Árvores

- **Árvore binária**
  Árvore onde cada nó tem no máximo dois filhos.
- **Árvore Binária de Busca**
  - **Árvore balanceada**
    →as sub-árvores à esquerda e à direita possuem a mesma altura
    → ou, pode-se utilizar um critério mais brando permitindo que a diferença de altura entre as sub-arvores seja no máximo 1.
  - **Balanceamento estático:** construir uma nova versão da árvore binária de busca, reorganizando-a
  - **Balanceamento dinâmico** → Árvore AVL
- **Árvore B**
  - é uma árvore n-ária, isto é, cada nodo pode ter n filhos, sendo que em geral este valor de n é escolhido de tal maneira a otimizar a blocagem física do arquivo de índices.
  - Método de armazenamento e recuperação em sistemas de arquivos grandes, que fornece acesso rápido aos dados com baixo custo

Árvores binárias de busca (ABB)

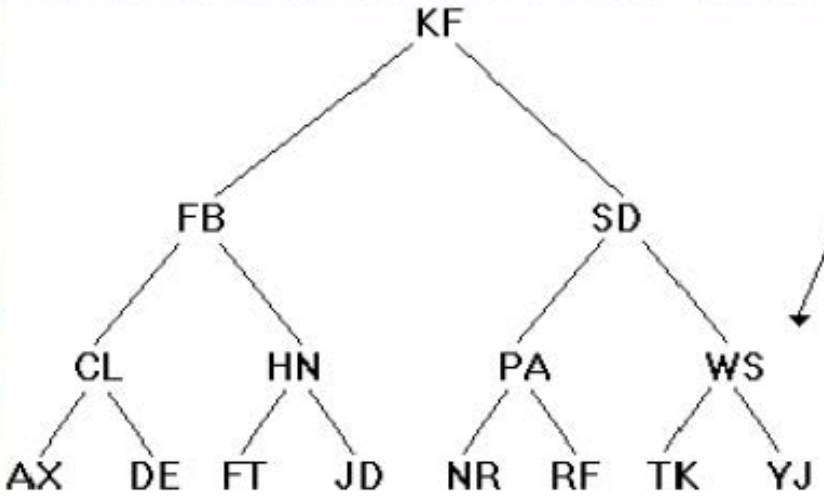
O Problema da manutenção de um índice em disco é que o acesso é muito lento, mesmo quando se utiliza Busca Binária

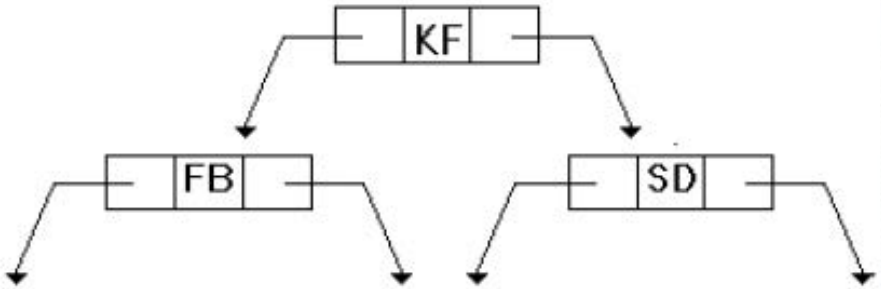
N(chaves)	$Log_2^{(N+1)}$
15	4
1.000	~10
100.000	~ 17
1.000.000	~ 20

- **Pode ficar muito caro manter um índice ordenado de forma a permitir busca binária. É necessário um método no qual a inserção e a eliminação de registros tenha apenas efeitos locais, isto é, não exija a reorganização total do índice.**

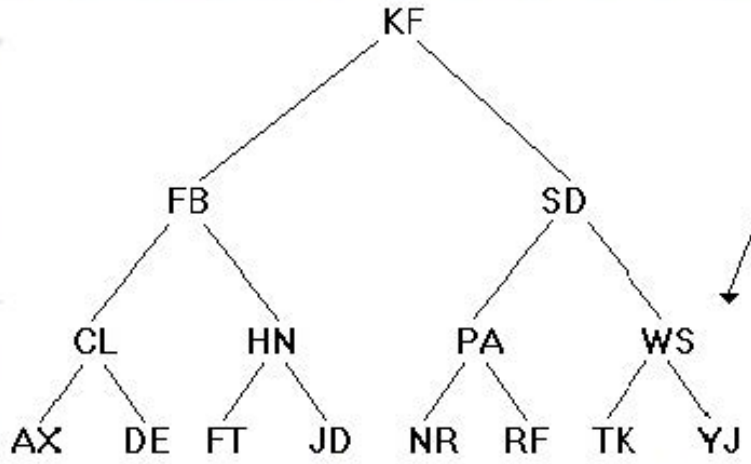
Solução por Árvores binárias de busca (ABB)

