Árvores-B
Introdução

Organização e Recuperação de Dados

Profa. Valéria

UEM – CTC – DIN

Slides preparados com base no Cap. 8 do livro FOLK, M.J. & ZOELLICK, B. *File Structures*. 2nd Edition, Addison-Wesley Publishing Company, 1992

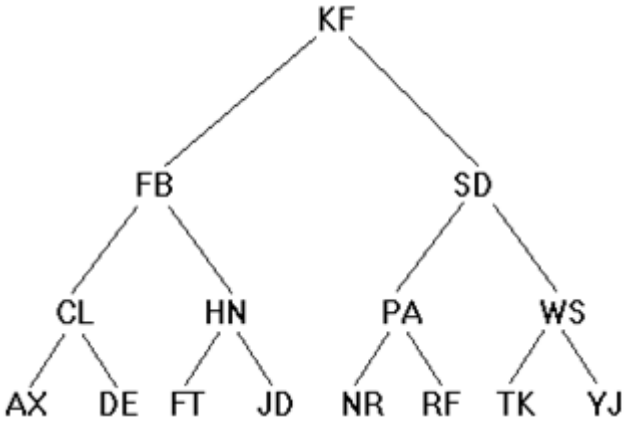
Motivação

- Problema dos índices lineares ⇨ **como manter índices de forma eficiente quando não podem permanecer na memória?**
- O problema fundamental associado à manutenção de um índice em memória secundária é o tempo de acesso
- Até este ponto, o melhor acesso a um índice ordenado foi dado pela **busca binária** (O($log_2\ n$)), porém com **dois problemas**:
  1. **Ordenação: é caro manter um índice ordenado em arquivo**
     - O ideal seria que a inserção e a remoção de novas chaves tivessem efeito local, i.e., não exigissem a reorganização do índice como um todo
  2. **Nº de acessos: a busca binária faz muitos acessos**
     - Por ex., uma busca binária em 1.000 chaves pode requerer até 10 leituras
     - O ideal seria realizarmos uma única leitura

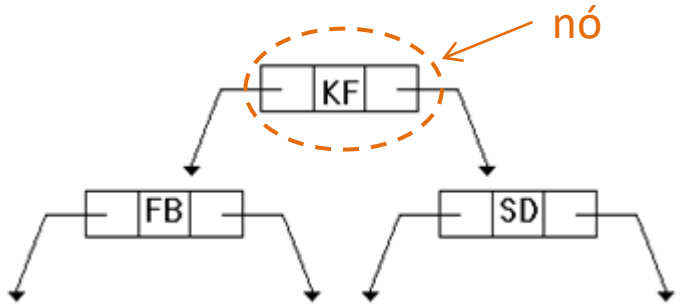
Motivação

- Uma solução para a questão da **ordenação** são as árvores binárias de busca (**ABB**)
  - **Nós** contendo **filhos esquerdo** e **direito**
  - Os filhos **menores** que o pai ficam na sub-árvore **esquerda**
  - Os filhos **maiores** que o pai ficam na sub-árvore **direita**

Chaves: AX, CL, DE, FB, FT, HN, JD, KF, NR, PA, RF, SD, TK, WS, YJ



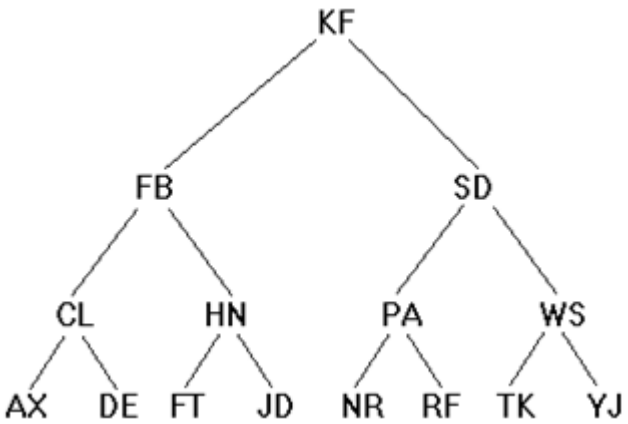
Representação lógica da ABB



Estrutura de referências de parte da ABB

Árvore Binária de Busca

- Como manter uma ABB em arquivo?
  - Cada nó da árvore é um **registro de tamanho fixo** com três campos: **chave**, RRN do **filho direito** e RRN do **filho esquerdo**



Representação lógica da ABB

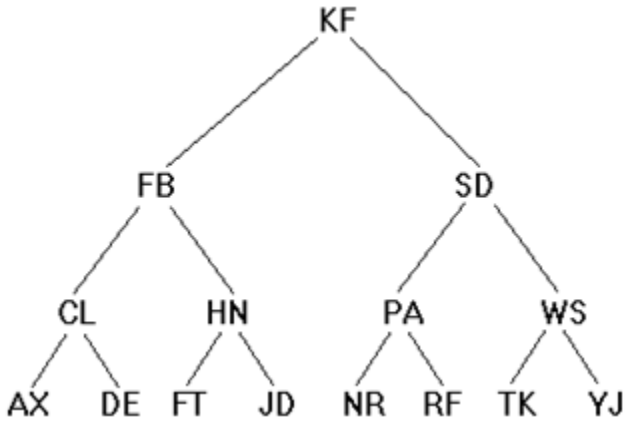
Raiz = 9

	key	filho esq.	filho dir.
0	FB	10	8
1	JD	-1	-1
2	RF	-1	-1
3	SD	6	13
4	AX	-1	-1
5	YJ	-1	-1
6	PA	11	2
7	FT	-1	-1

	key	filho esq.	filho dir.
8	HN	7	1
9	KF	0	3
10	CL	4	12
11	NR	-1	-1
12	DE	-1	-1
13	WS	14	5
14	TK	-1	-1

Motivação

- A busca em uma ABB têm custo $O(log_2 n)$ com a vantagem de **não exigir a ordenação física do arquivo**
- Elas resolvem o problema da ordenação, mas criam um outro → **como garantir o balanceamento da ABB?**
  - A busca em uma ABB degenerada é O(*n*)
  - Poderíamos balancear a árvore aplicando rotações → **Árvore AVL**



