UNIVERSIDADE FEDERAL DE JUIZ DE FORA
DEPARTAMENTO DE CIÊNCIA DA COMPUTAÇÃO

DAUANE JOICE
MICHEL ANDRADE
RODRIGO OLIVEIRA
WENDELL GUIMARÃES

TRABALHO DE ESTRUTURA DE DADOS II

"AVALIAÇÕES DE FILMES"

JUIZ DE FORA
2019

Objetivo

O vigente trabalho tem por finalidade o escopo da análise dos dados obtidos durante a execução dos algoritmos de ordenação propostos e comparar empiricamente seus respectivos desempenhos relacionando os resultados entre si.

Primeira parte

Cenário 1

Nesta etapa do presente trabalho está sendo avaliado o desempenho do método de ordenação Quicksort Recursivo considerando que os elementos a serem ordenados são inteiros armazenados em um vetor de tamanho N e que os elementos a serem ordenados são registros armazenados em um vetor de tamanho também N. Para cada tipo de dado, ocorreu a implementação do algoritmo Quicksort Recursivo, que recebe como entrada o conjunto de elementos a serem ordenados.

Foram contabilizados: o número de comparações de chaves, o número de cópias de registros e o tempo total gasto na ordenação. Foram desenvolvidos métodos para importar os conjuntos de elementos aleatórios e estes foram chamados uma vez para cada um dos N elementos a serem ordenados.

O Quicksort foi aplicado à entradas escolhidas aleatoriamente com diferentes tamanhos. Para cada valor de N, foram gerados 5 conjuntos de elementos diferentes, utilizando sementes diferentes para o gerador de números aleatórios.

Os algoritmos serão analisados comparando os valores médios de 10 execuções (uma para cada semente) para cada valor de N testado. Para cada valor de N lido, foi gerado um novo conjunto de elementos, depois esse foi ordenado, contabilizando as estatísticas de desempenho e armazenado as mesmas em um arquivo de saída (relatório).

Cenário 2

Neste cenário, foram comparados os desempenhos de diferentes variações do algoritmo Quicksort para ordenar um conjunto de N inteiros (e seus respectivos dados) armazenados em um vetor. Cada elemento do vetor contém um identificador chamado de USERID, importado diretamente do arquivo proveniente da base de dados analisada.

As variações do Quicksort implementadas e avaliadas nesta etapa são: Quicksort Recursivo, Quicksort Mediana e Quicksort Inserção. Foram realizados experimentos com as três variações considerando vetores aleatoriamente gerados com tamanhos N, com N sendo igual a 1000, 5000, 10000, 50000 e 100000. Para cada valor de N, foram realizados testes com 10 sementes diferentes e avaliados os valores médios do tempo de execução, do número de comparações de chaves e do número de cópias dos registros.

Cenário 3

Neste momento, foram comparadas as melhores variações do Quicksort com o Insertionsort, o Mergesort, o Heapsort e o Shellsort, este último figurando como o escolhido pelo grupo, para ordenar um conjunto de N inteiros. Considerando que cada elemento do vetor possui um identificador chamado USERID, os dados foram importados aleatoriamente.

O algoritmo escolhido, o Shellsort, foi criado em 1959 por Donald Shell. Considerando os algoritmos de complexidade quadrática, é o mais eficiente deles. O Shellsort pode ser considerado uma derivação do Insertionsort. Esse algoritmo é diferente do Insertionsort pois ao invés de levar em consideração o vetor a ser ordenado como sendo um único agrupamento de grande extensão, ele considera várias frações da entrada para que seja aplicado o algoritmo Insertionsort, resultando assim num melhor desempenho no tratamento de um grande número de dados.

Cenário 4

Já nessa ocasião, foi verificado o comportamento dos algoritmos para o tratamento de

colisões considerando dois parâmetros de desempenho: o número de comparações de chaves e o gasto de memória RAM.

Os algoritmos de tratamento de colisão que foram usados são os de Endereçamento (Sondagem Linear, Sondagem Quadrática, Duplo Hash), de Encadeamento Separado e de Encadeamento Coalescido (sem porão).

