C)

Comparações

	10	**100**	**1000**	**10000**
BubbleSort	4,5 x 10	$4{,}950 \times 10^{3}$	$4{,}999 \times 10^{5}$	$4{,}999 \times 10^{7}$
InsertionSort	2,6 x 10	$2{,}501 \times 10^{3}$	$2{,}546 \times 10^{5}$	$2{,}511 \times 10^{7}$
SelectionSort	5,4 x 10	$5{,}04 \times 10^{3}$	$5{,}004 \times 10^{5}$	5×10^{7}

Tempo (segundos)

	10	**100**	**1000**	**10000**
BubbleSort	5×10^{-6}	$7{,}8 \times 10^{-5}$	$5{,}197 \times 10^{-3}$	$6{,}198 \times 10^{-1}$
InsertionSort	4×10^{-6}	$3{,}2 \times 10^{-5}$	$1{,}869 \times 10^{-3}$	$1{,}801 \times 10^{-1}$
SelectionSort	$5{,}6 \times 10^{-5}$	$2{,}7 \times 10^{-5}$	$1{,}632 \times 10^{-3}$	$1{,}526 \times 10^{-1}$

D)

i) Algum algoritmo executou consideravelmente menos comparações considerando vetores com mais que 10 elementos?

Sim, o Insertion Sort.

ii) O algoritmo que executou menos comparações foi o que precisou de menos tempo para executar a ordenação?

Foi até um vetor de 100 elementos, após isso não.

iii) O que tem de interessante nos dois algoritmos com maior número de comparações em relação ao tempo de execução? Explique o resultado.

Existe uma grande diferença do tempo dos dois, porque no Selection Sort a troca do elemento no vetor acontece uma vez a cada n comparações, enquanto no Bubble Sort a troca é efetuada a cada comparação verdadeira no if.