

Projeto de Laboratórios de Informática 3

Grupo 20

Diogo Miguel Alves Rocha (A79751) Gabriela Sá Martins (A81987)
Ricardo Milhazes Veloso (A81919) Ricardo Jorge Silva Ferreira (82568)

12 de Junho de 2018

Resumo

O objetivo deste trabalho consistiu em criar um programa que permitisse a extração e análise de dados provenientes da base de dados correspondente ao StackOverflow. Tendo como objetivo final a resposta a um determinado número de queries, em que o tempo de execução será o mais reduzido possível. Esta segunda parte do projeto contém o mesmo objetivo da primeira mas desta feita a linguagem de programação usada é Java em vez de C.

Conteúdo

1	Introdução	2
2	Descrição do Problema	2
3	Concepção da Solução	2
3.1	Porquê da HashTable	2
4	A Estrutura	3
4.1	id_utilizador	3
4.2	id_perguntas	3
4.3	id_respostas	4
4.4	id_tags	4
5	Extração de Informação	5
5.1	Pergunta 1	5
5.2	Pergunta 2	5
5.3	Pergunta 3	5
5.4	Pergunta 4	5
5.5	Pergunta 5	5
5.6	Pergunta 6	5
5.7	Pergunta 7	6
5.8	Pergunta 8	6
5.9	Pergunta 9	6
5.10	Pergunta 10	6
5.11	Pergunta 11	6
6	LI3 em java vs LI3 em C	7
7	Resultados	7
8	Teste de Tempo	7

1 Introdução

O trabalho proposto tinha em vista a resolução de 11 interrogações da forma mais eficiente e rápida possível. Para que conseguíssemos corresponder a estas medidas criamos quatro estruturas, todas elas HashTables .

2 Descrição do Problema

O principal problema era criar uma estrutura capaz de armazenar todos os dados provenientes da base de dados correspondente ao StackOverFlow, de maneira que o acesso a esses mesmos dados fosse o mais rápido possível que por consequência a resposta às interrogações fosse ainda mais rápida, e sem memory leaks.

3 Concepção da Solução

Com vista a resolver o problema decidimos criar 4 HashTables, uma com os dados do utilizador, **id_-utilizador**, outra com os dados referentes aos posts do tipo pergunta **id_perguntas**, outra com os dados referentes aos posts do tipo resposta **id_espostas**, e por fim outra com as tags **id_tags**. Cada uma das anteriores será explicitada numa secção posterior referente às mesmas.

3.1 Porquê da HashTable

O principal motivo da escolha deste tipo de estrutura de dados foi o tempo de procura de um elemento que no melhor caso vai ser sempre constante e é o tipo de estrutura que apresenta o tempo de procura inferior em relação a outros.

Data Structure	Time Complexity			
	Average			
	Access	Search	Insertion	Deletion
Array	$O(1)$	$O(n)$	$O(n)$	$O(n)$
Stack	$O(n)$	$O(n)$	$O(1)$	$O(1)$
Queue	$O(n)$	$O(n)$	$O(1)$	$O(1)$
Singly-Linked List	$O(n)$	$O(n)$	$O(1)$	$O(1)$
Doubly-Linked List	$O(n)$	$O(n)$	$O(1)$	$O(1)$
Skip List	$O(\log(n))$	$O(\log(n))$	$O(\log(n))$	$O(\log(n))$
Hash Table	N/A	$O(1)$	$O(1)$	$O(1)$
Binary Search Tree	$O(\log(n))$	$O(\log(n))$	$O(\log(n))$	$O(\log(n))$
Cartesian Tree	N/A	$O(\log(n))$	$O(\log(n))$	$O(\log(n))$
B-Tree	$O(\log(n))$	$O(\log(n))$	$O(\log(n))$	$O(\log(n))$
Red-Black Tree	$O(\log(n))$	$O(\log(n))$	$O(\log(n))$	$O(\log(n))$
Splay Tree	N/A	$O(\log(n))$	$O(\log(n))$	$O(\log(n))$
AVL Tree	$O(\log(n))$	$O(\log(n))$	$O(\log(n))$	$O(\log(n))$
KD Tree	$O(\log(n))$	$O(\log(n))$	$O(\log(n))$	$O(\log(n))$

Figura 1: timecomplexity

4 A Estrutura

```
private static HashMap<Long,Utilizador> id_utilizador;  
private static HashMap<Long,Pergunta> id_perguntas;  
private static HashMap<Long,Resposta> id_respostas;  
private static HashMap<Long,Tag> id_tags;
```