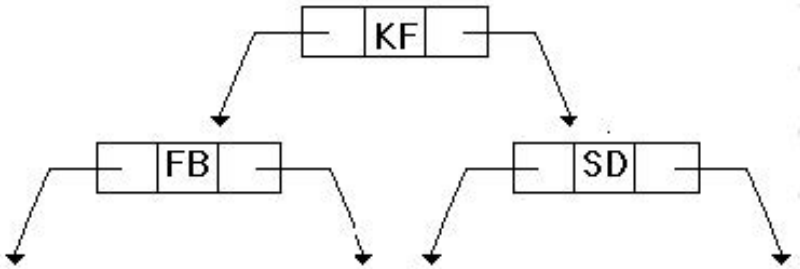
- os registros são mantidos num arquivo, e ponteiros (esq e dir) indicam onde estão os registros filhos.
- Esta estrutura pode ser mantida em memória secundária: os ponteiros para os filhos dariam o RRN das entradas correspondentes aos filhos.





Árvores binárias de busca (ABB)





Raiz = 9

	key	filho esq.	filho dir.
0	FB	10	8
1	JD		
2	RF		
3	SD	6	15
4	AX		
5	YJ		
6	PA	11	2
7	FT		

	key	filho esq.	filho dir.
8	HN	7	1
9	KF	0	3
10	CL	4	12
11	NR		
12	DE		
13	WS	14	5
14	TK		

- A ordem lógica dos registros não está associada à ordem física no arquivo.
- O arquivo físico do índice não precisa ser mantido ordenado→a seqüência física dos registros no arquivo é irrelevante. Para recuperar utiliza-se dos campos **esq** e **dir**.
- Se acrescentarmos uma nova chave ao arquivo é necessário saber onde inserir esta chave na árvore, de modo a mantê-la uma ABB

Solução por árvores-AVL

- **A eficiência do uso de árvores binárias de busca exige que estas sejam mantidas balanceadas:**
  - ➔ **Isso implica no uso de árvores AVL,**
  - ➔ **e algoritmos associados para inserção e eliminação de registros.**
- **Numa árvore AVL, o número máximo de comparações para localizar uma chave em uma árvore com N chaves é**
  - **igual à $1.44 * \log_2 (N+2)$.**
  - **Portanto, dadas 1.000.000 de chaves** a busca poderia percorrer até 28 níveis (**$1.44 \log_2 (1.000.000+2)$**).

