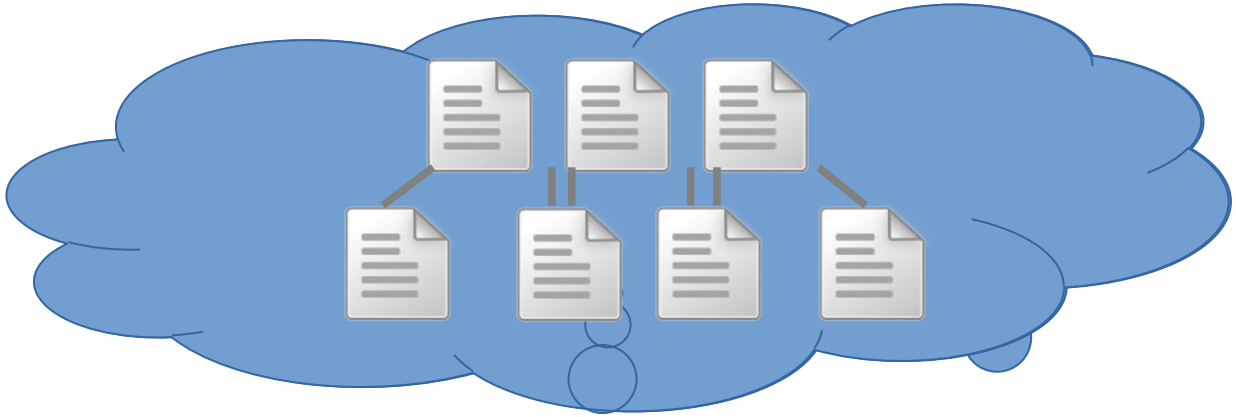
Árvore B

Joaquim Quinteiro Uchôa
Juliana Galvani Greghi

Árvore B

Árvores B são utilizadas principalmente para reduzir o tempo de acesso a dados em memória secundária



Bayer e McCreight - 1972

Árvore B - Propriedades

O tamanho de cada nó pode ser tão grande quanto o tamanho de um bloco em disco.

A raiz tem pelo menos duas subárvores, a menos que seja uma folha.

Todas as folhas estão no mesmo nível (balanceamento).

Por que não ABB?

Limite da quantidade de dados.

Pode ficar degenerada depois de várias inserções
→ desempenho baixo para busca.

Estrutura inadequada para dados em memória secundária (vários acessos durante as operações).

Árvore B - Ordem

Em uma árvore de ordem `m` cada nó não-raiz e não-folha contém `c-1` chaves e c ponteiros para as subárvores em que `m/2 < c < m.`

Cada nó folha contém `c-1` chaves, onde `m/2 < c < m.`

Entretanto, o conceito de ordem não é padrão...

Definições de ordem de Árvores B - 1/4

I. Wikipedia: Usually, the number of keys is chosen to vary between **d** and **2d**, where d is the **minimum number of keys**, and d+1 is the minimum degree or branching factor of the tree.

II. Cormen: Nodes have lower and upper bounds on the number of keys they can contain. We express these bounds in terms of a fixed integer t called the minimum degree of the B-tree: Every node other than the root must have at least t - 1 keys. (...) Every node may contain at most 2t - 1 keys.

Definições de ordem de Árvores B - 2/4

III. Knuth: a B-tree of order m is a tree which satisfies the following properties:
* Every node has at most m children.
* Every non-leaf node (except root) has at least ⌈m/2⌉ children.

IV. Aho & Hopcroft: Formally, a B-tree of order m is an m-ary search tree with the following properties:
1. The root is either a leaf or has at least two children.
2. Each node, except for the root and the leaves, has between [m/2] and m children.

Definições de ordem de Árvores B - 3/4

V. Horvick: More formally, each non-root node in the B-tree will contain at least T children and a maximum of 2 × T children. (...) Every non-root node will have at least T - 1 values. Every non-root node will have at most 2 × T - 1 values.

VI. Sedgewick: B tree of order M is a tree that either is empty or comprises k-nodes, with k-1 keys and k links (...) k must be between 2 and M at the root and between M/2 and M at every other node (...).

Definições de ordem de Árvores B - 4/4

As definições de ordem podem variar de diferentes maneiras entre os autores, podendo se referir a:

- número mínimo de chaves de um nó;
- número máximo de chaves de um nó;
- número mínimo de filhos de um nó não-folha;
- número máximo de filhos de um nó não-folha.

É importante estar atento ao que está sendo usado como definição, para evitar problemas.

Inserção em árvore B - 1/2

Todas as folhas tem que estar no mesmo nível.

Isso implica em construção bottom-up: da base para a raiz.
A raiz está em constante mudança!

Somente após todas as inserções o conteúdo do nó raiz é conhecido definitivamente.

Uma nova chave é sempre inserida em um nó folha.

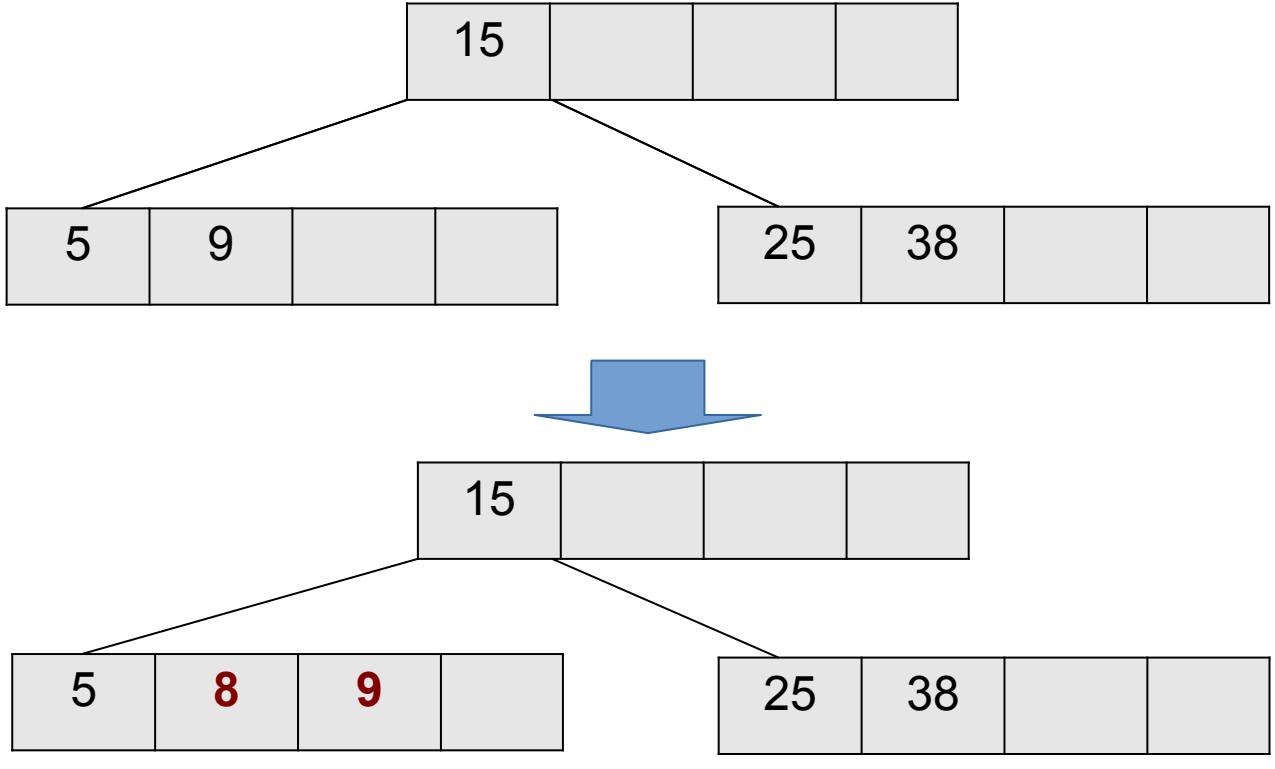
Inserção em árvore B - 2/2

Três casos devem ser tratados:

