# TP 2 - Recuperação de Informação

Yuri Diego Santos Niitsuma

O Trabalho consiste em criar um índice invertido para a coleção do TP1.

Este trabalho foi dividido em 3 passos.

 A primeira parte pass1 consiste em criar um vocabulário guardando as strings dos termos no arquivo terms.dump mais as informações de (tamanho, posição) no arquivo em outro arquivo hash\_table.tbl . Ao mesmo tempo que é criado um arquivo de mapeamento (posição/ponteiro) de cada i-ésimo documento no arquivo [nome\_do\_arquivo\_da\_colecao].index e as listas de URLs em

[nome\_do\_arquivo\_da\_colecao].urlist.

Os termos são guardados temporariamente em vários arquivos na pasta terms que será descrito na

seção seguinte.

- A segunda parte pass2 faz a ordenação e a união dos arquivos na pasta terms em um único arquivo terms.index (já ordenado). Nesta etapa o arquivo hash\_table.tbl também
  - é ordenado em ordem alfábetico dos termos para uma busca binária.
- A terceira parte pass3 é o programa que faz as consultas dos termos no índice invertido.
   Digitando /help lista algumas funções disponíveis. Para efetuar uma busca basta digitar os termos separados por espaços. O método utilizado é a busca booleana.

Algumas funções disponíveis são:

- /list: lista todos os termos junto com sua frequência totais
- o /s <termo> OU /seach <termo> : ???
- /quit : fecha o programa

# Execução

Compila em 3 binários pass1.out, pass2.out e pass3.out.

\$ make

Apaga todos os arquivos criados para o índice invertido e os binários executáveis.

\$ make clean

Executa os 3 passos no arquivo html\_pages.txt (estimado 16 minutos utilizando um SSD)

\$ make test

Executa os 3 passos no arquivo ri\_2018\_collection.dat (não faça isso)

```
$ make run
```

Caso deseje evitar repetir os dois primeiros passos e executar diretamente o passo 3. Execute

```
$ ./pass3.out <nome_do_arquivo_da_colecao>
```

# Estrutura dos arquivos

Todos os arquivos são armazenados no formato Endianness da máquina rodada. Nos exemplos seguintes os dados são descritos em Little Endian.

### [nome\_do\_arquivo].urllist

Arquivo contendo lista de URLs coletadas e separados por um '\n'. Será útil no próximo trabalho prático.

### [nome\_do\_arquivo].index

Arquivo que mantém informações das posições da URL do no arquivo **[nome\_do\_arquivo].urllist** e a posição do conteúdo do HTML no arquivo da coleção.

pointer to URL	pointer to HTML content
00 00 00 00 00 00 00	10 00 00 00 00 00 00 00
0D 00 00 00 00 00 00	61 2B 05 00 00 00 00 00

Os ponteiros apontam para a posição no arquivo da coleção utilizada no TP1.

```
|||github.com|<HTML> XXX </HTML>|||youtube.com|<HTML> XXX </HTML>|||
...
showshow.com|<HTML> XXX </HTML>|||
```

O i-ésimo documento é indicado ordenadamente neste arquivo. Iniciando do 0.

### hash\_table.tbl

Estrutura serve pra mapear o ID (hash\_id) e a posição (position) da string do termo com tamanho chars\_length no arquivo terms.dump.

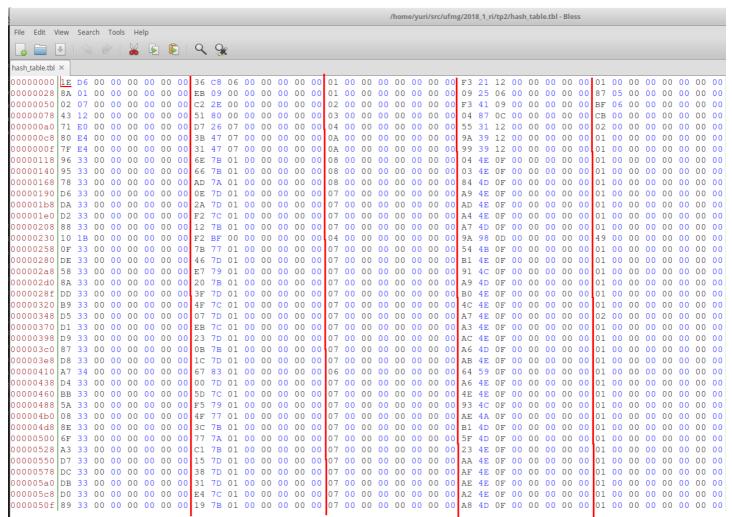
O hash\_id é uma identificação única para cada termo.

```
struct HashBlock {
    size_t hash_id; // unique id of term
    size_t position; // position to char dump
    size_t chars_length; // length of term
    size_t pointer_to_term; // ponteiro para o block do termo
    size_t freq; // freq of term on all documents
};
```

