ms



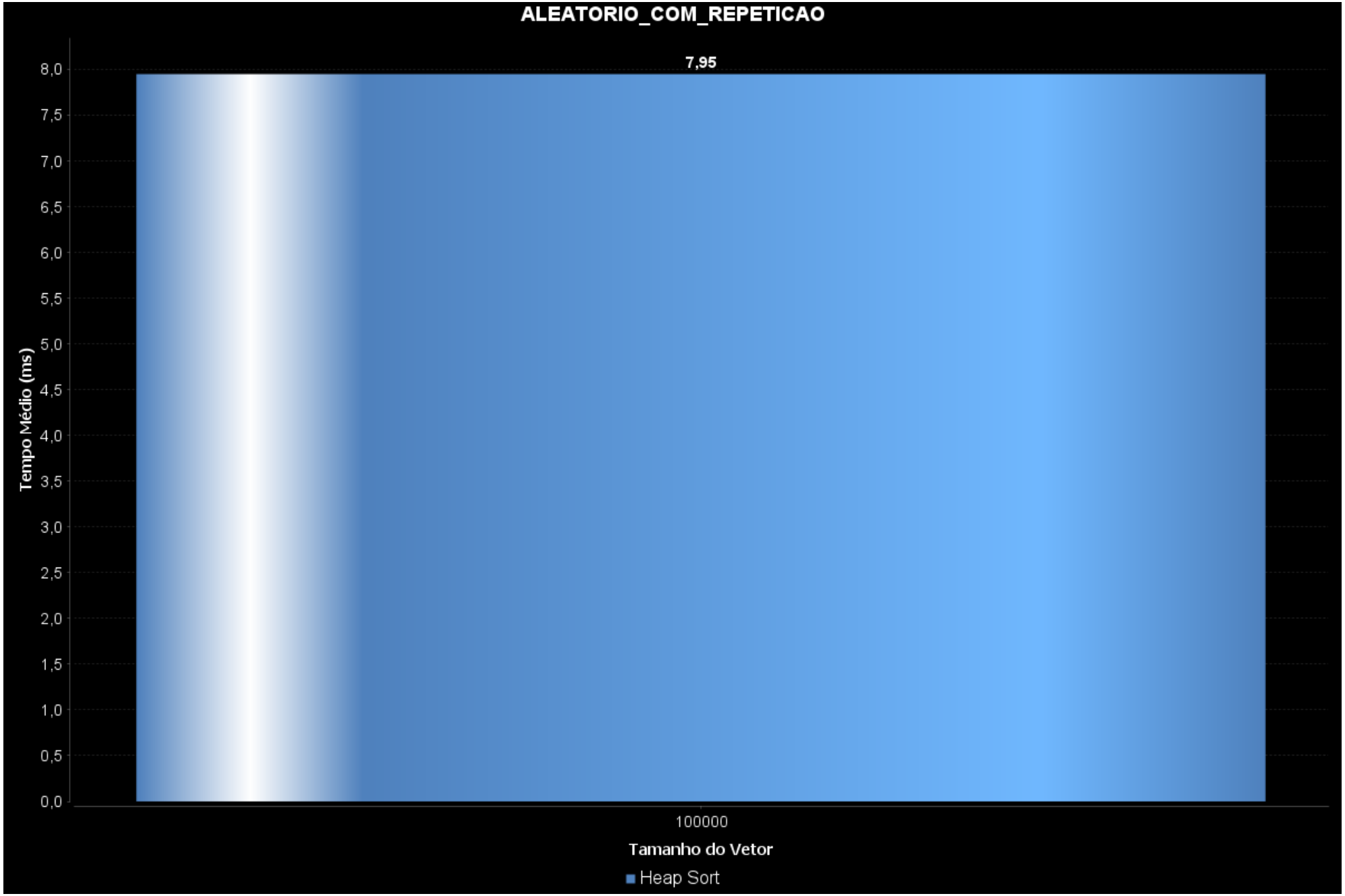
Relatório de Desempenho: CRESCENTE_SEM_REPETICAO

Algoritmo	100000
Heap Sort	4,86 ms


CRESCENTE_SEM_REPETICAO
4,86
5,0
4,8
4,6
4,4
4,2
4,0
3,8
3,6
3,4
3,2
3,0
2,8
2,6
2,4
2,2
2,0
1,8
1,6
1,4
1,2
1,0
0,8
0,6
0,4
0,2
0,0
Tempo Médio (ms)
100000
Tamanho do Vetor
Heap Sort