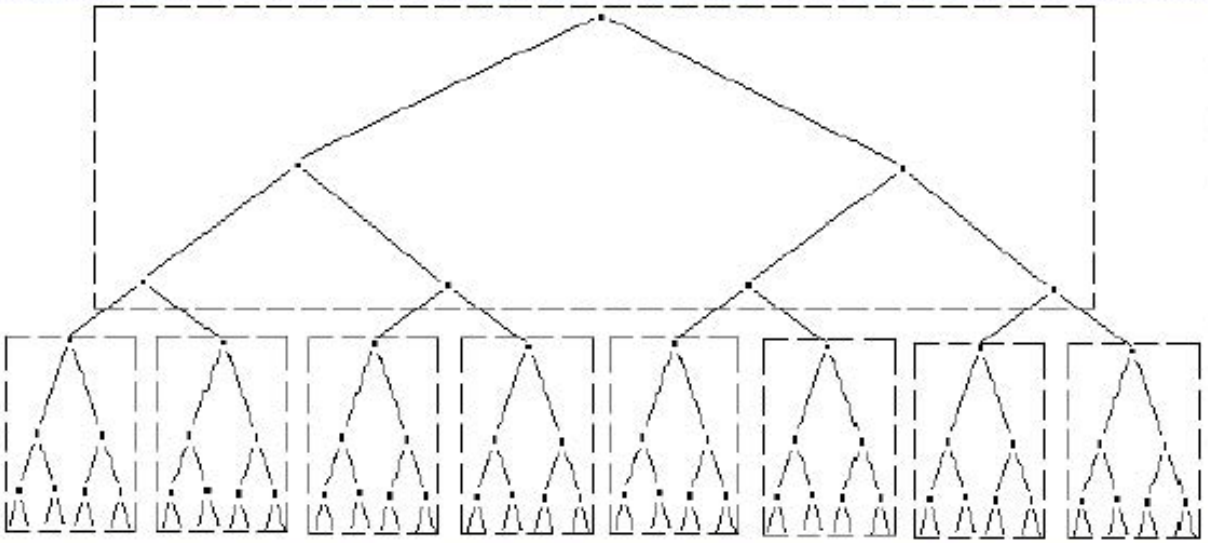
Solução por árvores-AVL

- Se as chaves estão em memória secundária, qualquer procedimento que exija mais do que 5 ou 6 acessos para localizar uma chave é altamente indesejável
  - 28 níveis da árvore AVL→ 28 *seeks* é inaceitável.
- Árvores balanceadas
  - são uma boa alternativa se considerarmos o problema da ordenação, pois não requerem a ordenação do índice e sua reorganização sempre que houver nova inserção.
  - Por outro lado, não resolvem o problema no número excessivo de acessos a disco.

Solução por Árvores Binárias Paginadas (*Paged Binary Trees*)

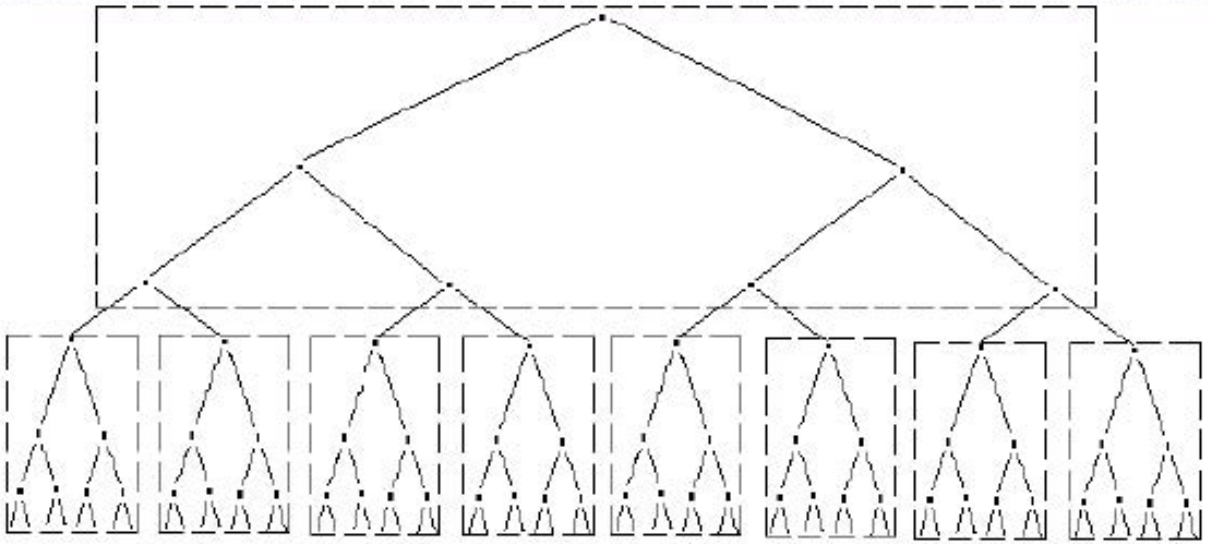
- A busca por uma posição específica do disco é muito lenta, mas uma vez encontrada a posição, pode-se ler uma grande quantidade registros seqüencialmente a um custo relativamente pequeno.
- Esta combinação de busca (*seek*) lenta e transferência rápida sugere a noção de página.
  - sistema "paginado“ →uma vez realizado um *seek*, que consome um tempo considerável, toda os registros em uma mesma "página" do arquivo são lidos.
  - Esta página pode conter um número bastante grande de registros, e se o próximo registro a ser recuperado estiver na mesma página, será economizado um acesso ao disco.

Solução por Árvores Binárias Paginadas (*Paged Binary Trees*)



- A divisão de uma árvore binária em páginas é ilustrada na figura acima.
- Nessa árvore de 9 páginas, quaisquer dos 63 registros podem ser acessados em, no máximo, 2 acessos.
- Se a árvore é estendida com um nível de paginação adicional, adicionamos 64 novas páginas, e poderemos encontrar qualquer um das 511 chaves armazenadas com apenas 3 *seeks*

Solução por Árvores Binárias Paginadas (*Paged Binary Trees*)