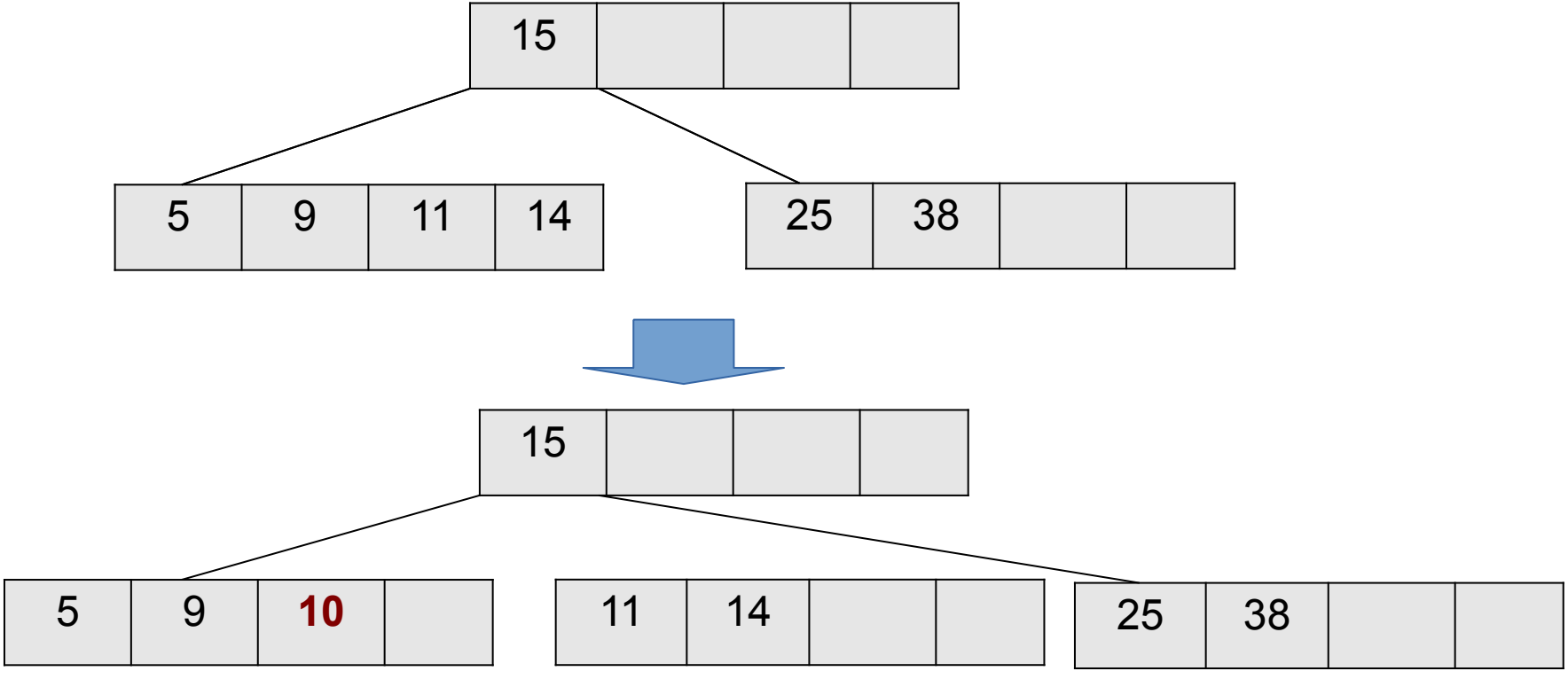
1. Uma chave é inserida em um nó que ainda tem espaço;
2. A folha na qual a chave precisa ser colocada está cheia;
3. A raiz da árvore está cheia.

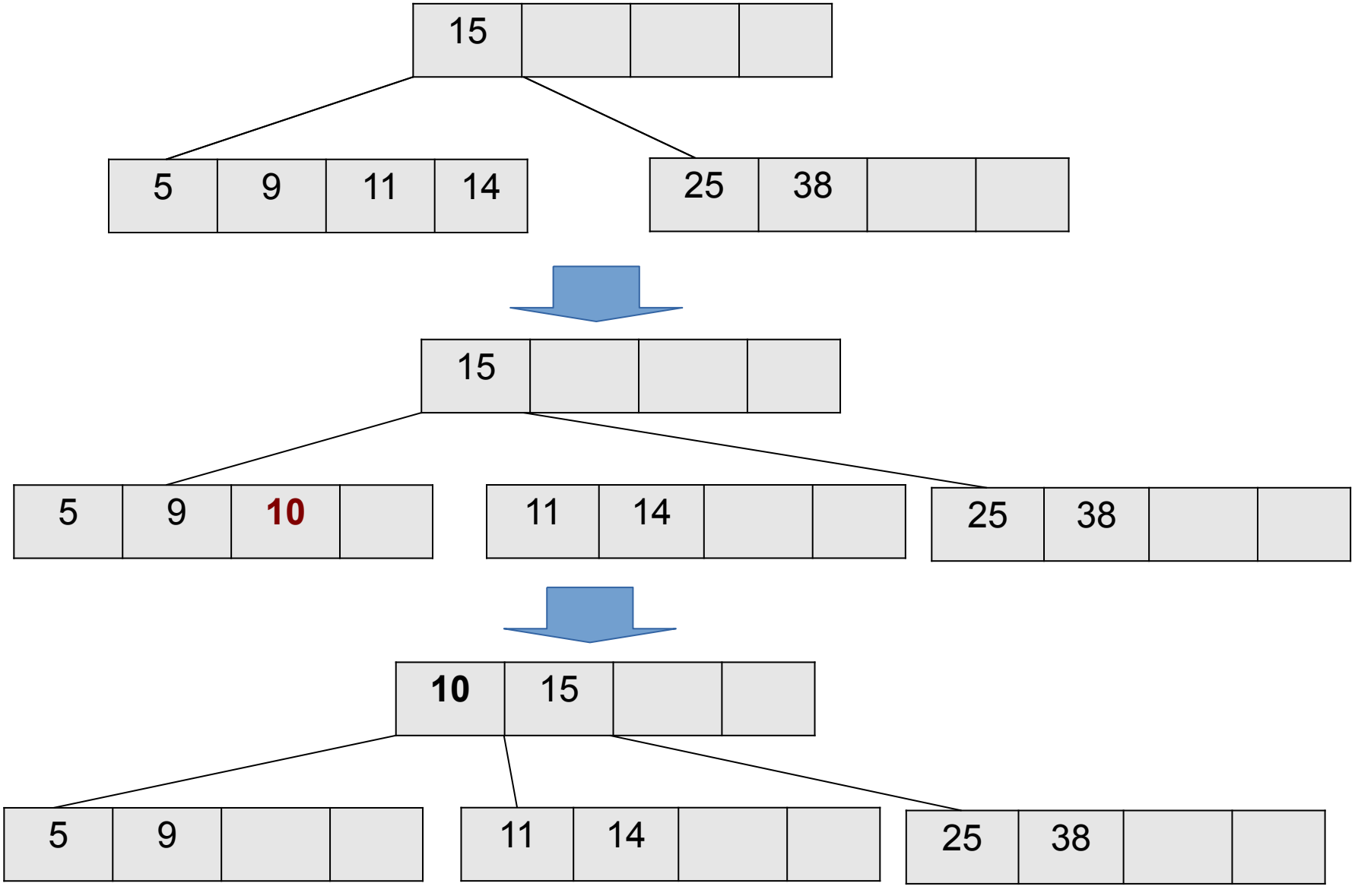
Caso 1: uma chave é inserida em uma folha que ainda tem espaço

