MC202 - Estrutura de Dados

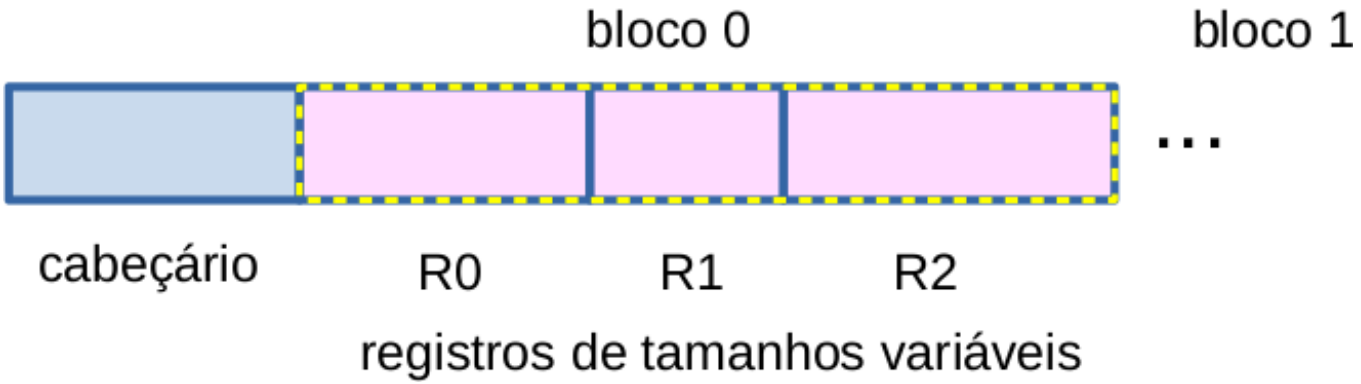
Alexandre Xavier Falcão

Instituto de Computação - UNICAMP

afalcao@ic.unicamp.br

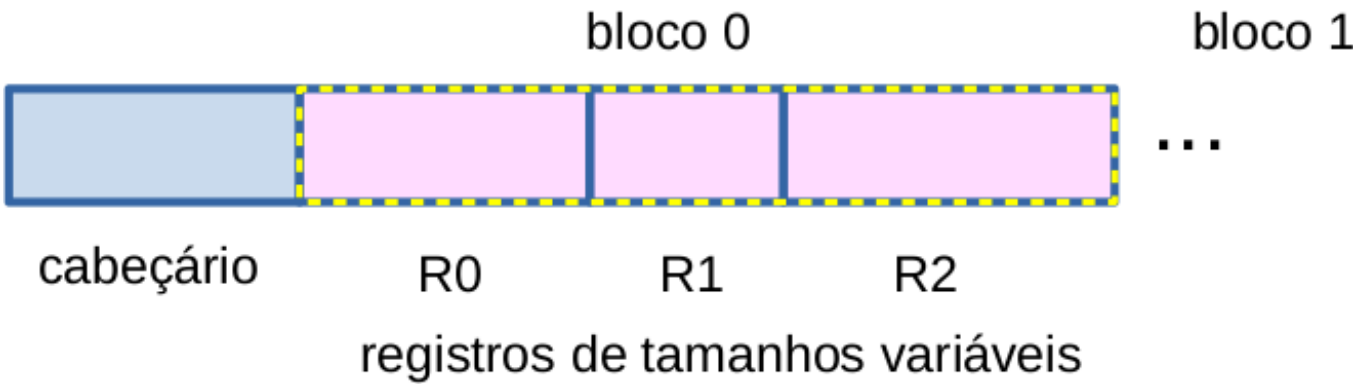
Introdução à Árvore B

Os registros de um arquivo binário com terabytes são armazenados sequencialmente em blocos do disco rígido (memória externa).



Introdução à Árvore B

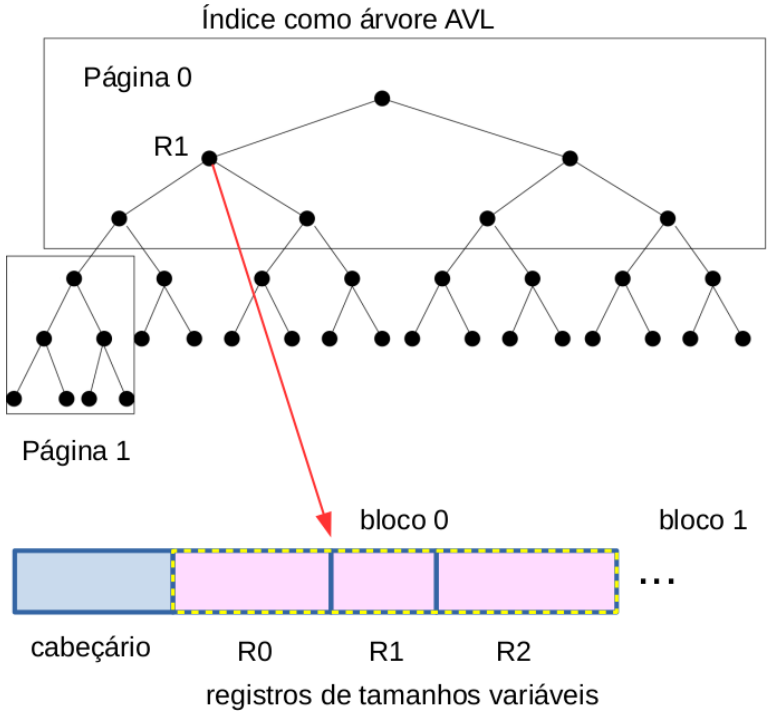
Os registros de um arquivo binário com terabytes são armazenados sequencialmente em blocos do disco rígido (memória externa).



