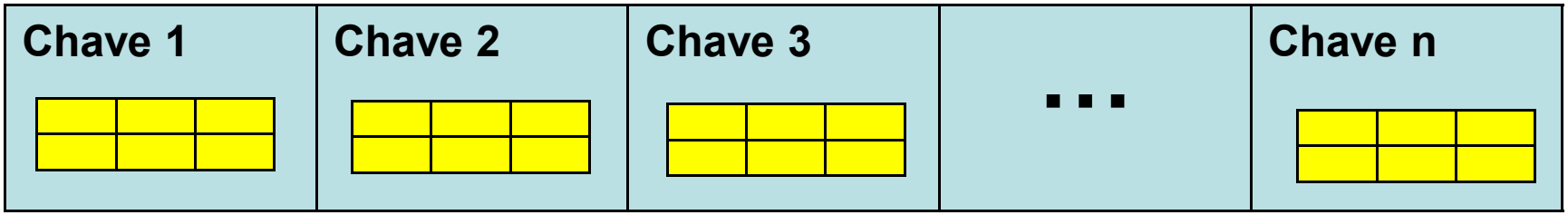
Armazenamento em disco com uso de índices

Vimos até o momento árvores onde a chave aparenta ser o único objeto de interesse. Contudo, como sugere a representação de um nó, constante no slide 82, a chave será apenas um campo de um registro. Neste casos, o vetor de chaves é um vetor de registros, cada um tendo um campo de identificador único e demais informações pertinentes para aplicação, conforme imagem abaixo.



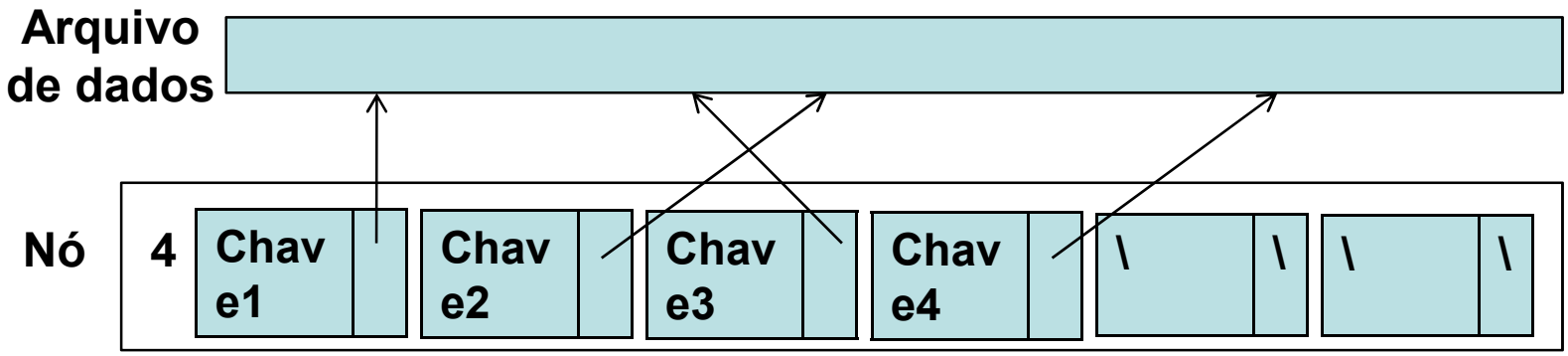
Armazenamento em disco com uso de índices

Contudo, apenas a chave é relevante no processo de busca de um registro, o que torna os demais itens apenas áreas subutilizadas no nó o que reflete-se em mais operações de transferências de dados entre memória principal e secundária durante o processo de busca.

Desta forma, uma abordagem adequada seria associar apenas mais uma informação à chave que representaria um endereço de um registro inteiro na armazenagem secundária, como ilustra a figura do slide a seguir.

UNIVASF

Armazenamento em disco com uso de índices



Se o conteúdo de tal nó também residisse no armazenamento secundário, cada acesso de chave exigiria dois acessos do armazenamento secundário. No longo prazo, isso é melhor do que manter os registros inteiros nos nós. Pois, armazenando o registro inteiro, os nós podem manter um número muito pequeno de tais registros. A árvore B resultante é muito profunda e os caminhos de busca através dela são muito mais longos do que em uma árvore B com os endereços de registros.

Árvores B+

Já que um nó de uma árvore B representa uma página de memória secundária ou um bloco, a passagem de um nó para um outro exige uma mudança demorada de página. Em consequência, gostaríamos de realizar a menor quantidade de acessos possíveis.