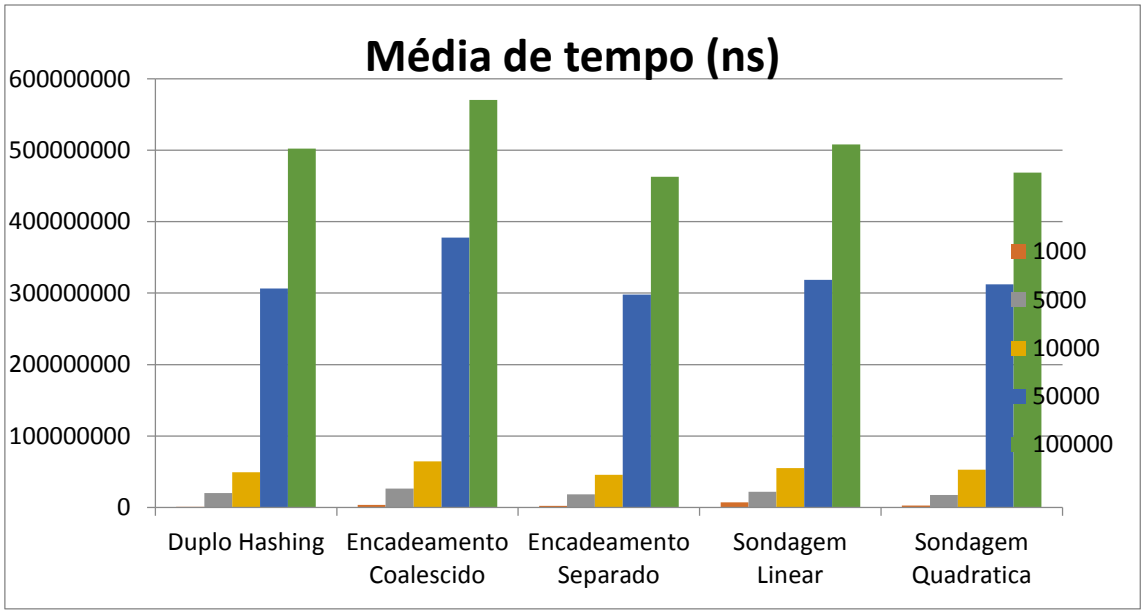
Foram realizados experimentos com os cinco algoritmos considerando vetores gerados com dados aleatoriamente de tamanho N, com N variando em 1000, 5000, 10000, 50000 e 100000. Para cada valor de N, foram realizados experimentos com 10 sementes e avaliados os valores médios do número de comparações de chaves e do gasto de memória RAM. Vale salientar também que cada elemento do vetor possui o ID de cada filme.

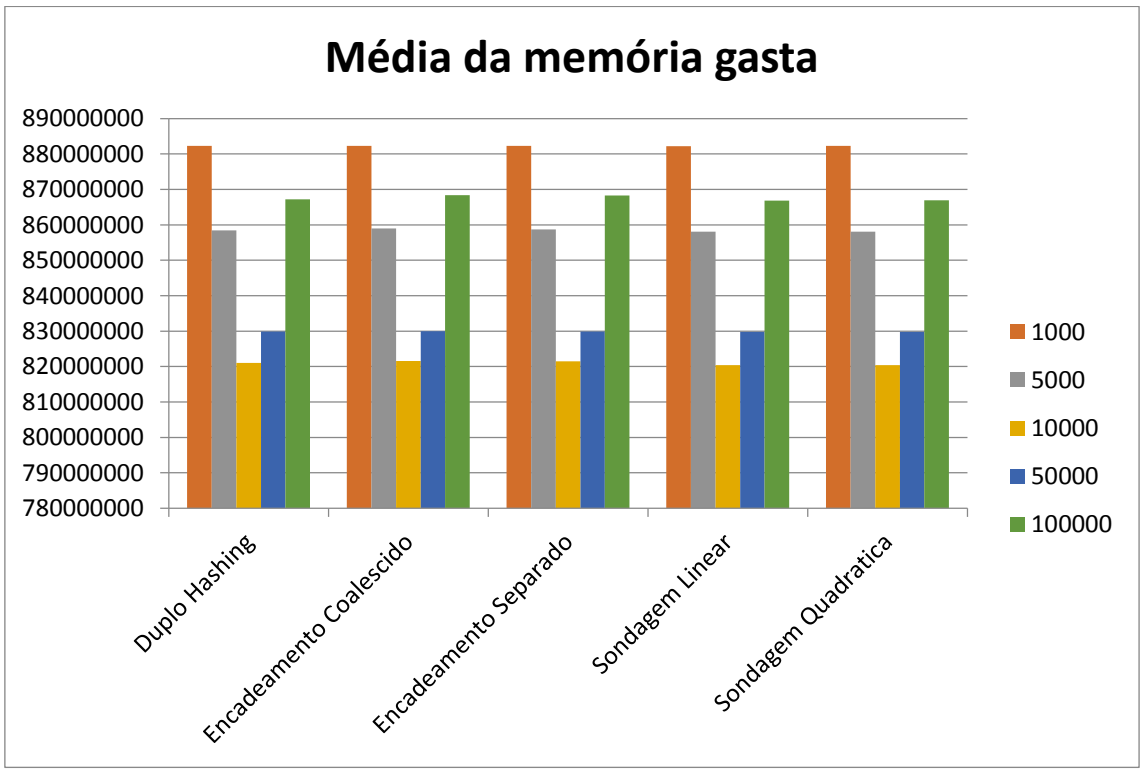
Tempo (ns)					
N	Duplo Hashing	Encadeamento Coalescido	Encadeamento Separado	Sondagem Linear	Sondagem Quadratica
1000	1053050	3439520	1997750	7059199,9	2697170
5000	20148390	26458290	18196070	21722800	17541500
10000	49247110	64415070,1	45872479,9	55154369,9	52687840
50000	306430970	377905970	297920000	318713080	312521230
100000	502139490	570282620	462847730	508236770	468751180