O acesso a esses registros é muito mais lento do que na memória interna (milisegundos versus nanosegundos).

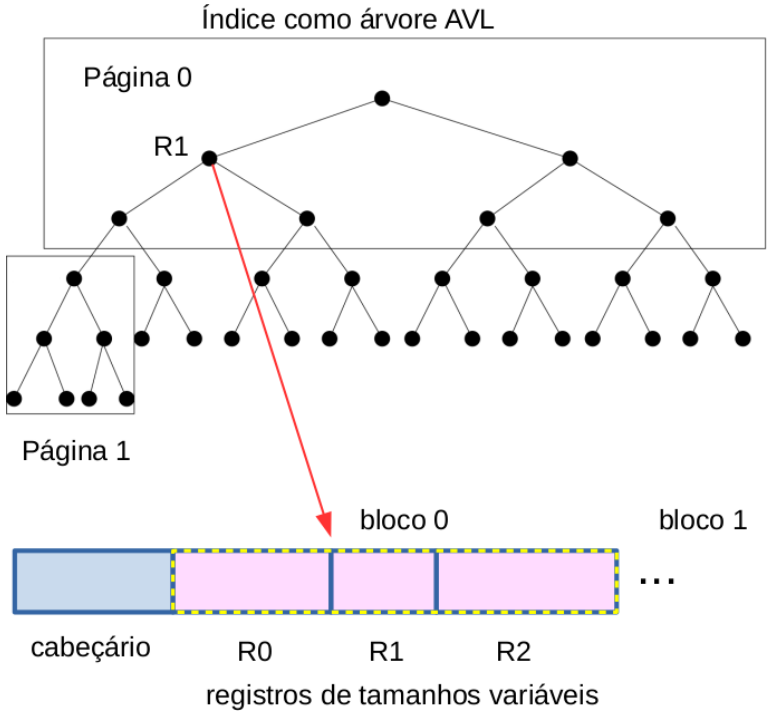
Introdução à Árvore B

Uma árvore de busca, cujos nós armazenam uma chave de **identificação única** dos registros e o deslocamento correspondente em bytes para acessá-los em disco, é denominada índice primário.



Introdução à Árvore B

Uma árvore de busca, cujos nós armazenam uma chave de **identificação única** dos registros e o deslocamento correspondente em bytes para acessá-los em disco, é denominada índice primário.



Para agilizar o acesso, o índice (e.g., armazenado no cabeçário) é carregado em memória interna por páginas (blocos) para realizar a busca por registros.

Introdução à Árvore B

Uma árvore B é uma árvore de busca de altura balanceada que permite armazenar $b > 1$ pares (chave, deslocamento) por nó. Para $b = 255$, por exemplo, é possível armazenar dados de $N = 4.27 \times 10^9$ registros com apenas $\log_b N$ níveis.

	Mínimo		Máximo	
Nível	Nós	Registros	Nós	Registros
1	1	1	1	255
2	2	2x127	256	256x255
3	2x128	2x128x127	256x256	256^2x255
4	32.768	4.161.536	16.777.216	4.278.190.080

Introdução à Árvore B

Uma árvore B é uma árvore de busca de altura balanceada que permite armazenar $b > 1$ pares (chave, deslocamento) por nó. Para $b = 255$, por exemplo, é possível armazenar dados de $N = 4.27 \times 10^9$ registros com apenas $\log_b N$ níveis.

	Mínimo		Máximo	
Nível	Nós	Registros	Nós	Registros
1	1	1	1	255
2	2	2x127	256	256x255
3	2x128	2x128x127	256x256	256^2x255
4	32.768	4.161.536	16.777.216	4.278.190.080

Também é possível construir índices secundários que armazenam pares (chave secundária, chave primária), formando uma lista invertida.

Agenda

- Definição formal de árvore B.

Agenda

- Definição formal de árvore B.
- Busca por uma chave (registro).

Agenda

- Definição formal de árvore B.
- Busca por uma chave (registro).
- Inserção.

Agenda

- Definição formal de árvore B.
- Busca por uma chave (registro).
- Inserção.
- Remoção.

Árvore B

Uma árvore B de ordem $b > 1$ é uma árvore de busca que satisfaz três condições adicionais.

