Relatório - Trabalhos Práticos da disciplina Algoritmos e Estruturas de Dados III

Daniel Valadares de Souza Felix[1], **Gustavo Silvestre Almeida Conceição**[1], **Larissa Valadares Silqueira**[1]

[1]Instituto de Ciências Exatas e Informática
Pontifícia Universidade Católica de Minas Gerais (PUC Minas)
Belo Horizonte – MG – Brasil

Resumo. *Este relatório se dedica a descrever o processo de desenvolvimento dos Trabalhos Práticos da disciplina Algoritmos e Estruturas de Dados 3, e exibir os resultados obtidos a partir da execução do sistema bancário criado. Essa descrição engloba todo o processo, desde a divisão das atividades entre os membros do grupo até as decisões de implementação utilizadas pelos mesmos, durante a programação do sistema. Na sessão de testes e resultados são exibidas capturas de tela do sistema, que mostram as funcionalidades desenvolvidas e seu uso prático em um caso de teste.*

1. Introdução

Este relatório visa apresentar o processo de desenvolvimento dos trabalhos práticos que compuseram a disciplina Algoritmos e Estruturas de Dados 3 do curso Ciência da Computação, realizados pelos autores, durante o segundo semestre de 2022. Nas próximas seções serão expostos os critérios de escolha utilizados para implementar o que foi pedido pelo professor da disciplina, Hayala Nepomuceno Curto, os resultados obtidos e também as dificuldades encontradas durante o processo.

No total, foram desenvolvidos três trabalhos práticos complementares, sendo a descrição de cada um deles:

1. Implementação de um sistema responsável por realizar operações de CRUD em um arquivo indexado.
    (a) Criação de um sistema bancário fictício, que possua as seguintes funcionalidades: criar uma conta, realizar uma transferência entre duas contas distintas, ler um registro, atualizar um registro e deletar um registro.
    (b) Ordenação externa do arquivo de registros do sistema, a partir da utilização do método de intercalação balanceada comum, considerando a memória principal com limitação de 5 registros e usando 2 caminhos.
    (c) Criação de arquivos de índices utilizando Árvore B+, Hash Estendido e Lista Invertida, sendo que para esse último, deveriam ser criados índices tanto para o campo Nome quanto para o campo Cidade das contas armazenadas.
2. Implementação dos algoritmos de compressão de dados Huffman e LZW sobre o arquivo onde estão armazenados os registros criados e manipulados pelo sistema bancário, exibindo a taxa de compressão, e os tempos de compactação e descompactação de cada um dos métodos.

3. Implementação de um algoritmo de casamento de padrões para pesquisar por uma palavra, de forma exata ou aproximada (à depender do algoritmo), dentro do arquivo de registro; e de um algoritmo de criptografia, ambos à escolha dos autores, a fim de que o campo Senha da conta seja cifrado antes de ser armazenado, e decifrado ao ser exibido para o usuário.

2. Desenvolvimento

O Trabalho Prático I aqui descrito foi desenvolvido por Daniel e Larissa, o aluno Gustavo realizou esse trabalho individualmente, juntando-se à dupla a partir do segundo Trabalho Prático. A codificação do sistema bancário foi feita por Daniel, a partir do ambiente de desenvolvimento integrado Visual Studio Code, na linguagem de programação Java.

O primeiro passo, foi criar a classe Conta e compô-la dos atributos e métodos especificados no enunciado do Trabalho Prático, feito pelo professor, a fim de que ela pudesse ser instanciada posteriormente, como objeto, e seus dados pudessem ser manipulados de maneira correta. Nela, há métodos que permitem o envio e retorno de valores que serão utilizados pelo sistema bancário, para gerir as contas de seus usuários. Também foram implementados métodos que enviam os dados de uma conta para serem exibidos e que convertem o mesmos em um vetor de Bytes a fim de serem armazenados no banco de dados criado com essa finalidade.

Esse banco de dados, por sua vez, é criado e manipulado por outra classe (ContaDAO), que é responsável por realizar as operações de leitura e escrita do arquivo, ou seja, é a partir dessa classe que se pode criar, deletar, atualizar ou ler (CRUD) uma conta. Uma decisão de implementação feita pelos autores foi utilizar o marcador lápide nas operações de atualização e remoção, de acordo com a necessidade, para que as exclusões no arquivo de dados não fossem físicas, o que demandaria alto custo para reajustar os dados no arquivo.

Assim, ao deletar uma conta, o marcador lápide é ativado, e ao atualizar um dado de uma conta, o sistema analisa se o espaço reservado para esse dado precisa de alguma modificação, se sim, a conta original é marcada para exclusão, e uma nova conta, com o mesmo ID da original é criada no fim do arquivo. Havendo a necessidade de exibir as contas armazenadas pelo sistema, aquelas contas que possuem o marcador lápide ativado ficam ocultas para o usuário.

Os métodos da classe ContaDAO também foram ajustados para comportar as implementações dos arquivos de índices Árvore B+, Hash Estendido e Lista Invertida [para os campos Nome e Cidade], que possuem operações de CRUD, e também métodos que possibilitam a boa utilização e manipulação do sistema, como a verificação de nomes de usuários (que precisam ser únicos), e de armazenamento de e-mails, já que um mesmo usuário, pode possuir mais de um e-mail cadastrado.

A ordenação do arquivo foi feita através da intercalação balanceada comum, conforme enunciado. Nesse algoritmo de ordenação externa são criados alguns arquivos temporários que são preenchidos com as chaves que se quer ordenar, no caso, os IDs da conta, e a partir da comparação entre os arquivos temporários, se criam outros até que o arquivo temporário final esteja completamente ordenado. A partir disso, o arquivo de

dados original é sobrescrito com a nova ordem de contas.

