



UNIVERSIDADE  
FEDERAL DE  
SERGIPE



DEPARTAMENTO  
DE COMPUTAÇÃO

# Árvore B+

## Estruturas de Dados

Bruno Prado

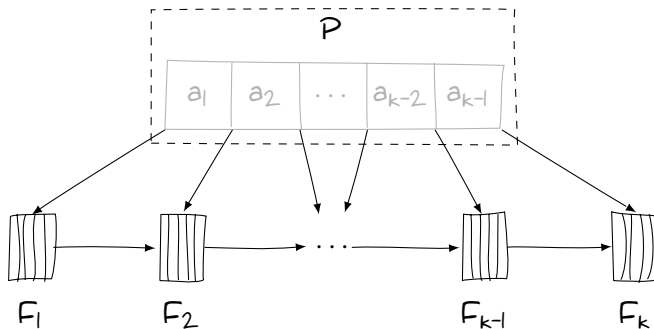
Departamento de Computação / UFS

# Introdução

- ▶ O que é uma árvore B+?
  - ▶ É uma variação da árvore B
  - ▶ Os nós internos armazenam somente as chaves e referências para outros nós (indexação)
  - ▶ Uma lista encadeada é construída com os nós folha que armazenam as referências para os dados

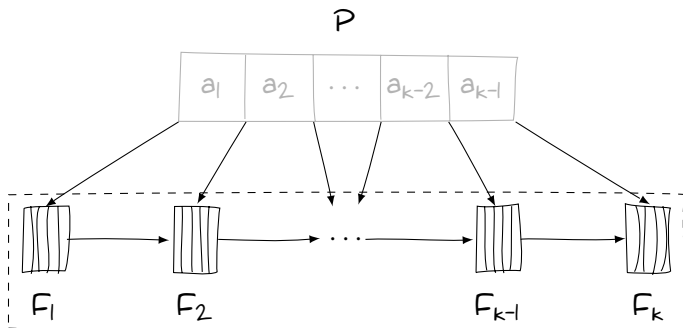
# Introdução

- ▶ Árvore B+ de ordem  $k$ 
  - ▶ Nó interno



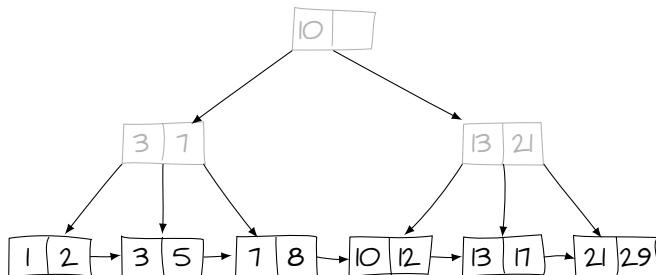
# Introdução

- ▶ Árvore B+ de ordem  $k$ 
  - ▶ Nós folha



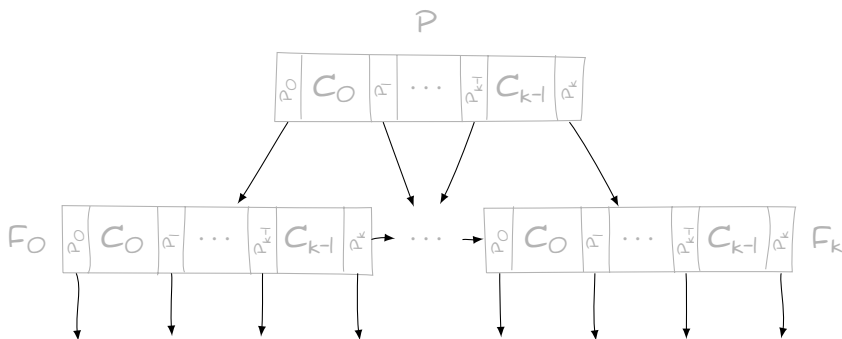
# Introdução

## ► Árvore B+ de ordem 3



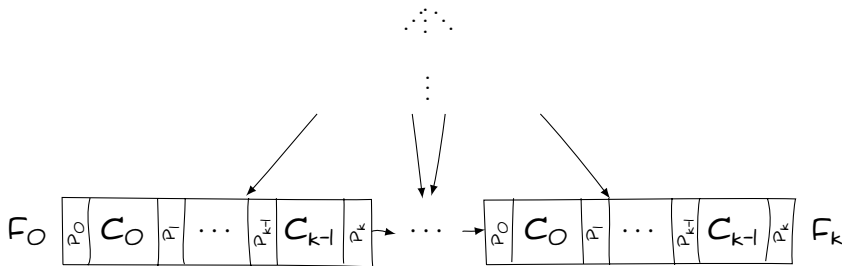
# Árvore B+

- ▶ Definição da estrutura
  - ▶ Nós internos



# Árvore B+

- ▶ Definição da estrutura
  - ▶ Nó folha



# Árvore B+

- ▶ Implementação em C
  - ▶ Estrutura e ponteiros

```
1 // Padrão de tipos por tamanho
2 #include <stdint.h>
3 // Estrutura de nó
4 typedef struct no {
5     // Vetor de chaves
6     uint32_t* C;
7     // Vetor de filhos
8     struct no** P;
9     // Quantidade utilizada
10    uint32_t n;
11 } no;
```