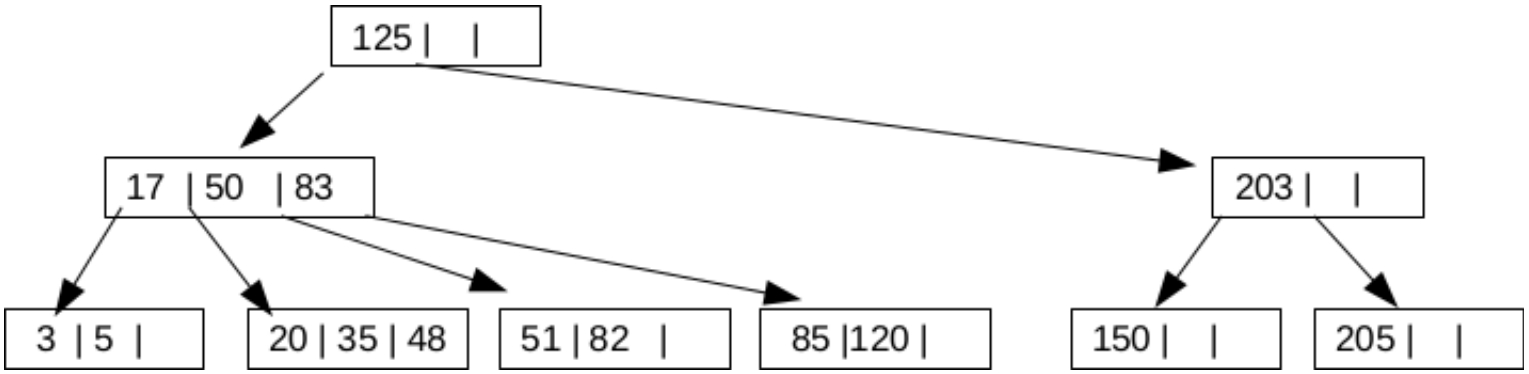
1. Todas as folhas têm o mesmo nível.

2. Cada nó interno armazena (chave,deslocamento) de um número variável r de registros e $r + 1$ filhos, onde
   a. $1 \leq r \leq b$ se o nó for a raiz da árvore.
   b. $\lfloor \frac{b}{2} \rfloor \leq r \leq b$ se o nó não for a raiz da árvore.

3. Cada nó folha tem um número variável r de registros satisfazendo às condições a e b do item 2.

Exemplo

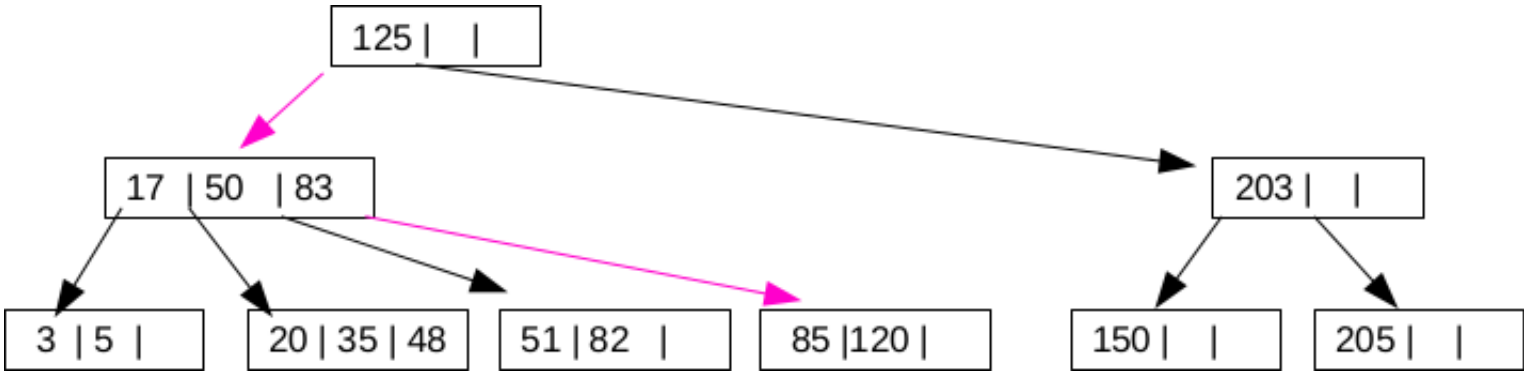
Uma árvore B de ordem $b = 3$ (vamos omitir os deslocamentos).



As chaves são mantidas em ordem no nó para agilizar à busca.

Busca de uma chave

Com $\log_b N$ acessos aos nós é possível encontrar qualquer uma entre N chaves.



A chave 85 pode ser encontrada com até 3 acessos (buscas binárias em nós acessados), mesmo que a árvore de altura 2 estivesse cheia com 33 chaves.

Inserção

Para inserir uma nova chave x,

- as inserções são sempre feitas em nó folha na volta da recursão,
- uma inserção pode gerar ou não *overflow* ($r > b$), e
- o *overflow* é tratado após a inserção com a divisão do nó seguida de inserção do registro da chave mediana no nó pai.