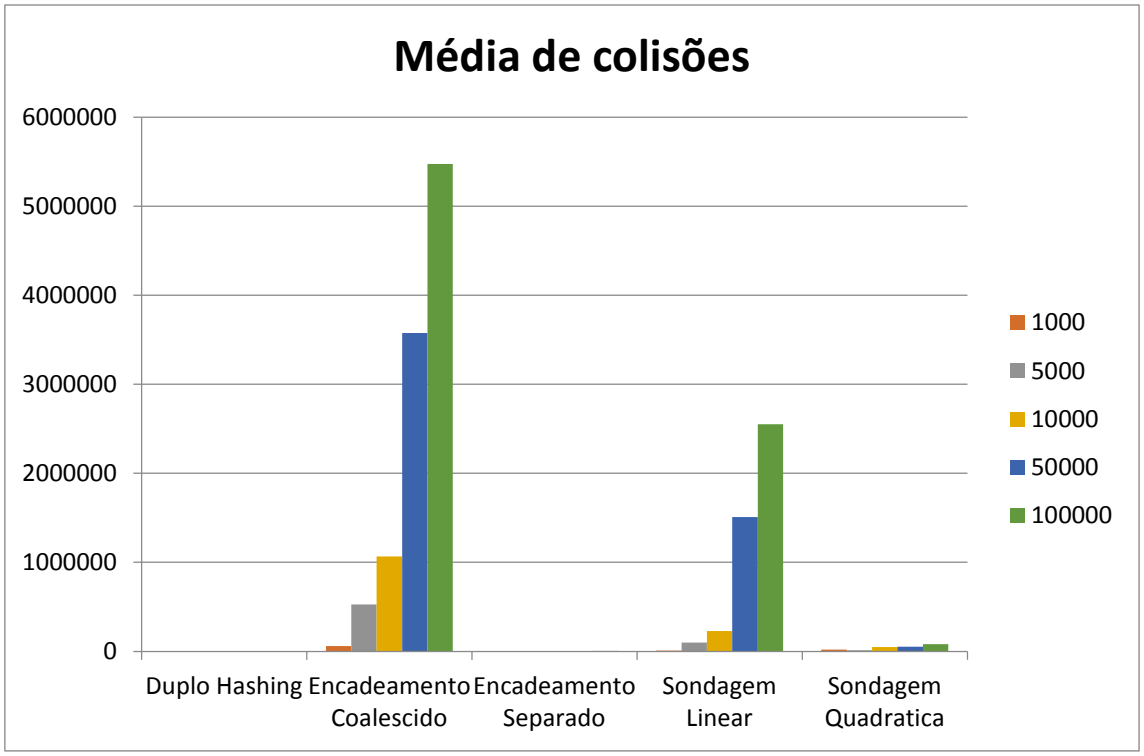
Memória gasta					
N	Duplo Hashing	Encadeamento Coalescido	Encadeamento Separado	Sondagem Linear	Sondagem Quadratica
1000	882265824,8	882267464,8	882265824,8	882162699,2	882265824,8
5000	858399269,6	858990770,4	858724602,4	858117135,2	858118775,2
10000	821030995,2	821555612,8	821485701,6	820395693,6	820430843,2
50000	829859772,8	829961276,8	829926901,6	829768196,8	829768196,8
100000	867166484,8	868411963,2	868299993,6	866807720	866959509,6

