UNIVERSIDADE FEDERAL
DE ALAGOAS

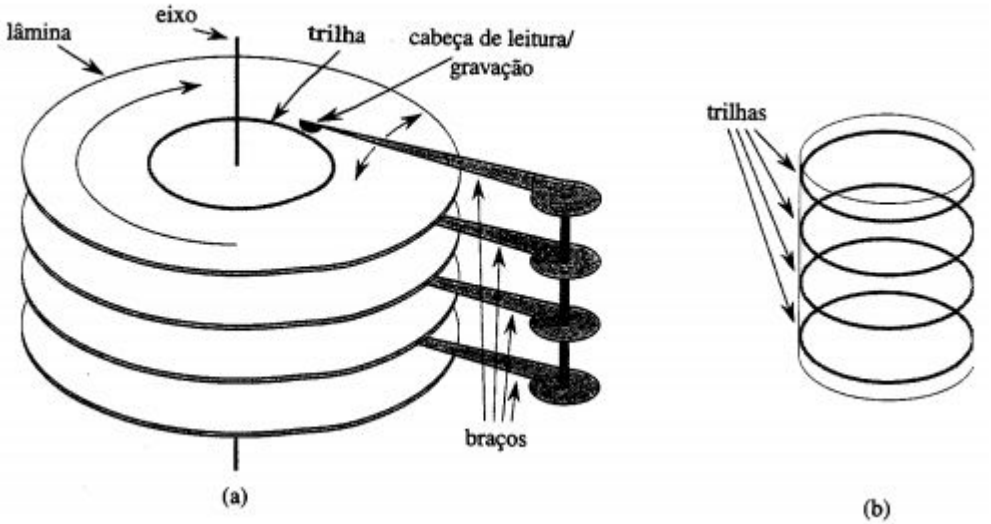
Árvore B

github.com/aleffarias/Project-P2.git

Antonio Alef Oliveira Farias
João Paulo Agostinho da Silva
Milena Balbino Nunes

Motivação

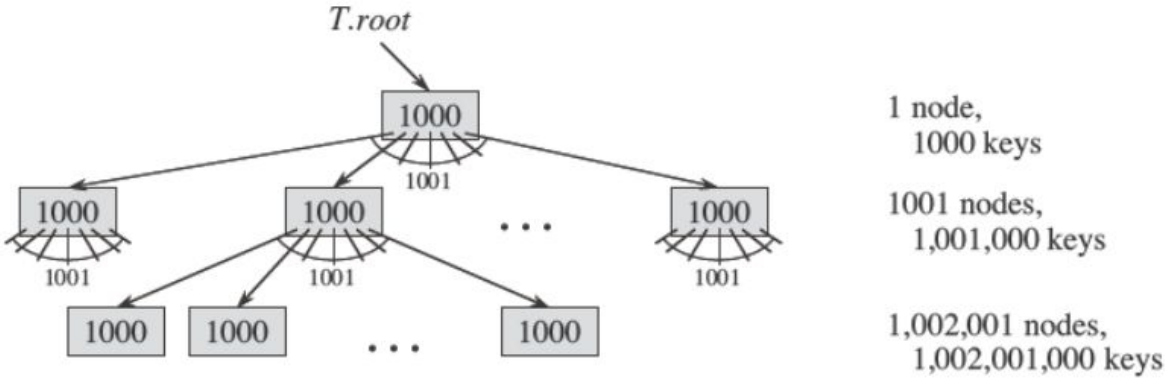
- Solução para cenários em que o volume de informação é alto e, portanto, apenas algumas **páginas** da árvore podem ser carregadas em memória primária;
- Diminuir o número de acesso ao disco;



Estrutura de Dados / Algoritmo

O que é?

- São árvores de pesquisa balanceadas projetadas para funcionar bem em discos magnéticos ou outros dispositivos de armazenamento secundário de acesso direto (Cormen, T.H.).

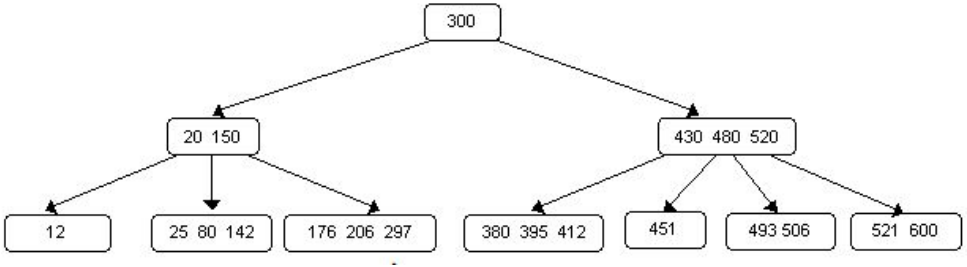


Para que serve?

Árvores B são a estrutura subjacente a muitos sistemas de arquivos e bancos de dados.

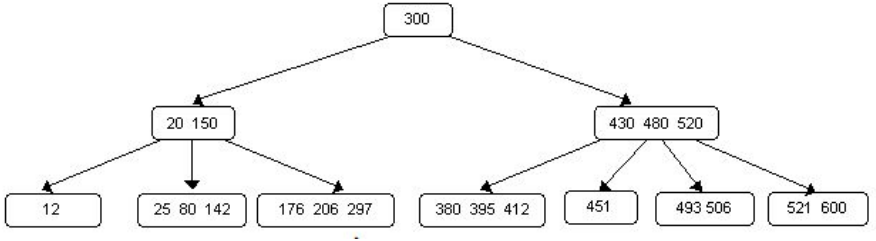
- O sistema de arquivos NTFS do Windows;
- O sistema de arquivos HFS do Mac;
- Os sistemas de arquivos ReiserFS, XFS, Ext3FS, JFS do Linux;
- Os bancos de dados ORACLE, DB2, INGRES, SQL e PostgreSQL.