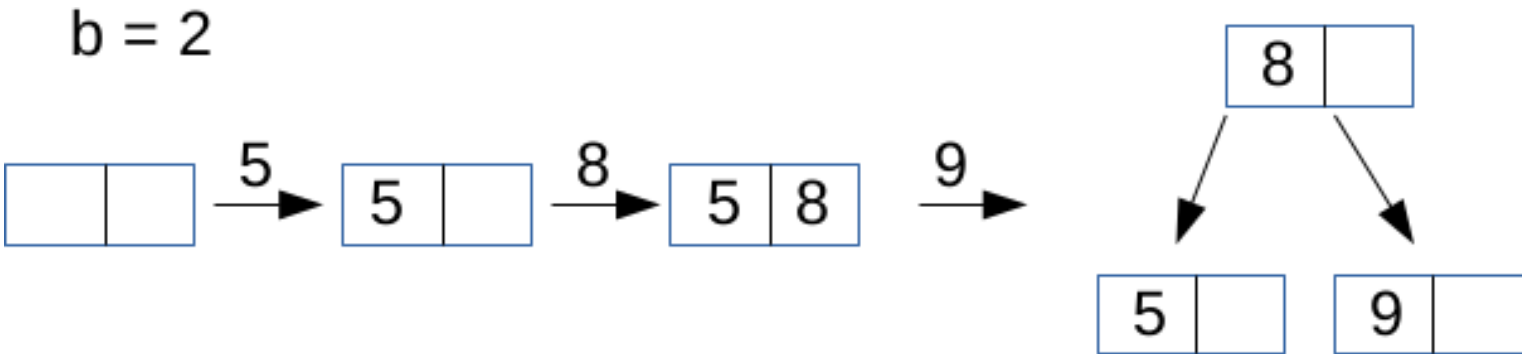
Inserção

Para inserir uma nova chave x,

- as inserções são sempre feitas em nó folha na volta da recursão,
- uma inserção pode gerar ou não *overflow* ($r > b$), e
- o *overflow* é tratado após a inserção com a divisão do nó seguida de inserção do registro da chave mediana no nó pai.

O nó criado com a divisão passa a ser filho à direita da chave mediana no nó pai e o *overflow* pode se propagar até aumentar a altura da árvore.

Inserção



Inserções sucessivas em uma árvore B de ordem $b = 2$ (também chamada árvore 2-3).

Inserção

A inserção de uma chave x busca de forma recursiva o nó folha e a posição onde será inserida a chave nele. Podem ocorrer dois casos:

1. A folha acomoda o registro sem gerar *overflow*. Após a inserção indica-se *overflow*= *false*.

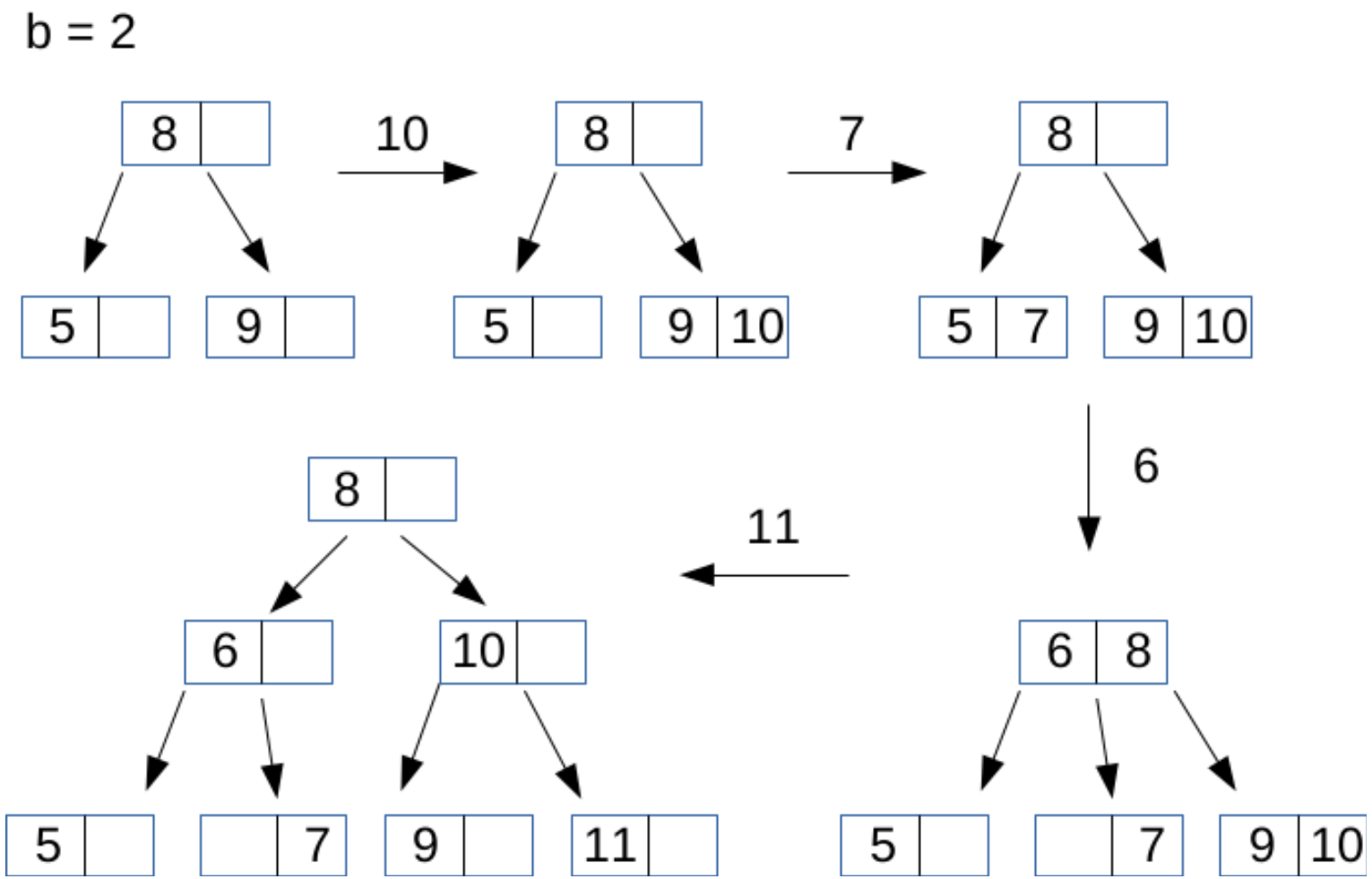
Inserção

A inserção de uma chave x busca de forma recursiva o nó folha e a posição onde será inserida a chave nele. Podem ocorrer dois casos:

1. A folha acomoda o registro sem gerar *overflow*. Após a inserção indica-se *overflow*= *false*.

2. A folha é dividida em dois nós, passando os registros acima da chave mediana para o novo nó, e retornando o registro com chave mediana para inserção no nó pai junto com o novo nó, que será o filho à direita da mediana no nó pai. Indica-se *overflow*= *true* para a inserção no nó pai.

Inserção



Inserções sucessivas em uma árvore B de ordem $b = 2$, ilustrando os casos 1 e 2.

Remoção

Para remover uma chave x,

- as remoções são sempre feitas em nó folha na volta da recursão. Portanto, se a chave não estiver em uma folha, ela deverá ser primeiro trocada com sua antecessora.

Remoção

Para remover uma chave x,

- as remoções são sempre feitas em nó folha na volta da recursão. Portanto, se a chave não estiver em uma folha, ela deverá ser primeiro trocada com sua antecessora.

- A busca pela chave na folha continua e a remoção na folha pode gerar ou não *underflow* ($r < \frac{b}{2}$).

Remoção

Para remover uma chave x,

- as remoções são sempre feitas em nó folha na volta da recursão. Portanto, se a chave não estiver em uma folha, ela deverá ser primeiro trocada com sua antecessora.
- A busca pela chave na folha continua e a remoção na folha pode gerar ou não *underflow* ($r < \frac{b}{2}$).
- O *underflow*= *true* é tratado após a remoção e

Remoção

Para remover uma chave x,

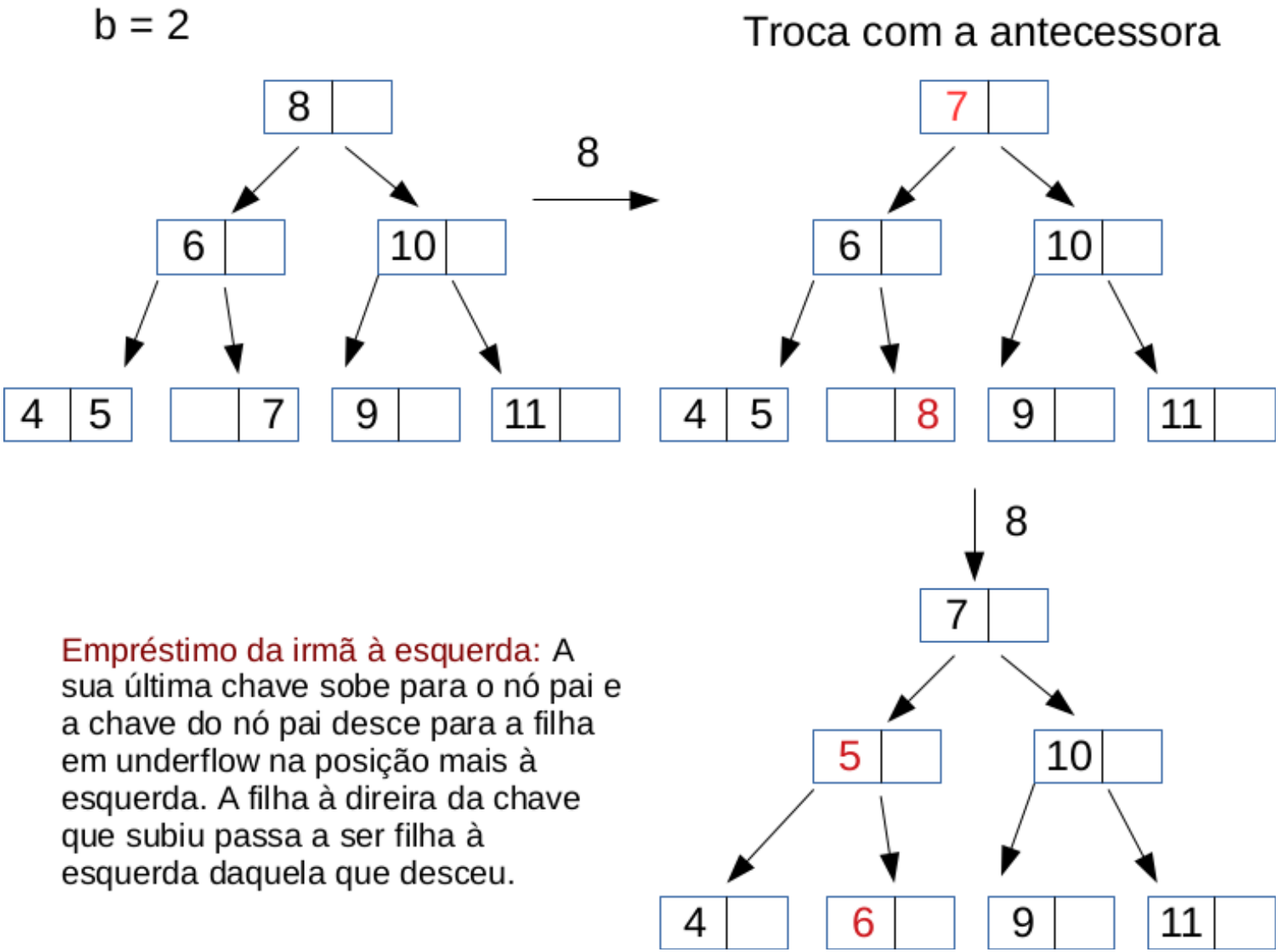
- as remoções são sempre feitas em nó folha na volta da recursão. Portanto, se a chave não estiver em uma folha, ela deverá ser primeiro trocada com sua antecessora.
- A busca pela chave na folha continua e a remoção na folha pode gerar ou não *underflow* ($r < \frac{b}{2}$).
- O *underflow*= *true* é tratado após a remoção e
  - pode ser que uma das irmãs (à esquerda ou à direita) tenha chave para emprestar ou que