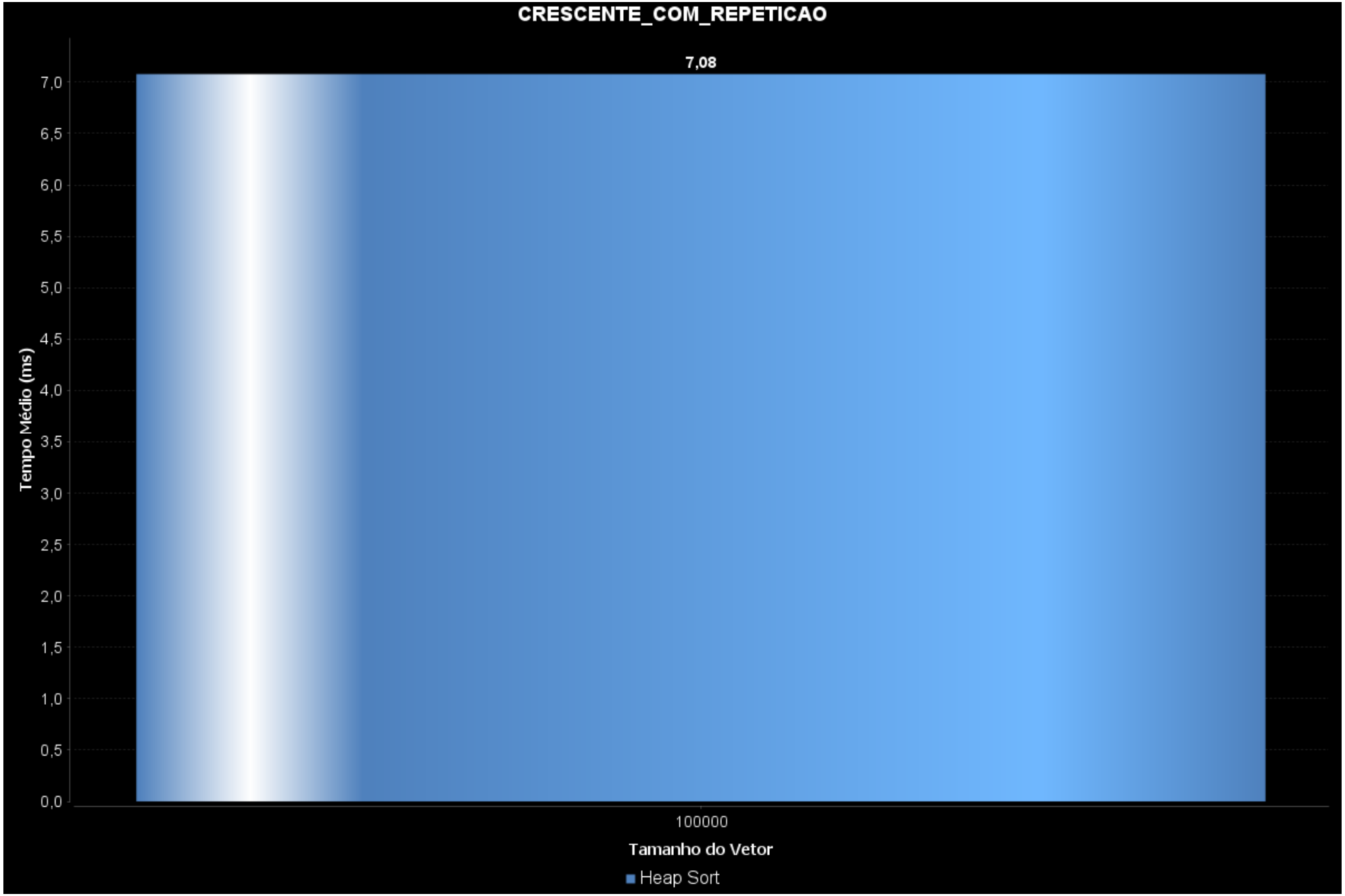
Relatório de Desempenho: ALEATORIO_COM_REPETICAO

Algoritmo	100000
Heap Sort	7,95 ms


ALEATORIO_COM_REPETICAO
7,95
8,0
7,5
7,0
6,5
6,0
5,5
5,0
4,5
4,0
3,5
3,0
2,5
2,0
1,5
1,0
0,5
0,0
Tempo Médio (ms)
100000
Tamanho do Vetor
Heap Sort

