METROCAMP

MÉTODOS DE ORDENAÇÃO

ESTRUTURA DE DADOS AVANÇADA

ALUNOS:

GABRIEL MALAQUIAS – 421.439.548-40
LUIZ FERNANDO SANTOS – 425.473.328.33
VINICIUS VEIGA - 376.182.038-06

PROFESSOR:
FÁBIO PELISSONI

CIÊNCIA DA COMPUTAÇÃO

CAMPINAS – 2015

Sumário

Introdução

Em vários momentos, nos deparamos com a necessidade de trabalhar com dados ordenados, mas nem sempre abstraímos estes dados da forma desejada. Por isso uma das atividades mais utilizada na computação é a ordenação.

Existem inúmeros algoritmos de ordenação, neste trabalho escolhemos 6 destes algoritmos para analisarmos seu funcionamento, complexidade e performance. Usando a linguagem de programação C, vamos analisar o tempo que cada algoritmo leva para organizar um vetor com valores randômicos, em ordem crescente e decrescente, contendo 15 mil posições em ordem crescente.

Os métodos de ordenação escolhidos foram:

- Bubble Sort;
- Insertion Sort;
- Quick Sort;
- Gnome Sort;
- HeapSort;
- Merge Sort.

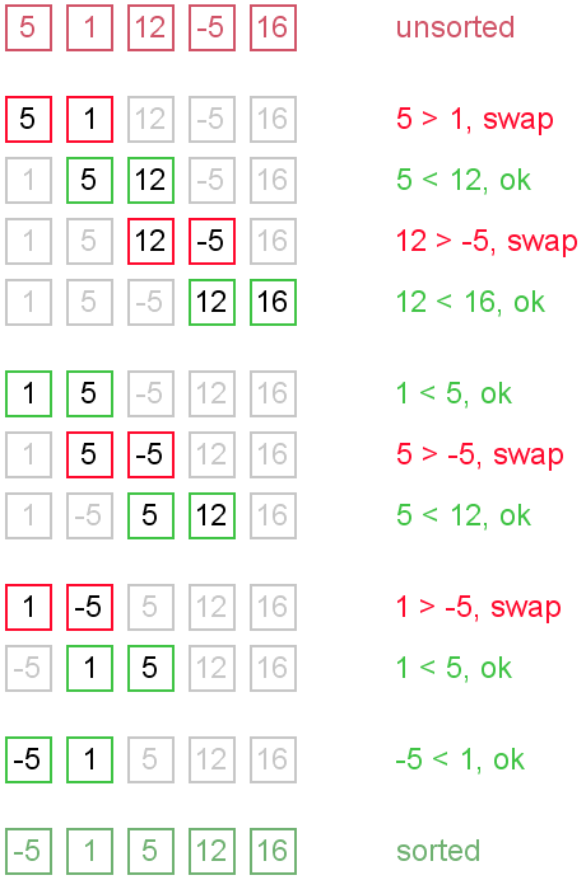
Os algoritmos utilizados para o estudo podem ser encontrados em:
https://github.com/gmalaquias/benchmark-metodos-ordenacao

Bubble Sort

O método Bubble Sort também chamado de ordenação por trocas consiste em comparar os pares consecutivos de elementos e trocá-los de posição de acordo com a ordem proposta;

A cada iteração o maior elemento fica na última posição do conjunto de elementos.

Exemplo:



Complexidade: Baixa
Performance: Baixa
Tempo de Execução:

