ÁRVORES - B

Árvores

- Motivação
  - Quando não conseguimos trabalhar na memória principal (ou primária), temos que usar a memória secundária...
  - Sabemos que o acesso aos dados em memória secundária é muito lento.
  - Precisamos de meios eficientes de acesso aos dados (provavelmente na forma de ‘’índices”)

Árvores

- Motivação (recordando...):
  - Assuma que um disco gire a 3600 RPM
  - Em 1 minuto faz 3.600 rotações, portanto uma rotação leva 1/60 de segundo, ou 16.7ms
  - Na média cada acesso gastaria 8ms
  - Parece ok até nos darmos conta que 120 acessos a disco consomem um segundo – o mesmo que 25 millhões de instruções
  - Ou seja, um acesso a disco é equivalente a 200.000 instruções

Árvores

- Soluções?
  - Árvores... (AVLs, Árvores-B,...)

Árvores

- Para árvores balanceadas com n itens, as operações na árvore (inserção etc) são *O(log n)* porque a altura da árvore é aproximadamente *log n.*

Exemplos:

binary tree c/ 1000 itens:

h ~= $\log_2$ 1000 ~= 10

10-ary tree c/ 1000 itens:

h ~= $\log_{10}$ 1000 ~= 3

Árvores

- Assuma que usaremos uma AVL para armazenar dados de motoristas (+/- 20 milhões de registros)
- Teríamos uma árvore bem alta (vários acessos a disco);
- $\log_2$ 20.000.000 é +/- 24, o que consome +/- 0.2 segundos
- A solução é aumentar o número de ramificações na árvore diminuindo, assim, a altura!

Árvores

TRADEOFF

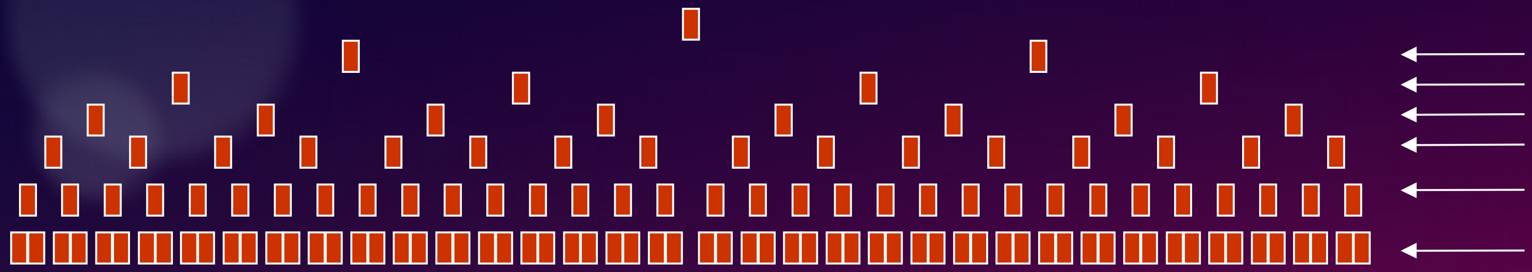
Fator de ramificação

- Complexidade de comparações
- Tamanho do nó

Árvores

- Árvores binárias são o caso extremo:
  - Fator mínimo de ramificação (2)
  - Máxima profundidade (muitos acessos)
- Se os acessos são caros (armaze-namento secundário), o desempenho cai…

Árvores



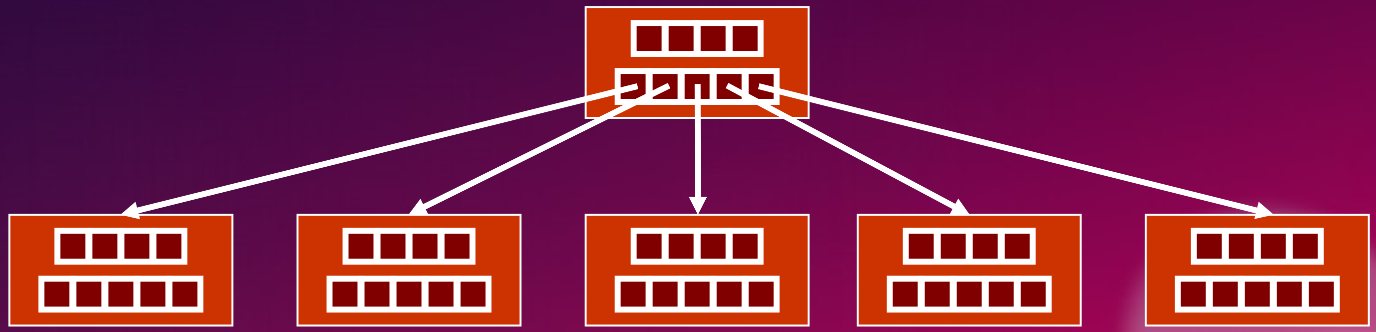
Árvore binária com 127 nós em 7 níveis.

Árvores

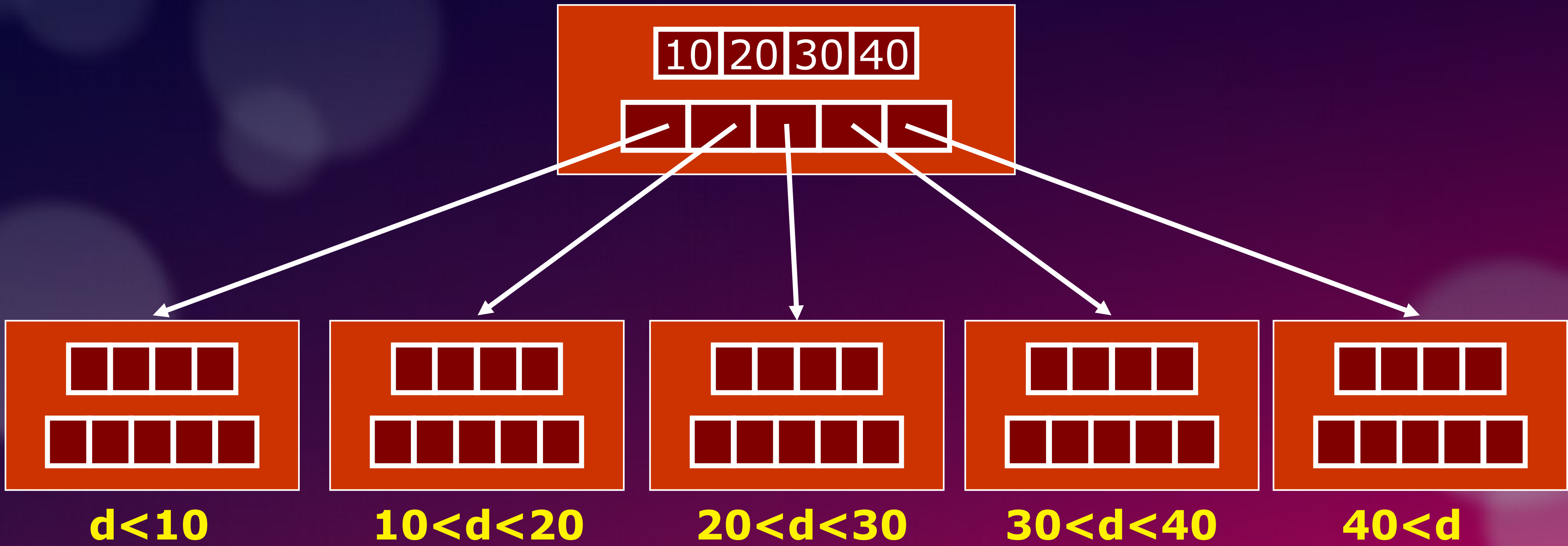
Árvore 10-aria com 127 nós em 3 níveis.

Árvores n-árias

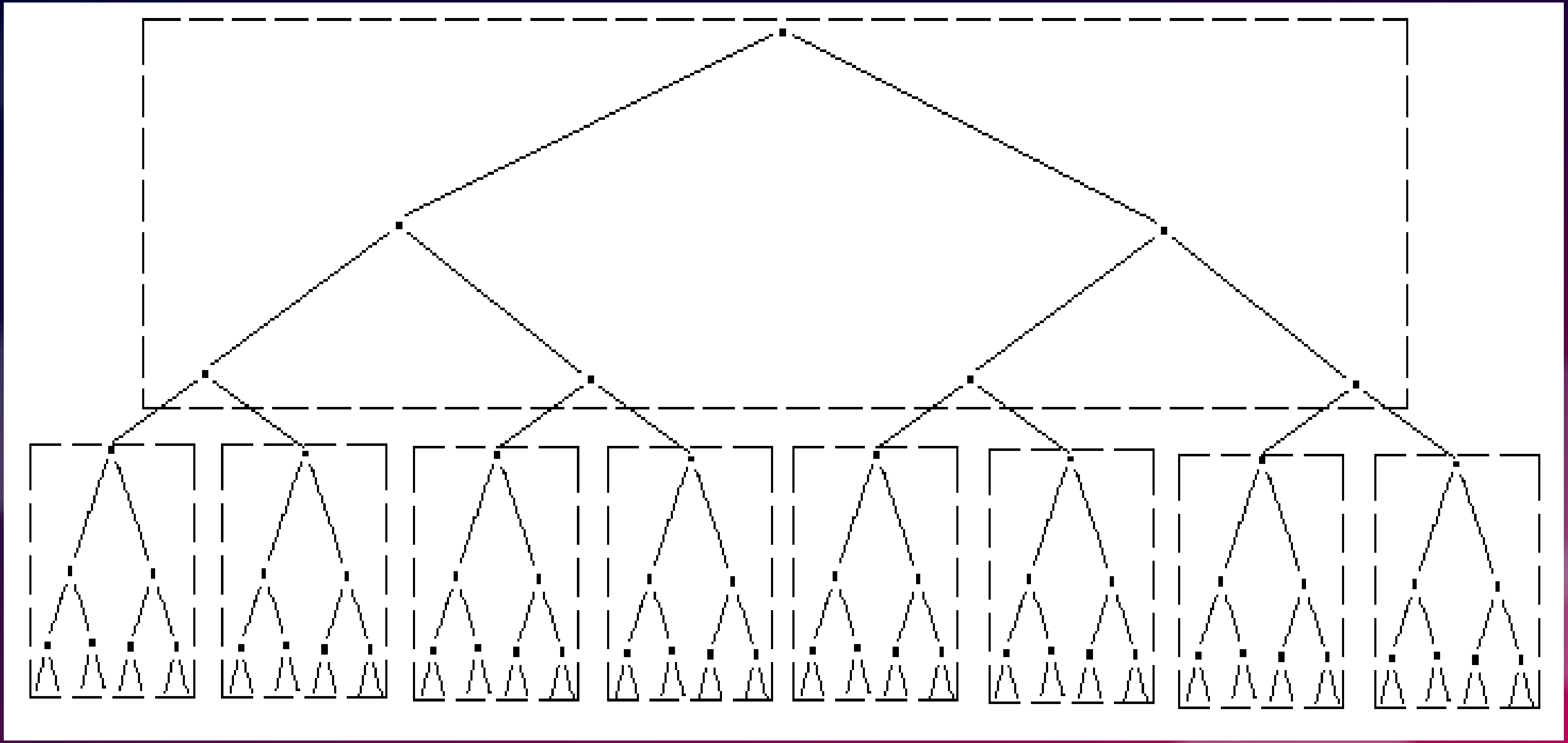
- n ponteiros
- n-1 chaves



Árvores n-árias



Árvores (paged binary trees)

