Universidade Tecnológica Federal do Paraná

Disciplina: Algoritmos e Estrutura de Dados 2
Aluno: Jean Carlos Martins Miguel R.A: 1640593
Professor: Dr. Juliano Henrique Foleis

1 Ordenação – Merge Sort e Quick Sort

1.1 Lista de exercícios 2

- a)

(Bubble) Tempo de execução(s)	Quantidade de Elementos
0.012923	1000
0.382617	10000
43.691433	100000
1138.470703	500000

(Insertion) Tempo de execução(s)	Quantidade de Elementos
003704	1000
0.199583	10000
17.918020	100000
1114.683594	500000

(Selection) Tempo de execução(s)	Quantidade de Elementos
0.006515	1000
0.197635	10000
15.873591	100000
358.621521	500000

(MergeSort) Tempo de execução(s)	Quantidade de Elementos
0.001223	1000
0.009444	10000
0.043620	100000
0.153209	500000

(QuickSort) Tempo de execução(s)	Quantidade de Elementos
0.000728	1000
0.006260	10000
0.047845	100000
0.117313	500000

  - b) Execute os mesmos testes que no item anterior, mas dessa vez, use vetores já ordenados como entrada. Anote apenas o tempo de execução de cada algoritmo em uma tabela como a apresentada acima.

(Bubble)	Tempo de execução(s)	Quantidade de Elementos
	0.006442	1000
	0.151689	10000
	12.366760	100000
	320.630646	500000

(Insertion)	Tempo de execução(s)	Quantidade de Elementos
	0.000036	1000
	0.000383	10000
	0.000716	100000
	0.003221	500000

(Selection)	Tempo de execução(s)	Quantidade de Elementos
	0.007761	1000
	0.178391	10000
	15.833282	100000
	396.453278	500000

(MergeSort)	Tempo de execução(s)	Quantidade de Elementos
	0.000546	1000
	0.007081	10000
	0.015408	100000
	0.078132	500000

(QuickSort)	Tempo de execução(s)	Quantidade de Elementos
	0.015148	1000
	0.332439	10000
	31.068428	100000
	Falha de segmentação	500000

c) De acordo com os resultados obtidos nos itens **a** e **b** , responda as perguntas a seguir:

* Qual algoritmo você usaria em um vetor que está praticamente ordenado, ou seja, tem apenas alguns elementos fora do lugar? Por quê?
  R:Utilizaria o mergeSort pois a complexidade desse algortimo é (n log n) ou seja a quantidade de operações será proporcional ao tamanho do vetor.

* Qual algoritmo você usaria para ordenar um vetor que está ordenado em ordem decrescente? Por quê? (vamos fazer de conta que essa é a melhor forma de ordenar um vetor ordenado em ordem decrescente)
  R: Utilizaria o merge sort pois a complexidade no pior caso ainda seria (n log n)

* Qual algoritmo você usaria em um vetor que você não faz idéia sobre a ordem dos elementos? Porquê?
  R: mergeSort, devido a complexidade, pois possa ser que venha um vetor que já esteja ordenado e se eu escolher o quicksort a complexidade será de n^2 com um desempenho tão baixo quanto o do insertion sort, já o mergeSort em qualquer caso a complexidade será a mesma.

* Você usaria os algoritmos ineficientes em alguma situação? Se sim, qual deles você usaria? Em que situação?
  R:Sim, utilizaria o selection sort, em casos em que a quantidade de elementos fosse razoavelmente pequena, pois dentre os 3 algortimos ineficientes o selection é o melhor.