→ **inserir 8**

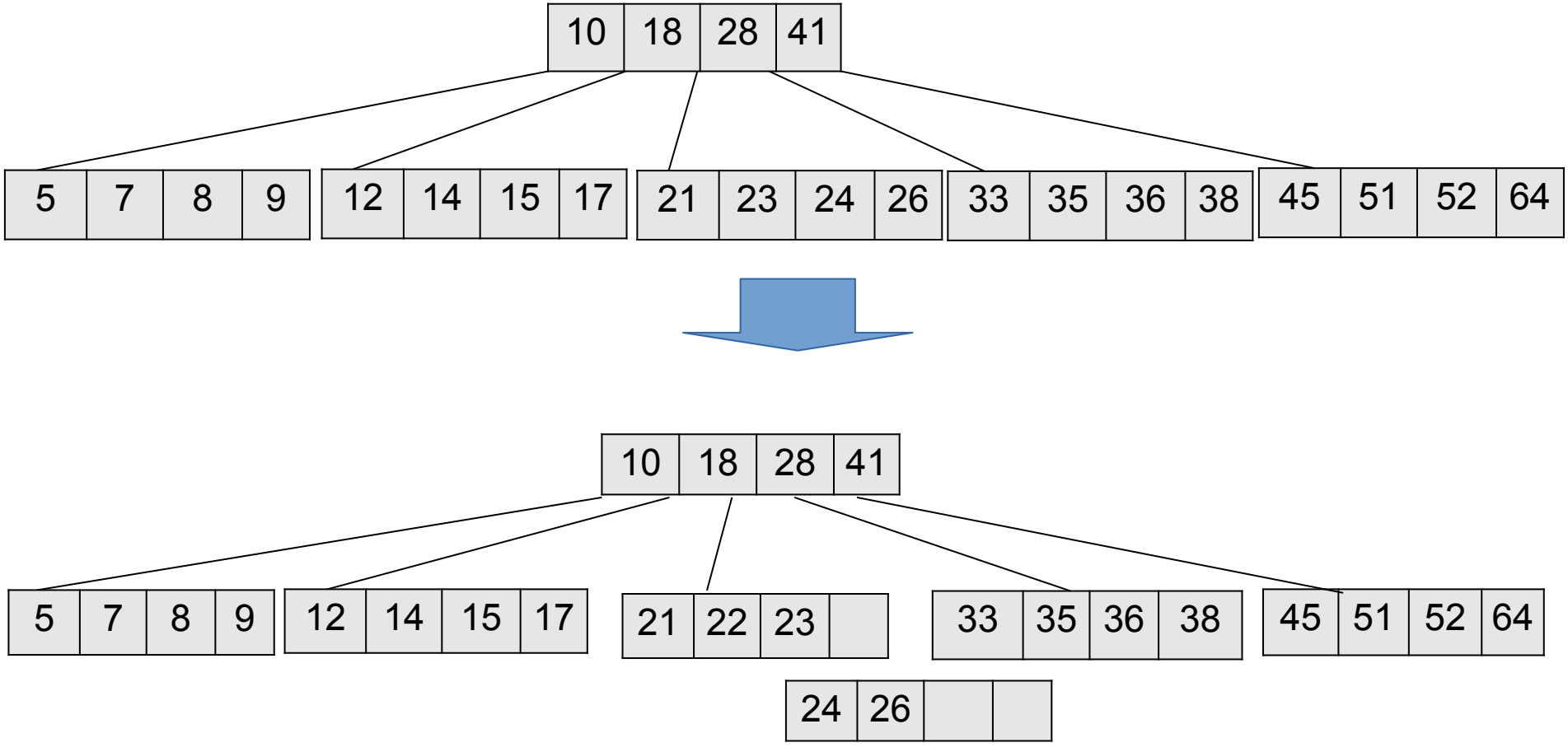


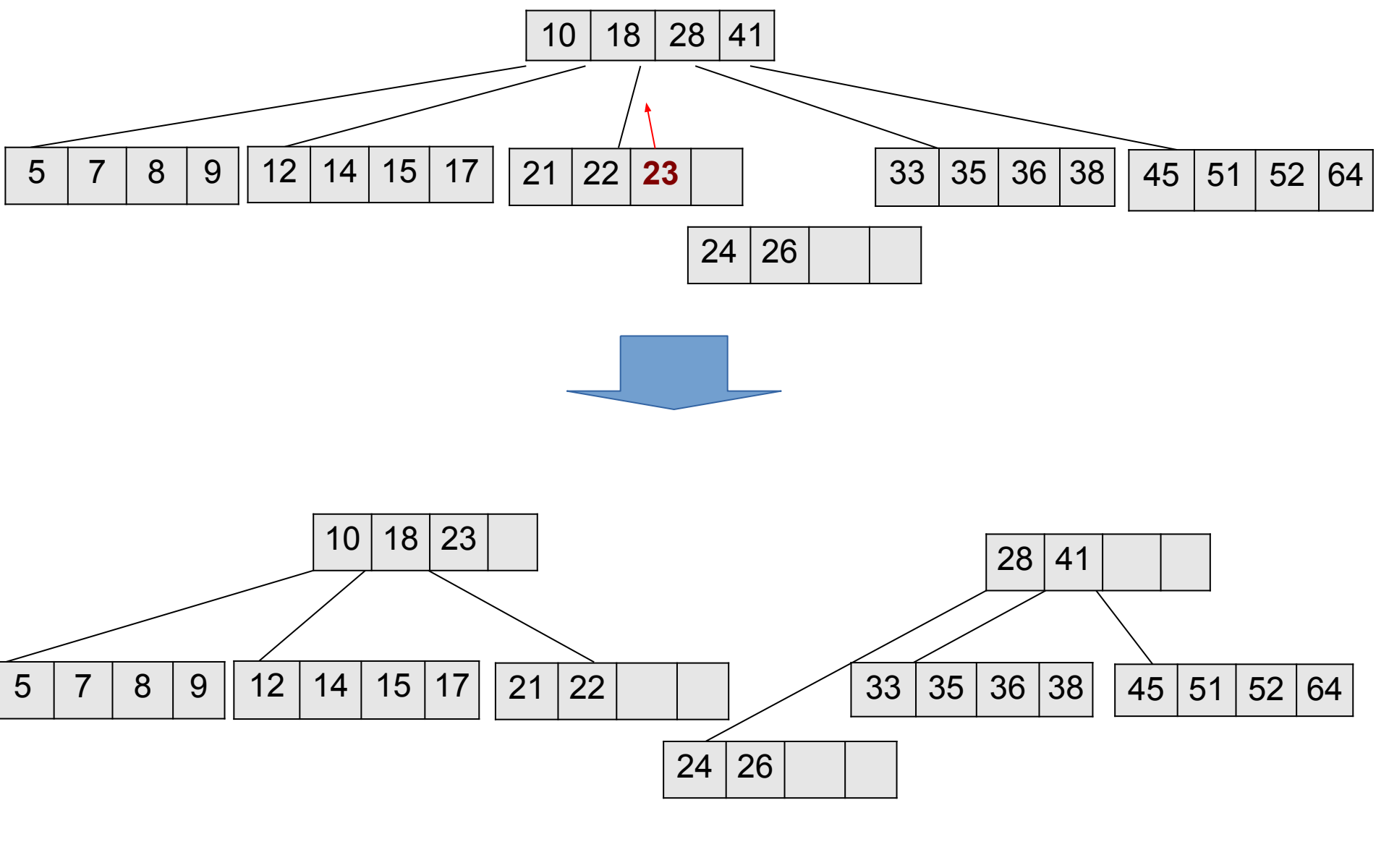
Caso 2: uma chave deve ser inserida em uma folha cheia
→ **inserir 10**

