CURSO DE SISTEMAS DE INFORMAÇÃO – CIN, UFPE

Renato Vimieiro
Eduardo Santos de Moura(esm7)

RESUMO

Este trabalho visa analisar o comportamento de três algoritmos de ordenação, submetidos a diversas condições a fim de propiciar o conhecimento mais aprofundado dos mesmos.

1 INTRODUÇÃO

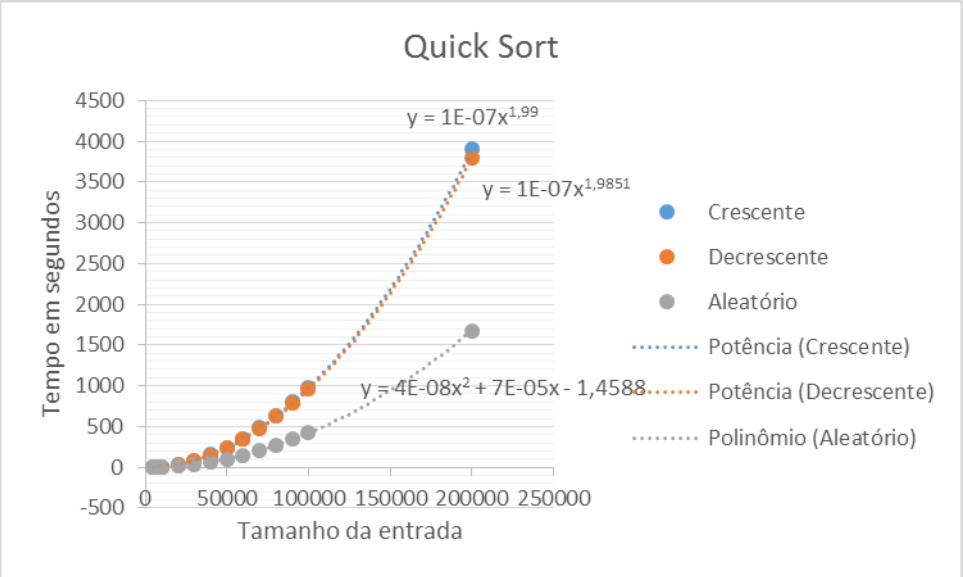
O objetivo deste documento é esclarecer sobre o tempo gasto pelos algoritmos QuickSort, Insertion Sort e Radix Sort, de acordo com diferentes entradas de vetores, organizados de maneiras diferentes.

Com este estudo, pode-se ter uma melhor noção do comportamento de cada algoritmo, explorando cada caso, observando os melhores e piores casos, para diferentes tamanhos de vetores e organizações.

Primeiramente é válido colocar que o tamanho dos vetores que compuseram a bateria de testes variaram de 5*10^3 a 6*10^5, totalizando 20 entradas distintas. É cabível ter em mente que alguns gráficos, nem todas as entradas foram consideradas, por efeito de melhor visualização gráfica ou não pelo fato de não ter sido possível obter o tempo pois o mesmo demandava quantidades absurdas para se chegar ao resultado.

Para a obtenção dos dados, entre 5*10^3 e 10^4, os tempos foram testados 10 vezes para cada, e no final tira-se uma média, para obter um resultado mais preciso, tendo base que a o número é pequeno e pequenas instabilidades na máquina podiam causar grandes erros na análise. Para vetores maiores entre 10^4 e 10^5, foram testados apenas 5 vezes, já que o tempo se tornaria gigantesco e como o número é maior que anteriormente, falhas da máquina serão proporcionalmente menores, não afetando tanto a análise. E para entradas maiores que 10^5, foram testadas apenas 3 vezes, dado o mesmo motivo anterior. Para os algoritmos com mais rápida execução, os 10 testes foram mantidos.

