Instituto de Computação – IComp **Universidade Federal do Amazonas**
Aluno: Ruan Gabriel Gato Barros Matrícula: 21553690
Professor: Raimundo Barreto

Questionário acerca dos métodos de ordenação

Perguntas

1) Considerando somente a métrica ‘‘tempo de execução’’ diga qual foi seu o melhor e o pior caso observado para cada método de ordenação?

O melhor caso de cada algoritmo correspondeu às espectativas : o melhor caso foi o de menos instância (1k) e o de pior caso o de maior instância (100k).

Método	Pior Caso	Melhor Caso
Bubble Sort	Instância de 100k	Instância de 1k
Selection Sort	Instância de 100k	Instância de 1k
Insertion Sort	Instância de 100k	Instância de 1k
Heap Sort	Instância de 100k	Instância de 1k
Merge Sort	Instância de 100k	Instância de 1k
Quick Sort	Instância de 100k	Instância de 1k

2) Considerando somente a métrica ‘‘quantidade de comparações’’ diga qual função de n melhor descreve o desempenho de cada método de ordenação? 2 2 2 2 2 Função aqui pode ser n, n , log 2 n, n.log 2 n, n .log 2 n, n - n.log 2 n, n - (n.log 2 n) , etc. Vamos supor que os dados das quantidade de comparações do método da bolha para cada valor de n sejam: 959.639, 24.540.091, 98.711.127, 396.084.194, 2.487.915.240 e 2 9.951.660.960. Quando você compara com, por exemplo, n , dá os seguintes valores: 1.000.000, 25.000.000, 100.000.000, 400.000.000, 2.500.000.000, 10.000.000.000. Note 2 que a quantidade de comparações se assemelha bastante com o valor de n . Repita o mesmo procedimento de busca para os outros métodos de ordenação considerando outras opções de funções. Sinta-se livre para propor a função que melhor se ajuste aos dados. Justifique através de dados numéricos o porquê da tua resposta.

Embora os algoritmos $O(nlogn)$ aparentam ter o comportamento logarítmo, seu crescimento na verdade é uma mescla entre logarítmo e linear, as funções encontradas com regressão mostraram uma aceitação.

A função encontrada para os algoritmos teoricamente $O(nlogn)$ não fazem parte do conjunto dos polinômios, são maiores que os de ordem $O(n)$ e menores que os de ordem $O(n^2)$, desclassificando-os como ordem linear e quadrática, sendo algo entre eles.

• O bubble sort se mostrou bastante pareável com a magnitude de um algoritmo quadrado :

Bubble Sort	1000	5000	10000	20000	50000	100000
Dados Correntes	980618	24579083	98964103	397008149	2489020219	9967140328
n	1000	5000	10000	20000	50000	100000
n²	1000000	25000000	100000000	400000000	2500000000	10000000000
log(n)	9,965784285	12,28771238	13,28771238	14,28771238	15,60964047	16,60964047
nlog(n)	9965,784285	61438,5619	132877,1238	285754,2476	780482,0237	1660964,047
n²log(n)	9965784,285	307192809,5	1328771238	5715084952	39024101186	1,66096E+11
n² - nlog(n)	999990,0342	24999987,71	99999986,71	399999985,7	2499999984	9999999983