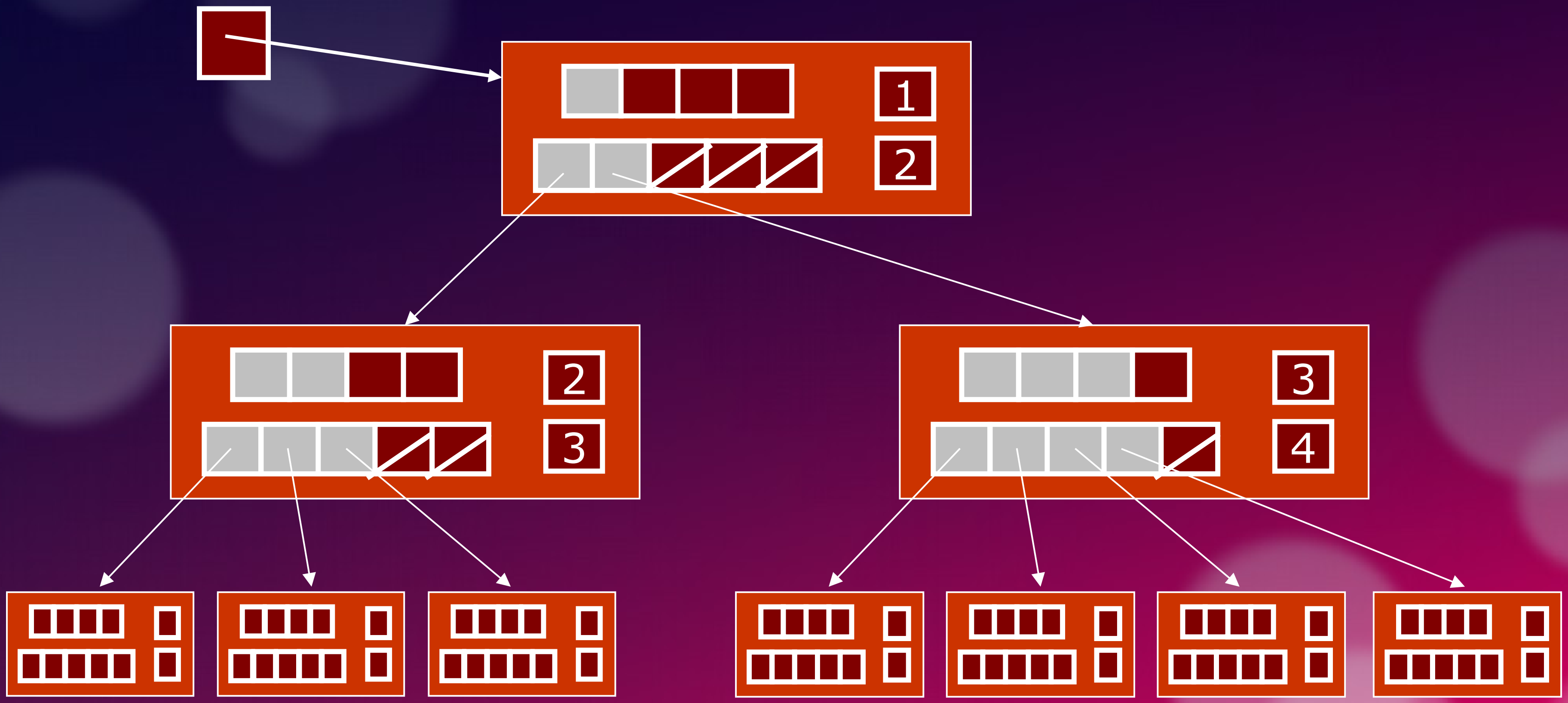


A divisão de uma árvore binária em páginas é ilustrada na figura acima. Nessa árvore de 9 páginas, quaisquer dos 63 registros pode ser acessado em, no máximo, 2 acessos.

Árvore B

```
class NO_BTree
{
Private:
   Tipo    chave[20];
   NO_BTree  p[21];
   int  Qdade_chaves;
Public:
  métodos…
}
```

Árvore B



Árvore B

Uma árvore B de **ordem *m*** é uma árvore *m*-way (i.e., uma árvore onde cada nó pode ter até *m* filhos) e que:

1 .O número de chaves em cada nó não folha é um a menos que o número de filhos e cada filho está organizado no contexto de árvore de busca;

2. Todas as folhas estão no mesmo nível;

3. Todas as não-folhas - menos a raíz – têm no mínimo $\lceil m / 2 \rceil$ filhos;

4. A raíz ou é uma folha ou tem de 2 a *m* filhos;

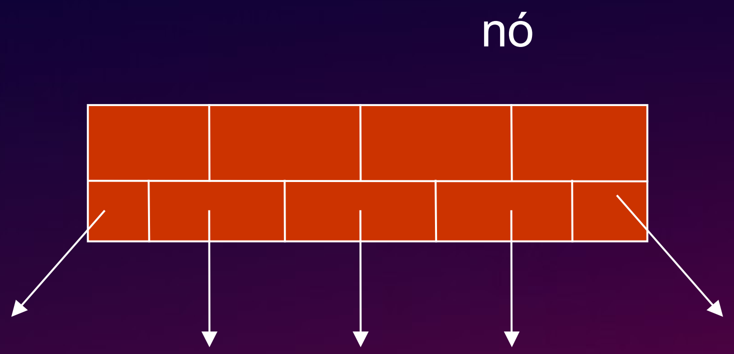
5. Um nó folha não contém mais que $m - 1$ chaves;

6. O número *m* deve ser sempre ímpar;

Árvore B

- **Ordem**

  A definição atual de **B-Tree** vincula a ordem de uma árvore B ao número de descententes de um nó (isto é, de ponteiros). Deste modo, numa árvore B de ordem **m**, o número máximo de chaves é m-1.
  Exemplo:
  Uma árvore B de ordem 8 tem um máximo de 7 chaves por página.

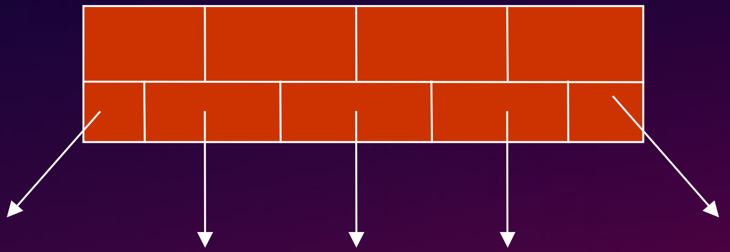


A árvore acima é de ordem 5.

Árvore B

- **Número mínimo de chaves por página**

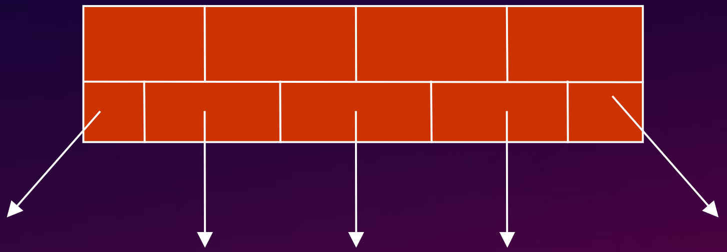
  Quando uma página é dividida na inserção (SPLIT), os nós são divididos igualmente entre as páginas velha e nova. Deste modo, o número mínimo de chaves em um nó é dado por m/2 - 1 (exceto para a raiz).



Árvore B

- **Número mínimo de chaves por página**

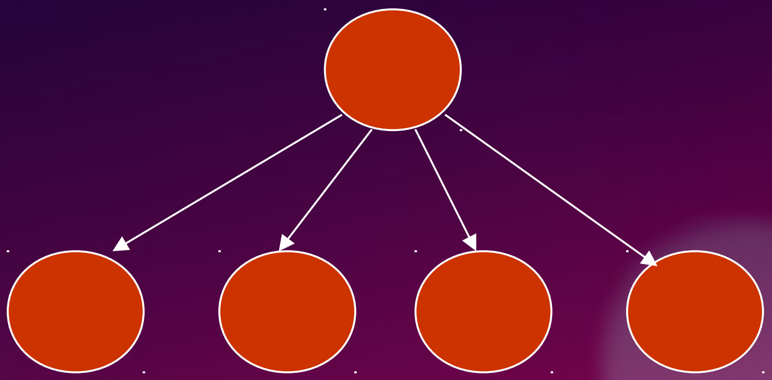
  Exemplo: Uma árvore B de ordem 8 (que tem um máximo de 7 chaves por página) terá um mínimo de 3 chaves por página.



Árvore B

- **Nó folha**

  Os nós folhas são aqueles alocados no nível mais baixo da árvore.