Para os arquivos de índices que foram solicitados, têm-se a implementação da Árvore B+ que gerou algumas dificuldades durante a codificação; do Hash Estendido, onde os registros ficam armazenados em Buckets (e as informações mais relevantes das contas passam a ser as chaves e seus endereços), que por sua vez estão inseridos em um diretório, que é cuidadosamente trabalhado a fim de comportar todos os Buckets, aumentando sua capacidade de acordo com a necessidade; e também da Lista Invertida, que é preenchida a cada inserção ou remoção feita no sistema. Quando é solicitado pelo usuário que se exiba todas as ocorrências de um nome ou cidade do arquivo de dados, a lista invertida retorna os endereços em que aparecem tal palavra-chave, se ela existir no banco de dados.

Por fim, tem-se a classe Menu que faz a conexão entre todas as classes citadas, e também entre as que foram criadas para os próximos trabalhos práticos [a serem descritos]. Nela, há funções que verificam se os dados que estão sendo escritos estão de acordo com os tipos especificados, a fim que os dados que estão entrando no sistema sejam armazenados sem erros. Há também as funções que manipulam o objeto para que as operações permitidas pelo sistema sejam executadas de forma eficaz, e o menu propriamente dito, que recebe as informações e faz o direcionamento necessário. Uma decisão de implementação dos autores é a opção de excluir ou não o banco de dados criado, que aparece durante a finalização do sistema, com o intuito de permitir que o utilizador decida entre manter o arquivo de dados, seja para visualização ou futuras manipulações no sistema, ou deletá-lo, caso ele não seja mais necessário.

O Trabalho Prático II foi codificado pelos alunos Daniel e Gustavo, sendo que Daniel ficou responsável por implementar o algoritmo de dicionário LZW. Esse método de compressão tem uma etapa de pré-processamento que consiste em criar um hashmap com todos os caracteres da tabela ascii. Após essa etapa, o arquivo de contas é percorrido, cuidadosamente, e a cada caractere lido, é inserido no arquivo ”compactado”, seu valor numérico, de acordo com o hashmap. A cada leitura é criado um novo campo no hashmap contendo a substring em análise mais o próximo caractere, de forma que trechos que se repetem ao longo do arquivo, possam aproveitar as chaves dessas combinações.

Gustavo, por sua vez, implementou o algoritmo Huffman, que consistiu na criação de quatro classes complementares. A classe de controle desse algoritmo é responsável por percorrer o arquivo das contas bancárias, e criar nós (classe HuffmanFolha) para cada um dos caracteres lidos. É válido mencionar que não se cria novos nós para caracteres que são lidos novamente (repetidos), nesses casos, é incrementado [em um, por vez] o valor que acompanha o nó de cada elemento, indicando a quantidade de vezes que aquele caractere aparece ao longo de todo o arquivo de registros.

Após a etapa de criação de nós, a árvore é montada, de forma que os pesos (quantidade/frequência) entre os nós de cada subárvore sejam equivalentes, ou seja, de modo que a árvore fique balanceada. A partir disso, um método da classe de controle percorre a árvore atribuindo o valor 0 (zero) aos galhos à esquerda e 1 (um) aos galhos à direita. Com isso é possível determinar a codificação de cada elemento existente no arquivo, e criar então um novo arquivo compactado, já que os bits da codificação ocuparão um espaço menor que o bytes existentes no arquivo de registros, e que representam, por

caractere, os dados armazenados pelo sistema bancário.

Por último, para o Trabalho Prático III, o aluno Daniel optou por realizar o casamento de padrões no sistema bancário a partir do algoritmo desenvolvido por Knuth, Morris e Pratt, também conhecido como KMP. A implementação desse método consistiu na criação de funções que usam um padrão escolhido pelo usuário do sistema (a partir da linha de entrada), para percorrer o arquivo de registros e retornar as ocorrências desse padrão. Inicialmente é construída uma função de retorno para o padrão, onde este é percorrido com ele mesmo, de modo que trechos que "remontam" o padrão possam ser aproveitados, aumentando a eficiência do algoritmo.

O método principal da classe utiliza a função de retorno criada para então percorrer o arquivo contas.db procurando por ocorrências do padrão. Quando o padrão é encontrado, o método guarda o endereço da conta lida, a fim de que a mesma seja exibida, juntamente com outras ocorrências do padrão, ao fim da execução do algoritmo, ou seja, quando o arquivo tiver sido percorrido por completo.

A parte de criptografia do campo Senha dos registros armazenados foi feita por Gustavo, a partir do método de transposição de colunas. Para a implementação desse algoritmo, as classes Conta e ContaDAO precisaram ser modificadas para comportar as novas alterações. Na classe Conta foi inserido um novo atributo contendo o valor da chave de criptografia que seria utilizada pelo método posteriormente, e no ContaDAO foram criadas funções que tratariam o CRUD base de acordo com a necessidade do método de criptografia. Uma decisão de implementação dos autores foi criar duas opções de exibição das contas durante a execução do sistema. Essas consistem em exibir os dados das contas com o campo Senha cifrado ou não, a fim de que a diferença pudesse ser analisada.

Para a codificação desse algoritmo foi criada uma matriz em que o número de colunas é igual ao tamanho da chave escolhida, as linhas são então preenchidas com a senha do usuário de forma sequencial, ou seja do primeiro caractere para o último e da esquerda para a direita. A cifragem é feita a partir da seleção das colunas da matriz, que são retiradas na ordem alfabética dos caracteres da chave. Já a decifragem realiza o processo inverso, com a mensagem cifrada é feito uma operação de mod entre o tamanho da mesma e o da chave, o resultado indica a quantidade de espaços preenchidos que aparecerão na última linha da matriz, da coluna mais à esquerda para a direita. A matriz é então preenchida de forma vertical com os trechos da mensagem codificada, respeitando os espaços vazios e o máximo de caracteres por coluna, calculados através de uma divisão simples entre os tamanhos mencionados, que retorna o número de linhas totalmente preenchidas. Ao final, a senha original pode ser retirada da matriz de forma sequencial a começar pela primeira linha e primeira coluna.