Motivação

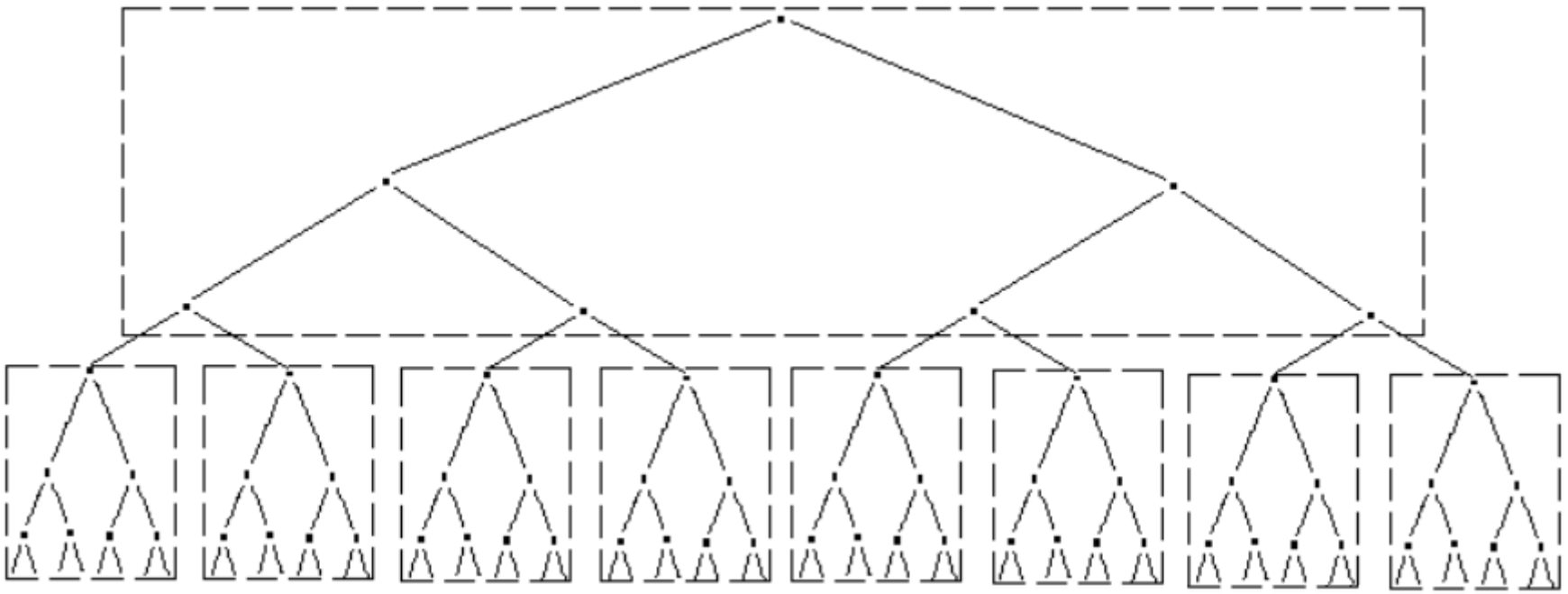
- As árvores AVL resolvem a questão da ordenação
  - Não requerem a ordenação física dos registros, nem sua reorganização sempre que houver nova inserção ou remoção (apenas as referências da árvore são ajustadas)

- Por outro lado, **não resolvem o problema do <u>número excessivo de acessos</u>**
  - Árvores binárias, mesmo balanceadas, são profundas
  - Uma busca em uma árvore AVL com 1 milhão de chaves poderia percorrer até 28 níveis → teríamos 28 leituras

- Uma solução seria armazenar e acessar a árvore AVL em bloco → conceito de **<u>página</u>**

Árvores Binárias Paginadas

- **Exemplo**: ABP com 9 páginas e 7 chaves por página
  - Qualquer um dos 63 registros pode ser acessado com, no máximo, 2 leituras (uma ABB normal poderia requer até 6 acessos)

