

Universidade Federal de Minas Gerais

Instituto de Ciências Exatas

Departamento de Ciência da Computação

Trabalho Prático 3

Máquina de Busca

Aluno: Giovanni Martins de Sá Júnior

Número de Matrícula: 2017001850

Matéria: DCC004 – Algoritmos e Estruturas de Dados II

2º semestre de 2018

1. Introdução:

- O Trabalho Prático 3 consistiu em implementar um programa que realizasse a pesquisa de várias palavras chave, guardando a frequência e os arquivos onde estas palavras aparecem.
- Além disso, ele também consistiu demonstrar o tempo gasto em procurar uma palavra, bem como em que arquivos ela se faz presente, através do caminho passado via linha de comando.
- Dessa maneira, este trabalho será dividido na construção do indexador em memória e um processador de consultas.

2. Implementação:

- Para a implementação de trabalho. Primeiramente, foram lidos e guardadas todas as palavras referentes ao arquivo StopWord_file.txt. As palavras contidas neste arquivo serão o parâmetro inicial para descartarmos aquelas que aparecem em uma frequência muito grande, tornando-se irrelevantes para a pesquisa.
- Dessa maneira, com base nesse arquivo, foi montada então uma matriz contendo todas as palavras (sem repetições, caracteres indesejados e com todas as letras minúsculas) que seriam utilizadas para a montagem do índice invertido.
- Consequentemente, foi então iniciado a coleta das informações a respeito de cada uma dessas palavras, sendo a frequência e o número de arquivo na qual aparecem. Contudo, para o armazenamento dessas palavras, será necessário utilizar a técnica da função Hash, que irá determinar a posição do vetor estático onde ficarão guardadas as informações associadas de cada palavra.

- Dessa maneira, a função Hash utilizada, foi no somatório do número de caracteres multiplicados pelo seu valor referente a tabela ASCII, divididos por 7. A escolha do sete se deve ao fato de ser um número primo, e relativamente pequeno em relação aos somatórios, o que permite um resultado final com números maiores e mais bem distribuídos, diminuindo as chances de ocorrerem colisões.
- Contudo, pelo grande número de palavras a serem verificadas, possivelmente haverá truncamentos após o cálculo da função Hash. Consequentemente, uma linha irá guardar informações de mais de uma palavra.
- Após a montagem de todas as listas encadeadas, as informações dessas palavras serão organizadas por ordem de frequência. Além disso, também será passado via linha de comando, o caminho de um arquivo que conterá uma palavra. Esta também deverá ser pesquisada e deverá informar em que arquivos ela também estará presente, bem como o tempo de pesquisa levado.

