DOCUMENTAÇÃO – TRABALHO II

Luiz Eduardo Pereira.

0021619.

29/07/2016.

Instituto Federal de Minas Gerais, Formiga, MG.

RESUMO:

Neste relatório foi feito uma comparação entre alguns algoritmos de ordenação. São eles: Bubble Sort, Insertion Sort, Quicksort e Quicksort otimizado usando Insertion Sort.

PALAVRAS-CHAVE: Bubble Sort; Insertion Sort; Quicksort;

INTRODUÇÃO:

Existe uma variedade de algoritmos de ordenação. Cada um possui uma metodologia diferente para ordenar, e para que seja possível a escolha de um, é necessário analisar alguns requisitos, como quantidade de números a ser ordenado, tamanho do registro, etc.

Para facilitar a execução deste programa, foi disponibilizado o uso de macros via comando "-D". Estão disponíveis os macros de tamanho, número de execuções e método a ser utilizado. Para executar o programa corretamente é necessário chamar o comando "-lm" para usar a função "math.h". Exemplo:

gcc –Wall ordenacao.c –DTAMANHO=(tamanho) –DEXECUCAO=(n. de execuções) –DMETODO=(*) –lm –o main

* *1 = Bubble Sort / 2 = Insertion Sort / 3 = Quicksort / 4 = Quicksort Turbinado.*

BUBBLE SORT:

O método da bolha consiste em percorrer o vetor diversas vezes, fazendo com que o maior elemento chegue ao topo. Sua complexidade é $O(n^2)$. O método da bolha é ineficiente para grandes quantidades de elementos a ser ordenados.

3 Execuções	500	1.000	10.000	100.000	300.000
Comparações	124.052	494.956	49.511.358	4.950.031.609	44.547.292.951
Desvio Padrão	159,58	1.923,49	30.171.69	824.706,03	5.648.086,98
Trocas	57.077	222.931	22.504.400	2.250.057.363	20.256.570.836
Desvio Padrão	847,24	956,74	177.932,79	5.078.008,26	26.225.994,05
Tempo	2,2218 ms	9,0629 ms	1,0361 s	105,2928 s	954,4254 s
Desvio Padrão	0,00	0,15	3,69	80,53	3.657,29

INSERTION SORT:

O método da inserção consiste em pegar um conjunto de elementos ordenados e inserir novos elementos em suas devidas posições. Sua complexidade é $O(n^2)$. Igualmente ao método da bolha, o método da inserção é ineficiente para grandes quantidades de elementos.

3 Execuções	500	1.000	10.000	100.000	300.000
Comparações	67.257	270.340	27.313.248	2751466404	24.729.732.288
Desvio Padrão	1.130,25	1.813,38	232.199,88	7711259.24	17.669.645,97
Trocas	67.257	270.340	27.313.248	2751466404	24.729.732.288
Desvio Padrão	1.130,25	1.813,38	232.199,88	7711259.24	17.669.645,97
Tempo	0,9277 ms	3,6607 ms	366,5322 ms	37,0618 s	341,4442 s
Desvio Padrão	0,01	0,02	3,11	219.00	664,49

QUICKSORT:

O Quicksort adota a estratégia da divisão e conquista. Incialmente, ele escolhe um elemento pivô. Com o pivô selecionado, ela particiona o conjunto em dois. De um lado ficam os elementos menores que o pivô, do outro, os maiores. Ele segue dividindo o conjunto até que todos os elementos estejam ordenados. Sua complexidade é $O(n*logn)$ no melhor caso e $O(n^2)$ no pior caso. Ele é muito eficiente para conjuntos com muitos elementos. Mas deve-se tentar evitar o pior caso, usando alguns meios alternativos, como escolher o pivô por mediana de 3, ou embaralhar os elementos do conjunto primeiro.