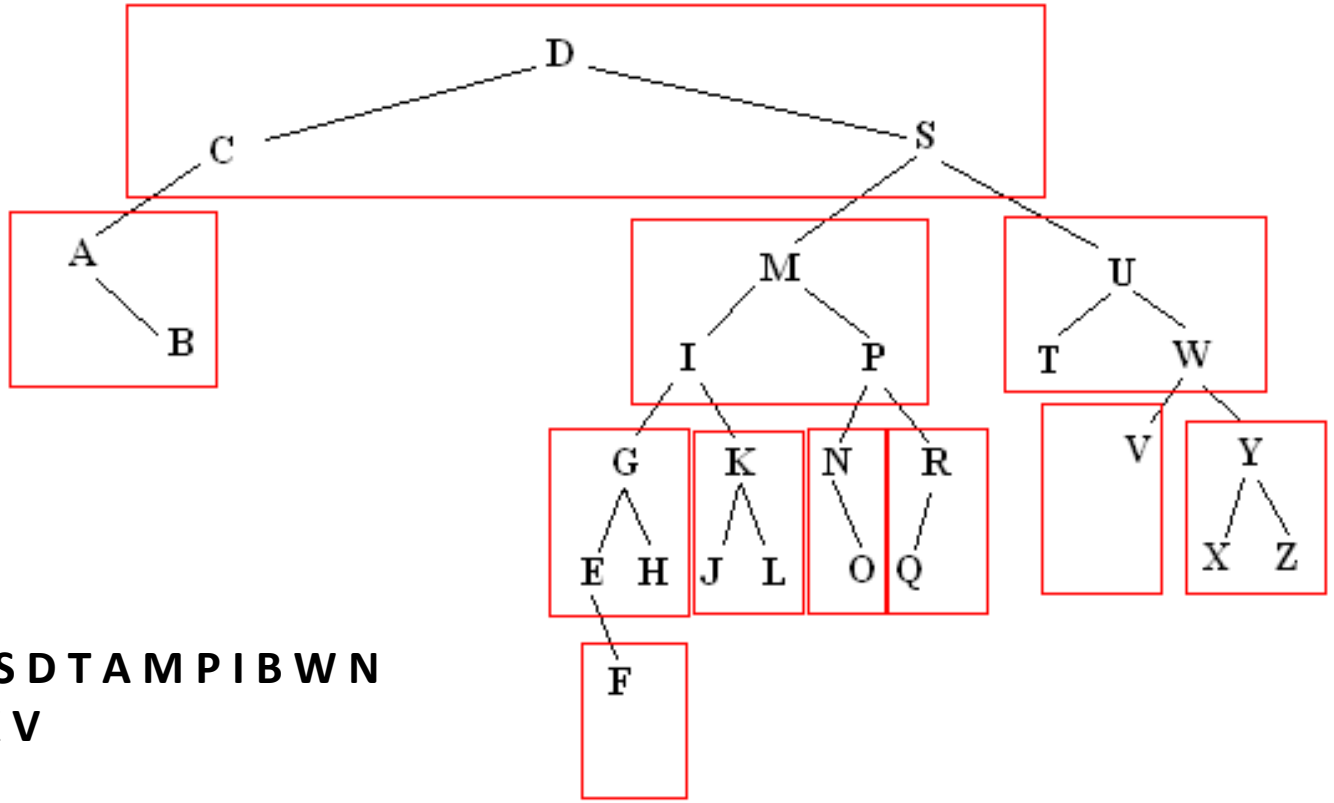


Se a árvore for estendida com mais um nível de paginação, adicionaremos 64 novas páginas e poderemos encontrar qualquer uma das 511 chaves armazenadas com apenas 3 *seeks* (uma busca binária nessas 511 chaves poderia exigir até 9 *seeks*)

Árvores Binárias Paginadas

- Exemplo de árvore binária paginada
  - Cada vez que uma chave é inserida, a <u>árvore binária dentro da página</u> pode ser rotacionada (AVL), se necessário, para manter o balanceamento

Outro problema surge: como rotacionar páginas?



Sequência de inserção: C S D T A M P I B W N G U R K E H O L J Y Q Z F X V

ABCDEFGHIJKLMNOPQRSTUVWXYZ

Motivação

- Com as árvores paginadas, as seguintes questões permaneciam abertas:
  - Como garantir que as chaves na página raiz são boas separadoras, i.e., dividem o conjunto de chaves de maneira balanceada?
  - Como impedir o agrupamento de chaves que não deveriam estar na mesma página (por ex., chaves em sequência, como C e D)?

Essas questões estão relacionadas ao fato das ABPs serem construídas de forma ***top-down***

  - Como garantir que cada página armazene um número mínimo de chaves (para que não haja desperdício de espaço)?
    - Uma nova página poderia ser criada para armazenar apenas uma chave

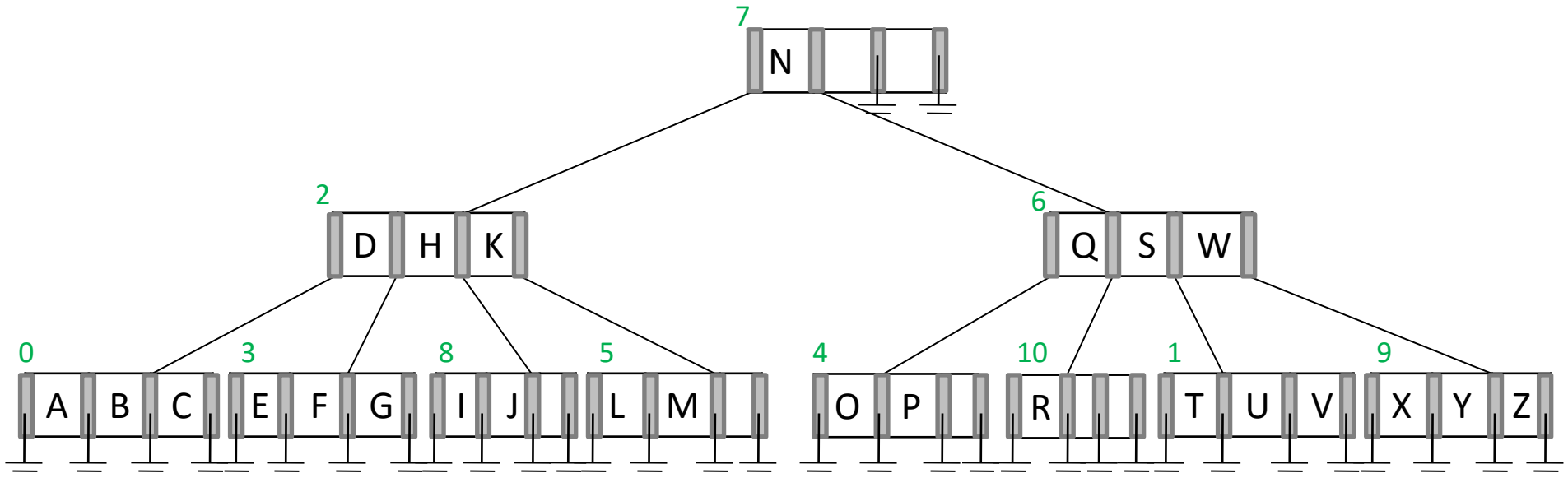
Árvore-B

- Foi nesse contexto que a **árvore-B** foi proposta
  - Bayer & McGreight, em 1972, publicaram o artigo: *"Organization and Maintenance of Large Ordered Indexes"*
  - Nos anos 80, o uso de árvores-B já era o padrão adotado em sistemas de arquivos de propósito geral para a manutenção de índices para bases de dados grandes

- As árvores-B resolvem o problema da ordenação e minimizam o número de acessos, pois elas se mantêm balanceadas

Árvore-B

- Características das árvores-B:
  - **Páginas** são um bloco de chaves e referências
  - Construção no sentido das folhas para a raiz → ***bottom-up***



Árvore-B

- Cada **página** é formada por uma **sequência ordenada de chaves** e um **conjunto de referências**
- O número máximo de **referências** é igual ao número máximo de descendentes de uma página → **ordem da árvore**
- Exemplo de uma **página** de ordem 8
  - Armazena 7 chaves e 8 referências → os blocos em cinza estão representando referências nulas

- O número de referências em uma página sempre é igual ao número de chaves + 1
- As páginas que têm todas as referências nulas são **folhas**