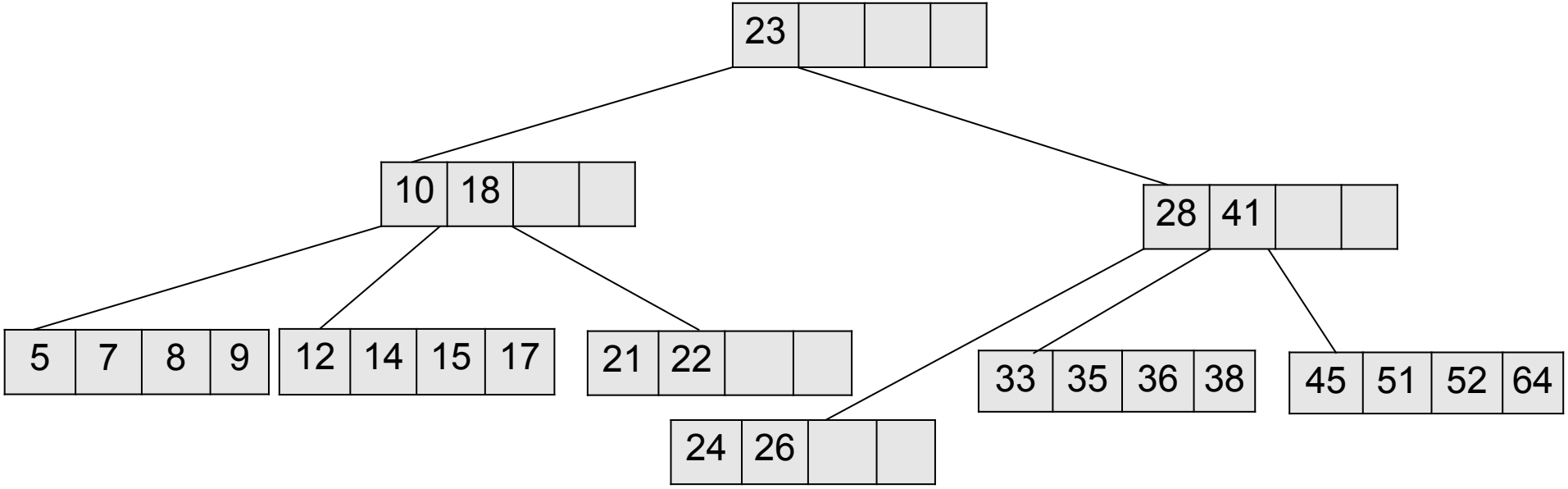



15
5
9
11
14
25
38
15
5
9
10
11
14
25
38
10
15
5
9
11
14
25
38

Caso 3: a raiz está cheia
→ **inserir 22**




10 18 28 41
5 7 8 9
12 14 15 17
21 22 23
33 35 36 38
45 51 52 64
24 26
10 18 23
5 7 8 9
12 14 15 17
21 22
28 41
33 35 36 38
45 51 52 64
24 26


23
10
18
28
41
5
7
8
9
12
14
15
17
21
22
24
26
33
35
36
38
45
51
52
64

Inserção em Árvore B - Algoritmo

```
Insere (C)
   Encontre o nó folha para inserir C;
   Encontre a posição apropriada para C;
   Se nó não está cheio
      Insere C;
   Senão
      Divida o nó em nó1 e nó2;
      Distribua as chaves e ponteiros igualmente
         entre nó1 e nó2;
      Se nó era raiz
         Crie uma nova raiz como ascendente de
            no1 e no2;
         Ajuste os ponteiros para no1 e no2;
```

Remoção em árvore B

Semelhante à remoção em árvore binária, mas com preocupação adicional: nós não podem ficar com menos da metade preenchido.

Algumas vezes nós serão fundidos, ou chaves serão rotacionadas.

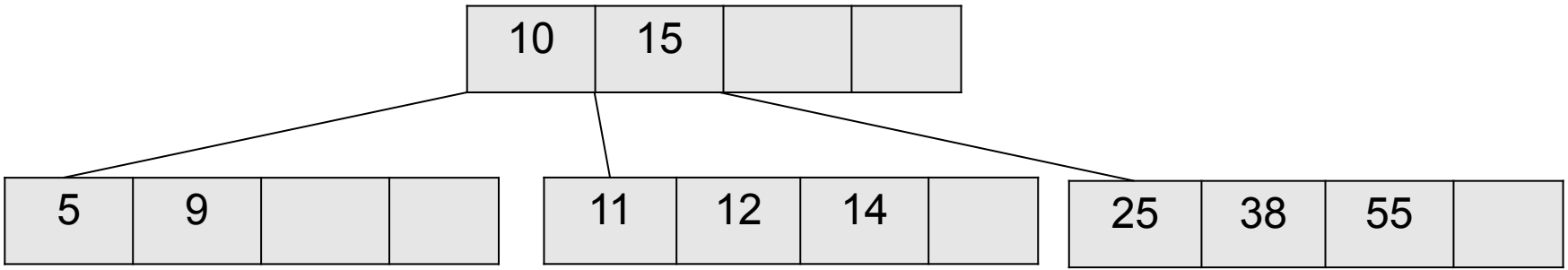
Casos principais:

- Remover uma chave em uma folha;
- Remover uma chave em um nó não-folha.

Remoção em Árvore B - Caso 1

Caso 1: depois de remover *C* a folha está pelo menos metade cheia e somente chaves maiores que *C* são movidas para a esquerda.

