Estruturas de Dados II

Árvore B: remoção

Prof^a. Juliana de Santi
Prof. Rodrigo Minetto
Universidade Tecnológica Federal do Paraná
Material compilado de: Cormen, Notas de aula IC-UNICAMP e IME-USP

Sumário

Árvore B - Remoção

A operação de **remoção** em uma árvore B é um pouco mais complicada do que a operação de inserção. Note que uma chave pode ser eliminada de qualquer nó – não apenas de uma folha – e a eliminação de uma chave qualquer pode exigir que os nós sejam reorganizados.

Árvore B - Remoção

Assim como na inserção, onde tivemos de garantir que um nó não se tornasse grande demais, devemos nos assegurar que um nó não ficará **pequeno demais** durante a eliminação.

Árvore B - Remoção

Importante: lembre-se que em uma Árvore B, com **exceção** da raiz que pode ter menos que o número mínimo de **t - 1** chaves, todas as outras chaves da árvore devem respeitar a regra de um **mínimo** de **t - 1** chaves e **máximo** de **2t - 1** chaves.

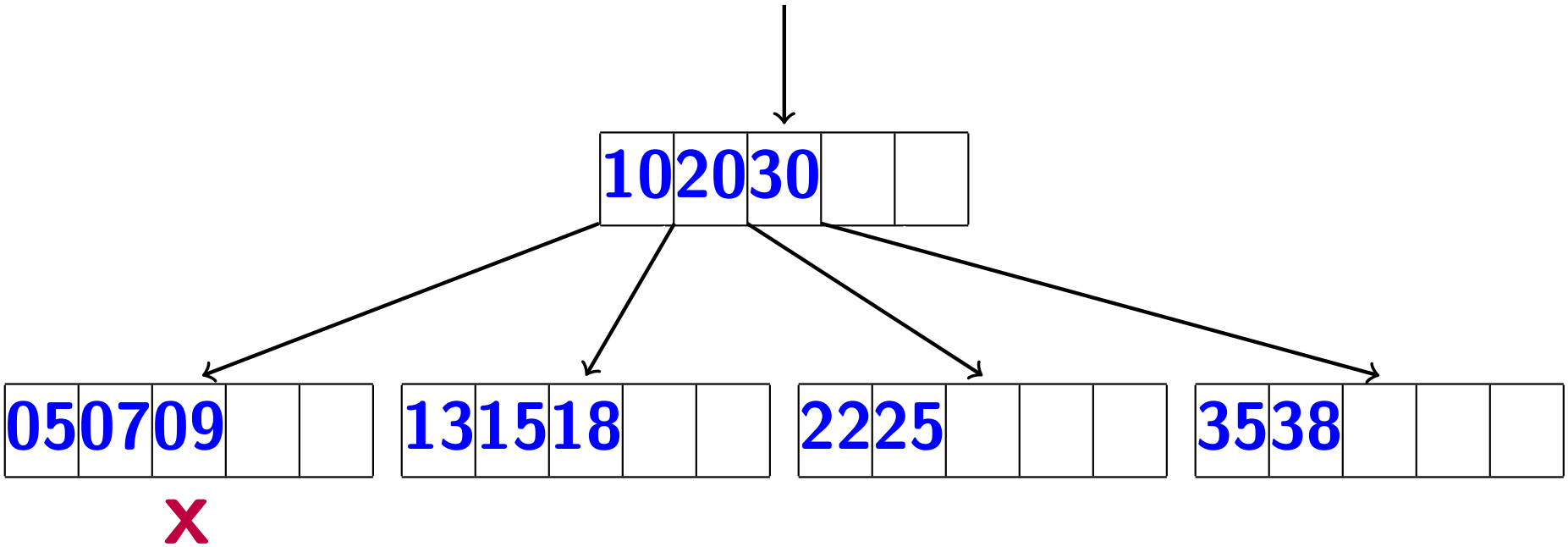
Sumário

Árvore B - Remoção

Existem, **três casos gerais** de eliminação de chaves em uma **árvore B**: **(1)** remoção simples de chave em um nó folha; **(2)** remoção de chave em nó interno; **(3)** remoção de chave em nó folha ou interno que necessita de reorganização. Estes casos são divididos em situações especiais conforme detalhado a seguir.

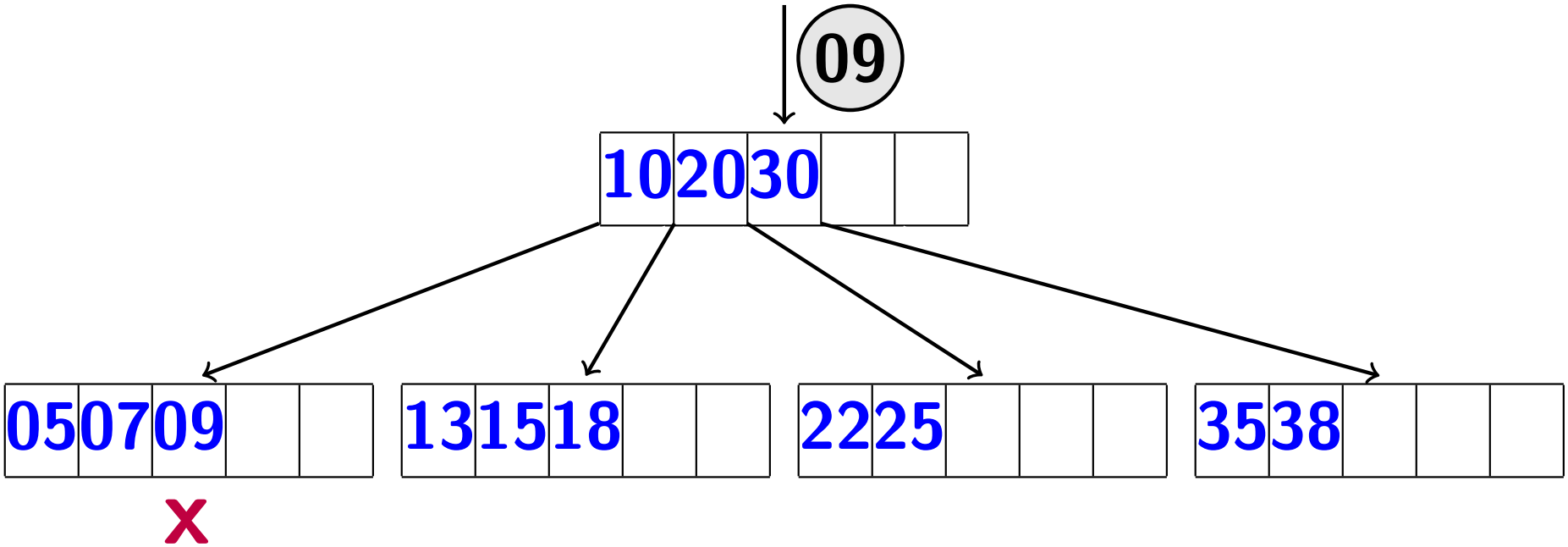
Árvore B - Remoção

Caso 1) se a chave **k** está no nó **x** e **x** é uma folha, elimine a chave **k** de **x**.



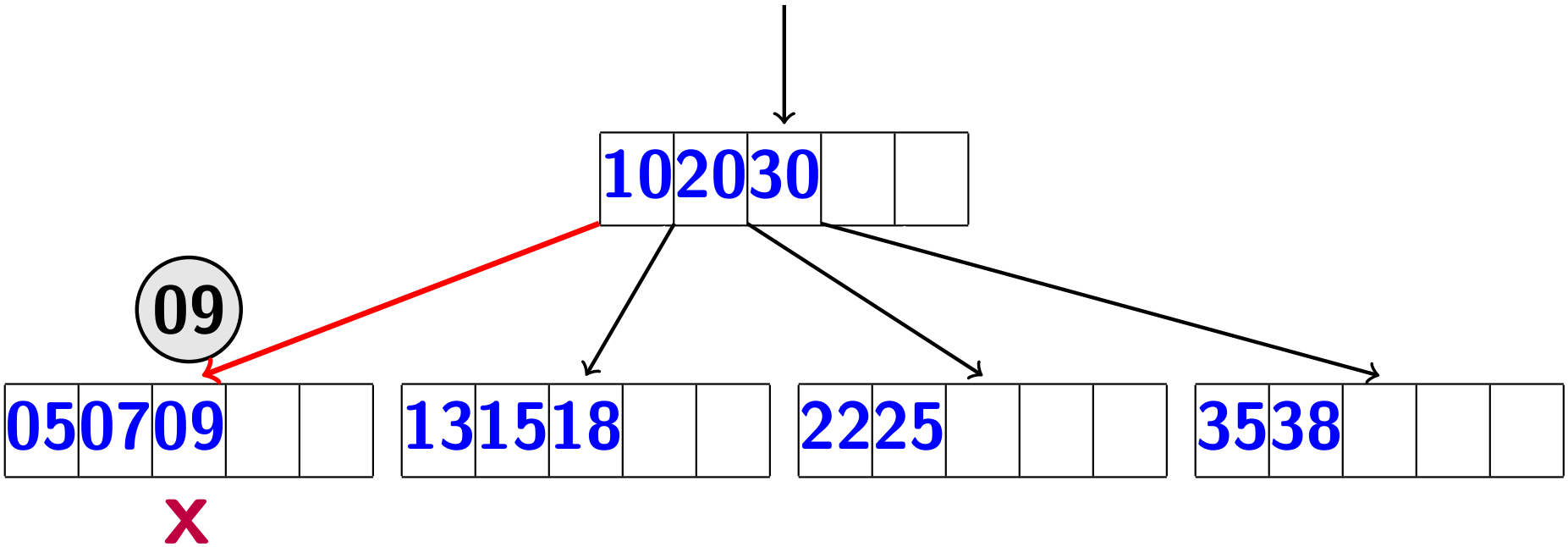
Árvore B - Remoção

Caso 1) se a chave **k** está no nó **x** e **x** é uma folha, elimine a chave **k** de **x**.



