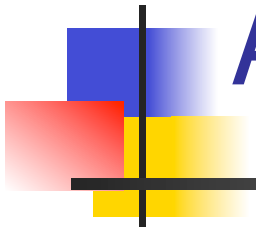


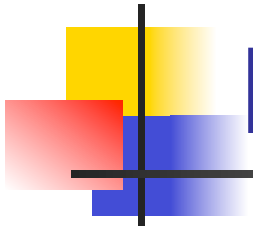
Árvores-B: Motivação

Thiago A. S. Pardo

Leandro C. Cintra

M.C.F. de Oliveira

Cristina D. A. Ciferri



Problema

- Cenário até então
  - acesso a disco é caro (lento)
  - pesquisa binária é útil em índices ordenados...
- Mas
  - com índice grande que não cabe em memória principal, pesquisa binária exige muitos acessos a disco
    - Exemplo: 15 itens podem requerer 4 acessos, enquanto 1.000 itens podem requerer até 11 acessos

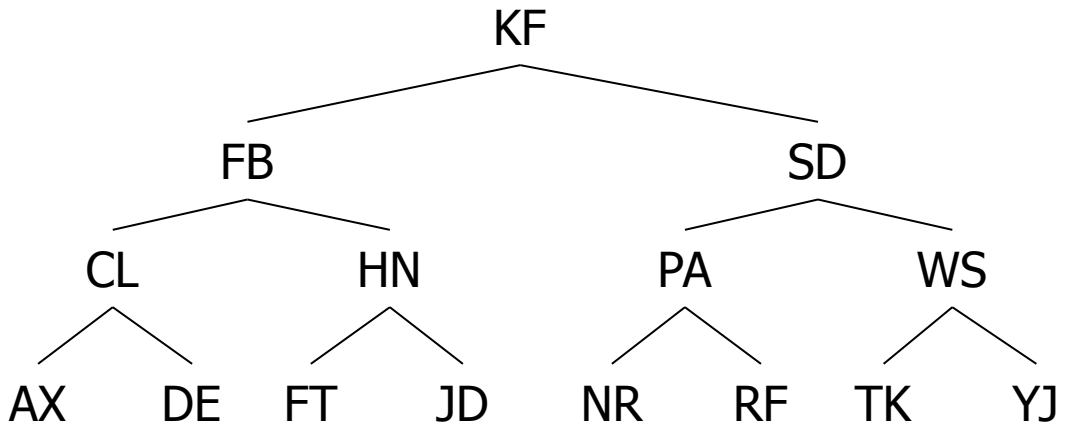
➢ Manter em disco um índice ordenado para busca binária tem custo proibitivo

Ideia

Criar um método no qual
as inserções e remoções tenham
apenas efeitos locais, ou seja,
um método que não exija
a reorganização total do índice

