Quanto maior for o valor de $k$ mais compacta é a função *hash* perfeita mínima resultante. Assim, os usuários podem permutar espaço por tempo de avaliação variando o valor de $k$ na implementação. Entretanto, o melhor é utilizar valores para $k$ que sejam potências de dois (por exemplo, $k = 2^{b_k}$ para alguma constante $b_k$), o que permite trocar as operações de multiplicação, divisão e módulo pelas operações de deslocamento de *bits* à esquerda, à direita, e "*and*" binário, respectivamente. O valor $k = 256$ produz funções compactas e o número de *bits* para codificar $k$ é $b_k = 8$.

O Programa 5.42 mostra o procedimento para computar o valor da função *hash* perfeita mínima *hpm* para uma tabela com intervalo $[0, N - 1]$.

***Programa 5.42** Função de transformação perfeita mínima*

```
function hpm (Chave       : TipoChave;
              r           : Tipor;
              var Pesos   : TipoTodosPesos;
              var g       : Tipog;
              var Tr      : TipoTr;
              k           : TipoK;
              var TabRank: TipoTabRank): TipoIndice;
var i, j, u, Rank, Byteg: TipoIndice;
begin
  u := hp (Chave, r, Pesos, g);
  j := u div k;        Rank := TabRank[j];
  i := j * k;          j := i;
  Byteg := j div 4;    j := j + 4;
  while (j < u) do
    begin
    Rank := Rank + Tr[g[Byteg]];
    j := j + 4;   Byteg := Byteg + 1;
    end;
  j := j - 4;
  while (j < u) do
    begin
    if (ObtemValor2Bits (g,j) <> NAOATRIBUIDO) then Rank := Rank+1;
    j := j + 1;
    end;
  hpm := Rank;
end; { hpm }
```

**Análise** Segundo Botelho (2008, p. 33), quando $M = cN$, $c > 2$ e $r = 2$, a probabilidade $P_{r_a}$ de gerar aleatoriamente um 2-grafo bipartido acíclico $G_2 = (V, A)$, para $N \to \infty$, é:

$$P_{r_a} = \sqrt{1 - \left(\frac{2}{c}\right)^2}.$$



Por exemplo, quando $c = 2,09$, temos que $P_{r_a} = 0,29$. Logo, o número esperado de iterações para gerar um 2-grafo bipartido acíclico é $1/P_{r_a} = 1/0,29 \approx 3,45$. Isso significa que, em média, aproximadamente 3,45 grafos serão testados antes que apareça um 2-grafo bipartido acíclico para ser usado na geração da função de transformação. Novamente o custo para gerar cada grafo é linear no número de arestas do grafo. Botelho (2008) mostra também que é possível obter um 3-grafo 3-partido acíclico em 1 tentativa com probabilidade tendendo para 1 quando $N$ tende para infinito, sempre que $M = cn$ e $c \geq 1{,}23$. Logo, o custo para gerar cada grafo é linear no número de arestas do grafo.

O procedimento GrafoAciclico para verificar se um hipergrafo é acíclico do Programa 7.10 tem complexidade $O(|V| + |A|)$. Como $|A| = O(|V|)$ para grafos esparsos como os considerados aqui, a complexidade de tempo para gerar a função de transformação é proporcional ao número de chaves a serem inseridas na tabela *hash*.

O tempo necessário para avaliar a função *hp* da Eq.(5.1) envolve a avaliação de três funções *hash* universais, com um custo final $O(1)$. O tempo necessário para avaliar a função *hpm* da Eq.(5.2) tem um custo final $O(1)$. Para isso utilizamos uma **estrutura de dados sucinta** que permite computar em tempo constante o número de posições atribuidas antes de uma dada posição em um arranjo. A tabela $T_r$, gerada pelo Programa 5.41, permite contar o número de vértices atribuídos em $\epsilon \log M$ *bits* com custo $O(1/\epsilon)$, onde $0 < \epsilon < 1$. Mais ainda, a avaliação da função *rank* é muito eficiente já que, para diversas aplicações, tanto TabRank quanto $T_r$ cabem inteiramente na memória *cache* da CPU.

A seguir vamos determinar o espaço ocupado pela Eq.(5.1) para calcular o valor da função *hash* perfeita *hp*. Como somente quatro valores distintos são armazenados em cada entrada do arranjo $g$, então são necessários apenas 2 *bits* por valor armazenado. Como o tamanho de $g$ para um 3-grafo é $M = cN$, onde $c = 1{,}23$, o espaço necessário para armazenar o arranjo $g$ é de $2cn = 2{,}46$ *bits* por entrada. Logo, uma função *hash* perfeita pode ser armazenada em aproximadamente 2,46 *bits* por chave.

A seguir vamos determinar o espaço ocupado pela Eq.(5.2) para calcular o valor da função *hash* perfeita mínima *hpm*. Nesse caso, o espaço necessário para armazenar a função é igual ao espaço necessário para o arranjo $g$ mais o espaço para a tabela TabRank. Isso se traduz na seguinte equação:

$$|g| + |\text{TabRank}| = 2cn + 32 * (cn/k),$$

assumindo que cada uma das $cn/k$ entradas da tabela TabRank armazena um inteiro de 32 *bits* e que cada uma das $cn$ entradas de $g$ armazena um inteiro de 2 *bits*. Se tomarmos $k = 256$, teremos:

$$2cn + (32/256)cn = (2 + 1/8)cn = (2 + \epsilon)cn, \text{ para } \epsilon = 1/8 = 0.125.$$