ED - Lista 1

Gabriel Sansigolo

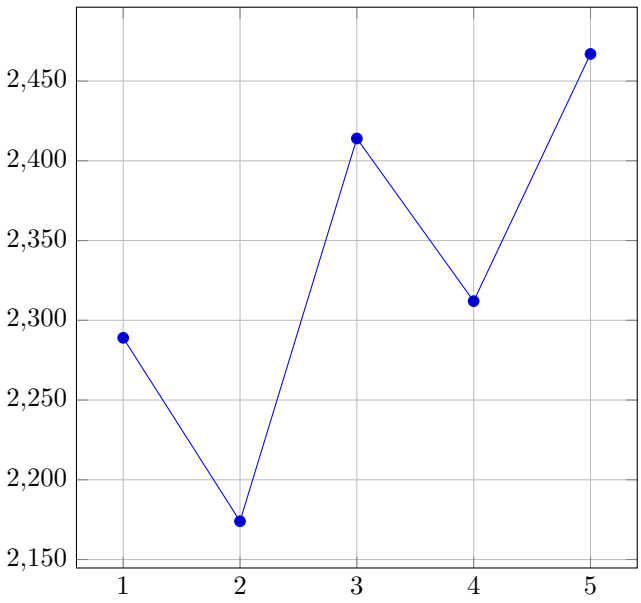
May 2018

Exercício 7

R: Após executar o programa 5 vezes e usando um vetor de tamanho 10000, foi possivel obter dados suficientes para plotar os seguintes gráficos.

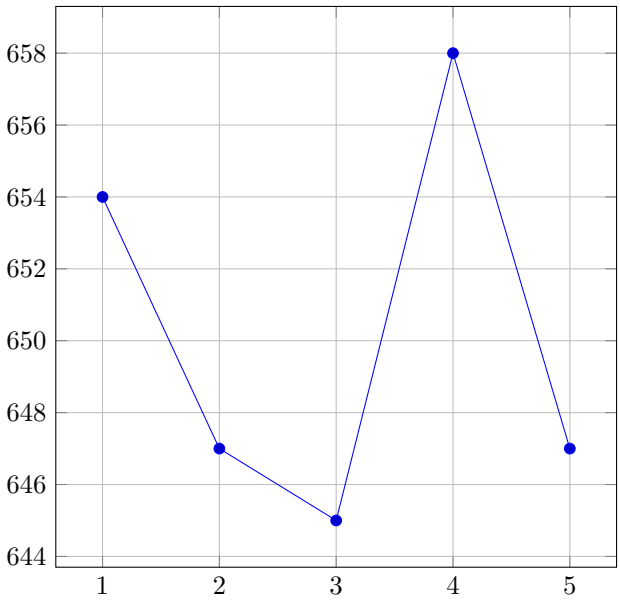
Foi plotado o tempo de execução dos algoritmos de ordenação em milisegundos. Primeiro será plotado os algoritmos usando vetores aleatórios e depois vetores ordenados.

selectSort() - Vetor Aleatório



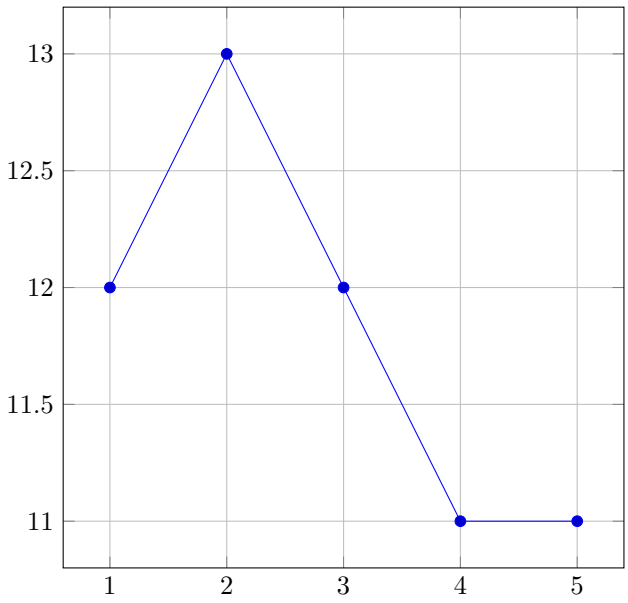
O selectSort() possui o pior tempo. **Média:** 2331.2 milisegundos