Figura 2: Estrutura

4.1 id_utilizador

A estrutura contém informações relativas ao **utilizador**, e apresenta 7 parâmetros diferentes:

id_u -ID do utilizador;

name -Nome do utilizador;

shortb -Short bio do utilizador;

totalposts -Número total de posts;

reputacao -Reputação do utilizador;

votes -Número de votes do utilizador;

```
public class Utilizador  
{  
    private long id_u;  
    private String name;  
    private String shortb;  
    private int totalposts;  
    private long reputacao;  
    private int votes;
```

Figura 3: id_utilizador

4.2 id_perguntas

A estrutura contém informações relativas às **Perguntas**, e apresenta 7 parâmetros diferentes:

id_post -ID da pergunta;

id_autor -ID do autor da pergunta;

title -Título da pergunta;

tags -Tag da pergunta;

reputacao -Reputação do utilizador;

n_resp -Número de respostas àquela pergunta;

d -Data da pergunta;

```
public class Pergunta  
{  
    private long id_post;  
    private long id_autor;  
    private String title;  
    private String tags;  
    private long score;  
    private long n_resp;  
    private String d;
```

Figura 4: id_perguntas

4.3 id_respostas

A estrutura contém informações relativas às **Respostas**, e apresenta 6 parâmetros diferentes:

id_post -ID da Resposta;

id_perg -ID da pergunta correspondente à resposta;

id_autor -ID do autor da resposta;

score -Score da resposta;

n_coments -Número de comentários aquela resposta;

d -Data da Resposta;

```
public class Resposta
{
    private long id_post;
    private long id_perg;
    private long id_autor;
    private long score;
    private long n_coments;
    private String d;
```

Figura 5: id_respostas

4.4 id_tags

A estrutura contém informações relativas às **Tags**, e apresenta 3 parâmetros diferentes:

id_tag -ID da tag;

tag -Tag;

count -Conta o número de vezes que uma Tag é utilizada num intervalo de tempo;

```
public class Tag
{
    private long id_tag;
    private String tag;
    private long count;
```

Figura 6: id_tags

5 Extração de Informação

5.1 Pergunta 1

Na pergunta 1 temos de retornar um par com o título do post correspondente ao id recebido, e com o nome do utilizador a que esse post pertence. Para isso acedemos à **id_perguntas** retiramos o título e o nome do autor correspondente ao id recebido e retornamos um par com os respetivos.

5.2 Pergunta 2

Nesta pergunta queremos saber o Top N utilizadores com maior número de posts e para isso consultamos a variável **totalposts**, através do método que implementamos na classe **Utilizador** **getTotalposts()**, após utilizarmos um **ArrayList** **infoPost** que guarda toda a informação existente na **id_utilizador**. De seguida ordenamos esse **ArrayList** pelo número total de posts. E no fim retornamos um **List** com os **id_u** ordenados.

5.3 Pergunta 3

Nesta pergunta queremos obter o número total de posts num determinado periodo de tempo mas com perguntas e respostas em separado. Para isso percorremos a **id_perguntas** e a **id_respostas**, e utilizando o método **compareTo()**, retiramos todas as perguntas e respostas, respetivamente que se encontram entre as datas recebidas. No fim retornamos um par com essas perguntas e respostas.

5.4 Pergunta 4

Nesta pergunta queremos obter o número total de perguntas contendo uma determinada tag num determinado periodo de tempo, e retornar essas perguntas por cronologia inversa. Para isso percorremos a **id_perguntas** e utilizando o método **compareTo** e o método **contains**, retiramos todas as perguntas que se encontram entre as datas recebidas e que contém uma tag igual à recebida, e adicionamos estas perguntas a um **ArrayList** **TagsData**. Em seguida ordenamos este **ArrayList** por data (da mais recente para a mais antiga) utilizando novamente o método **compareTo()**. Por fim adicionamos numa **List** todos os **id_post** que contém aquela tag, retornado essa **List**.

5.5 Pergunta 5

Nesta pergunta queremos devolver a informação do perfil do utilizador (short bio) e os IDs dos seus 10 últimos posts (perguntas ou respostas), ordenados por cronologia inversa. Para isso utilizamos um **Utilizador** **u** para guardar a informação de um utilizador corresponde ao id recebido, depois acedemos a esse **Utilizador** para retirar a shortbio. Em seguida criamos um **ArrayList** **ultimasPerg** e um **ArrayList** **ultimasResp** e utilizamos o método **Long.compare** em ambos os casos para comparar se o id recebido corresponde a uma pergunta ou a uma resposta se algum dos casos se verificar adiciona uma pergunta na **ultimasPerg** ou uma resposta na **ultimasResp**. Em seguida criamos uma **LinkedHashMap** **result** e adicionamos o **id_post** e a data retirados do **ultimasPerg** e do **ultimasResp**. Depois ordenamos esse **LinkedHashMap** pela data. Por fim criamos um **List**(**res**) que recebe todas as keys da **LinkedHashMap** já ordenada e depois adicionamos noutra **List**(**f**) as 10 primeiras keys. E retornamos um par com a shortbio e com a **List** **f**.