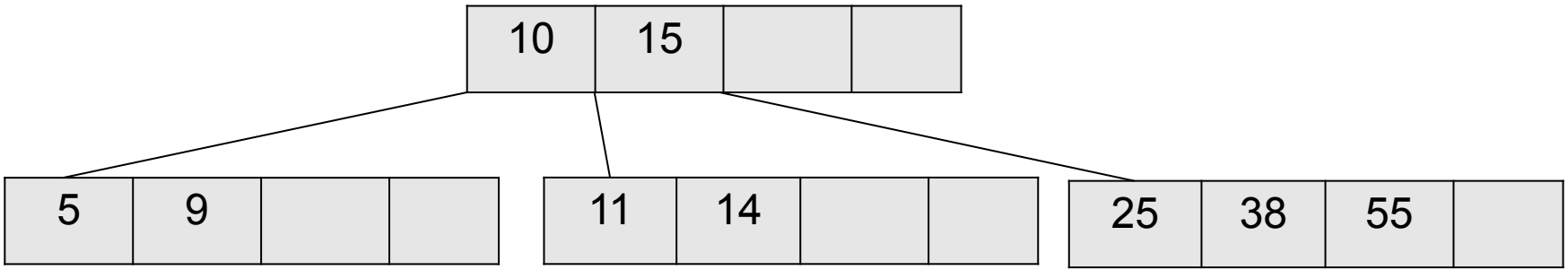
⇨ Remover 12



Remoção em Árvore B - Caso 1

Caso 1: depois de remover *C* a folha está pelo menos metade cheia e somente chaves maiores que *C* são movidas para a esquerda.

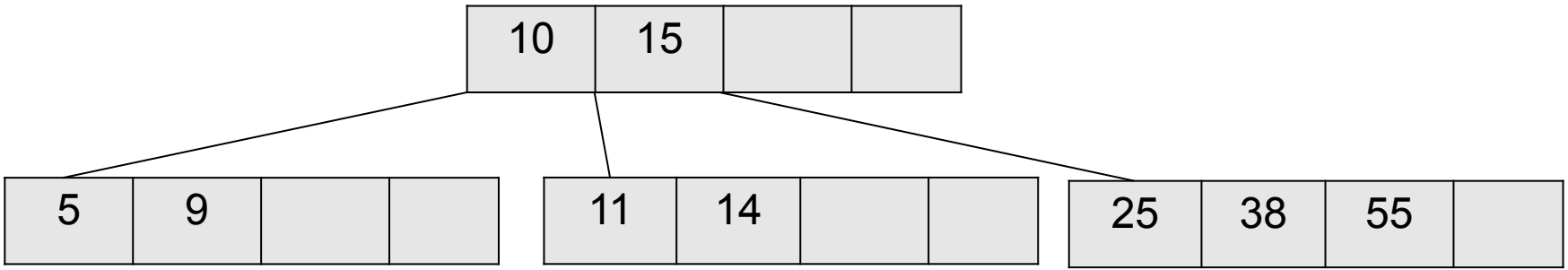
⇨ Remover 12



Remoção em Árvore B - Caso 2

Caso 2: depois de remover C, o número de chaves na folha é menor do que m/2 -1.

- Remover 14

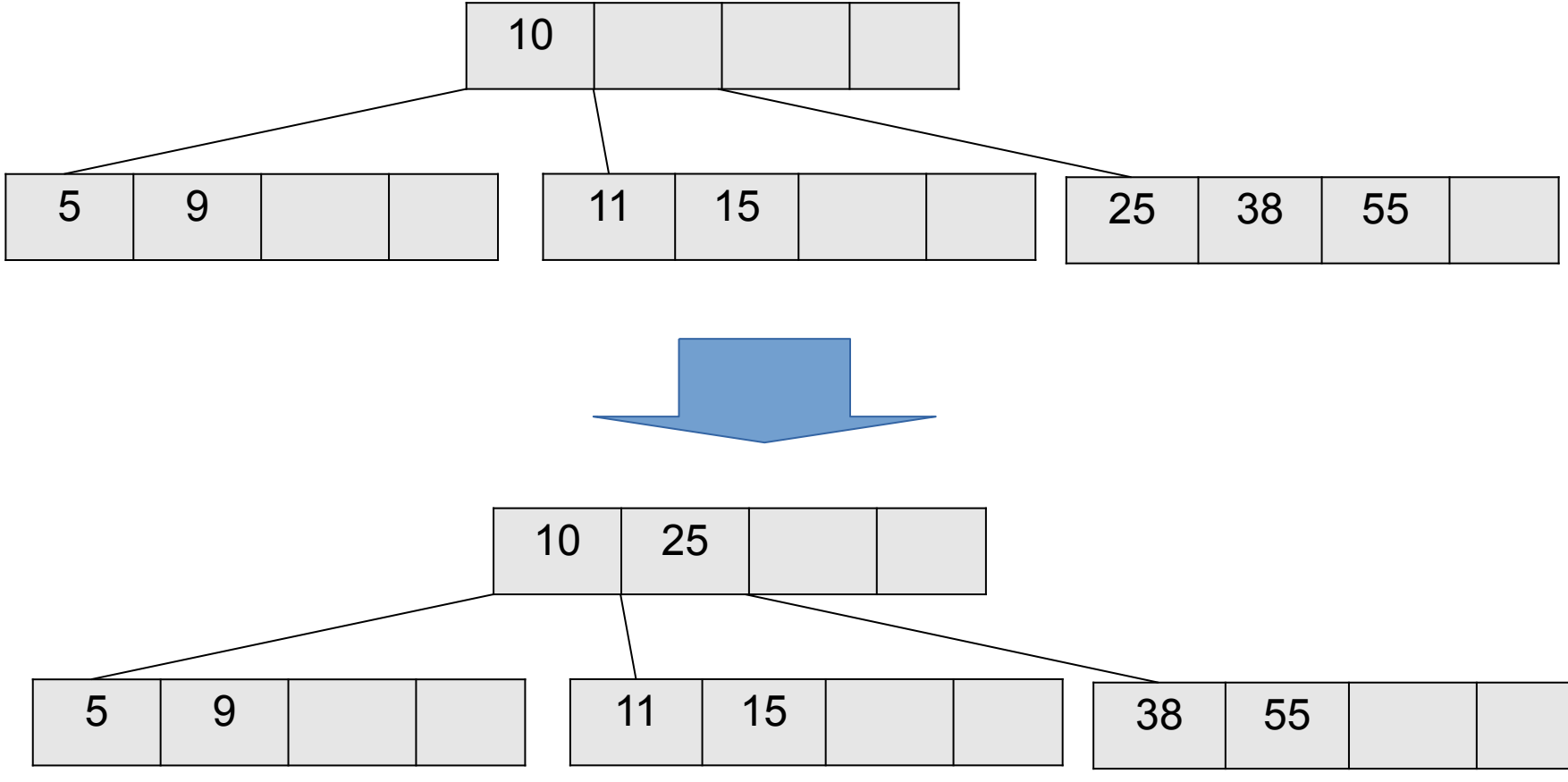


Remoção em Árvore B - Caso 2A

Caso 2A: Se existe um irmão à esquerda ou à direita com o número de chaves excedendo o mínimo m/2 -1, todos os nós dessa folha e do irmão são redistribuídas movendo-se a chave separadora do ascendente para a folha e movendo uma chave do irmão para o ascendente.

A chave do ascendente separando o nó atual e o irmão, ocupará o lugar da chave removida.