O que acontece se solicitamos que todas as chaves na árvore B sejam impressas na ordem ascendente?

Como vimos anteriormente, o percurso de árvore in-ordem pode ser usado, que é fácil de implementar, mas, para nós não terminais, somente uma chave é mostrada de cada vez e então outra página tem que ser acessada.

UNIVASF

Árvores B+

Desta forma, teríamos de melhorar as árvores B para nos permitir acessar os dados sequencialmente, de uma maneira mais rápida que o percurso em in-ordem. Uma árvore B+ oferece uma solução (Wedeking, 1974).

Em uma árvore B referências aos dados são feitas a partir de quaisquer nós da árvore, mas em uma árvore B+ essas referências são feitas somente a partir das folhas. Os nós internos de uma árvore B+ são indexados para acesso rápido dos dados; essa parte da árvore é chamada de conjunto de índices.

UNIVASF

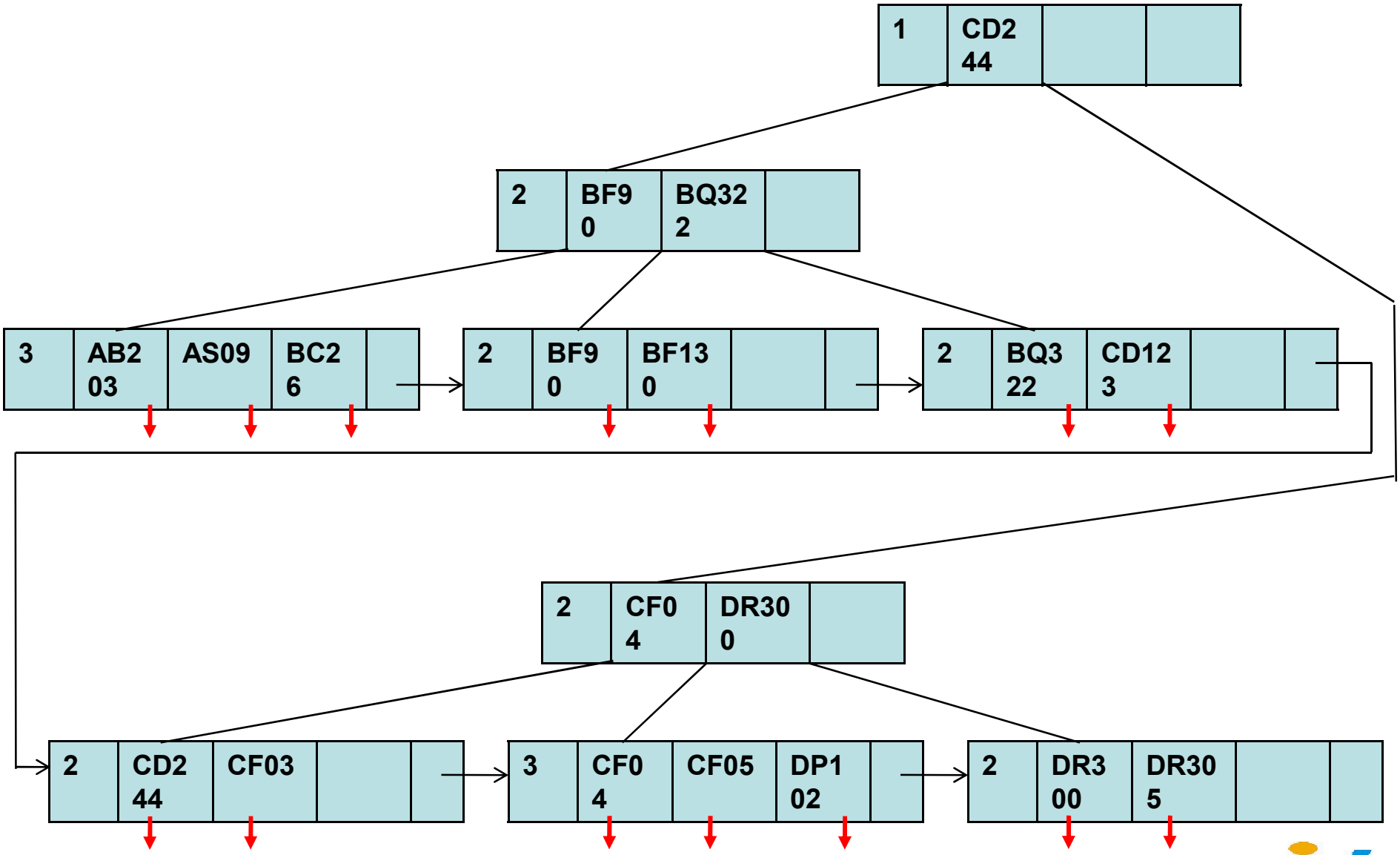
Árvores B+

As folhas têm uma estrutura diferente que a dos outros nós da árvore B+, e usualmente são vinculadas sequencialmente para formar um **conjunto de sequências**, de modo que varrer esta lista de folhas resulta nos dados obtidos na ordem ascendente.

