

Árvores: árvores B

Prof. Jefferson T. Oliva

Algoritmos e Estrutura de Dados II (AE23CP)
Engenharia de Computação
Departamento Acadêmico de Informática (Dainf)
Universidade Tecnológica Federal do Paraná (UTFPR)
Campus Pato Branco

- Árvores B
 - Definições
 - Estrutura de dados
- Operações em Árvore B

- Busca em memória primária
 - Busca sequencial
 - Busca sequencial indexada
 - Busca binária
 - Busca por interpolação
 - Hashing
 - Árvore binária de busca
 - Árvore AVL
 - Árvore rubro-negra

Introdução

- Pesquisa em memória primária vs. secundária
 - Primária: acesso mais rápido, portanto, mais caro



- Secundária: mais barato, portanto, acesso mais lento

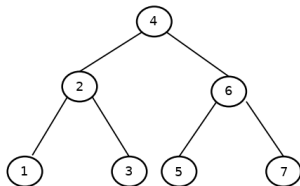


Árvores B

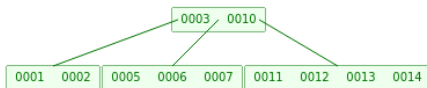
- Amplamente utilizadas para o armazenamento em memória secundária
- Diversas aplicações/ferramentas utilizam árvores B (*B-tree*)
 - Sistemas de arquivos (e.g. NTFS e ext4)
 - Banco de dados (e.g. Oracle e PostgreSQL)
- Árvores B são estruturas de dados projetadas para a indexação de dados
 - Estrutura mais avançada em relação ao arquivo sequencial indexado

Árvores B

- Árvore B é n-ária
 - Exemplo de árvore binária de busca



- Exemplo de árvore B



- Árvore binária de busca: cada nó tem um elemento e entre 0 e 2 nós filhos
- Árvore n -ária: cada nó pode ter entre 1 e $n - 1$ elementos e até n filhos

- Árvore de busca balanceada
- O nó de uma árvore B também é chamado de **página**
 - Cada nó pode ter mais de um elemento
- Ordem de árvore B: é o número máximo de páginas descendentes que uma página pode ter (Knuth)
 - Em Cormen et al. (2012) é definida como o número mínimo de páginas descendentes que uma página pode ter
 - Nesse material de aula é considerada a definição de Knuth

Árvores B

Definições

- Dada uma árvore B de ordem N :
 - Página raiz
 - Elementos: entre 1 (mínimo) e $N - 1$ (máximo)
 - Descendentes (subárvores): entre 2 e N (caso não seja página folha)
 - Página interna
 - Elementos: entre $\lceil \frac{N}{2} \rceil - 1$ e $N - 1$
 - Descendentes (subárvores): entre $\lceil \frac{N}{2} \rceil$ e N
 - Página folha
 - Elementos: entre $\lceil \frac{N}{2} \rceil - 1$ e $N - 1$
 - Não possui descendentes
 - Todas as páginas folhas estão no mesmo nível (*i.e.* possuem a mesma profundidade)

Árvores B

Definições

- Exemplos para uma árvore B de ordem 5 ($N = 5$):

Página raiz

- Elementos: entre 1 (mínimo) e $N - 1$ (máximo)
- Descendentes (subárvores): entre 2 e N (caso não seja página folha)

1 elemento



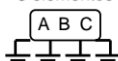
2 subárvores

2 elementos



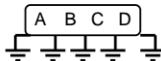
3 subárvores

3 elementos



4 subárvores

4 elementos



5 subárvores

- Quando uma página de árvore B de ordem N tem $N - 1$ elementos, diz-se que a mesma está cheia

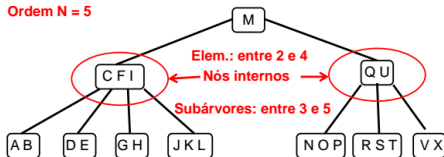
Árvores B

Definições

- Exemplos para uma árvore B de ordem 5 ($N = 5$):

Página (nó) interna

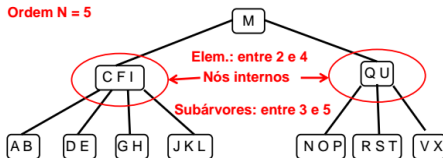
- Elementos: entre $\lceil \frac{N}{2} \rceil - 1$ e $N - 1$
- Descendentes (subárvores): entre $\lceil \frac{N}{2} \rceil$ e N



Árvores B

Definições

- Exemplos para uma árvore B de ordem 5 ($N = 5$):