#### Exemplo antes da ordenação

hash_id	position	chars_length	pointer_to_term	freq
0	0	3	0	50
1	3	10	50	11
2	13	6		13
n	position(n)	size(n)		5

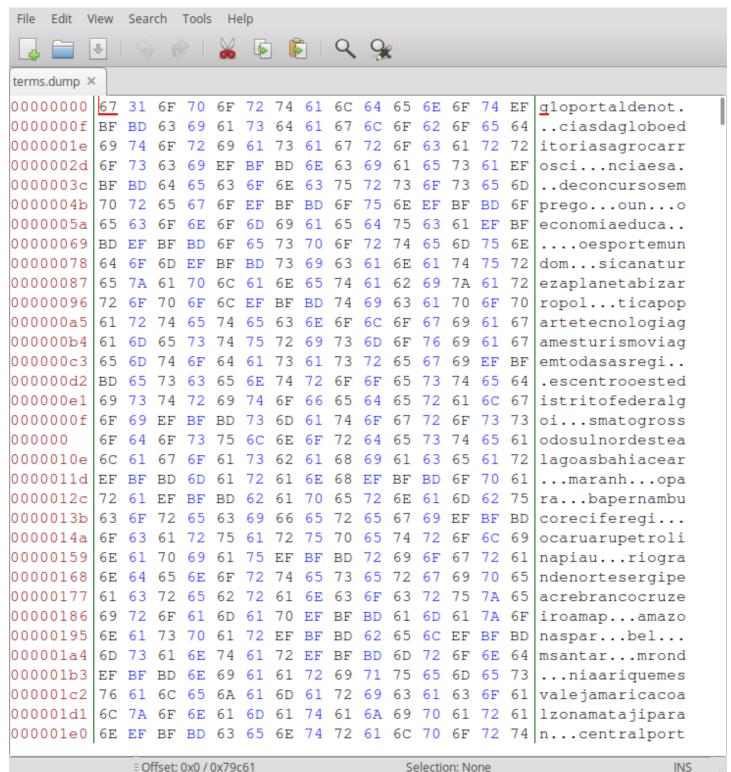
Exemplo após a ordenação, observe que os **hash\_ids** (8 bytes da esquerda) estão aparentemente, aleatórios. Como descrito anteriormente, a ordenação foi alfabética consultando o terms.dump.



#### terms.dump

Contém strings concatenadas para consulta utilizando parâmetros do arquivo hash\_table.tbl.

<term1><term2><term3>...<termN>



### coleção de termos na pasta terms

Esta pasta serve para armazenar a frequência e posição do termo no HTML do arquivo da coleção (não é a posição na coleção).

Ele segue a estrutura:

```
struct {
    size_t hash_id;
    size_t document_id;
    size_t position;
};
```

hash_id	document_id	position
00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00	B1 09 00 00 00 00 00 00
01 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00	B6 09 00 00 00 00 00 00

Cada termo tem um ID único (**hash\_id**) para evitar criar um arquivo e evitar *paddings* dos arquivos no sistema operacional. Para cada termo é dividido em blocos na pasta termos decidido pelo.

```
(hash_id / MANY_ON_DAT_BLOCK)
```

#### MANY\_ON\_DAT\_BLOCK

MANY\_ON\_DAT\_BLOCK é definido no term\_manage.h.