3. Testes e Resultados

Os testes e demonstrações dos trabalhos práticos, este último feito por meio da criação de vídeos explicativos sobre cada parte do sistema e seu funcionamento, foram realizados pela aluna Larissa, responsável também por relatar todo o processo de desenvolvimento do projeto. A seguir estão representadas cada uma das funcionalidades do sistema, a partir de imagens retiradas do sistema bancário em execução.

Figura 1. Inicialização do Sistema

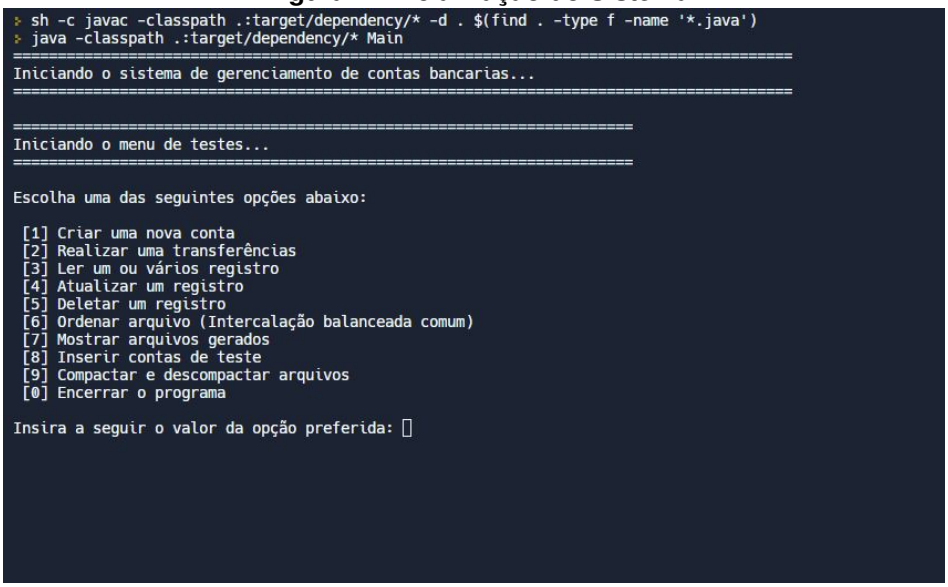


Figura 2. Criação de uma Conta

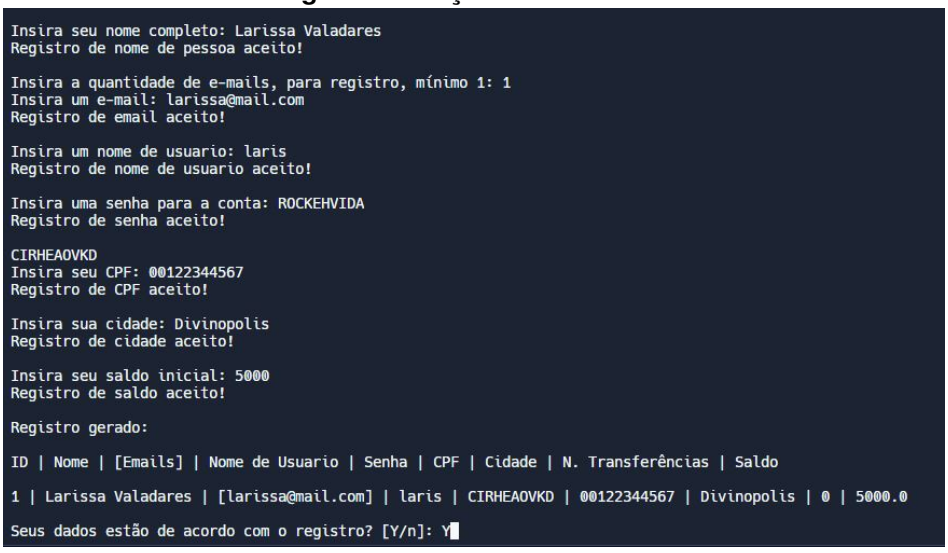


Figura 3. Transferência entre Contas Distintas

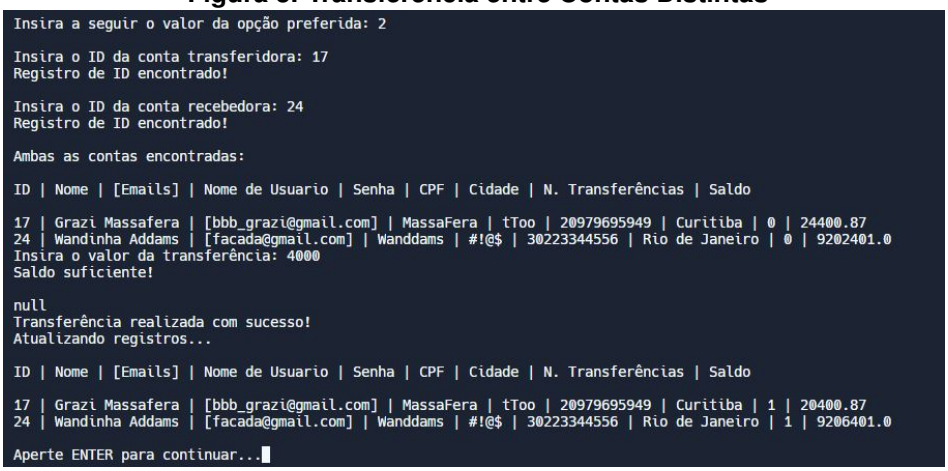
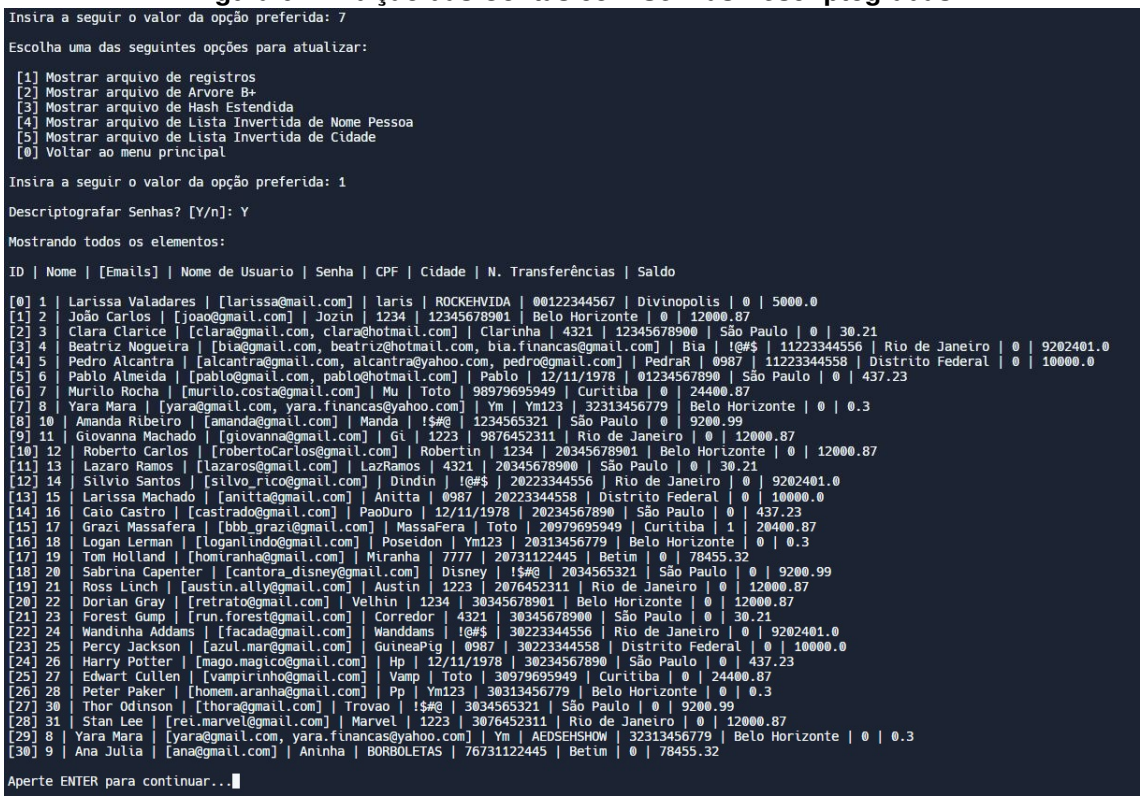