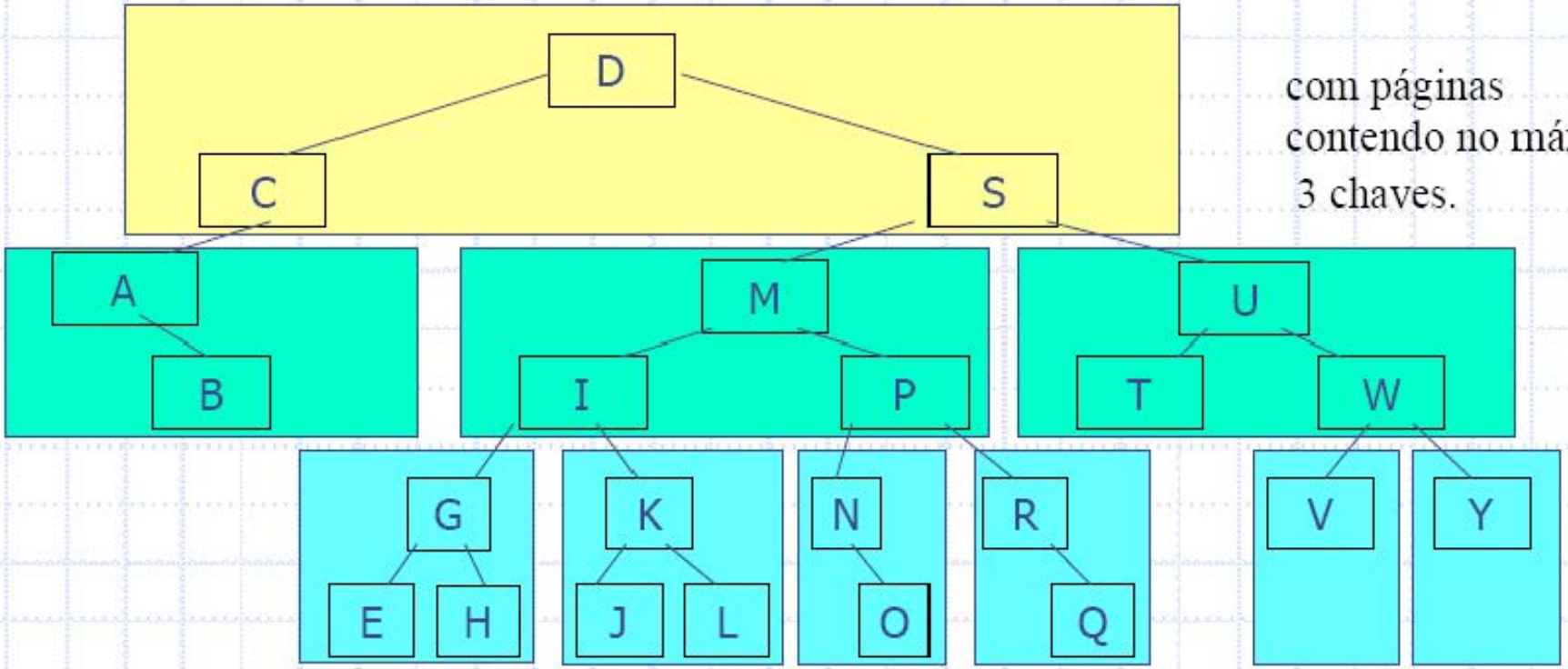


- Um exemplo típico de tamanho de página ocupa 8KB, permitindo a armazenagem de 511 pares chave-referência;
- Se cada página contiver uma árvore completa perfeitamente balanceada, então a árvore toda pode armazenar um total de 134.217.727 chaves
- Neste caso qualquer chave pode ser acessada em, no máximo, 3 acessos ao disco.

Exemplo

- conjunto de dados:

  C S D T A M P I B W N G U R K E H O L J Y Q Z F X V



com páginas contendo no máx 3 chaves.

- Nesse exemplo C e D não deveriam estar no topo
- A árvore construída dessa forma não está balanceada.

Solução por Árvores Binárias Paginadas

Problema da construção *Top-Down* de árvores paginadas

Questões:

- **Como garantir que as chaves da raiz são boas separadoras ?**
- **Como impedir o agrupamento de chaves que não deveriam estar na mesma página (C, D e S).**
- **Como garantir que cada página contenha um número mínimo de chaves ?**

Solução

Utilização de árvores B

→ árvore de busca balanceada, onde ao se inserir uma chave ela é colocada sempre numa folha

→ por meio de sub-divisão (split) e promoção (promote), a árvore fica sempre balanceada

Árvores-B
Construção *Bottom-Up*

- Cada página é formada por um seqüência ordenada de chaves e um conjunto de ponteiros.
- Não existe uma árvore explícita dentro de uma página.
- O número de ponteiros em um nó excede o número de chaves em 1.
- O número máximo de ponteiros que podem ser armazenados em um nó é a ordem da árvore.
- O número máximo de ponteiros é igual ao número máximo de descendentes de um nó.
- Exemplo: uma árvore-B de ordem 8 possui nós com, no máximo, 7 chaves e 8 filhos.
- Os nós folha não possuem filhos, e seus ponteiros são nulos.

