Universidade Federal Rural de Pernambuco

Departamento de Estatística e Informática

Coordenação de Graduação em Ciência da Computação

Exercicio de Algoritmos

Algoritmos de Ordenação

Aluno
Giuseppe Fiorentino Neto

Professor
Rodrigo Nonamor Pereira Mariano de Souza

Recife
junho–2017

1. Introdução

A ordenação é um problema comum e fundamental do cotidiano e é tratado muitas vezes como um problema simples e sem possuir uma devida atenção. Porém diferente da mente humana os computadores não conseguem organizar de forma tão intuitiva quanto esta. Assim surgiram os algoritmos que indicava o passo a passo para a realização da ordanação de numeros. Permutar os elementos é rearranja-los de tal modo que fique em ordem crescente(ou decrescente), isto é, de tal forma que tenhamos i<=i+1<=i+2<.=..<=i+n.

Atualmente existem inumeros algoritmos de ordenação e cada um possui sua importância. O que difere entre estes algoritmos é o seu desempenho que pode tornar o trabalho de ordenação em algo totalmente impossivel de se resolver ou algo tão simples e rapido.

Neste trabalho esta sendo analisado os principais algoritmos de ordenação, tais como inserção, seleção, heapsort e mergesort Além de que será feito analises tanto nas suas formas recursiva e interativa. E por fim feito uma comparação de desempenho dos algoritmos entre si e entra as formas de recursão e iteração.

O estudo esta organizado da seguinte forma:A Seção 2 descreve detalhadamente o método desenvolvido. A Seção 3 apresenta a metodologia experimental utilizada para avaliar o método proposto. A Seção 4 apresenta os resultados alcançados. Finalmente, a Seção 4 destaca as conclusões.

2. Trabalho Desenvolvido

Nesta seção fomularemos nosso estudo a respeito da ordenação. Especialmento, o problema de permutação dos elementos de um vetor de tal modo que o tenhamos em ordem crescente:

$$v[0] \leq v[1] \leq \ldots \leq v[n-1].$$

111	222	333	444	555	555	666	777	888	999	999

2.1 Insertion Sort

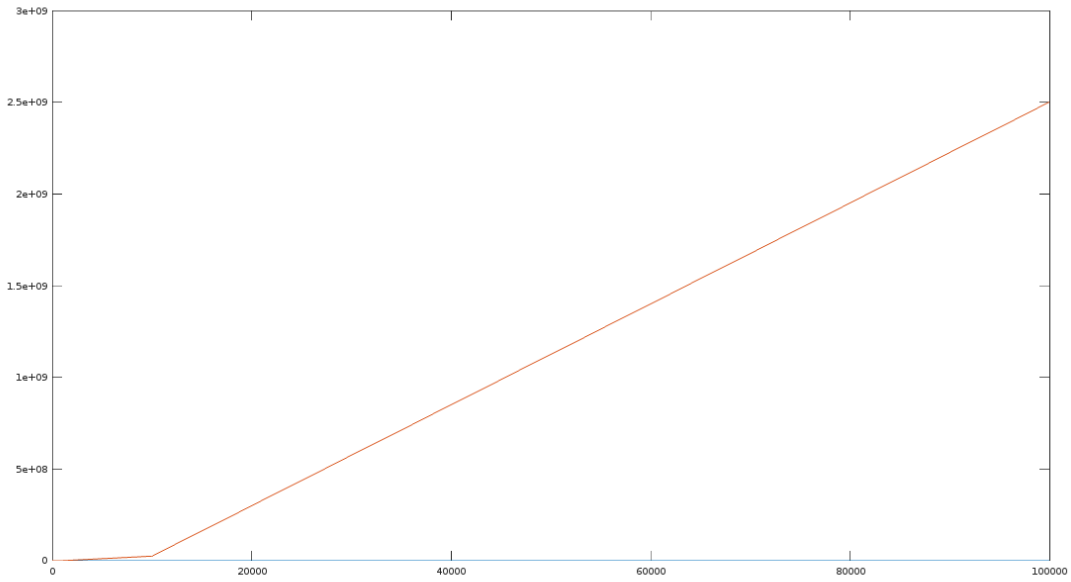
O Algoritmo insertion sort é um algoritmo que possui a complexidade de tempo, no pior caso, na ordem de n^2.