3. Estudo de Complexidade:

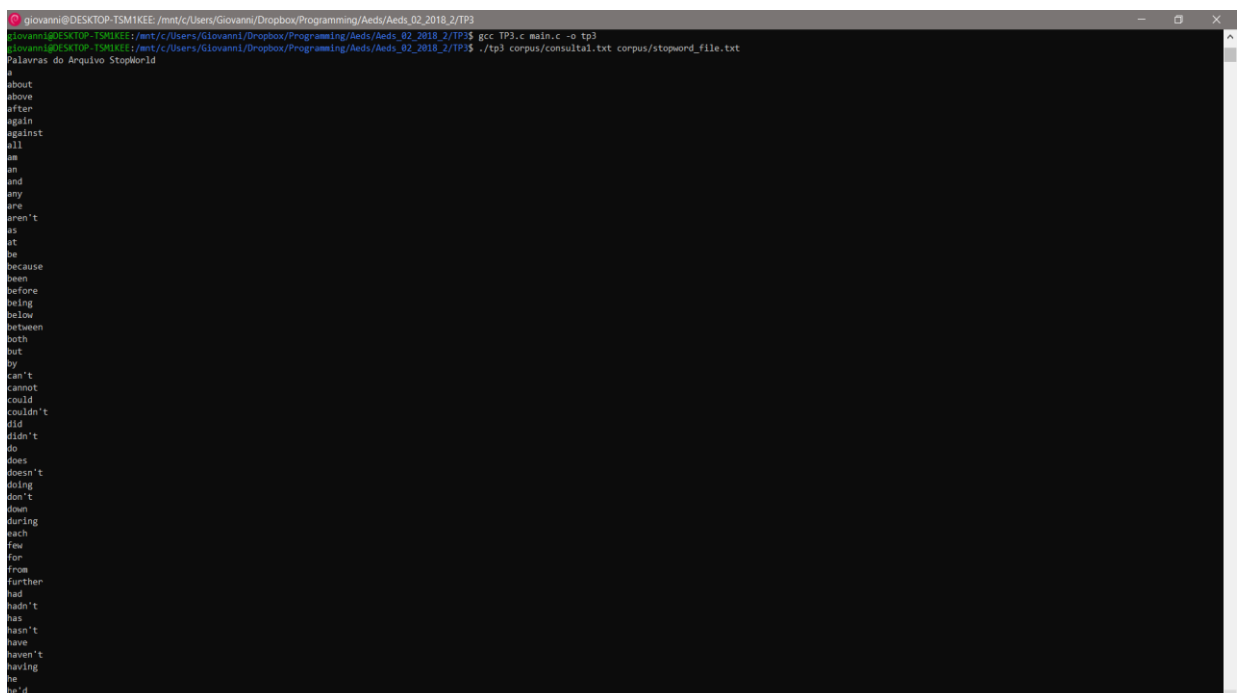
- Nesta seção da documentação, serão declarados o desempenho das funções escritas no trabalho, através do tempo de execução dos procedimentos envolvidos. Para isso, foi utilizada a notação O , adotando um tamanho n para os tabuleiros.
- A seguir serão declaradas das funções, as suas ordens de complexidade, bem como uma breve descrição/justificativa quanto a sua própria ordem de complexidade.
- **Cria Tabela Hash:** $O(1)$. Alocação simples de memória.

- **Lista Vazia:** $O(1)$. Alocação simples de memória e algumas atribuições.
- **Insere Tabela Hash:** $O(1)$. Alocação simples de memória e algumas atribuições.
- **Calcula Pos Hash:** $O(n)$. Atribuições dentro de um laço.
- **Altera Caminho:** $O(1)$. Avalia o número passado como parâmetro na função, altera e concatena as strings.
- **Trata Palavra:** $O(n)$. Dois laços de tamanho variável que removem caracteres indesejados inicialmente presentes no scanf.
- **Is StopWord:** Depende. Caso a palavra avaliada tenha comprimento nulo, ela apenas realiza um return, sendo $O(1)$. Contudo, caso contrário, ela é comparada dentro de um laço em um laço de tamanho equivalente ao número de palavras do arquivo StopWorld, sendo nesse caso, $O(n)$.
- **Is Saved:** $O(n)$. Atribuições e comparações dentro de laços de tamanhos variáveis.
- **Abre StopWord:** Depende. Caso o arquivo não tenha sido encontrado para abertura, é enviada apenas uma mensagem, e então o programa é interrompido, sendo assim, $O(1)$. Caso o arquivo tenha sido aberto com sucesso, dependerá do número(variável) de scanf's realizados, sendo então $O(n)$.
- **Guarda Palavra Matriz:** Depende. Caso o arquivo não tenha sido encontrado para abertura, é enviada apenas uma mensagem, e então o programa é interrompido, sendo assim, $O(1)$. Caso o arquivo tenha sido aberto, do laço de tamanho variável e da chamada de outras funções. Logo, é $O(n^2)$.
- **Frequencia Palavra:** $O(n^2)$. Duas estruturas de repetição enlaçadas.