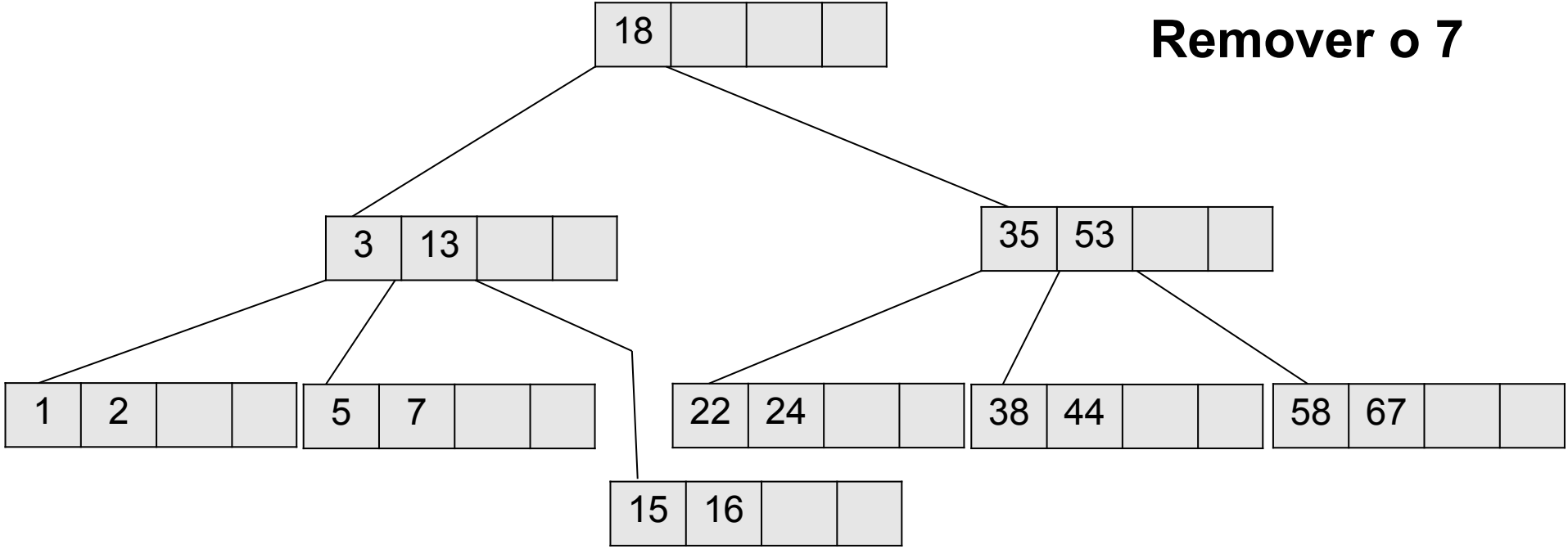
Remoção em Árvore B - Caso 2A



Remoção em Árvore B - Caso 2B

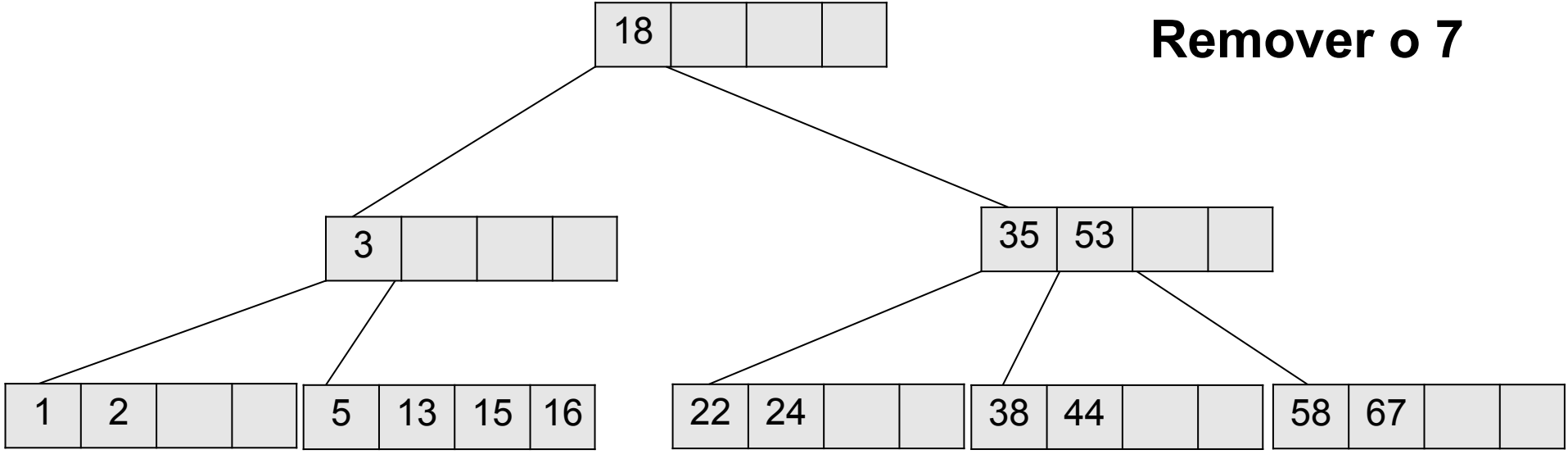
Caso 2B: Se o número de chaves do irmão é m/2-1, a folha e o irmão são fundidos; as chaves da folha, seu irmão e a chave de separação do ascendente são todas colocadas na folha e o irmão é apagado. As chaves no ascendente são movidas caso fique um espaço vazio.