5.6 Pergunta 6

Nesta pergunta queremos saber, num intervalo de tempo arbitrário, quais as N respostas com mais votos, devolvendo uma lista com a variável **id_post**. Para isso criamos um **ArrayList**. Em seguida percorremos essa a **id_respostas** e utilizando o método **compareTo** adicionamos à **resp_Data** todas as respostas que se encontrem entre a data recebida. Em seguida ordenamos a **resp_data** através

do Score(por ordem decrescente) , e criamos um **List** res um N id_post . Por fim retorna esse mesmo res.

5.7 Pergunta 7

Nesta pergunta queremos obter os IDs das N perguntas com mais respostas, por ordem decrescente num determinado intervalo de tempo. Para isso consultamos a variável n_resp, através do método que implementamos na classe Pergunta **getN_resp()**, após utilizarmos um **ArrayList<Pergunta>** que guarda toda a informação existente na **id_perguntas** dentro das datas fornecidas. De seguida, ordenamos esse ArrayList de acordo com o número de respostas (por ordem decrescente) e por fim, retornamos um List com os N **id_post** das perguntas com mais respostas ordenado.

5.8 Pergunta 8

Nesta pergunta queremos obter os IDs das N perguntas cujos títulos contém uma palavra específica, ordenados por cronologia inversa. Para isso percorremos a **id_perguntas** e utilizando **compareTo** e o método **contains**. Guardamos todas as perguntas cujos títulos continham a palavra fornecida num **ArrayList<Pergunta>**, e depois ordenamos o mesmo de acordo com a data da pergunta. Por fim, retornamos um List com os N **id_post** das perguntas que continham no título a palavra dada, ordenados por cronologia inversa.

5.9 Pergunta 9

Nesta pergunta recebendo o ID de dois utilizadores devolver as últimas N perguntas que em que participaram esses dois utilizadores. Para isso criamos um **ArrayList** resps e um **ArrayList** perg final e guardamos todas as respostas da id_respostas na resps. Em seguida usamos o método **Long.compare** para comparar o id do autor com o id1 recebido e adicionamos em p1 a pergunta que contenha aquele id. Após isto se o id2 for igual o id_autor adiciona p1 na pergsFinal. Em seguida utiliza o mesmo método mas para a resposta. Depois ordena o pergsFinal por data , cria uma List res e adiciona N id_post a essa lista res. Por fim retorna essa mesma lista.

5.10 Pergunta 10

Nesta pergunta queremos obter a melhor resposta a uma pergunta específica, utilizando a função de média (Score da resposta x 0.45) + (Reputação do utilizador x 0.25) + (número de Votos recebidos pela resposta x 0.2) + (número de comentários recebidos pela resposta x 0.1). Para isso percorremos a **id_respostas** guardando num **ArrayList<Resposta>** as respostas correspondentes à pergunta cujo ID foi dado, através do método **getID_perg()**. Em seguida, é percorrida a **id_utilizador** e são guardados os utilizadores a quem corresponde a resposta. Depois acedemos às variáveis score, n_coments e reputacao através dos métodos **getScore()**, **getN_coments()**, **getReputacao()**, respetivamente, implementadas na classe Utilizador. De seguida são efetuados os cálculos de acordo com a fórmula anteriormente referida e os valores resultantes vão sendo comparados até se obter a melhor resposta e esta ser retornada.

5.11 Pergunta 11

Infelizmente não conseguimos um resultado positivo nesta pergunta.

6 LI3 em java vs LI3 em C

No projeto de LI3 em java chegamos á conclusao que, comparativamente com este em C, este apresenta beneficios ao nivel do tratamento de exceções , uso de memória, compilação do código e a portabilidade. Tratamento de exceções A linguagem java apresenta beneficios ao nivel do tratamento de exceções pois esta admite recursos para o tratamento das mesmas

Uso de memoria Na linguagem java o uso de memoria é mais superficial comparado com a linguagem C. Em C foi necessario o uso de apontadores e definicao da capacidade dos arrays etc. Enquanto que em java nao é necessaria essa preocupacao pois o proprio java aloca o espaço necessario para o carregamento da estrutura que etsmaos a definir O que torna a programacao em java mais facil mas tambem menos controlavel

Utilizacao de bibliotecas Em Java existem bibliotecas em que a ordenacao, insercao em hashtable, tratamento de exceções etc ja estao implementadas e bem implementadas. Enquanto que em C existem menos e a maioria tem de ser escrita o que proporciona uma maior probabilidade de erros e bugs.