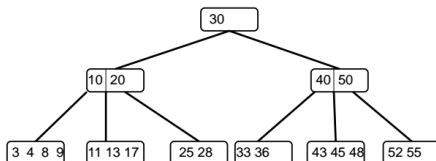


- Dada uma árvore B de ordem 5
 - Cada página interna ou folha pode ter entre 2 (mínimo) e 4 (máximo) elementos
 - Cada página interna pode ter entre 3 (mínimo) e 5 (máximo) descendentes

Árvores B

Definições

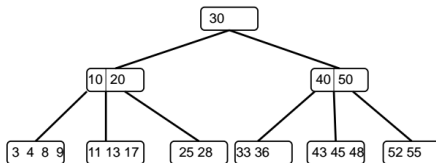
- Qual é a ordem da árvore B abaixo:



Árvores B

Definições

- Qual é a ordem da árvore B abaixo:
 - R.: $N = 5$



Árvores B

Estrutura de dados

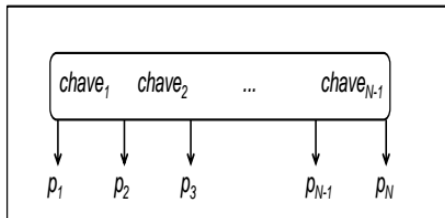
```
#define N ?

typedef struct NodeB NodeB;

struct NodeB{
    int nro_chaves;
    int chaves[N - 1];
    NodeB *filhos[N];
    int eh_no_folha;
};
```


Árvores B

Estrutura de dados



- Função para inicialização de um ponteiro do tipo NodeB (1)

```
NodeB* criar() {  
    NodeB *tree = (NodeB *) malloc(sizeof(NodeB));  
    int i;  
  
    tree->eh_no_folha = 1;  
    tree->nro_chaves = 0;  
  
    for (i = 0; i < N; i++)  
        tree->filhos[i] = NULL;  
  
    return tree;  
}
```

- Função para inicialização de um ponteiro do tipo NodeB (2)

```
NodeB* criar() {  
    NodeB *tree = (NodeB *) calloc(sizeof(NodeB));  
    int i;  
  
    tree->eh_no_folha = 1;  
  
    return tree;  
}
```

Operações em Árvores B

Operações em Árvores B

- Pesquisa
- Inserção
- Remoção

Operações em Árvores B

Pesquisa

- Semelhante à busca em árvores binárias de busca (incluindo as balanceadas, como as do tipo AVL e rubro-negra)
 - Em vez de termos até duas opções (esquerda e direita) para continuar a busca a partir de um nó, em árvores B podemos ter várias opções
 - Por exemplo, em uma árvore B de ordem N podemos ter até N caminhos para continuar a busca a partir de uma página
- Outro exemplo que podemos utilizar para comparar com a árvore B é a busca sequencial indexada
 - A busca começa na página raiz (de forma similar à tabela de índices): caso a chave não seja encontrada, devemos continuar a busca em uma das páginas filhas (se existir)
 - Na busca sequencial indexada, a busca continuaria em um "bloco" do respectivo arquivo
 - Em árvores B, esses "blocos" seriam páginas filhas

- Implementação com busca sequencial

```
int pesquisaSequencial(int key, NodeB *tree){  
    int i;  
    if (tree != NULL){  
        for (i = 0; i < tree->nro_chaves && key < tree->chaves[i];  
             i++)  
            if (key == tree->chaves[i])  
                return 1;  
        return pesquisaSequencial(key, tree->filhos[i]);  
    }  
    return 0;  
}
```

- Implementação com busca binária

```
int busca_binaria(int key, NodeB *tree){
    int ini, fim, meio;
    if (tree != NULL){
        ini = 0;
        fim = tree->nro_chaves - 1;
        while (ini <= fim){
            meio = (ini + fim) / 2;
            if (tree->chaves[meio] == key)
                return meio;
            else if (tree->chaves[meio] < key)
                ini = meio + 1;
            else
                fim = meio - 1;
        }
        return ini;
    }
    return -1;
}
```

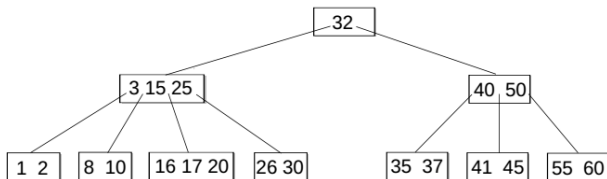

- Implementação com busca binária

```
int pesquisar(int key, NodeB *tree){
    int pos = busca_binaria(key, tree);
    if (pos >= 0){
        if (tree->chaves[key] == key)
            return 1;
        else
            return pesquisar(key, tree->filhos[pos]);
    }
    return 0;
}
```