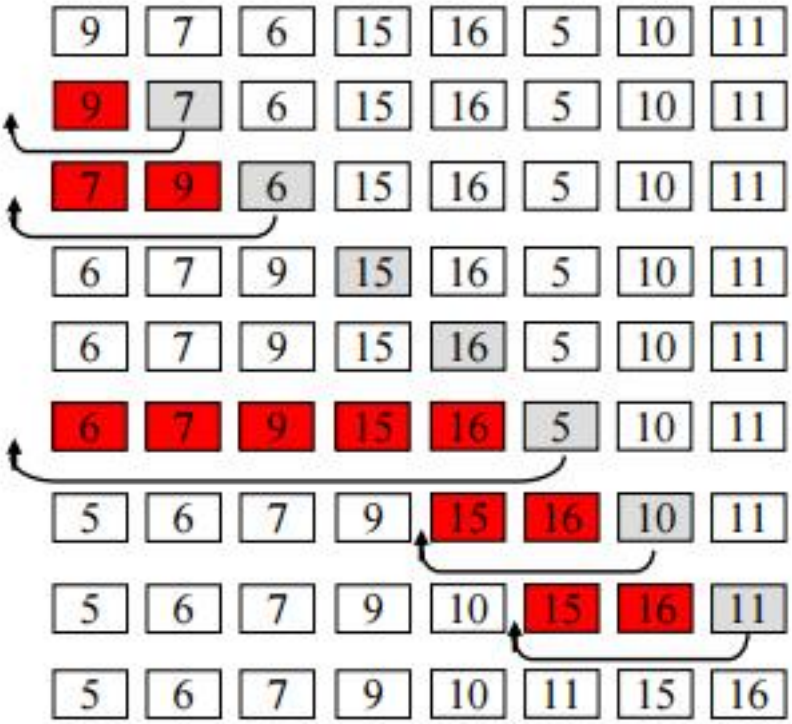
- Vetor Randômico: 975ms;
- Vetor Crescente: 428ms;
- Vetor Decrescente: 954ms.

Insertion Sort

O método Insertion Sort ou ordenação por inserção tem este nome por estar baseada na inserção de cada um dos elementos no conjunto de elementos anteriores a ele segundo a ordem desejada;

Inicia-se o algoritmo no segundo índice o vetor, e vai inserindo no local correto.

Exemplo:



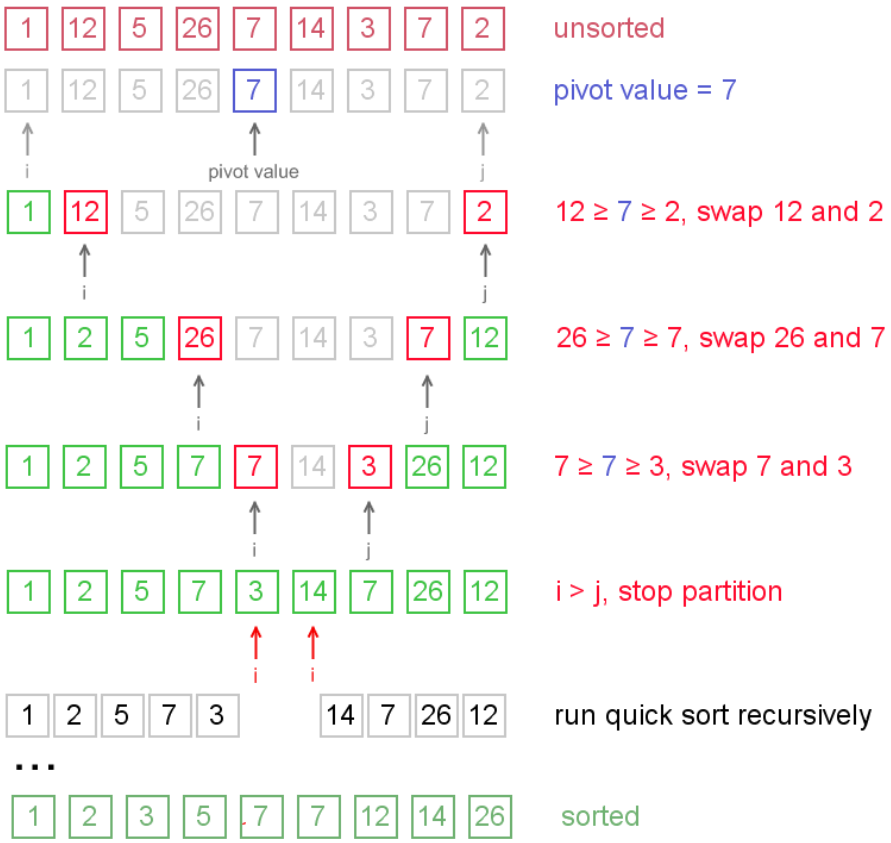
Complexidade: Baixa
Performance: Média
Tempo de Execução:

- Vetor Randômico: 273ms;
- Vetor Crescente: <1ms;
- Vetor Decrescente: 538ms.