Árvore B

- **Capacidade Máxima**

Nós com no máximo 1000 elementos:

- h 0: 1000
- h 1: 1000+1001*1000 = 1.002.000
- h 2: ~1 Bilhão

Árvore B

- **Capacidade Mínima**
  (para árvore de 2 níveis)

Nós com no máximo 1000 elementos:

~500,000

(500*501*2 + 500*2 + 1)

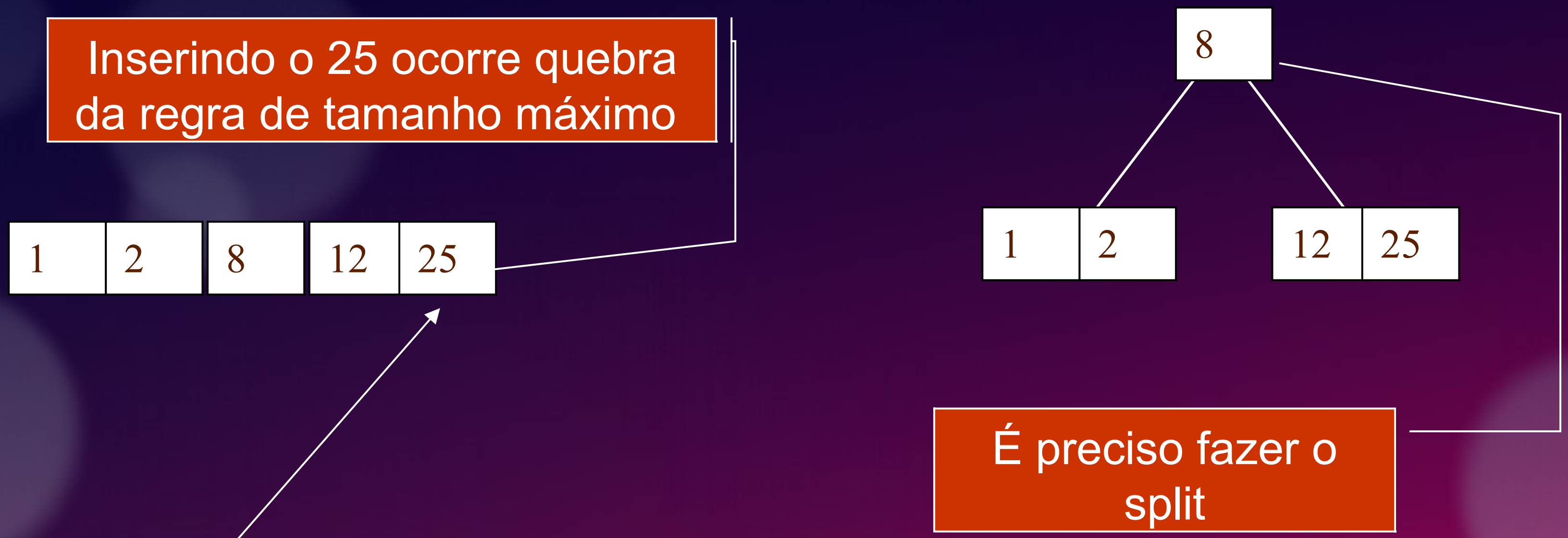
Árvore B

- Suponha que iniciemos com uma árvore B vazia e as chaves devem ser inseridas na seguinte ordem:1 12 8 2 25 6 14 28 17 7 52 16 48 68 3 26 29 53 55 45
- Queremos construir uma árvore B de ordem 5
- Os 4 primeiros elementos vão para a raíz:

1	2	8	12

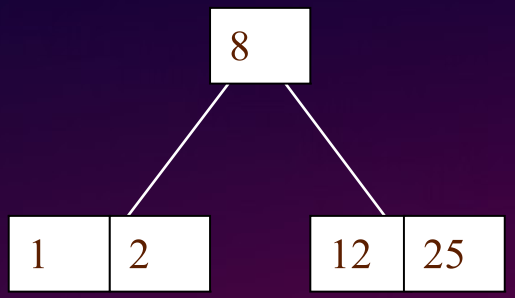
- O quinto elemento extrapola o tamanho do nó
- Assim, quando inserimos o 25 devemos dividir o nó em duas partes e colocar o elemento do meio como nova raiz

Árvore B

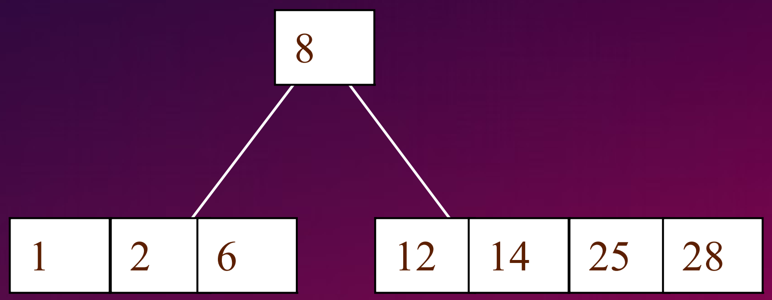


25 5 14 28 17 7 52 16 48 68 3 26 29 53 55 45

Árvore B



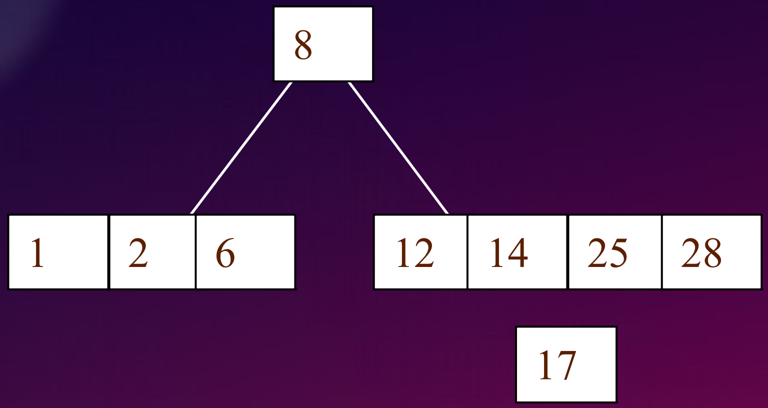
Em seguida colocamos 6, 14 e 28 :