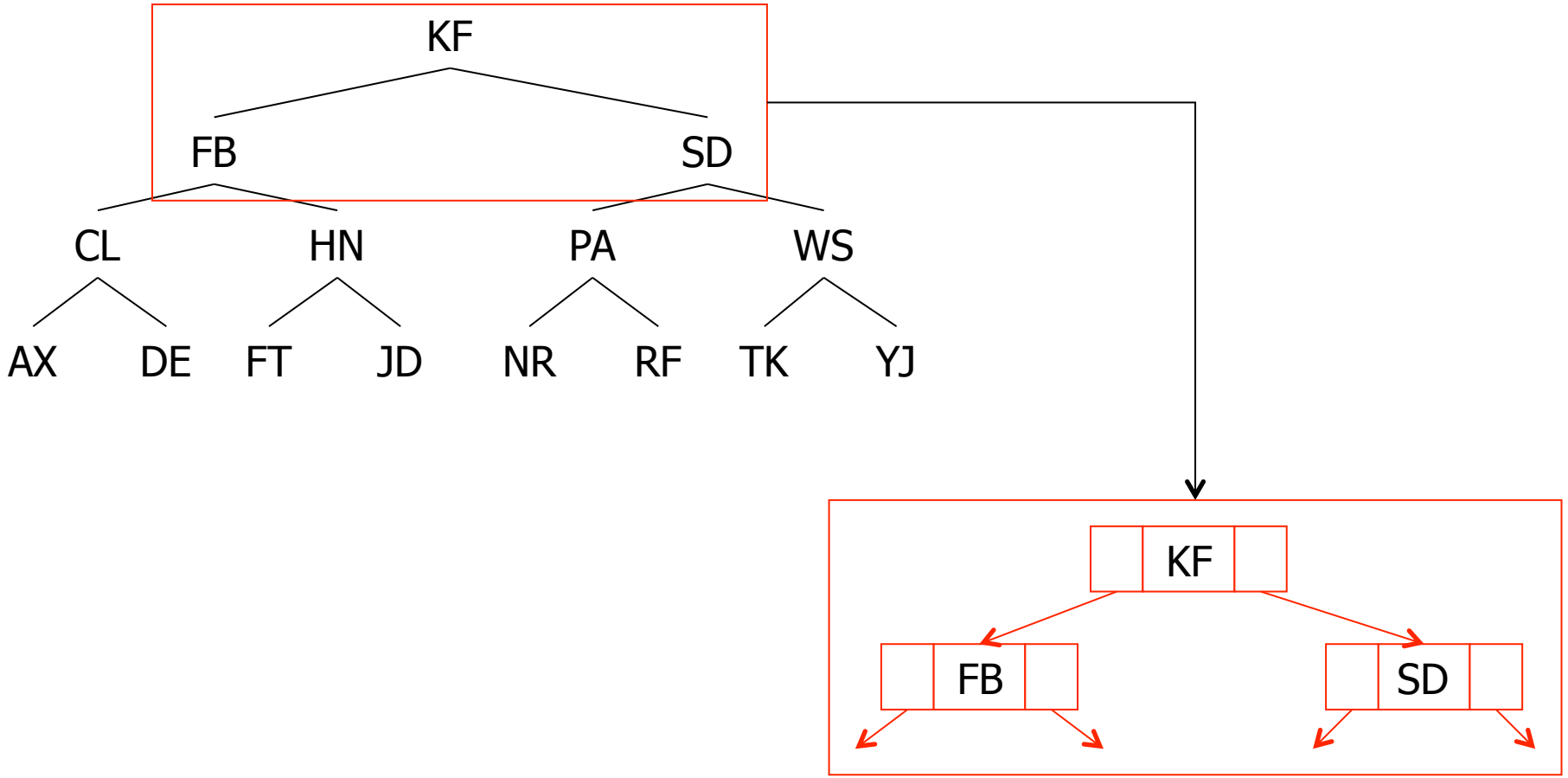
Árvores binárias de busca

AX, CL, DE, FB, FT, HN, JD, KF, NR, PA, RF, SD, TK, WS, YJ

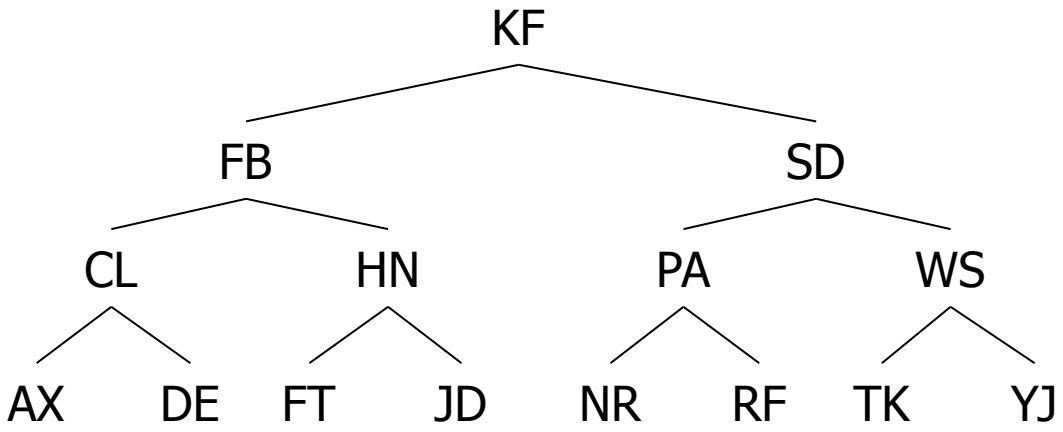


Vetor ordenado e representação por árvore binária

Árvores binárias de busca

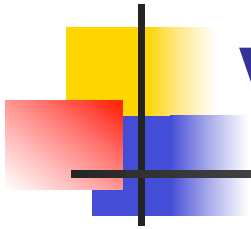


Representação da árvore binária de busca no arquivo



		Chave	Filho esq.	Filho dir.
	0	FB	10	8
	1	JD	-1	-1
	2	RF	-1	-1
	3	SD	6	13
	4	AX	-1	-1
	5	YJ	-1	-1
	6	PA	11	2
	7	FT	-1	-1
	8	HN	7	1
raiz →	9	KF	0	3
	10	CL	4	12
	11	NR	-1	-1