Quick Sort

A estratégia básica do quicksort é a de "dividir para conquistar". Inicia-se com a escolha de um elemento da lista, designado pivô.
A lista é então rearranjada de forma que todos os elementos maiores do que o pivô fiquem de um dos lados do pivô e todos os elementos menores fiquem do outro lado (ficando assim o pivô na sua posição definitiva); recursivamente, repete-se este processo para cada sub-lista e, no final, o resultado é uma lista ordenada.

Exemplo:



Complexidade: Alta
Performance: Alta
Tempo de Execução:

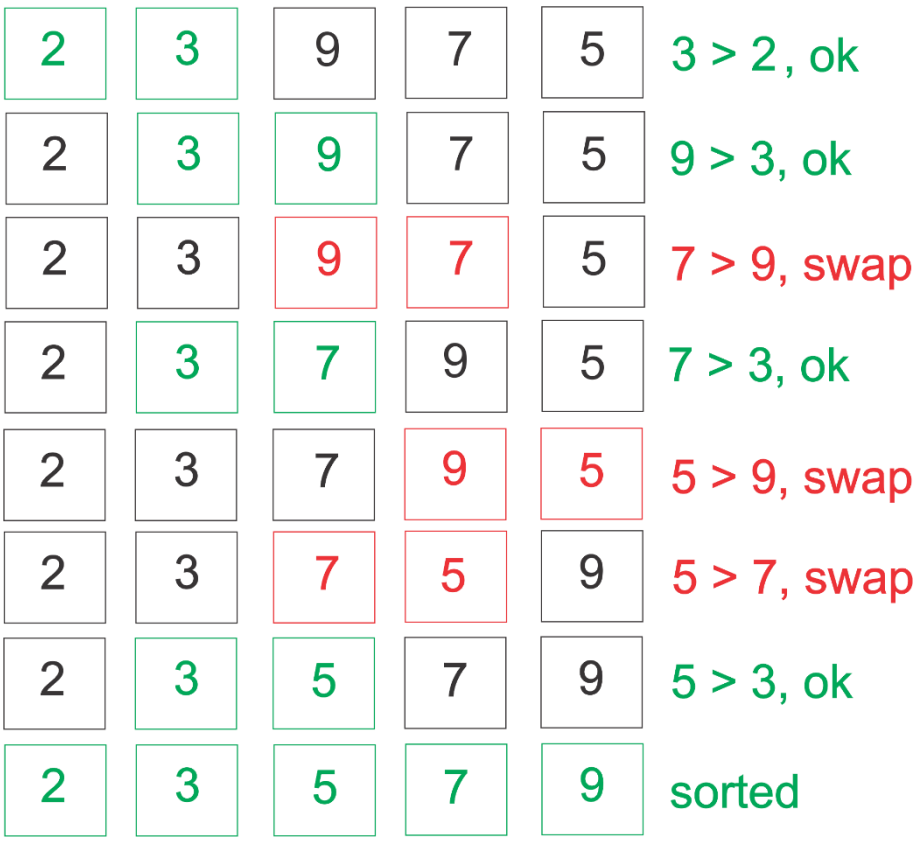
- Vetor Randômico: 3ms;
- Vetor Crescente: 1ms;
- Vetor Decrescente: 1ms.

Gnome Sort

Algoritmo similiar ao Insertion sort com a diferença que o Gnome sort leva um elemento para sua posição correta, com uma seqüencia grande de trocas assim como o Bubble sort

O algoritmo percorre o vetor comparando seus elementos dois a dois, assim que ele encontra um elemento que está na posição incorreta, ou seja, um número maior antes de um menor, ele troca a posição dos elementos, e volta com este elemento até que encontre o seu respectivo lugar.

