MAC 323 - Estruturas de Dados (Pós: MAC 5710)
Segundo semestre de 1998

Quarto Exercício-Programa – Devolução: 2 de dezembro de 1998

Códigos de Huffman

1. Introdução

O objetivo deste exercício-programa é fazer um programa que permita codificar e decodificar arquivos usando códigos de Huffman.

Você deve entregar um disquete contendo o programa fonte e também as listagens do programa e de sua saída (tudo, de preferência, dentro de um saco plástico) até o dia 2 de dezembro. Não deixe de guardar cópias do programa que você entregar.

2. Código de Huffman

Uma aplicação interessante de árvores binárias em codificação de informações a serem transmitidas e compactação de arquivos são os *códigos de Huffman*.

Para enviar uma mensagem através de um cabo de transmissão, os caracteres da mensagem são enviados um a um, modificados através de alguma tabela de codificação. Em geral, este código forma um número binário. Como a velocidade de transmissão é importante, é interessante tornar a mensagem tão curta quanto possível sem perder a capacidade de decodificar o texto enviado.

Um algoritmo de determinação de códigos binários de comprimentos variados para caracteres (não é o caso que cada caractere seja representado com o mesmo número de bits), baseado na freqüência de uso destes caracteres, foi sugerido por Shannon [5] (veja também [6]) Fano [3] e logo depois esta idéia foi aperfeiçoada por D.A. Huffman [4]. A idéia é associar números binários com menos bits aos caracteres mais usados num texto. Desta maneira espera-se que o número de bits economizados para codificar os caracteres que ocorrem com maior freqüência em um texto seja mais que o suficiente para cobrir o déficit que ocorre ao codificarmos os caracteres que ocorrem raramente com cadeias longas de bits.

O algoritmo de Huffman que será implementado para codificar um arquivo consistirá de três fases:

1. Na primeira fase a freqüência de cada caractere que ocorre no texto deverá ser calculada.
2. A segunda fase do algoritmo de Huffman consiste em construir uma árvore binária baseada na freqüência de uso destas letras de modo que as mais freqüentemente usadas apareçam mais perto da raiz que as menos freqüentemente usadas. Chamaremos esta árvore binária de *árvore de Huffman*.

   Em cada passo desta fase teremos uma coleção de árvores binárias (ou seja, uma floresta formada por árvores binárias). As folhas de cada uma destas árvores correspondem a um conjunto de caracteres que ocorrem no texto. À raiz de cada uma destas árvores será associado um número que corresponde à freqüência com que os caracteres associados às folhas desta árvore ocorrem no texto. Escolheremos então as duas árvores desta floresta com a menor freqüência e as transformaremos em uma única árvore, ligando-as a uma nova raiz cujo valor será dado pela soma dos valores das freqüências das duas subárvores (veja um exemplo na próxima seção).
3. Finalmente na terceira fase a árvore de Huffman será usada para codificar e decodificar o texto. Na Seção 4 e 5 é mostrado um exemplo de como usar a árvore para codificar e decodificar.

A árvore de Huffman construída durante a codificação será utilizada na decodificação (logo, a árvore de Huffman deverá ser armazenada junto com o texto que foi codificado).

[Mais sobre códigos de Huffman pode ser encontrado (entre outros livros) no: Capítulo 5, Seção 1 de [1]; Capítulo 17, Seção 3 de [2]; Capítulo 10, Seção 7 de [7]; e Capítulo 5, Seção 1 de [8].]

3. Construção da Árvore de Huffman: Um exemplo

Suponha que uma mensagem seja composta pelos caracteres `A, B, C, D, E, R` e que a freqüência destas letras na mensagem seja, respectivamente, 22, 9, 10, 12, 16, 8.

A	B	C	D	E	R
22	9	10	12	16	8

O algoritmo de Huffman contrói uma árvore binária (a árvore de Huffman) baseada na freqüência de uso destas letras de modo que as mais freqüentemente usadas {`A,E`} apareçam mais perto da raiz que as menos freqüentemente usadas {`B,R`}.

A construção desta árvore binária é feita de baixo para cima (das folhas para a raiz), começando a partir das letras menos usadas até atingir a raiz. Nesta árvore, as letras serão representadas nas folhas e os seus vértices internos conterão um número correspondente à soma das freqüências dos seus descendentes.

Vejamos a construção da árvore de Huffman para o alfabeto e freqüências do exemplo acima.

Em cada passo do algoritmo existe uma coleção de árvores de Huffman. São escolhidas duas com as menores freqüências associadas e elas são transformadas em uma só árvore cujo valor é a soma dos valores dessas duas.

Inicialmente, cada uma das letras será uma árvore composta apenas pela raiz e cujo conteúdo é a freqüência com que esta letra ocorre no texto (veja Figura 1).

