METROCAMP

MÉTODOS DE ORDENAÇÃO

ESTRUTURA DE DADOS AVANÇADA

|  |  |
| --- | --- |
|  | ALUNOS:  GABRIEL MALAQUIAS – 421.439.548-40  LUIZ FERNANDO SANTOS – 425.473.328.33  PROFESSOR:  FÁBIO PELISSONI  CIÊNCIA DA COMPUTAÇÃO |
|  |  |

CAMPINAS – 2015

Sumário

[Introdução 1](#_Toc414475771)

[Insertion Sort 2](#_Toc414475772)

[Bubble Sort 2](#_Toc414475773)

[Quick Sort 2](#_Toc414475774)

[Gnome Sort 2](#_Toc414475775)

[HeapSort 2](#_Toc414475776)

[Merge Sort 2](#_Toc414475777)

# 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 cronometrar o tempo que cada algoritmo demora para organizar um mesmo vetor com 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: 975ms

# 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;

Exemplo:

Complexidade: Baixa

Performance: Média

Tempo de Execução: 260ms

# 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: 3ms

# 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: 671ms

# 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.

Exemplo:

Complexidade: Alta

Performance: Alta

Tempo de Execução: 3ms

# 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: 84ms

# Bibliografia

<http://www.knoow.net/ciencinformtelec/informatica/quicksort.htm>

<http://rosettacode.org/wiki/Sorting_algorithms/Gnome_sort>

<https://www.youtube.com/watch?v=bj-H47-puSU> – HeapSort

<http://www.ebah.com.br/content/ABAAAAn6EAC/ordenacao-dados-heapsort>

<http://pt.slideshare.net/luizaguerra/estudo-do-algoritmo-de-ordenao-mergesort>

<https://www.youtube.com/watch?v=qO5jmDB7VZU> – Merge Sort