Departamento de Informática

Mestrado Integrado em Engenharia Informática

Comunicação de Dados

Trabalho Prático: Implementação da Ferramenta Shafa

Realizado por:

(Rodrigo Rodrigues A93201

Rui Monteiro A93179

Rui Moreira A93232

Bernardo Saraiva A93189

André Presa A82917)

Universidade do Minho

Braga, 03 de Janeiro de 2021

Rodrigo Rodrigues

Rui Monteiro

Bernardo Saraiva

Rui Moreira

André Presa

Índice

Organização do grupo em torno do trabalho desenvolvido

A ferramenta a desenvolver é constituída por quatro módulos independentes. A fim de dividir tarefas, o grupo de 8 elementos foi subdividido em 4 grupos de dois elementos e foi atribuído a cada grupo um módulo do trabalho prático. Eis os grupos e os respetivos módulos:

Módulo A: Rodrigo Rodrigues (A93201) & Rui Monteiro (A93179)

- Compressão RLE
- Cálculo das Frequências dos símbolos

Módulo B: Rui Moreira (A93232) & Bernardo Saraiva (A93189)

- Tabelas de codificação
- Representação dos códigos no ficheiro cod

Módulo C: André Gil & André Presa

- Construir a sequência codificada de bits
- Otimização da codificação binária SF com matrizes de bytes

Módulo D: Bruno Silva & Ricardo Veloso

- Descodificar a sequência comprimida de bits
- Otimização da descodificação binária SF

Nota: Módulo **D** não foi entregue devido ao incumprimento da sua realização por parte do subgrupo encarregue. Módulo **C** foi entregue apenas por um dos membros (André Presa).

Estratégias e Mecanismos adotados

- ## Módulo A

Armazenamento dos dados do ficheiro de entrada num Buffer em memória que consiste num array dinâmico, de modo a reduzir a complexidade e aumentar a eficiência do processamento dos dados. Por uma questão de desempenho, fez-se, para cada bloco, a contagem dos símbolos e a compressão RLE.

Implementação de uma função "***rle_aux***" que faz a compressão individual de um bloco: Recebe um Buffer inicial e percorre o array dos símbolos, um a um, e gera o padrão RLE {0}{símbolo}{nº repeticoes} quando um símbolo se repete pelo menos 4 vezes. Nesta função existe ainda uma exceção (feita com uma função auxiliar "***excecao"***) para o caso de termos o símbolo {0}. Esta função escreve num ficheiro final (".rle") e é chamada na função "***rle***" de modo a aplicar a compressão bloco a bloco.

Implementação de uma função "***freq***": Recebe um Buffer inicial (armazena o conteúdo do bloco) e calcula as frequências dos símbolos de um bloco e organiza-as num array de acordo com a ordem dos símbolos ASCII, guardando num Buffer final. Esta função é chamada na função "***rle_to_freq***" de modo a aplicar o cálculo das frequências dos símbolos para cada bloco. Para tal, define um Buffer que armazena os dados do ficheiro (".rle"), distingue cada bloco através de um índice e vai formatando o Buffer final de acordo com as especificações do enunciado.

- ## Módulo B

Para a realização deste módulo decidimos armazenar o conteúdo do ficheiro que recebemos (".freq") num Buffer que consiste num array dinâmico. Optamos por este método para que ao longo de todo o processo não termos de lidar, diretamente, com alocação de memória, e para isso, criamos o header "buffer", cujas funções lidam automaticamente com essas questões.

No processo de codificação, e para conseguir implementar o processamento por blocos, usamos 3 Buffers: o inicial (como já referido, armazena o conteúdo do ficheiro inicial), o intermédio (que auxilia o processamento armazenando sucessivamente o conteúdo do bloco a processar) e o final (que armazena os códigos já processados com a formatação ideal para escrever no ficheiro a devolver).