7 Resultados

```
Query 1 ->
[What are the actual risks of giving www-data sudo nopasswd access?, WebM(nja)
Query 2 ->
[15811, 449, 158442, 59676, 167858, 367165, 295286, 93977, 35795, -1]
Query 3 ->
[3545, 4812]
Query 4 ->
[276174, 276829, 274462, 274324, 274216, 274141, 274188, 272937, 272813, 272754, 272666, 272565, 272458, 272313, 271816, 271683, 271647, 270853, 270688, 270528, 270488, 270188, 270014, 269876, 269781, 269
995, 268501, 268155, 267746, 267656, 267625, 266742, 266335, 266016, 265531, 265483, 265443, 265347, 265104, 265067, 265028, 264764, 264762, 264616, 264525, 264292, 263816, 263740, 263468, 263405, 263378,
263253, 262733, 262574]
Query 5 ->
<p>Hi, I'm Oli and I'm a "full-stack" web-dev-op. Eurgh. I'm also allergic to jargon BS.</p>
<p>I spend most of my professional time writing Django websites and webapps for SMEs. I write a lot of Python outside of Django sites too. I administer various Linux servers for various tasks. I contribut
e to the open source projects that I use when I can.</p>
<p>I'm a full-time Linux user and that has lead to helping other people live the dream. I am an official Ubuntu Member and I earn my ♦ on SE's own <a href="http://askubuntu.com/users/449/oli">Ask Ubuntu
/a> in <a href="https://askubuntu.com/election/1">2011's moderator election/a>. That's probably where I spend most of my <em>unpaid</em> time.</p>
<p>I also run <a href="http://thepcspy.com" rel="nofollow">thepcspy.com/a> which has been my place to write for the last decade or so.</p>
<p>If you need to contact me for extended help, <a href="https://thepcspy.com/contact/" rel="nofollow">you can do so via my website/a>, just remember that I have bills so if I feel your request is above
and beyond normal duty, I might ask for remuneration for one-on-one support. For more social contact, you can usually find me (or just my computer) lurking in the <a href="https://chat.stackexchange.com/r
oom/281/ask-ubuntu-general-room">Ask Ubuntu General Chat Room/a> and on Freenode in #ubuntu and #ubuntu-uk under the handle Oli or Oli' .</p>
[867, 991, 1401, 1596, 1667, 1763, 3094, 3182, 527591, 5348]
Query 6 ->
[781775, 697197, 694568, 696641, 693722]
Query 7 ->
[505586, 506510, 508221, 505368, 505581, 506824, 508029, 505333, 507139, 508358]
Query 8 ->
[980835, 979082, 974117, 974105, 973832, 971812, 971056, 968451, 964999, 962770]
Query 9 ->
[594, 594, 594, 594, 594, 594, 594, 594, 594]
Query 10 ->
id:5963
Query 11 ->
```

Figura 7: Resultados

8 Teste de Tempo

```
1. INIT -> 0 ms
2. LOAD -> 12869 ms
3. Query 1: -> 1 ms
4. Query 2 -> 148 ms
5. Query 3 -> 65 ms
6. Query 4 -> 38 ms
7. Query 5 -> 113 ms
8. Query6 -> 58 ms
9. Query 7 -> 45 ms
10. Query 8 -> 49 ms
11. Query9 -> 4371 ms
12. Query 10 -> 27 ms
13. Query 11 -> 0 ms
14. CLEAN -> 7 ms
```

Figura 8: tempo

9 Conclusão

Em suma quase todos os objetivos propostos foram atingidos, apenas os resultados pretendidos não foram atingidos na query 5 em que o return não é o correto e na query 11 que não conseguimos criar uma maneira de a resolver. Toda a extração de informação proveniente da base de dados do site **StackOverflow** e análise da mesma é feita de uma maneira eficiente, com um tempo de execução bastante aceitável e sem qualquer perda de informação. Esta segunda fase do projeto foi bastante útil para desenvolver competencias na linguagem de programação java bem como aprender a utilizar a ferramenta **Maven**. Por tudo isto acreditamos que a segunda fase do projeto foi concluída com sucesso.