3 Execuções	500	1.000	10.000	100.000	300.000
Comparações	5.033	11.199	146.445	1.806.080	5.842.683
Desvio Padrão	76,23	154,62	2.125,69	11.428,32	24.299,87
Trocas	1.982	4.492	61.818	788.571	2.597.401
Desvio Padrão	30,14	34,48	354,26	2.982,58	12.649,37
Tempo	0,1286 ms	0,2681 ms	3,0743 ms	35,5982 ms	112,5757 ms
Desvio Padrão	0,00	0,01	0,02	0,25	1,14

QUICKSORT TURBINADO:

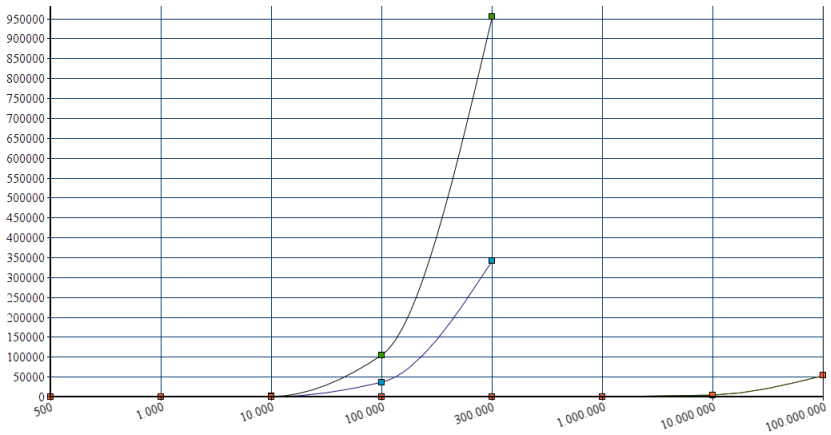
O Quicksort turbinado nada mais é que o Quicksort normal com o Insertion Sort. Quando o número de elementos é maior que 40, o Quicksort é usado. Mas quando é menor, o método da inserção termina a ordenação. Isso ocorre porque quando o número de elementos é pequeno, o método da inserção ordena mais rápido que o quicksort.

3 Execuções	500	1.000	10.000	100.000	300.000
Comparações	5.227	11.307	149.820	1.835.827	5.897.912
Desvio Padrão	52,32	209,78	43,91	21.616,55	42.300,47
Trocas	1.980	4.485	62.541	781.572	2.621.555
Desvio Padrão	9,67	13,89	141,84	2.876,89	17.027,18
Tempo	0,1313 ms	0,2684 ms	3,1468 ms	35,2719 ms	114,5366 ms
Desvio Padrão	0,01	0,01	0,03	0,43	1,32

COMPARAÇÕES:

Nas tabelas do Quicksort e Quicksort Turbinado podemos ver que o Quicksort Turbinado ficou para traz. Isso ocorreu, pois os vetores são gerados aleatoriamente, e o tamanho dos vetores não permite que o verdadeiro potencial do Quicksort Turbinado seja mostrado. A seguir está o gráfico de tempo de execução dos métodos:

Bubble, Insertion, Quicksort, Quicksort Turbinado.



CONCLUSÃO:

O método da bolha é um algoritmo muito fácil de ser programado. Só é recomendado quando não existe outro algoritmo em mãos ou quando a quantidade de elementos foi muito pequena.

O método da Inserção é mais recomendado que o método da bolha. Apesar de possuir a mesma complexidade, é capaz de executar mais rápido que o Bubble Sort. Ele deve ser usado quando há a necessidade de manter a ordem das chaves dos elementos, pois ele é estável.

O Quicksort é um algoritmo muito mais eficiente, mas não é estável. Seu código possui uma implementação mais complexa do que os outros métodos citados acima, mas é o mais recomendado em ocasiões em que existam muitos elementos no conjunto.

Teoricamente o Quicksort Turbinado devia ser muito mais rápido que o Quicksort normal. Mas como foi possível observar, em conjuntos com menos de um bilhão de elementos não haverá quase nenhuma diferença entre eles.

REFERÊNCIA:

Contador de Nanossegundos - http://stackoverflow.com/questions/361363/how-to-measure-time-in-milliseconds-using-ansi-c - (Acesso em 26/07/2016).

Gerador de Gráficos - http://www.onlinecharttool.com/graph?selected_graph=line