	12	DE	-1	-1
	13	WS	14	5
	14	TK	-1	-1

Registros são mantidos em arquivo, e ponteiros (**esq** e **dir**) indicam onde estão os registros filhos



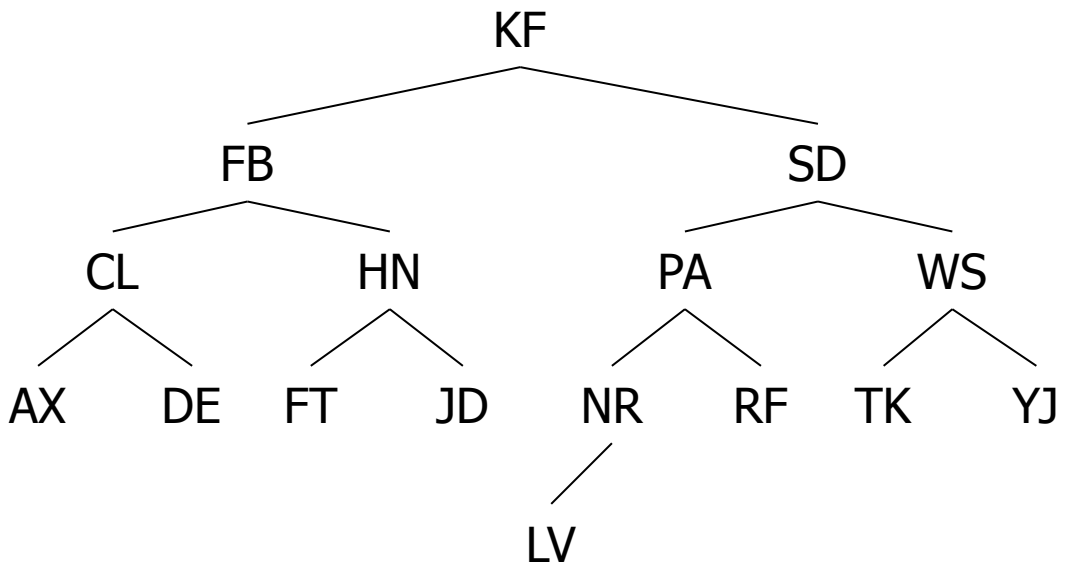
Vantagens

- Ordem lógica dos registros <> ordem física no arquivo
  - Ordem lógica: dada por ponteiros **esq** e **dir**
  - Registros não precisam estar fisicamente ordenados

- Inserção de uma nova chave no arquivo
  - É necessário saber onde inserir
  - Busca pelo registro é necessária, mas reorganização do arquivo não