6 14 28 17 7 52 16 48 68 3 26 29 53 55 45

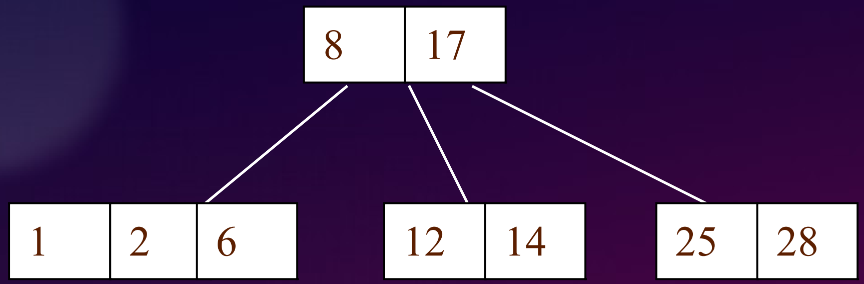
Árvore B

Adicionando 17 à árvore teremos outro split…



17 7 52 16 48 68 3 26 29 53 55 45

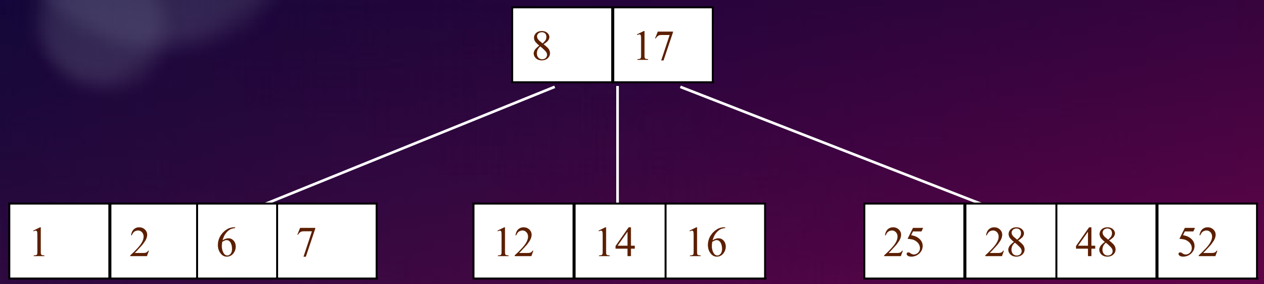
Árvore B



17 7 52 16 48 68 3 26 29 53 55 45

Árvore B

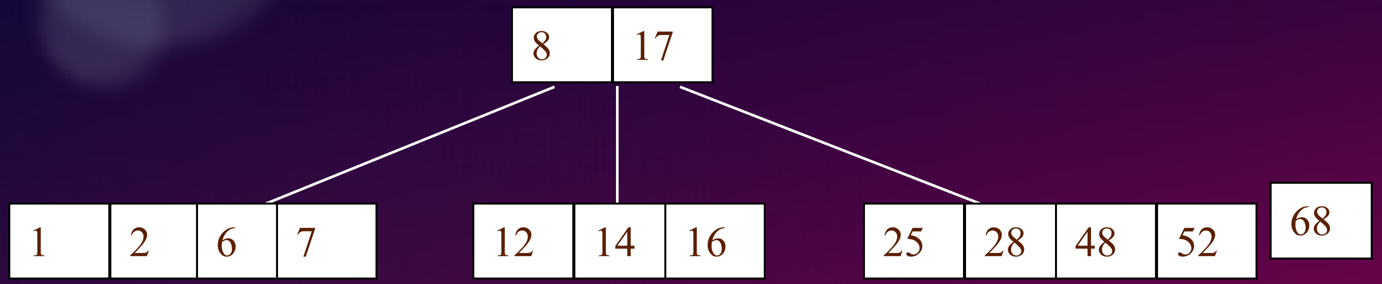
Continuando com 7, 52, 16 e 48



7 52 16 48 68 3 26 29 53 55 45

Árvore B

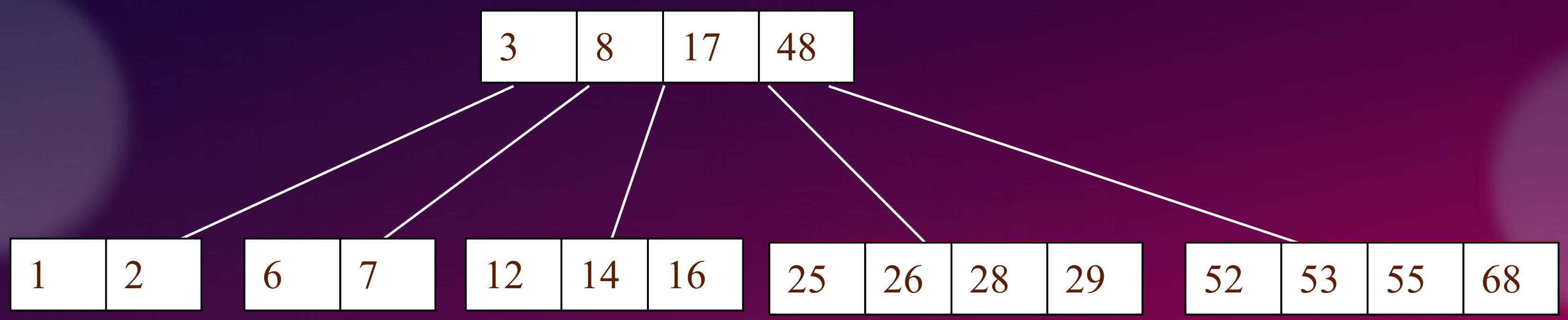
E agora, inserindo o 68…



68 3 26 29 53 55 45

Árvore B

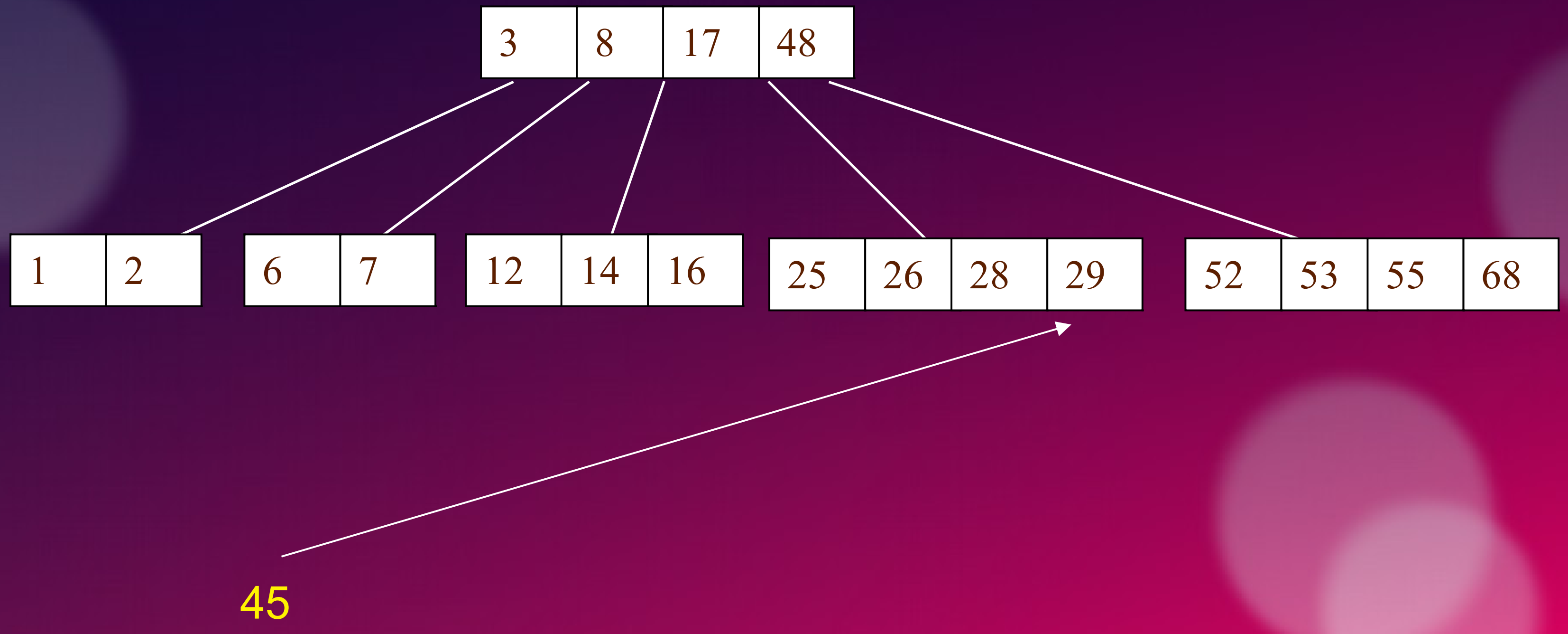
Adicionando 68 à árvore causa um “split” na folha mais à direita, fazendo com que o 48 suba à raiz. Quando inserimos o 3 o “split” é na folha mais à esquerda (o 3 sobe); 26, 29, 53, 55 vão para as folhas:



68 3 26 29 53 55 45

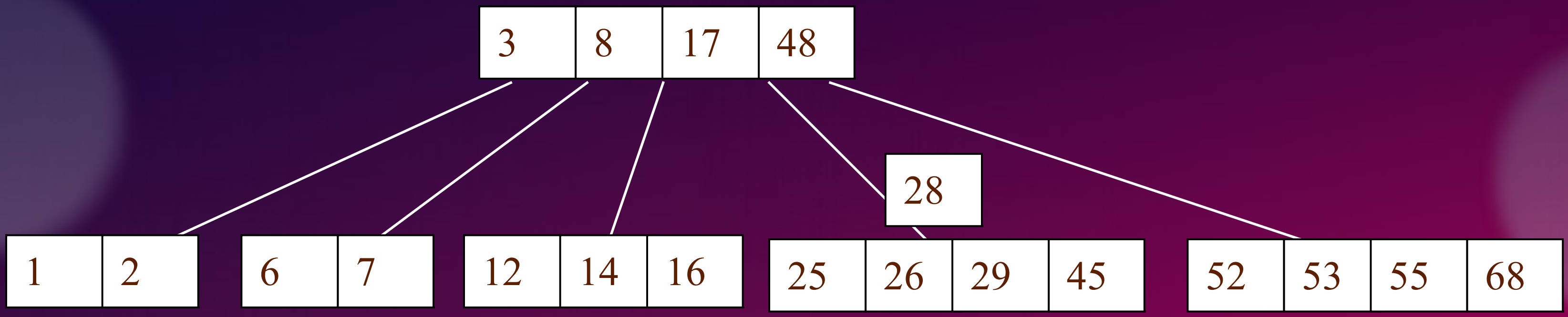
Árvore B

Por fim o 45:



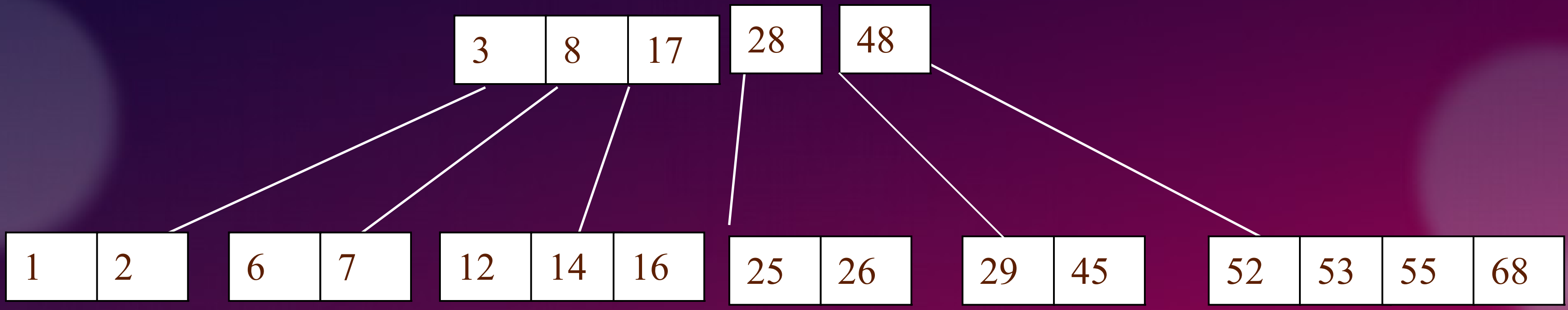
Árvore B

Por fim, quando inserimos o 45, isso forçará com que o 28 suba para a raiz… Mas a raiz também está cheia !



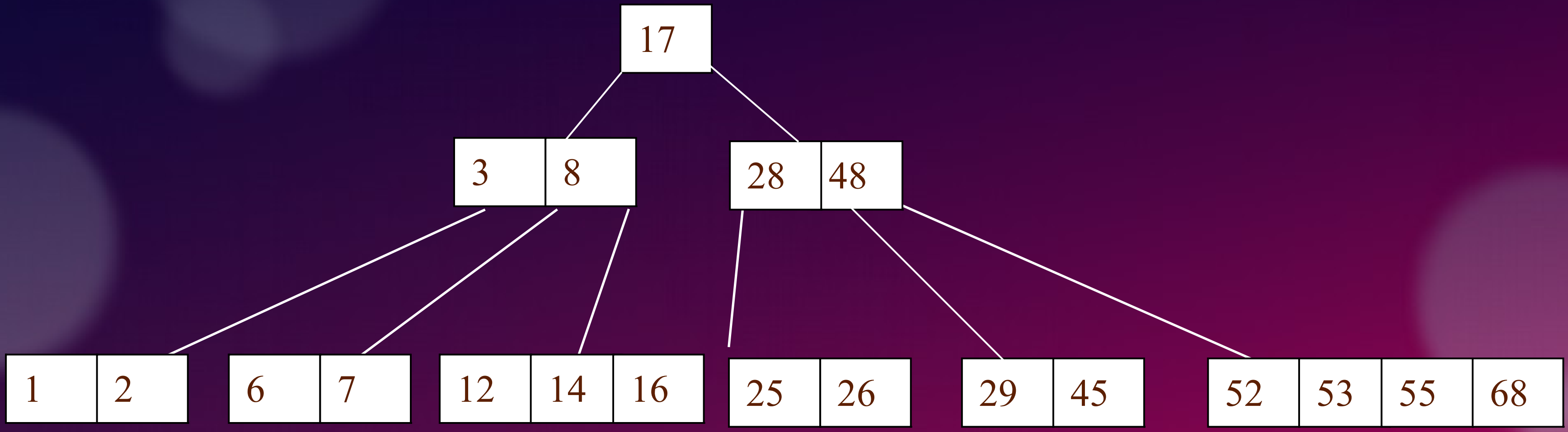
Árvore B

Por fim, quando inserimos o 45, isso forçará com que o 28 suba para a raiz… Mas a raiz também está cheia !