Figura 1: Algoritmo Bubble Sort

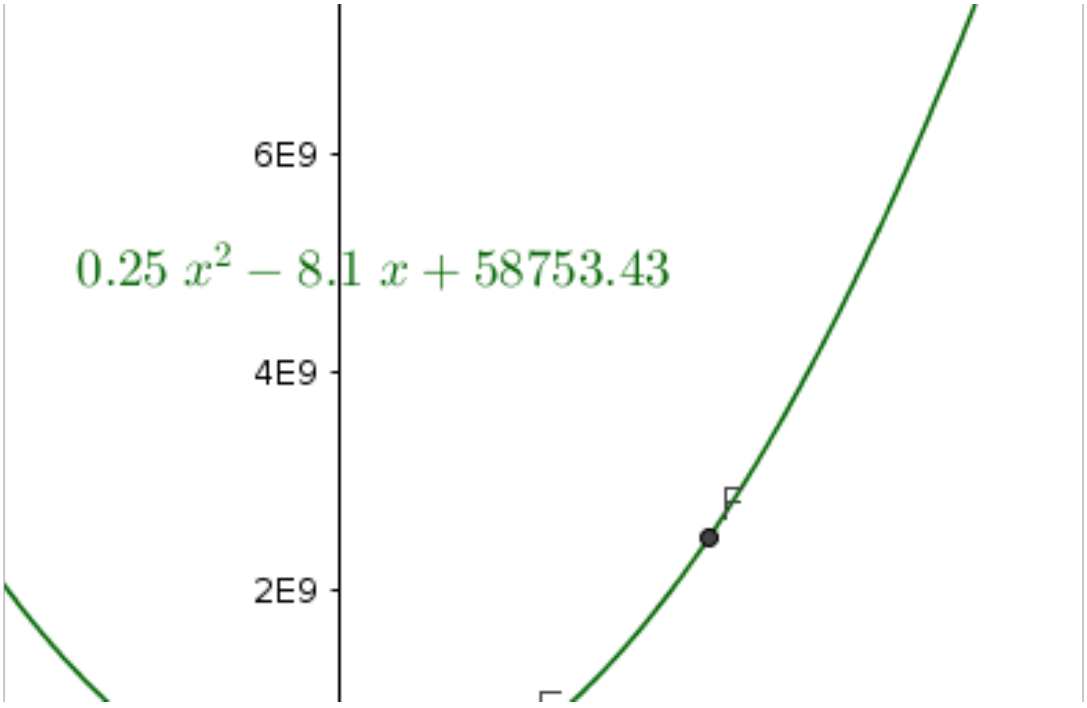

Figura 2: Algoritmo Bubble Sort - Comportamento parabólico no gráfico

• O selection sort também se mostrou semelhante com a magnitude de um algoritmo quadrado, porém aparenta ter custo mais amortizado:

Selection Sort	1000	5000	10000	20000	50000	100000
Dados Correntes	500500	12502500	50005000	200010000	1250025000	5000050000
n	1000	5000	10000	20000	50000	100000
n²	1000000	25000000	100000000	400000000	2500000000	10000000000
log(n)	9,965784285	12,28771238	13,28771238	14,28771238	15,60964047	16,60964047
nlog(n)	9965,784285	61438,5619	132877,1238	285754,2476	780482,0237	1660964,047
n²log(n)	9965784,285	307192809,5	1328771238	5715084952	39024101186	1,66096E+11
n² - nlog(n)	999990,0342	24999987,71	99999986,71	399999985,7	2499999984	9999999983

