

UNIVERSIDADE DE SÃO PAULO - ICMC  
BACHARELADO EM SISTEMAS DE INFORMAÇÃO  
INTRODUÇÃO À CIÊNCIA DA COMPUTAÇÃO II

Professor Claudio Fabiano Motta Toledo  
Monitor da Disciplina Jesimar da Silva Arantes

# Trabalho Ordenação e Buscas

Nome:Guilherme Balloni

N USP 7143510

Linkon José Santos Louvison

N USP 7143698

São Carlos, Novembro de 2017

## SUMÁRIO

<a href="#"><u>SUMÁRIO</u></a>	<a href="#"><u>1</u></a>
<a href="#"><u>INTRODUÇÃO</u></a>	<a href="#"><u>2</u></a>
<a href="#"><u>MENU E APRESENTAÇÃO</u></a>	<a href="#"><u>3</u></a>
<a href="#"><u>TESTES DE ORDENAÇÃO</u></a>	<a href="#"><u>4</u></a>
<a href="#"><u>TESTES DE BUSCA</u></a>	<a href="#"><u>5</u></a>
<a href="#"><u>CONCLUSÃO</u></a>	<a href="#"><u>6</u></a>

## INTRODUÇÃO

O propósito deste trabalho é implementar os algoritmos de ordenação: Quicksort, Heapsort e Counting Sort e os algoritmos de busca: Busca Sequencial, Busca Binária, Busca por Interpolação e Árvore Binária de Busca.

Para realizar essas implementações utilizamos de notas de aulas, onde o professor e o monitor da disciplina disponibilizaram os principais algoritmos, e também exercícios de runcodes que serviram de base para a elaboração desse projeto.

Os testes foram realizados utilizando as duas tabelas dadas pelo professor, a Countries-of-the World.xls e a Estimativa-População-Municipios.xlsx. Para podermos utilizar essas tabelas realizamos um pré processamento nelas, nesse pré-processamento tiramos as linhas que julgamos não serem necessárias para o trabalho e separamos as colunas por um “;” (ponto e vírgula). Ao final de cada linha da tabela utilizamos dois “;”(ponto e vírgula) para demarcar a mudança de linha no arquivo.

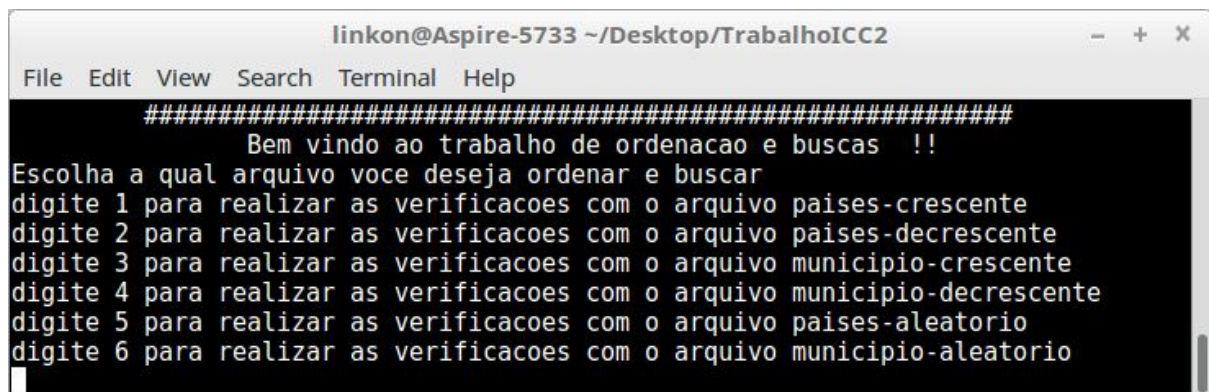
Separamos o trabalho em funções para ordenação (arquivos ordenacao.c e ordenação.h), funções para busca (arquivos busca.c e busca.h) e o main.c que utiliza todas essas funções.

O trabalho foi compilado no ambiente Linux utilizando o arquivo de Makefile que criamos para facilitar a compilação, junto a pasta do trabalho encontra-se um arquivo Leia-me.txt que explica como compilar e executar o trabalho com o nosso Makefile.