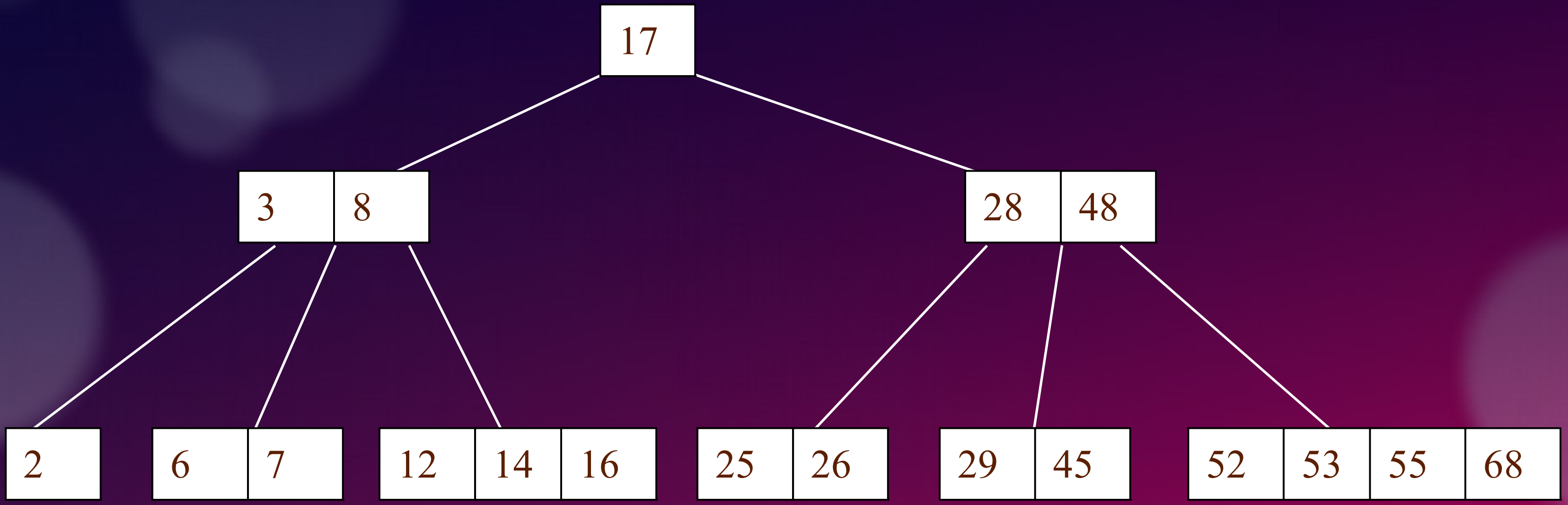


Árvore B

O 17 tem que subir para se tornar a nova raiz… lembrem-se que a raiz pode ter um único elemento.



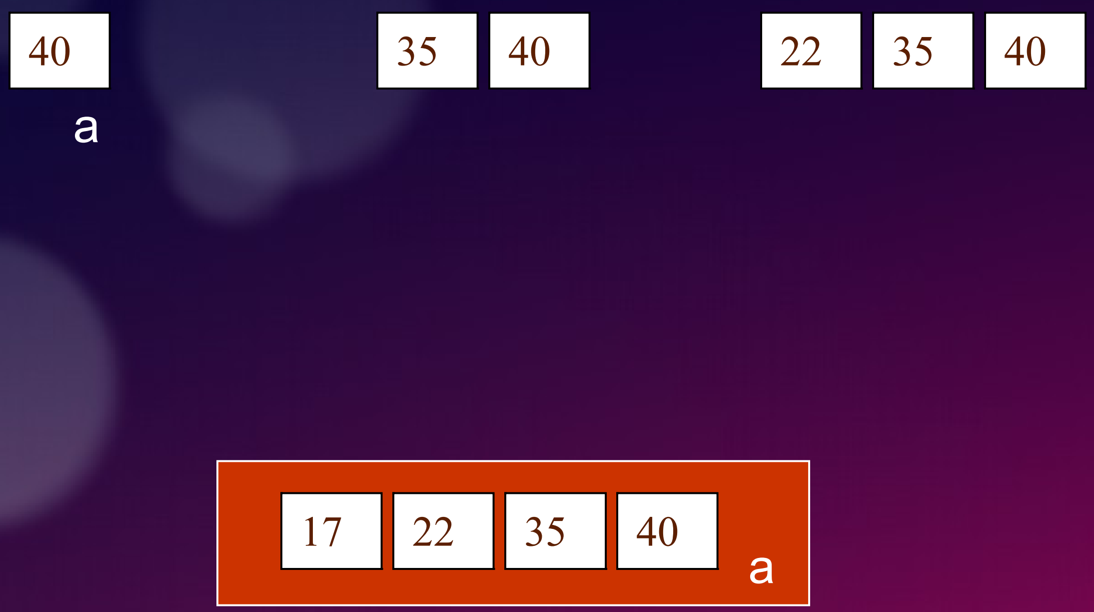
Árvore B



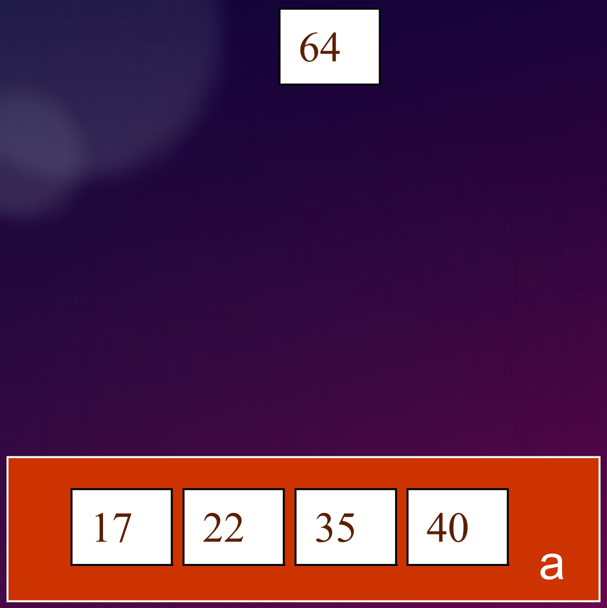
Árvore B

40 35 22 17 64 128 256 79 110 45 20 11 200 419 66 75 318 44 122 350 390 12 -45 14 -4 -10 5 500 100 13 25 32 49 39 93 7 3 -12