Árvores-B

Sub-divisão(*Splitting*) e Promoção (*Promoting*)

- Seja a seguinte página inicial de uma árvore-B de ordem 8, que armazena 7 chaves.

*	A	*	B	*	C	*	D	*	E	*	F	*	G	*

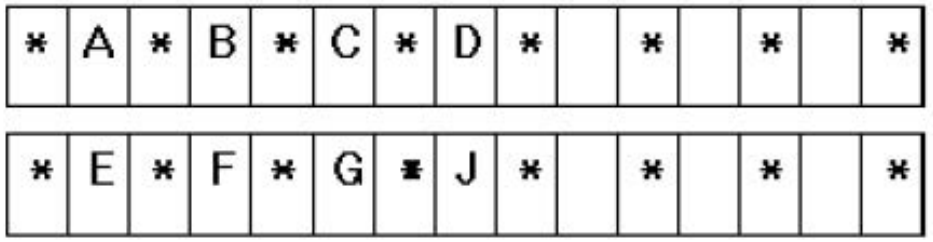
- Observe que, em uma situação real, além das chaves e ponteiros armazena-se outras informações associadas às chaves, como uma referência para a posição do registro associado à chave em um arquivo de dados.
- Esta folha que, coincidentemente, é também a raiz da árvore, está cheia.
- Como inserir uma nova chave, digamos J ?

Árvores-B

Sub-divisão(*Splitting*) e Promoção (*Promoting*)

Sub-divisão

Sub-dividimos (*split*) o nó folha em dois nós folhas, distribuindo as chaves igualmente entre os nós.



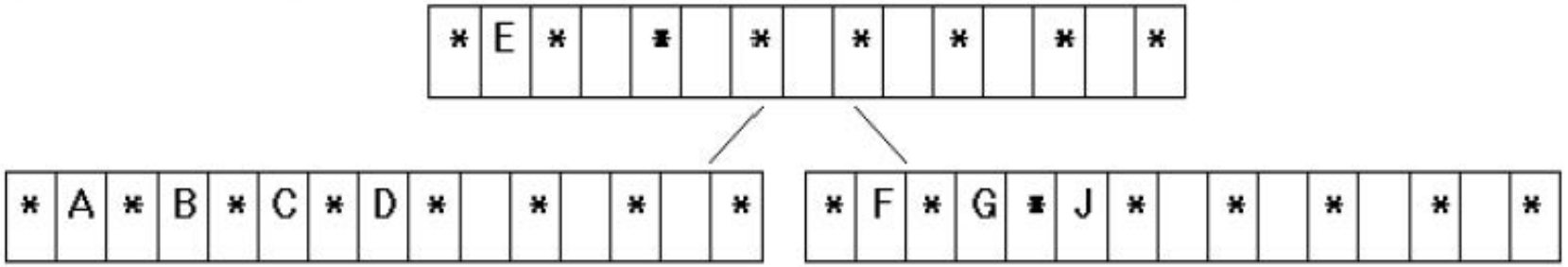
- Temos agora duas folhas: precisamos criar uma nova raiz.
- Fazemos isso "promovendo", ou "subindo", uma das chaves que estão nos limites de separação das folhas.

Árvores-B

Sub-divisão(*Splitting*) e Promoção (*Promoting*)

Promoção

Nesse caso, "promovemos" a chave E para a raiz:

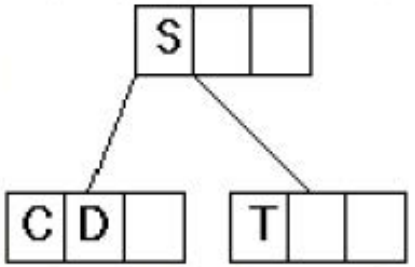


Exemplo de inserção

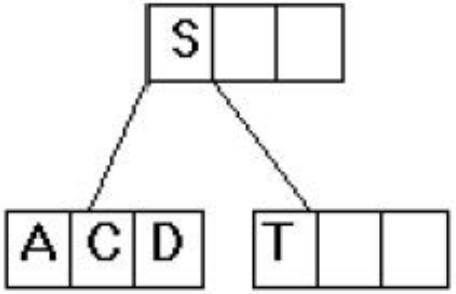
Suponha que o conjunto de dados consiste em letras do alfabeto, que serão fornecidas na seguinte ordem: C S D T A M P I B W N G U R K E H O L J Y Q Z F X V

Inserção de C, S e D dentro da página inicial

Inserção de T força o sub-divisão e a promoção de S.

