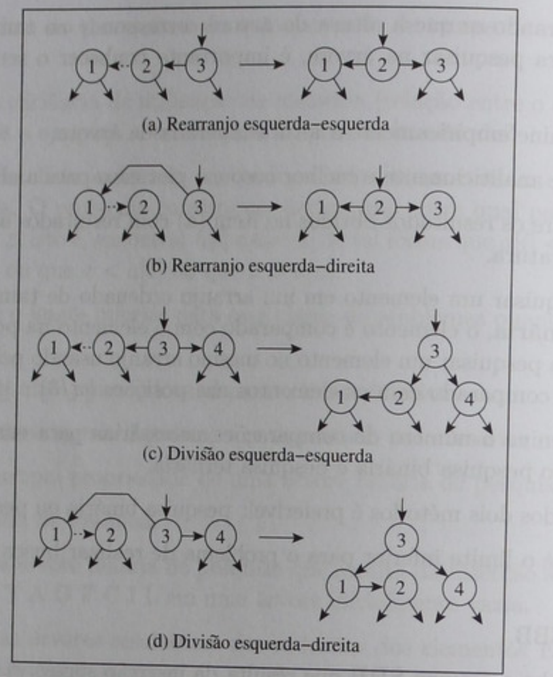

Figura 5.21 *Transformações propostas por Olivié (1980).*

Bayer (1972), Olivié (1980) e também Wirth (1976) usaram dois *bits* por nó em suas implementações para indicar se os apontadores às subárvores direita e esquerda são horizontais ou verticais. Entretanto, apenas um *bit* é necessário: a informação indicando se o apontador à direita (esquerda) é horizontal ou vertical pode ser armazenada no filho à direita (esquerda). Além do fato de demandar menos espaço em cada nó, o retorno ao longo do caminho de pesquisa para procurar por dois apontadores horizontais pode ser terminado mais cedo, porque a informação sobre o tipo de apontador que leva a um nó é disponível sem a necessidade de retornar até seu pai.

Implemente as novas transformações mostradas na Figura 5.21. Utilize apenas 1 *bit* por nó para manter a informação sobre a inclinação dos apontadores.

12. Quais as características de uma boa função *hash*?

13. Implemente uma função para achar o k-ésimo elemento de um dicionário implementado como (Guedes Neto, 2010):

a) árvore binária de pesquisa;

b) tabela *hash*.

Quais as características de uma boa função *hash*?

14. Um dos métodos utilizados para se organizar dados é pelo uso de tabelas *hash*.

a) Em que situações a tabela *hash* deve ser utilizada?

b) Descreva dois mecanismos diferentes para resolver o problema de **colisões** de várias chaves em uma mesma posição da tabela. Quais são as vantagens e desvantagens de cada mecanismo?

15. Em uma tabela *hash* com cem entradas, as **colisões** são resolvidas usando listas encadeadas. Para reduzir o tempo de pesquisa, decidiu-se que cada lista seria organizada como uma árvore binária de pesquisa. A função utilizada é $h(k) = k \bmod 100$. Infelizmente, as chaves inseridas seguem o padrão $k_i = 50i$, onde k_i corresponde à i-ésima chave inserida.

a) Mostre a situação da tabela após a inserção de k_i, com $i = 1, 2, \cdots, 13$. (Faça o desenho.)

b) Depois que mil chaves são inseridas de acordo com o padrão acima, inicia-se a inserção de chaves escolhidas de forma randômica (isto é, não seguem o padrão das chaves já inseridas). Assim, responda:

i) Qual é a ordem do pior caso (isto é, o maior número de comparações) para se inserir uma chave?

ii) Qual é o número esperado de comparações para se inserir uma chave? (Assuma que cada uma das cem entradas da tabela é igualmente provável de ser endereçada pela função h.)

16. *Hashing.*

Substitua XXXXXXXXXXXX pelas 12 primeiras letras do seu nome, desprezando brancos e letras repetidas, nas duas partes dessa questão. Para quem não tiver doze letras diferentes no nome, completar com as letras PQRSTUVWXYZ, nesta ordem, até completar 12 letras. Por exemplo, eu deveria escolher N I V O Z A P Q R S T U. A segunda letra I de NIVIO não entra porque ela já apareceu antes, e assim por diante (Árabe, 1992).

a) Desenhe o conteúdo da tabela *hash* resultante da inserção de registros com as chaves XXXXXXXXXXX, nesta ordem, em uma tabela inicialmente vazia de tamanho 7 (sete), usando listas encadeadas. Use a função *hash* $h(k) = k \bmod 7$ para a k-ésima letra do alfabeto.

b) Desenhe o conteúdo da tabela *hash* resultante da inserção de registros com as chaves XXXXXXXXXXX, nesta ordem, em uma tabela inicialmente vazia de tamanho 13 (treze), usando *endereçamento aberto* e *hashing* linear para resolver as **colisões**. Use a função *hash* $h(k) = k \bmod 13$ para a k-ésima letra do alfabeto.

17. *Hashing* – Endereçamento aberto.

a) *Hashing* Linear. 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* linear para a escolha de localizações alternativas. Use a função *hash* $h(k) = k$ **mod** 13 para a k-ésima letra do alfabeto.