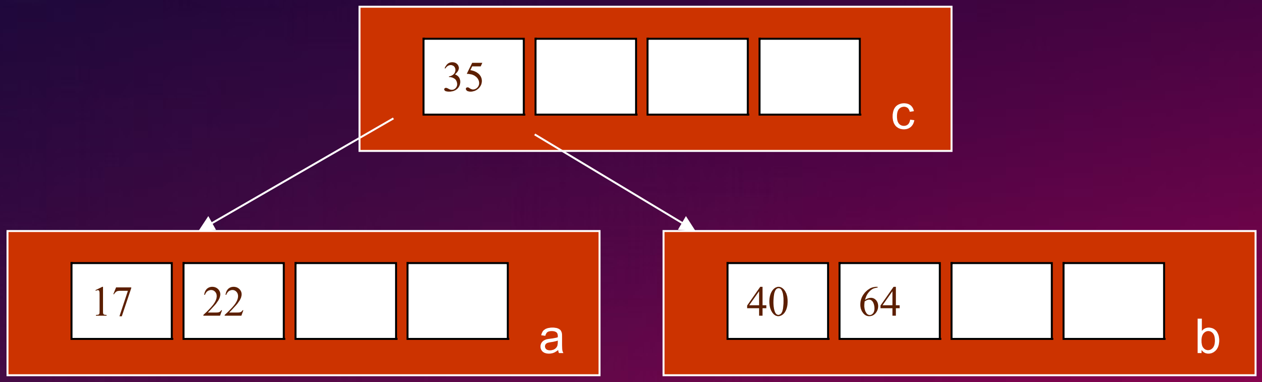
Árvore B



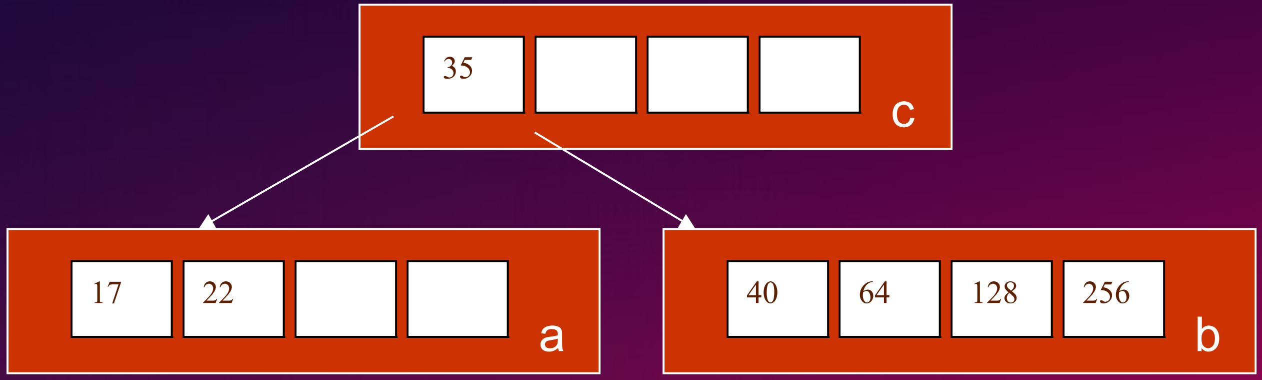
Árvore B



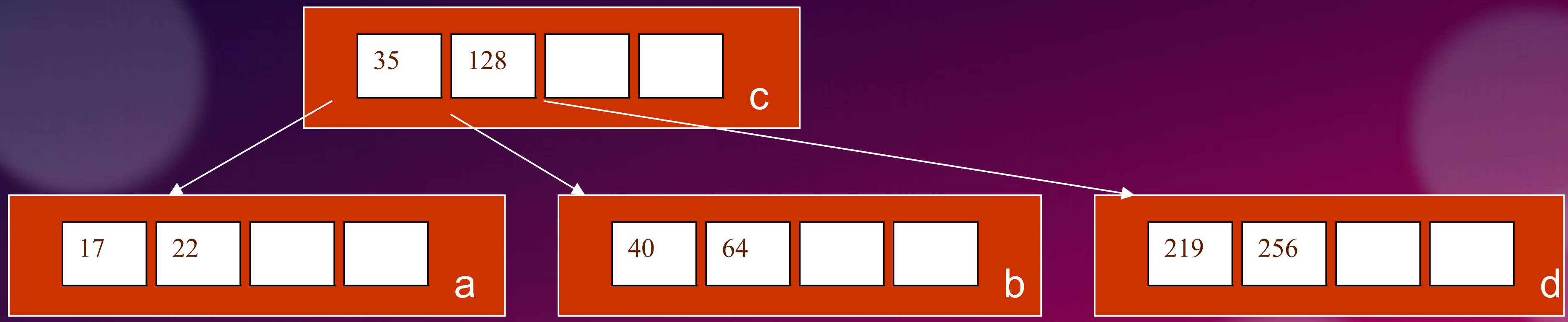
Árvore B



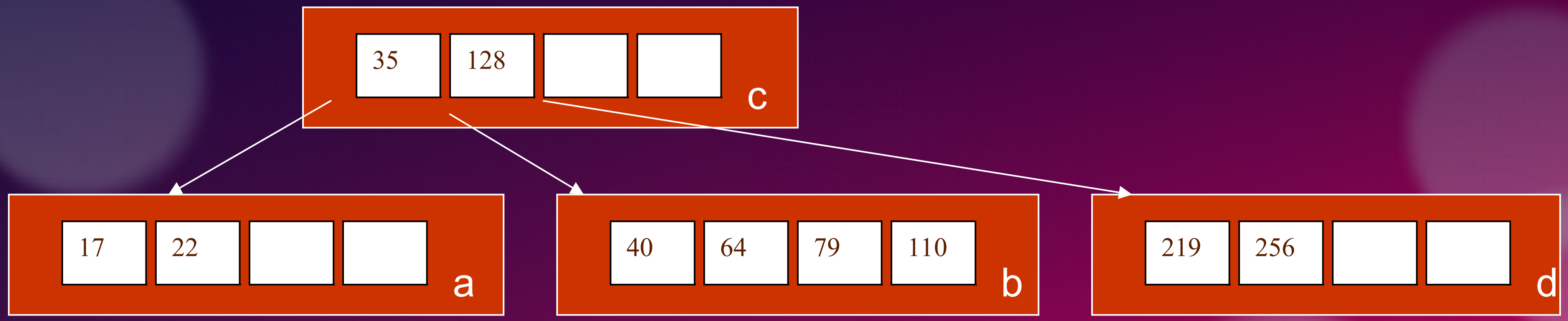
Árvore B



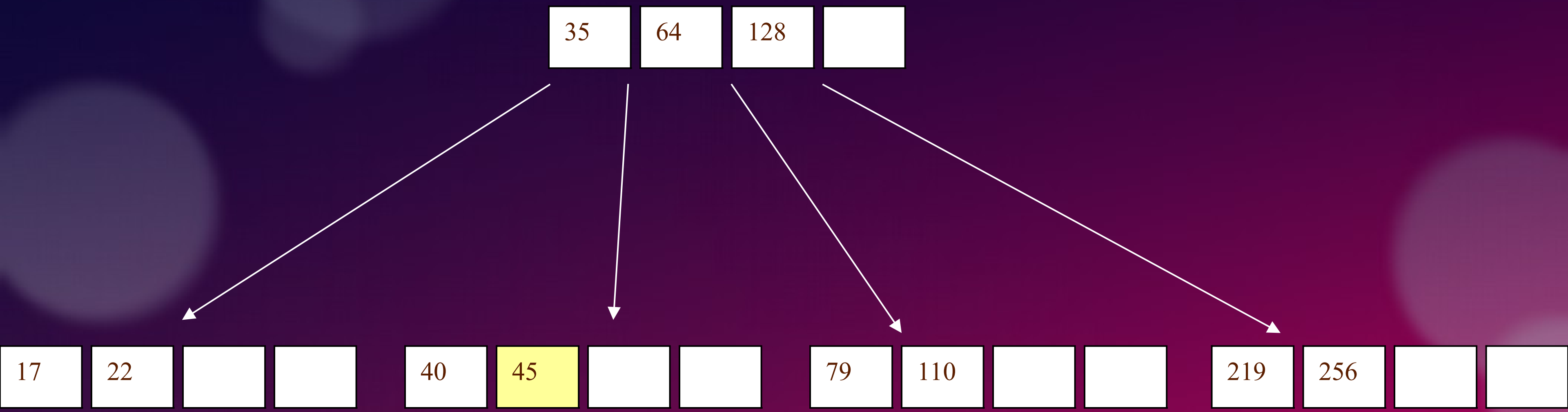
Árvore B



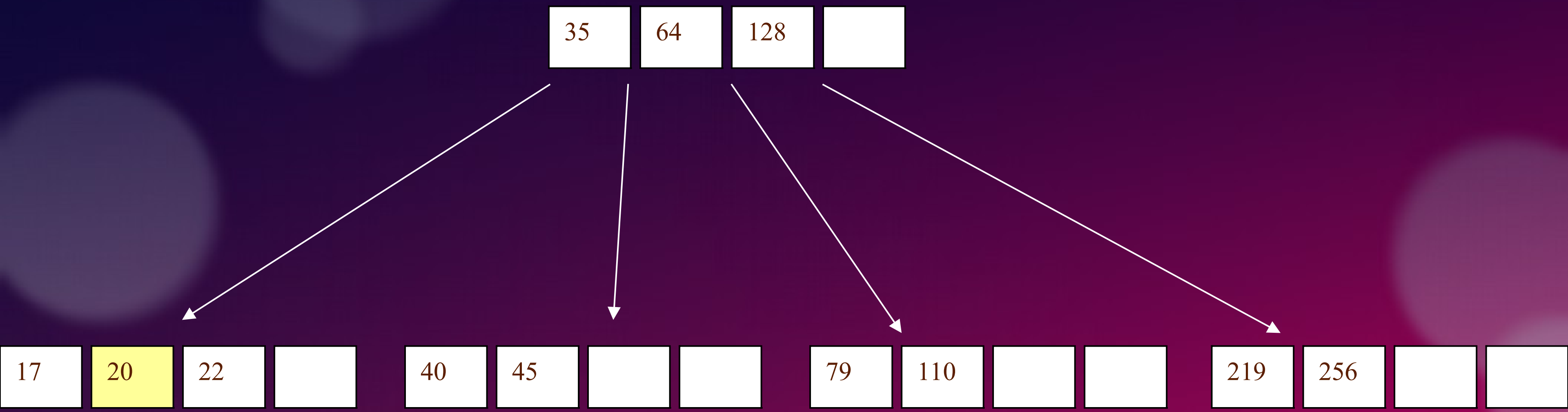
Árvore B



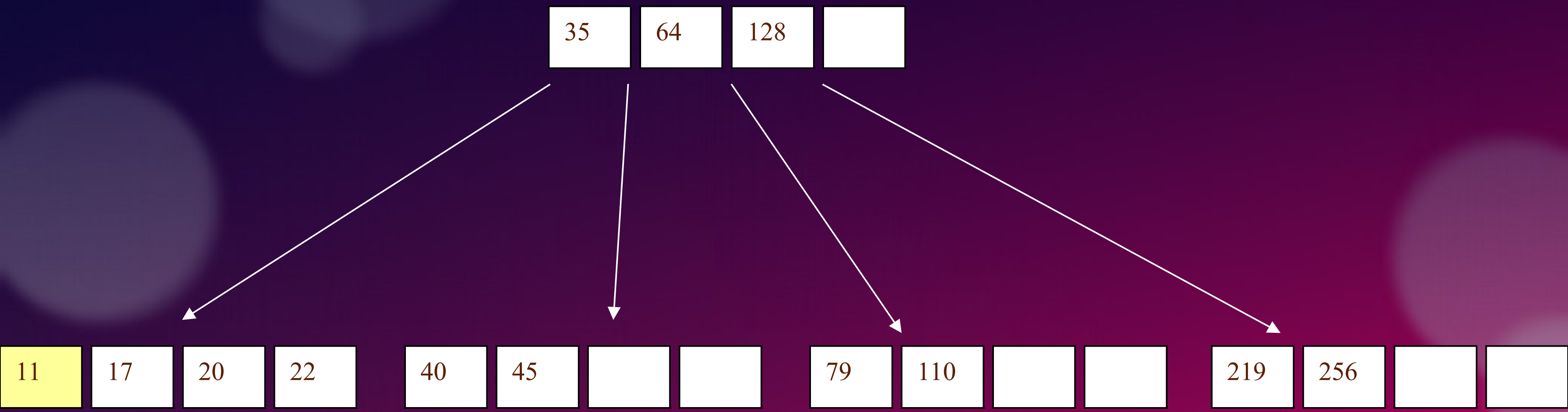
Árvore B



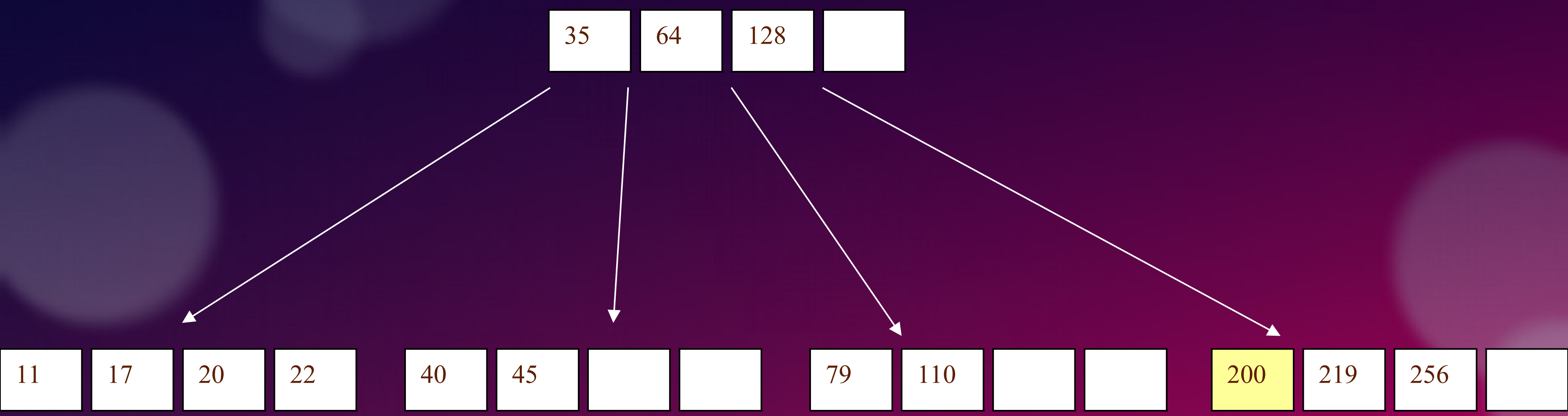
Árvore B



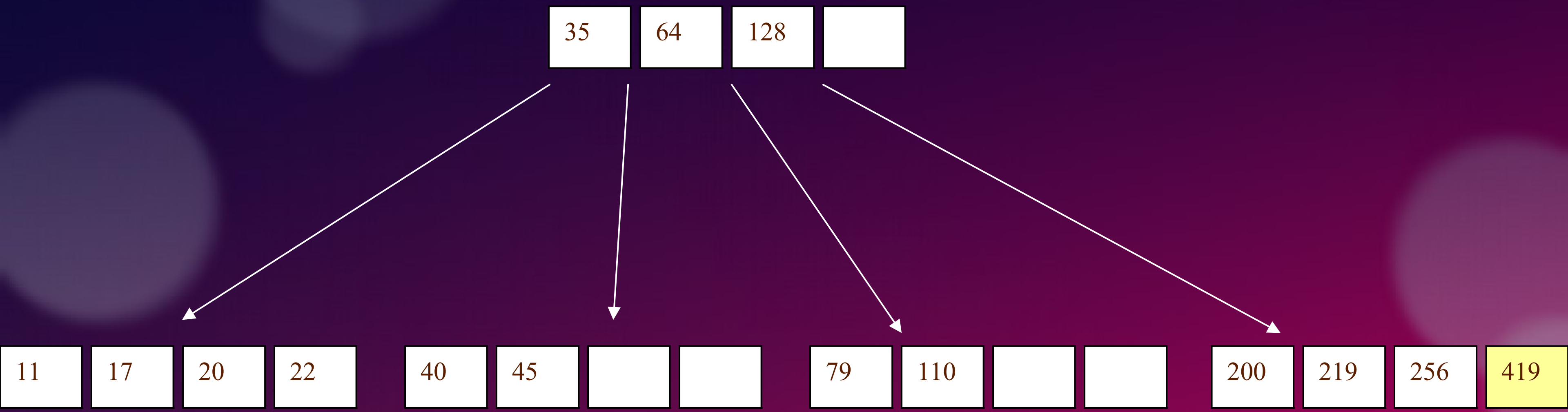
Árvore B



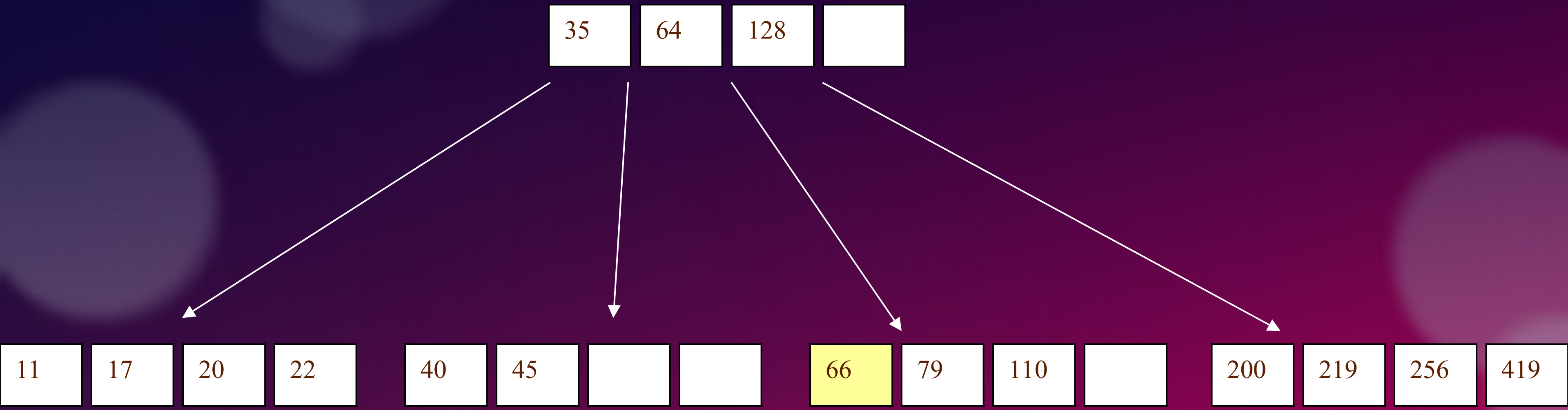
Árvore B



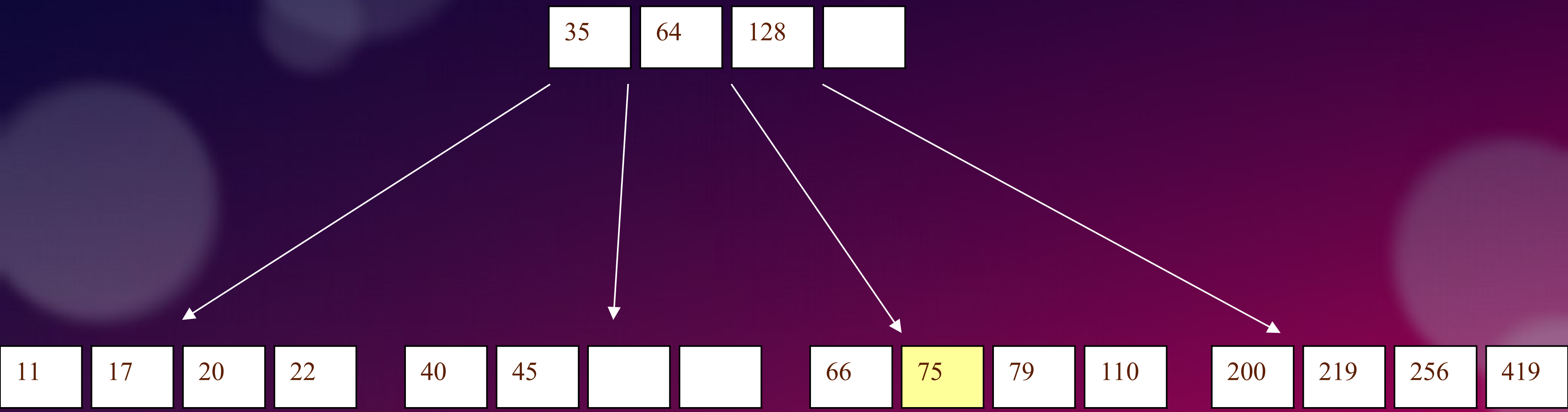
Árvore B



Árvore B



Árvore B



Árvore B (inserção)

- Tente inserir a nova chave em um nó folha (na posição adequada)
- Se isso fizer com que o nó fique cheio, divida a folha em duas partes e suba o elemento central para o nó pai;
- Se isso fizer com que o pai fique cheio repita o processo;
- A estratégia poderá ser repetida até o nó raiz;