$$T(n) = O(n^2)$$

Sua versao interativa consome mais tempo e espaço que a versão recursiva. Tornando-se ainda mais custoso se o vetor estiver em ordem decrescente. Porém se o vetor estiver um pouco ordenado ele ira consumer menos. Assim no melhor caso o o desempenho do algoritmo é muito bom. Tendo o consumo de tempo proporcional a n.

$$T(n) = O(n)$$

#Iterativo	#Recursivo	#Diferenca	#Tamanho
28	15	13	10
23	15	8	10
2608	127	2481	100
253670	1023	252647	1000
25037608	11811	25025797	10000
2502032523	131071	1793065844	100000



O grafico que é mostrado acima é o grafico do algoritmo de ordenação insertion sort. A reta vermelhra representa o algoritmo iterativo e a azul recursivo. Mostrando assim que o algoritmo iterativo cresce muito mais rapido que o recursivo.

2.2 Selection Sort

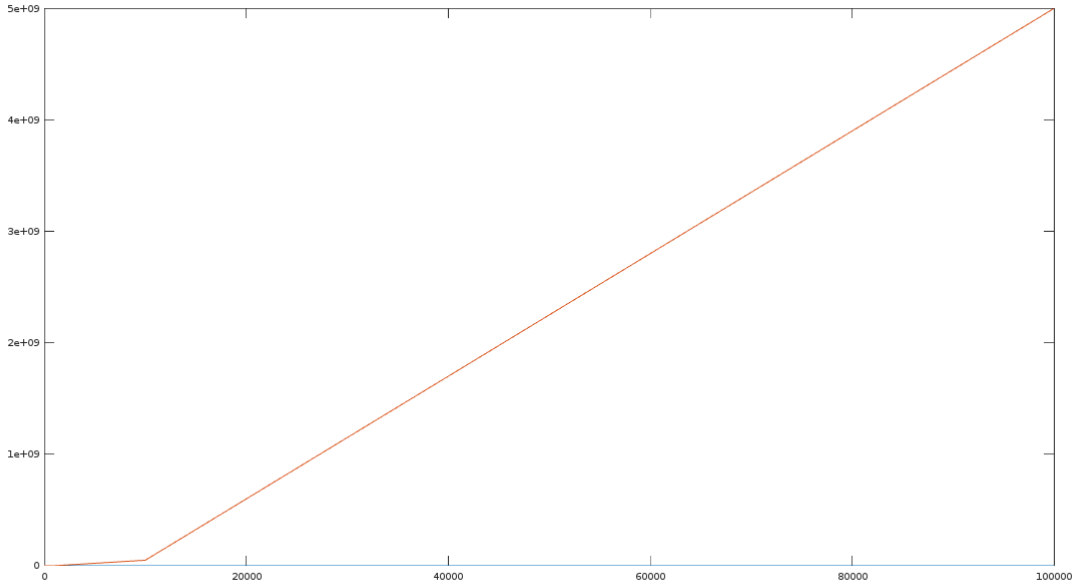
Assim como Algoritmo insertion sort o algoritmo de ordenação por seleção é um algoritmo que possui a complexidade de tempo, no pior caso, na ordem de n^2. Porém diferente deste que no melhor caso faz cerca de $n^2/2$ comparações entre elementos.

$$T(n) = O(n^2)$$

#Iterativo	#Recursivo	#Diferenca	#Tamanho

45	9	36	10
4950	99	4851	100
499500	999	498501	1000
49995000,	9999	49985001	10000
4999950000	99999	704882705	100000

Diferento do Insertion que com 100000 elementos em um vetor conseguiu rodar em um tempo rasoavelmente consideravel o algoritmo de seleção , devido ao fato de tando no melhor caso como no pior caso ser n², levou bastante tempo e teve que ser parado, pois ainda estava a executar.