Para realizar os testes utilizamos o Notebook Acer Aspire 5733 com processador I3 370M (2,4Gz) e 6Gb de Ram.

## MENU E APRESENTAÇÃO DO PROGRAMA

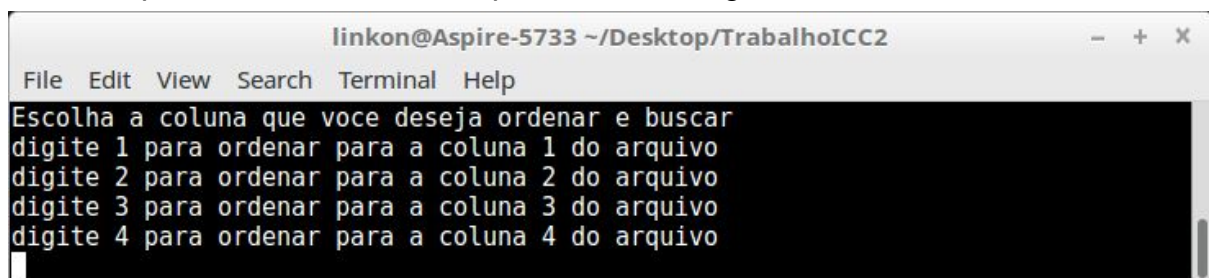
Segue abaixo a tela de menu inicial do programa:



```
linkon@Aspire-5733 ~/Desktop/TrabalhoICC2
File Edit View Search Terminal Help
#####
      Bem vindo ao trabalho de ordenacao e buscas !!
Escolha a qual arquivo voce deseja ordenar e buscar
digite 1 para realizar as verificacoes com o arquivo paises-crescente
digite 2 para realizar as verificacoes com o arquivo paises-decrescente
digite 3 para realizar as verificacoes com o arquivo municipio-crescente
digite 4 para realizar as verificacoes com o arquivo municipio-decrescente
digite 5 para realizar as verificacoes com o arquivo paises-aleatorio
digite 6 para realizar as verificacoes com o arquivo municipio-aleatorio
█
```

Aqui o usuário deve escolher com qual arquivo deve realizar as verificações que o programa faz.

Depois de se escolher a arquivo temos a seguinte tela:

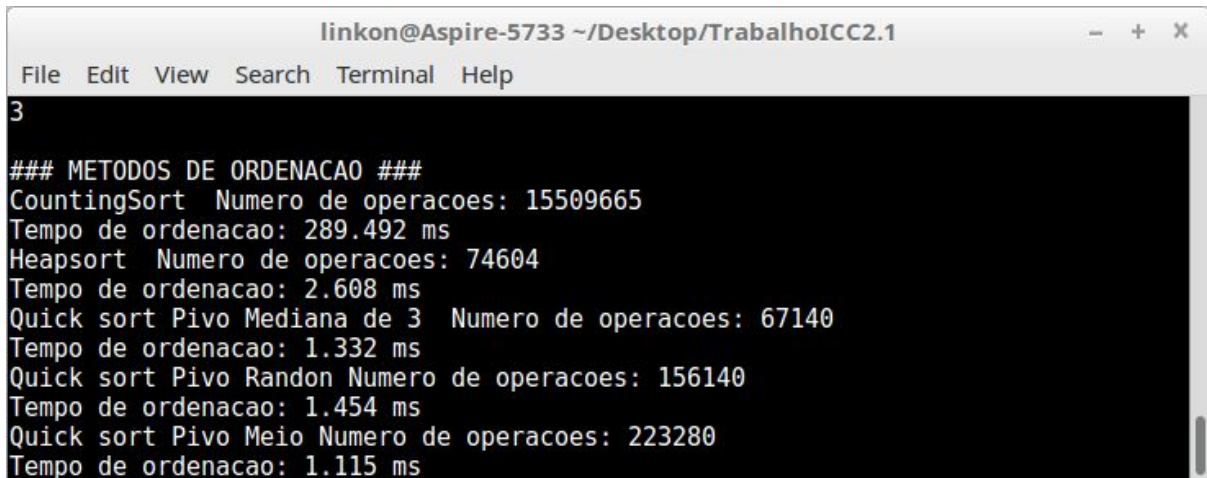


```
linkon@Aspire-5733 ~/Desktop/TrabalhoICC2
File Edit View Search Terminal Help
Escolha a coluna que voce deseja ordenar e buscar
digite 1 para ordenar para a coluna 1 do arquivo
digite 2 para ordenar para a coluna 2 do arquivo
digite 3 para ordenar para a coluna 3 do arquivo
digite 4 para ordenar para a coluna 4 do arquivo
█
```