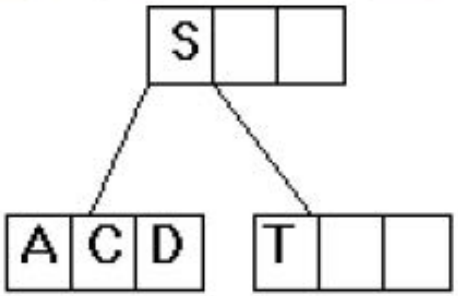


Adição de A.

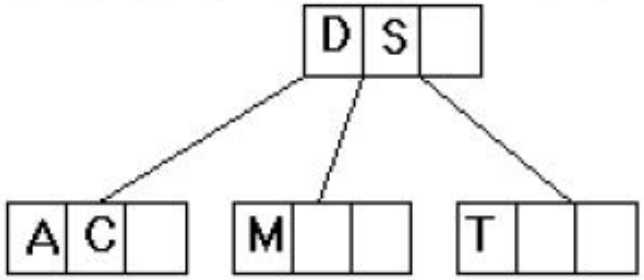


Exemplo

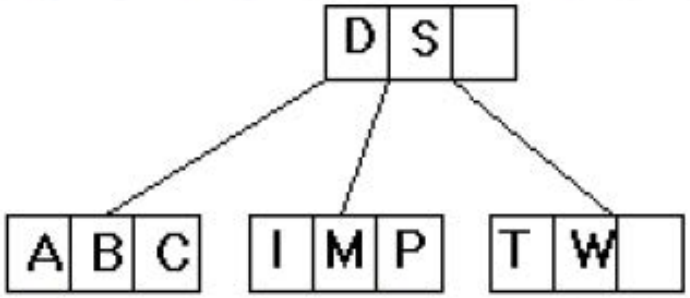
conjunto de dados : C S D T A M P I B W N G U R K E H O L J Y Q Z F X V



Inserção de M força a sub-divisão e a promoção de D.

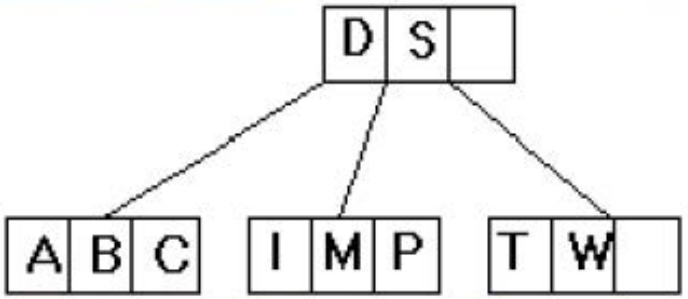


Inserção de P, I, B,e W nas páginas existentes.

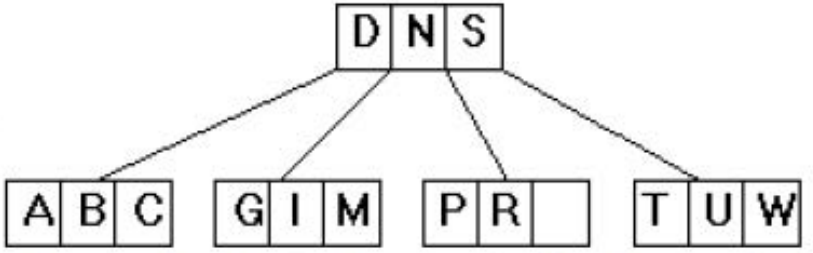


Exemplo

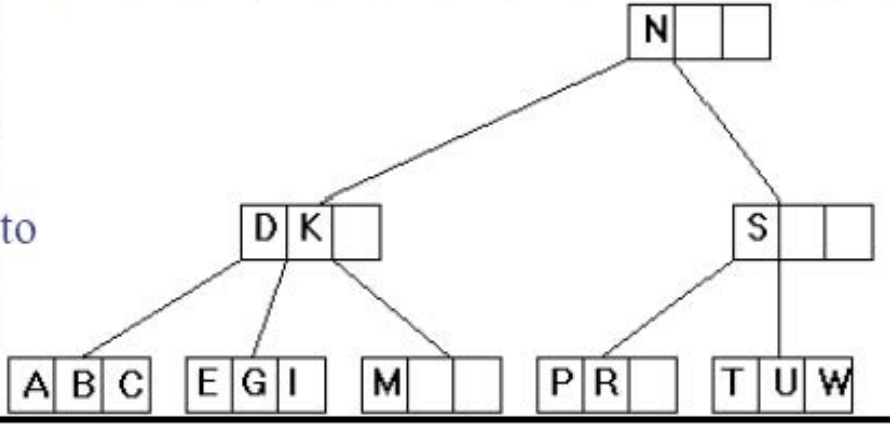
Conjunto de dados: C S D T A M P I B W **N G U R K E** H O L J Y Q Z F X V



Inserção de N, G, U e R precisa de sub-divisão e a promoção de N.
Obs: A B-Tree cresce para um ponto em que a sub-divisão da raiz é iminente.

