b) *Hashing Duplo*. Desenhe o conteúdo da tabela *hash* resultante da inserção de registros com as chaves Q U E S T A O F C I L, nesta ordem, em uma tabela inicialmente vazia de tamanho 13 (treze) usando *endereçamento aberto* com ***hashing* duplo**. Use a função *hash* $h_1(k) = k$ **mod** 13 para calcular o endereço primário e $j = 1 + (k$ **mod** $11)$ para resolver as colisões, ou seja, para a escolha de localizações alternativas. Logo $h_i(k) = (h_{i-1}(k) + j)$ **mod** 13, para $2 \leq i \leq M$ (Sedgewick, 1988).

18. Considere as seguintes estruturas de dados: *heap*, árvore binária de pesquisa, vetor ordenado, tabela *hash* com solução para **colisões** usando endereçamento aberto, tabela *hash* com solução para colisões usando listas encadeadas.

Para cada um dos problemas abaixo, sugira a estrutura de dados mais apropriada dentre as listadas anteriormente, de forma a minimizar tempo esperado e espaço necessário. Indique o tempo esperado e o espaço necessário em cada escolha e por que a estrutura de dados escolhida é superior às outras.

a) inserir/retirar/encontrar um elemento dado;

b) inserir/retirar/encontrar o elemento de valor mais próximo ao solicitado;

c) coletar um conjunto de registros, processar o maior elemento, coletar mais registros, processar o maior elemento, e assim por diante;

d) mesma situação descrita no item anterior adicionada da operação extra de ajuntar ("merge") duas estruturas.

19. Índice Remissivo.

O objetivo deste trabalho é o de projetar e implementar um sistema de programas, incluindo as estruturas de dados e os algoritmos. Nesse trabalho, o aluno terá a oportunidade de exercitar parcialmente o conceito de independência de implementação, por meio da utilização de duas estruturas de dados distintas para implementar o mesmo problema. Nesse caso, o módulo que implementa cada uma das estruturas de dados deverá permitir o intercâmbio entre uma estrutura e outra, causando o menor impacto possível em outras partes do programa.

Problema: Criação de **índice remissivo**

Várias aplicações necessitam de um relatório de referências cruzadas. Por exemplo, a maioria dos livros apresenta um índice remissivo, que corresponde a uma lista alfabética de palavras-chave ou palavras relevantes do texto com a indicação dos locais no texto onde cada palavra-chave ocorre. Na verdade, o índice remissivo é um **arquivo invertido**, um tipo de índice apresentado na Seção 8.1.

Como exemplo, suponha um arquivo contendo um texto constituído por:

Linha 1: Good programming is not learned from
Linha 2: generalities, but by seeing how significant
Linha 3: programs can be made clean, easy to
Linha 4: read, easy to maintain and modify,

Linha 5: human-engineered, efficient, and reliable,
Linha 6: by the application of common sense and
Linha 7: by the use of good programming practices.

Assumindo que o índice remissivo seja constituído das palavras-chave:

programming, programs, easy, by, human-engineered, and, be, to,

o programa para criação do índice deve produzir a seguinte saída:

and	4 5 6
be	3
by	2 6 7
easy	3 4
human-engineered	5
programming	1 7
programs	3
to	3 4

Note que a lista de palavras-chave está em ordem alfabética. Adjacente a cada palavra está uma lista de números de linhas, um para cada vez que a palavra ocorre no texto. Uma estrutura de dados desse tipo é conhecida como **arquivo invertido**. O arquivo invertido é um mecanismo muito utilizado em arquivos constituídos de texto, como as **máquinas de busca na Web**.

Projete um sistema para produzir um índice remissivo. O sistema deverá ler um número arbitrário de palavras-chave que deverão constituir o índice remissivo, seguido da leitura de um texto de tamanho arbitrário, que deverá ser esquadrinhado à procura de palavras que pertençam ao índice remissivo. Para extrair as palavras de um texto, utilize o procedimento ExtraiPalavra mostrado no Programa 5.43.

Cabe ressaltar que:

a) Uma palavra é definida como uma sequência de letras e dígitos, começando com uma letra;

b) Apenas os primeiros c_1 caracteres devem ser retidos nas chaves. Assim, duas palavras que não diferem nos primeiros c_1 caracteres são consideradas idênticas;

c) Palavras constituídas por menos do que c_1 caracteres devem ser preenchidas por um número apropriado de brancos.

Utilize um método eficiente para verificar se uma palavra lida do texto pertence ao índice. Para resolver esse problema, você deve utilizar duas estruturas de dados distintas:

a) Implementar o índice como uma árvore de pesquisa;

b) Implementar o índice como uma tabela *hash*, usando o método *hashing* linear para resolver **colisões**.