INSTITUTO FEDERAL
Ceará

Joao Gabriel Carneiro Medeiros
<joao.gabriel.carneiro02@aluno.ifce.edu.br>

Avaliação 3 | Sistemas Multimídia | 2021.1

1 mensagem

Formulários Google <forms-receipts-noreply@google.com> 20 de junho de 2021 18:07
Para: joao.gabriel.carneiro02@aluno.ifce.edu.br

Agradecemos o preenchimento de Avaliação 3 | Sistemas Multimídia | 2021.1

Veja as respostas enviadas.

Avaliação 3 | Sistemas Multimídia | 2021.1

Seu e-mail (**joao.gabriel.carneiro02@aluno.ifce.edu.br**) foi registrado quando você enviou este formulário.

Compare as taxas de compressão da imagem a seguir usando codificação de Huffman e codificação RLE. *

1	1	1	1	5	5	5	5	2	2	2	2
1	1	1	5	5	5	5	5	5	2	2	3
1	1	5	5	5	5	5	2	2	3	3	2
1	1	1	1	5	5	5	2	2	2	2	2
1	1	1	1	1	1	5	2	2	2	3	2
1	1	1	1	1	1	1	1	1	1	1	1

*

Para o método de Huffman, onde tem-se o uso de uma árvore binária cujo os nós se relacionam com os filhos através de um valor de bit, seja ele "0" ou "1". E, assim, símbolos/caracteres que mais apareçam na mensagem analisada deverão ficar nos níveis mais superiores da árvore enquanto os que menos aparecerem ficarão nos níveis mais inferiores da árvore. Após essa breve revisão do método, com base numa ferramenta para cálculos usada pelo aluno disponível no site " planetcalc.com" (Mais especificamente em: https://planetcalc.com/2481/) que realiza operações de codificação/descodificação usando do método em destaque, temos os seguintes resultados:
>> Codificação dos caracteres no método de Huffman:
1 = 0; 2 = 111; 3 = 110; 5 = 10.
>> Total de bits na imagem:
Com compressão usando Código de Huffman: 135 bits << (Baseado no tamanho, em bits, dos códigos achados para cada caractere que achamos acima e suas "frequências" na mensagem).
Sem compressão usando Código de Huffman: 144 bits << (pois cada um dos 4 números é representado por 2 bits).
>> Temos uma taxa de compressão relativamente boa para a imagem de entrada acima:
135/144 ()
**

Para a codificação RLE ("Run Lenght Encode"), que trabalha com "Redundância Espacial", temos um método usado para abreviar grandes sequências de caracteres que possuam o mesmo valor, utilizando para tal tarefa os parâmetros (comprimento e símbolo) que são representações "reduzidas", num espaço de 3 bytes, do tamanho físico de uma sequência. O primeiro byte geralmente representa um caractere especial ("escape code") que vai permitir ao algoritmo saber da existência de uma repetição, enquanto que o segundo indica o caractere que está a se repetir. O último byte indica o número de repetições a fazer. Usando o algoritmo obtemos o seguinte resultado na saída (Irei, ignorar o "escape code" para facilitar mais a visualização no formulário)
>> Codificação dos caracteres no método LZW (Cada parêntesis representa o espaço de 3 bytes reservado de que falei anteriormente que é comumente usado por esse método na sua codificação):
(01, 04) (05, 04) (02, 04)
(01, 03) (05, 06) (02, 02) (03, 01)
(01, 02) (05, 05) (02, 02) (03, 02) (02, 01)
(01, 04) (05, 03) (02, 05)
(01, 06) (05, 01) (02, 03) (03, 01) (02, 01)
(01, 12)
>> Total de bits na imagem: Muitos mesmo! kkkkk. Esse método é muito usado em imagens onde se tem muitas repetições em larga escala de símbolos distribuídos uniformemente na região encobrida pelo método, na imagem da questão não há tanta uniformidade assim, as repetições são muito poucas, então, há muitos espaços de 3 bytes (que por si só são bem grandes se comparados a bits) reservados para símbolos que nem se repetir eles estão sendo em determinada sessão, o que faz com que, nesse caso,o número de bits na imagem aumente muito mais do que se tinha antes (que era 144 como já foi dito).
Logo, se compararmos as taxas de compressão de um método com o outro, definitivamente as do de Huffman são melhores, nem cálculo precisa pois a situação das repetições dos símbolos presentes na imagem de entrada da questão está longe de ser favorável para o uso do método RLE, é melhor usar do de Huffman que serve para repetições constantes ou inconstantes de símbolos em curtas e médias distâncias e tem um bom desempenho em sua compressão de dados.