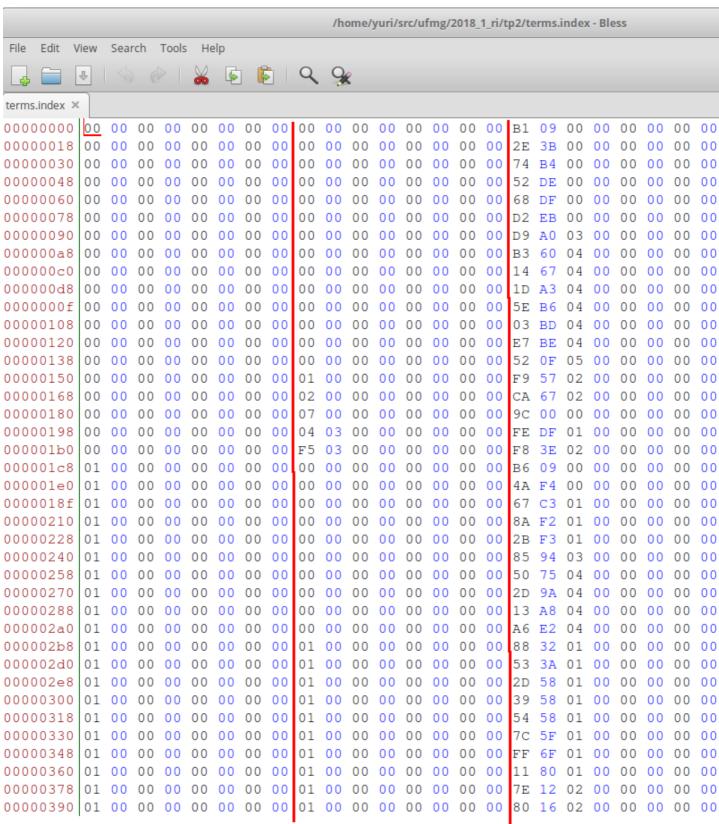
Os primeiros MANY\_ON\_DAT\_BLOCK termos são armazenados no arquivo 0.

Os próximos MANY\_ON\_DAT\_BLOCK no 1 e assim por diante.

Observe que como a varredura segue sequencialmente nos documentos, o document\_id será ordenado nestes arquivos.

Após o primeiro passo, é feito a ordenação em cada arquivo temporário e depois é feito a junção em um único arquivo terms.dump.

A ordenação utilizada nestes arquivos é feito utilizando MergeSort externo interativo.



# Ordenação

A ordenação é feita utilizando apenas o acesso ao disco.

#### Merge Sort Externo

A complexidade O(n log(n)).

Lembrando que cada bloco de termo tem tamanho de 24 bytes.

Alguns testes temporal utilizando a ordenação.

Bytes	Blocos	Тетро
25790232	1074593	117.857 seconds
2747616	114484	9.77569 seconds
1552728	64697	6.3397 seconds
680808	28367	1.99901 seconds
475200	19800	1.49599 seconds
395712	16488	0.920757 seconds
31848	1327	0.0907093 seconds

A ordenação é feita nos arquivos dos termos e no hash\_table.tbl (alfabeticamente).