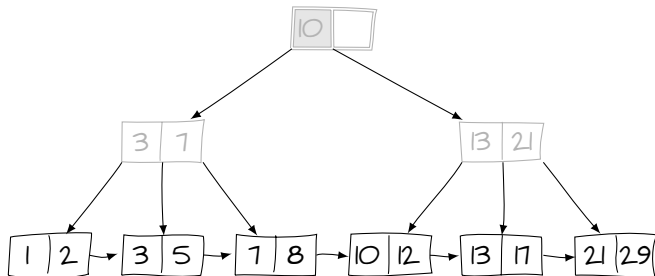


# Árvore B+

- ▶ Operações básicas
  - ▶ Busca
    - ▶ Exata
    - ▶ Intervalo
  - ▶ Inserção
  - ▶ Remoção

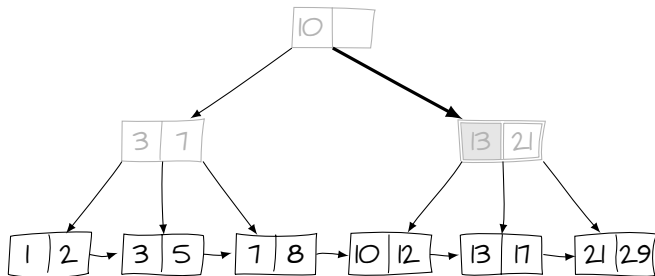
# Árvore B+

- ▶ Operação de busca exata
  - ▶ Parâmetro de chave: 13
  - ▶ A busca tem início pela raiz da árvore, checando as chaves do nó e acessando as subárvores



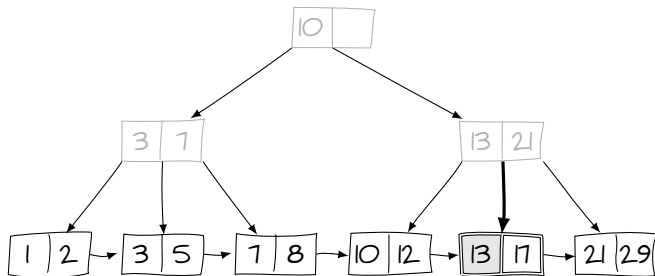
# Árvore B+

- ▶ Operação de busca exata
  - ▶ Parâmetro de chave: 13
  - ▶ A busca tem início pela raiz da árvore, checando as chaves do nó e acessando as subárvores



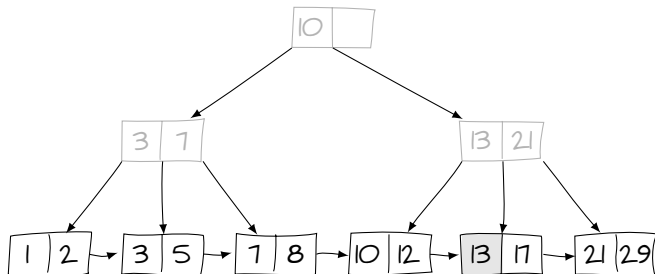
# Árvore B+

- ▶ Operação de busca exata
  - ▶ Parâmetro de chave: 13
  - ▶ A busca tem início pela raiz da árvore, checando as chaves do nó e acessando as subárvores



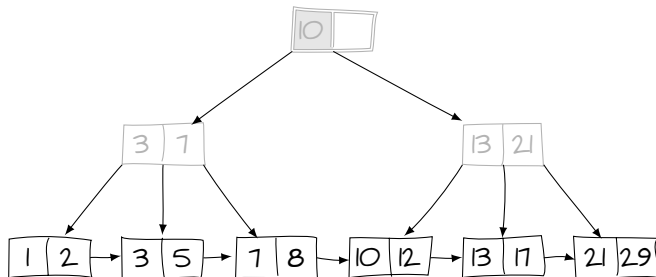
# Árvore B+

- ▶ Operação de busca exata
  - ▶ Parâmetro de chave: 13
  - ▶ A referência do nó encontrado é retornada



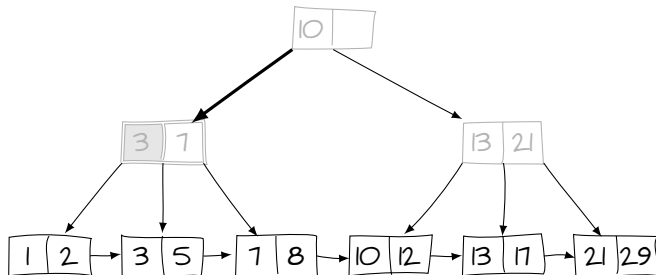
# Árvore B+

- ▶ Operação de busca por intervalo
  - ▶ Parâmetros de chave: 4 e 11
  - ▶ A busca tem início pela raiz da árvore, checando as chaves do nó e acessando as subárvores



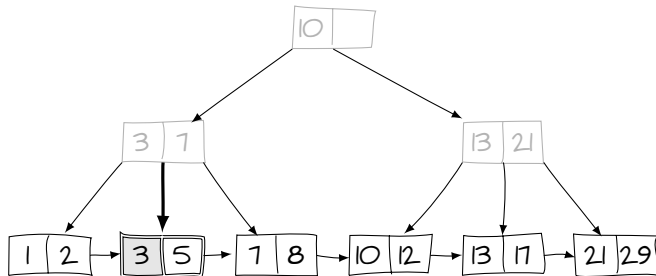
# Árvore B+

- ▶ Operação de busca por intervalo
  - ▶ Parâmetros de chave: 4 e 11
  - ▶ A busca tem início pela raiz da árvore, checando as chaves do nó e acessando as subárvores