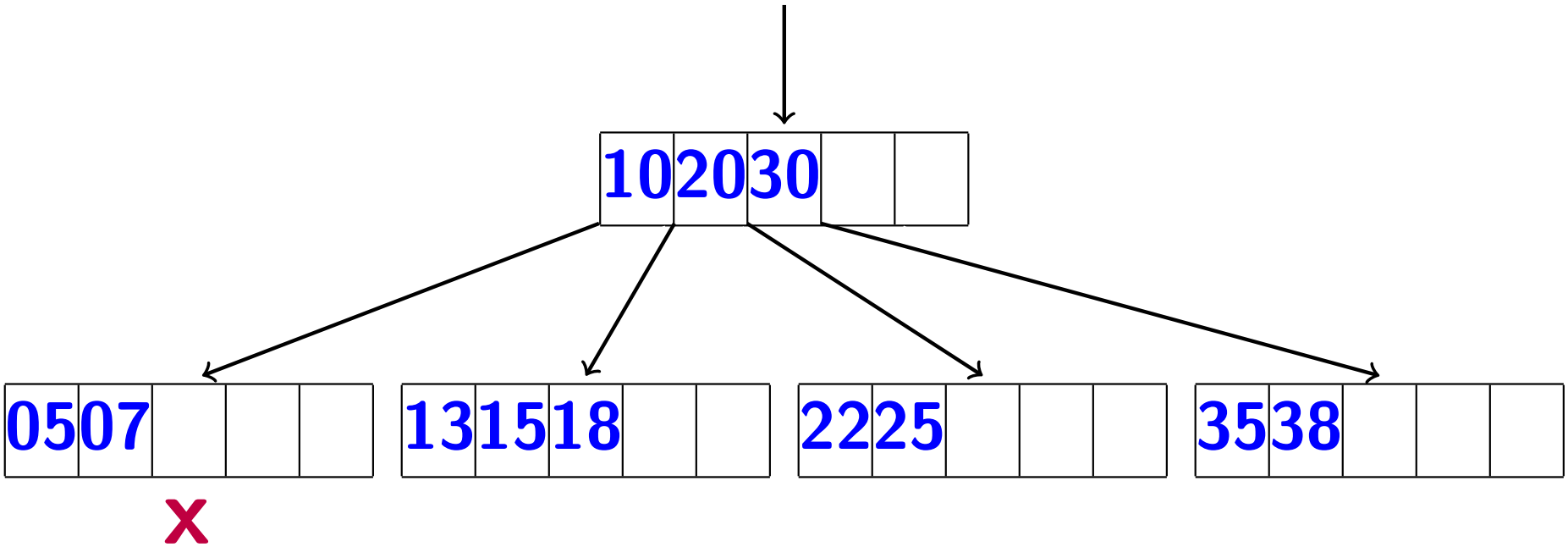
Árvore B - Remoção

Caso 1) se a chave **k** está no nó **x** e **x** é uma folha, elimine a chave **k** de **x**.



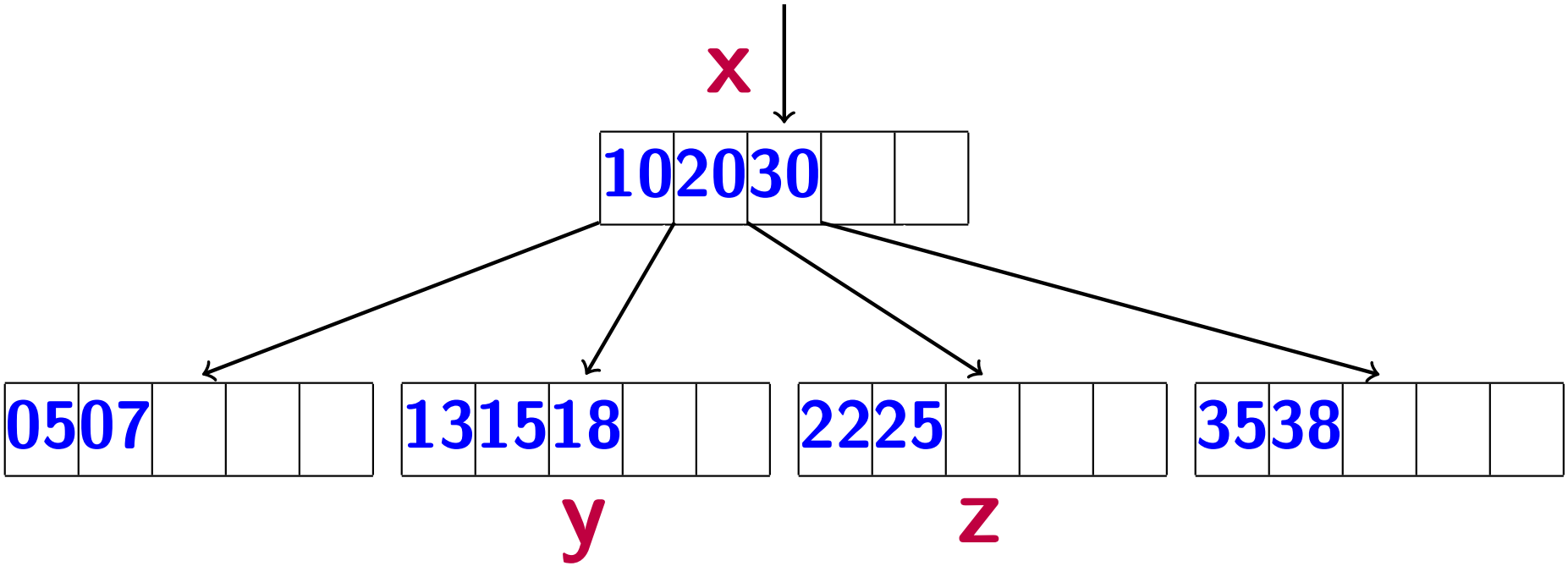
Árvore B - Remoção

Caso 1) se a chave **k** está no nó **x** e **x** é uma folha, elimine a chave **k** de **x**.



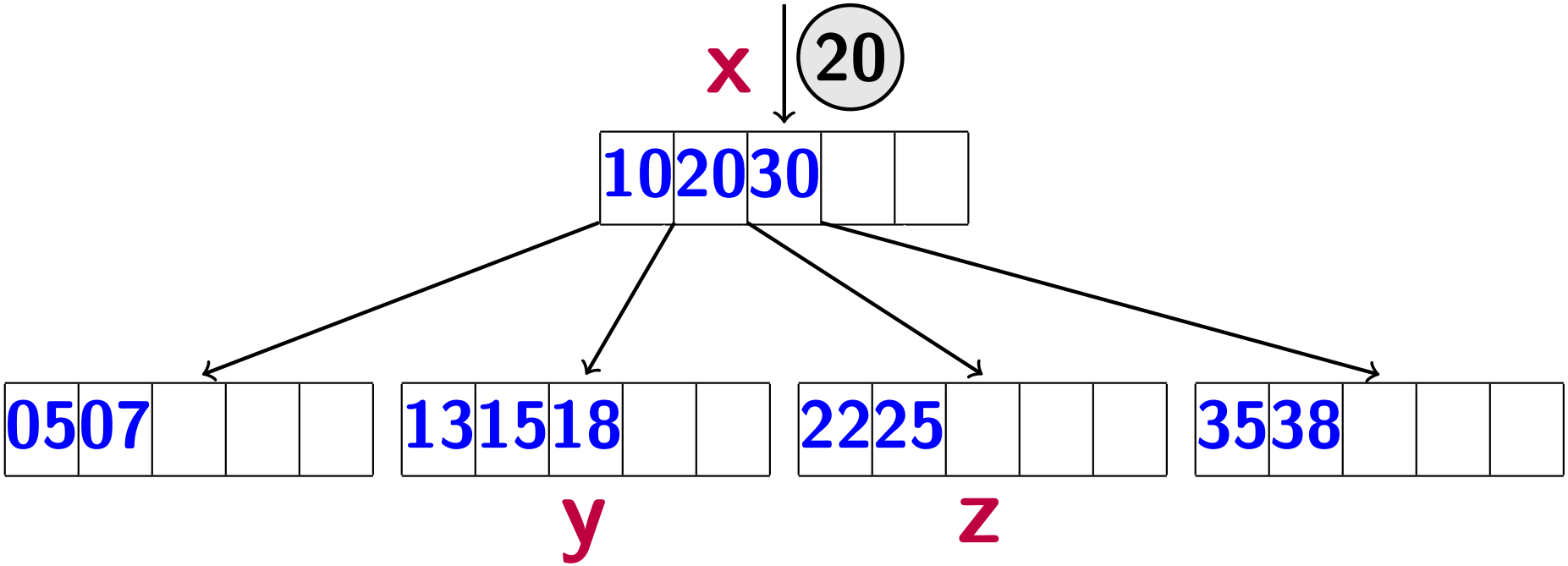
Árvore B - Remoção

Caso 2A ou 2B) se a chave **k** está nó **x**, **x** é um nó interno e algum dos filhos de **k** tem pelo menos t chaves ...



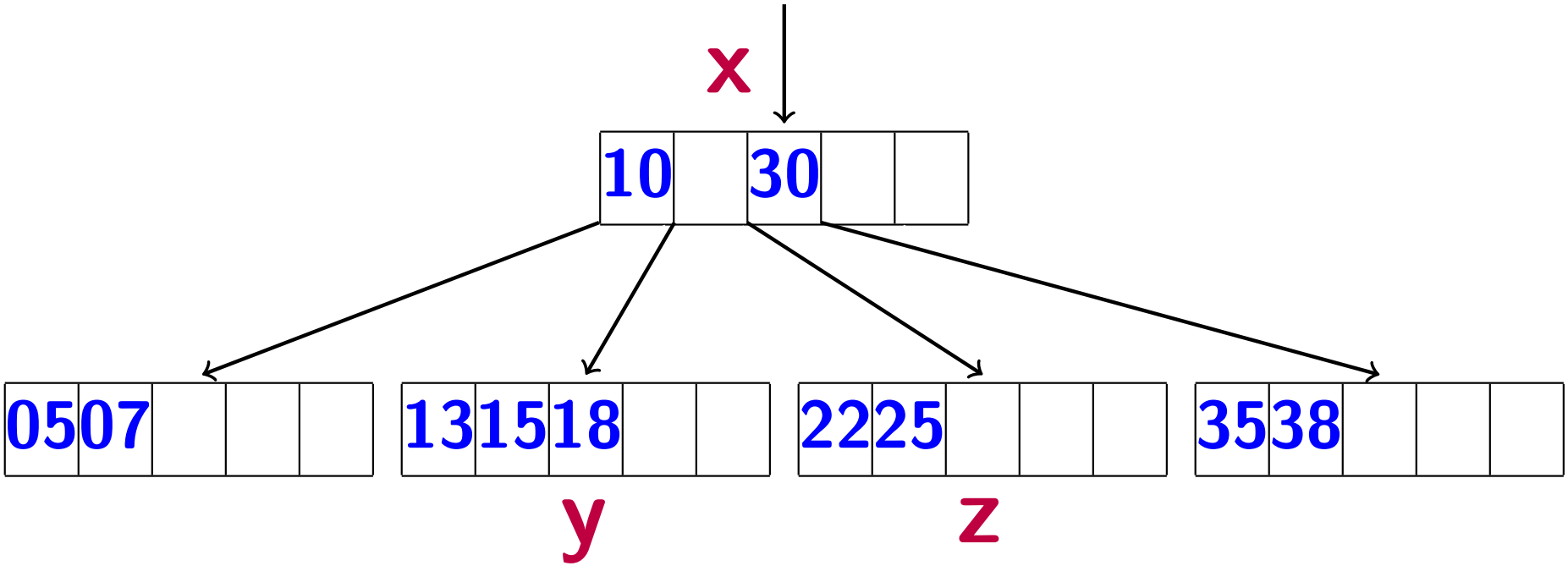
Árvore B - Remoção

Caso 2A ou 2B) se a chave **k** está nó **x**, **x** é um nó interno e algum dos filhos de **k** tem pelo menos t chaves ...