A cada bloco, organizamos as frequências num array de inteiros e aplicamos às frequências não nulas o algoritmo Shannon-Fano, que insere os códigos referentes aos símbolos numa matriz também ela dinâmica. A partir daí, inserimos os códigos no Buffer final, pela ordem antes da ordenação implícita no algoritmo.

- ## Módulo C

Escolheu-se dividir as tarefas dos dois elementos do grupo entre;

“Tarefa Principal”:

Onde se leva a cabo a manipulação dos ficheiros .txt, .RLE e .cod para criar o ficheiro .shaf e;

Para esta tarefa, determinou-se que se deveria proceder da seguinte forma;

1. Determinar o código ASCII para cada símbolo do ficheiro .txt
2. Usar esse código como “chave” para encontrar a codificação SF do símbolo no ficheiro .cod
3. Substituir cada símbolo do ficheiro .txt pelo seu código SF
4. Resolver o problema de codificação do ficheiro .RLE da mesma forma, evitando a conversão dos símbolos próprios da codificação .RLE, por exemplo: {0}1 e {31}1.

“Tarefas Auxiliares”:

Onde são resolvidos problemas como

- Leitura de ficheiros
- Criação de estruturas ARRAY
- Contagem do tempo de execução – ( )
- Impressão de texto de saída na consola

É de referir que grande parte das soluções de programação necessárias para completar as duas tarefas eram desconhecidas dos elementos do grupo e exigiram pesquisa e procura de exemplos.

Explicação e Análise Crítica das Principais Funções

• Módulo A

- Função **exec_moduloA**: função principal
  Esta função encontra-se dividida em etapas:
  1. Lê o ficheiro recebido e escreve num Buffer de "origem"
  2. Aplica a compressão RLE e faz o cálculo das frequências
  3. Escreve no terminal toda a informação sobre a sua execução

  Limitações: Incapacidade de forçar compressão

- Função **rle** -> função que aplica a compressão RLE, bloco a bloco, aplicando-lhes a função rle_aux, e que coloca num Buffer os resultados da compressão.

- Função **rle_aux** -> Faz a compressão individual de um bloco.
  Esta função encontra-se dividida em etapas:
  1. Distinção entre o caso de exceção (quando temos o símbolo {0}) e os restantos casos
  2. Distinção entre símbolos que se repetem pelo menos 4 vezes e os que se repetem menos de 4 vezes
  3. Aplicação do padrão RLE {0}{símbolo}{nº repeticoes} aos primeiros e repetição dos segundos tal como estavam originalmente
  4. Escrever num ficheiro de saída (".rle")

- Função **freq** -> função que faz o cálculo das frequências dos símbolos para um bloco individual e guarda as frequências dos símbolos num array organizado pela ordem dos símbolos ASCII.
  Esta função encontra-se dividida em etapas:
  1. Identificação dos casos de exceção (**)
  2. Armazenar num array as frequências de cada símbolo

  **Quando temos{0}{símbolo}{nº repeticões}
  **Nº repetições pode ter 1, 2 ou 3 algarismos

- Função **rle_to_freq** -> Função que calcula as frequências dos símbolos dos blocos e escreve num ficheiro ".freq" a frequência de cada símbolo conforme as especificações do enunciado.

- # Módulo B

- Função exec_moduloB -> função principal
  Esta função encontra-se dividida em 2 etapas:
  1. Lê o ficheiro recebido e escreve no Buffer inicial e depois da codificação, escreve o Buffer final previamente alocado no ficheiro a retornar.
  2. Escreve no terminal toda a informação sobre a sua execução

- Função code -> esta função gere a divisão por blocos do Buffer inicial aplicando a estes a função codeBlock e coloca no Buffer final os dados que não precisam de processamento.

- Função codeBlock -> surge como solução para o processamento individual para cada bloco. Esta função encontra-se dividida em 3 etapas:
  1. Organizamos por ordem decrescente as frequências num array (com o algoritmo BubbleSort);
  2. Aplicamos a codificação propiamente dita com a função calculcar_codigos_SF e armazenamos o resultado numa matriz dinâmica (optamos por esta solução por não saber exatamente o tamanho de cada código);
  3. Com a codificação finalizada copiamos para o Buffer final os códigos pela ordem correta e para isso, usamos o array de frequências ainda desordenado que surge inicialmente para encontrar o índice onde o código deve ser escrito.

