UNIVERSIDADE FEDERAL DE PERNAMBUCO – UFPE
CENTRO DE INFORMÁTICA – CIn

Relatório do projeto da disciplina: Processamento de Cadeias de Caracteres (2015.2)

Equipe:
João Guilherme Farias Duda
Paulo de Barros e Silva Filho
Raul Maia Falcão

Recife, 10 de Janeiro de 2016

Conteúdo

1 Introdução

Este documento é sobre a ferramenta *ipmt*. Essa ferramenta é capaz de pré-processar um arquivo de texto, gerando um índice. Sucessivas buscas podem ser feitas através desse índice, sem a necessidade de percorrer o texto novamente.

ipmt primeiro gera um índice para o texto usando o algoritmo LSA (Linear Suffix Array), depois esse índice é comprimido em um arquivo juntamente com o texto usando o algorítmo LZ78. Para realizar buscas, primeiramente o arquivo é descomprimido usando o lz78-decode e depois o casamento de padrões é realizada de acordo com o LSA.

Os integrantes da equipe foram responsáveis pelas seguintes tarefas:

- João:
- Paulo: Implementação do algorítmo LZ78 e da interface de comunicação entre os algoritmos.
- Raul: Implementação da estrutura de indexação LSA.

2 Descrição de uso da ferramenta *ipmt*

O projeto contém um arquivo Makefile. Após a execução do comando make, o executável *ipmt* será gerado no diretório bin. A ferramenta *ipmt* possui 4 modos de execução:

- Modo de indexação - *ipmt* index file.txt

  O comando acima irá criar o arquivo file.idx, que contém o conteúdo de file.txt e que possibilita a realização de buscas.

- Modo de busca - *ipmt* search -c herself file.idx

  O comando acima irá listar a quantidade de ocorrência do padrão "herself"encontradas no arquivo indexado file.idx. O argumento -c é opcional, caso não informado todas as linhas contendo ocorrências serão impressas.

- Modo de compressão - *ipmt* compress file.txt

  O comando acima irá comprimir o arquivo file.txt em um arquivo file.comp.

- Modo de descompressão - *ipmt* decompress file.comp

  O comando acima irá descomprimir o arquivo file.comp em um arquivo file.comp.decomp.

3 Implementação

[[Add some intro]]

3.1 Descrição dos algoritmos implementados

3.1.1 Linear Suffix Array

O algoritmo de indexação implementado teve como base [**?**] para a construção em tempo linear de um array de sufixos. Em suma, o array de sufixos é um array de inteiros que armazena a permutação de n índices ordenados lexicograficamente, onde n é o tamanho do texto. Uma vez construído o array de sufixos, a complexidade da busca passa a ser linear com relação ao tamanho do padrão.

3.1.2 Linear Suffix Tree

O algoritmo de indexação implementado teve como base [**?**] para a construção em tempo linear de uma árvore de sufixos. A estrutura implementada representa todos os sufixos de uma cadeia. A implementação contém alguns truques para que a construção seja feita em tempo linear. Um desses truque é adicionar aos nós suffix links, também chamados como transições de falha ou fronteiras. Devido ao alto consumo de memória ao gerar a árvore de sufixo, resolvemos deixar a feature de melhorar o gerenciamento de memória para o futuro. Segundo [**?**] se o núcleo da implementação for orientada a objeto, a árvore de sufixo apresenta efeitos indesejáveis de memória fragmentada.

3.1.3 LZ78

[[Add text]]

3.2 Detalhes de implementação

3.2.1 Linear Suffix Array

Na construção do Linear Suffix Array há uma etapa de criação de dois arrays de sufixos, S1 e S2. Seja index a posição de um caracter em um texto: A

função buildS1andS2 constrói o array de sufixo S1 que contém sufixos tal que index % 3 = 0 e também constrói o array de sufixo S2 que contém sufixos tal que index % 3 != 0. Após a construção de S1 e S2, estes são ordenados através de uma implementação do Radix Sort com o objetivo de otimizar essa etapa. Como o Radix Sort não faz comparações entre valores, nesse contexto, o seu desempenho é superior a um algoritmo de ordenação por comparação. A ordenação de S1 e S2 foi necessária para a etapa de merge (S1 U S2 = SA) de tal forma que o custo do merge é realizado em tempo linear. Após o merge, obtemos os índices devidamente ordenados.

3.2.2 LZ78

[[Add text]]

3.3 Descrição do formato .idx

A ferramenta ipmt gera e lê arquivos no formato .idx. Esse arquivo é gerado da seguinte forma para um arquivo de entrada file.txt:

- Primeiramente é gerado o Linear Suffix Array a partir do conteúdo de file.txt.
- Após isso, conta-se o número de linhas de file.txt.
- É criado um novo arquivo com o seguinte contéudo:

```
[Número de linhas contidas em file.txt]
[Conteúdo de file.txt]
[Elementos do LSA separados por um espaço]
```

  `(Note que quebras de linha separam os elementos acima.)`

- Esse novo arquivo é então comprimido usando o LZ78, gerando o arquivo file.idx.

Na hora de ler o arquivo .idx, primeiro é realizada a descompressão. Através do resultado, a ferramenta sabe que a primeira linha contém o número de linhas do texto. Logo, as linhas seguintes são referentes ao LSA, que é utilizado pela busca.

4 Experimentos

Realizamos experimentos para responder às seguintes perguntas:

1. Como a nossa implementao do LZ78 se compara ao gzip?
2. Como a etapa de busca de padrão do *ipmt* se compara ao grep?
3. Como a nossa implementao do *ipmt* se compara ao codesearch [**?**]?