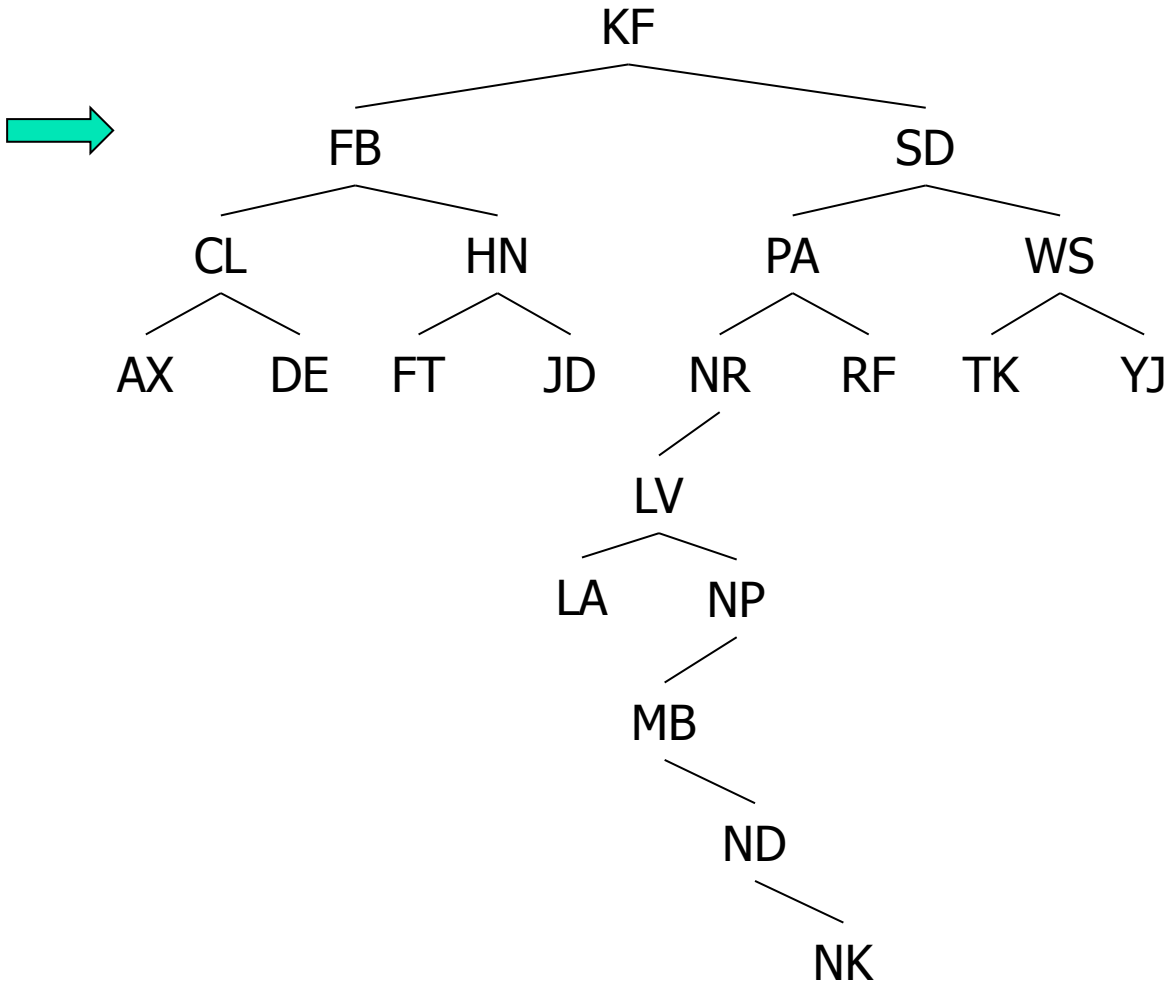
Inserção de chave

Inserção da chave LV



Problema: desbalanceamento

Inserção das chaves NP, MB, TM, LA, UF, ND, TS e NK

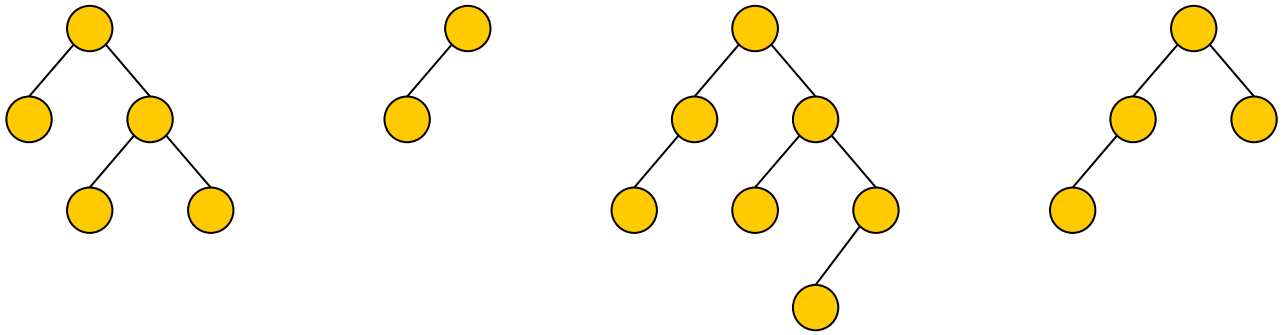


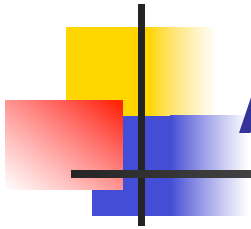
Ideia

Reorganizar os nós da árvore à medida que são inseridas novas entradas, mantendo uma estrutura de árvore quase ótima

Árvores-AVL

- Diferença limitada entre níveis
  - Tradicionalmente, 1 nível de diferença, no máximo
    - Procedimentos específicos de inserção e remoção

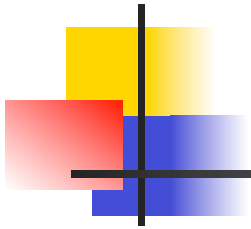




Árvores-AVL

- Características importantes
  - estabelecendo uma diferença máxima de altura, garante-se um desempenho mínimo (não menor do que) na pesquisa
  - manutenção da árvore envolve o uso de uma possível rotação, confinada a uma única área local da árvore (efeitos locais)
    - rotação mais complexa requer 5 rearranjos de ponteiros