# Árvore B+

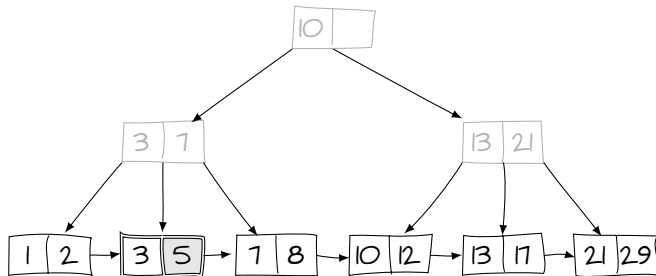
- ▶ Operação de busca por intervalo
  - ▶ Parâmetros de chave: 4 e 11
  - ▶ A busca tem início pela raiz da árvore, checando as chaves do nó e acessando as subárvores





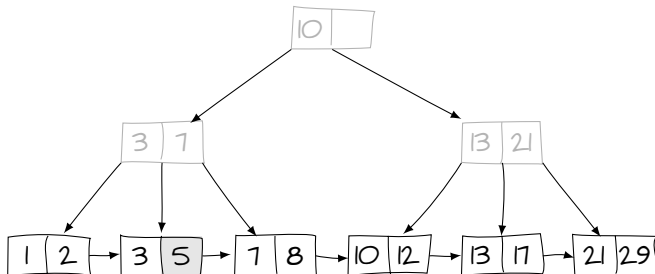
# Árvore B+

- ▶ Operação de busca por intervalo
  - ▶ Parâmetros de chave: 4 e 11
  - ▶ A busca tem início pela raiz da árvore, checando as chaves do nó e acessando as subárvores



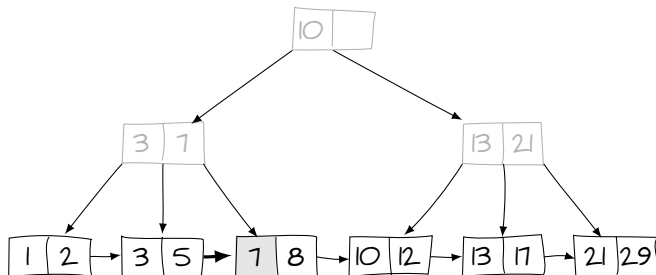
# Árvore B+

- ▶ Operação de busca por intervalo
  - ▶ Parâmetros de chave: 4 e 11
  - ▶ As referências dos nós no intervalo são retornadas



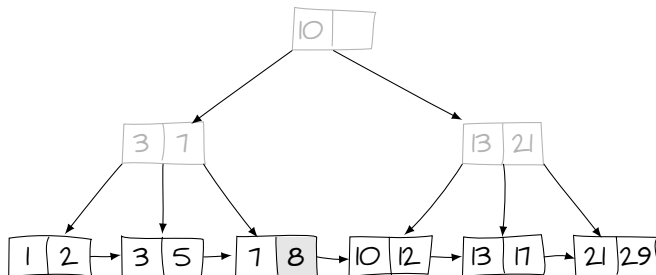
# Árvore B+

- ▶ Operação de busca por intervalo
  - ▶ Parâmetros de chave: 4 e 11
  - ▶ As referências dos nós no intervalo são retornadas



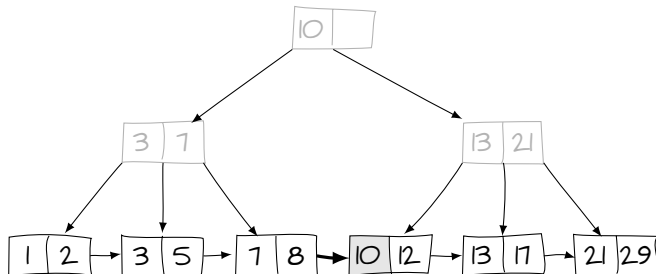
# Árvore B+

- ▶ Operação de busca por intervalo
  - ▶ Parâmetros de chave: 4 e 11
  - ▶ As referências dos nós no intervalo são retornadas



# Árvore B+

- ▶ Operação de busca por intervalo
  - ▶ Parâmetros de chave: 4 e 11
  - ▶ As referências dos nós no intervalo são retornadas



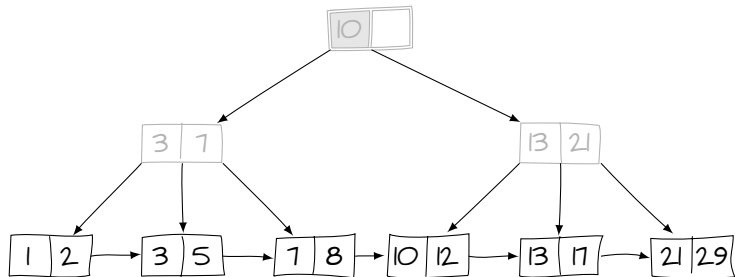
# Árvore B+

- ▶ Busca por intervalo
  - ▶ Retorna nós no intervalo de chaves

```
1 // Procedimento de busca por intervalo
2 nos* busca_intervalo(nos* x, uint32_t a, uint32_t b) {
3     uint32_t ini = min(a, b), fim = max(a, b);
4     nos* r = criar_nos();
5     no* f = busca_folha(x, ini, fim);
6     while(f != NULL && f->C[0] <= fim) {
7         adicionar_no(r, f);
8         f = f->P[k];
9     }
10    return r;
11 }
```