O grafico que é mostrado acima é o grafico do algoritmo de ordenação selection sort. A reta vermelhra representa o algoritmo iterativo e a azul recursivo. Mostrando assim como o insertion sort a versão iterativo cresce muito mais rapido que o recursivo.

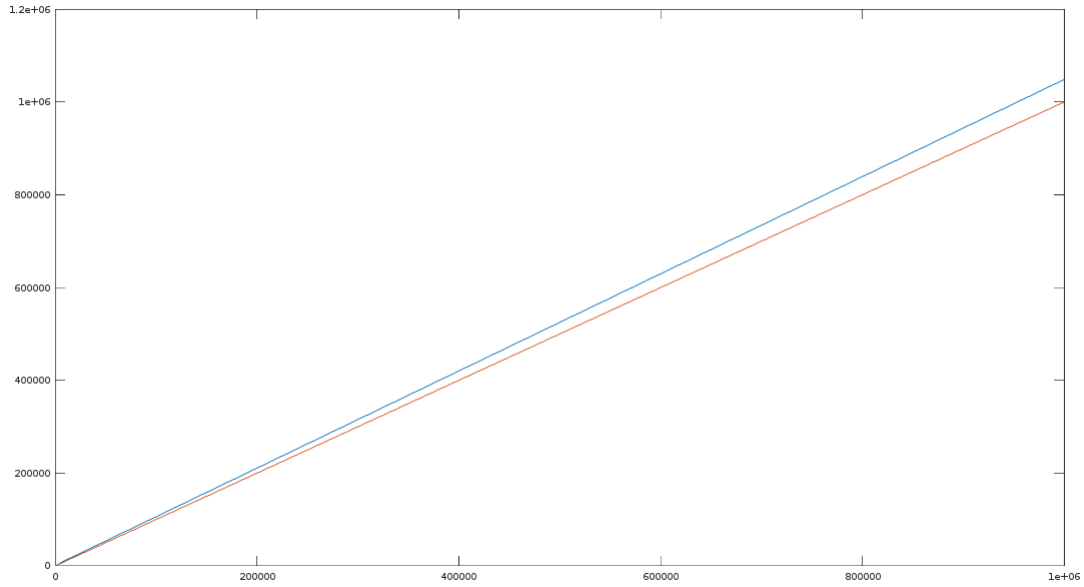
2.3 Merge Sort

O merge sorte é um dos algoritmos considerado rapido, porém éum algoritmo mais complexo que os anteriores. A a cada passo da recursão do mergesort o tamanho do vetor é reduzido à metade assim o numero de chamadas é logn. E a cada chamada, a função intercala executa 2n açoes. Sendo assim a complexidade do algortimot que utiliza a intercalação pra ordenar é:

$$T(n)=O(n);$$

#Iterativo	#Recursivo	#Diferenca	#Tamanho
9	11	10	10
99	127	28	100
9999	11807	1808	10000
999999	1048575	48576	1000000

O mergesort é um algoritmo rapido até mesmo para vetores de tamanhos consideravelmente grandes. Com até 1000000 elementos em um vetor ele não levou mais de 5s pra resolver o problema. Porém devido ao fato da limetação do hardware da maquina testada com 10000000 ele já deu Falha de segmentação.



O grafico que é mostrado acima é o grafico do algoritmo de ordenação merge sort. A reta vermelhra representa o algoritmo iterativo e a azul recursivo. Mostrando assim que a versão iterativa e recursiva não possui tantas diferenças quanto na maioria dos outros algoritmos que a versao recursiva normalmente era muito mais superior que a iterativa