- **Procura Palavra:** $O(n^2)$. Duas estruturas de repetição enlaçadas.
- **Imprime Tabela Hash:** $O(n)$. Apesar de possuir dois whiles enlaçados, o primeiro é apenas uma condição para a continuidade do laço interior.
- **Saida Arquivo:** $O(n^2)$. Percorre um for, onde cada execução percorre uma lista encadeada, salvando as informações do trabalho no arquivo.
- **Inicia Programa:** $O(n^3)$. Chama várias funções na composição de si própria. Como estas funções, que no pior dos casos são $O(n^2)$, e são chamadas dentro de laços, então esta é uma função $O(n^3)$, **não apresentando um bom desempenho, o que justifica o tempo de sua execução** (na máquina testada, levou cerca de 5 minutos).

4. Testes de Execução:

- Para melhor visualização dos resultados, peço que seja dado um zoom nas fotos abaixo:
- Algumas palavras do arquivo StopWorld, salvas em uma matriz:



```

giovanni@DESKTOP-TSM1KEE: /mnt/c/Users/Giovanni/Dropbox/Programming/Aeds/Aeds_02_2018_2/TP3
giovanni@DESKTOP-TSM1KEE: /mnt/c/Users/Giovanni/Dropbox/Programming/Aeds/Aeds_02_2018_2/TP3$ gcc TP3.c main.c -o tp3
giovanni@DESKTOP-TSM1KEE: /mnt/c/Users/Giovanni/Dropbox/Programming/Aeds/Aeds_02_2018_2/TP3$ ./tp3 corpus/consultal.txt corpus/stopword_file.txt
Palavras do Arquivo StopWorld
a
about
above
after
again
against
all
am
an
and
any
are
aren't
as
at
be
because
been
before
being
below
between
both
but
by
can't
cannot
could
couldn't
did
didn't
do
does
doesn't
doing
don't
down
during
each
few
for
from
further
had
hadn't
has
hasn't
have
haven't
having
he
h'd

```

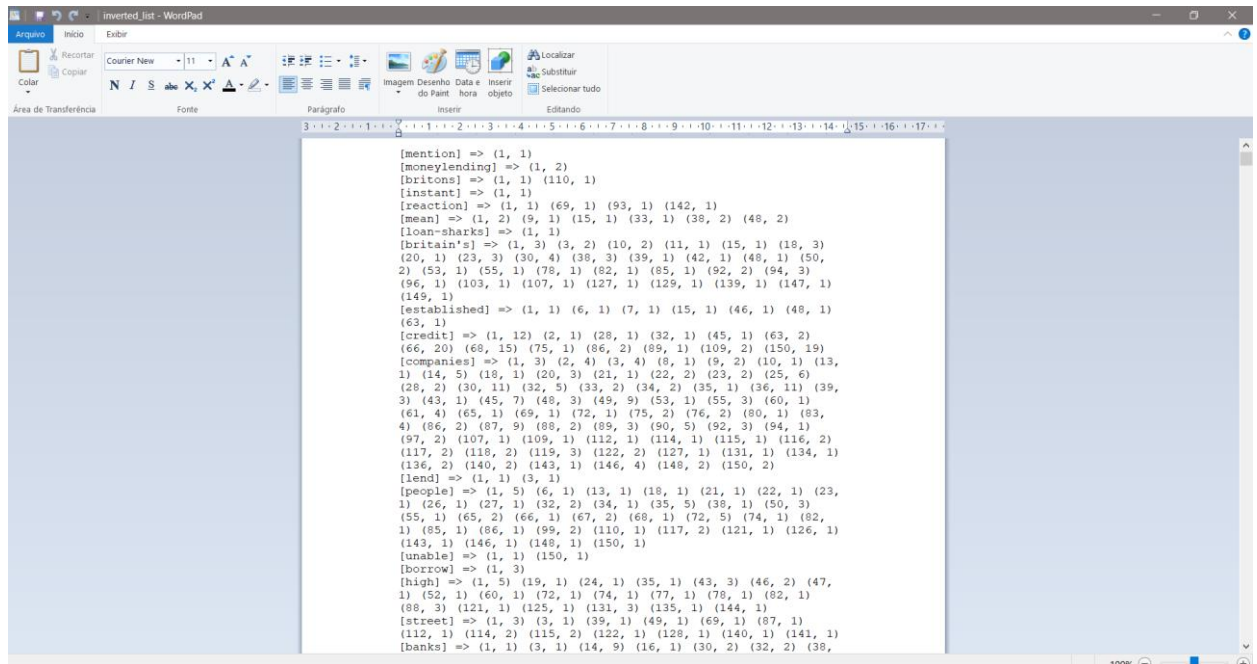
- Algumas palavras de todo o índice invertido, também salvas em uma matriz:

```
giovanni@DESKTOP-TSM1KEE:/mnt/c/Users/Giovanni/Dropbox/Programming/Aeds/Aeds_02_2018_2/TP3$ gcc TP3.c main.c -o tp3
giovanni@DESKTOP-TSM1KEE:/mnt/c/Users/Giovanni/Dropbox/Programming/Aeds/Aeds_02_2018_2/TP3$ ./tp3 corpus/consultat.txt corpus/stopword_file.txt
Palavras do Índice Invertido:
[0]: mention
[1]: moneylending
[2]: britons
[3]: instant
[4]: reaction
[5]: mean
[6]: loan-sharks
[7]: britain's
[8]: established
[9]: credit
[10]: companies
[11]: lend
[12]: people
[13]: unable
[14]: borrow
[15]: high
[16]: street
[17]: banks
[18]: card
[19]: say
[20]: public
[21]: image
[22]: menacing
[23]: debt
[24]: collectors
[25]: far
[26]: cry
[27]: conduct
[28]: regulated
[29]: business
[30]: collector
[31]: someone
[32]: relationship
[33]: customer
[34]: totally
[35]: different
[36]: process
[37]: says
[38]: eddie
[39]: cran
[40]: chief
[41]: executive
[42]: financial
[43]: services
[44]: group
[45]: cattle's
[46]: runs
[47]: door-to-door
[48]: moneylenders
[49]: shopcheck
[50]: firms
[51]: tied
```