Remoção

Para remover uma chave x,

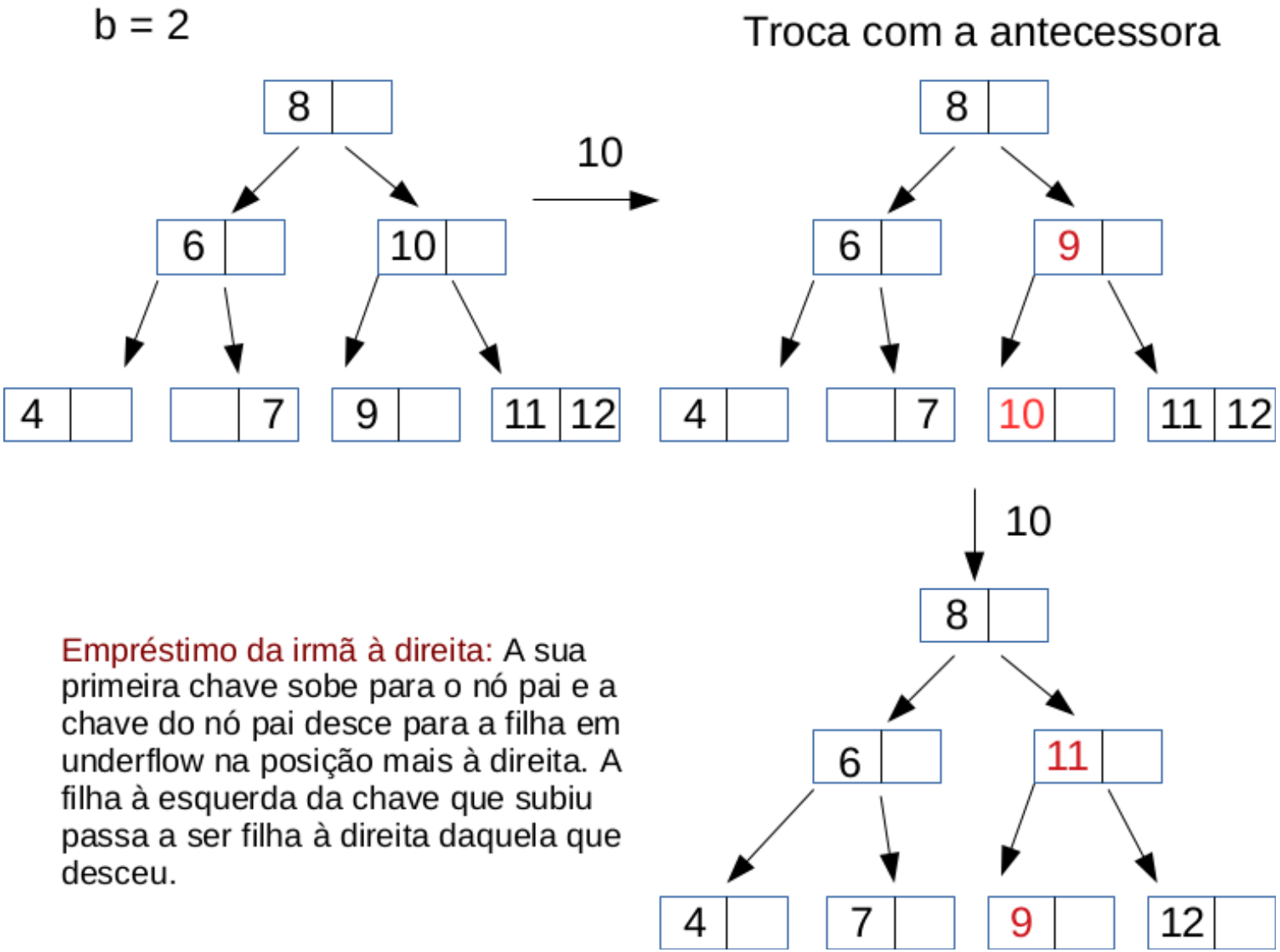
- as remoções são sempre feitas em nó folha na volta da recursão. Portanto, se a chave não estiver em uma folha, ela deverá ser primeiro trocada com sua antecessora.
- A busca pela chave na folha continua e a remoção na folha pode gerar ou não *underflow* ($r < \frac{b}{2}$).
- O *underflow*$=$ *true* é tratado após a remoção e
  - pode ser que uma das irmãs (à esquerda ou à direita) tenha chave para emprestar ou que
  - o empréstimo não seja possível, ocasionando a propagação do *underflow*$=$ *true* e podendo reduzir a altura da árvore.