Figura 3: Algoritmo Selection Sort

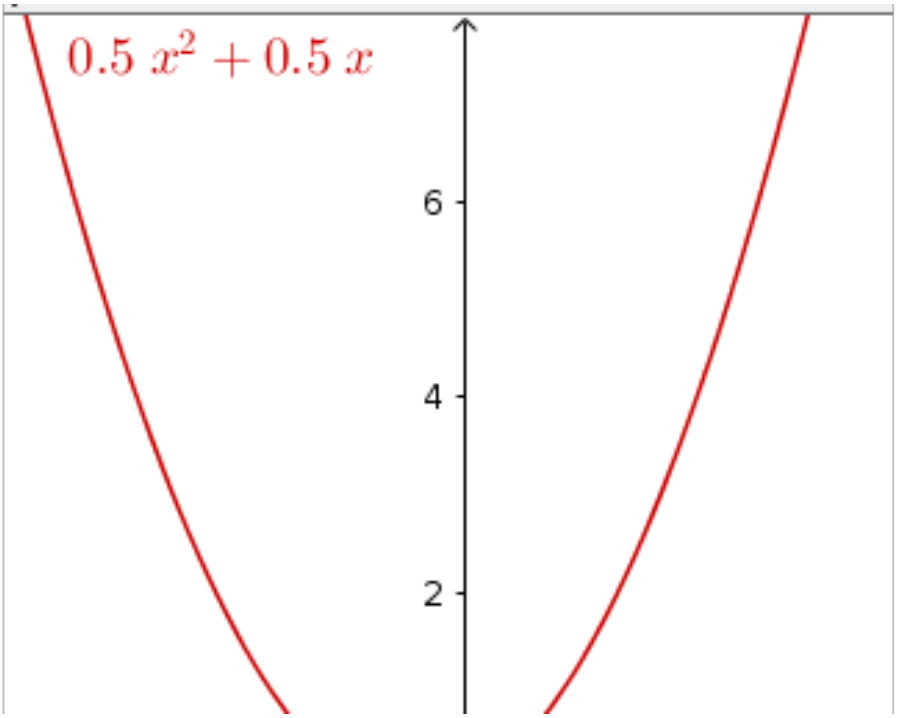

Figura 4: Algoritmo Selection Sort - Comportamento parabólico no gráfico

• O insertion sort, analogamente, se mostrou parecido com a magnitude de um algoritmo quadrado, mas com o porém de ser o menor dos algoritmos quadráticos testados :

Insertion Sort	1000	5000	10000	20000	50000	100000
Dados Correntes	251376	6261036	25010716	99862715	624211764	2497627894
n	1000	5000	10000	20000	50000	100000
n²	1000000	25000000	100000000	400000000	2500000000	10000000000
log(n)	9,965784285	12,28771238	13,28771238	14,28771238	15,60964047	16,60964047
nlog(n)	9965,784285	61438,5619	132877,1238	285754,2476	780482,0237	1660964,047
n²log(n)	9965784,285	307192809,5	1328771238	5715084952	39024101186	1,66096E+11
n² - nlog(n)	999990,0342	24999987,71	99999986,71	399999985,7	2499999984	9999999983

Figura 5: Algoritmo Insertion Sort

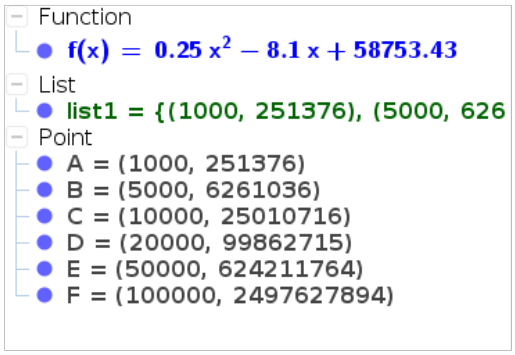

Figura 6: Algoritmo Insertion Sort - Comportamento parabólico no gráfico

• O Heapsort, mantendo a característica de um algoritmo de tempo igualitário em seus casos, não possui amortizações tão evidentes :

Heap Sort	1000	5000	10000	20000	50000	100000
Dados Correntes	12156	78099	171169	372334	1030647	2211224
n	1000	5000	10000	20000	50000	100000
n²	1000000	25000000	100000000	400000000	2500000000	10000000000
log(n)	9,965784285	12,28771238	13,28771238	14,28771238	15,60964047	16,60964047
nlog(n)	9965,784285	61438,5619	132877,1238	285754,2476	780482,0237	1660964,047
n²log(n)	9965784,285	307192809,5	1328771238	5715084952	39024101186	1,66096E+11
n² - nlog(n)	999990,0342	24999987,71	99999986,71	399999985,7	2499999984	9999999983

Figura 7: Algoritmo Heap Sort

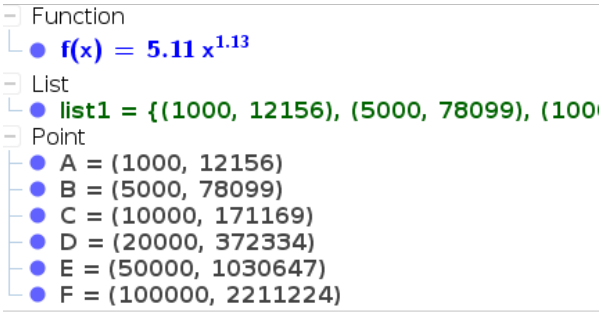

Figura 8: Algoritmo Heap Sort - Comportamento logarítmo no gráfico

• O Merge Sort com certeza é o mais elegante dos algoritmos logarítmicos lineares possui o comportamento bastante parecido com os casos de teste :