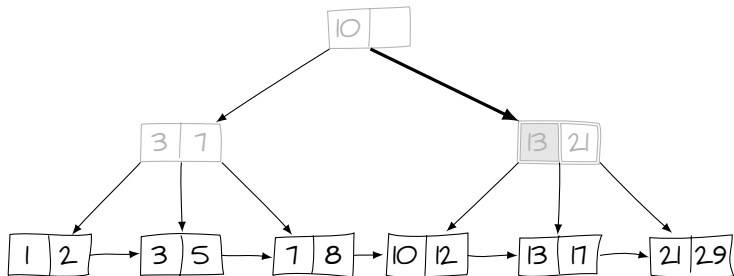
# Árvore B+

- ▶ Operação de inserção
  - ▶ Parâmetro de chave: 19



# Árvore B+

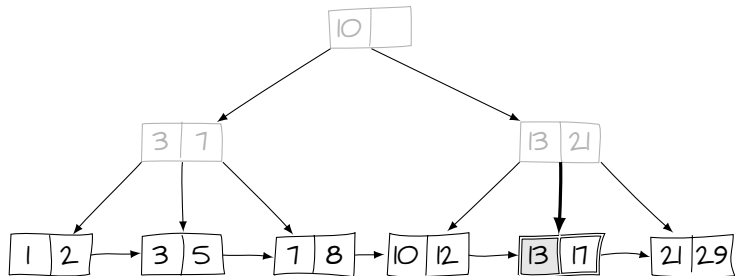
- ▶ Operação de inserção
  - ▶ Parâmetro de chave: 19