Propriedades da Árvore-B

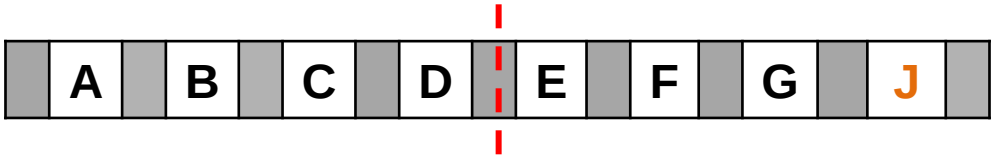
- Para uma árvore-B de ordem *m*:
  - Toda página tem um máximo de *m* descendentes
  - Toda página, exceto a raiz e as folhas, tem no mínimo $\lceil m/2 \rceil$ descendentes
  - A raiz tem pelo menos dois descendentes (a menos que também seja uma folha)
  - **Todas as folhas estão no mesmo nível**
  - Uma página não-folha com *k* descendentes contém *k* - 1 chaves
  - Uma página folha contém no mínimo $\lceil m/2 \rceil$ - 1 chaves e no máximo *m* - 1 chaves

Inserção em Árvore-B

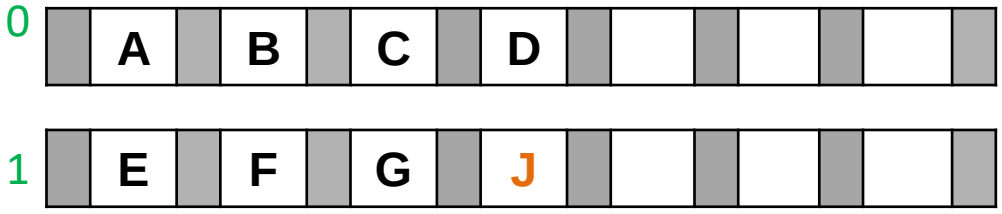
- Em uma árvore-B, toda inserção se inicia em uma folha (*bottom-up*)
- Se há espaço disponível na folha, a inserção é simples
  - A página será lida para a memória e a inserção será feita, assim como qualquer reordenação necessária, e a página é regravada no arquivo
- Se a página está cheia, a inserção envolverá as operações de **divisão** e **promoção**
- Considere a página abaixo, que é uma folha e também é a raiz de uma árvore-B de ordem 8
  - **Como inserir uma chave J nesta folha?**

Inserção em Árvore-B

- **Divisão**
  - Passamos o conteúdo da página cheia para uma página auxiliar maior e inserimos a chave na ordem correta

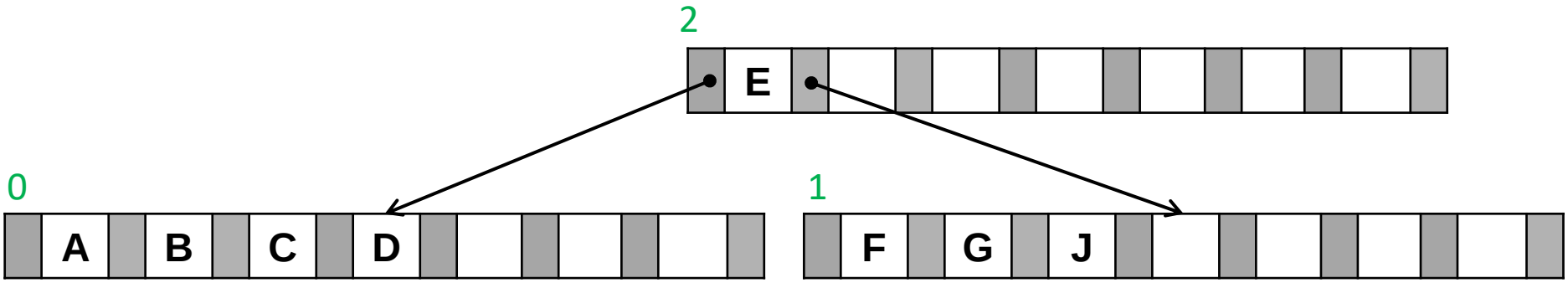


  - Criamos uma página nova e distribuímos as chaves igualmente entre as páginas



Inserção em Árvore-B

- **Promoção**
  - Para criar uma nova raiz, promovemos uma das chaves que estão nos limites de separação das folhas
  - Neste exemplo, promovemos a chave E → 1ª chave da página nova
    - **Escolhemos promover a 1ª chave da filha direita**
    - Essa política será a adotada em todos os exemplos daqui para frente



Inserção em Árvore-B

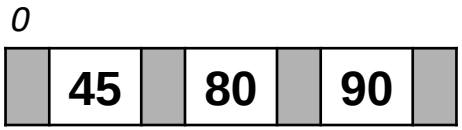
- Exercício
  - Vamos inserir a sequência de chaves abaixo em uma árvore-B de ordem 4.