2.4 Heap Sort

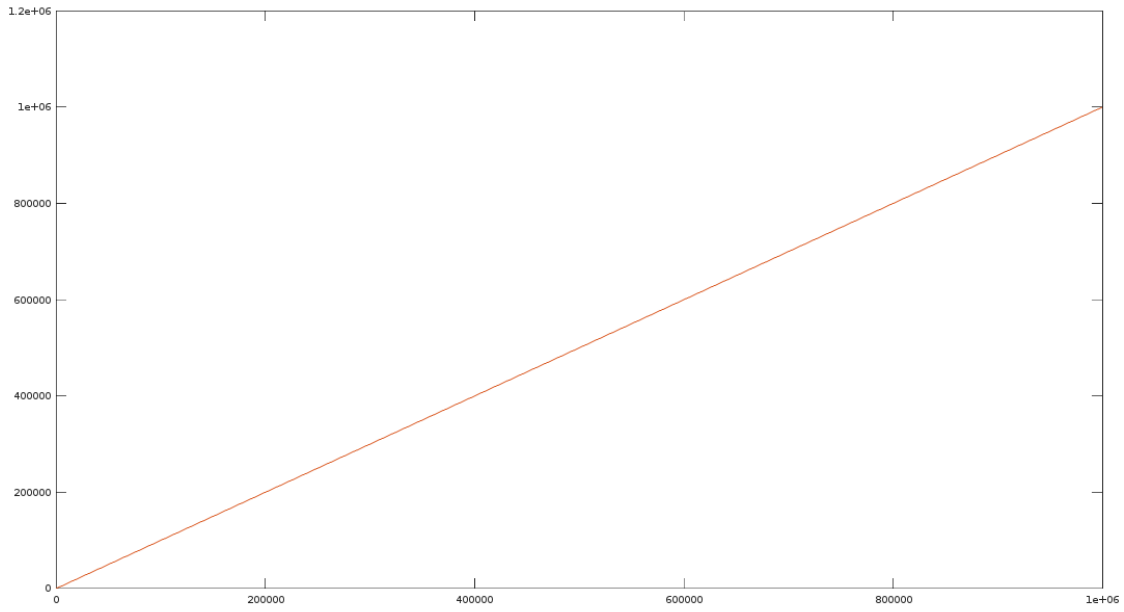
O Heap sorte é o segundo dos algoritmos considerado rapido e diferente dos outros mesmo no pior caso ele ainda continua sendo rapido. A base que torna o heapsort um algoritmo rapido é uma fila de prioridades muito eficiente. E a partir dessa fila de propiedades o heap se torna proporcional ao número de comparações entre elementos do vetor e daí temos um algoritmo eficiente e rapido mesmo no pior caso

$$T(n)=O(n);$$

#Iterativo	#Recursivo	#Diferenca	#Tamanho
7	10	3	9

99	100	1	100
998	999	1	1000
9998	9999	1	10000
100007	100008	1	100000
1000005	1000006	1	1000000

O Heapsort é o algoritmo rapido dos que foam aqui analisados. E mesmo com diferença entre a peneira recursiva e iterativa manteve praticamente o mesmo tempo de execução e de chamadas.



Devido ao fota da diferença dentre eles ser tao pequena o grafico ficou praticamente um em cima do outro. Mostrando que não ha tanta diferença entre o algoritmo com a função peneira recursiva e interativa.