- Função calcular_codigos_SF -> algoritmo disponibilizado pelo professor e adaptado para escrever os códigos na matriz dinâmica.

- # Módulo C

- Função **Main:** função principal

  Esta função tem como objetivo ler e correr todas as funções secundárias.

- Função **timeCode** -> função que faz a contagem do tempo que demora a executar o programa.

- Função **fileOpener** -> função que abre e lê o ficheiro.

- Função **Contador** -> Esta função está dividida em etapas:
  1. Abre e lê um ficheiro .cod
  2. Utiliza um contador para contar especificamente 4 ocorrências do caracter "@"
  3. A partir da 4ª ocorrência de "@", cria um array onde vai guardar todas as ocorrências de vazios (por cada vazio acrescenta no array o "-1") e todas as ocorrências de números Shannon-Fano

- Função **filetxt** -> Esta função abre e guarda os elementos do ficheiro .txt num array.

- Função **fsize** -> função disponibilizada pelo professor para dar o tamanho dos blocos do ficheiro

Resultados da Execução dos Módulos

- **Módulo A**

aaa.txt

.rle	.rle.freq

Versão do grupo:
abcbdbcbabdbcbdbcbabacdbacdaabcbdd{0}$_1$
a{4}$_1$bbcbc{0}$_1$a{7}$_1$bcbcbdacabb{0}$_1$a{21}
$_1${0}$_1$b{21}$_1${0}$_1$c{10}$_1${0}$_1$d{6}$_1$bcbdbabd{
0}$_1$b{12}$_1${0}$_1$a{21}$_1${0}$_1$b{21}$_1${0}$_1$c{4}$_1${0
}$_1$d{10}$_1${0}$_1$b{37}$_1${0}$_1$a{56}$_1${0}$_1$b{12}$_1${
0}$_1$c{7}$_1$dbabaaababaacacdaddcababababa
bababbabababababababbababcdcdcdcdba
babdccdbacddcbacdbacdbccdbabbabbdbbdb
a{0}$_1$b{29}$_1${0}$_1$a{34}$_1${0}$_1$b{5}$_1$dbcbdcdbc
dbcbdbbcbabcbdbcbabdbcbdbcbabacdbacda
abcbdd{0}$_1$a{4}$_1$bbcbc{0}$_1$a{7}$_1$bcbcbdacab
b{0}$_1$a{21}$_1${0}$_1$b{21}$_1${0}$_1$c{10}$_1${0}$_1$d{6}$_1$
bcbdbabd{0}$_1$b{12}$_1${0}$_1$a{21}$_1${0}$_1$b{21}$_1${
0}$_1$c{4}$_1${0}$_1$d{10}$_1${0}$_1$b{37}$_1${0}$_1$a{56}$_1${0