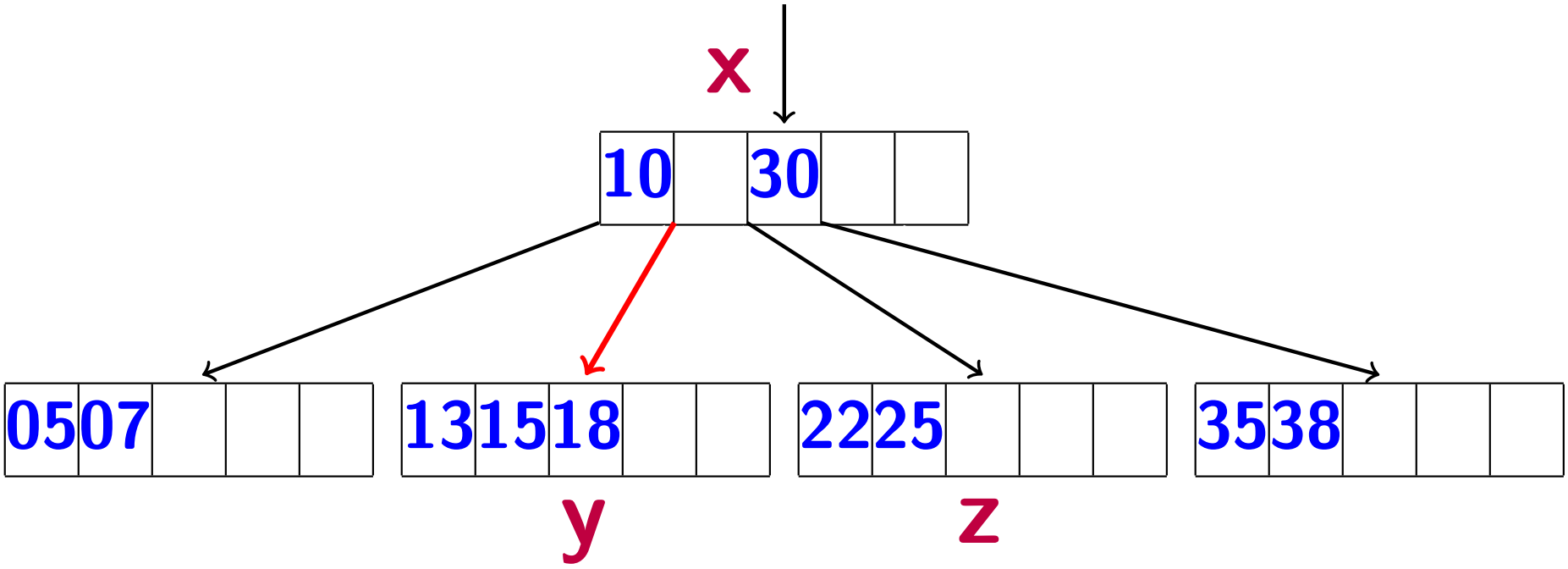
Árvore B - Remoção

Caso 2A ou 2B) se a chave **k** está nó **x**, **x** é um nó interno e algum dos filhos de **k** tem pelo menos t chaves ...



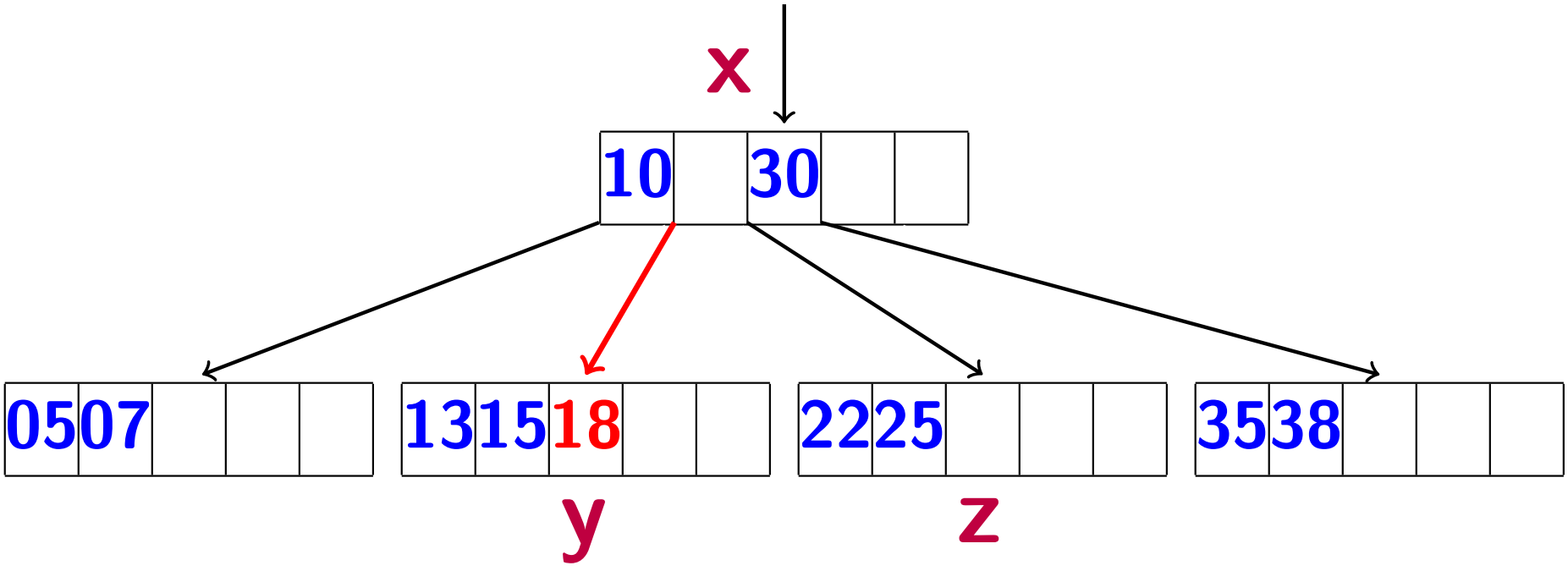
Árvore B - Remoção

Caso 2A ou 2B) se a chave **k** está nó **x**, **x** é um nó interno e algum dos filhos de **k** tem pelo menos t chaves ...



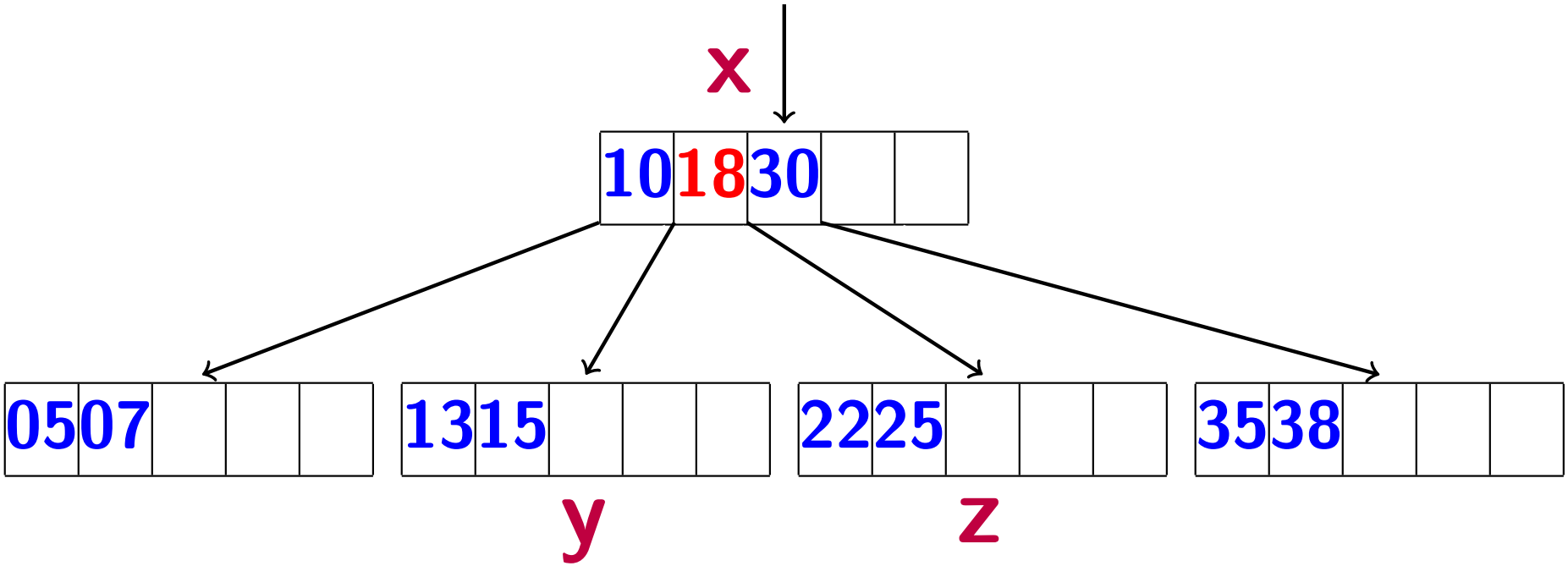
Árvore B - Remoção

Caso 2A ou 2B) se a chave **k** está nó **x**, **x** é um nó interno e algum dos filhos de **k** tem pelo menos t chaves ...