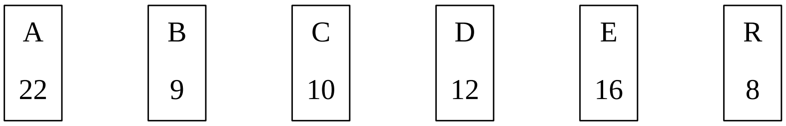

Figura 1. Floresta no início do algoritmo.

Escolha as duas raízes de menor freqüência (árvores com `R` e `B` com freqüências 8 e 9, respectivamente) e junte essas duas árvores, formando uma nova árvore, que fará parte da coleção (veja Figura 2).

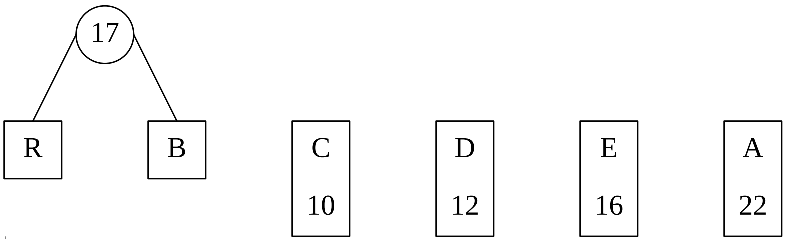

Figura 2. Floresta após termos juntado duas árvores (das freqüências 8 e 9).

Repetindo o processo, escolha as árvores com raízes 10 e 12, e junte-as. A coleção de árvores tem a forma mostrada na Figura 3:

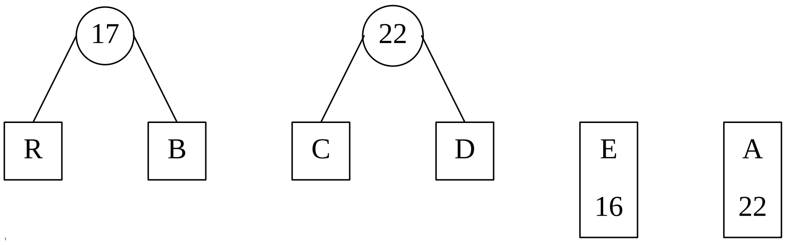

Figura 3. Floresta após termos juntado as árvores com freqüências 10 e 12, formando a árvore com freqüência associada 22.

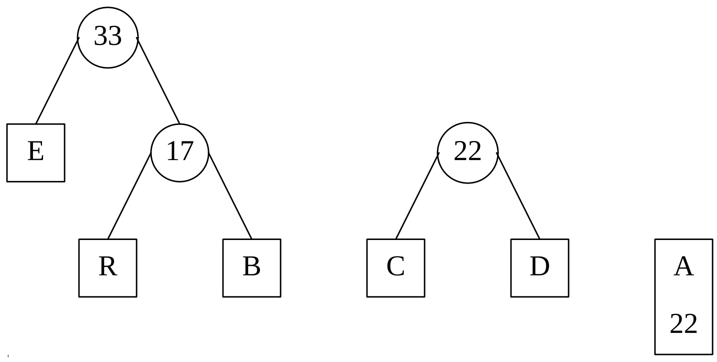

Figura 4. Floresta após termos juntado as árvores com freqüências 16 e 17.

Continuando o mesmo processo, escolha as árvores com freqüências 16 e 17, formando a floresta que pode ser vista na Figura 4.

Depois, juntamos as árvores com freqüências 22 e 22 formando a floresta exibida na Figura 5.

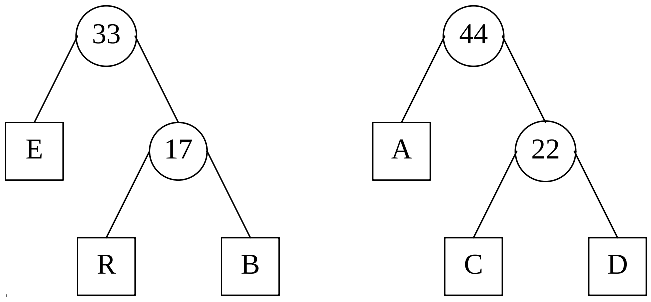

Figura 5. Floresta após termos juntado as árvores com freqüências 22 e 22.

E, finalmente juntamos as duas árvores restantes para obtermos a árvore mostrada na Figura 6.

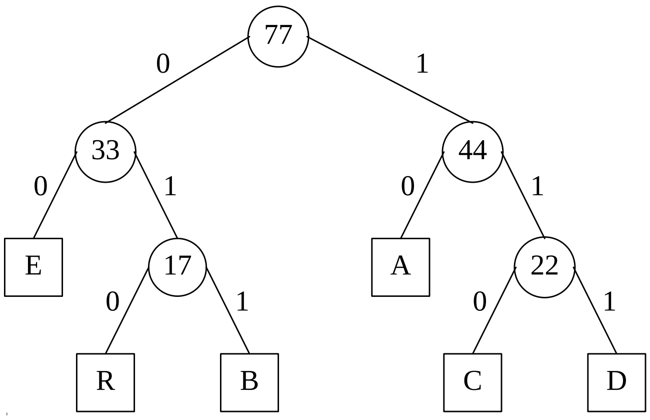

Figura 6. Árvore de Huffman.