2.5 Quick Sort

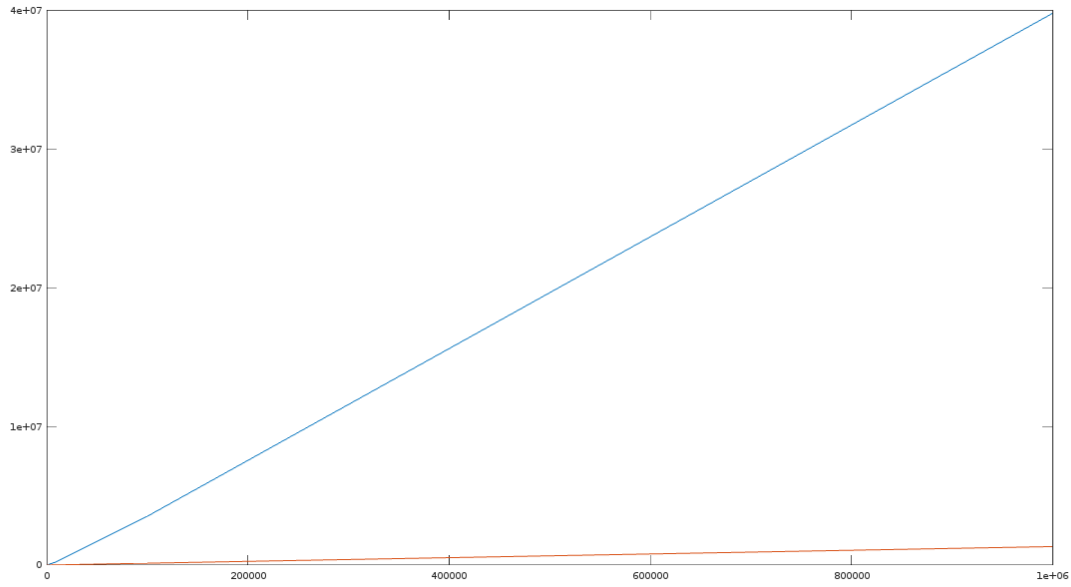
O Algoritmo de ordenação quick é um algoritmo que em geral é o mais rápidodos algoritmos de ordenação analisados neste documento, porém em algumas certas instâncias, o Quicksort pode ser tão lento quanto os algoritmos insertion e selection. .Oalgoritmo consome tempo proporcional a nlog n em média

$$T(n)=O(n);$$

e proporcional a n^2 no pior caso.

$$T(n)=O(n^2);$$

#Iterativo	#Recursivo	#Diferenca	#Tamanho
13	34	21	10
137	1008	871	100
1331	19242	17911	1000
13395	251452	238057	10000
133455	3518452	3384997	100000
1333807	39771806	38437999	1000000



O grafico que é mostrado acima é o grafico do algoritmo de ordenação quick sort. A reta vermelhra representa o algoritmo iterativo e a azul recursivo. Diferente dos outros algoritmos que possuiam a versão recursiva mais rapida que a iterativa o algoritmo quick sort possui a recursiva mais lenta que a iterativa. Para certos tipos de vetores o quick é o mais rapido, porém se o vetor já estiver ordenado ou quase ordenado seu numero de comparações sera n^2 não sendo melhor que os outros algoritmos.

3. Comparação entros os tipos de algoritmos de ordenação

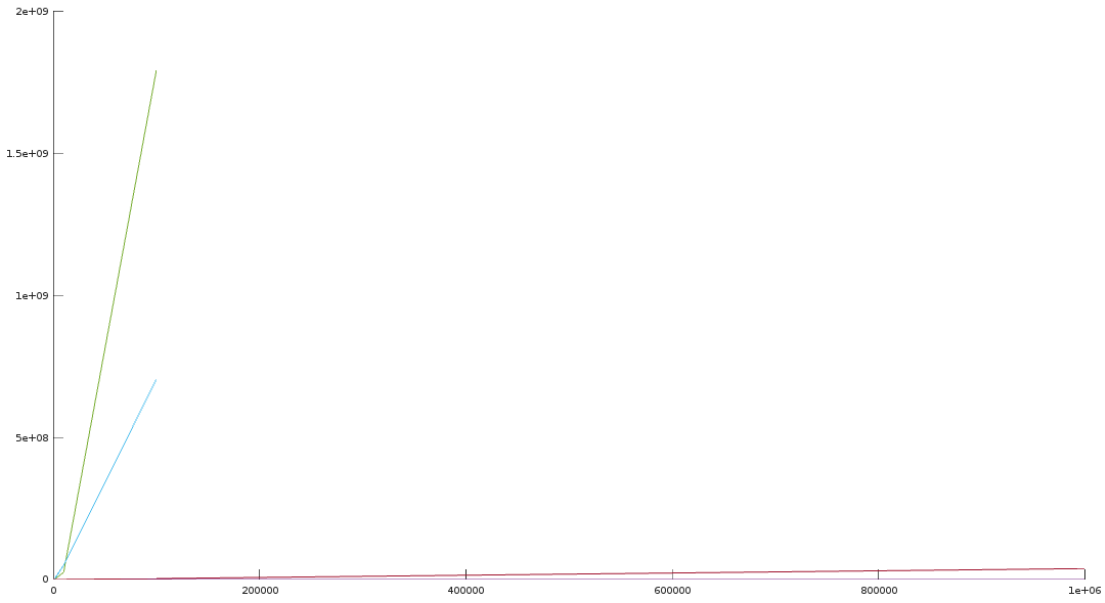
Você pode ver na tabela abaixo a comparação entre os principais algoritmos existente.