Comparação de Desempenho: 1.000.000 de entradas

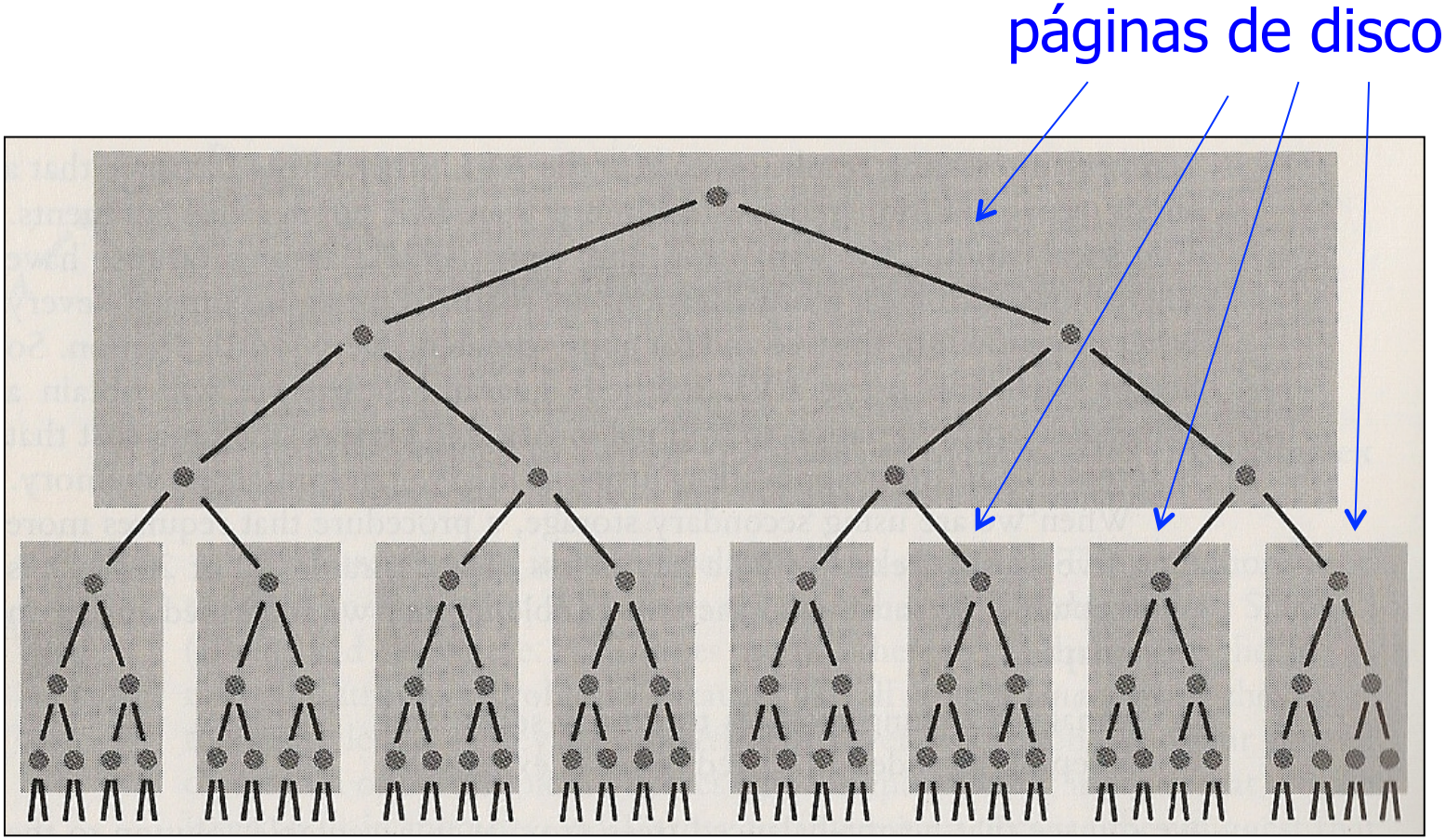


- Pesquisa binária em arquivo de índice ordenado
  - $\log_2(n) + 1$ = 20 acessos a disco
- Arquivo de índice na forma de uma árvore binária
  - completamente balanceada
    - $\log_2(n+1)$ = 20 acessos a disco
  - completamente desbalanceada
    - n = 1.000.000 acessos a disco
- Arquivo de índice na forma de uma árvore-AVL
  - 1,44 $\log_2(n+2)$ = acessos a disco

Ideia

Utilizar o conceito de paginação, de forma que vários registros sejam recuperados do disco simultaneamente, diminuindo a quantidade de acessos a disco necessários

Árvores Binárias Paginadas (*Paged Binary Trees*)

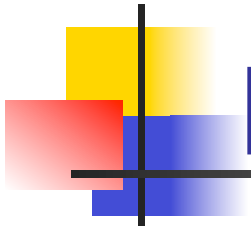


Árvores Binárias Paginadas (*Paged Binary Trees*)

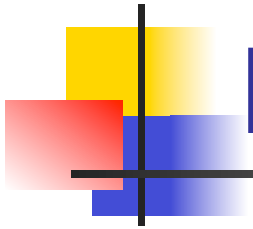
no exemplo corrente ...