Por isso, uma árvore B+ é verdadeiramente uma árvore B mais: ela é um índice implementado como uma árvore B regular mais uma lista ligada de dados. O slide a seguir contém um exemplo de uma árvore B+.

Note que os nós internos armazenam chaves, ponteiros e um contador de chaves. As folhas armazenam chaves, referências aos registros em um arquivo de dados associados com as chaves (representados por uma seta vermelha), um ponteiro para a próxima folha e um contador de chaves.

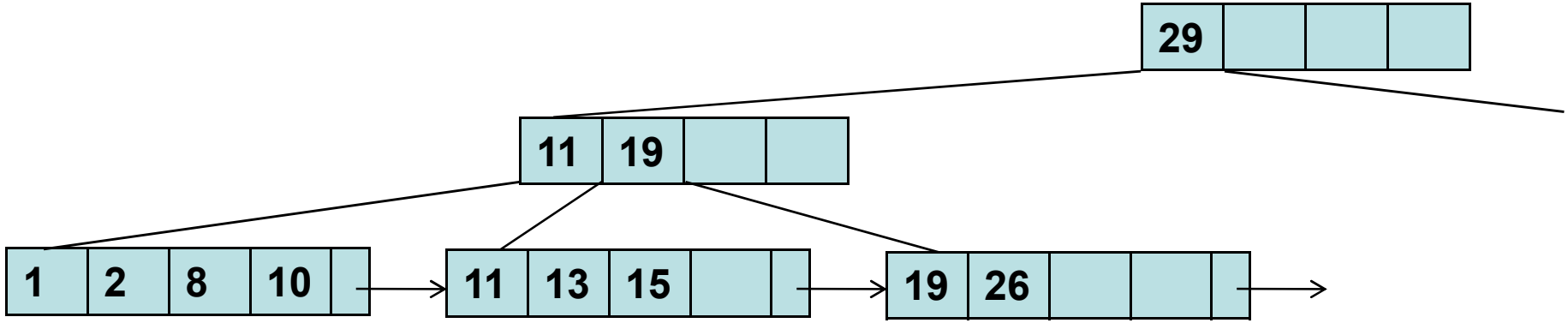
Exemplo de árvores B+



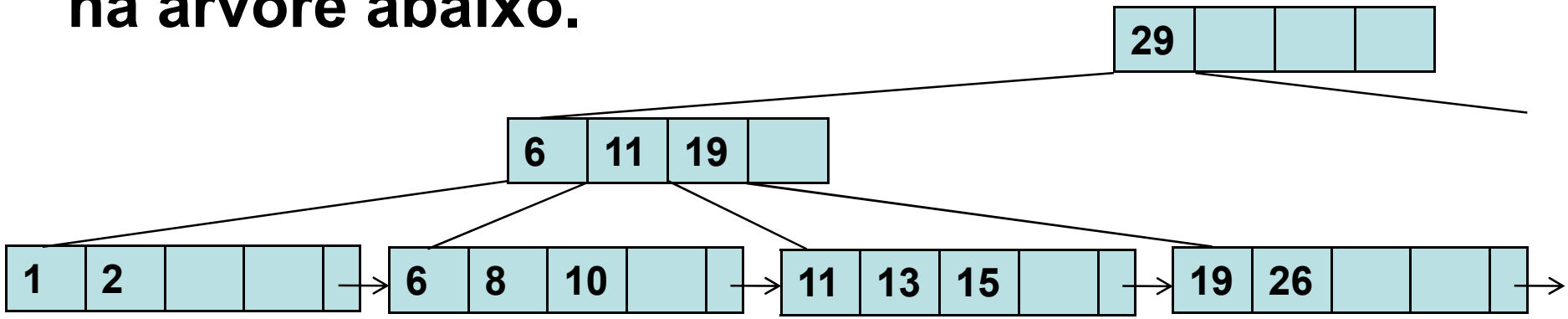
Árvores B+

As operações nas árvores B+ não são muito diferentes das operações das árvores B.