Remoção



Empréstimo da irmã à esquerda: A sua última chave sobe para o nó pai e a chave do nó pai desce para a filha em underflow na posição mais à esquerda. A filha à direira da chave que subiu passa a ser filha à esquerda daquela que desceu.

Remoção



Empréstimo da irmã à direita: A sua primeira chave sobe para o nó pai e a chave do nó pai desce para a filha em underflow na posição mais à direita. A filha à esquerda da chave que subiu passa a ser filha à direita daquela que desceu.

Remoção

Quando as irmãs não possuem chaves para emprestar,

- A filha em *underflow*= *true* une-se com uma das irmãs,
- a chave do nó pai desce para o nó da união,
- sendo removida do nó pai.

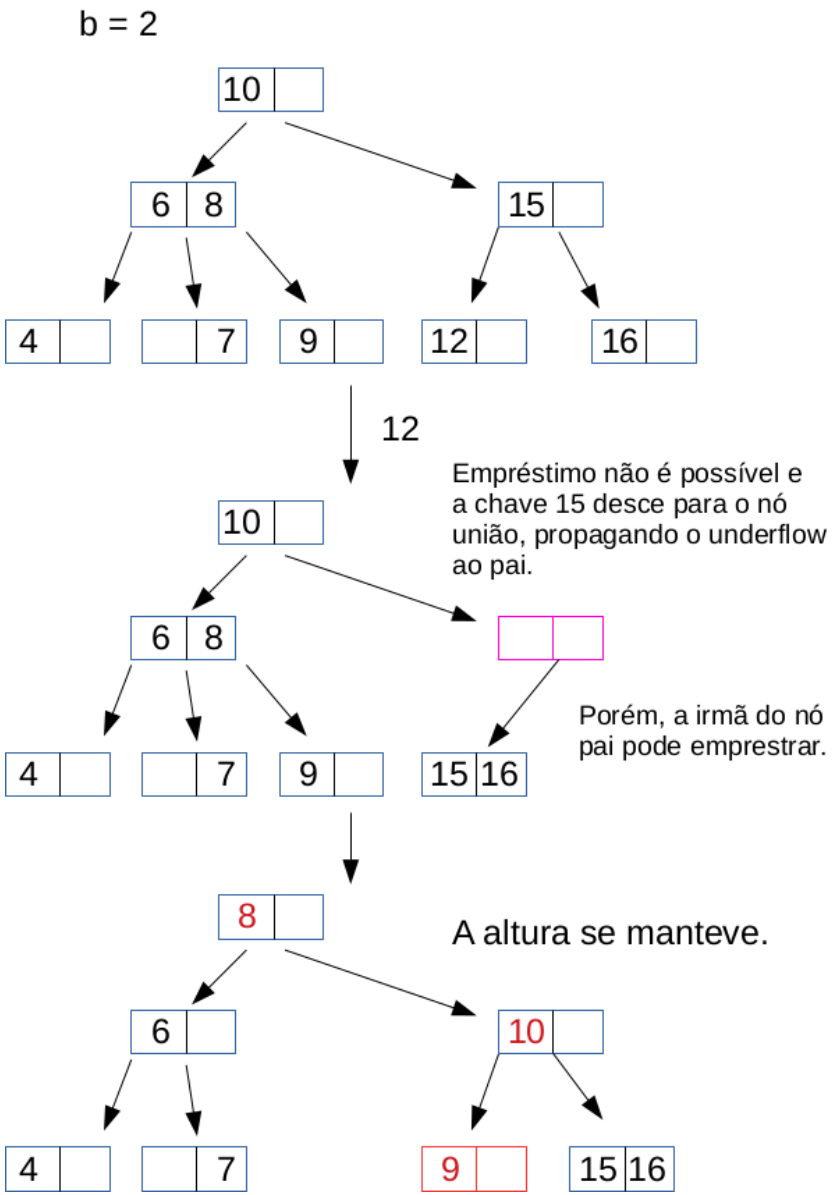
Remoção

Quando as irmãs não possuem chaves para emprestar,

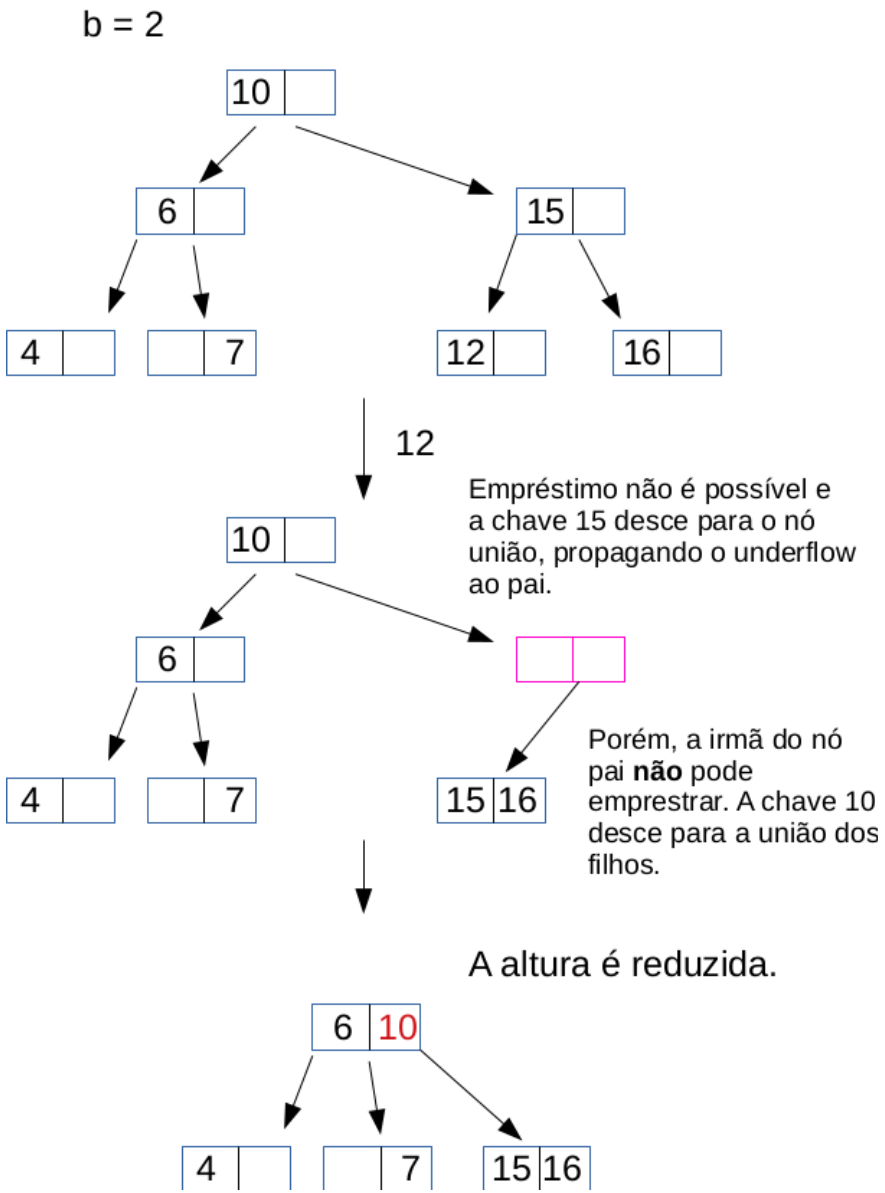
- A filha em *underflow*= *true* une-se com uma das irmãs,
- a chave do nó pai desce para o nó da união,
- sendo removida do nó pai.

A remoção da chave no nó pai pode ocasionar um segundo *underflow*, propagando o problema acima e podendo reduzir a altura da árvore.

Remoção


b = 2
10
6 8
15
4
7
9
12
16
12
Empréstimo não é possível e a chave 15 desce para o nó união, propagando o underflow ao pai.
10
6 8
Porém, a irmã do nó pai pode emprestrar.
4
7
9
15 16
8
A altura se manteve.
6
10
4
7
9
15 16

Remoção


b = 2
10
6
15
4
7
12
16
12
Empréstimo não é possível e a chave 15 desce para o nó união, propagando o underflow ao pai.
10
6
4
7
15 16