Figura 4. Atualização de Dado da Conta

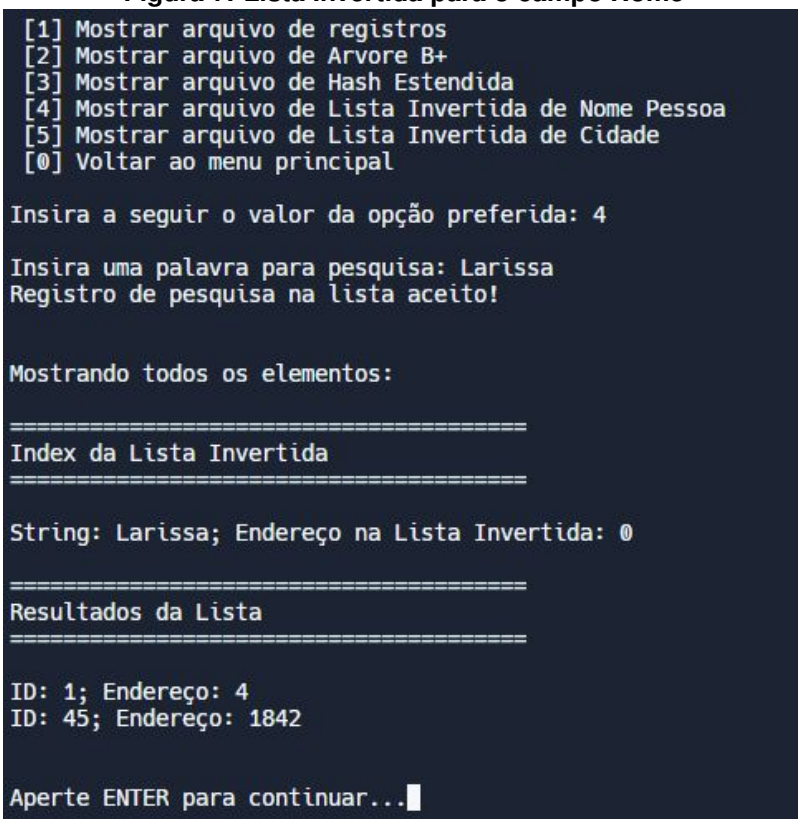
Figura 5. Exclusão de um Registro

Figura 6. Exibição das Contas com Senhas Descriptogradas



```
Insira a seguir o valor da opção preferida: 7

Escolha uma das seguintes opções para atualizar:

 [1] Mostrar arquivo de registros
 [2] Mostrar arquivo de Arvore B+
 [3] Mostrar arquivo de Hash Estendida
 [4] Mostrar arquivo de Lista Invertida de Nome Pessoa
 [5] Mostrar arquivo de Lista Invertida de Cidade
 [0] Voltar ao menu principal

Insira a seguir o valor da opção preferida: 1

Descriptografar Senhas? [Y/n]: Y

Mostrando todos os elementos:

ID | Nome | [Emails] | Nome de Usuario | Senha | CPF | Cidade | N. Transferências | Saldo

[0] 1 | Larissa Valadares | [larissa@mail.com] | laris | ROCKEHVIDA | 00122344567 | Divinopolis | 0 | 5000.0
[1] 2 | João Carlos | [joao@gmail.com] | Jozin | 1234 | 12345678901 | Belo Horizonte | 0 | 12000.87
[2] 3 | Clara Clarice | [clara@gmail.com, clara@hotmail.com] | Clarinha | 4321 | 12345678900 | São Paulo | 0 | 30.21
[3] 4 | Beatriz Nogueira | [bia@gmail.com, beatriz@hotmail.com, bia.financas@gmail.com] | Bia | !@#$ | 11223344556 | Rio de Janeiro | 0 | 9202401.0
[4] 5 | Pedro Alcantra | [alcantra@gmail.com, alcantra@yahoo.com, pedro@gmail.com] | PedraR | 0987 | 11223344558 | Distrito Federal | 0 | 10000.0
[5] 6 | Pablo Almeida | [pablo@gmail.com, pablo@hotmail.com] | Pablo | 12/11/1978 | 01234567890 | São Paulo | 0 | 437.23
[6] 7 | Murilo Rocha | [murilo.costa@gmail.com] | Mu | Toto | 98979695949 | Curitiba | 0 | 24400.87
[7] 8 | Yara Mara | [yara@gmail.com, yara.financas@yahoo.com] | Ym | Ym123 | 32313456779 | Belo Horizonte | 0 | 0.3
[8] 10 | Amanda Ribeiro | [amanda@gmail.com] | Manda | !$#@ | 1234565321 | São Paulo | 0 | 9200.99
[9] 11 | Giovanna Machado | [giovanna@gmail.com] | Gi | 1223 | 9876452311 | Rio de Janeiro | 0 | 12000.87
[10] 12 | Roberto Carlos | [robertoCarlos@gmail.com] | Robertin | 1234 | 20345678901 | Belo Horizonte | 0 | 12000.87
[11] 13 | Lazaro Ramos | [lazaros@gmail.com] | LazRamos | 4321 | 20345678900 | São Paulo | 0 | 30.21
[12] 14 | Silvio Santos | [silvo_rico@gmail.com] | Dindin | !@#$ | 20223344556 | Rio de Janeiro | 0 | 9202401.0
[13] 15 | Larissa Machado | [anitta@gmail.com] | Anitta | 0987 | 20223344558 | Distrito Federal | 0 | 10000.0
[14] 16 | Caio Castro | [castrado@gmail.com] | PaoDuro | 12/11/1978 | 20234567890 | São Paulo | 0 | 437.23
[15] 17 | Grazi Massafera | [bbb_grazi@gmail.com] | MassaFera | Toto | 20979695949 | Curitiba | 1 | 20400.87
[16] 18 | Logan Lerman | [loganlindo@gmail.com] | Poseidon | Ym123 | 20313456779 | Belo Horizonte | 0 | 0.3
[17] 19 | Tom Holland | [homiranha@gmail.com] | Miranha | 7777 | 20731122445 | Betim | 0 | 78455.32
[18] 20 | Sabrina Capenter | [cantora_disney@gmail.com] | Disney | !$#@ | 2034565321 | São Paulo | 0 | 9200.99
[19] 21 | Ross Linch | [austin.ally@gmail.com] | Austin | 1223 | 2076452311 | Rio de Janeiro | 0 | 12000.87
[20] 22 | Dorian Gray | [retrato@gmail.com] | Velhin | 1234 | 30345678901 | Belo Horizonte | 0 | 12000.87
[21] 23 | Forest Gump | [run.forest@gmail.com] | Corredor | 4321 | 30345678900 | São Paulo | 0 | 30.21
[22] 24 | Wandinha Addams | [facada@gmail.com] | Wanddams | !@#$ | 30223344556 | Rio de Janeiro | 0 | 9202401.0
[23] 25 | Percy Jackson | [azul.mar@gmail.com] | GuineaPig | 0987 | 30223344558 | Distrito Federal | 0 | 10000.0
[24] 26 | Harry Potter | [mago.magico@gmail.com] | Hp | 12/11/1978 | 30234567890 | São Paulo | 0 | 437.23
[25] 27 | Edwart Cullen | [vampirinho@gmail.com] | Vamp | Toto | 30979695949 | Curitiba | 0 | 24400.87
[26] 28 | Peter Paker | [homem.aranha@gmail.com] | Pp | Ym123 | 30313456779 | Belo Horizonte | 0 | 0.3
[27] 30 | Thor Odinson | [thora@gmail.com] | Trovao | !$#@ | 3034565321 | São Paulo | 0 | 9200.99
[28] 31 | Stan Lee | [rei.marvel@gmail.com] | Marvel | 1223 | 3076452311 | Rio de Janeiro | 0 | 12000.87
[29] 8 | Yara Mara | [yara@gmail.com, yara.financas@yahoo.com] | Ym | AEDSEHSHOW | 32313456779 | Belo Horizonte | 0 | 0.3
[30] 9 | Ana Julia | [ana@gmail.com] | Aninha | BORBOLETAS | 76731122445 | Betim | 0 | 78455.32

Aperte ENTER para continuar...
```