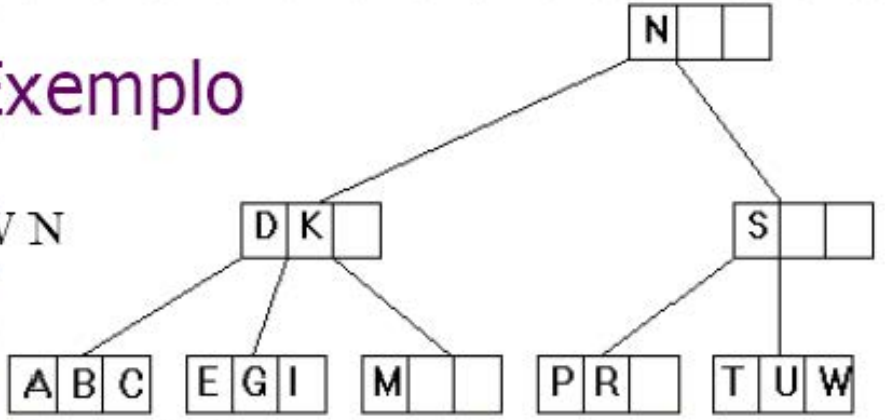


Inserção de K resulta na sub-divisão no nível folha, seguido pela promoção de k. Isto resulta na sub-divisão da raiz. N é promovido para ser a nova raiz e E é posto como nó folha.

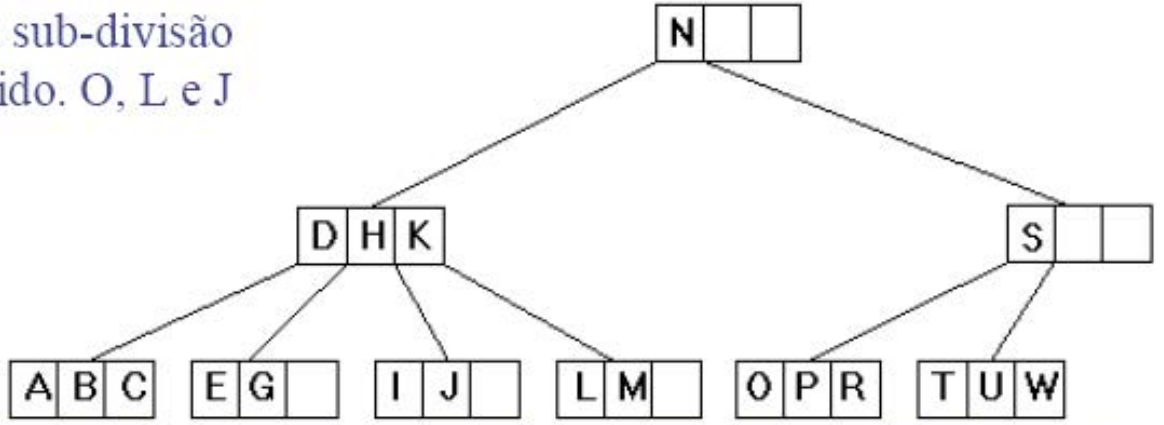


Exemplo

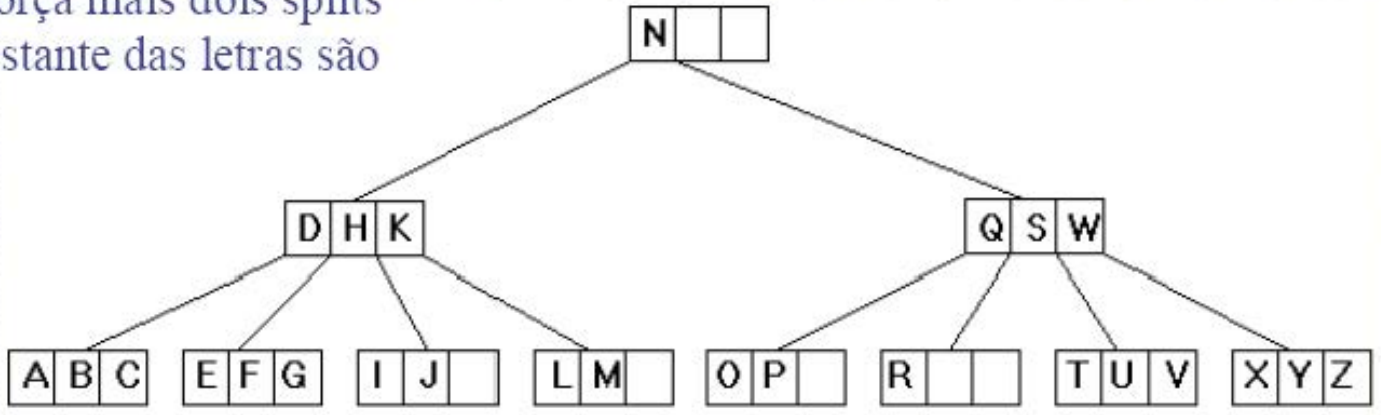
Conjunto de dados: C S D T A M P I B W N
G U R K E H O L J Y Q Z F X V