Exemplo:



Complexidade: Baixa
Performance: Baixa
Tempo de Execução:

- Vetor Randômico: 687ms;
- Vetor Crescente: <1ms;
- Vetor Decrescente: 1363ms.

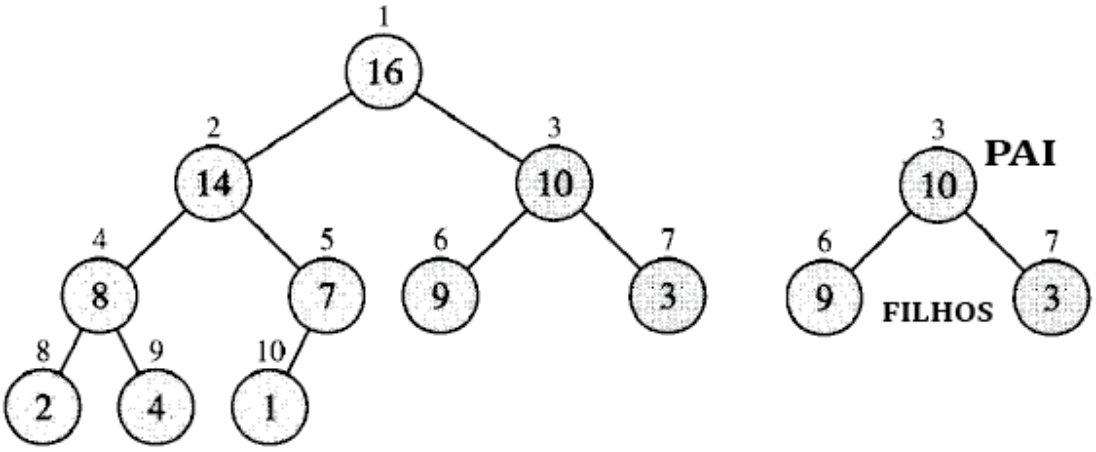
HeapSort

O heapsort utiliza uma estrutura de dados chamada heap, para ordenar os elementos a medida que os insere na estrutura. Assim, ao final das inserções, os elementos podem ser sucessivamente removidos da raiz da heap, na ordem desejada.

A heap pode ser representada como uma árvore ou como um vetor.

A cada ciclo do algoritmo o pai de todos os ramos é o maior entre todos os valores, então ele sai da arvore e é inserido na última posição do vetor, novamente inicia-se as comparações até que o pai seja a maior valor e possa sair da arvore, assim ele continua até organizar todo o vetor.

Exemplo:



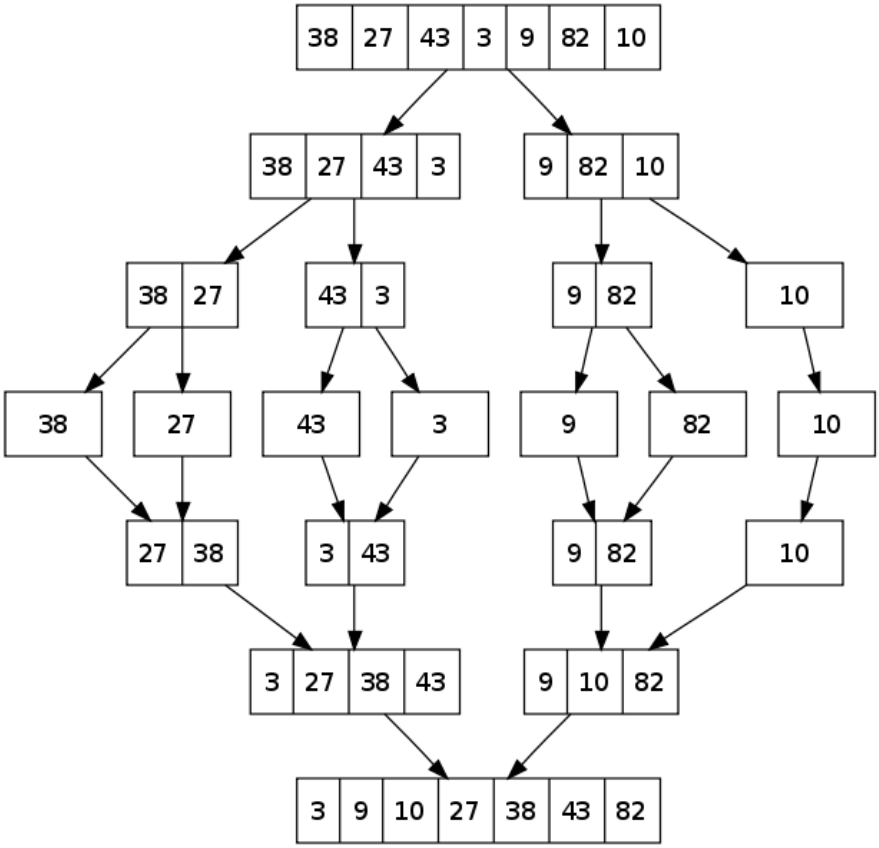
Complexidade: Alta
Performance: Alta
Tempo de Execução:

- Vetor Randômico: 3ms;
- Vetor Crescente: 2ms;
- Vetor Decrescente: 2ms.

Merge Sort

Assim como o quick sort o merge sort utiliza a estratégia de "dividir para conquistar". Inicia-se dividindo todo o vetor ao meio recursivamente até que não seja mais possível esta divisão. E logo se inicia a comparação criando vetores já ordenados. Ao final ao restar 2 vetores, através de mais comparações, é criado um vetor único com seus elementos ordenados.

Exemplo:



Complexidade: Alta
Performance: Alta
Tempo de Execução:

- Vetor Randômico: 44ms;
- Vetor Crescente: 99ms;
- Vetor Decrescente: 98ms.

Conclusão

Ao final dos testes, podemos concluir que o melhor algoritmo de ordenação é o Quick Sort, já que independentemente da forma que o vetor já está ordenado, ele sempre será rápido. Porém as outras opções sempre são uteis, já que você pode não levar em consideração o tempo mas sim a complexidade de criar o algoritmo.

Segue abaixo o ranking de métodos por cada tipo de vetor que foi ordenado:

Vetor Randômico

Posição	Nome	Tempo
1º	Quick Sort	3ms
1º	HeapSort	3ms
2º	Merge Sort	44ms
3º	Insertion Sort	273ms
4º	Gnome Sort	687ms
5º	Bubble Sort	975ms

Vetor Crescente

Posição	Nome	Tempo
1º	Insertion Sort	<1ms
1º	Gnome Sort	<1ms
2º	Quick Sort	1ms
3º	HeapSort	2ms
4º	Merge Sort	99ms
5º	Bubble Sort	428ms

Vetor Decrescente

Posição	Nome	Tempo
1º	Quick Sort	1ms
2º	HeapSort	2ms
3º	Merge Sort	98ms
4º	Insertion Sort	538ms
5º	Bubble Sort	954ms
6º	Gnome Sort	1363ms

Média das Ordenações

Posição	Nome	Tempo
1º	Quick Sort	2ms
2º	HeapSort	3ms
3º	Merge Sort	80ms
4º	Insertion Sort	270ms
5º	Gnome Sort	779ms
6º	Bubble Sort	785ms

Referências

QuickSort <http://www.knoow.net/ciencinformtelec/informatica/quicksort.htm> Acesso em: 20/03/2015.

GnomeSort <http://rosettacode.org/wiki/Sorting_algorithms/Gnome_sort Acesso em: 20/03/2015.

Explicação HeapSort <https://www.youtube.com/watch?v=bj-H47puSU> Acesso em: 20/03/2015.

HeapSort <http://www.ebah.com.br/content/ABAAAAn6EAC/ordenacao-dados-heapsort> Acesso em: 20/03/2015.

Estudo Merge Sort http://pt.slideshare.net/luizaguerra/estudo-do-algoritmo-de-ordenao-mergesort> Acesso em: 21/03/2015.

QuickSort <http://www.algostructure.com/sorting/quicksort.php> Acesso em: 21/03/2015.

Estudo de métodos de ordenação <http://nicholasandre.com.br/sorting> Acesso em: 21/03/2015.