    80, 45, 90, 95, 70, 15, 25, 30, 29, 8, 6, 10, 1, 2

Página de ordem 4

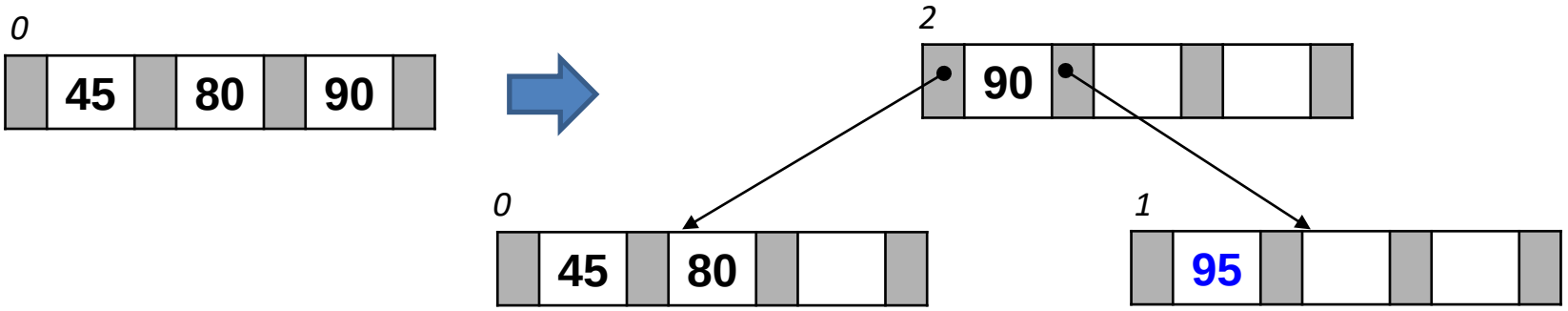
Inserção em Árvore-B

- 80, 45, 90, 95, 70, 15, 25, 30, 29, 8, 6, 10, 1, 2



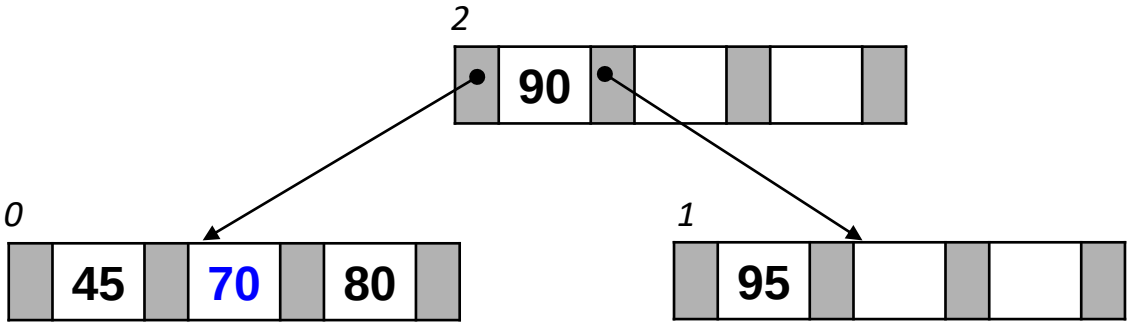
Inserção em Árvore-B

- 80, 45, 90, 95, 70, 15, 25, 30, 29, 8, 6, 10, 1, 2



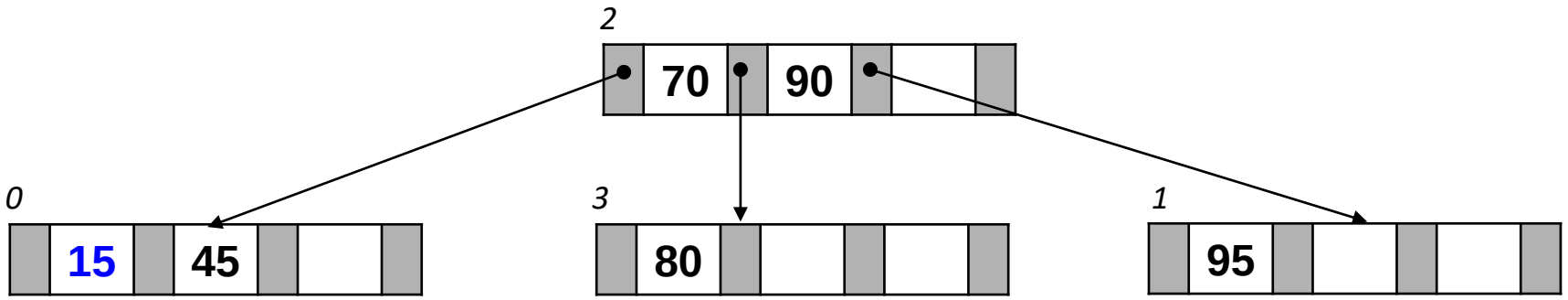