Remoção em Árvore B - Caso 2B



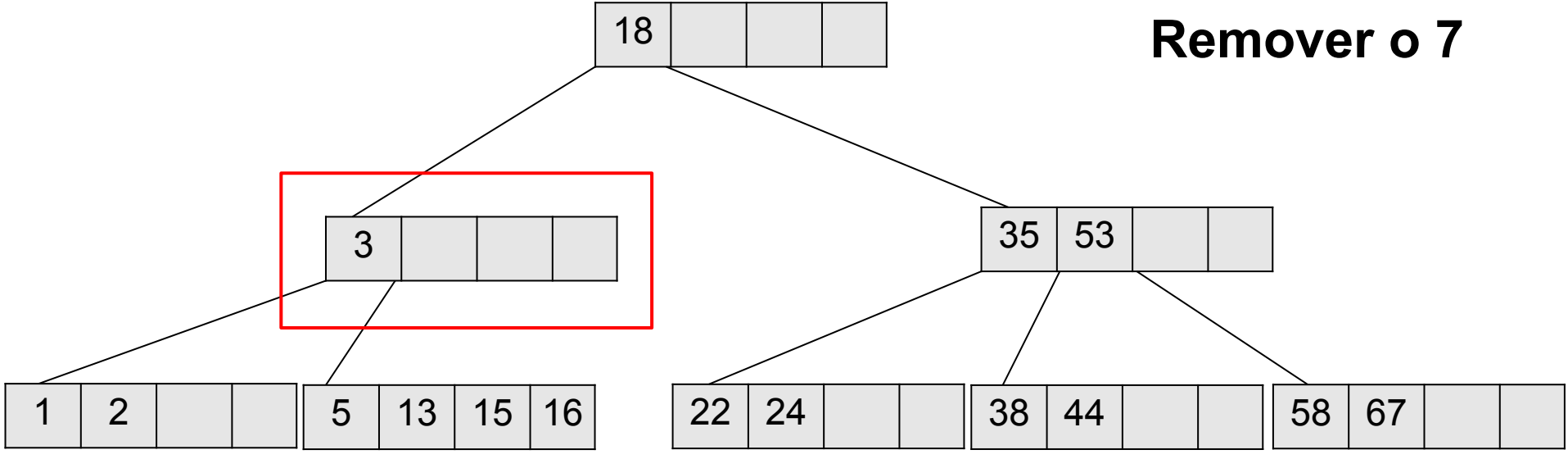
Remover o 7

Remoção em Árvore B - Caso 2B



Remover o 7

Remoção em Árvore B - Caso 2B

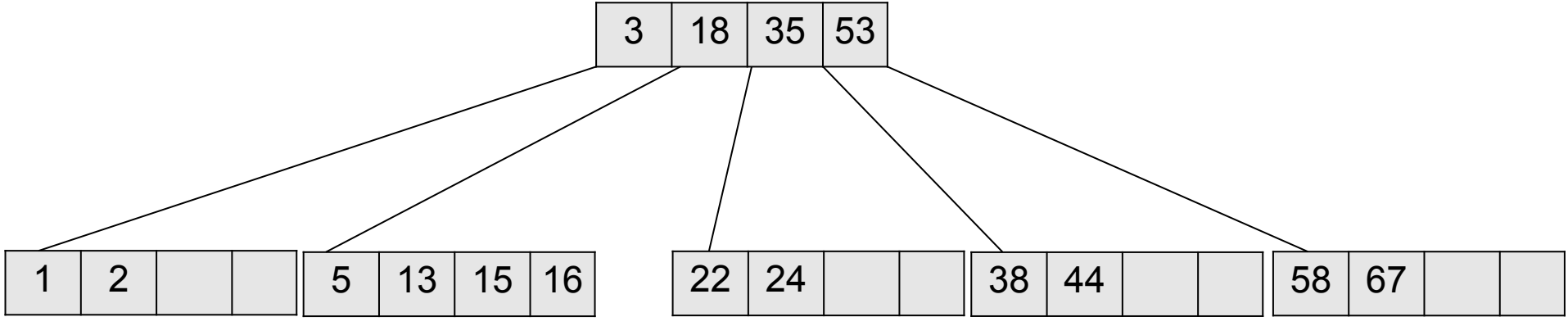


Remover o 7

Cadeia de operações!

Remoção em Árvore B - Caso 2B

Remover o 7



A operação é repetida, até que o nó fique com pelo menos o mínimo ou se alcance a raiz.

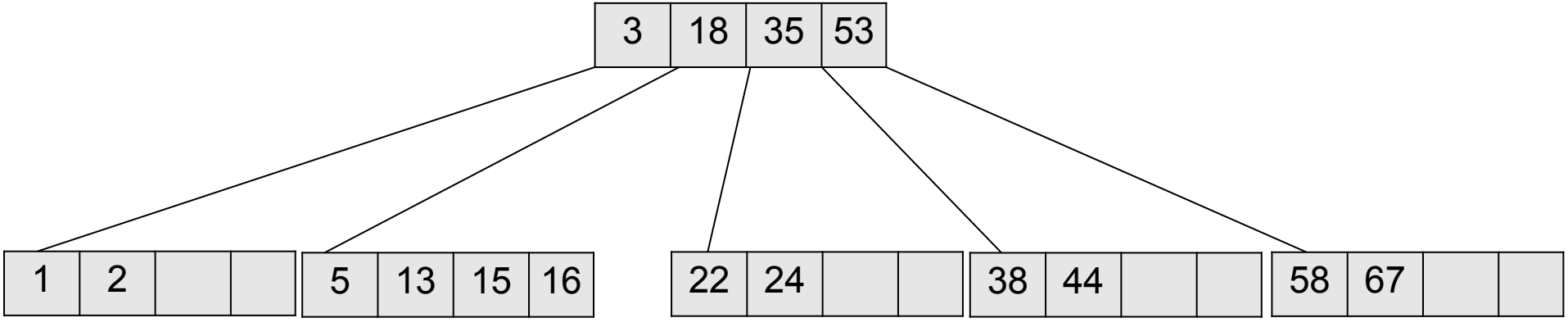
Remoção em Árvore B - Caso 3

Caso 3: A chave a ser removida está em um nó não folha → pode levar a problemas na reorganização da árvore.
Para evitar isso, chave a ser removida é substituída por seu sucessor imediato (ou antecessor), que só pode ser encontrado em uma folha.

O tratamento dado à remoção do sucessor (ou antecessor) em seu nó folha é similar aos vistos anteriormente.