- Se necessário o nó raiz deverá ser também divido e o elemento central será transformado em nova raiz (fazendo com que a árvore fique mais alta)

Árvore B - Exercício

- Insira os seguintes números em uma árvore B de ordem 5:
- 3, 7, 9, 23, 45, 1, 5, 14, 25, 24, 13, 11, 8, 19, 4, 31, 35, 56

Árvore B (remoção)

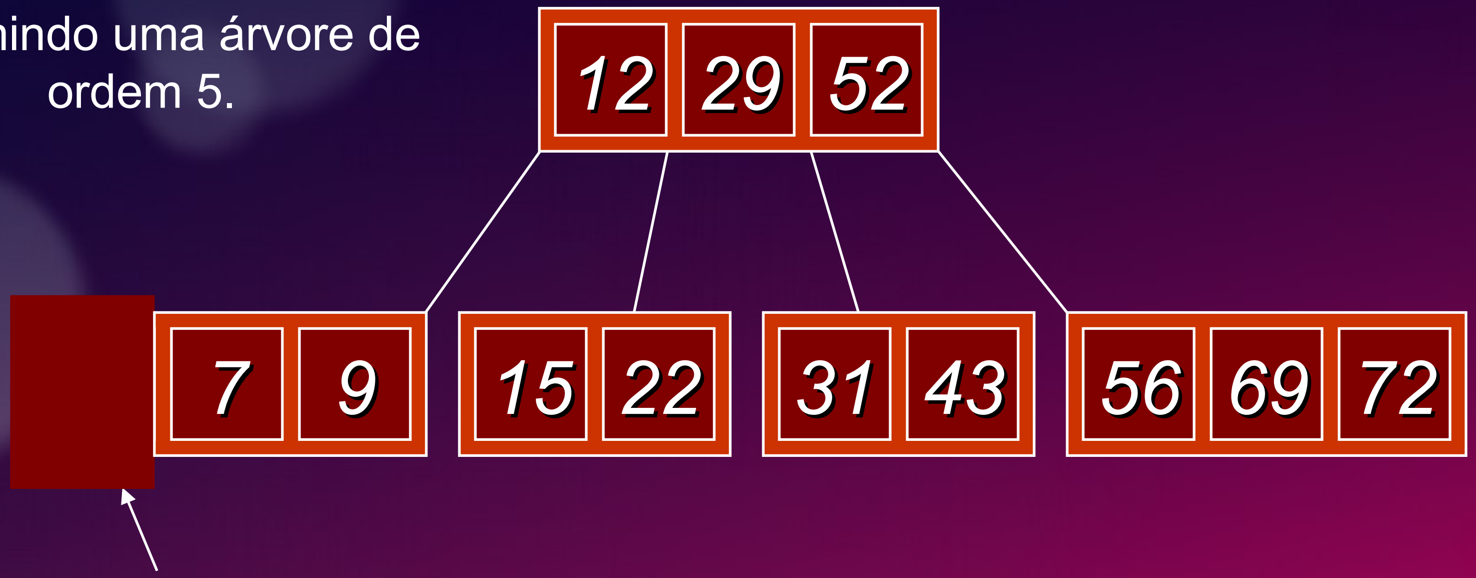
- Durante a inserção, a chave sempre vai para a folha. Na remoção desejamos remover da folha. Assim, temos 3 possibilidades:
- 1 – Se a chave já está em um nó folha e sua remoção não faz com que o nó fique com poucos elementos (menos que $\lceil m / 2 \rceil$ filhos), então apenas elimine-a.
- 2 – Se a chave não é folha, então é garantido que seu predecessor ou sucessor esteja em um nó folha – e neste caso podemos eliminar a chave e subir o predecessor ou sucessor para a posição ocupada pela chave eliminada.