# **Testes**

### **Tempo**

A criação do índice invertido se mostrou muito lenta no arquivo ri\_2018\_collection.dat, levando 24 horas atingindo apenas 39014 páginas analisadas.

```
0.997680 seconds to read document 39002
        173891 terms added.
           1%
                Position: 2412218691
                                        Document ID: 39003
Progress:
1.169221 seconds to read document 39003
        175132 terms added.
           1%
                Position: 2412277349
                                        Document ID: 39004
Progress:
0.800832 seconds to read document 39004
        173891 terms added.
                Position: 2412307714
                                        Document ID: 39005
Progress:
           1%
0.838978 seconds to read document 39005
        175134 terms added.
                                        Document ID: 39006
Progress:
           1%
                Position: 2412341368
0.765874 seconds to read document 39006
        173891 terms added.
                                        Document ID: 39007
           1%
                Position: 2412382696
Progress:
0.776896 seconds to read document 39007
        175134 terms added.
Progress: 1%
                                        Document ID: 39008
                Position: 2412427357
1.131615 seconds to read document 39008
        173891 terms added.
Progress: 1%
                Position: 2412462672
                                        Document ID: 39009
1.884037 seconds to read document 39009
        175137 terms added.
Progress: 1%
                Position: 2412496451
                                        Document ID: 39010
1.743768 seconds to read document 39010
        173891 terms added.
Progress: 1%
                Position: 2412534511
                                        Document ID: 39011
1.213798 seconds to read document 39011
        175137 terms added.
Progress: 1%
                Position: 2412586920
                                        Document ID: 39012
4.707040 seconds to read document 39012
        173919 terms added.
                                        Document ID: 39013
Progress: 1%
                Position: 2412673693
3.964296 seconds to read document 39013
        175153 terms added.
Progress: 1%
                Position: 2412763821
                                        Document ID: 39014
                                       failed
^CMakefile:19: recipe for target 'run'
make: *** [run] Interrupt
```

#### Os motivos são:

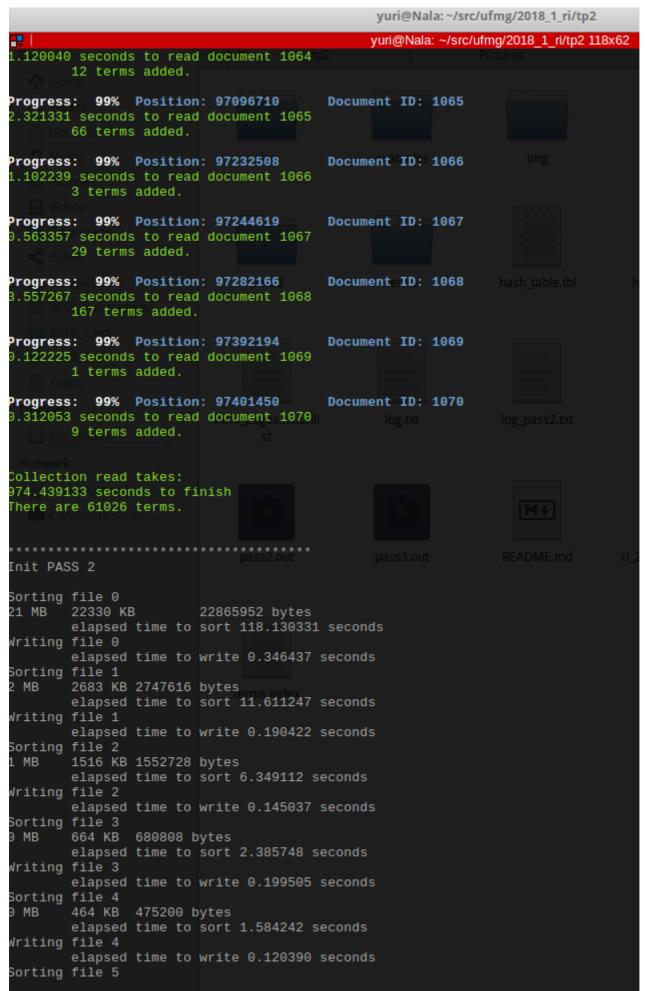
- latência ao acesso do HD externo.
- complexidade de busca no arquivo hash\_table.tbl a cada termo. Se tiver **n** termos já adicionados no hash\_table.tbl e a página provem **m** termos candidatos (que podem ser adicionados ou registrado a frequência), temos uma complexidade *O(nm)* por página.

Um exemplo na tentativa de criar o arquivo invertido no ri\_2018\_collection.dat .

Após de 2.5GBs analisados, o hash\_table.tbl se encontrava com 34MB e cada página levava em

média 3 segundos para ser analisada e aumentando logaritmicamente conforme o arquivo crescia. Tornando inviável coletar quase os 300GBs do arquivo.

O arquivo de testes fornecido html\_pages.txt contendo 93MB gastam 16 minutos para criar o índice invertido.



### **Busca dos termos**

Seguem *screenshots* com exemplos do programa pass3.