Onde o usuário deve escolher sobre qual coluna do arquivo ele quer realizar a ordenação e as buscas.

## TESTES DE ORDENAÇÃO

Segue abaixo um exemplo de saída da ordenação de uma coluna da tabela:

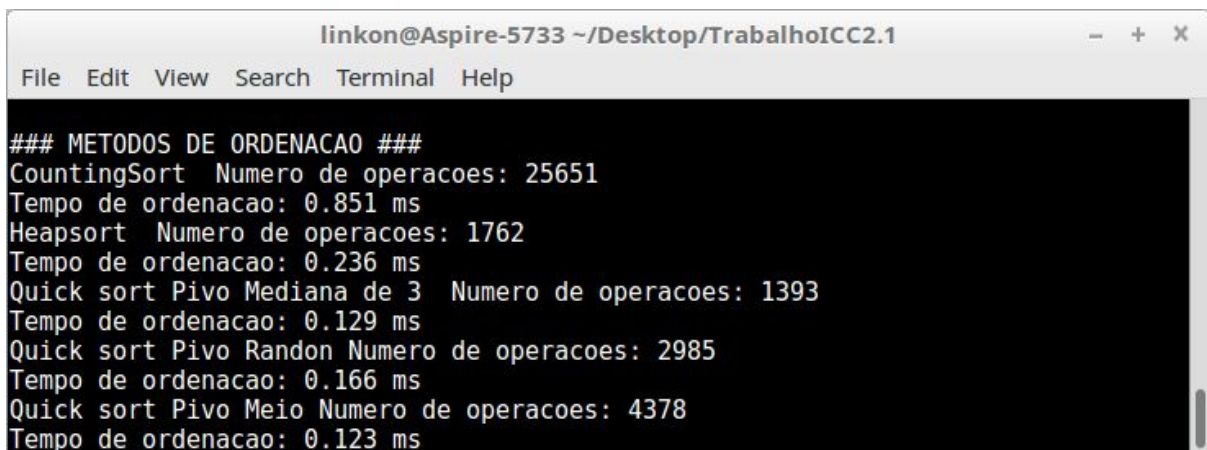
A terminal window titled 'linkon@Aspire-5733 ~/Desktop/TrabalhoICC2.1' with a menu bar (File, Edit, View, Search, Terminal, Help). The output shows the number '3' followed by a separator '### METODOS DE ORDENACAO ###'. It then lists performance metrics for CountingSort, Heapsort, and three variants of Quick sort (Pivo Mediana de 3, Pivo Randon, Pivo Meio), including the number of operations and execution time in milliseconds.

```
linkon@Aspire-5733 ~/Desktop/TrabalhoICC2.1
File Edit View Search Terminal Help
3
### METODOS DE ORDENACAO ###
CountingSort Numero de operacoes: 15509665
Tempo de ordenacao: 289.492 ms
Heapsort Numero de operacoes: 74604
Tempo de ordenacao: 2.608 ms
Quick sort Pivo Mediana de 3 Numero de operacoes: 67140
Tempo de ordenacao: 1.332 ms
Quick sort Pivo Randon Numero de operacoes: 156140
Tempo de ordenacao: 1.454 ms
Quick sort Pivo Meio Numero de operacoes: 223280
Tempo de ordenacao: 1.115 ms
```

Neste exemplo utilizamos o arquivo de município aleatório e a tabela 3 para realizar a ordenação, vale lembrar que a tabela 3 é do tipo inteiro.