Inserir uma chave em uma folha que ainda tenha algum espaço exige que se coloque as chaves desta folha em ordem. Nenhuma mudança é feita no conjunto de índices. Se uma chave é inserida em uma folha cheia, a folha é dividida, o novo nó folha é incluído no conjunto de sequência, as chaves são distribuídas homogeneamente entre a folha velha e a nova folha e a primeira chave do novo nó é copiada (**não** movida, como na árvore B) para o ascendente. Veja o slide a seguir.



A inserção da chave 6 da árvore acima resulta na árvore abaixo.



Se o ascendente não está cheio, isso pode exigir reorganização local das chaves do ascendente como no caso acima.

Se o ascendente está cheio, o processo de divisão é realizado do mesmo modo que nas árvore B. Depois de tudo, o conjunto de índices é uma árvore B.

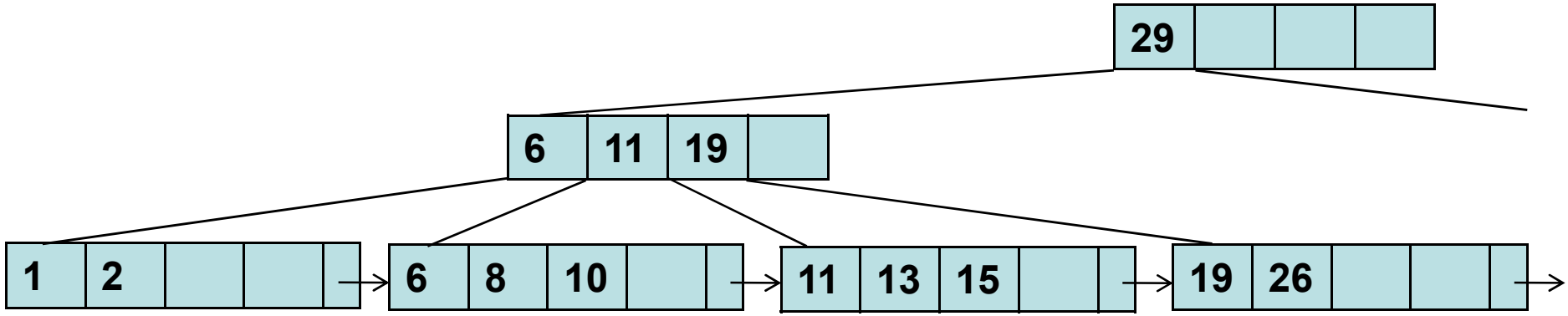
Árvores B+

Remover uma chave de uma folha sem subaproveitamento exige colocar as chaves remanescentes em ordem, nenhuma mudança é feita ao conjunto de índices. Em particular, se uma chave que não ocorre somente em uma folha é removida, ela é simplesmente removida da folha, mas pode permanecer no nó interno.

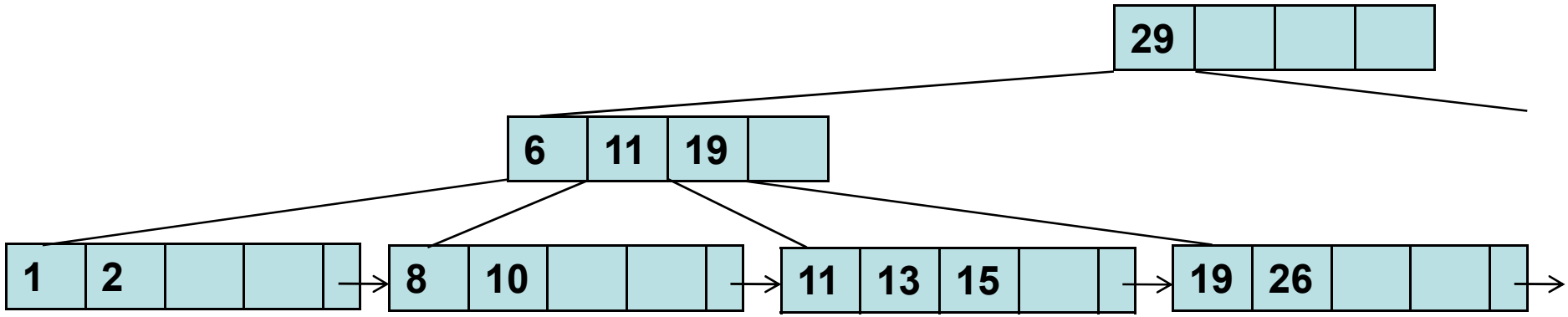
A razão é que ele ainda serve como guia apropriado quando se navega para baixo na árvore B+, porque ainda separa apropriadamente as chaves entre dois filhos adjacentes mesmo se o próprio separador não ocorre em nenhum dos filhos. Observe o processo no próximo slide.