insertSort() - Vetor Aleatório



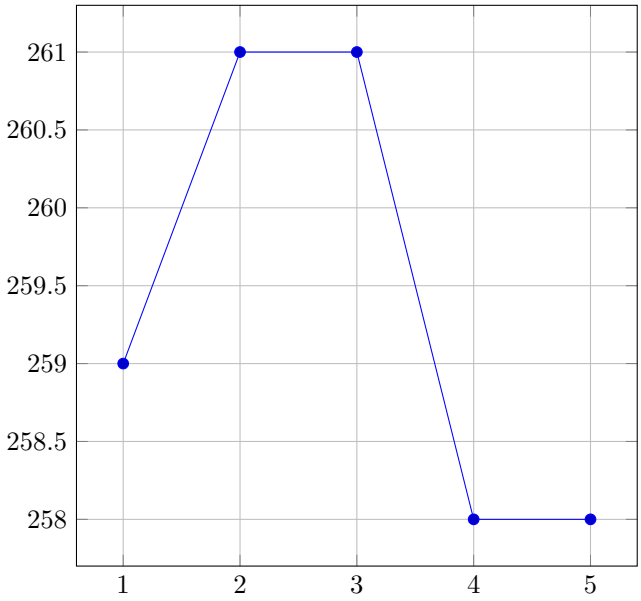
O insertSort() tem o segundo pior tempo. **Média:** 652 milisegundos

heapSort() - Vetor Aleatório



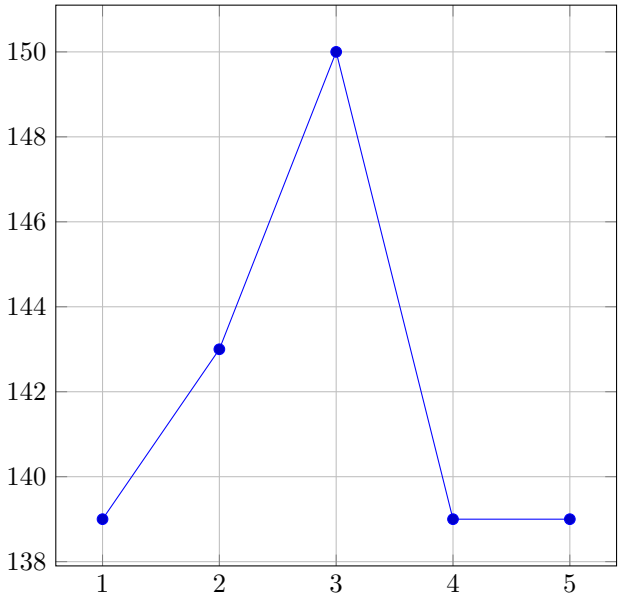
O heapSort() foi o algoritmo que obteve o melhor tempo. **Média:** 11.8 milis.

mergetSort() - Vetor Aleatório



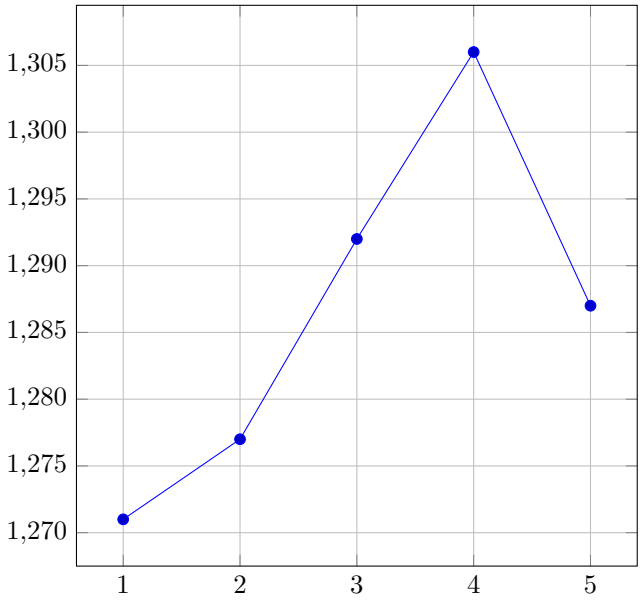
O mergeSort() foi o terceiro algoritmo com melhor tempo. **Média:** 259.4 milis

quickSort() - Vetor Aleatório



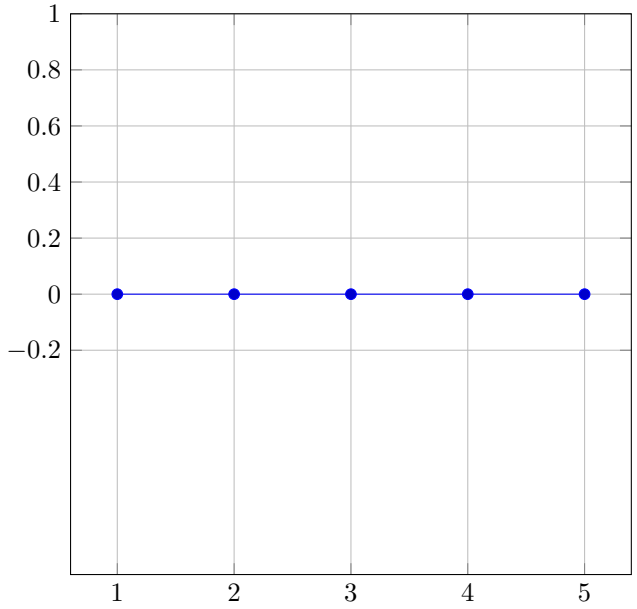
O quickSort() é o algoritmo com o segundo melhor tempo. **Média:** 142 milis