Operações em Árvores B

Pesquisa

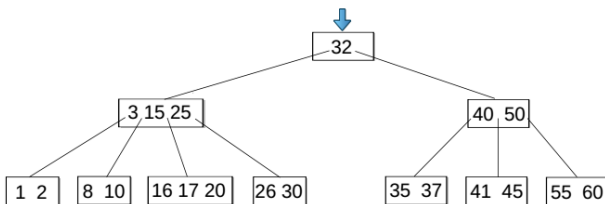
- Exemplo: busca sequencial pela chave 20



Operações em Árvores B

Pesquisa

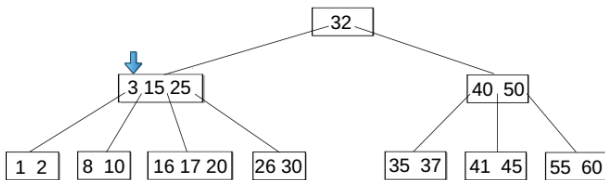
- Exemplo: busca sequencial pela chave 20



Operações em Árvores B

Pesquisa

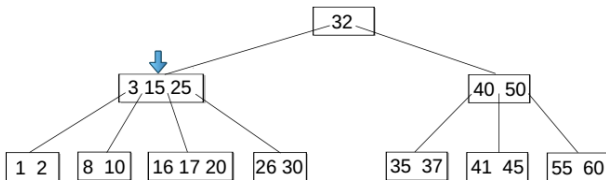
- Exemplo: busca sequencial pela chave 20



Operações em Árvores B

Pesquisa

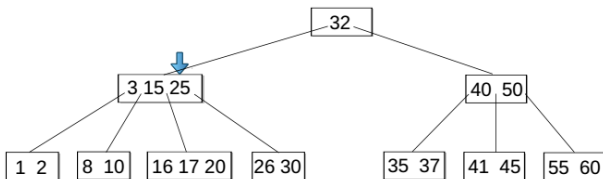
- Exemplo: busca sequencial pela chave 20



Operações em Árvores B

Pesquisa

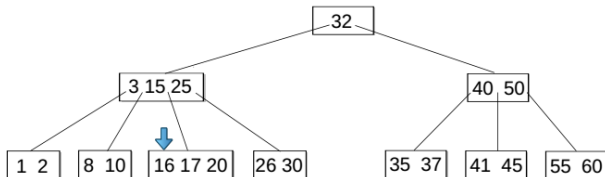
- Exemplo: busca sequencial pela chave 20



Operações em Árvores B

Pesquisa

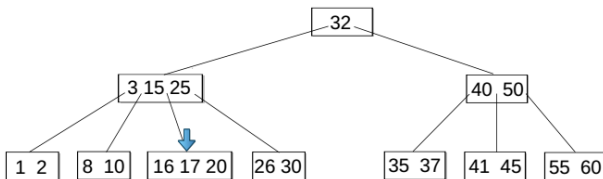
- Exemplo: busca sequencial pela chave 20



Operações em Árvores B

Pesquisa

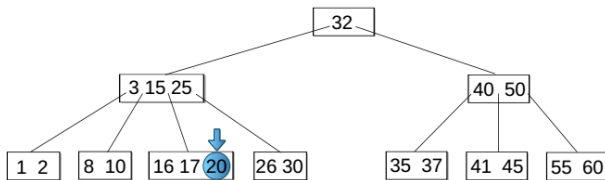
- Exemplo: busca sequencial pela chave 20



Operações em Árvores B

Pesquisa

- Exemplo: busca sequencial pela chave 20



Operações em Árvores B

Pesquisa

- Complexidade, onde N é a ordem da árvore e M é a quantidade total de chaves
 - Pesquisa sequencial
 - Tempo (pior caso): $O(N \log_N M)$
 - Espaço: $\Theta(M)$
 - Pesquisa binária
 - Tempo (pior caso): $O(\log_2 M)$, pois $\log_2 N * \log_N M = \log_2 M$
 - Espaço: $\Theta(M)$

Operações em Árvores B

Inserção

- Primeiramente é procurada uma página folha para a inserção de um elemento
 - Diferentemente de árvores binárias de busca, conforme visto em aulas anteriores, nem sempre é necessária a criação de uma nova página folha
 - Se a página não estiver cheia (ou seja, tem menos de $N - 1$ elementos), o novo elemento é alocado nessa página
 - Se a página estiver cheia, deverá ser criada uma nova página

Operações em Árvores B

Inserção

- Criação de nova página:
 - 1 Na página cheia, onde deveria ser inserido o novo elemento, é escolhido um valor intermediário, incluindo a nova chave (elemento)
 - 2 Em seguida, deve ser criado uma nova página, onde todos os elementos maiores que o valor intermediário deverão ser alocados
 - 3 A página que está cheia deverá contar com apenas os elementos menores que o valor intermediário, ou seja, a página passará ter a metade dos elementos em relação quando a mesma estava cheia
 - 4 O valor intermediário deve ser inserido na página pai, que passará a ter uma página filha a mais
 - Caso a página pai esteja cheia, as etapas de 1 a 3 são repetidas