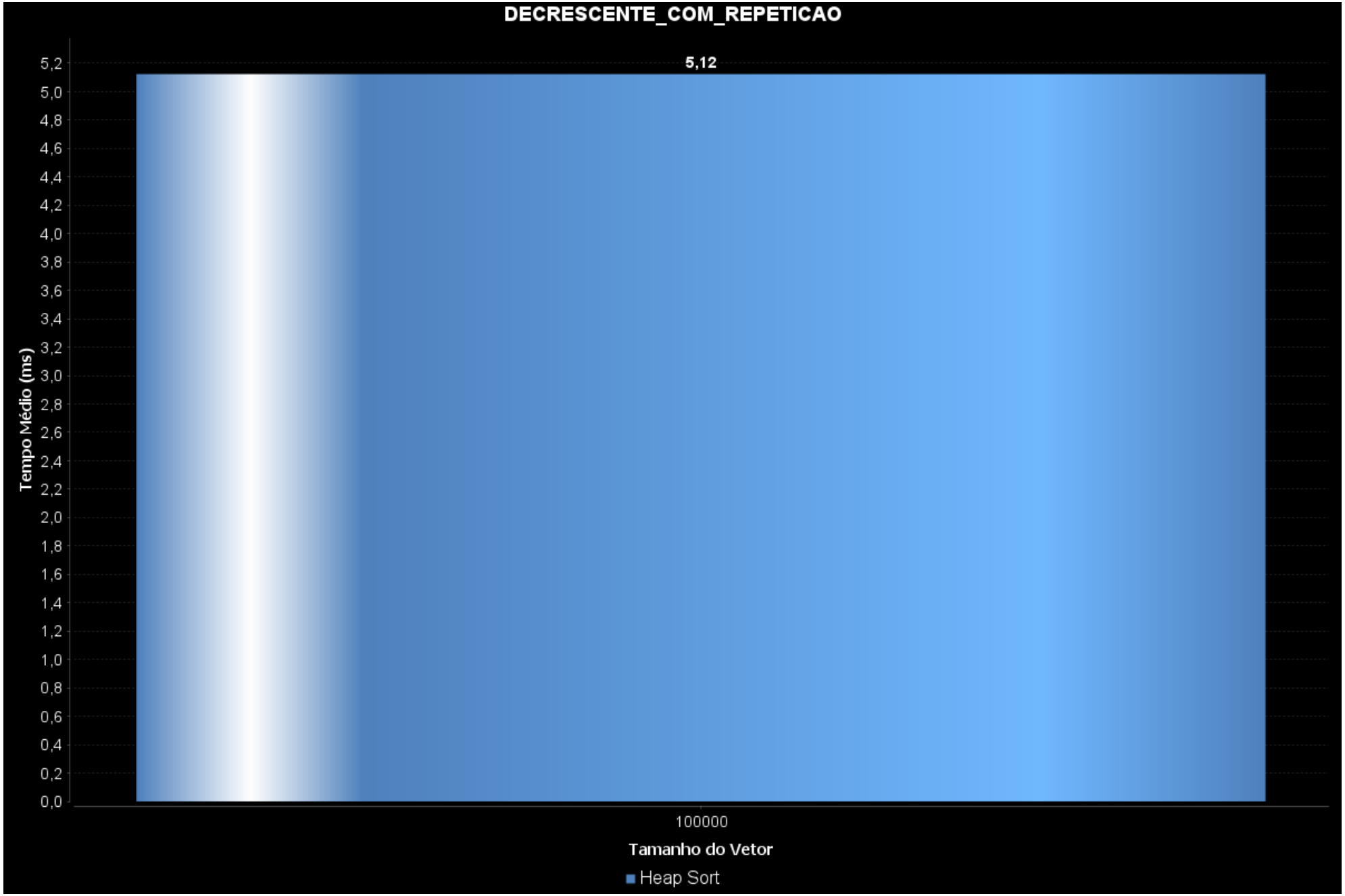
Relatório de Desempenho: CRESCENTE_COM_REPETICAO

Algoritmo	100000
Heap Sort	7,08 ms


CRESCENTE_COM_REPETICAO
7,08
7,0
6,5
6,0
5,5
5,0
4,5
4,0
3,5
3,0
2,5
2,0
1,5
1,0
0,5
0,0
Tempo Médio (ms)
100000
Tamanho do Vetor
Heap Sort