selectSort() - Vetor Ordenado



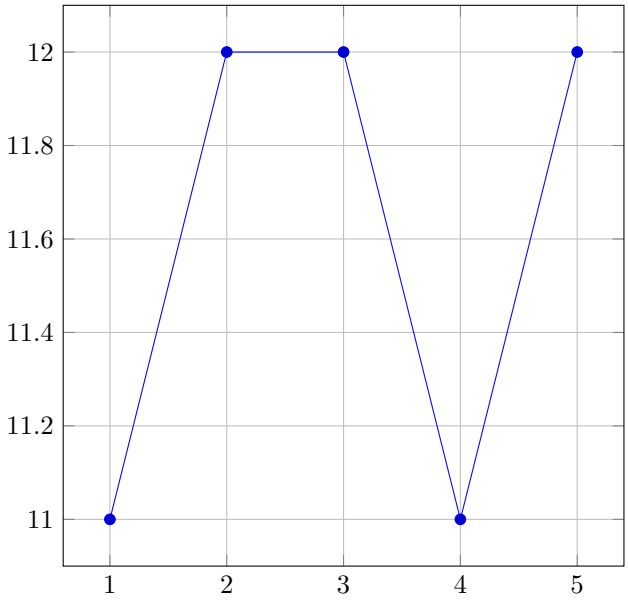
O selectSort() ordenado tem metade do tempo do aleatorio. **Média:** 1286.6

insertSort() - Vetor Ordenado



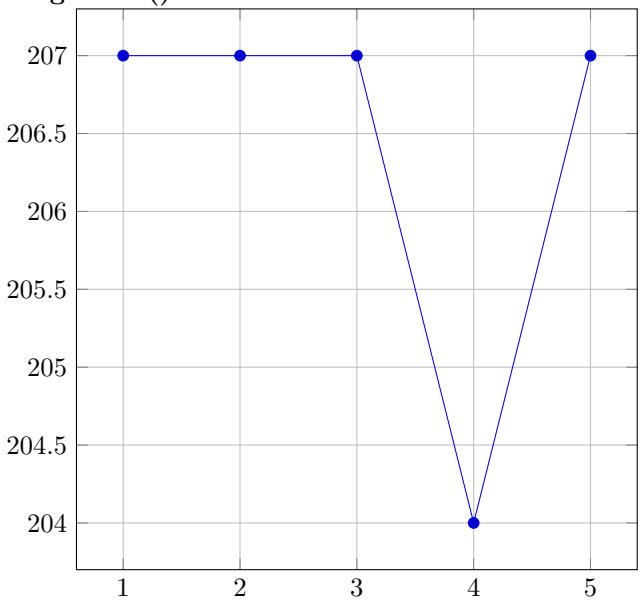
O insertSort() ordenado tem tempo 0. **Média:** 0 milisegundos

heapSort() - Vetor Ordenado



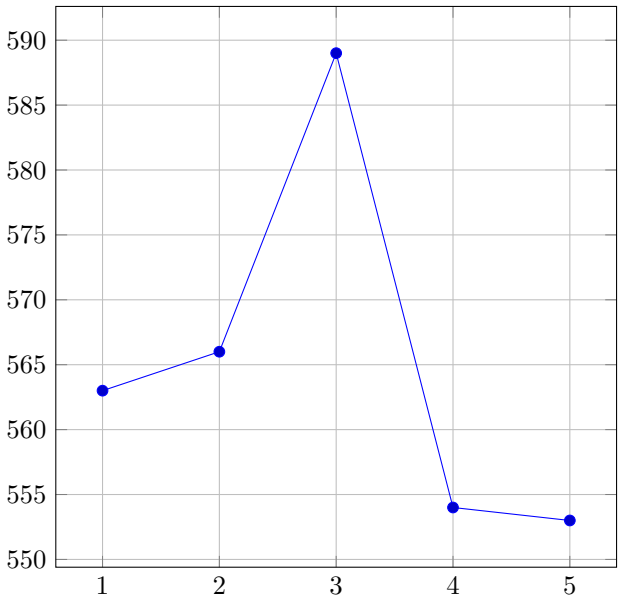
O heapSort() ordenado é semelhante ao aleatorio. **Média:** 11.6 milisegundos

mergetSort() - Vetor Ordenado



O mergetSort() possui quase o mesmo tempo do aleatorio. **Média:** 206.4 milis

quickSort() - Vetor Ordenado



O quickSort() é quase 5x maior que o aleatorio. **Média:** 565 milisegundos