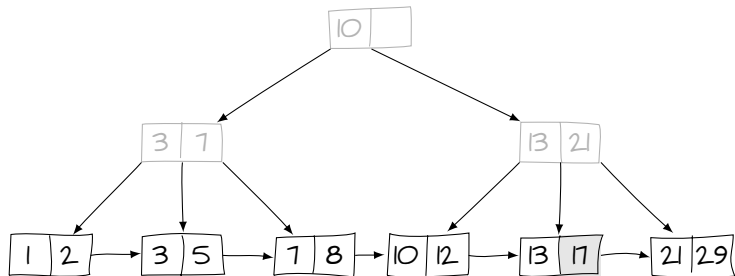
# Árvore B+

- ▶ Operação de inserção
  - ▶ Parâmetro de chave: 19



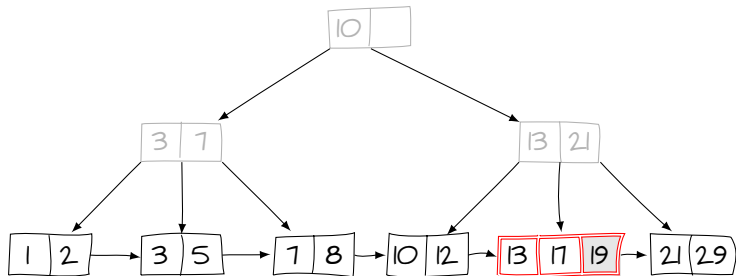
# Árvore B+

- ▶ Operação de inserção
  - ▶ Parâmetro de chave: 19



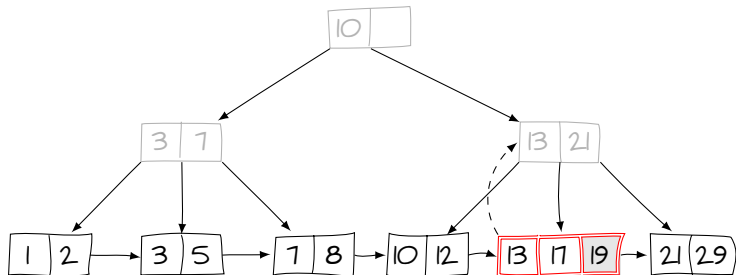
# Árvore B+

- ▶ Operação de inserção
  - ▶ Parâmetro de chave: 19



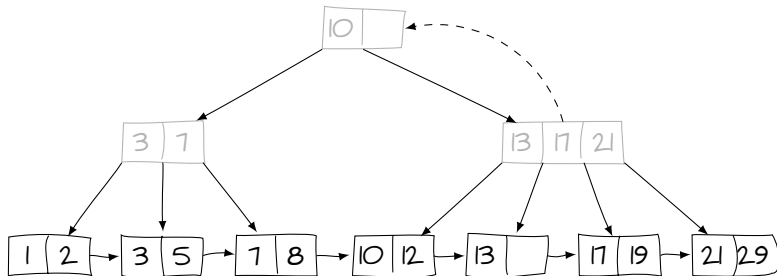
# Árvore B+

- ▶ Operação de inserção
  - ▶ Parâmetro de chave: 19



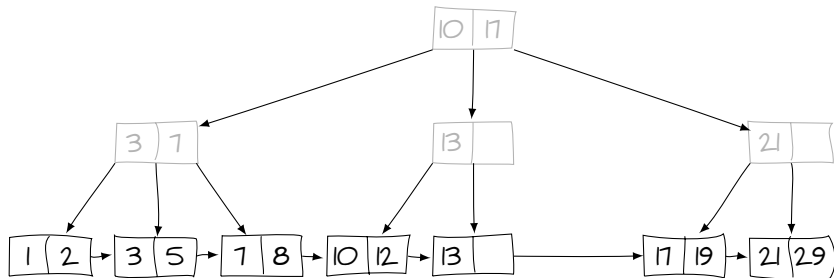
# Árvore B+

- ▶ Operação de inserção
  - ▶ Parâmetro de chave: 19



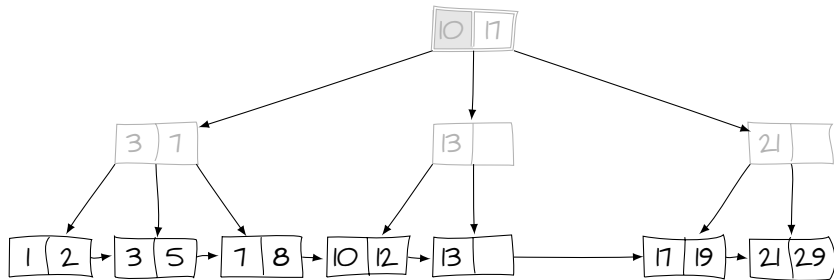
# Árvore B+

- ▶ Operação de inserção
  - ▶ Parâmetro de chave: 19



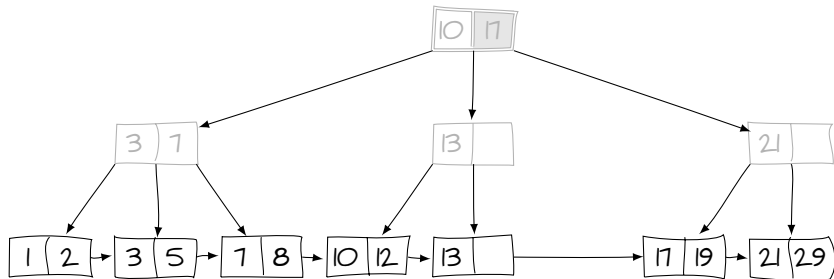
# Árvore B+

- ▶ Operação de remoção
  - ▶ Parâmetro de chave: 21
  - ▶ Caso 1: nó interno com filho  $n > \frac{k}{2}$



# Árvore B+

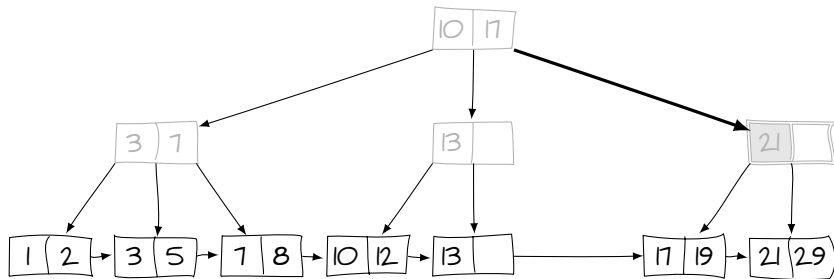
- ▶ Operação de remoção
  - ▶ Parâmetro de chave: 21
  - ▶ Caso 1: nó interno com filho  $n > \frac{k}{2}$





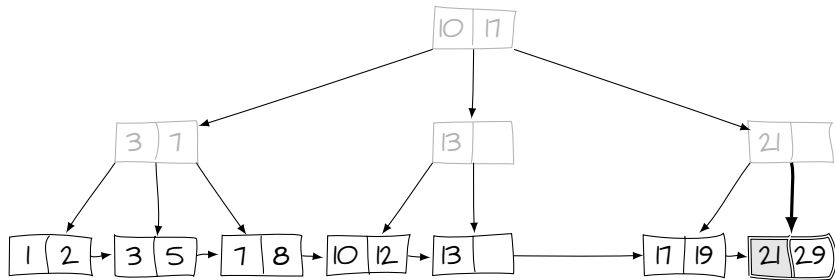
# Árvore B+

- ▶ Operação de remoção
  - ▶ Parâmetro de chave: 21
  - ▶ Caso 1: nó interno com filho  $n > \frac{k}{2}$



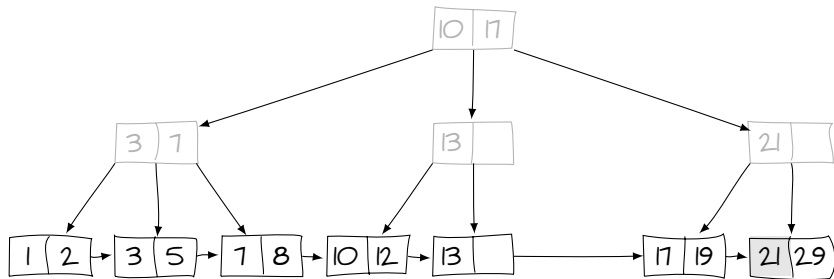
# Árvore B+

- ▶ Operação de remoção
  - ▶ Parâmetro de chave: 21
  - ▶ Caso 1: nó interno com filho  $n > \frac{k}{2}$



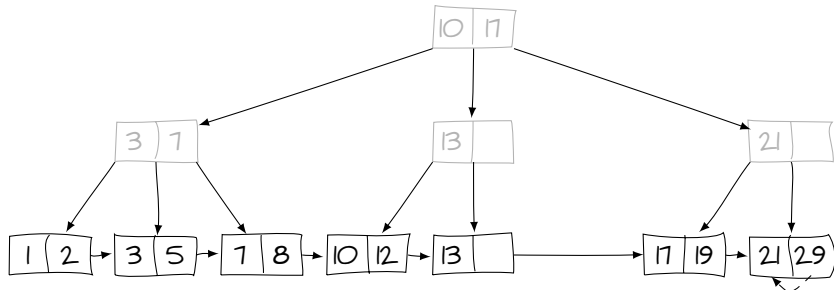
# Árvore B+

- ▶ Operação de remoção
  - ▶ Parâmetro de chave: 21
  - ▶ Caso 1: nó interno com filho  $n > \frac{k}{2}$



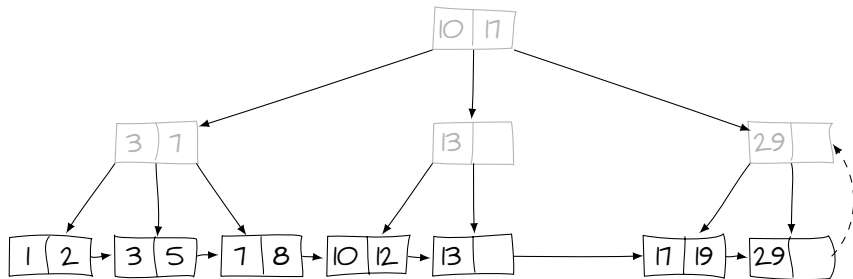
# Árvore B+

- ▶ Operação de remoção
  - ▶ Parâmetro de chave: 21
  - ▶ Caso 1: nó interno com filho  $n > \frac{k}{2}$