Inserção em Árvore-B

- 80, 45, 90, 95, 70, 15, 25, 30, 29, 8, 6, 10, 1, 2



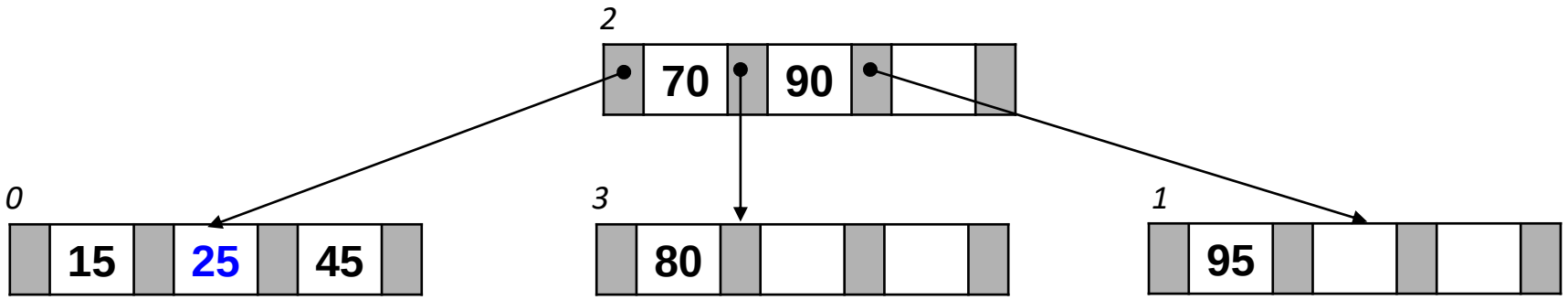
Inserção em Árvore-B

- 80, 45, 90, 95, 70, 15, 25, 30, 29, 8, 6, 10, 1, 2



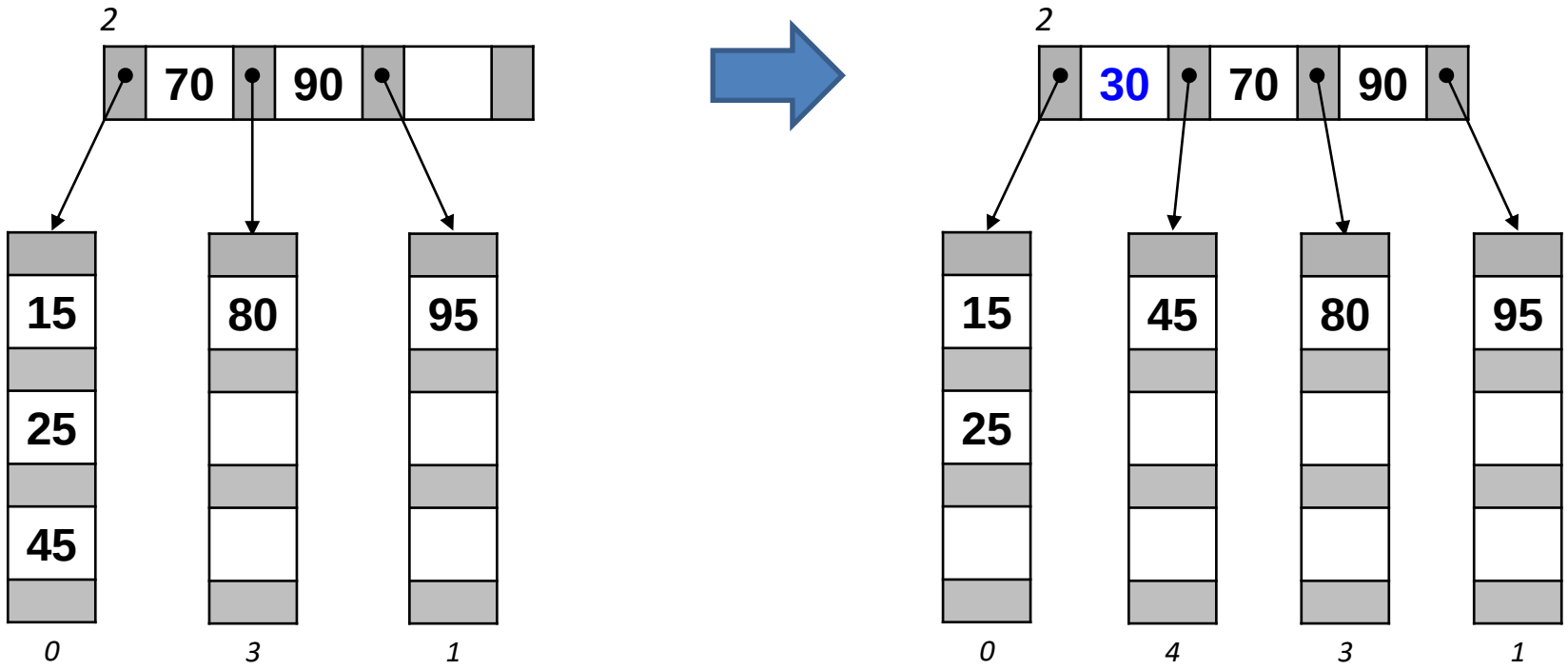
Inserção em Árvore-B

- 80, 45, 90, 95, 70, 15, 25, 30, 29, 8, 6, 10, 1, 2



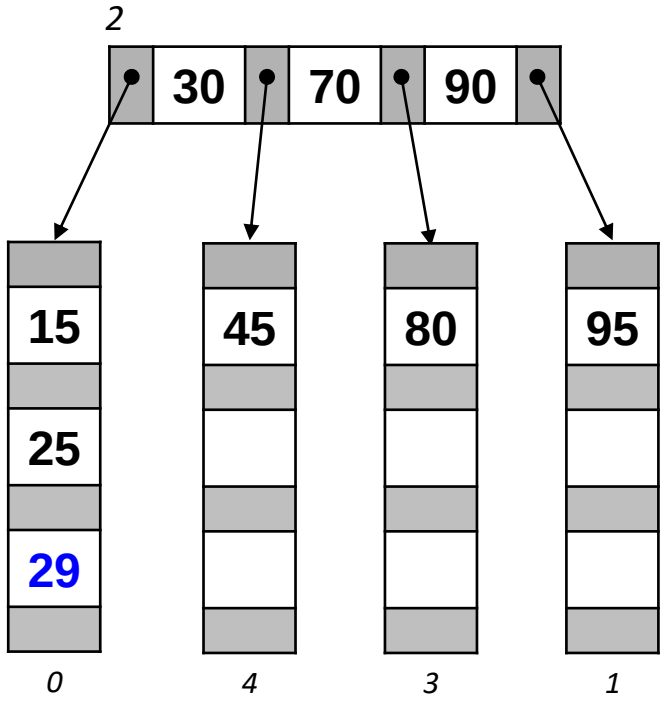