Assim, a *árvore de Huffman* está completamente construída.

4. Codificação

Associe 0 às arestas da árvore de Huffman que ligam um nó com seu filho esquerdo e 1, às arestas que ligam um nó com seu filho direito. O código correspondente a cada letra será formado pelo número binário associado ao caminho da raiz até a folha correspondente. Com isso, a tabela de códigos resultante da árvore de Huffman da Figura 6 é:

A	B	C	D	E	R
10	011	110	111	00	010

Repare que na codificação alguns caracteres diminuem de tamanho (menos de 8 bits), mas outros podem aumentar (até 256 bits).

5. Decodificação

Para decodificar uma mensagem obtida através da tabela acima, por exemplo:

$$1\,0\,0\,1\,1\,0\,1\,0\,1\,0\,1\,1\,0\,1\,0\,1\,1\,1\,1\,0\,0\,1\,1\,0\,1\,0\,1\,0,$$

basta ir utilizando cada bit da mensagem para percorrer a árvore de Huffman desde a raiz até alguma folha, quando se obtém o símbolo decodificado. Volte então para a raiz e continue a percorrer a árvore para decodificar o próximo símbolo. Para o exemplo da seqüência de bits acima, o texto correspondente é: `ABRACADABRA`. Observe como este texto foi compactado de 11 bytes (88 bits) para 28 bits.

6. Especificações da implementação

Faça um programa que permita codificar e decodificar arquivos, usando códigos de Huffman.

Codificação: O programa deve ler (e imprimir) um arquivo texto `ep4.dat` e gerar os arquivos:

- `ep4.cod` que é um arquivo de inteiros contendo a codificação de ep4.dat e
- `ep4.arv` que é um arquivo contendo a árvore de Huffman.

A árvore de Huffman deve ser representada utilizando-se alocação dinâmica.

Em cada passo da construção da árvore de Huffman o algoritmo escolhe, dentre uma floresta de árvores com freqüências, as duas árvores com as menores freqüências. Para implementar este passo do algoritmo o seu programa deve utilizar uma fila de prioridades (*heap*). Cada célula da fila de prioridades deve conter a freqüência de uma das árvores da floresta junto com uma referência (apontador) para a raiz da árvore correspondente.

O programa deve imprimir a árvore de Huffman (de alguma maneira) exibindo o código associado a cada caractere e também a tabela com a freqüência de cada caractere que ocorre no arquivo ep4.dat. Imprima também a taxa de redução do tamanho do arquivo (número de bytes do arquivo codificado dividido pela número de bytes do arquivo original).

Decodificação: O programa deve ler os arquivos ep4.cod e ep4.arv, e gerar um arquivo texto `ep4.dec` contendo a decodificação de ep4.cod. Para isto, é preciso reconstruir a árvore Huffman correspondente ao texto—que foi guardada em `ep4.arv`.

O programa deve imprimir a árvore e o arquivo decodificado.

Observação. O seu programa deve permitir a leitura dos nomes dos arquivos `.dat` para codificação e `.cod` e `.arv` para decodificação.

Referências

[1] T.C. Bell, J.G. Cleary, and I.H. Witten, *Text compression*, Prentice Hall, Englewood Cliffs, NJ, 1990.

[2] T.H. Cormen, C.E. Leiserson, and R.L. Rivest, *Introduction to algorithms*, The MIT Press, McGraw-Hill Book Company, 1990, QA758 C811i.

[3] R.M. Fano, *The transmission of information*, Tech. report, Research Laboratories of Electronic, MIT, 1949.

[4] D.A. Huffman, *A method for the construction of minimum-redundancy codes*, Proc. Institute of Eletrical and Radio Engineers, vol. 40, September 1952, pp. 1098–1101.

[5] C.E. Shannon, *The mathematical theory of communication*, Bell System Technical Journal **27** (1948), 379–423, 623–656, Reprinted in Shannon and Weaver [6].

[6] C.E. Shannon and W. Weaver, *The mathematical theory of communication*, University of Illinois Press, Urbana, Illinois, 1949.

[7] J.L. Szwarcfiter and L. Markenzon, *Estruturas de dados e seus algoritmos*, LTC–Livros Técnicos e Científicos Ed., Rio de Janeiro, 1994, QA758 S998e.

[8] R. Terada, *Desenvolvimento de algoritmos e estruturas de dados*, McGraw-Hill, Makron, 1991.