Definições



1. Todo o nó **X** possui:
   - **n**, o número de chaves armazenadas em **X**;
   - as **n** chaves $k_1, k_2 \ldots k_n$ são armazenadas em ordem crescente;
   - **folha**, que indica se **X** é uma folha ou um nó interno.
2. Se **X** é um nó interno então ele possui **n+1** ponteiros $f_1, f_2 \ldots f_{n+1}$ para seus filhos (podendo alguns serem nulos).
3. Se k_i é qualquer chave na sub-árvore com raiz f_i, ***então:***
   - $$k_1 \leq X \rightarrow k_1 \leq k_2 \leq X \rightarrow k_2 \leq \ldots \leq X \rightarrow k_n \leq k_{n+1}$$
4. Toda folha tem a mesma profundidade, que é a altura da árvore.

Definições



5. Existe um número máximo e mínimo de filhos em um nó. Este número pode ser descrito em termos de um inteiro fixo **t** maior ou igual a 2 chamado grau mínimo.

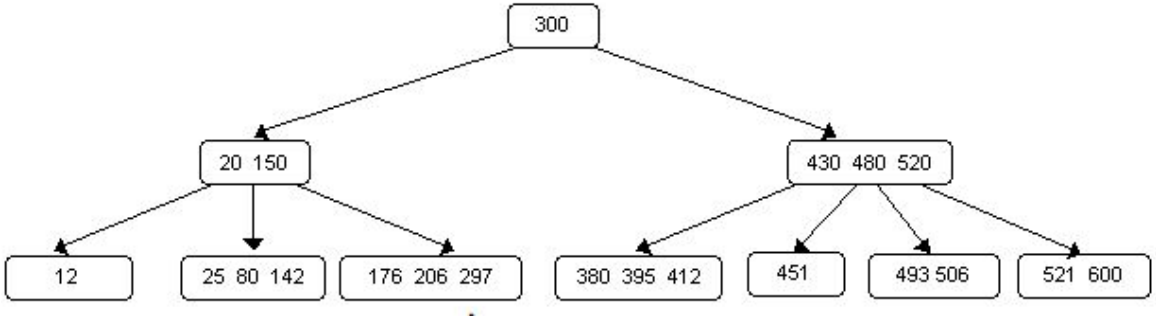
    - Todo o nó diferente da raiz deve possuir pelo menos **t-1** chaves. Todo o nó interno diferente da raiz deve possuir pelo menos **t** filhos. Se a árvore não é vazia, então a raiz possui pelo menos uma chave.

    - Todo o nó pode conter no máximo **2t - 1** chaves. Logo um nó interno pode ter no máximo **2t** filhos. Dizemos que um nó é **completo** se ele contém **2t - 1** chaves.

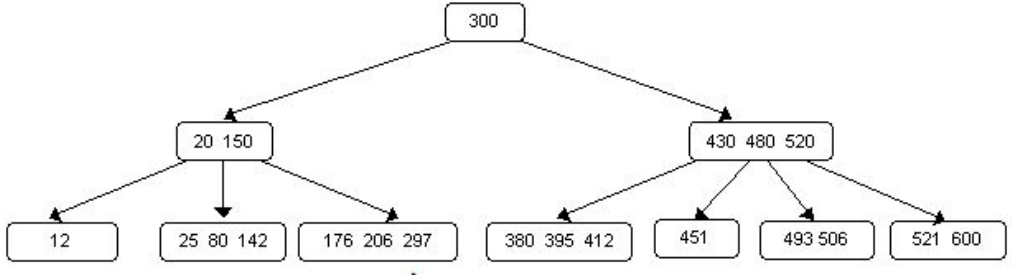
Estrutura do nó

```
const t = 2;
typedef struct no_arvoreB
arvoreB;


struct no_arvoreB {
        int num_chaves;
        int chaves[2*t-1];
        arvoreB *filhos[2*t];
        bool folha;
};
```



Busca




```c
int busca_binaria(arvoreB *no, int info) {
  int meio, i, f;
  i = 0;
  f = no->num_chaves - 1;

  while (i <= f) {
    meio = (i + f) / 2;
    if (no->chaves[meio] == info)
      return (meio);
    else if (no->chave[meio] > info
            f = meio - 1;
        else i = meio + 1;
  }
  return (i);
}
```

```c
bool busca(arvoreB *raiz, int info) {
  arvoreB *no;
  int pos;

  no = raiz;
  while (no != NULL) {
    pos = busca_binaria(no, info);
    if (pos < no->num_chaves && no->chaves[pos] == info)
      return (true);
    else
      no = no->filhos[pos];
  }
  return (false);
}
```

Referências

1. T.H. Cormen, C.E. Leiserson, R.L. Rivest, C. Stein, Introduction to Algorithms, 3rd edition, MIT Press, 2009.
2. IME. Disponível em: <https://www.ime.usp.br/~pf/estruturas-de-dados/aulas/B-trees.html> Acesso em: 19 de março de 2019.
3. LCAD. Disponível em: <http://www.lcad.icmc.usp.br/~nonato/ED/B_arvore/btree.htm> Acesso em: 30 de março de 2019.