- Cada página de disco
  - contém 7 nós (7 chaves de busca)
  - pode subdividir-se em até 8 novas páginas
- Localização de uma chave de busca
  - 1 nível – 1 página de disco – 7 chaves
  - 2 níveis – 9 páginas de disco – 63 chaves
  - 3 níveis – 73 páginas de disco – 511 chaves
  - 4 níveis – 585 páginas de disco – 4.095 chaves
  - ...

Eficiência da árvore binária paginada: situação mais real



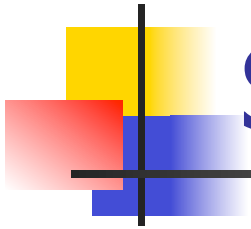
- Cada página
  - ocupa 8KB
  - armazena 511 chaves
  - contém uma árvore completamente balanceada
- Árvore com 3 níveis
  - pode armazenar 134.217.727 chaves
  - permite que qualquer uma dessas chaves seja recuperada em, no máximo, 3 acessos a disco
- Desempenho
  - $\log_{K+1}(n+1)$, onde n é o número de chaves e k o número de chaves por página de disco



Desvantagens

- Maior tempo na transmissão de dados
  - cada acesso a uma página de disco requer a transmissão de uma grande quantidade de dados, muitos dos quais não serão usados
  - *ok, pode-se pagar o preço*
- Manutenção da organização da árvore
  - devido a cada operação de inserção e remoção de chaves no índice
  - *problema com a construção top-down*

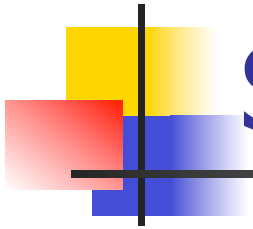
Construção Top-Down Situação 1 (*bulk-loading*)



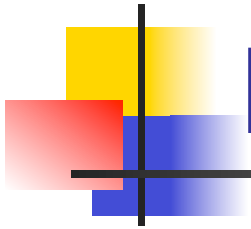
- Conhecimento de todo o conjunto de chaves antes da construção da árvore
- Passos
  - ordenação das chaves
  - construção da árvore com base na ordenação
- Característica
  - *middle key*: chave do nó raiz, para garantir o balanceamento da árvore

construção de uma estrutura muito boa

Construção Top-Down Situação 2 (*inserção 1 a 1*)

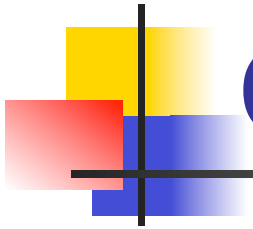


- Chaves são recebidas aleatoriamente e inseridas na árvore imediatamente
- Dificuldade
  - chegada de chaves inapropriadas
    - chaves adjacentes em sequencia
    - chaves do início do conjunto de chaves
- Chaves inapropriadas
  - não representam boas chaves de divisão da estrutura
  - podem causar desbalanceamento



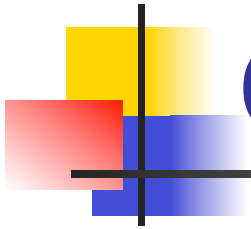
Desbalanceamento

- Características da rotação
  - a rotação de chaves deve preferencialmente ocorrer dentro de uma página para manter a subárvore balanceada
  - rotação entre páginas pode conduzir a muitos acessos a disco
- Desvantagens de chaves inapropriadas
  - dificuldade de se realizar a rotação de modo a realizar apenas efeitos locais em uma página
    - necessidade de rearranjar muitas páginas



Conclusão

- Agrupar chaves em uma página é uma boa solução :-)
  - reduz a quantidade de seeks realizados
- Abordagem top-down não permite escolher as melhores chaves
  - produção de uma estrutura desbalanceada que requer a rotação :-(



Questões em aberto

- Escolher boas chaves separadoras
  - chaves que dividem o conjunto de outras chaves de forma mais ou menos uniforme
- Evitar agrupamentos de chaves extremas
  - chaves que não deveriam estar na mesma página
- Garantir uma boa taxa de ocupação das páginas
  - cada página deve possuir pelo menos um número mínimo de chaves

Solução: Árvore-B