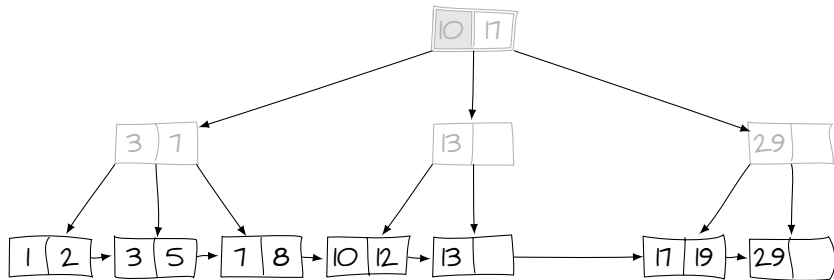
# Árvore B+

- ▶ Operação de remoção
  - ▶ Parâmetro de chave: 21
  - ▶ Caso 1: nó interno com filho  $n > \frac{k}{2}$



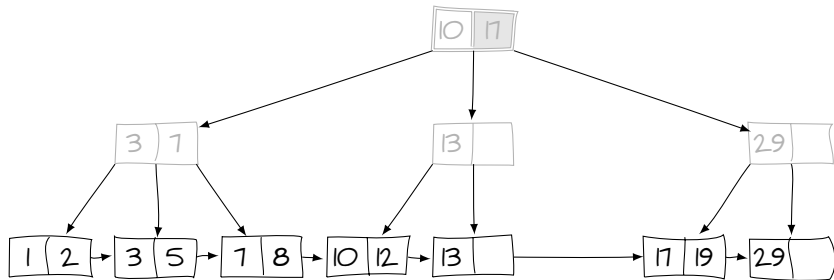
# Árvore B+

- ▶ Operação de remoção
  - ▶ Parâmetro de chave: 29
  - ▶ Caso 1: nó interno com filho  $n > \frac{k}{2}$



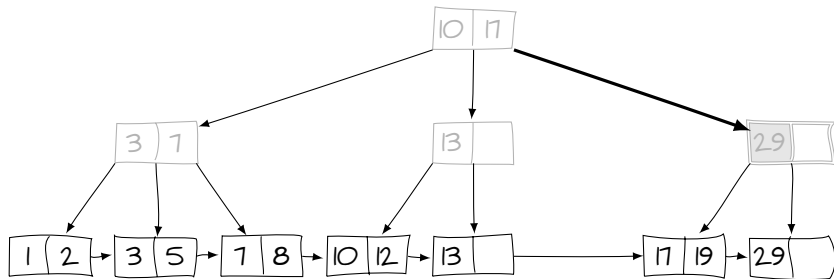
# Árvore B+

- ▶ Operação de remoção
  - ▶ Parâmetro de chave: 29
  - ▶ Caso 1: nó interno com filho  $n > \frac{k}{2}$



# Árvore B+

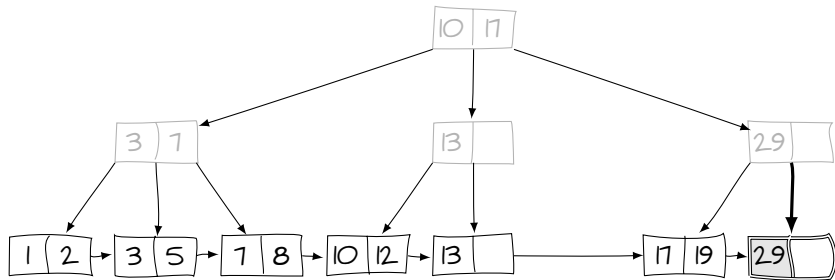
- ▶ Operação de remoção
  - ▶ Parâmetro de chave: 29
  - ▶ Caso 1: nó interno com filho  $n > \frac{k}{2}$





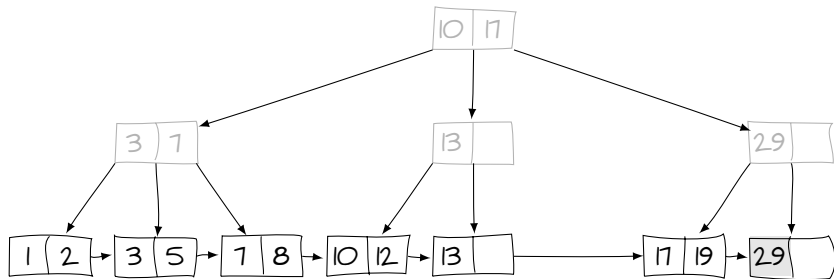
# Árvore B+

- ▶ Operação de remoção
  - ▶ Parâmetro de chave: 29
  - ▶ Caso 1: nó interno com filho  $n > \frac{k}{2}$



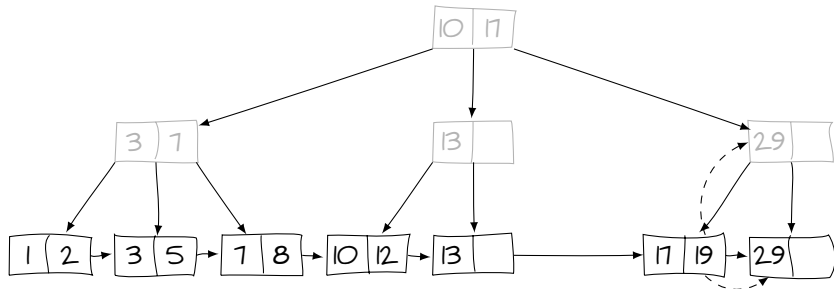
# Árvore B+

- ▶ Operação de remoção
  - ▶ Parâmetro de chave: 29
  - ▶ Caso 1: nó interno com filho  $n > \frac{k}{2}$