Algoritmo	Comparações			Movimentações			Espaço	Estável	In situ
	Melhor	Médio	Pior	Melhor	Médio	Pior			
Bubble	$O(n^2)$			$O(n^2)$			$O(1)$	**Sim**	**Sim**
Selection	$O(n^2)$			$O(n)$			$O(1)$	**Não***	**Sim**
Insertion	$O(n)$	$O(n^2)$		$O(n)$	$O(n^2)$		$O(1)$	**Sim**	**Sim**
Merge	$O(n\ log\ n)$			–			$O(n)$	**Sim**	**Não**
Quick	$O(n\ log\ n)$		$O(n^2)$	–			$O(n)$	**Não***	**Sim**
Shell	$O(n^{1.25})$ ou $O(n\ (ln\ n)^2)$			–			$O(1)$	**Não**	**Sim**

* Existem versões estáveis.

** Alguns algoritmos que estão nesta tambela não foram aqui mencinados. Tais como buble e shell.

E de forma mais grafica segue uma comparação entre os algoritmos mencionados neste trabalho:



4. Metodologia Experimental

Neste trabalho, foram realizadas duas análises: uma que avalia o desempenho de um algoritmo recursivo e a outra que avalia o desempenho de um algoritmo iterativo. Para avaliar o desempenho dos algoritmos, uma comparação foi feita usando um conjunto de dados de valores simulados aleatoriamente.

O método iterativo enumera todos os grupos possíveis, garantindo que a solução ótima seja encontrada. E o metodo recursivo trabalha em função de si própria para resolver o seu problema. Para a resolvução dos problemas foram utilizadas vetores de 8 elementos ate 10000. Para ser justo com o método aleatório, foi

adotado o mesmo número de soluções candidatas que as utilizadas pela proposta. Os dois métodos foram implementados em C usando o ambiente do gedit e o compilador gcc 5.4.0. Todas as execuções foram realizadas em um computador laptop com um processador AMD E1-1500 APU with Radeon(tm), 4GB RAM e 1.5 GHz. Para a produção dos graficos foi utilizado o software livre octave .O objetivo desta análise é verificar a correlação IterativoxRecursivo e comparar os diferentes algoritmos.

Para a realização da análise, foi realizado vários testes em que cada um deles os vetores possuiam o tamanho cada vez maior. Para que não houvesse interações e modificações foram criados 2 vetores com os mesmos elementos e mesmo tamanho.

Uma vez apresentado o estudo para cada caso, os métodos são comparados entre si. Cada método, que foi aplicado exclusivamente sobre um vetor diferente, porem com valores iguais. O objetivo é avaliar se, na prática, o método iterativo ou o recursivo possui melhores desempenhos. O desempenho de cada algoritmo leva em consideração o numero de ações e ou chamadas recursivas.

5.Conclusão

Os Algoritmos de ordenação é uma representação computacional de um problema comum do dia a dia do ser huma. É feito em uma linguagem de programação e serve, assim como no “mundo real”, para ordenar um conjunto de dados. Existem diversos tipos de algoritmos de ordenação cada um com seu tipo de implementação e seu objetivo levando em conta os diversos modos de ordenação que o ser humano faz sem perceber.

A importância da ordenação vem do fato de que precisamos varias vezes de dados ordenados para que possamo realizar uma devida ação tais como busca e pesquisa.