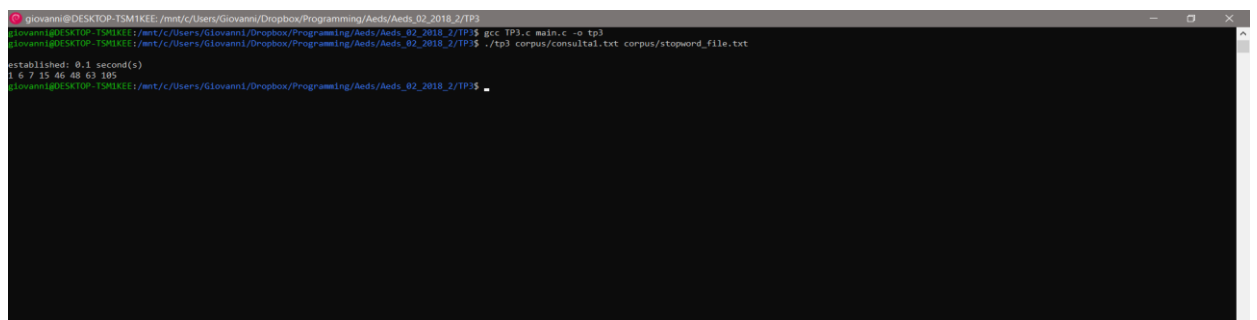
- Palavras do índice invertido, acompanhadas de suas frequências:

```
giovanni@DESKTOP-TSM1KEE:/mnt/c/Users/Giovanni/Dropbox/Programming/Aeds/Aeds_02_2018_2/TP3$ gcc TP3.c main.c -o tp3
giovanni@DESKTOP-TSM1KEE:/mnt/c/Users/Giovanni/Dropbox/Programming/Aeds/Aeds_02_2018_2/TP3$ ./tp3 corpus/consultat.txt corpus/stopword_file.txt
mention -> (1, 1)
moneylending -> (1, 2)
britons -> (1, 1) (110, 1)
instant -> (1, 1)
reaction -> (1, 1) (69, 1) (93, 1) (142, 1)
mean -> (1, 2) (9, 1) (15, 1) (33, 1) (38, 2) (48, 2)
loan-sharks -> (1, 1)
britain's -> (1, 3) (3, 2) (18, 2) (11, 1) (15, 1) (18, 3) (20, 1) (23, 3) (30, 4) (38, 3) (39, 1) (42, 1) (48, 1) (50, 2) (53, 1) (55, 1) (78, 1) (82, 1) (85, 1) (92, 2) (94, 3) (96, 1) (103, 1) (107, 1) (127, 1) (129, 1) (139, 1) (147, 1) (149, 1)
established -> (1, 1) (6, 1) (7, 1) (15, 1) (46, 1) (48, 1) (63, 1)
credit -> (1, 12) (2, 1) (28, 1) (32, 1) (45, 1) (63, 2) (66, 20) (68, 15) (75, 1) (86, 2) (89, 1) (109, 2) (150, 19)
companies -> (1, 3) (2, 4) (3, 4) (8, 1) (9, 2) (10, 1) (13, 1) (14, 5) (18, 1) (28, 3) (21, 1) (22, 2) (23, 2) (25, 6) (28, 2) (30, 11) (32, 5) (33, 2) (34, 2) (35, 1) (36, 11) (39, 3) (43, 1) (45, 7) (48, 3) (49, 9) (53, 1) (55, 3) (58, 1) (61, 4) (65, 1) (69, 1) (72, 1) (75, 2) (76, 2) (80, 1) (83, 4) (86, 2) (87, 9) (88, 2) (89, 3) (90, 5) (92, 3) (94, 1) (97, 2) (107, 1) (109, 1) (112, 1) (114, 1) (115, 1) (116, 2) (117, 2) (118, 2) (119, 3) (122, 2) (127, 1) (131, 1) (134, 1) (136, 2) (140, 2) (143, 1) (146, 4) (148, 2) (150, 2)
lend -> (1, 1) (3, 1)
people -> (1, 5) (6, 1) (13, 1) (18, 1) (21, 1) (22, 1) (23, 1) (26, 1) (27, 1) (32, 2) (34, 1) (35, 5) (38, 1) (50, 3) (55, 1) (65, 2) (66, 1) (67, 2) (68, 1) (72, 5) (74, 1) (82, 1) (85, 1) (86, 1) (99, 2) (110, 1) (117, 2) (121, 1) (126, 1) (143, 1) (146, 1) (148, 1) (150, 1)
unable -> (1, 1) (150, 1)