Um decodificador recebe a seguinte mensagem binária 0 100 11 0 1010 0 1011 0 100 11 0 e o dicionário (A: 0, B: 100, C: 1010, D: 1011, R:11). Decodifique a mensagem. Qual método de compressão pode ter produzido esse dicionário? Justifique sua resposta. *

*
Decodificando a mensagem binária temos: "ABRACADABRA".
**
Agora para qual 'método de compressão' que possa ter produzido tal dicionário, prefiro responder através da representação de uma árvore binária. Pegando a mensagem "ABRACADABRA" que acabamos de descobrir na decodificação pedida, podemos representá-la na seguinte, de algumas possíveis configurações, representação de uma árvore binária que segue as exigências de montagem pedidas no método de Huffman:
Obs. Saiba que as frequências dos símbolos na mensagem ABRACADABRA são as seguintes: A=5, B=2, R=2, C=1, D=1 <<< (O que dá um total de 11 elementos no total).
(11)
0 / \ 1
[A: 0] (A=5) (6)
0 / \
(4) \ 1
0 / \ 1 \
[B: 100] (B=2) (2) (R=2) [R:11]
0 / \ 1
[C: 1010] (C=1) (D=1) [D: 1011]
***
Logo, provavelmente o método de compressão utilizado foi um que usa do método de compressão de Huffman. Pois, não somente mostrou-se acima a possibilidade de uma árvore binária desse método que represente essa mensagem binária, como o próprio método de Huffman em si é criado (e dividido em seus dois casos: Adaptativo e Estático) para que se pudesse usar dessa compressão em casos onde nos quais não se conhecem as frequências de ocorrência de cada caractere no texto, que é o caso do exemplo do enunciado acima, note que nos é dado uma mensagem binária como entrada e um dicionário mas nenhuma frequência dos símbolos nessa mensagem. Com Huffman, mesmo sem o dicionário podemos chegar a um (Dicionário) facilmente através dessa estratégia de que gira em torno da construção da árvore de Huffman que em seu fim nos dá um dicionário como o que é dado no enunciado da questão. Tudo isso através dessa "simples" ideia base do método: De atribuir um código de bits de comprimento variável a cada símbolo, onde os símbolos com menos probabilidade tem um código de bits de comprimento maior em relação aos símbolos que têm maior probabilidade, que possuem um código menor (Tudo isso, mais uma vez destacando essa propriedade do método que é uma de suas grandes vantagens, sem necessitar de conhecer a estatística dos símbolos da mensagem!).

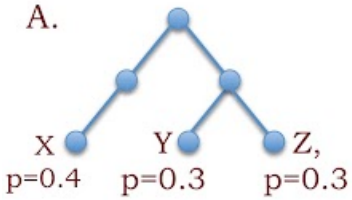
Temos uma imagem de tamanho 100x100 pixels. Todos os pixels da metade superior tem o valor de cinza 32 e todos os pixels da metade inferior tem valor 189. É correto afirmar que: *

- [ ] Os dados fornecidos são insuficientes para o cálculo da entropia.

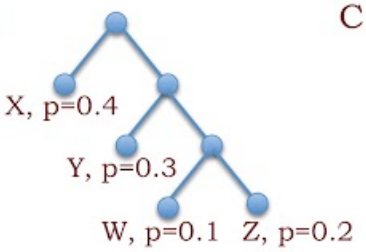
- [x] São necessários 8 bits para armazenar o valor de cinza de cada pixel, mesmo com uso de compressão de dados.
- [x] Entre os métodos de compressão estudados em aula, o que apresenta maior taxa de compressão para essa imagem é a codificação Run Length.
- [ ] Métodos baseados em redundância espacial não alcançam taxas de compressão tão elevadas quanto os métodos baseados em codificação de tamanho variável.

Consideramos a probabilidade p de cada símbolo X, Y, W e Z em um canal de dados. Quais das árvores abaixo podem ser obtidas na codificação de Huffman? *

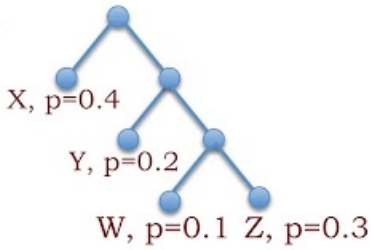
A.



B.



C.



- [ ] A
- [x] B
- [ ] C

Queremos codificar o texto "ABRACADABRANTESCAMENTE". Como você compara o método de Huffman com a codificação LZW no que se refere a taxas de compressão desse texto? Explique a vantagem ou desvantagem de um método com relação ao outro. *

*

Para o método de Huffman, como já explicado nas questões subjetivas anteriores a esta, temos o uso de uma árvore binária cujo os nós se relacionam com os filhos através de um valor de bit, seja ele "0" ou "1". Assim, símbolos/caracteres que mais apareçam na mensagem analisada deverão ficar nos níveis mais superiores da árvore enquanto os que menos aparecerem ficarão nos níveis mais inferiores da árvore. Após essa breve revisão do método, com base numa ferramenta para cálculos usada pelo aluno disponível no site " planetcalc.com" (Mais especificamente em: https://planetcalc.com/2481/) que realiza operações de codificação/descodificação usando do método em destaque, temos os seguintes resultados:
>> Codificação dos caracteres no método de Huffman:
A = 10; B = 000; C = 001; D = 01110; E = 110; R = 1110; N = 1111; T = 010; S = 01111; M = 0110.
>> Total de bits na imagem:

Com o método de Huffman: 69 bits << (Baseado no tamanho, em bits, dos códigos achados para cada caractere que achamos acima e suas "frequências" na mensagem).
Sem o método de Huffman: 88 bits << (Baseado no fato de que para cada caractere o mesmo é representado por um total de até 4 bits).
**
Para o método de LZW, temos uma compressão de dados baseada no uso de um dicionário que produz códigos de comprimento fixo, que se associam a sequências de símbolos de comprimento variável. Os códigos gerados durante o processo de codificação ou descodificação correspondem a sequências de símbolos cada vez mais longas que vão sendo adicionadas ao dicionário para catalogar com melhor precisão e organização as respectivas identificações dadas a essas sequências de símbolos que podem vir a aparecer em determinada mensagem sob analise pelo método. Após essa breve revisão do método, com base numa ferramenta para cálculos usada pelo aluno disponível no site " planetcalc.com" (Mais especificamente em: https://planetcalc.com/9069/) que realiza operações de codificação/descodificação usando do método em destaque, temos os seguintes resultados:
>> Codificação dos caracteres no método LZW: Para o senhor ter ideia, o meu dicionário passou de 25 elementos cada um sus devidas codificações que os identifiquem caso apareçam em determinado trecho de uma sequencia (começando pela codificação de número 1 para o caractere "A" e indo para codificações para grupos de até 3 símbolos combinados entre si por exemplo).
>> Total de bits na imagem:
Com o método LZW: 84 bits. << (Baseado no tamanho, em bits, dos códigos achados para cada caractere que achamos acima e suas "frequências" na mensagem).
Sem o método LZW: 88 bits.
***
Quanto as vantagens/desvantagens de um Método para com o outro é que primeiramente no LZW a taxa de compressão do mesmo é melhor do que a taxa de compressão do de Huffman, uma vez que o método LZW (que usa de "dicionário" e não de uma 'árvore' como estrutura 'suporte' para seus cálculos) pode usar de códigos de tamanho fixo para representar uma sequência de tamanho "variável". No LZW queremos, então, conseguir achar na mensagem em análise, sequências de letras dos mais variados tipos que se repitam, e, quanto maiores forem essas maior será a compressão feita! Isso, nos possibilita representar toda uma sequência por um único código, o que não acontece no método de Huffman que não dispõe dessa mesma "flexibilidade" na identificação dos elementos de uma mensagem.

Uma imagem em níveis de cinza apresenta os seguintes valores de pixels e suas respectivas frequências em porcentagem: 23 (10%), 39 (15%), 124 (15%), 189 (40%) e 222 (20%). Podemos afirmar: *

- [x] A codificação de Huffman deve usar apenas 1 bit para o valor de cinza 189.
- [ ] A codificação de Shannon-Fano deve usar apenas 1 bit para o valor 10.
- [ ] As informações disponíveis não permitem estimar a taxa de compressão obtida pela codificação Run Lenth.
- [x] A taxa de compressão do método LZW é mais elevada que a da codificação de Huffman.

***ABAIXO AS DUAS RESPOSTAS QUE ESTAVAM ERRADAS E SUAS CORREÇÕES, NA PROVA DISPONIBILIZADA ACIMA:**

✗ Uma imagem em níveis de cinza apresenta os seguintes valores de pixels e suas respectivas frequências em porcentagem: 23 (10%), 39 (15%), 124 (15%), 189 (40%) e 222 (20%). Podemos afirmar: * 0,5 / 1

- [x] A codificação de Huffman deve usar apenas 1 bit para o valor de cinza 189. ✓
- [ ] A codificação de Shannon-Fano deve usar apenas 1 bit para o valor 10.
- [ ] As informações disponíveis não permitem estimar a taxa de compressão obtida pela codificação Run Lenth.
- [x] A taxa de compressão do método LZW é mais elevada que a da codificação de Huffman. ✗

Resposta correta

- [x] A codificação de Huffman deve usar apenas 1 bit para o valor de cinza 189.
- [x] As informações disponíveis não permitem estimar a taxa de compressão obtida pela codificação Run Lenth.

Adicionar feedback individual

Enviada: 20/06/2021 18:07

✗ Temos uma imagem de tamanho 100x100 pixels. Todos os pixels da metade superior tem o valor de cinza 32 e todos os pixels da metade inferior tem valor 189. É correto afirmar que: * 0,5 / 1

- [ ] Os dados fornecidos são insuficientes para o cálculo da entropia.
- [x] São necessários 8 bits para armazenar o valor de cinza de cada pixel, mesmo com uso de compressão de dados. ✗
- [x] Entre os métodos de compressão estudados em aula, o que apresenta maior taxa de compressão para essa imagem é a codificação Run Length. ✓
- [ ] Métodos baseados em redundância espacial não alcançam taxas de compressão tão elevadas quanto os métodos baseados em codificação de tamanho variável.

Resposta correta

- [x] Entre os métodos de compressão estudados em aula, o que apresenta maior taxa de compressão para essa imagem é a codificação Run Length.

Adicionar feedback individual