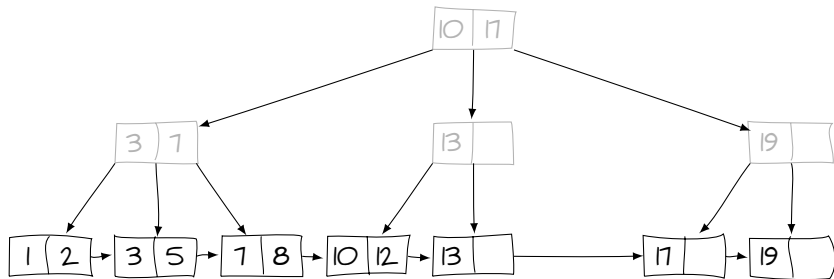
# Árvore B+

- ▶ Operação de remoção
  - ▶ Parâmetro de chave: 29
  - ▶ Caso 1: nó interno com filho  $n > \frac{k}{2}$



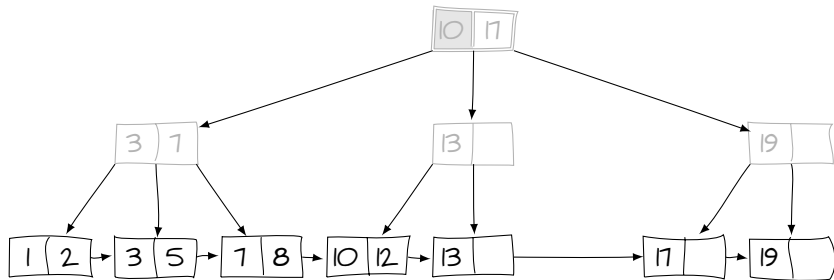
# Árvore B+

- ▶ Operação de remoção
  - ▶ Parâmetro de chave: 29
  - ▶ Caso 1: nó interno com filho  $n > \frac{k}{2}$



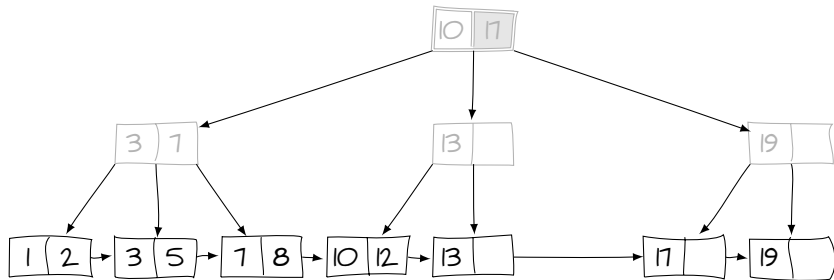
# Árvore B+

- ▶ Operação de remoção
  - ▶ Parâmetro de chave: 19
  - ▶ Caso 2: nó interno com filhos  $n = \frac{k}{2}$



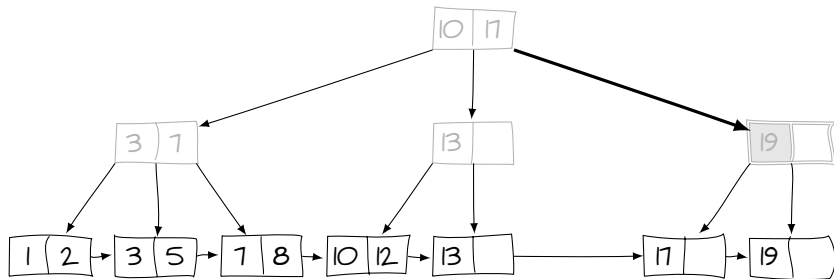
# Árvore B+

- ▶ Operação de remoção
  - ▶ Parâmetro de chave: 19
  - ▶ Caso 2: nó interno com filhos  $n = \frac{k}{2}$



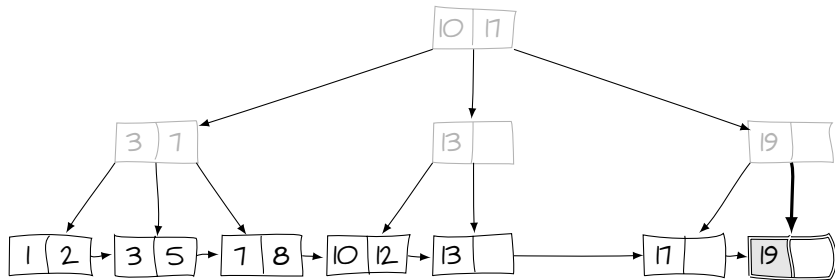
# Árvore B+

- ▶ Operação de remoção
  - ▶ Parâmetro de chave: 19
  - ▶ Caso 2: nó interno com filhos  $n = \frac{k}{2}$



# Árvore B+

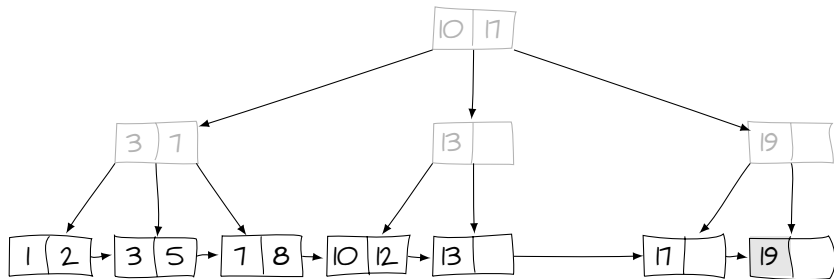
- ▶ Operação de remoção
  - ▶ Parâmetro de chave: 19
  - ▶ Caso 2: nó interno com filhos  $n = \frac{k}{2}$