Árvore B (remoção)

- Se (1) ou (2) ocasionam uma folha a ter um número menor que o mínimo então temos que observar os irmãos adjacentes ao nó em questão :
  - 3: Se um deles tem número de chaves maior que o mínimo então pode-se subir uma chave deste nó para o nó pai e pegar a chave do nó pai para a posição da chave eliminada;
  - 4: Se ambos irmãos não têm número de chaves maior que o mínimo, então suas chaves devem ser combinadas com a chave do nó pai. Se este passo fizer com que o nó pai fique com menos chaves que o permitido o processo deve ser repetido até o nó raiz (se necessário).

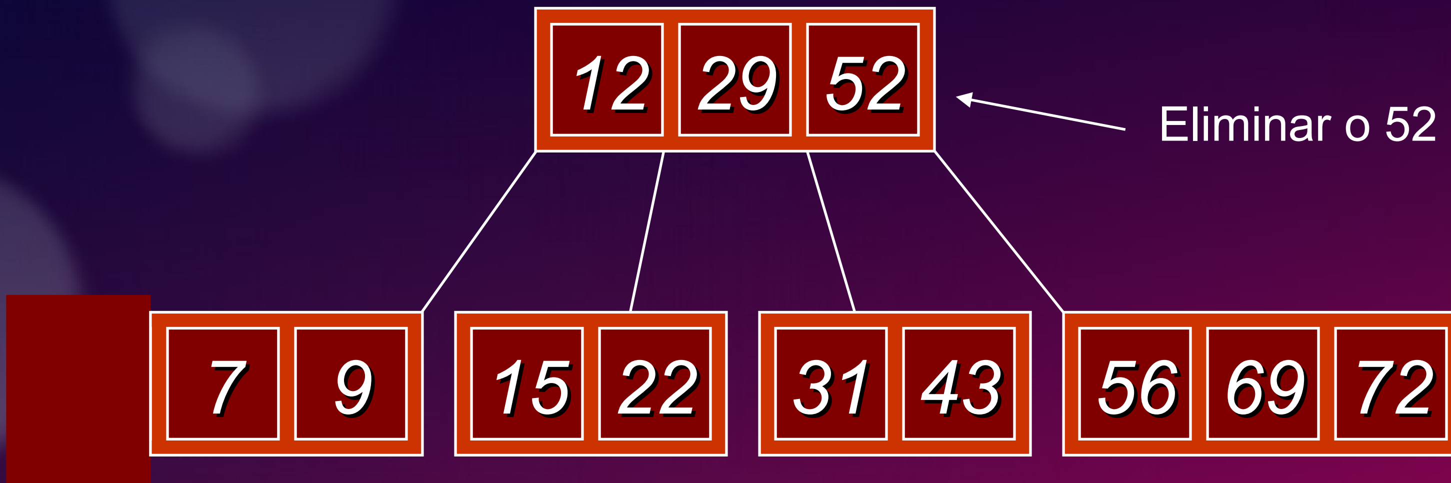
Árvore B – (remoção – caso 1)

Assumindo uma árvore de ordem 5.

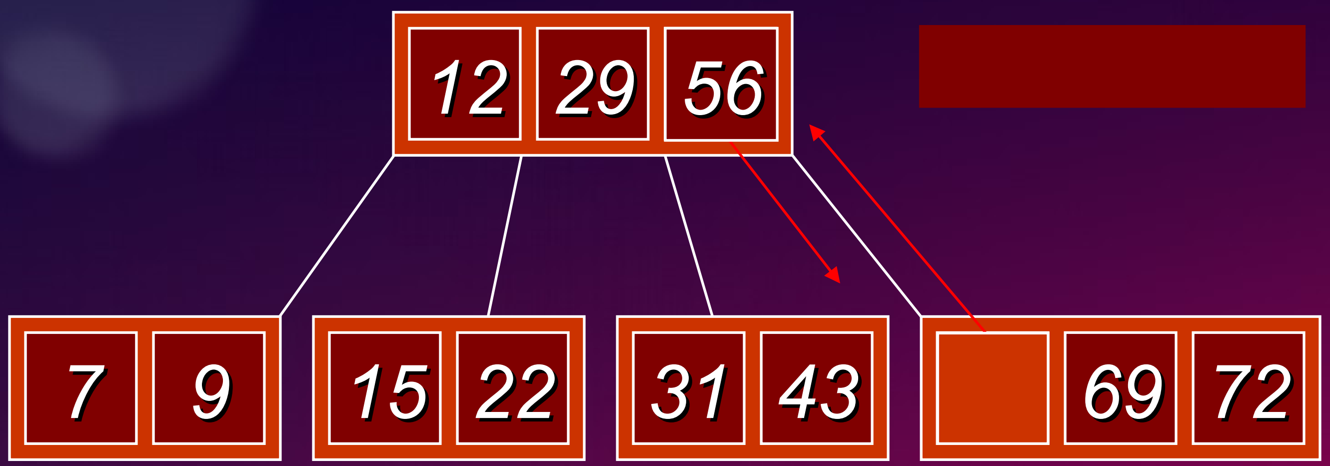


Eliminar o 2: Há chaves suficientes

Árvore B (remoção – caso 1)



Árvore B
(remoção de nó não folha)



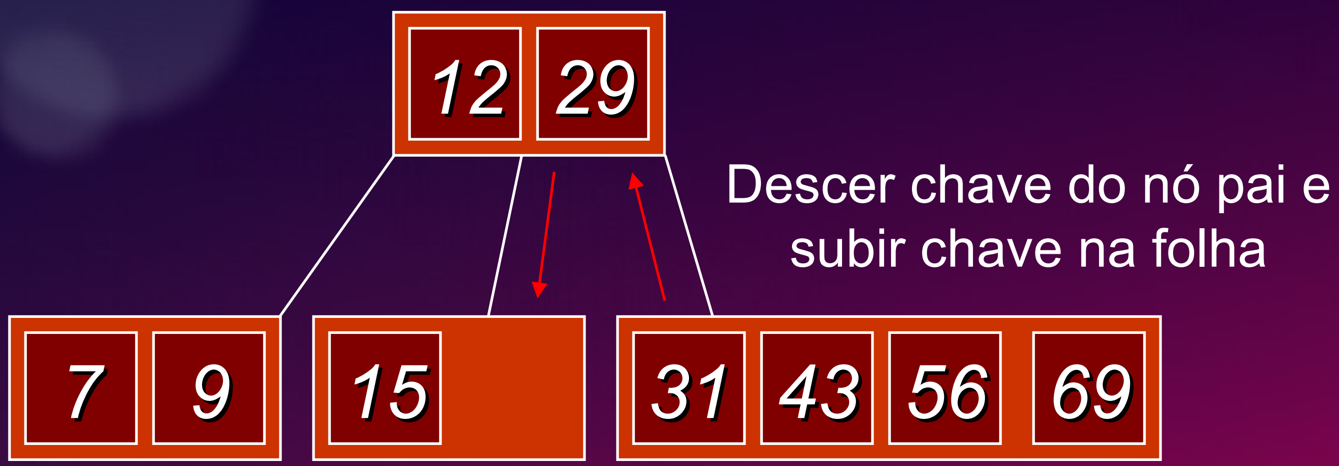
Árvore B
(remoção de nó não folha)

Árvore B
(remoção - Poucas chaves nos nós irmãos)

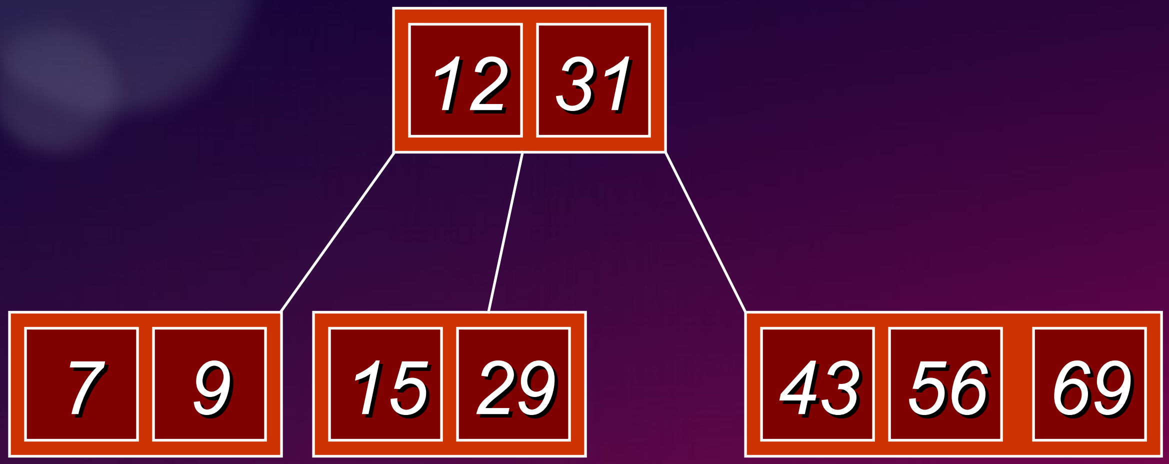
Árvore B
(remoção - Poucas chaves nos nós irmãos)

Eliminar o 22

Árvore B
(remoção - irmão OK)



Árvore B
(remoção - irmão OK)



Animação

http://slady.net/java/bt/view.php?w=600&h=450

Análise de árvore B

- O número máximo de elementos em uma árvore B de ordem *m* e altura *h é*:

  raiz $m - 1$

  nivel 1 $m(m - 1)$

  nivel 2 $m^2(m - 1)$

  . . .

  nivel h $m^h(m - 1)$

- Assim, o total de elementos é

  $(1 + m + m^2 + m^3 + \ldots + m^h)(m - 1) =$

  $[(m^{h+1} - 1)/ (m - 1)]\ (m - 1) = \mathbf{m^{h+1} - 1}$

- Quando $m = 5$ e $h = 2$ temos $5^3 - 1 = 124$

Razões para usar árvores B

- Na busca de dados no disco, o custo de cada acesso é alto (mas não depende muito do tempo de transferência do dado – principalmente se forem consecutivos)
  - Se usarmos uma árvore B de ordem 101 podemos transferir cada nó para a memória primária com um acesso a disco
  - Uma árvore B de ordem 101 e altura 3 pode armazenar (101^4 – 1) chaves (aproximadamente 100 milhões) e qualquer elemento pode ser acessado com no máximo 3 operações de leitura (assumindo que a raiz permanece na memória)
- Se tomarmos *m* = 3, temos uma árvore **2-3**, na qual um nó não folha tem 2 ou 3 filhos