Figura 7. Lista Invertida para o campo Nome



```
 [1] Mostrar arquivo de registros
 [2] Mostrar arquivo de Arvore B+
 [3] Mostrar arquivo de Hash Estendida
 [4] Mostrar arquivo de Lista Invertida de Nome Pessoa
 [5] Mostrar arquivo de Lista Invertida de Cidade
 [0] Voltar ao menu principal

Insira a seguir o valor da opção preferida: 4

Insira uma palavra para pesquisa: Larissa
Registro de pesquisa na lista aceito!


Mostrando todos os elementos:

========================================
Index da Lista Invertida
========================================

String: Larissa; Endereço na Lista Invertida: 0

========================================
Resultados da Lista
========================================

ID: 1; Endereço: 4
ID: 45; Endereço: 1842


Aperte ENTER para continuar...
```

Figura 8. Hash Estendido

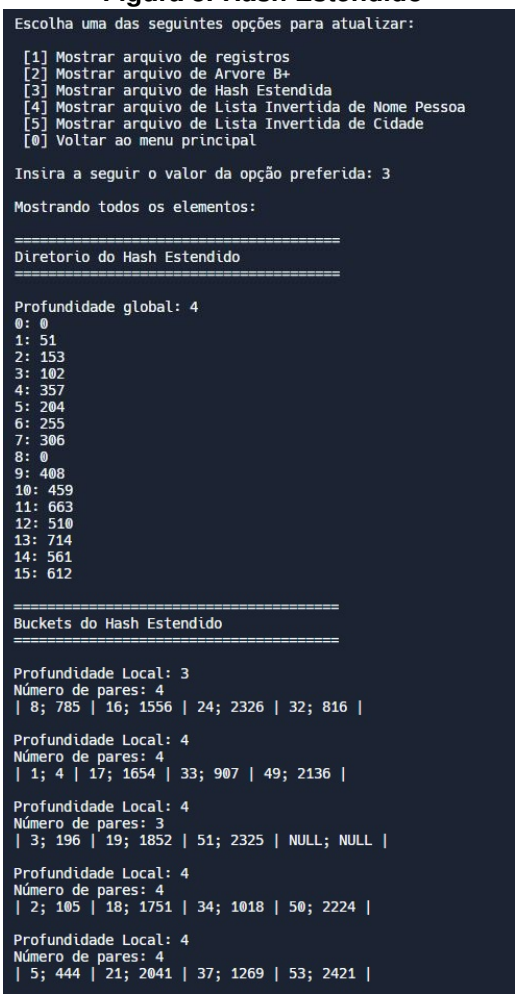


Figura 9. Compressão do Arqivo de Registros

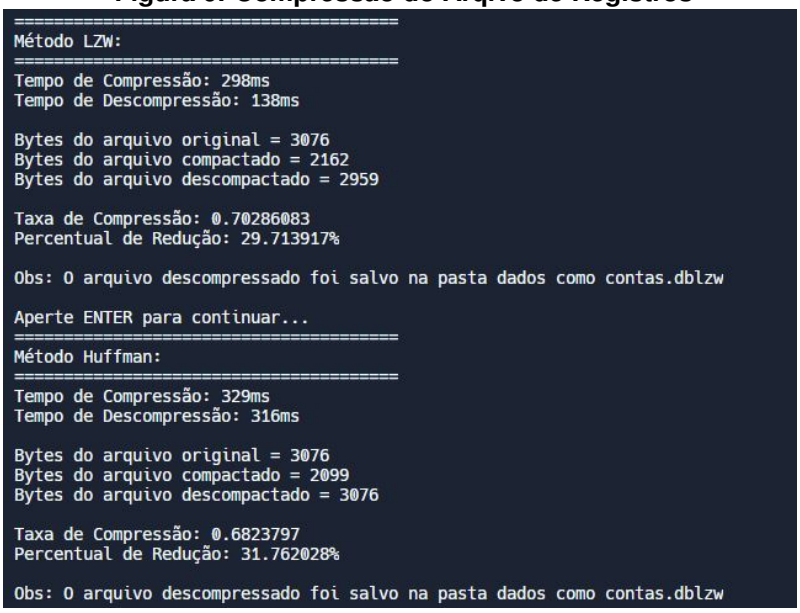
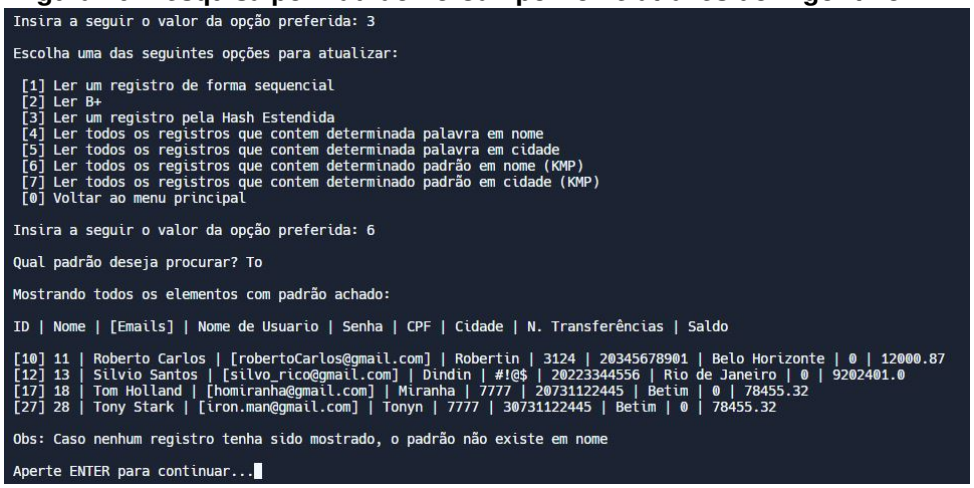
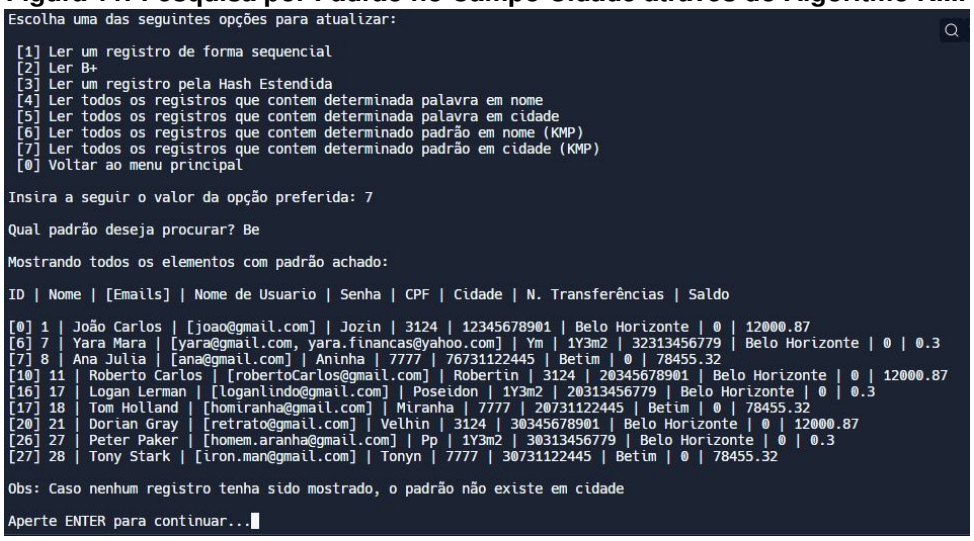