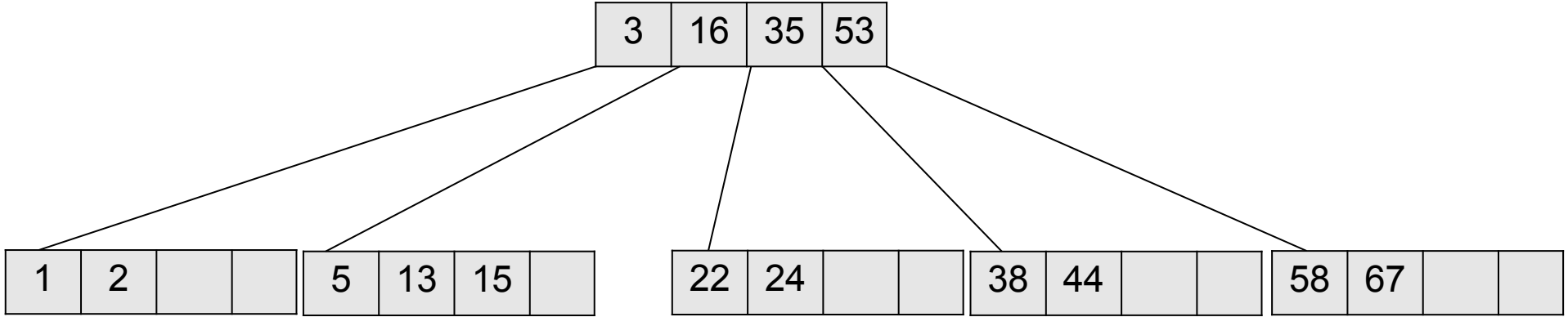
Remoção em Árvore B - Caso 3

Remover o 18



Remoção em Árvore B - Caso 3

Remover o 18

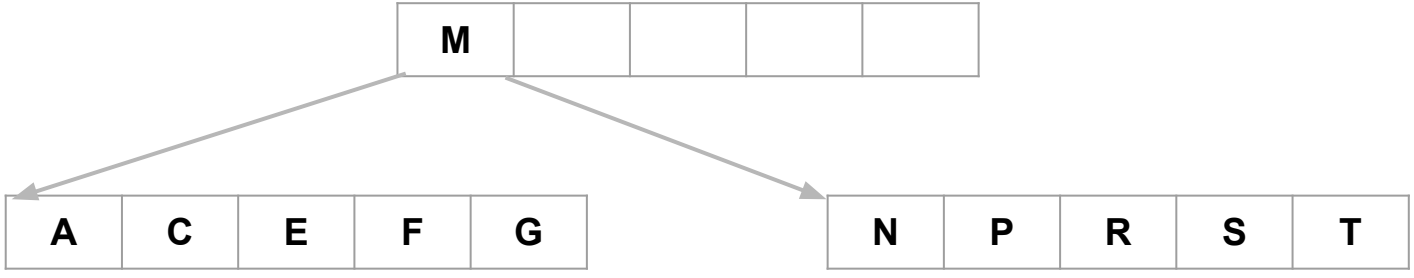


Árvores B*

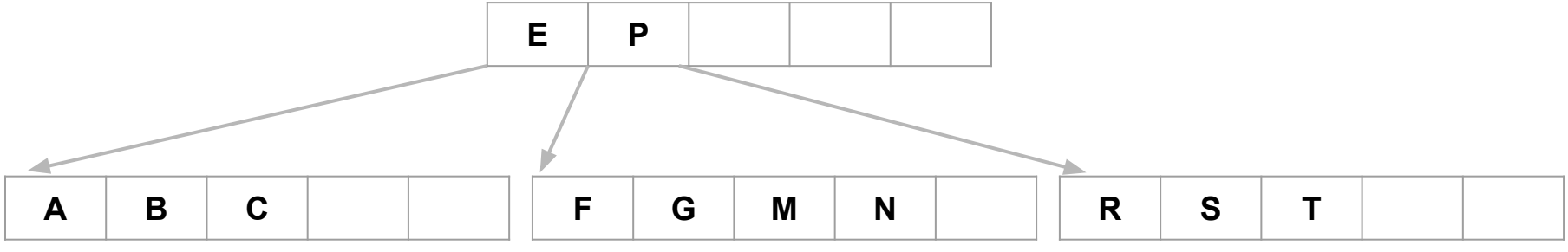
Árvores B* foram propostas por Donald Knuth, com duas alterações significativas:

1. Cada filho possui no máximo m e no mínimo (2m -1)/3 filhos, o que implica em uma taxa de ocupação mínima de ⅔ em cada nó.
2. A divisão do nó (split) é postergada até que duas páginas irmãs estejam cheias. As duas páginas são divididas em três (two-to-three split).

Exemplo de Inserção com Divisão



Insere o B



Árvores B* - Considerações

O adiamento da divisão ocorre por meio de rotação de chaves, de maneira similar ao que já ocorre na remoção em árvores B: caso um nó esteja cheio, mas não seu irmão, então ocorre rotação de chaves para liberar espaço.

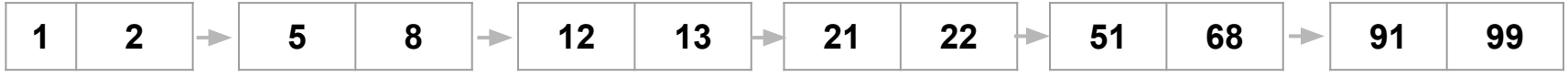
Uma questão a ser tratada de forma diferenciada é a raiz: como a raiz é única, como dividir dois nós em três. Duas soluções ocorrem: a raiz tem o dobro do tamanho, dividindo-se em três ou no caso da raiz é aceito que ela se divida em duas ao invés de três.

Árvores B+

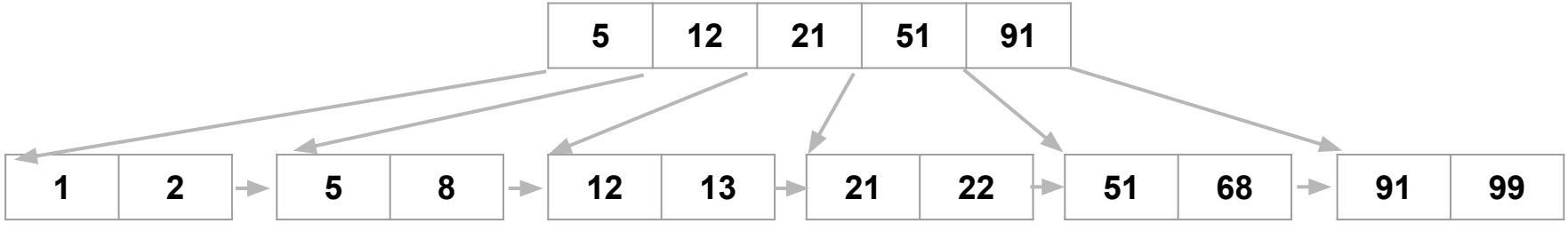
Em uma árvore B+, os dados são armazenados nas folhas, e os nós intermediários são utilizados apenas como índices.

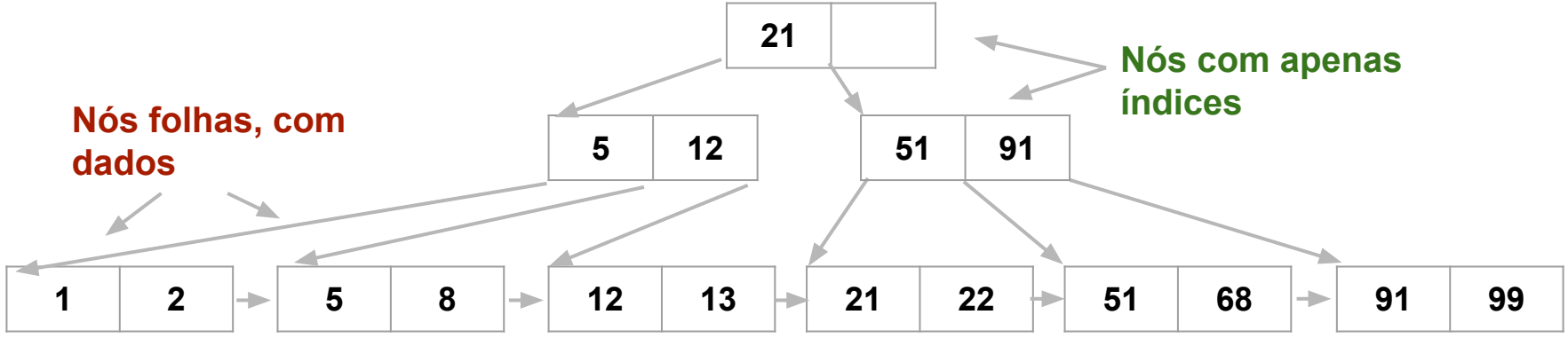
Pode ser entendida como uma árvore B de índices para um sequence set. Esse é a melhor combinação, pois a árvore B permite a busca rápida, enquanto o sequence set permite o acesso sequencial ao arquivo.

Construindo Árvores B+ a partir de um Sequence Set



Usando ≥ para comparação





Aplicações de árvores B, B+ e B*

Bancos de Dados: vários SGBDs são implementados em árvores B, ou modificações dessa estrutura. DB2, Informix, SQL Server, Oracle 8, Sybase ASE e SQLite suportam árvores b+ para índices de tabelas.

Sistemas de Arquivos: HFS+, NTFS, jfs2, btrfs e ext4 usam árvores B. HFS e Reiser4 usam árvores B*. ReiserFS, NSS, XFS usam árvores B+. NTFS usa árvores B+ para árvores de diretórios e informações de segurança. O ext4 usa árvores de extents (modificação da árvore B+) para indexação de arquivos.

Referências

Drozdek. Estrutura de Dados e Algoritmos em C++, Cengage Learning, 2002.

D.E. Knuth, The Art of Computer Programming, vol. 1 (Fundamental Algorithms) e vol. 3 (Sorting and Searching), Addison-Wesley, 1973.

R. Sedgewick, Algorithms in C (parts 1-4), 3rd. edition, Addison-Wesley/Longman, 1998.

R. Bayer, E. McCreight, Organization and Maintenance of Large Ordered Indexes, Acta Informatica, Vol. 1, Fasc. 3, 1972, pp. 173-189.