}$_1$b{12}$_1${0}$_1$c{7}$_1$dbabaaababaacacdaddcab
ababababababbbabababababababbbababc
dcdcdcdbababdccdbacddcbacdbacdbccdbab
babbdbbdba{0}$_1$b{29}$_1${0}$_1$a{34}$_1${0}$_1$b{5}$_1$
dbcbdcdbcdbcbdbbcbcabcbdbcbabdbcbdbcb
abacdbacdaabcbdd{0}$_1$a{4}$_1$bbcbc{0}$_1$a{7}$_1$
bcbcbdacabb{0}$_1$a{21}$_1${0}$_1$b{21}$_1${0}$_1$c{10
}$_1${0}$_1$d{6}$_1$bcbdbabd{0}$_1$b{12}$_1${0}$_1$a{21}$_1$
{0}$_1$b{21}$_1${0}$_1$c{4}$_1${0}$_1$d{10}$_1${0}$_1$b{37}$_1$
{0}$_1$a{56}$_1${0}$_1$b{12}$_1${0}$_1$c{7}$_1$dbbaaababa
acacdddcababababababababbabababababa
babbababcdcdcdcdbababdccdbacddcbacdba
cdbccdbabbabbdbbdba{0}$_1$b{29}$_1${0}$_1$a{34}
$_1${0}$_1$b{5}$_1$dbcbdcdbcdbcbdbbcbcabcbdbcba
bdbcbdbcbabacdbacdaabcbdd{0}$_1$a{4}$_1$bbcb
c{0}$_1$a{7}$_1$bcbcbdacabb{0}$_1$a{21}$_1${0}$_1$b{21
}$_1${0}$_1$c{10}$_1${0}$_1$d{6}$_1$bcbdbabd{0}$_1$b{12}$_1$
{0}$_1$a{21}$_1${0}$_1$b{21}$_1${0}$_1$c{4}$_1${0}$_1$d{10}$_1${
0}$_1$b{37}$_1${0}$_1$a{56}$_1${0}$_1$b{12}$_1${0}$_1$c{7}$_1$d
babaaababaacacdaddcababababababababbab
abababababababbababcdcdcdcdbababdccdb
acddcbacdbacdbccdbabbabbdbbdba{0}$_1$b{2
9}$_1${0}$_1$a{34}$_1${0}$_1$b{5}$_1$dbcbdcdbcdbcbdbb
cbabcbdbcbabdbcbdbcbabacdbacdaabcbdd{
0}$_1$a{4}$_1$bbcbc{0}$_1$a{7}$_1$bcbcbdacabb{0}$_1$a{
21}$_1${0}$_1$b{21}$_1${0}$_1$c{10}$_1${0}$_1$d{6}$_1$bcbdba
bd{0}$_1$b{12}$_1${0}$_1$a{21}$_1${0}$_1$b{21}$_1${0}$_1$c{4
}$_1${0}$_1$d{10}$_1${0}$_1$b{37}$_1${0}$_1$a{56}$_1${0}$_1$b{1
2}$_1${0}$_1$c{7}$_1$dbabaaababaacacdaddcabababa
babababbababababababababbbababcdcdcd
cdbababdccdbacddcbacdbacdbccdbabbabbd
bbdba{0}$_1$b{29}$_1${0}$_1$a{34}$_1${0}$_1$b{5}$_1$dbcbd
cdbcdbcbdbbcbcabcbdbcbabdbcbdbcbabacd
bacdaabcbdd{0}$_1$a{4}$_1$bbcbc{0}$_1$a{7}$_1$bcbcb
dacabb{0}$_1$a{21}$_1${0}$_1$b{21}$_1${0}$_1$c{10}$_1${0}
	Versão do grupo:
	@R@1@2574@0;;
	;;;1129;1
	254;328;289;0;;
	;;;
	;;;@0