Porém, a irmã do nó pai **não** pode emprestrar. A chave 10 desce para a união dos filhos.
A altura é reduzida.
6 10
4
7
15 16

Remoção

Em resumo, a remoção pode ser reduzida a três casos.

1. A folha acomoda a remoção sem gerar *underflow* (i.e., *underflow*= *false* após remoção).

Remoção

Em resumo, a remoção pode ser reduzida a três casos.

1. A folha acomoda a remoção sem gerar *underflow* (i.e., *underflow*= *false* após remoção).

2. A remoção gera *underflow* (i.e., *underflow*= *true*), mas um dos irmãos tem para emprestar.

Remoção

Em resumo, a remoção pode ser reduzida a três casos.

1. A folha acomoda a remoção sem gerar *underflow* (i.e., *underflow*= *false* após remoção).

2. A remoção gera *underflow* (i.e., *underflow*= *true*), mas um dos irmãos tem para emprestar.

3. A remoção gera *underflow*, não há possibilidade de empréstimo, gerando uma união de irmãos com inserção da chave que é removida do nó pai, o que propaga o *underflow* para cima e pode causar redução na altura da árvore.

Remoção

Em resumo, a remoção pode ser reduzida a três casos.

1. A folha acomoda a remoção sem gerar *underflow* (i.e., *underflow*= *false* após remoção).

2. A remoção gera *underflow* (i.e., *underflow*= *true*), mas um dos irmãos tem para emprestar.

3. A remoção gera *underflow*, não há possibilidade de empréstimo, gerando uma união de irmãos com inserção da chave que é removida do nó pai, o que propaga o *underflow* para cima e pode causar redução na altura da árvore.

Vamos ver como fica o código...