Inserção em Árvore-B

- 80, 45, 90, 95, 70, 15, 25, 30, 29, 8, 6, 10, 1, 2



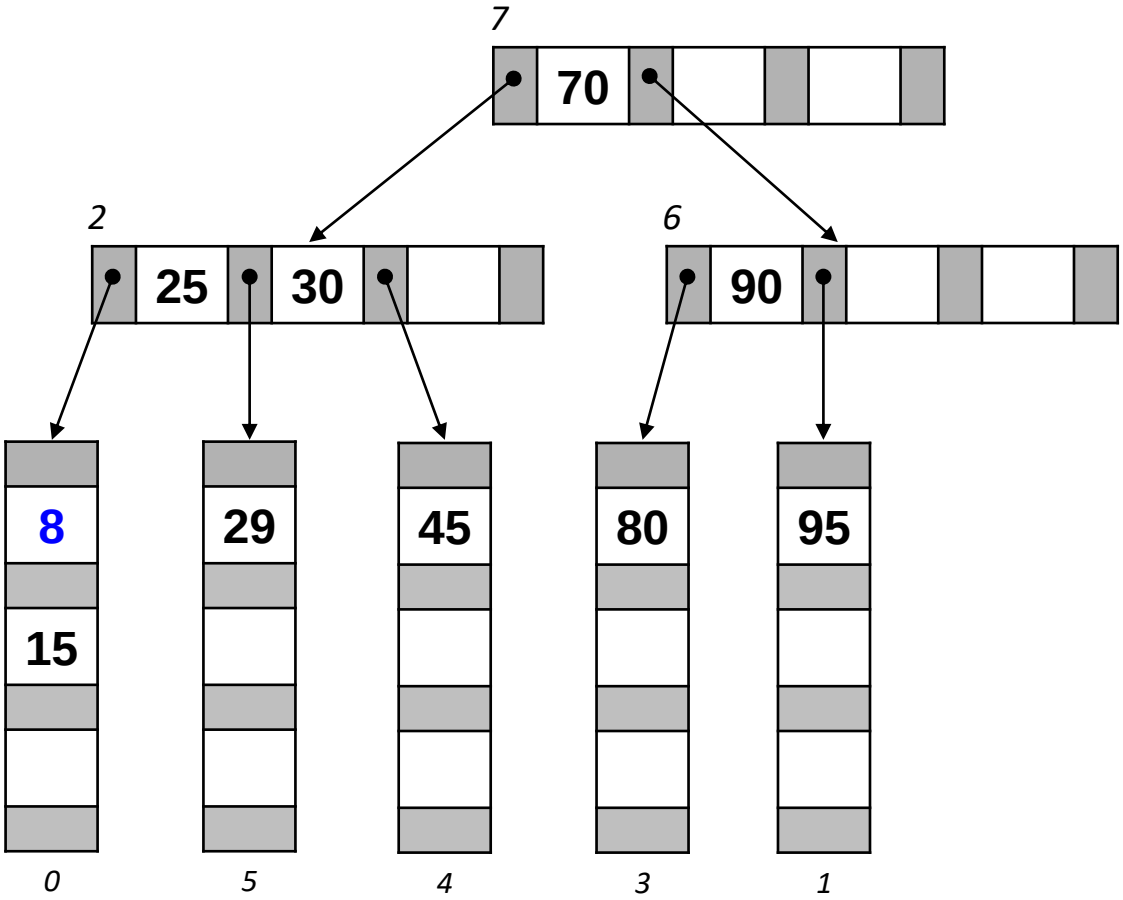
Inserção em Árvore-B

- 80, 45, 90, 95, 70, 15, 25, 30, 29, 8, 6, 10, 1, 2



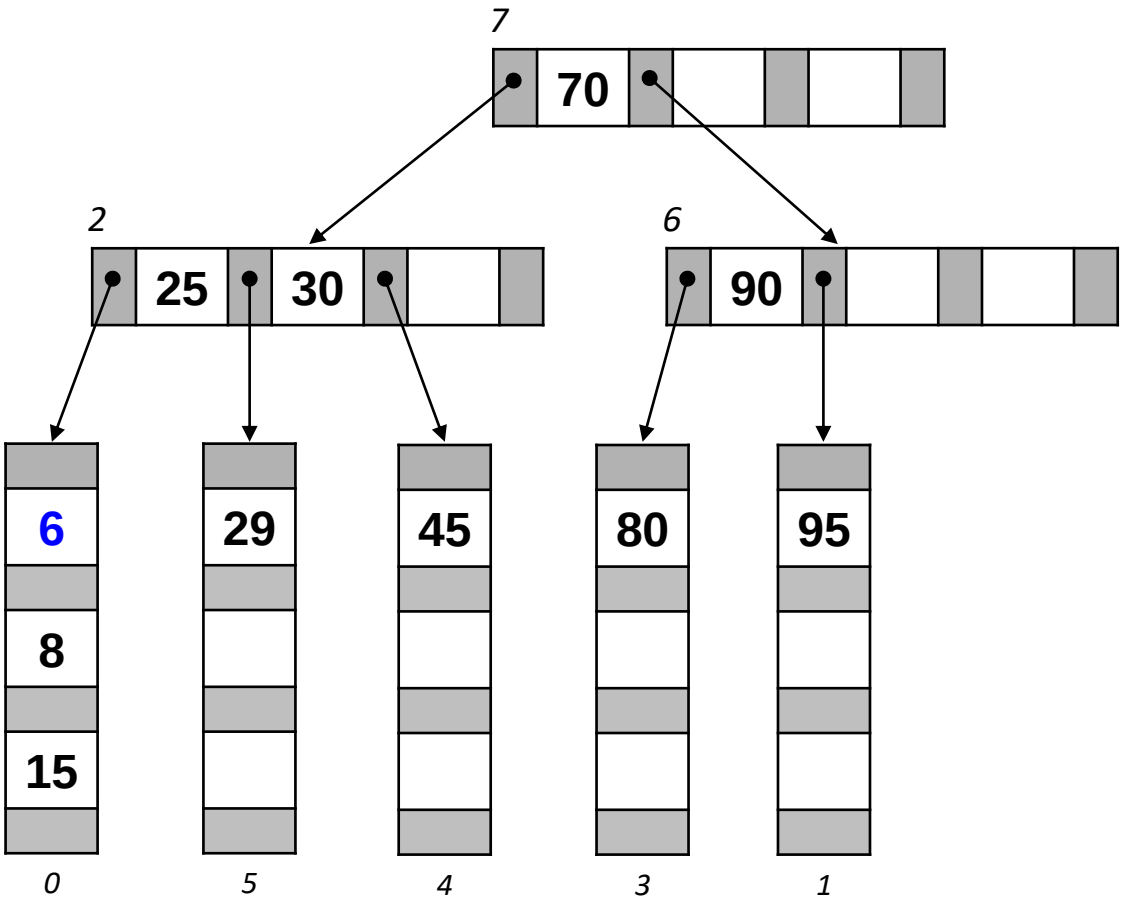
Inserção em Árvore-B

- 80, 45, 90, 95, 70, 15, 25, 30, 29, 8, 6, 10, 1, 2



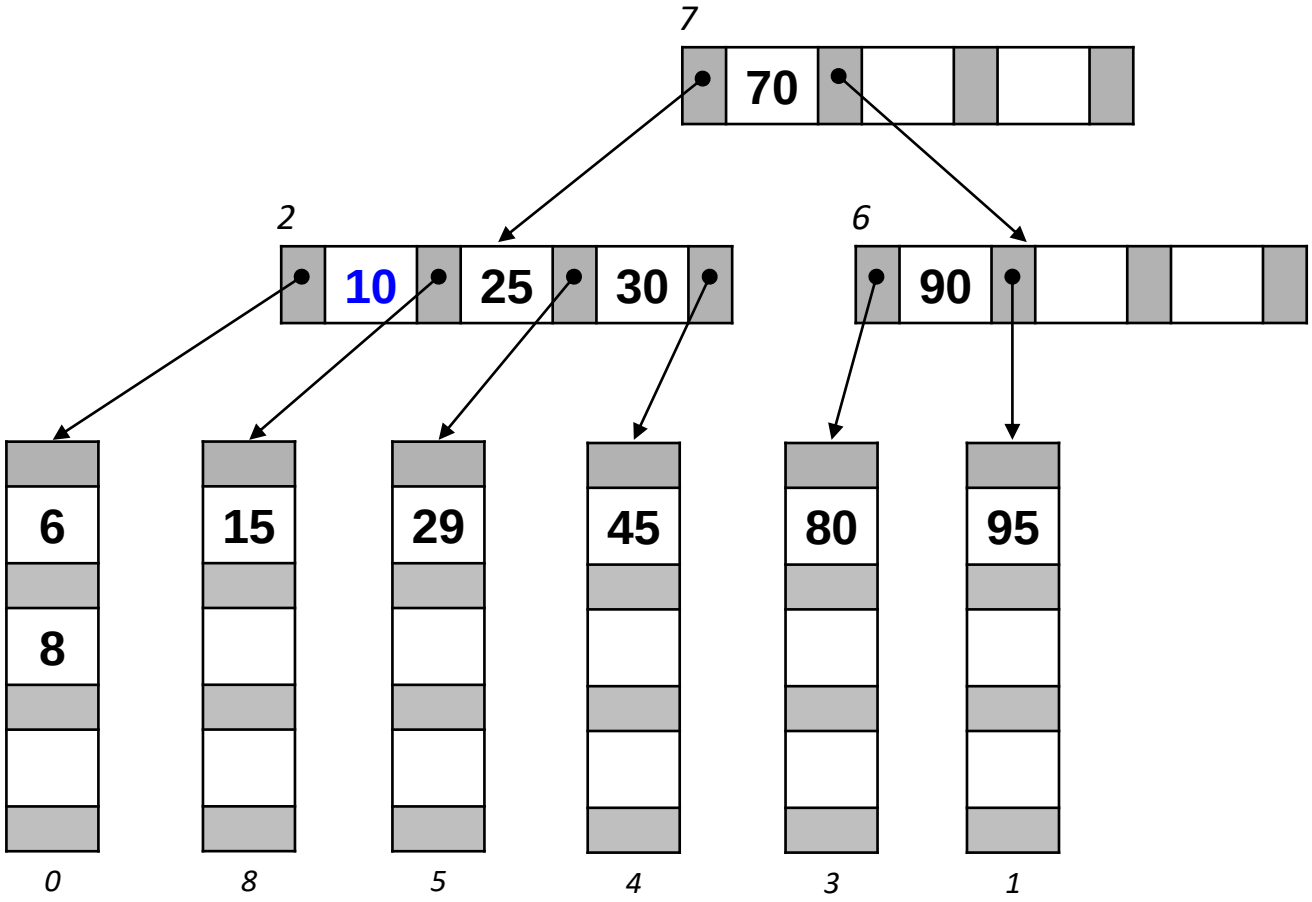