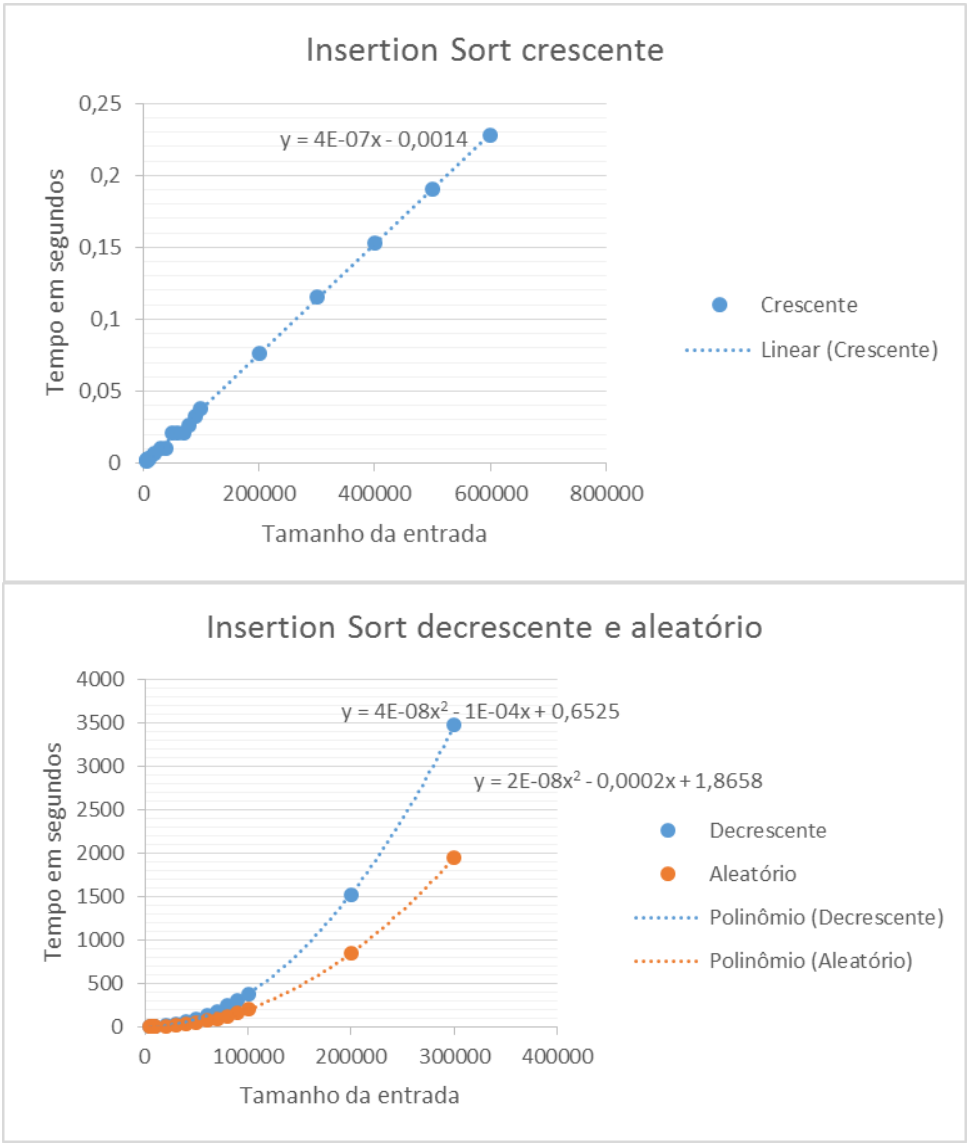
2 QUICKSORT



Como podemos observar acima com o gráfico o comportamento desse algoritmo é polinomial de grau 2. O uso mais efetivo do QuickSort é quando se sabe que os números estão organizados de maneira aleatória, pois caso o vetor já esteja parcialmente ordenado de maneira crescente ou decrescente, o seu tempo de execução se aproximará do pior caso, como demonstra o gráfico. Pode-se perceber também, que entre um vetor crescente e um decrescente, o primeiro é o pior caso, mas para tamanhos menores, a diferença é quase imperceptível, mas à medida que o mesmo aumenta, a diferença entre eles fica cada vez mais visível.

Esse algoritmo pode também ser implementado de maneira recursiva, e é extremamente eficiente para vetores pequenos, principalmente aleatórios, mas caso contrário, o limite da profundidade recursiva será facilmente ultrapassado.

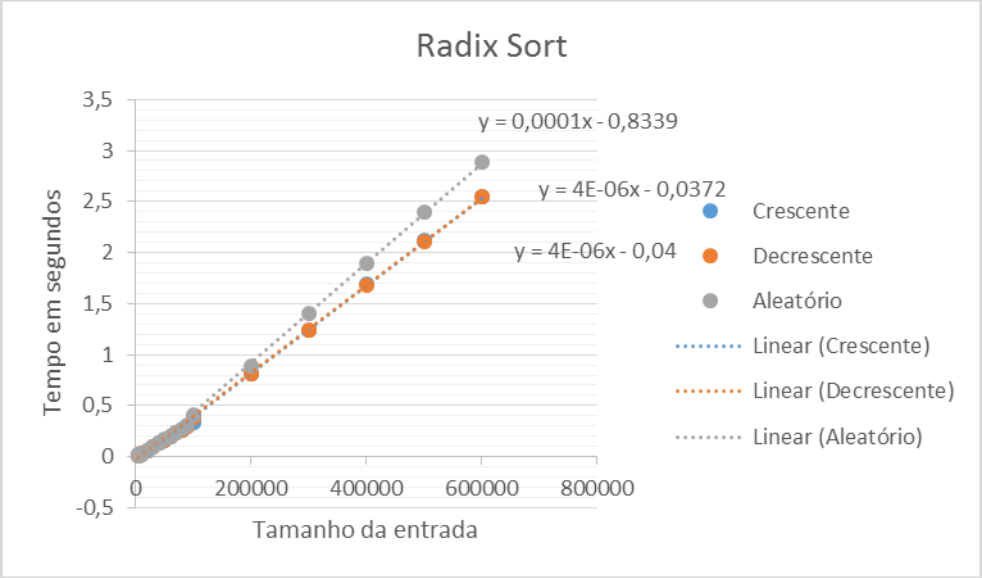
3 INSERTION SORT



O Insertion Sort é um algoritmo bem peculiar, ele tem como pior caso uma entrada ordenada de forma decrescente. A entrada de forma aleatória é um pouco melhor manipulada, mas ambos possuem comportamento polinomial de grau 2. Mas o algoritmo conta com uma extrema vantagem, o seu melhor caso, vetor ordenado de forma crescente com o comportamento linear.