UNIVASF

Árvores B+

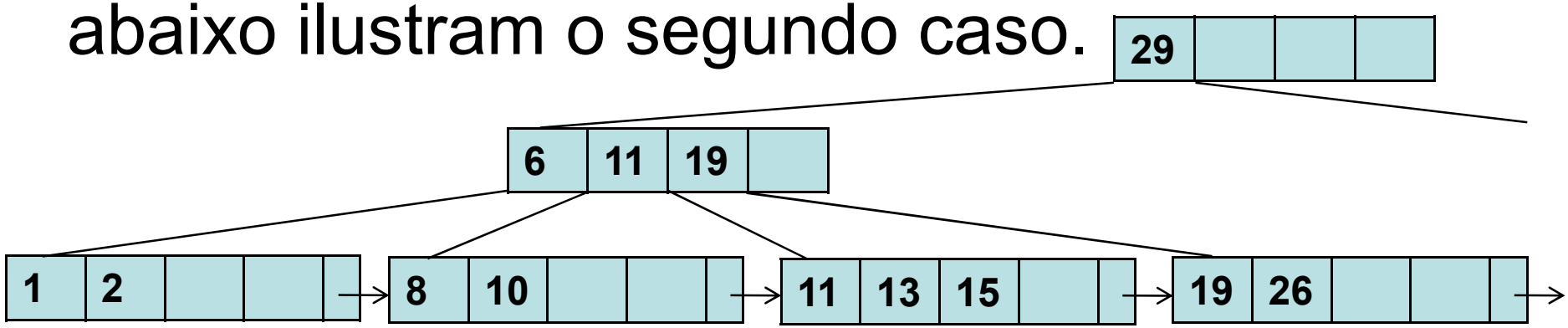


A remoção da chave 6 da árvore acima resulta na árvore abaixo. Note que o número 6 não é removido de um nó interno.

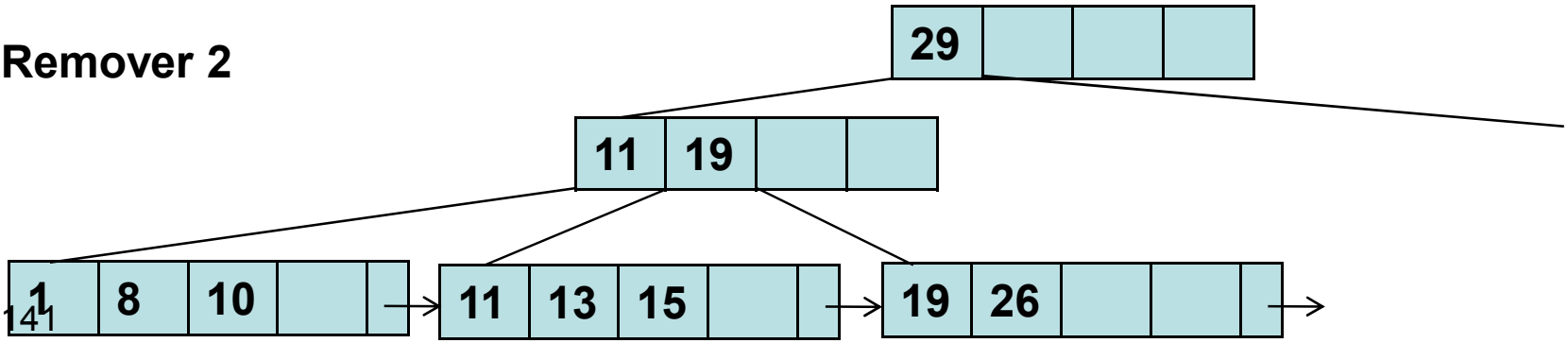


Árvores B+

Quando a remoção de uma chave causa subaproveitamento, ambas as chaves dessa folha e as chaves de um irmão são redistribuídas entre a folha e seu irmão, ou a folha é removida e as chaves remanescentes são incluídas no seu irmão. As árvores abaixo ilustram o segundo caso.



Remover 2



Árvores B+

Como vimos, depois de remover o número 2, um subaproveitamento ocorre e duas folhas são combinadas para formar uma folha. A primeira chave do irmão à direita do nó que permanece depois da fusão é copiada ao nó ascendente, e as chaves no ascendente são colocadas em ordem. Essas duas operações exigem atualização do separador no ascendente. Além disso, remover uma folha pode disparar fusões no conjunto de índices.

UNIVASF