borrow -> (1, 3)
high -> (1, 5) (19, 1) (24, 1) (35, 1) (43, 3) (46, 2) (47, 1) (52, 1) (60, 1) (72, 1) (74, 1) (77, 1) (78, 1) (82, 1) (88, 3) (121, 1) (125, 1) (131, 3) (135, 1) (144, 1)
street -> (1, 3) (3, 1) (39, 1) (49, 1) (69, 1) (87, 1) (112, 1) (114, 2) (115, 2) (122, 1) (128, 1) (140, 1) (141, 1)
banks -> (1, 1) (5, 1) (14, 9) (16, 1) (40, 2) (42, 2) (48, 1) (48, 4) (50, 4) (53, 1) (61, 5) (63, 3) (66, 9) (68, 4) (69, 4) (74, 1) (79, 3) (83, 1) (101, 1) (104, 3) (108, 8) (109, 1) (112, 11) (114, 9) (115, 12) (119, 3) (120, 13) (122, 10) (125, 1) (126, 1) (128, 4) (130, 9) (141, 2) (146, 3) (150, 5)
card -> (1, 1) (2, 1) (28, 1) (45, 1) (75, 1) (86, 2) (89, 1)
say -> (1, 1) (5, 2) (9, 2) (21, 1) (25, 1) (30, 1) (33, 2) (40, 1) (48, 3) (69, 1) (70, 1) (71, 1) (73, 1) (86, 1) (88, 1) (93, 1) (108, 1) (121, 5) (123, 1) (125, 2)
public -> (1, 1) (8, 1) (26, 1) (28, 2) (34, 1) (40, 1) (49, 1) (55, 3) (60, 1) (63, 1) (67, 1) (75, 1) (87, 1) (90, 1) (99, 1) (101, 1) (109, 1) (119, 1) (121, 1) (126, 1) (128, 2) (137, 1) (141, 1) (148, 1)
image -> (1, 1)
menacing -> (1, 1)
debt -> (1, 1) (16, 3) (20, 2) (39, 2) (60, 1) (74, 3) (83, 3) (101, 1) (120, 1) (121, 5) (140, 1)
collectors -> (1, 2) (5, 1)
far -> (1, 1) (2, 1) (6, 1) (7, 2) (14, 1) (18, 1) (19, 1) (21, 1) (31, 1) (32, 1) (36, 1) (38, 2) (43, 1) (50, 1) (51, 1) (52, 1) (53, 1) (60, 1) (73, 1) (82, 1) (92, 1) (97, 1) (107, 1) (110, 1) (117, 1) (119, 1) (123, 1) (124, 2) (127, 1) (129, 2) (131, 1) (135, 1) (140, 1)
cry -> (1, 1) (110, 1)
conduct -> (1, 2)
regulated -> (1, 1) (24, 1) (34, 1) (69, 1) (128, 1) (148, 1)
business -> (1, 6) (2, 2) (3, 3) (5, 2) (10, 1) (15, 1) (18, 1) (20, 1) (22, 1) (23, 1) (26, 1) (27, 3) (28, 1) (30, 1) (35, 1) (36, 2) (38, 5) (39, 1) (45, 1) (46, 1) (48, 2) (50, 1) (53, 2) (62, 2) (65, 1) (72, 1) (73, 1) (75, 2) (78, 2) (80, 1) (82, 1) (83, 2) (85, 1) (86, 1) (89, 2) (94, 1) (101, 3) (107, 4) (108, 1) (112, 1) (114, 1) (115, 1) (116, 1) (126, 3) (127, 4) (134, 1) (140, 1) (141, 1) (146, 1)