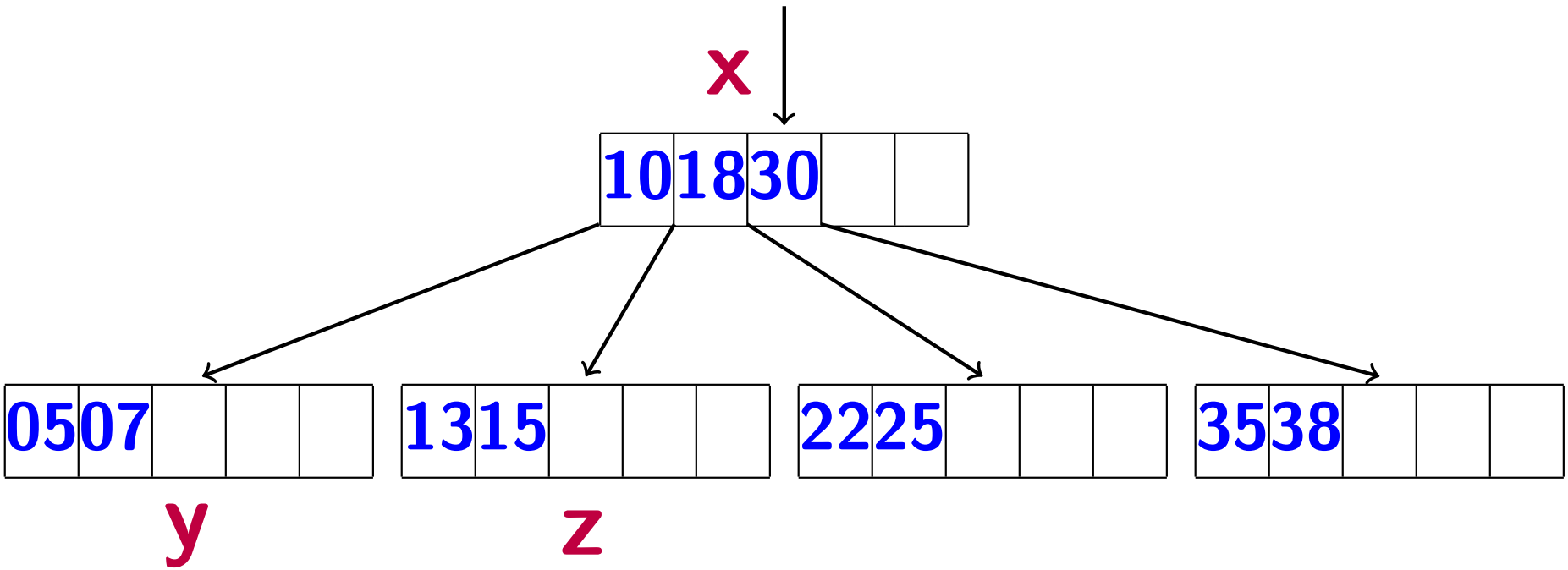
Árvore B - Remoção

Caso 2A ou 2B) se a chave **k** está nó **x**, **x** é um nó interno e algum dos filhos de **k** tem pelo menos t chaves ...



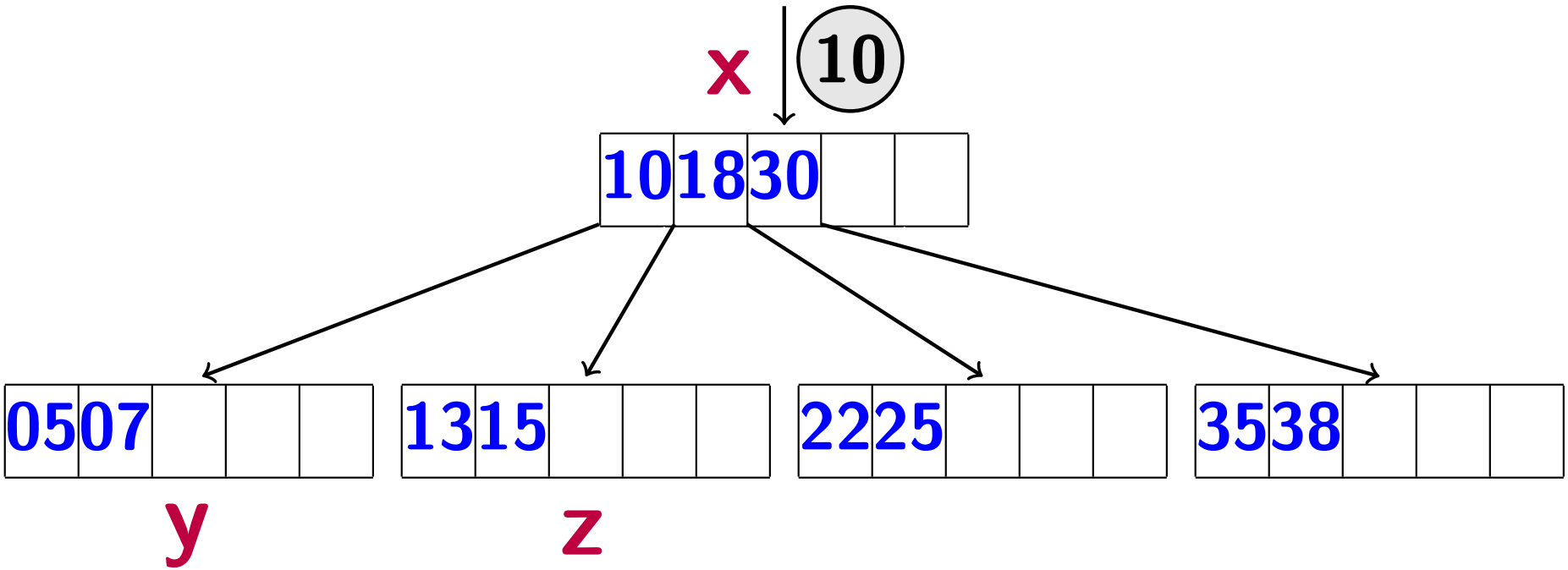
Árvore B - Remoção

Caso 2C) se a chave **k** está nó **x** e **x** é um nó interno, mas os filhos têm $t - 1$ chaves.



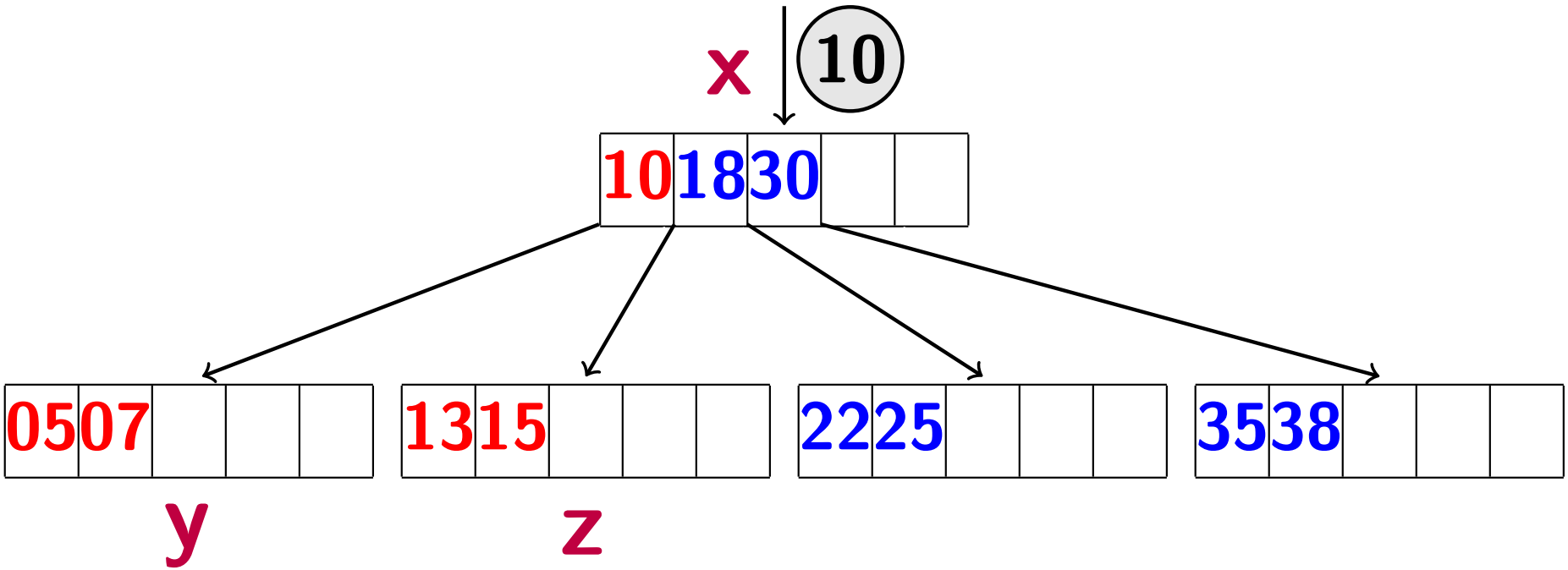
Árvore B - Remoção

Caso 2C) se a chave **k** está nó **x** e **x** é um nó interno, mas os filhos têm $t - 1$ chaves.



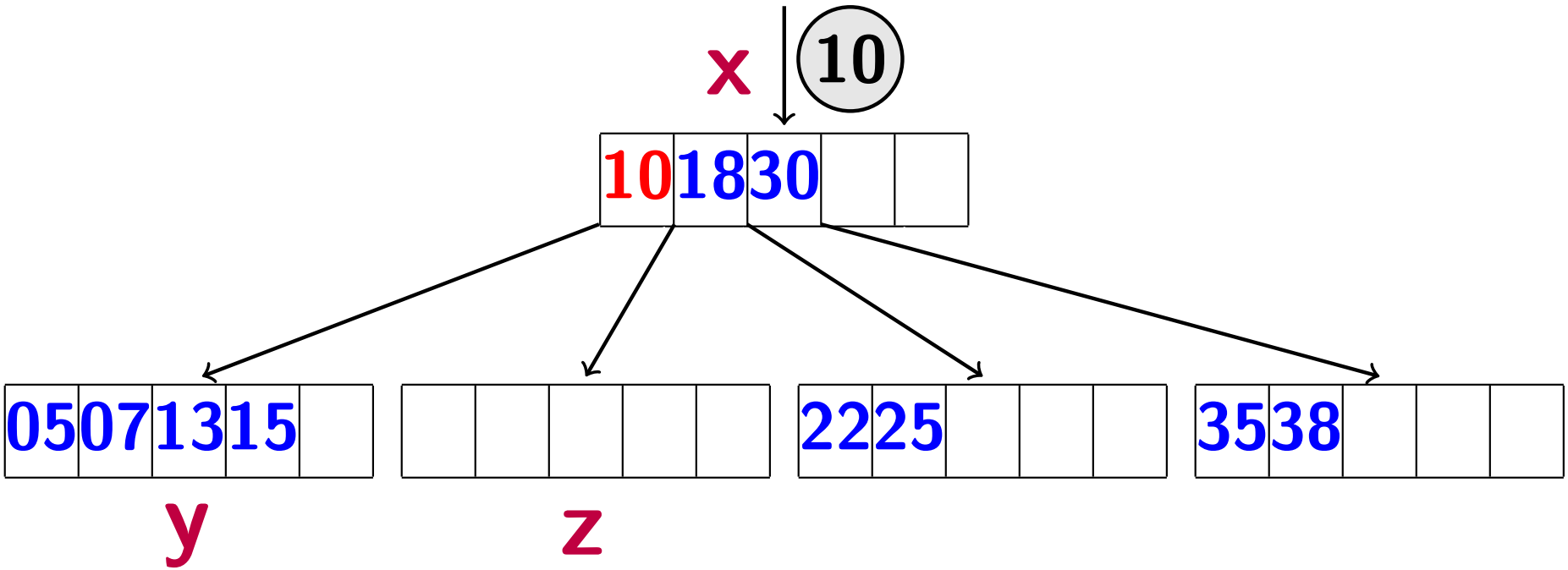
Árvore B - Remoção

Caso 2C) se a chave **k** está nó **x** e **x** é um nó interno, mas os filhos têm $t - 1$ chaves.