#### Busca simples de um termo

```
./pass3.out html_pages.txt
                                                    ./pass3.out html pages.txt 118x62
yuri@Nala ~/src/ufmg/2018_1_ri/tp2 / master • ./pass3.out html_pages.txt
Search terms (Boolean mode) type "/help" for more info.
> /help
Just type your terms separated with ' ' to search the available documents
  Example: apoiam instagram
Commands:
/list
  print all terms available in hashtable file
 search <term> or /s <term>
  Search if term exist in hash table
  Close program
> /s yuri
Found term in position 58859
> /s yuria
Not found!
List of document's ids that found the term(s): <id, freq in doc>
id 3 freq 1
id 498 freq 1
id 696 freq 1
id 772 freq 1
id 891 freq 1
List of document's ids that found the term(s): <id, freq in doc>
id 1 freq 2
id 22 freq 2
id 26 freq 2
id 55 freq 5
id 81 freq 2
id 82 freq 5
id 90 freq 5
id 112 freq 7
id 410 freq 9
id 438 freq 2
id 570 freq
id 617 freq 2
id 630 freq 2
id 650 freq 1
id 719 freq 1
id 746 freq 2
id 894 freq 9
id 956 freq 2
id 997 freq 9
```

#### Busca com 3 termos simultâneos

```
./pass3.out html_pages.txt
                                                  ./pass3.out html pages.txt 118x62
 yuri@Nala > ~/src/ufmg/2018_1_ri/tp2 > 7 master • > ./pass3.out html_pages.txt
Search terms (Boolean mode)
type "/help" for more info.
> windows
List of document's ids that found the term(s): <id/
id 1 freq 4
id 2 freq 6
id 17 freq 3
id 48 freq 2
id 55 freq 1
id 64 freq 10
id 81 freq 1
id 82 freq 1
id 90 freq 1
id 458 freq 1
id 535 freq22
id 617 freq 2
id 623 freq 7
id 719 freq 1
id 894 freq 9
id 997 freq 4
id 1025 freq 1
id 1031 freq 2
> linux
List of document's ids that found the term(s): <id, freq in doc>
id 1 freq 1
id 64 freq 1
id 442 freq 1
id 847 freq 1
> xbox
List of document's ids that found the term(s): <id, freq in doc>
id 1 freq 2
id 22 freq 2
id 26 freq 2
id 55 freq 5
id 81 freq 2
id 82 freq 5
id 90 freq 5
id 112 freq 7
id 410 freq 9
id 438 freq 2
id 570 freq 7
id 617 freq 2
id 630 freq 2
id 650 freq 1
id 719 freq 1
id 746 freq 2
id 894 freq 9
id 956 freq 2
id 997 freq 9
> windows linux xbox
List of document's ids that found the term(s): <id>
```

#### Busca com 2 termos simultâneos

```
./pass3.out html_pages.txt
                                                   ./pass3.out html pages.txt 118x62
                                ri/tp2 / master • ./pass3.out html_pages.txt
 yuri@Nala
Search terms (Boolean mode)
type "/help" for more info.
> xbox
List of document's ids that found the term(s): <id, freq in doc>
id 1 freq 2
id 22 freq 2
id 26 freq 2
id 55 freq 5
id 81 freq 2
id 82 freq 5
id 90 freq 5
id 112 freq 7
id 410 freq 9
id 438 freq 2
id 570 freq
id 617
       freq
            2
id 630
       freq 2
id 650
       freq 1
id
   719 freq
            1
id 746 freq
id 894 freq 9
id 956 freq 2
id 997 freq 9
> windows
List of document's ids that found the term(s): <id, freq in doc>
id 1 freq 4
id 2 freq 6
id 17 freq 3
id 48 freq 2
   55 freq 1
   64 freq 10
id 81 freq
id 82 freq
id 90 freq 1
id 458 freq 1
id 535 freq 2
id 617
       freq
id 623 freq
   719 freq
id 894 freq 9
id 997 freq 4
id 1025 freq 1
id 1031 freq 2
> xbox windows
List of document's ids that found the term(s): <id>
55
81
82
90
617
719
894
997
```