Inserção de H resulta em sub-divisão no nó folha. H é promovido. O, L e J são adicionados.

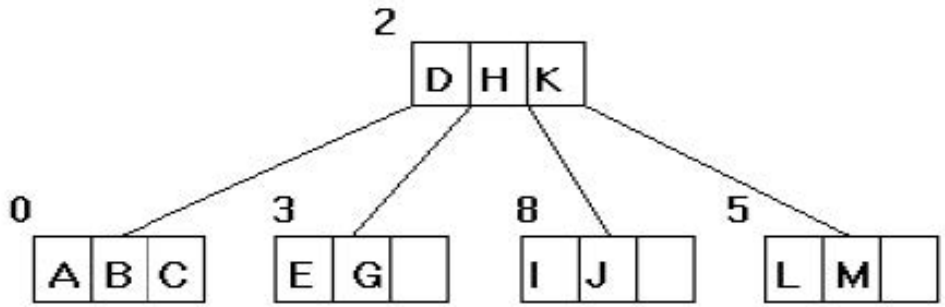


Inserção de Y e Q força mais dois splits nos nós folhas. O restante das letras são adicionados.



Árvores-B
Busca e Inserção

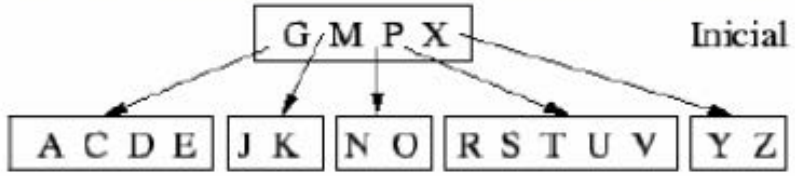
- Um exemplo de parte de uma árvore-B de ordem 4 é dado na figura abaixo. Um nó interno e 4 nós folha, são explicitados os RRN de cada página (o RRN é um número de página válido, e os ponteiros das folhas apontam para nil (que pode ser -1).
- O arquivo que contém a árvore-B é um arquivo com registros de tamanho fixo, sendo que cada registro contém uma página da árvore.



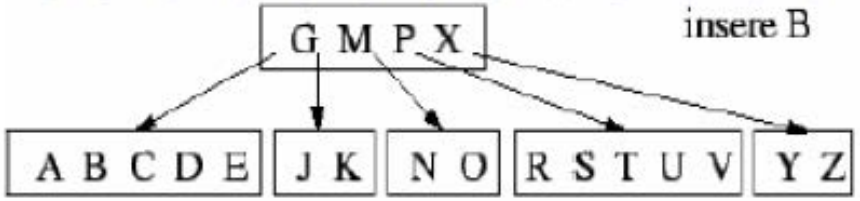
Conteúdo das páginas 2 e 3:

Árvores-B - Exercício: incluir novos elementos em uma árvore-B de ordem 6:

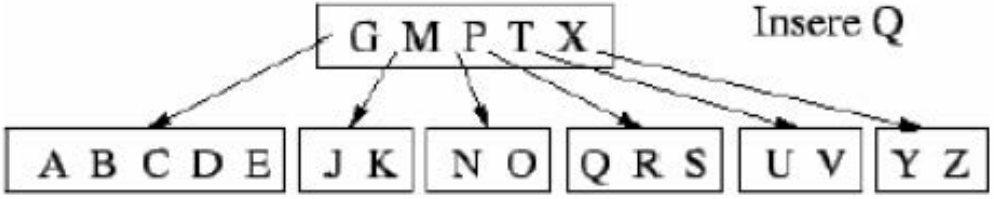
a) Inicial



a) Incluir a chave B



b) Incluir a chave Q



a) Incluir a chave F