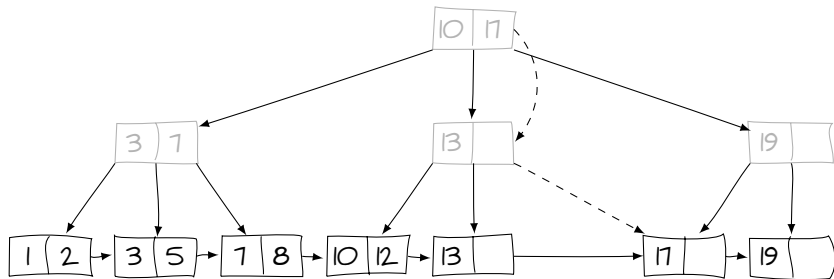
# Árvore B+

- ▶ Operação de remoção
  - ▶ Parâmetro de chave: 19
  - ▶ Caso 2: nó interno com filhos  $n = \frac{k}{2}$



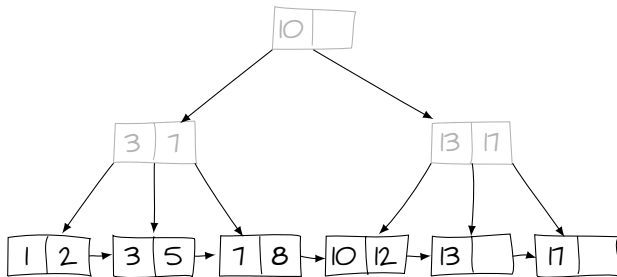
# Árvore B+

- ▶ Operação de remoção
  - ▶ Parâmetro de chave: 19
  - ▶ Caso 2: nó interno com filhos  $n = \frac{k}{2}$



# Árvore B+

- ▶ Operação de remoção
  - ▶ Parâmetro de chave: 19
  - ▶ Caso 2: nó interno com filhos  $n = \frac{k}{2}$



# Árvore B+

- ▶ Análise de complexidade
  - ▶ Com ordem  $k$ , no pior caso, as operações percorrem a altura  $h = \log_k n$  da árvore com  $n$  nós, entretanto, na busca por intervalo, o pior caso é  $O(n)$ , uma vez que o intervalo pode conter todos os nós da árvore
  - ▶ Espaço:  $\Theta(n)$
  - ▶ Tempo:  $\Omega(\log_k n)$  e  $O(n)$

# Exemplo

- ▶ Construa uma árvore B+ de ordem 3
  - ▶ Insira os elementos com chaves 13, 2, 34, 11, 7, 43 e 9
  - ▶ Realize a remoção dos elementos de chave 7 e 9
  - ▶ Compare as diferenças entre as árvores B e B+

# Exercício

- ▶ A empresa de tecnologia Poxim Tech está desenvolvendo um banco de dados distribuído para arquivos baseado em *blockchain* e árvore B+
  - ▶ Os arquivos possuem nomes + extensão com até 30 caracteres, compostos exclusivamente por letras
  - ▶ A codificação do código *hash* é feita em hexadecimal de 128 bits com caracteres maiúsculos, sendo utilizado como chave para buscas
  - ▶ Operações disponíveis:
    - ▶ **INSERT** nome tamanho hash
    - ▶ **SELECT** hash
    - ▶ **SELECT RANGE** hash1 hash2

# Exercício

- ▶ Formato de arquivo de entrada
  - ▶ [*#Ordem da árvore*]
  - ▶ [*#Quantidade de arquivos( $n$ )*]
  - ▶ [*Nome<sub>1</sub>*] [*Tamanho<sub>1</sub>*] [*Hash<sub>1</sub>*]
  - ▶  $\vdots$
  - ▶ [*Nome<sub>n</sub>*] [*Tamanho<sub>n</sub>*] [*Hash<sub>n</sub>*]
  - ▶ [*#Número de operações( $m$ )*]
  - ▶ [*Operação<sub>1</sub>*]
  - ▶  $\vdots$
  - ▶ [*Operação<sub>m</sub>*]

## Exercício

- ▶ Formato de arquivo de entrada

```

1      3
2      5
3      f.b_1_00000000000000000000000000000000
4      f.c_2_000000000000000000000000000000001
5      f.d_3_000000000000000000000000000000002
6      f.e_4_000000000000000000000000000000003
7      f.f_5_000000000000000000000000000000004
8      5
9      INSERT_f.a_0_FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
10     SELECT_000000000000000000000000000000004
11     SELECT_123456789ABCDEF0123456789ABCDEF0
12     SELECT_RANGE_000000000000000000000000000000005_
        FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFE
13     SELECT_RANGE_000000000000000000000000000000006_
        FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF

```



# Exercício

- ▶ Formato de arquivo de saída
  - ▶ Conteúdo armazenado pelo nó da árvore

```
1  [000000000000000000000000000000000004]
2  f.f:size=5,hash=000000000000000000000000000000000004
3  f.a:size=0,hash=FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
4  [123456789ABCDEF0123456789ABCDEF0]
5  -
6  [000000000000000000000000000000000005,
   FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFE]
7  -
8  [000000000000000000000000000000000006,
   FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF]
9  f.f:size=5,hash=000000000000000000000000000000000004
10 f.a:size=0,hash=FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
```