1d{6}1bcbdbabd{0}1b{12}1{0}1a{21}1{0}1b{21}1{0}1c{4}1{0}1d{10}1{0}1b{37}1{0}1a{56}1{0}1b{12}1{0}1c{7}1dbabaaababaacacdaddcababababababababbababababababababababcdcdcdcdbababdccdbacddcbacdbacdbccdbabbabbdbbdba{0}1b{29}1{0}1a{34}1{0}1b{5}1dbcbdcdbcdbcbdbbcbcaaad	
Versão do professor: abcbdbcbabdbcbdbcbabacdbacdaabcbdd{0}1a{4}1bbcbc{0}1a{7}1bcbcbdacabb{0}1a{21}1{0}1b{21}1{0}1c{10}1{0}1d{6}1bcbdbabd{0}1b{12}1{0}1a{21}1{0}1b{21}1{0}1c{4}1{0}1d{10}1{0}1b{37}1{0}1a{56}1{0}1b{12}1{0}1c{7}1dbabaaababaacacdaddcababababababababababbababababababababbababcdcdcdcdbababdccdbacddcbacdbacdbccdbabbabbdbbdba{0}1b{29}1{0}1a{34}1{0}1b{5}1dbcbdcdbcdbcbdbbcbabcbdbcbabdbcbdbcbabacdbacdaabcbdd{0}1a{4}1bbcbc{0}1a{7}1bcbcbdacabb{0}1a{21}1{0}1b{21}1{0}1c{10}1{0}1d{6}1bcbdbabd{0}1b{12}1{0}1a{21}1{0}1b{21}1{0}1c{4}1{0}1d{10}1{0}1b{37}1{0}1a{56}1{0}1b{12}1{0}1c{7}1dbabaaababaacacdaddcababababababababbababababababababbababcdcdcdcdbababdccdbacddcbacdbacdbccdbabbabbdbbdba{0}1b{29}1{0}1a{34}1{0}1b{5}1dbcbdcdbcdbcbdbbcbcabcbdbcbabdbcbdbcbabacdbacdaabcbdd{0}1a{4}1bbcbc{0}1a{7}1bcbcbdacabb{0}1a{21}1{0}1b{21}1{0}1c{10}1{0}1d{6}1bcbdbabd{0}1b{12}1{0}1a{21}1{0}1b{21}1{0}1c{4}1{0}1d{10}1{0}1b{37}1{0}1a{56}1{0}1b{12}1{0}1c{7}1dbbaaababaacacdddcabababababababababbababababababababbababcdcdcdcdbababdccdbacddcbacdbacdbccdbabbabbdbbdba{0}1b{29}1{0}1a{34}1{0}1b{5}1dbcbdcdbcdbcbdbbcbcabcbdbcbabdbcbdbcbabacdbacdaabcbdd{0}1a{4}1bbcbc{0}1a{7}1bcbcbdacabb{0}1a{21}1{0}1b{21}1{0}1c{10}1{0}1d{6}1bcbdbabd{0}1b{12}1{0}1a{21}1{0}1b{21}1{0}1c{4}1{0}1d{10}1{0}1b{37}1{0}1a{56}1{0}1b{12}1{0}1c{7}1dbabaaababaacacdaddcababababababababbababababababababbababcdcdcdcdbababdccdbacddcbacdbacdbccdbabbabbdbbdba{0}1b{29}1{0}1a{34}1{0}1b{5}1dbcbdcdbcdbcbdbbcbabcbdbcbabdbcbdbcbabacdbacdaabcbdd{0}1a{4}1bbcbc{0}1a{7}1bcbcbdacabb{0}1a{21}1{0}1b{21}1{0}1c{10}1{0}1d{6}1bcbdbabd{0}1b{12}1{0}1a{21}1{0}1b{21}1{0}1c{4}1{0}1d{10}1{0}1b{37}1{0}1a{56}1{0}1b{12}1{0}1c{7}1dbabaaababaacacdaddcababababababababbababababababababbababcdcdcdcdbababdccdbacddcbacdbacdbccdbabbabbdbbdba{0}1b{29}1{0}1a{34}1{0}1b{5}1dbcbdcdbcdbcbdbbcbcabcbdbcbabdbcbdbcbabacdbacdaabcbdd{0}1a{4}1bbcbc{0}1a{7}1bcbcbdacabb{0}1a{21}1{0}1b{21}1{0}1c{10}1{0}1d{6}1bcbdbabd{0}1b{12}1{0}1a{21}1{0}1b	Versão do professor: @R@2@991@74;0;;;9;4;;9;0;;8;0;8;0;;;;;;;;16;0;;;;;;4;0;;;;4;0;;4;0;;;;;;;;;;;;;;;;;;4;0;;211;333;156;143;0;;;@431@34;0;;;3;2;;3;0;;4;0;4;0;;;;;;;;8;0;;;;;;;2;0;;;;2;0;;2;0;;;;;;;;;;;;;;;;;;2;0;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;96;141;64;62;0;;;@0

{21}$_1${0}$_1$c{4}$_1${0}$_1$d{10}$_1${0}$_1$b{37}$_1${0}$_1$a{56}$_1${0}$_1$b{12}$_1${0}$_1$c{7}$_1$dbabaaababaacacdaddcababababababababbababababababababbababcdcdcdcdbababdccdbacddcbacdbacdbccdbabbabbdbbdba{0}$_1$b{29}$_1${0}$_1$a{34}$_1${0}$_1$b{5}$_1$dbcbdcdbcdbcbdbbcbcaaad	

Nota: As colunas "cortam" mas continuam na página seguinte. Só existem duas "secções" distintas: a versão do grupo e a versão do professor (com o início identificado).