Figura 10. Pesquisa por Padrão no Campo Nome através do Algoritmo KMP



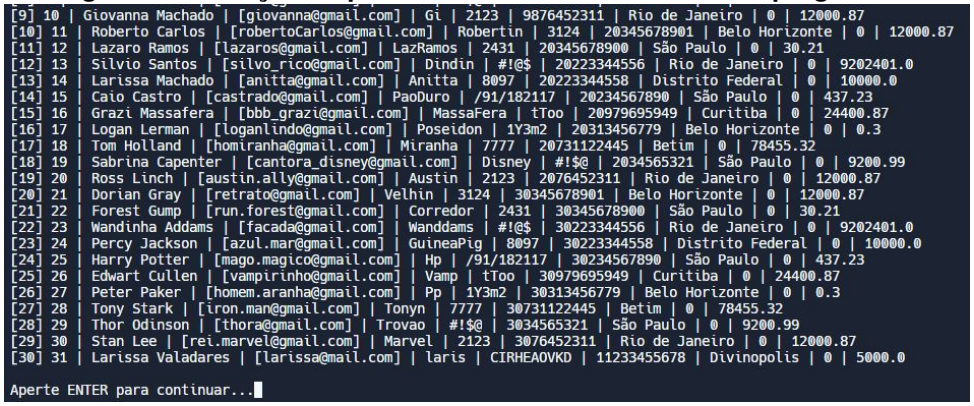
```
Insira a seguir o valor da opção preferida: 3

Escolha uma das seguintes opções para atualizar:

 [1] Ler um registro de forma sequencial
 [2] Ler B+
 [3] Ler um registro pela Hash Estendida
 [4] Ler todos os registros que contem determinada palavra em nome
 [5] Ler todos os registros que contem determinada palavra em cidade
 [6] Ler todos os registros que contem determinado padrão em nome (KMP)
 [7] Ler todos os registros que contem determinado padrão em cidade (KMP)
 [0] Voltar ao menu principal

Insira a seguir o valor da opção preferida: 6

Qual padrão deseja procurar? To

Mostrando todos os elementos com padrão achado:

ID | Nome | [Emails] | Nome de Usuario | Senha | CPF | Cidade | N. Transferências | Saldo

[10] 11 | Roberto Carlos | [robertoCarlos@gmail.com] | Robertin | 3124 | 20345678901 | Belo Horizonte | 0 | 12000.87
[12] 13 | Silvio Santos | [silvo_rico@gmail.com] | Dindin | #!@$ | 20223344556 | Rio de Janeiro | 0 | 9202401.0
[17] 18 | Tom Holland | [homiranha@gmail.com] | Miranha | 7777 | 20731122445 | Betim | 0 | 78455.32
[27] 28 | Tony Stark | [iron.man@gmail.com] | Tonyn | 7777 | 30731122445 | Betim | 0 | 78455.32

Obs: Caso nenhum registro tenha sido mostrado, o padrão não existe em nome

Aperte ENTER para continuar...
```

Figura 11. Pesquisa por Padrão no Campo Cidade através do Algoritmo KMP



```
Escolha uma das seguintes opções para atualizar:

 [1] Ler um registro de forma sequencial
 [2] Ler B+
 [3] Ler um registro pela Hash Estendida
 [4] Ler todos os registros que contem determinada palavra em nome
 [5] Ler todos os registros que contem determinada palavra em cidade
 [6] Ler todos os registros que contem determinado padrão em nome (KMP)
 [7] Ler todos os registros que contem determinado padrão em cidade (KMP)
 [0] Voltar ao menu principal

Insira a seguir o valor da opção preferida: 7

Qual padrão deseja procurar? Be

Mostrando todos os elementos com padrão achado:

ID | Nome | [Emails] | Nome de Usuario | Senha | CPF | Cidade | N. Transferências | Saldo

[0] 1 | João Carlos | [joao@gmail.com] | Jozin | 3124 | 12345678901 | Belo Horizonte | 0 | 12000.87
[6] 7 | Yara Mara | [yara@gmail.com, yara.financas@yahoo.com] | Ym | 1Y3m2 | 32313456779 | Belo Horizonte | 0 | 0.3
[7] 8 | Ana Julia | [ana@gmail.com] | Aninha | 7777 | 76731122445 | Betim | 0 | 78455.32
[10] 11 | Roberto Carlos | [robertoCarlos@gmail.com] | Robertin | 3124 | 20345678901 | Belo Horizonte | 0 | 12000.87
[16] 17 | Logan Lerman | [loganlindo@gmail.com] | Poseidon | 1Y3m2 | 20313456779 | Belo Horizonte | 0 | 0.3
[17] 18 | Tom Holland | [homiranha@gmail.com] | Miranha | 7777 | 20731122445 | Betim | 0 | 78455.32
[20] 21 | Dorian Gray | [retrato@gmail.com] | Velhin | 3124 | 30345678901 | Belo Horizonte | 0 | 12000.87
[26] 27 | Peter Paker | [homem.aranha@gmail.com] | Pp | 1Y3m2 | 30313456779 | Belo Horizonte | 0 | 0.3
[27] 28 | Tony Stark | [iron.man@gmail.com] | Tonyn | 7777 | 30731122445 | Betim | 0 | 78455.32

Obs: Caso nenhum registro tenha sido mostrado, o padrão não existe em cidade

Aperte ENTER para continuar...
```