Merge Sort	1000	5000	10000	20000	50000	100000
Dados Correntes	8714	55223	120468	260922	718145	2211224
n	1000	5000	10000	20000	50000	100000
n²	1000000	25000000	100000000	400000000	2500000000	10000000000
log(n)	9,965784285	12,28771238	13,28771238	14,28771238	15,60964047	16,60964047
nlog(n)	9965,784285	61438,5619	132877,1238	285754,2476	780482,0237	1660964,047
n²log(n)	9965784,285	307192809,5	1328771238	5715084952	39024101186	1,66096E+11
n² - nlog(n)	999990,0342	24999987,71	99999986,71	399999985,7	2499999984	9999999983

Figura 9: Algoritmo Merge Sort

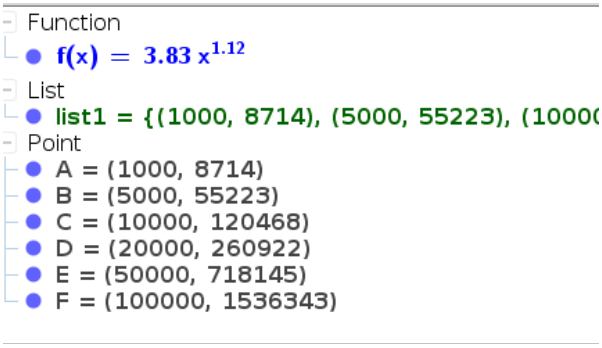

Figura 10: Algoritmo Merge Sort - Comportamento logarítmo no gráfico

• Durante os experimentos, o quicksort não permutou em uma sequência que causaria sua degeneração para o pior caso :

Quick Sort	1000	5000	10000	20000	50000	100000
Dados Correntes	7944	47846	104561	233384	634780	1361043
n	1000	5000	10000	20000	50000	100000
n^2	1000000	25000000	100000000	400000000	2500000000	10000000000
log(n)	9,965784285	12,28771238	13,28771238	14,28771238	15,60964047	16,60964047
nlog(n)	9965,784285	61438,5619	132877,1238	285754,2476	780482,0237	1660964,047
n^2log(n)	9965784,285	307192809,5	1328771238	5715084952	39024101186	1,66096E+11
n^2 - nlog(n)	999990,0342	24999987,71	99999986,71	399999985,7	2499999984	9999999983

Figura 11: Algoritmo Quick Sort

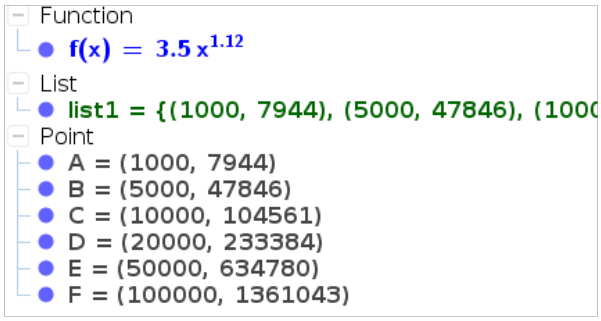

Figura 12: Algoritmo Quick Sort - Comportamento logarítmo no gráfico

3) `Considerando as métricas ‘‘quantidade de trocas’’ e ‘‘quantidade de comparações’’ faça uma relação entre essas duas métricas e diga uma função que represente tal relação.`

Observando o comportamento da razão entre o número de trocas e de comparações, consegui amortizar as funções em funções lineares.

		1k	5k	10k	20k	50k	100k
Relação	Bubble Sort	3,90100	3,92572	3,95687	3,97554	3,98746	3,99064
	Selection Sort	503,11620	2506,11369	5006,70832	10006,10342	25006,30146	50005,40053
	Insertion Sort	0,99604	0,99920	0,99960	0,99980	0,99992	0,99996
	Heap Sort	1,33984	1,36779	1,37799	1,38702	1,39754	1,40389
	Merge Sort	0,43674	0,44673	0,45080	0,45420	0,45773	0,46028
	Quick Sort	3,06143	3,07247	3,11214	3,24917	3,26390	3,30737

Figura 13: Relação entre número de comparações enúmero de trocas

Método	Função de Relação
Bubble Sort	$f(x) = 4x$
Selection Sort	$INDETERMINADA$
Insertion Sort	$f(x) = x$
Heap Sort	$f(x) = (13/10)x$
Merge Sort	$f(x) = (2/5)x$
Quick Sort	$f(x) = 3x$