Operações em Árvores B

Inserção

- Criação de nova página:
 - Caso a página pai for raiz e esteja cheia após sucessivas inserções, é criada uma nova página que passará ser raiz
 - Nesse caso, a árvore cresce em altura
 - Após a inserção, a árvore é mantida balanceada

Operações em Árvores B

Inserção

- Exemplo: inserção dos seguintes valores em uma árvore B de ordem 5: 10, 20, 30, 40, 50



Operações em Árvores B

Inserção

- Exemplo: inserção dos seguintes valores em uma árvore B de ordem 5: **10**, 20, 30, 40, 50

10

Operações em Árvores B

Inserção

- Exemplo: inserção dos seguintes valores em uma árvore B de ordem 5: **20**, 30, 40, 50

10 20

Operações em Árvores B

Inserção

- Exemplo: inserção dos seguintes valores em uma árvore B de ordem 5: **30**, 40, 50

10 20 30

Operações em Árvores B

Inserção

- Exemplo: inserção dos seguintes valores em uma árvore B de ordem 5: **40**, 50

10 20 30 40

Operações em Árvores B

Inserção

- Exemplo: inserção dos seguintes valores em uma árvore B de ordem 5: **50**

10 20 30 40

 50

Operações em Árvores B

Inserção

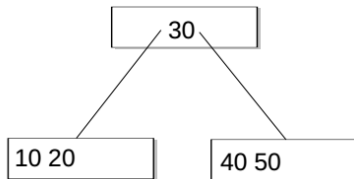
- Exemplo: inserção dos seguintes valores em uma árvore B de ordem 5: **50**



Operações em Árvores B

Inserção

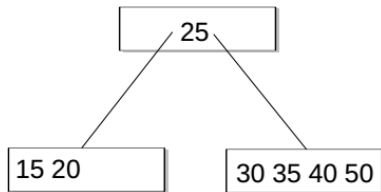
- Exemplo: inserção dos seguintes valores em uma árvore B de ordem 5: **50**



Operações em Árvores B

Inserção

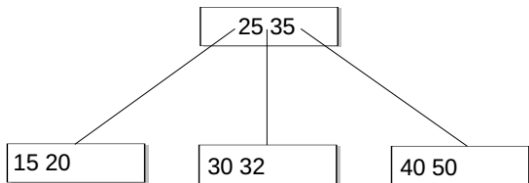
- Outro exemplo: inserção da chave 31 na seguinte árvore B de ordem 5:



Operações em Árvores B

Inserção

- Outro exemplo: inserção da chave 31 na seguinte árvore B de ordem 5:



Operações em Árvores B

Inserção

- Complexidade de tempo: $O(\log_2 M)$
 - Caso a inserção for implementada com a busca binária
- Complexidade de espaço: $\Theta(M)$

Operações em Árvores B

Remoção

- A remoção ocorre após a busca por uma determinada chave
- Após a remoção, as propriedades de árvore B devem ser mantidas
- Para essa operação, devem ser considerados os seguintes casos
 - Remoção em página folha: caso a página tiver quantidade de elementos a partir de $N/2$, basta reorganizar a página
 - Remoção em página não folha: a chave é trocada com uma sucessora na página folha e, em seguida, eliminar a chave na folha conforme o caso anterior
 - Caso a remoção causar desequilíbrio na página, deve ser procurada uma página irmã que contenha quantidade de chaves maior que o mínimo para redistribuição das chaves

Operações em Árvores B

Remoção

- Para essa operação, devem ser considerados os seguintes casos
 - Caso a remoção causar desequilíbrio na página e não houver página irmã que viabilize a redistribuição de chaves, a solução é concatenar duas páginas e atualizar a(s) página(s) pai(s)
 - Caso a página pai ficar desbalanceada, repetir o processo de redistribuição ou concatenação de páginas
- Complexidade da operação de remoção (caso seja aplicada a operação de busca binária): $O(\log_2 n)$

- Variações da árvore B:
 - B+
 - B*



Cormen, T. H., Leiserson, C. E., Rivest, R. L., Stein, C.
Introduction to Algorithms.
Third edition, The MIT Press, 2009.



Marin, L. O.
Árvores B. AE23CP - Algoritmos e Estrutura de Dados II.
Slides. Engenharia de Computação. Dainf/UTFPR/Pato Branco, 2017.



Ziviani, N.
Projeto de Algoritmos - com implementações em Java e C++.
Thomson, 2007.