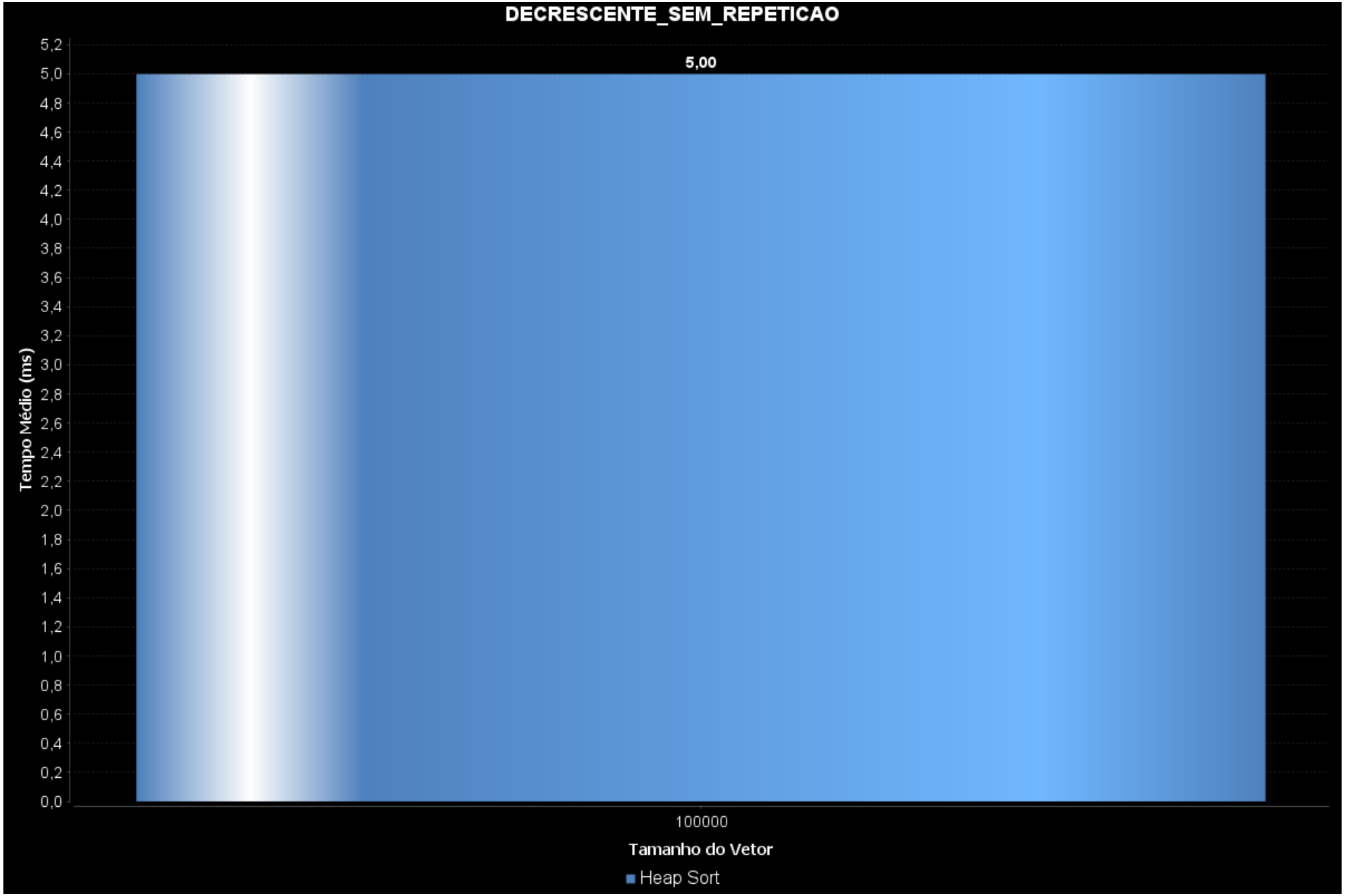
Relatório de Desempenho: DECRESCENTE_COM_REPETICAO

Algoritmo	100000
Heap Sort	5,12 ms


DECRESCENTE_COM_REPETICAO
5,12
5,2
5,0
4,8
4,6
4,4
4,2
4,0
3,8
3,6
3,4
3,2
3,0
2,8
2,6
2,4
2,2
2,0
1,8
1,6
1,4
1,2
1,0
0,8
0,6
0,4
0,2
0,0
Tempo Médio (ms)
100000
Tamanho do Vetor
Heap Sort

Relatório de Desempenho: DECRESCENTE_SEM_REPETICAO

Algoritmo	100000
Heap Sort	5,00 ms


DECRESCENTE_SEM_REPETICAO
5,00
5,2
5,0
4,8
4,6
4,4
4,2
4,0
3,8
3,6
3,4
3,2
3,0
2,8
2,6
2,4
2,2
2,0
1,8
1,6
1,4
1,2
1,0
0,8
0,6
0,4
0,2
0,0
Tempo Médio (ms)
100000
Tamanho do Vetor
Heap Sort