Inserção em Árvore-B

- 80, 45, 90, 95, 70, 15, 25, 30, 29, 8, 6, 10, 1, 2



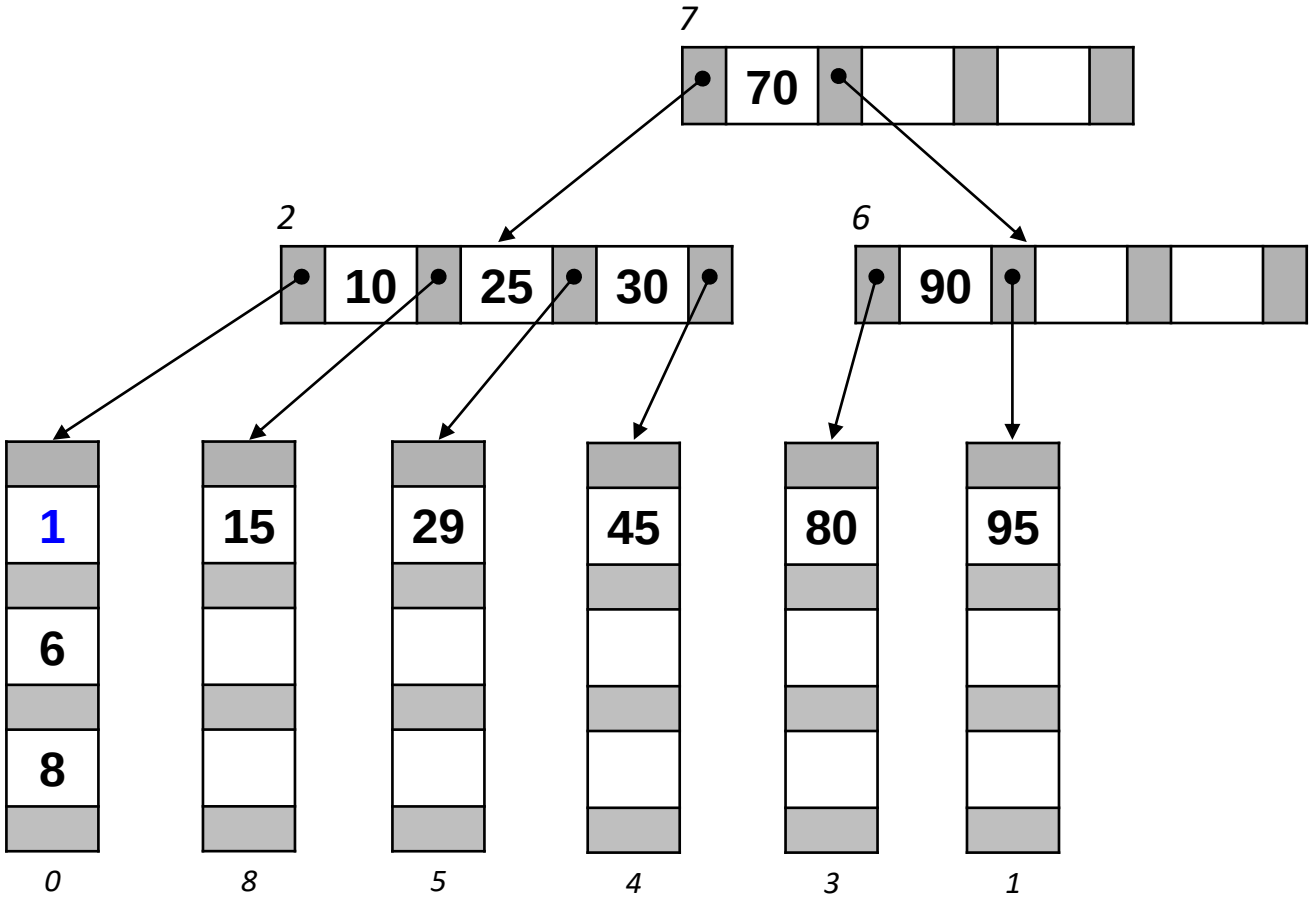
Inserção em Árvore-B

- 80, 45, 90, 95, 70, 15, 25, 30, 29, 8, 6, 10, 1, 2



Inserção em Árvore-B

- 80, 45, 90, 95, 70, 15, 25, 30, 29, 8, 6, 10, 1, 2



Inserção em Árvore-B

- 80, 45, 90, 95, 70, 15, 25, 30, 29, 8, 6, 10, 1, 2