Podemos verificar que os tempos de ordenação muito pequenos para os métodos heapsort, e as implementações do quick sort.

Segue abaixo outro exemplo de saída para ordenação

A terminal window titled 'linkon@Aspire-5733 ~/Desktop/TrabalhoICC2.1' with a menu bar (File, Edit, View, Search, Terminal, Help). The output shows a separator '### METODOS DE ORDENACAO ###' followed by performance metrics for CountingSort, Heapsort, and three variants of Quick sort (Pivo Mediana de 3, Pivo Randon, Pivo Meio), including the number of operations and execution time in milliseconds.

```
linkon@Aspire-5733 ~/Desktop/TrabalhoICC2.1
File Edit View Search Terminal Help
### METODOS DE ORDENACAO ###
CountingSort Numero de operacoes: 25651
Tempo de ordenacao: 0.851 ms
Heapsort Numero de operacoes: 1762
Tempo de ordenacao: 0.236 ms
Quick sort Pivo Mediana de 3 Numero de operacoes: 1393
Tempo de ordenacao: 0.129 ms
Quick sort Pivo Randon Numero de operacoes: 2985
Tempo de ordenacao: 0.166 ms
Quick sort Pivo Meio Numero de operacoes: 4378
Tempo de ordenacao: 0.123 ms
```

Aqui pegamos como entrada a tabela de países e ordenamos ela segundo a sua segunda coluna.

## TESTES DE BUSCA

Segue abaixo um exemplo de saída do teste de busca

```
linkon@Aspire-5733 ~/Desktop/TrabalhoICC2.1
File Edit View Search Terminal Help
### METODOS DE BUSCA###
Busca Sequencial Encontra na posicao: 4732    Numero de operacoes: 4733
Tempo de Busca: 0.072 ms
Busca Binaria Encontra na posicao: 4732    Numero de operacoes: 12
Tempo de Busca: 0.001 ms
Busca Interpolacao Encontra na posicao: 4732    Numero de operacoes: 15
Tempo de Busca: 0.011 ms
Busca Arvore Binaria Encontra: 1 e Numero de operacoes: 3470
Tempo de Busca: 0.147 ms
linkon@Aspire-5733 ~/Desktop/TrabalhoICC2.1 $
```

Onde buscamos na tabela 3 , pela coluna 3 que corresponde código do município pelo código 30808.

Segue abaixo um exemplo com busca alfanumérica

```
linkon@Aspire-5733 ~/Desktop/TrabalhoICC2.1
File Edit View Search Terminal Help
### METODOS DE BUSCA###
Busca Sequencial Encontra na posicao: 69    Numero de operacoes: 70
Tempo de Busca: 0.004 ms
Busca Binaria Encontra na posicao: 69    Numero de operacoes: 8
Tempo de Busca: 0.001 ms
Floating point exception
linkon@Aspire-5733 ~/Desktop/TrabalhoICC2.1 $
```

Onde buscamos na tabela 2 pela coluna 1 que corresponde ao nome do País, buscamos pelo País “France “ , vale lembrar que devemos alterar no programa o elemento que desejamos fazer a busca.

A busca por interpolação não funciona para valores alfanuméricos.

## CONCLUSÃO

Podemos verificar que os algoritmos heapsort e quicksort são muito rápidos se comparados com o counting sort( ele só é bom para casos particulares ). também conseguimos verificar que o tempo de busca do quicksort supera o do heapsort nos testes , entretanto as 3 versões de quicksort não mostram grande diferença em tempo de processamento, seria necessário uma base de dados maior para verificar essa diferença de tempo.

Já quanto aos algoritmos de busca verificamos uma larga vantagem na utilização da busca binária, e uma larga desvantagem na utilização da busca sequencial.

Com isso podemos concluir que a melhor escolha para realizarmos a ordenação e busca de arquivos sejam eles numéricos ou alfanuméricos é a combinação do algoritmo quicksort com o algoritmo da busca binária.