collector -> (1, 1)
someone -> (1, 1) (26, 1) (67, 1) (69, 1) (88, 1) (99, 1) (108, 1)
relationship -> (1, 1) (97, 1)
customer -> (1, 4) (13, 1) (20, 1) (23, 1) (39, 1) (83, 1) (84, 1) (85, 2) (101, 1) (126, 1) (134, 1) (140, 1) (143, 1)
totally -> (1, 2) (15, 1) (97, 1)
different -> (1, 1) (5, 2) (8, 1) (10, 1) (13, 1) (20, 1) (23, 1) (34, 1) (39, 1) (44, 2) (66, 1) (78, 2) (83, 1) (84, 1) (109, 1) (121, 1) (130, 2) (140, 1) (143, 1) (145, 2) (148, 1) (150, 2)
process -> (1, 1) (2, 1) (9, 2) (15, 1) (33, 2) (48, 2) (89, 1) (100, 1) (119, 1) (132, 1)
says -> (1, 10) (3, 1) (8, 1) (50, 1) (72, 1) (75, 1) (108, 1) (110, 3) (129, 1) (139, 1)
eddie -> (1, 1)
cran -> (1, 5)
chief -> (1, 1) (2, 1) (10, 2) (13, 1) (16, 1) (17, 1) (20, 1) (21, 2) (22, 1) (23, 1) (39, 1) (42, 1) (44, 1) (46, 1) (49, 1) (53, 2) (55, 1) (64, 2) (76, 1) (78, 1) (83, 2) (85, 1) (86, 1) (91, 1) (92, 1) (94, 1) (101, 1) (107, 2) (116, 1) (118, 1) (126, 1) (127, 2) (134, 1) (136, 1) (140, 2) (141, 2) (143, 1) (145, 1) (147, 1)
executive -> (1, 1) (2, 1) (10, 2) (13, 1) (16, 1) (15, 1) (21, 2) (22, 1) (23, 2) (24, 1) (44, 1) (45, 1) (46, 1) (49, 1) (50, 1) (53, 3) (55, 5) (64, 2) (66, 1) (69, 1) (70, 2) (75, 1) (78, 1) (83, 1) (85, 1) (86, 2) (89, 1) (92, 1) (94, 2) (101, 1) (107, 3) (110, 1) (116, 1) (117, 1) (119, 1) (126, 1) (127, 3) (134, 1) (137, 1) (139, 1) (140, 1) (143, 1) (145, 1)
financial -> (1, 5) (3, 1) (10, 1) (14, 8) (18, 2) (20, 1) (21, 1) (23, 2) (24, 2) (30, 3) (32, 7) (38, 4) (39, 1) (43, 1) (48, 3) (50, 2) (55, 1) (60, 1) (61, 1) (64, 1) (69, 6) (70, 1) (74, 3) (78, 1) (79, 1) (83, 1) (85, 1) (94, 1) (104, 1) (109, 1) (112, 3) (114, 2) (115, 3) (119, 11) (122, 1) (125, 1) (128, 2) (129, 1) (131, 1) (134, 1) (140, 1) (141, 1) (146, 7)
```