Figura 12. Exibição de parte das Contas com Senhas Criptografadas



```
[9] 10 | Giovanna Machado | [giovanna@gmail.com] | Gi | 2123 | 9876452311 | Rio de Janeiro | 0 | 12000.87
[10] 11 | Roberto Carlos | [robertoCarlos@gmail.com] | Robertin | 3124 | 20345678901 | Belo Horizonte | 0 | 12000.87
[11] 12 | Lazaro Ramos | [lazaros@gmail.com] | LazRamos | 2431 | 20345678900 | São Paulo | 0 | 30.21
[12] 13 | Silvio Santos | [silvo_rico@gmail.com] | Dindin | #!@$ | 20223344556 | Rio de Janeiro | 0 | 9202401.0
[13] 14 | Larissa Machado | [anitta@gmail.com] | Anitta | 8097 | 20223344558 | Distrito Federal | 0 | 10000.0
[14] 15 | Caio Castro | [castrado@gmail.com] | PaoDuro | /91/182117 | 20234567890 | São Paulo | 0 | 437.23
[15] 16 | Grazi Massafera | [bbb_grazi@gmail.com] | MassaFera | tToo | 20979695949 | Curitiba | 0 | 24400.87
[16] 17 | Logan Lerman | [loganlindo@gmail.com] | Poseidon | 1Y3m2 | 20313456779 | Belo Horizonte | 0 | 0.3
[17] 18 | Tom Holland | [homiranha@gmail.com] | Miranha | 7777 | 20731122445 | Betim | 0 | 78455.32
[18] 19 | Sabrina Capenter | [cantora_disney@gmail.com] | Disney | #!$@ | 2034565321 | São Paulo | 0 | 9200.99
[19] 20 | Ross Linch | [austin.ally@gmail.com] | Austin | 2123 | 2076452311 | Rio de Janeiro | 0 | 12000.87
[20] 21 | Dorian Gray | [retrato@gmail.com] | Velhin | 3124 | 30345678901 | Belo Horizonte | 0 | 12000.87
[21] 22 | Forest Gump | [run.forest@gmail.com] | Corredor | 2431 | 30345678900 | São Paulo | 0 | 30.21
[22] 23 | Wandinha Addams | [facada@gmail.com] | Wanddams | #!@$ | 30223344556 | Rio de Janeiro | 0 | 9202401.0
[23] 24 | Percy Jackson | [azul.mar@gmail.com] | GuineaPig | 8097 | 30223344558 | Distrito Federal | 0 | 10000.0
[24] 25 | Harry Potter | [mago.magico@gmail.com] | Hp | /91/182117 | 30234567890 | São Paulo | 0 | 437.23
[25] 26 | Edwart Cullen | [vampirinho@gmail.com] | Vamp | tToo | 30979695949 | Curitiba | 0 | 24400.87
[26] 27 | Peter Paker | [homem.aranha@gmail.com] | Pp | 1Y3m2 | 30313456779 | Belo Horizonte | 0 | 0.3
[27] 28 | Tony Stark | [iron.man@gmail.com] | Tonyn | 7777 | 30731122445 | Betim | 0 | 78455.32
[28] 29 | Thor Odinson | [thora@gmail.com] | Trovao | #!$@ | 3034565321 | São Paulo | 0 | 9200.99
[29] 30 | Stan Lee | [rei.marvel@gmail.com] | Marvel | 2123 | 3076452311 | Rio de Janeiro | 0 | 12000.87
[30] 31 | Larissa Valadares | [larissa@mail.com] | laris | CIRHEAOVKD | 11233455678 | Divinopolis | 0 | 5000.0

Aperte ENTER para continuar...
```

4. Conclusão

Com a realização destas atividades práticas foi possível compreender, implementar e desenvolver a maioria dos algoritmos vistos em sala durante este semestre, sendo possível observar e pontuar as diferenças notáveis entre cada um dos métodos de ordenação externa, indexação, compressão e criptografia estudados.

De cada método aprendeu-se um pouco sobre suas características principais, tanto a partir da codificação do sistema bancário quanto a partir da resolução das listas teóricas disponibilizadas, que foram de grande auxílio para a compreensão do funcionamento dos algoritmos. Também foi possível determinar o cenário onde cada um dos métodos possuiria uma maior eficiência em relação aos demais, definindo assim, a aplicabilidade ideal de cada algoritmo.

Dessa maneira, pode-se concluir que desenvolver os Trabalhos Práticos solicitados pelo professor, foi de grande importância não apenas para a realização da disciplina, mas também para a formação dos alunos diante da utilização e implementação desses métodos, que podem ser necessários em diversos cenários, como projetos de pesquisa na área, trabalhos como programador/desenvolvedor, entre outros.