Foram implementados scripts em BASH para controlar os experimentos e fazer medições, gerando arquivos .raw de saída contendo resultados. Foram também implementados scripts em R que desenham gráficos de acordo com os arquivos .raw gerados pelos scripts BASH.

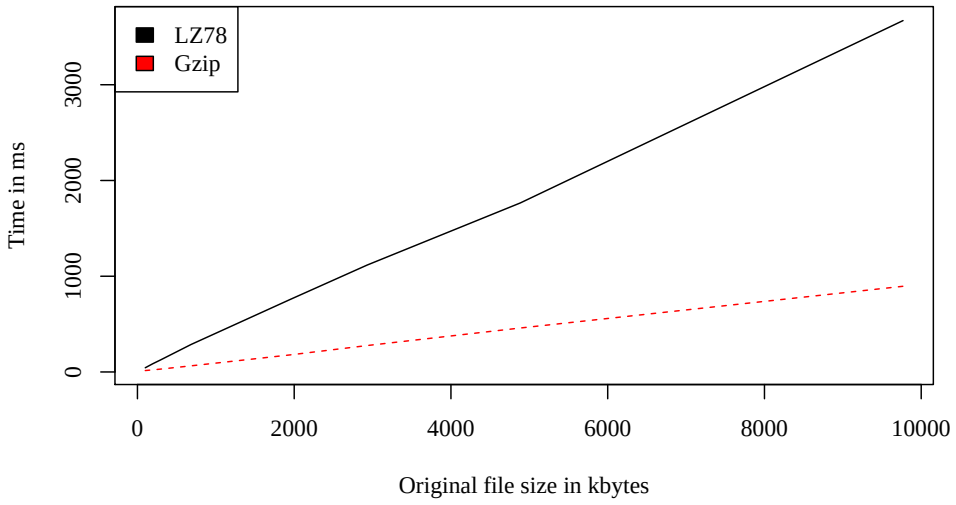
Todos os experimentos foram realizados em uma máquina com processador Intel Core i5 2.6Ghz e 8Gb de RAM. Cada medição de tempo nos experimentos foi realizada 10 vezes, e somente a média foi considerada e reportada nos resultados. Todos os scripts e resultados estão disponíveis no diretório experiments.

4.1 Como a nossa implementao do LZ78 se compara ao gzip?

Nós comparamos a nossa implementação do LZ78 com o gzip em dois aspectos: Tempo e taxa de compressão. Para realizar essa comparação, nós escolhemos 11 arquivos de tamanhos distintos: 100KB, 200KB, 300KB, 700KB, 1MB, 2MB, 3MB, 5MB, 50MB e 1GB. Todos os arquivos continham somente textos em inglês.

4.1.1 Tempo

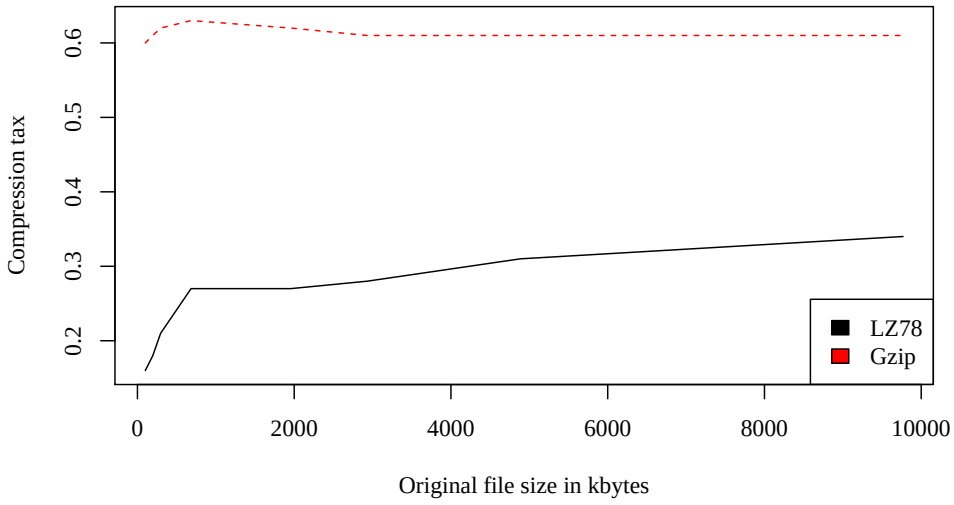
Abaixo está um gráfico que relaciona o tempo que leva para comprimir um arquivo com o tamanho dele, para ambos o nosso LZ78 e o gzip.



Ambas as funções são (ou se aproximam muito de) retas, o que comprova que a nossa implementação do LZ78, bem como o gzip, acontecem em tempo linear de acordo com o tamanho do arquivo. Porém, a constante que multiplica a função do gzip é bem menor do que a nossa. Isso acontece porque o gzip está em desenvolvimento a mais de 23 anos, onde experts estão sempre otimizando o algoritmo, fazendo com que essa constante da função linear seja cada vez menor.

4.1.2 Taxa de compressão

Abaixo está um gráfico que relaciona a taxa de compressão de um arquivo com o tamanho dele, para ambos o nosso LZ78 e o gzip.


Compression tax
0.2
0.3
0.4
0.5
0.6
0
2000
4000
6000
8000
10000
Original file size in kbytes
LZ78
Gzip