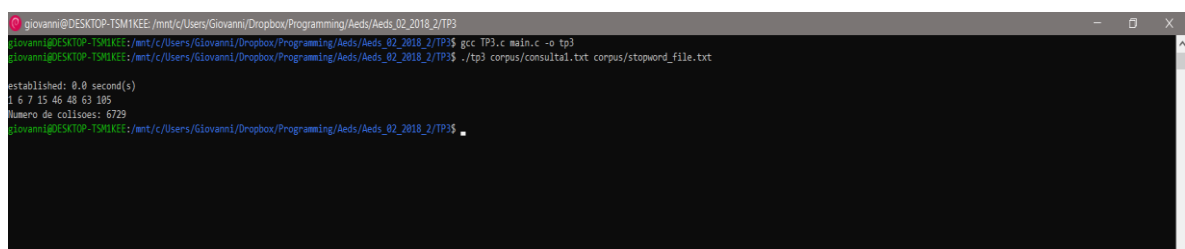
- Palavras do índice invertido, acompanhadas de suas frequências em um arquivo de saída, demonstradas no WordPad:



- Pesquisa da palavra contida no arquivo de argv[1]:



- Número de colisões:



5. Conclusão:

- Apesar do cronograma ter sido extremamente apertado e ter sido entregue com atraso, foi possível concluir a maioria dos requisitos pedidos pelo trabalho. Contudo, a única parte faltante foi ordenar, por ordem de frequência, os arquivos que contêm as palavras buscadas.
- Na minha opinião, os dois grandes desafios dessa atividade foi conseguir organizar os dados a serem lidos e posteriormente comparados. Explicando de uma maneira mais aprofundada, foi necessário comparar quase 7000 palavras (sem repetições, já tratadas e fora do arquivo StopWord), em 150 arquivos diferentes, contendo centenas ou milhares de palavras, resultando em milhões de comparações.
- Por conta de todo esse volume de comparações, a geração do arquivo de saída demanda um tempo considerável até a sua completa execução. No computador utilizado para execução e compilação dos arquivos, o tempo total foi próximo dos 5 minutos até sua execução completa.
- Por fim, mesmo com todas as dificuldades envolvidas no trabalho, foi satisfatório poder aplicar os conceitos mais importantes envolvidos na disciplina.

6. Bibliografia:

- Linguagem C: Completa e Descomplicada – André Backes
- Projeto de Algoritmos com Implementação em Pascal e C – Nívio Ziviani