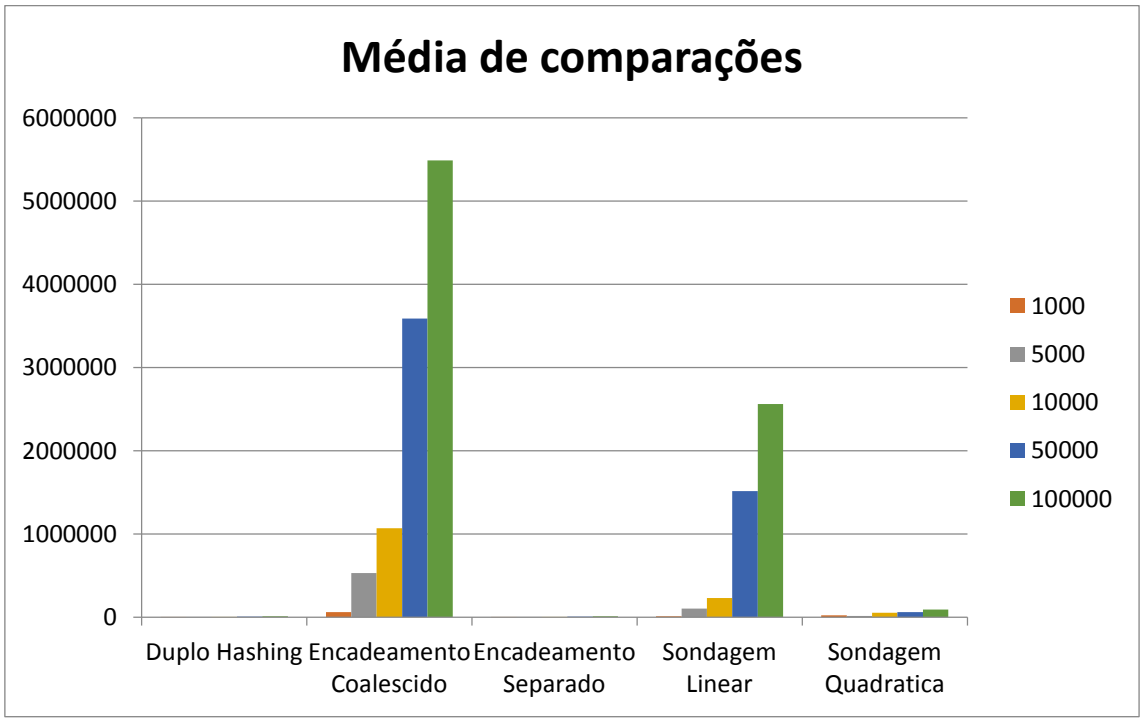
Comparação					
N	Duplo Hashing	Encadeamento Coalescido	Encadeamento Separado	Sondagem Linear	Sondagem Quadratica
1000	779,2	62176,8	779,2	10511,6	20330,1
5000	2411,8	528378,6	2411,8	102086,6	16446,5
10000	3534,2	1068669,6	3534,2	230266,5	51768,5
50000	7361,8	3585459	7361,8	1513951,7	60795,8
100000	9617,8	5485791,6	9617,8	2561402	90947,8

Colisão					
N	Duplo Hashing	Encadeamento Coalescido	Encadeamento Separado	Sondagem Linear	Sondagem Quadratica
1000	0	61208,9	389,1	9732,4	19550,9
5000	0	525420,5	1205,4	99674,8	14034,7
10000	0	1064365,7	1766,6	226732,3	48234,3
50000	0	3576682,2	3680,4	1506589,9	53434
100000	0	5474439,5	4808,4	2551784,2	81330


Média de tempo (ns)
600000000
500000000
400000000
300000000
200000000
100000000
0
Duplo Hashing
Encadeamento Coalescido
Encadeamento Separado
Sondagem Linear
Sondagem Quadratica
1000
5000
10000
50000
100000


Média da memória gasta
890000000
880000000
870000000
860000000
850000000
840000000
830000000
820000000
810000000
800000000
790000000
780000000
Duplo Hashing
Encadeamento Coalescido
Encadeamento Separado
Sondagem Linear
Sondagem Quadratica
1000
5000
10000
50000
100000


Média de colisões
6000000
5000000
4000000
3000000
2000000
1000000
0
Duplo Hashing
Encadeamento Coalescido
Encadeamento Separado
Sondagem Linear
Sondagem Quadratica
1000
5000
10000
50000
100000


Média de comparações
6000000
5000000
4000000
3000000
2000000
1000000
0
Duplo Hashing
Encadeamento Coalescido
Encadeamento Separado
Sondagem Linear
Sondagem Quadratica
1000
5000
10000
50000
100000

Segunda parte

Implementação dos usuários mais ativos

O algoritmo lê os filmes assistidos de todos os usuários e soma todo o número de filmes assistidos individualmente e por usuário. Foi utilizada a tabela de hash para armazenar os usuários e para pesquisar e inserir as tags dos filmes assistidos por cada usuário. Foi implementado uma função hash que suporte chaves que são códigos ASCII. O programa imprime os N primeiros usuários mais ativos, os N usuários que menos assistiram filme, os N primeiros filmes que mais foram vistos e os que menos foram vistos.