Nota: A função que faz o rle não está a gerar o ficheiro como esperado. No entanto, a função que rle_to_freq funciona perfeitamente (a função que gera o ficheiro das frequências do ficheiro dos símbolos rle). Ou seja, se fizermos um teste com o ficheiro (".rle") do professor, conseguimos gerar um ficheiro (".rle.freq") tal como esperado (pois esta função funciona corretamente). No entanto, como a nossa função que gera o ficheiro (".rle") está a gerar mal o ficheiro, se depois aplicarmos a função que faz o cálculo das frequências vamos obter um ficheiro diferente do esperado.

- **Módulo B**

.freq	.cod
@R@2@991@74;0;;;9;4;;9;0;;8;0;8;0;;;;;;;;16;0;;;;;;;4;0;;;;4;0;;4;0;;;;;;;;;;;;;;;;;;4;0;;211;333;156;143;0;;;@431@34;0;;;3;2;;3;0;;4;0;4;0;;;;;;;;8;0;;;;;;;2;0;;;;2;0;;2;0;;;;;;;;;;;;;;;;;;2;0;;96;141;64;62;0;;@0	@R@2@991@1110;;;;1111010;11111011;11111100;1111011;;;1111100;;11111010;;;;;;;;111100;;;;;;;11111101;;;;;11111110;;;111111110;;;;;;;;;;;;;;;;;;111111111;;;01;00;10;110;;@431@110;;;;111100;1111011;1111100;1111010;;;111010;;111011;;;;;;;;11100;;;;;;;1111101;;;;;1111110;;;11111110;;;;;;;;;;;;;;;;;11111111;;;01;00;100;101;;@0
@N@2@2048@0;;;768;853;228;199;0;;@952@0;;361;401;100;90;0;;	@N@2@2048@;;;10;0;110;111;;;@952@;;;10;0;110;111;;;

;;; ;;;;;;;;;;;;@0	;; ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;@0

Conclusões

No Módulo A, gostaríamos de ter acrescentado a funcionalidade de forçar a compressão. Não foi possível adicioná-la uma vez que as tentativas de implementação causavam erros que comprometiam o bom funcionamento do resto do programa. Decidimos, portanto, manter o “grosso” do programa a funcionar, ainda que sem esta funcionalidade. A função que faz a compressão **RLE** não está a gerar o ficheiro como esperado. No entanto, a função que rle_to_freq funciona perfeitamente (a função que gera o ficheiro das frequências do ficheiro dos símbolos do ficheiro rle). Ou seja, se fizermos um teste com o ficheiro (“.rle”) do professor, conseguimos gerar um ficheiro (“.rle.freq”) tal como esperado (pois esta função funciona corretamente). No entanto, como a nossa função que gera o ficheiro (“.rle”) está a gerar mal o ficheiro, se depois aplicarmos a função que faz o cálculo das frequências vamos obter um ficheiro diferente do esperado.

A fim de completar a ferramenta ***Shafa*** e capacitá-la de todas as funcionalidades previstas, gostaríamos de ter implementado os Módulos C e D (que, por incumprimento dos colegas encarregues pelos módulos, não foi possível entregar).

Referências

- **Bibliográficas**
  1. Bibliografia das disciplinas de Comunicação de Dados e Programação Imperativa

- **Recursos da internet**
  1. www.sanfoundry.com
  2. www.javatpoint.com
  3. pt.stackoverflow.com/