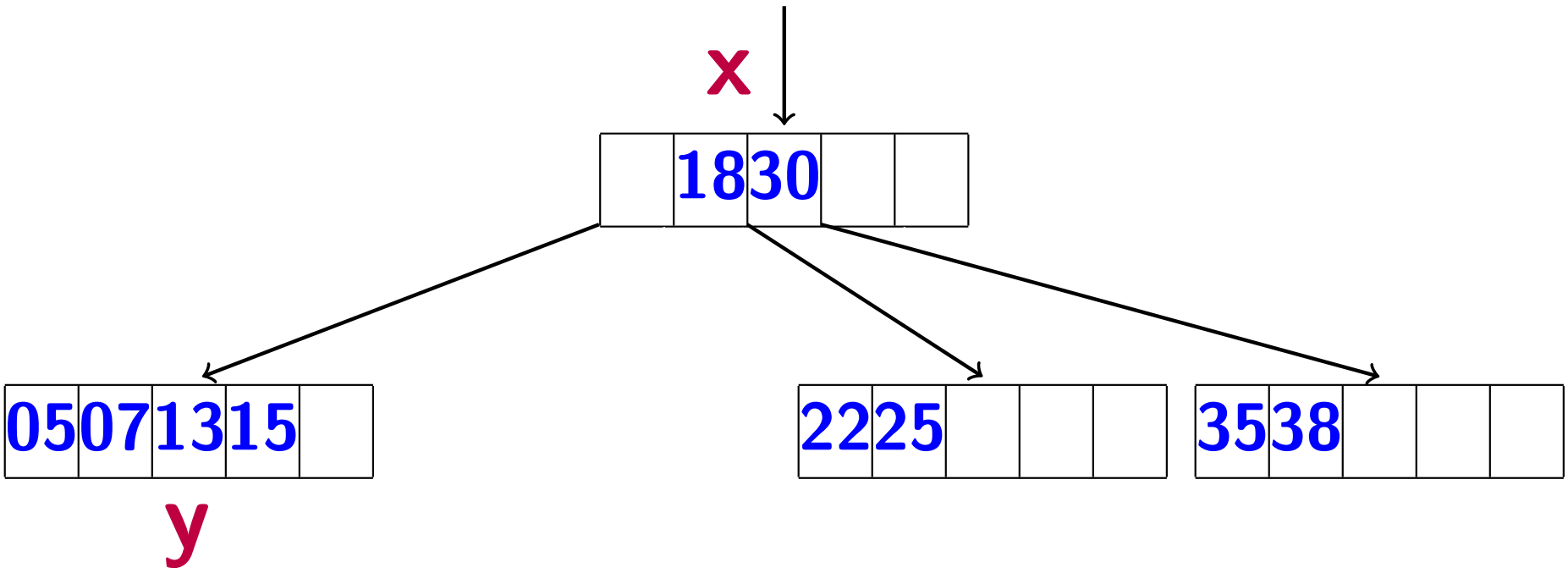
Árvore B - Remoção

Caso 2C) se a chave **k** está nó **x** e **x** é um nó interno, mas os filhos têm $t - 1$ chaves.



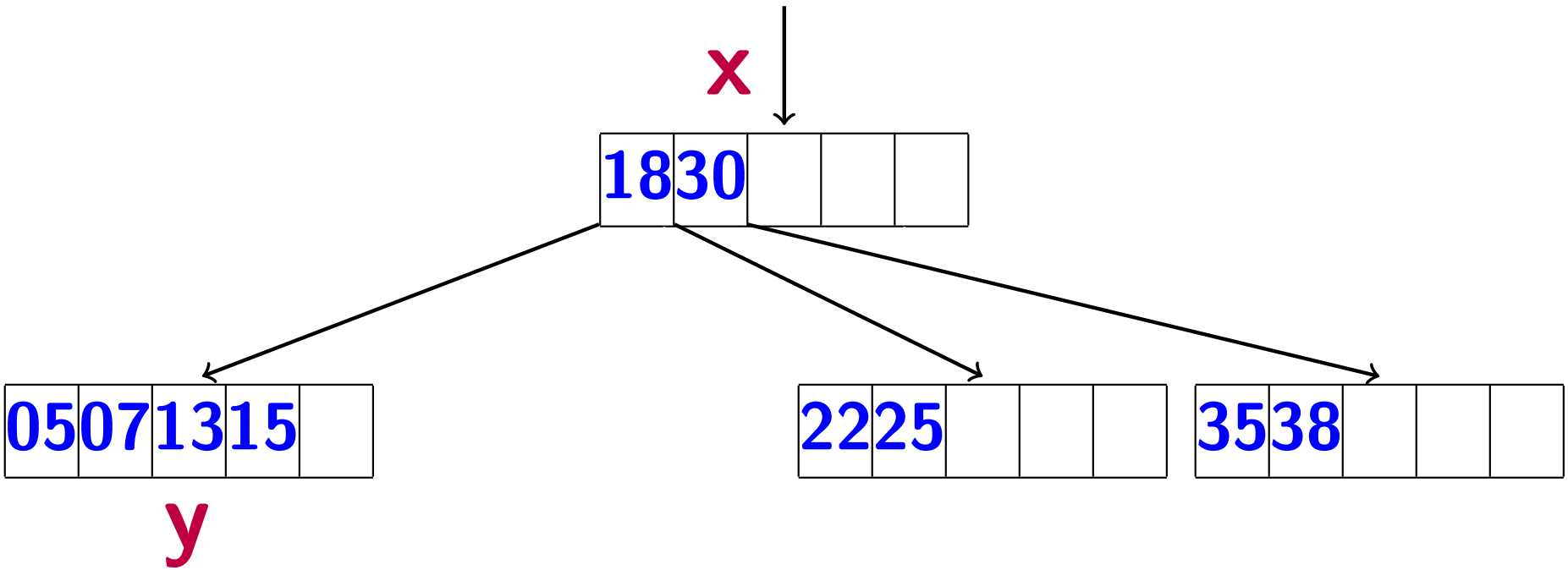
Árvore B - Remoção

Caso 2C) se a chave **k** está nó **x** e **x** é um nó interno, mas os filhos têm $t - 1$ chaves.



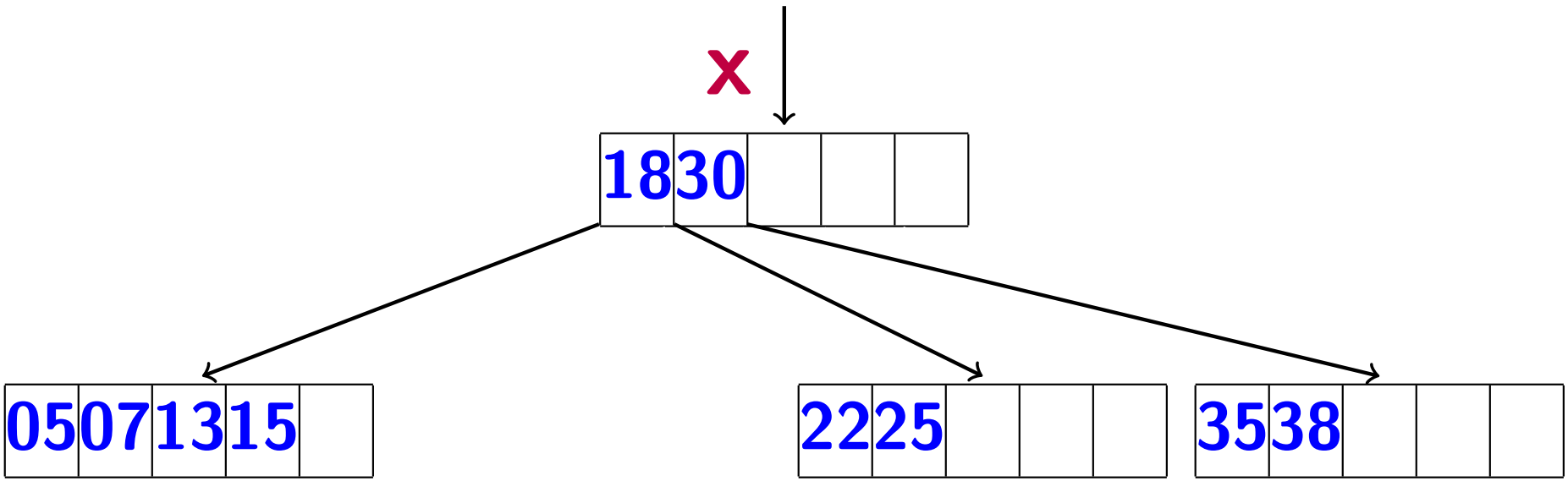
Árvore B - Remoção

Caso 2C) se a chave **k** está nó **x** e **x** é um nó interno, mas os filhos têm $t - 1$ chaves.



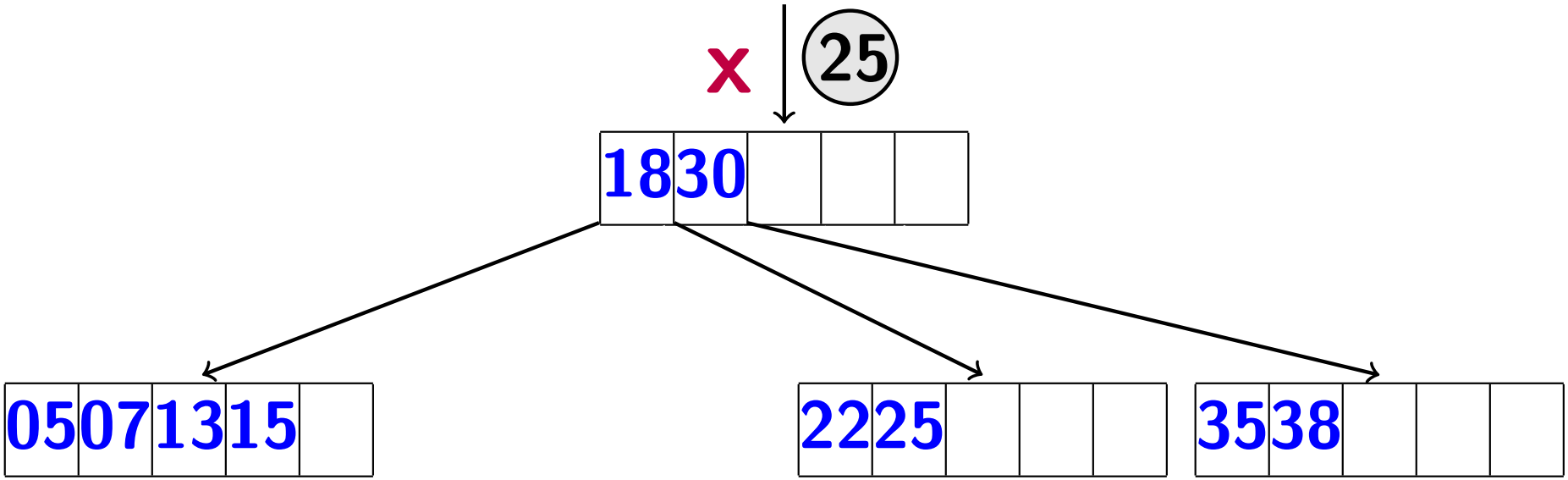
Árvore B - Remoção

Caso 3A) se a chave **k** não está no nó interno **x**, mas presente em um filho com $t - 1$ chaves:



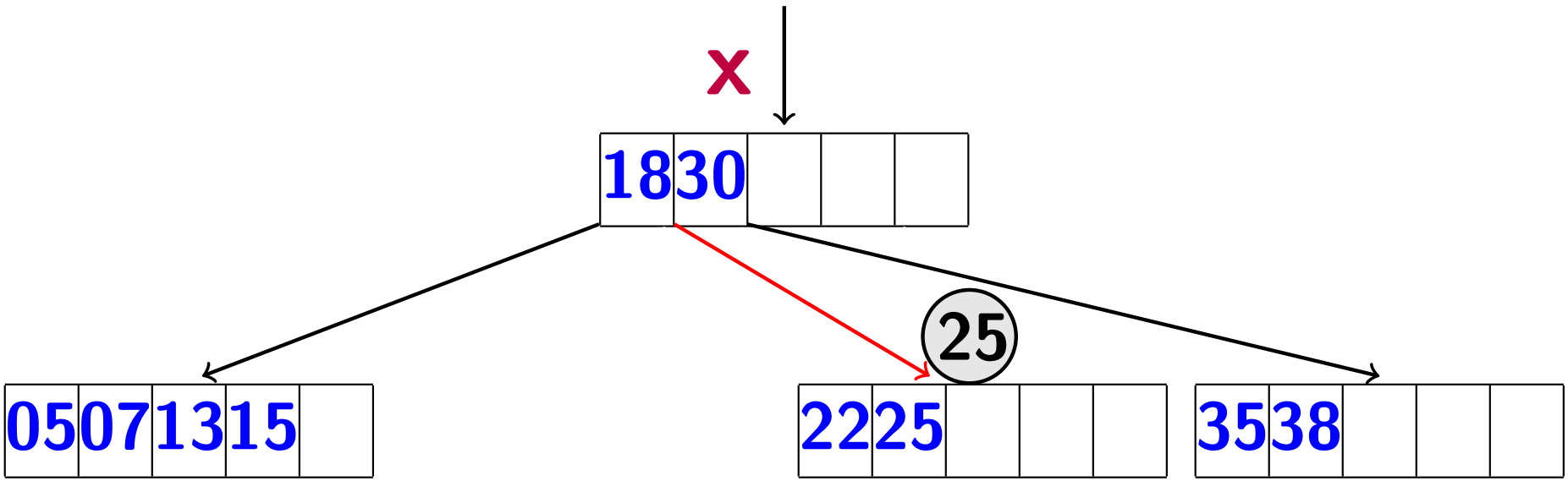
Árvore B - Remoção

Caso 3A) se a chave **k** não está no nó interno **x**, mas presente em um filho com $t - 1$ chaves:



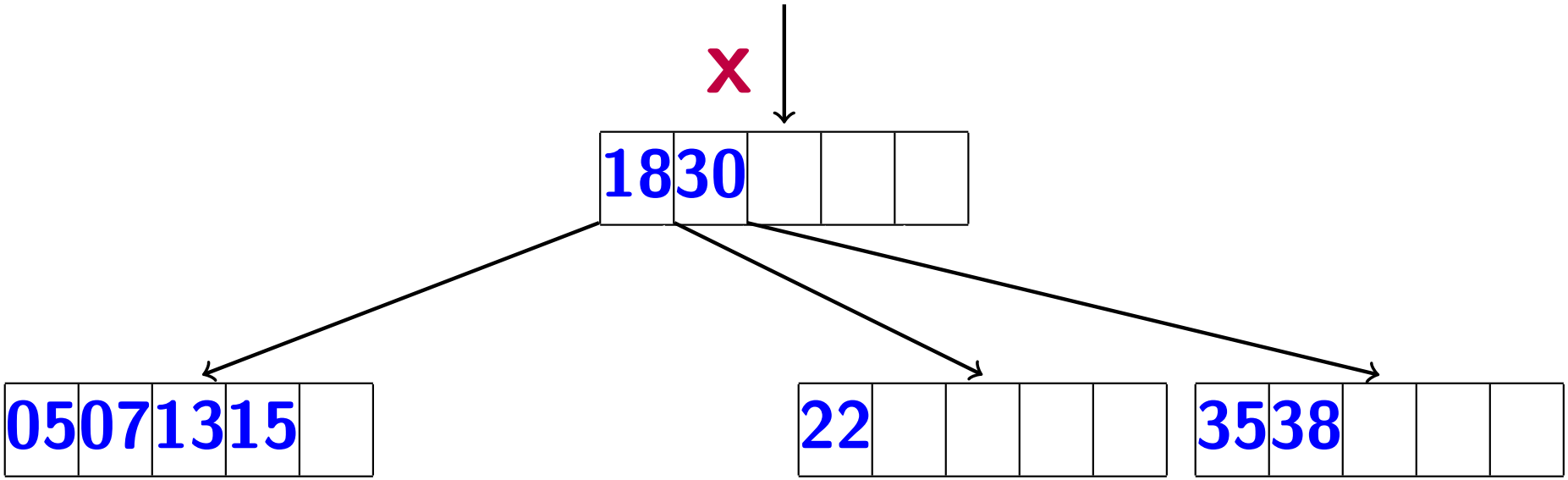
Árvore B - Remoção

Caso 3A) se a chave **k** não está no nó interno **x**, mas presente em um filho com $t - 1$ chaves:



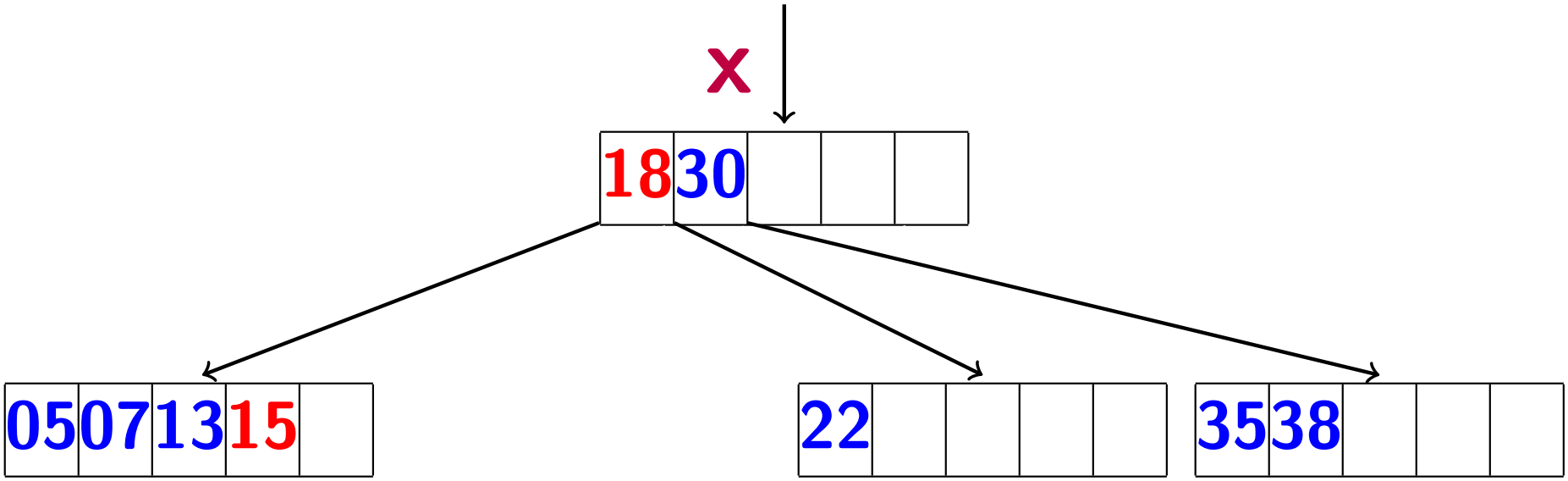
Árvore B - Remoção

Caso 3A) se a chave **k** não está no nó interno **x**, mas presente em um filho com $t - 1$ chaves:



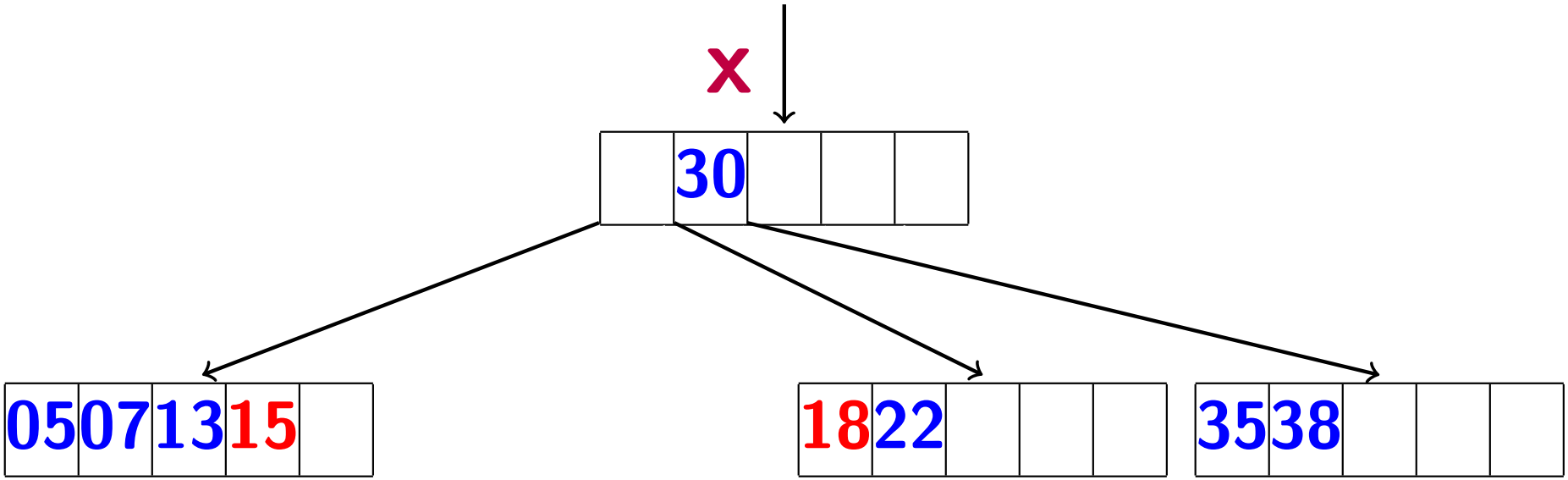
Árvore B - Remoção

Caso 3A) se a chave **k** não está no nó interno **x**, mas presente em um filho com $t - 1$ chaves:



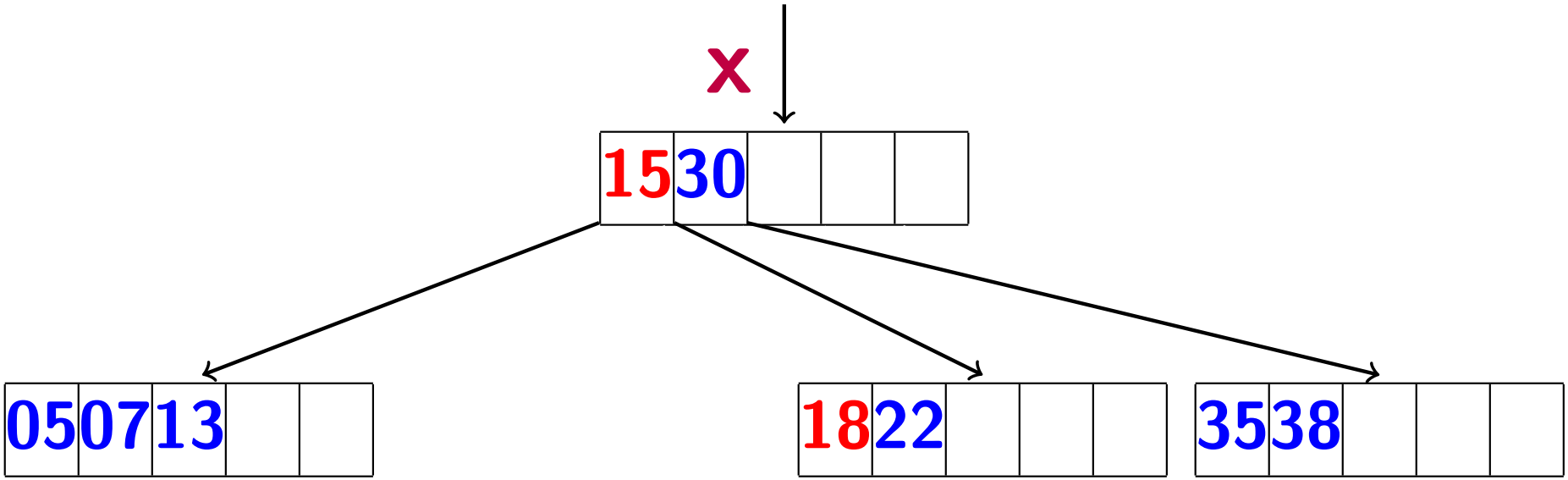
Árvore B - Remoção

Caso 3A) se a chave **k** não está no nó interno **x**, mas presente em um filho com $t - 1$ chaves:



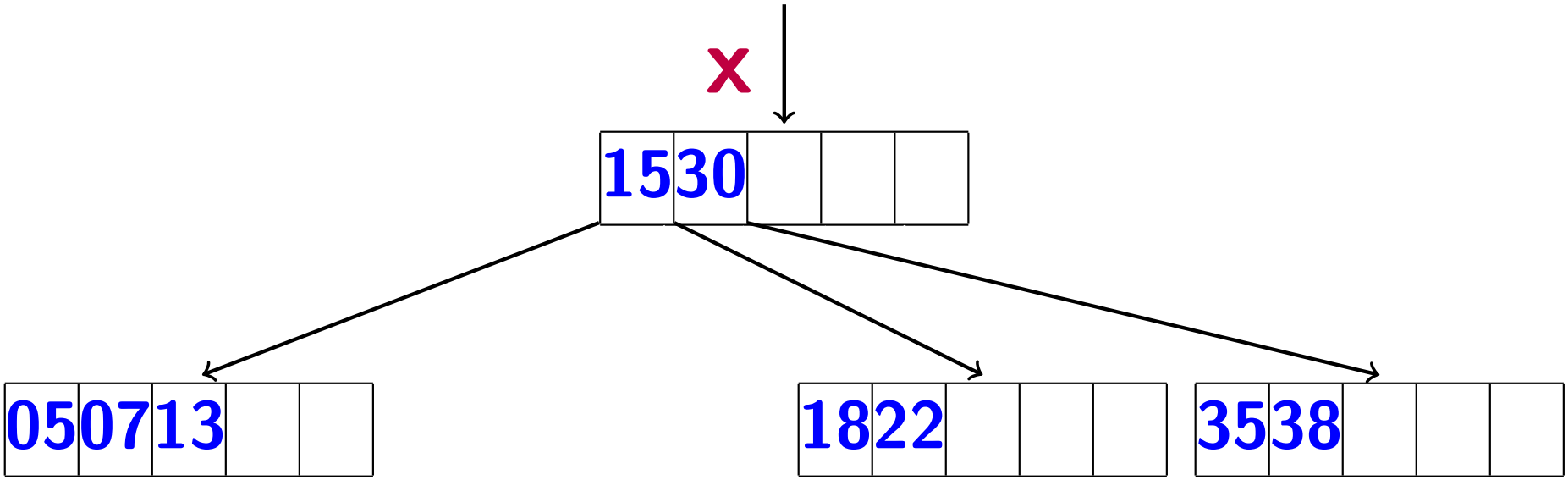
Árvore B - Remoção

Caso 3A) se a chave **k** não está no nó interno **x**, mas presente em um filho com $t - 1$ chaves:



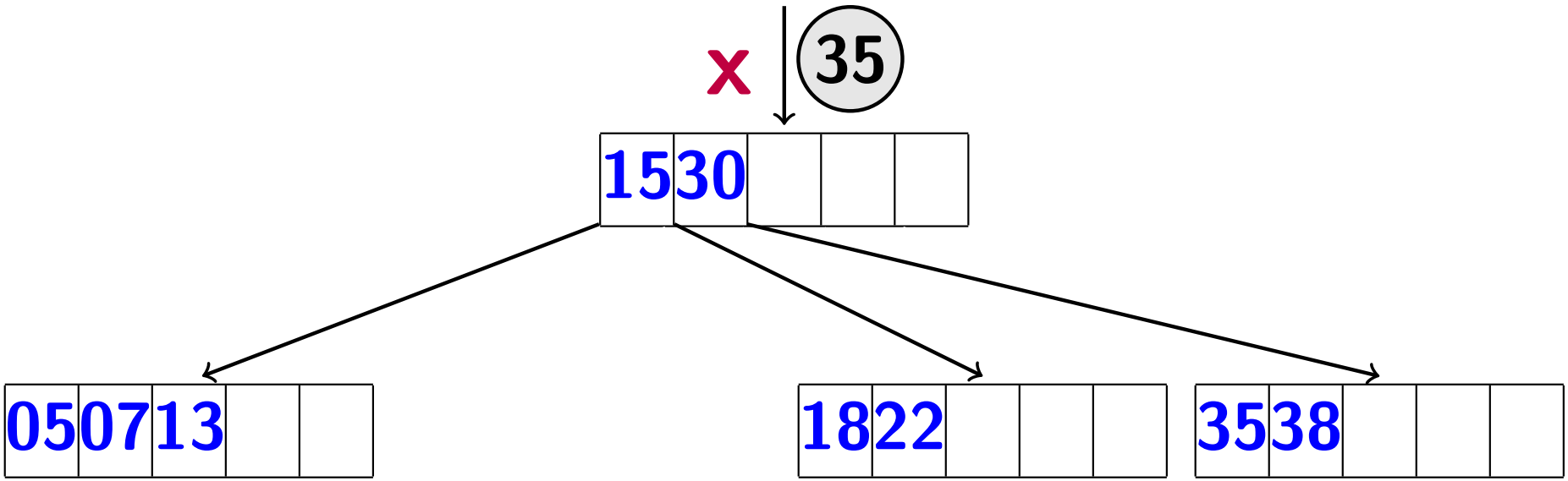
Árvore B - Remoção

Caso 3B) se a chave **k** não está no nó interno **x**, mas presente em um filho com $t - 1$ chaves:



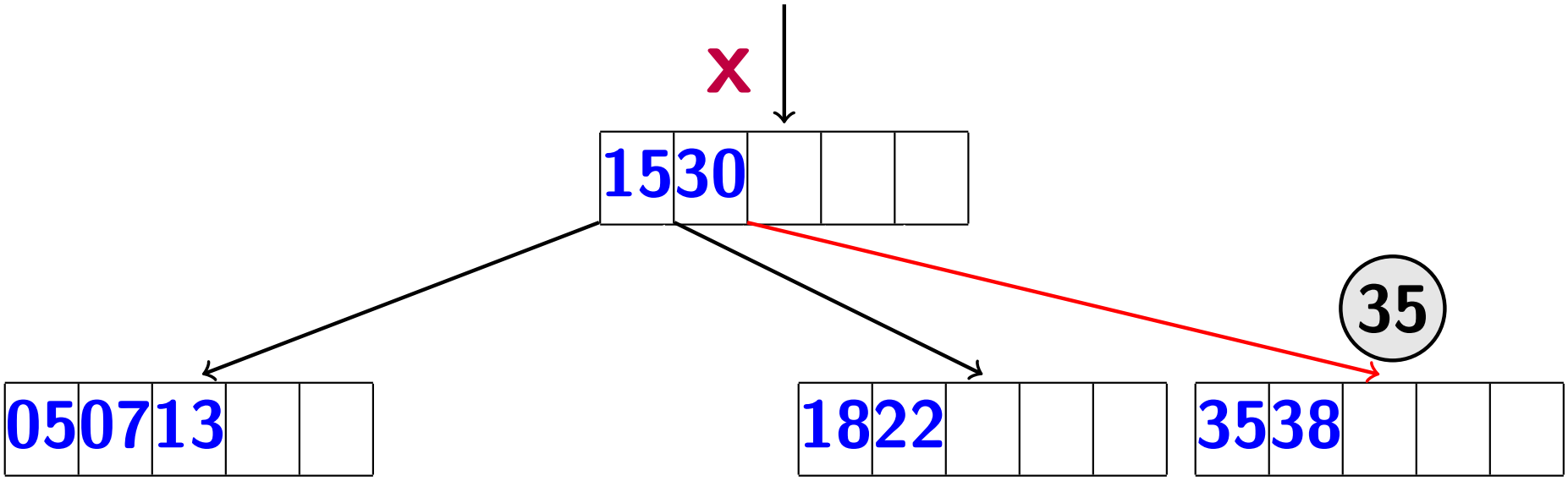
Árvore B - Remoção

Caso 3B) se a chave **k** não está no nó interno **x**, mas presente em um filho com $t - 1$ chaves:



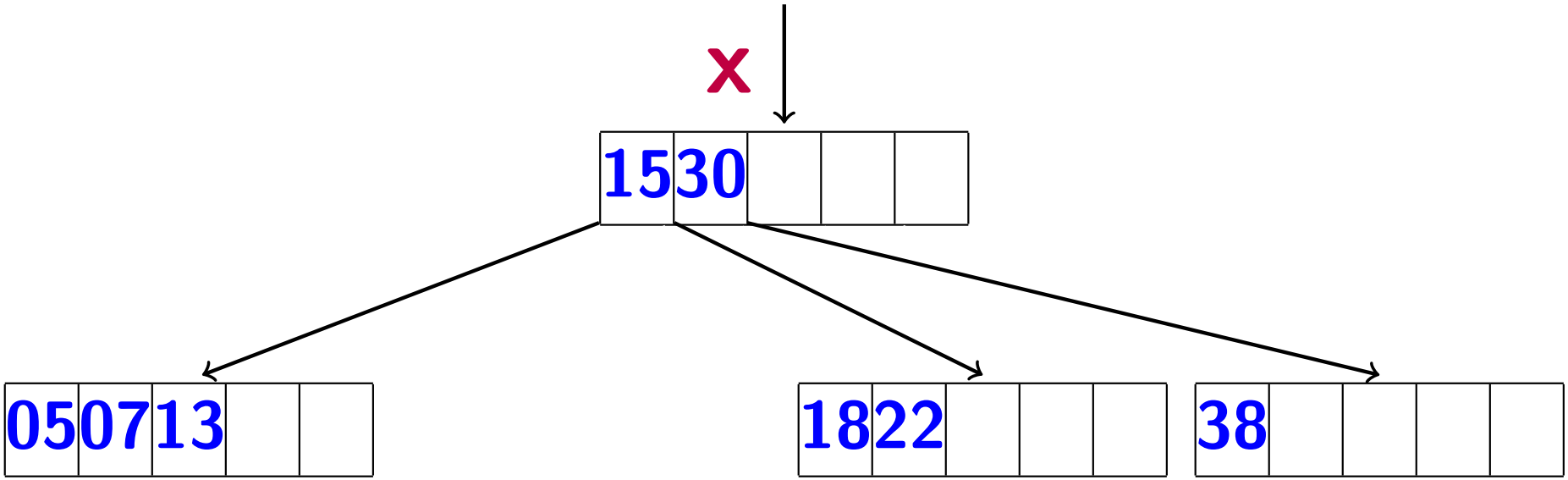
Árvore B - Remoção

Caso 3B) se a chave **k** não está no nó interno **x**, mas presente em um filho com $t - 1$ chaves:



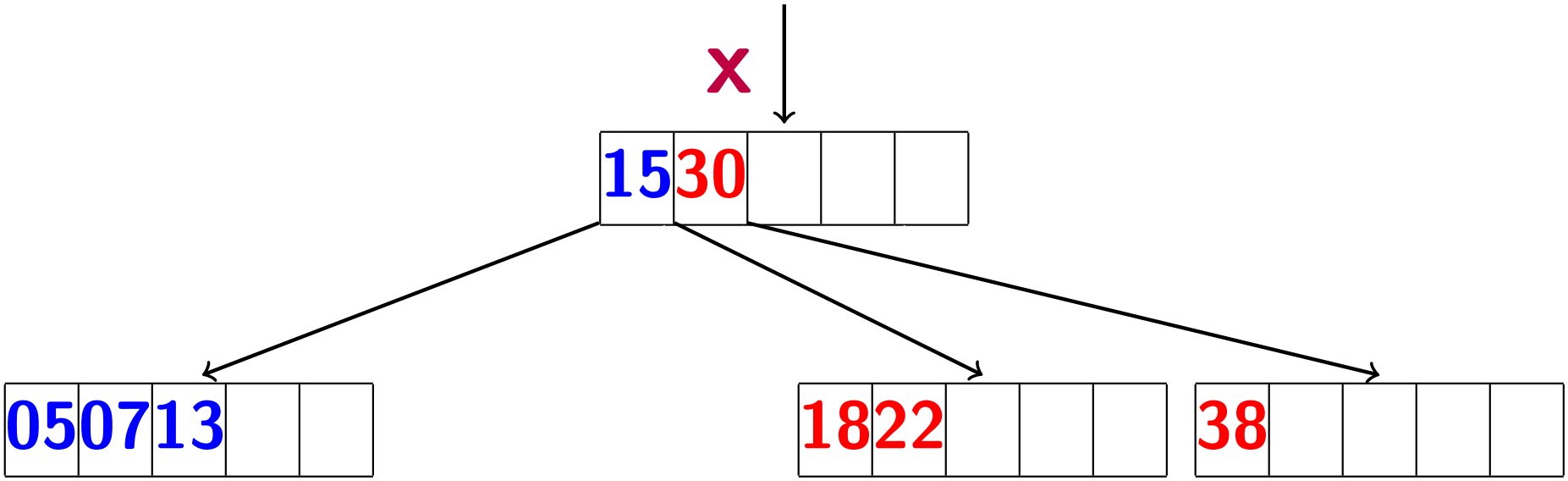
Árvore B - Remoção

Caso 3B) se a chave **k** não está no nó interno **x**, mas presente em um filho com $t - 1$ chaves:



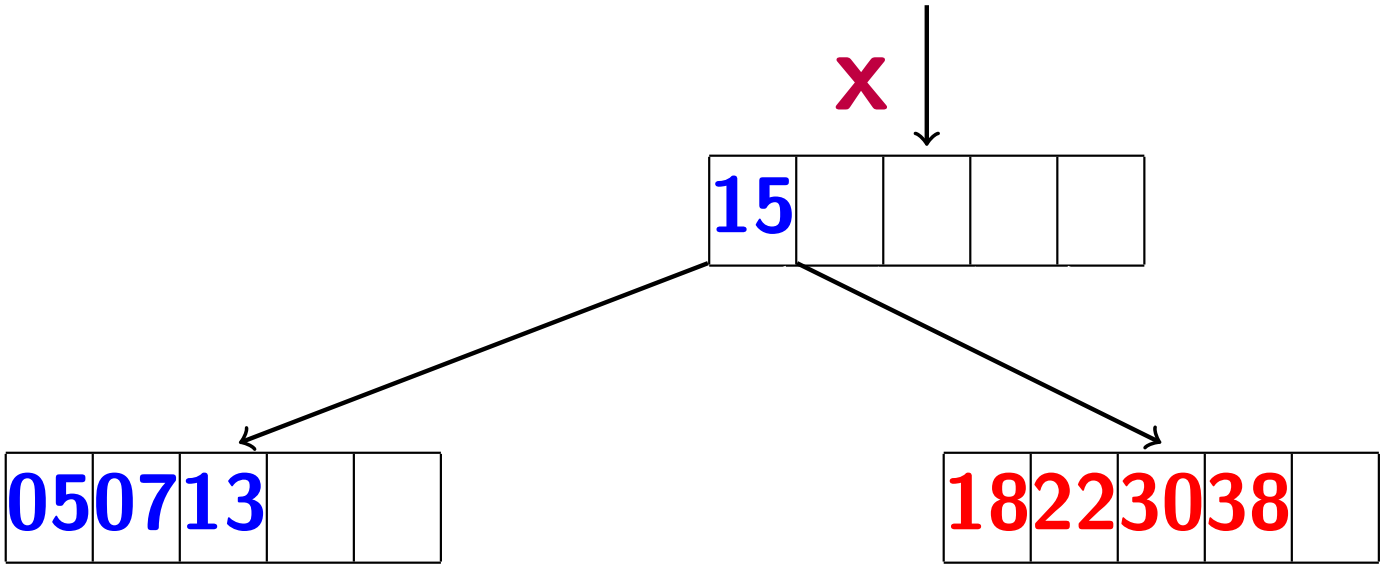
Árvore B - Remoção

Caso 3B) se a chave **k** não está no nó interno **x**, mas presente em um filho com $t - 1$ chaves:



Árvore B - Remoção

Caso 3B) se a chave **k** não está no nó interno **x**, mas presente em um filho com $t - 1$ chaves:



Sumário

Árvore B - Remoção

Complexidade: embora a operação de remoção parece complicada, ela envolve apenas $\mathcal{O}(\mathbf{h})$ acessos ao disco para uma árvore B de altura **h**. Portanto a operação de remoção tem um custo de $\mathcal{O}(\mathbf{th}) = \mathcal{O}(\mathbf{t} \log_t n)$.

Sumário

Árvore B - Remoção - Caso 1

Caso 1 (formal): se a chave **k** está nó **x**, e **x** é uma **folha**, simplesmente elimine a chave **k** de **x**.

Árvore B - Remoção - Caso 2

Caso 2 (formal): se a chave **k** está nó **x** e **x** é um nó interno, faça:

A) Se o filho y que precede k no nó **x** possui pelo menos t chaves, encontre o predecessor k' de **k** na sub-árvore com raiz em y. Elimine recursivamente k', e substitua **k** por k' em **x**.

Árvore B - Remoção - Caso 2

Caso 2 (formal): se a chave **k** está nó **x** e **x** é um nó interno, faça:

B) Simetricamente, se o filho z que segue **k** no nó **x** tem pelo menos t chaves, então encontre o sucessor k' de **k** na subárvore com raiz em z. Elimine recursivamente k', e substitua k por k' em **x**.

Árvore B - Remoção - Caso 2

Caso 2 (formal): se a chave **k** está nó **x** e **x** é um nó interno, faça:

C) Caso ambos y e z possuem somente $t - 1$ chaves, copie todos os elementos de z em y, libere a memória ocupada por z e remova o apontador em **x** e remova **k** de **x**.

Árvore B - Remoção - Caso 3

Caso 3 (formal): se a chave **k** não está presente no nó interno **x**, determine a subárvore $c_i[\mathbf{x}]$ apropriada que deve conter **k**. Se $c_i[\mathbf{x}]$ tiver somente $t-1$ chaves, execute os passos a) ou b):

A) Se $c_i[\mathbf{x}]$ possui pelo menos $t-1$ chaves, mas tiver um irmão com t chaves, copie para $c_i[\mathbf{x}]$ uma chave extra, movendo uma chave de **x** para $c_i[\mathbf{x}]$ em seguida movendo uma chave do irmão esquerdo ou direito imediato de $c_i[\mathbf{x}]$ para dentro de **x** e ajustando o apontador para o nó correspondente.

Árvore B - Remoção - Caso 3

Caso 3 (formal): se a chave **k** não está presente no nó interno **x**, determine a subárvore $c_i[\mathbf{x}]$ apropriada que deve conter **k**. Se $c_i[\mathbf{x}]$ tiver somente $t-1$ chaves, execute os passos a) ou b):

B) Se $c_i[\mathbf{x}]$ e todos os irmãos de $c_i[\mathbf{x}]$ têm $t-1$ chaves, faça a intercalação de $c_i[\mathbf{x}]$ com um único irmão, o que envolve mover uma chave **x** para baixo até o novo nó intercalado, a fim de se tornar a chave mediana para esse nó.

Sumário

Referências

[1] Algoritmos: Teoria e prática. Cormen.