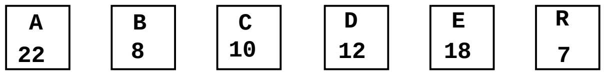
Códigos de Huffman
Terceiro Exercicío Programa de MAC 2301

Entrega: 18 de junho de 2001

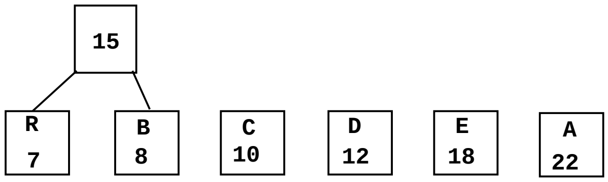
Uma aplicação interessante de árvores binárias em transmissão e compactação de arquivos é dada pelos códigos de Huffman. Como sabemos, as mensagens enviadas por cabos, ou mesmo pelo ar, são muitas vezes codificadas em forma binária. Deste modo, a velocidade de transmissão depende do comprimento da codificação. Torna-se então importante garantir que a mensagem será o mais curta possível. Um algoritmo de determinação de códigos binários para caracteres baseado na sua freqüência de aparição foi sugerido por Huffman. A idéia é associar números binários com menos bits aos caracteres mais usados nos textos. Se você quiser saber mais sobre códigos de Huffman, consulte o ótimo livro [1].

Por exemplo, suponha que a mensagem seja composta pelos caracteres A, B, C, D, E e R e que o número de vezes que estas letras são usadas na mensagem seja resp. 22, 8, 10, 12, 18 e 7. A idéia é construir uma árvore binária baseada na freqüência de uso destas letras de tal forma que as folhas correspondam às letras; e as letras mais freqüentemente usadas (como A e E) estejam mais próximas da raiz que as menos usadas (como B e R). Para isso, construiremos a árvore binária a partir das folhas, colocando nos nós internos um número correspondente à soma da freqüência dos seus descendentes.

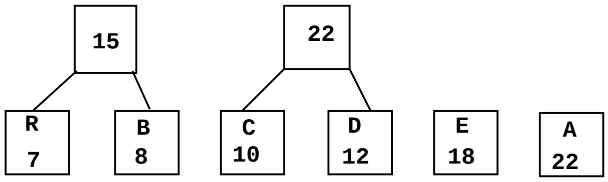
A seguir, mostramos a construção da árvore binária para nosso exemplo. Em cada passo do algoritmo teremos uma coleção de árvores binárias. À raiz de cada uma dessas árvores está associado um número (freqüência de seus descendentes). Escolhemos então, as duas árvores da coleção com menor freqüência e as transformamos numa só, ligando-as a uma nova raiz cujo valor será dado pela soma dos valores das duas subárvores. Observe o exemplo abaixo:



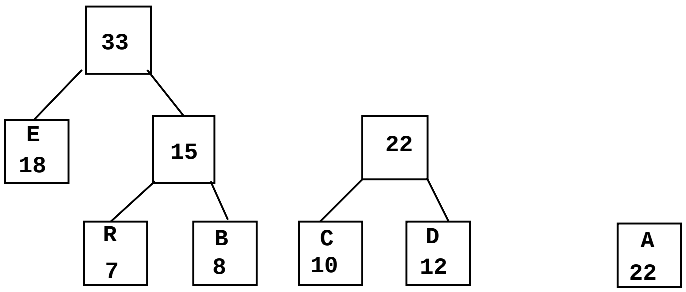
Selecionamos as duas árvores de menor valor (8 e 7) e as juntamos em uma árvore só.



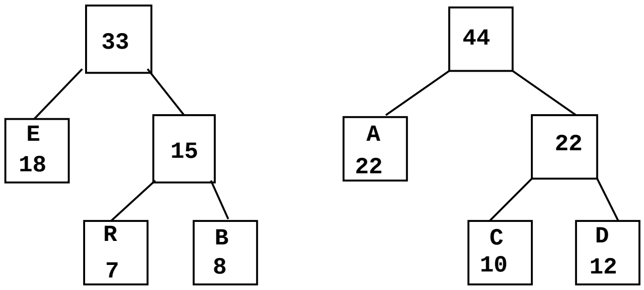
Repetindo o processo, escolhemos as árvores com raiz (10 e 12).



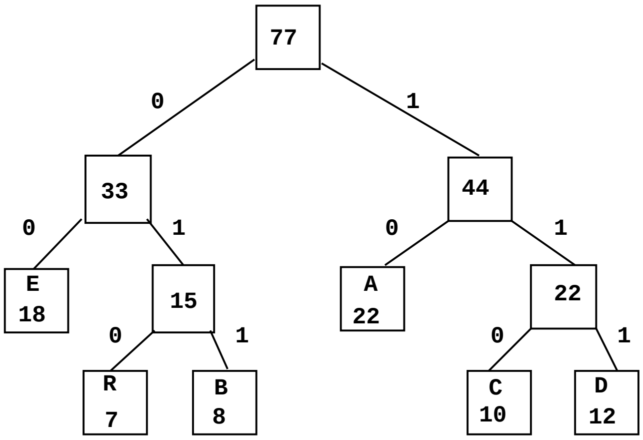
Agora escolhemos (18 e 15),



Depois (22 e 22),



E finalmente (33 e 44).



Associamos, então, à árvore final assim construída 0 às arestas que ligam o vértice com seu filho esquerdo e 1 às arestas que ligam o vértice com seu filho direito. O código correspondente a cada letra será então formado pelo número binário associado ao caminho da raiz até a folha correspondente. No caso acima, temos a seguinte tabela:

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

No exemplo acima foi possíveltamanho da codificação de todos os caracteres quando comparado ao código dado pela tabela ASCII que usa 1 byte para cada caracter, mas dependendo da árvore isso nem sempre acontece.

Utilizando a tabela acima, a mensagem ABRACADABRA será codificada pela seqüência de bits 1001101010110101111001101010. Assim, conseguimos compactar a mensagem original de 88 bits para 28 bits.

O objetivo deste exercício–programa é construir um compactador de arquivos baseado na idéia mostrada acima. Duas estratégias possíveis de compactação são as seguintes:

- árvore fixa.
  Inicialmente lêem-se diversos tipos de textos (programas, arquivos de

dados, textos escritos), contando o número de ocorrências de cada caractere. A partir destas frequências seu programa montará uma árvore como descrito neste enunciado, e usará a mesma árvore para compactar e descompactar os textos.

- árvore variável.
  Para cada texto lido constrói-se a "árvore ótima" para aquele texto, que será compactado usando essa árvore específica. A desvantagem nesse caso é que devemos guardar a árvore junto com o arquivo compactado para descompactar futuramente.

Compare essas duas estratégias (em taxa de compactação, tempo de processamento, etc.) e compare também seu programa com pacotes comerciais (winzip e outros que você quiser) de compactação de textos.

Nesse exercício–programa, além da listagem e do disquete, você deve entregar um pequeno relatório com seus testes comparando as diversas estratégias (pode implementar outras!) e os pacotes comerciais.

Obs: Em breve vamos disponibilizar na página do curso um conjunto de 10 arquivos para testes.

Referências

[1] T. H. Cormen, C. E. Leiserson, R. L. Rivest, *Introduction to Algorithms* (1990), McGraw Hill.

[2] D. A. Huffman, "A Method for the Construction of Minimum Redundancy Codes", *Proceedings IRE* 10, 40, 1098 – 1101 (1952).

[3] D. A. Lelewer, D. S. Hirschberg, "Data Compression", *ACM* 19, 3, 261–296 (1987).