O Insertion pode compor uma efetiva funcionalidade de checar se a lista já está ordenada de forma crescente e se não estiver fazer pequenas mudanças, com um tempo de execução muito pequeno.

4 RADIX SORT



Entre todos os algoritmos analisados, esse é o que melhor se adequa à situação para caso seja desconhecida a organização dos vetores, pois para todos os testes, o mesmo teve respostas ótimas. Todos os tempos apresentaram um comportamento linear, ou seja, para entradas muito grandes, o tempo do resultado não será absurdamente afetado. Seu pior caso é o vetor gerado aleatoriamente, mas mesmo assim, compensa usar esse algoritmo aos outros anteriores mesmo que se tenha conhecimento que a entrada é aleatória.

5 TABELA DE CONSULTA

Tamanho	I1	I2	I3	Q1	Q2	Q3	R1	R2	R3
5000	0,0016	0,454	0,2284	2,4894	2,4681	1,0667	0,0128	0,013	0,0129
6000	0,002	0,6507	0,3318	3,5735	3,5685	1,5904	0,0157	0,0154	0,0156
7000	0,0023	0,8954	0,4522	4,8939	4,9222	2,1251	0,0181	0,018	0,018
8000	0,0026	1,162	0,5907	6,523	6,465	2,7328	0,0207	0,0205	0,0206
9000	0,0033	1,4746	0,7465	8,2398	8,1347	3,4071	0,0234	0,0231	0,0232
10000	0,0033	1,817	0,9	10,3642	10,1375	4,2455	0,0259	0,0311	0,0311
20000	0,0066	14,7126	7,3996	40,0836	39,0092	16,3666	0,0635	0,0627	0,0641
30000	0,0103	32,966	16,5885	84,0061	83,2729	35,7604	0,0965	0,0951	0,0974
40000	0,0104	59,2033	30,9583	152,459	150,0982	64,2565	0,1292	0,1283	0,1321
50000	0,0208	93,2031	47,3695	242,284	237,7704	103,1525	0,1624	0,1623	0,1659
60000	0,0208	136,2601	71,677	352,8518	346,1438	148,9906	0,1958	0,1939	0,2029
70000	0,0208	185,3838	97,6099	483,502	474,948	204,6983	0,2291	0,2282	0,2383
80000	0,026	245,9466	127,7189	638,7746	629,6067	268,9028	0,2623	0,262	0,2762
90000	0,033	309,8301	165	811,9086	793,0232	348,0332	0,2963	0,2938	0,3079
100000	0,038	382,345	206,3421	972,8231	959,9123	425,1235	0,3309	0,3829	0,4088
200000	0,076	1523,0092	844,9992	3907,316	3792,864	1673,174	0,8136	0,8119	0,8913
300000	0,1153	3470,598	1949,454	Not possib	Not possib	3800,239	1,2451	1,2455	1,3982
400000	0,1533	Not possibl	3491,884	Not possib	Not possib	Not possib	1,6882	1,6858	1,899
500000	0,1906	Not possibl	Not possib	Not possib	Not possib	Not possib	2,115	2,1108	2,3973
600000	0,228	Not possibl	Not possib	Not possib	Not possib	Not possib	2,5488	2,5483	2,8903

6 CONSIDERAÇÕES FINAIS

Diante dessa análise, foi possível ter um esclarecimento sobre o tempo de execução de cada algoritmo dado o tamanho da entrada e a ordenação. Para vetores ordenados de maneira crescente, sem sombra de dúvidas o que melhor se encaixa é o Insertion Sort. Para uma ordenação decrescente o mais indicado é o Radix Sort, que responde bem a todas as entradas. Para ordem aleatória, pode ser usado também o Radix, mas caso a entrada não seja enorme, uma ótima opção é o QuickSort recursivo.

7 REFERÊNCIAS (EXEMPLOS)

https://docs.google.com/viewer?a=v&pid=sites&srcid=Y2luLnVmcGUuYnJ8aWY5NjktYWVkcy1ydmltaWVpcm98Z3g6NzMwYmQyOTk1MTQxMDYyNg

https://docs.google.com/viewer?a=v&pid=sites&srcid=Y2luLnVmcGUuYnJ8aWY5NjktYWVkcy1ydmltaWVpcm98Z3g6NzY1NzQyNDhhOTEwNWZkMg

https://docs.google.com/viewer?a=v&pid=sites&srcid=Y2luLnVmcGUuYnJ8aWY5NjkNjktYWV1ydmltaWVpcm98Z3g6MTQ5NjY5MGQ1Y2UxNDZmNg

https://www.youtube.com/watch?v=e5dKAK4Df04