INE5408
Estruturas de Dados

Árvores B

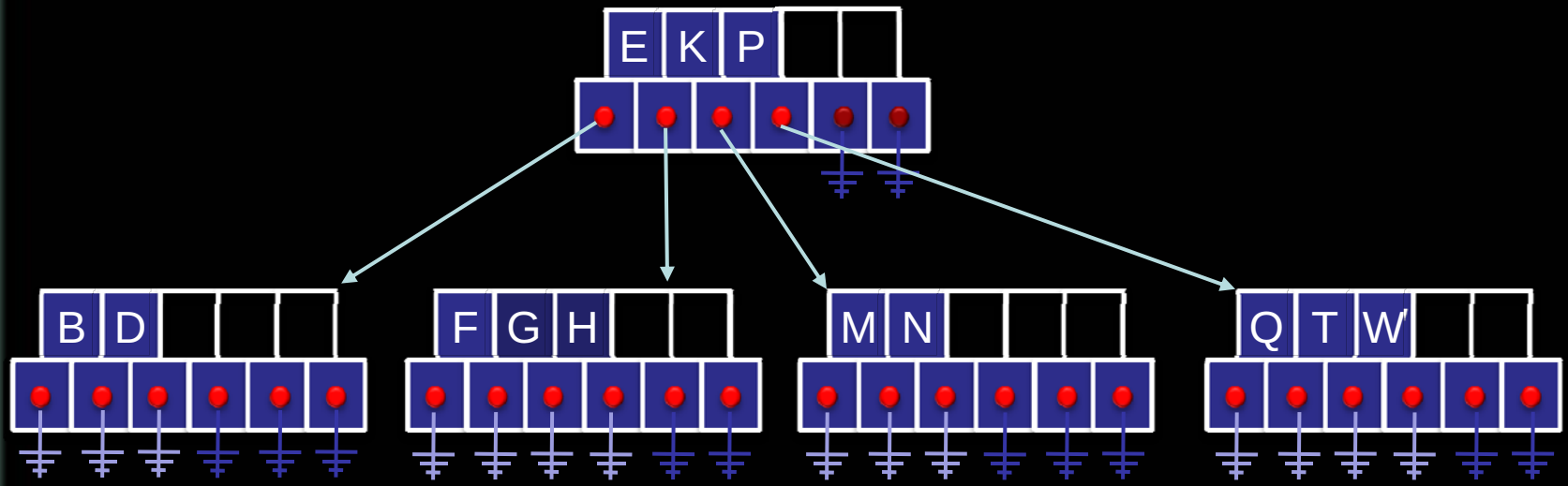
- Estrutura e peculiaridades
- Algoritmos de inserção, pesquisa e deleção

Definição

Árvores B são árvores de pesquisa balanceadas especialmente projetadas para a pesquisa de informação em discos magnéticos e outros meios de armazenamento secundário:

- minimizam o número de operações de movimentação de dados (escrita / leitura) numa pesquisa ou alteração;
- o grau de um nodo pode ser alto;
- podem ser consideradas como uma generalização natural das árvores de pesquisa binárias.

Exemplo Simples (ordem t=3)



Definição

Uma árvore B de ordem mínima t >= 3 é uma árvore com as seguintes propriedades:

- cada nodo possui os seguintes campos:

<NChaves,Folha,P1,<K1,PR1>,P2,<K2,PR2>,...,<Kq-1,PRq-1>,Pq>

- Descrição dos campos:
  - **NChaves – número de chaves armazenadas no nodo;**
  - **Folha – flag indicando se o nodo é folha ou não;**
  - **P_i – ponteiro para o nodo filho *i*;**
  - **K – chaves;**
  - **PR_i – ponteiro para o elemento de dados na memória ou para um registro em disco onde se encontra a chave K_i;**
  - **Nº de chaves q < 2t.**

Definição

Outras propriedades:

- as chaves estão sempre em ordem crescente;
- cada chave K_i separa as subárvores P_i e P_{i+1}.
  - as chaves da subárvore P_i são menores do que K_i e as chaves da subárvore P_{i+1} são maiores do que K_i;
- existem limites inferior e superior ao número de chaves em cada nodo:
  - cada nodo, exceto a raiz, contém no mínimo $q = t-1$ chaves (t filhos);
  - cada nodo tem no máximo $q = 2t-1$ chaves ($2t$ filhos);
- todos os nodos-folha estão no mesmo nível.

Inserção em uma árvore B

O algoritmo de inserção em uma árvore B é bastante simples:

- desça a árvore em busca da chave;
  - a inserção é sempre feita nas folhas;
- ao retornar da inserção:
  - verifique se o nodo está cheio (se contém 2t-1 chaves);
  - se o nodo estiver cheio:
    - dividir o nodo ao meio;
    - colocar a chave do meio no nodo pai (*se este também encher, repetir o processo recursivamente*).

Inserção em uma árvore B

```
insere_B(nodoB raiz, infoB dado)
   variaveis
      nodoB filho = nulo; /* inicializa p/folha */
   inicio
      SE raiz->folha ENTÃO
         /* insira na folha */
         insere_folha_B(raiz, dado);
      SENÃO
         /* continua descendo */
         filho <- selecionaRamoDescida_B(raiz, dado);
         insere_B(filho, dado);
      FIM SE
      /* retorna ajustando */
      SE nodo_cheio_B(filho) ENTÃO
         /* se nodo possuir 2t-1 chaves, divida,
            lembrando de passar a raiz que se altera */
         divide_nodo_B (raiz, filho);
      FIM SE
   fim
```

Exemplos de inserção

Para exemplificar o processo de inserção, será considerada uma árvore B vazia de ordem t=3

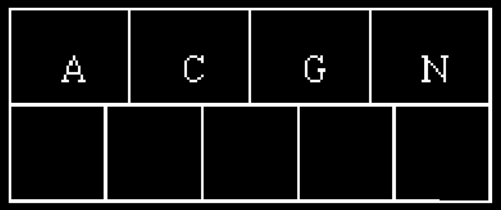
- nodo é dividido quando atingir 5 chaves.

Os elementos a serem inseridos, nessa ordem, são:

C N G A H E K Q M F W L T Z D P R X Y S

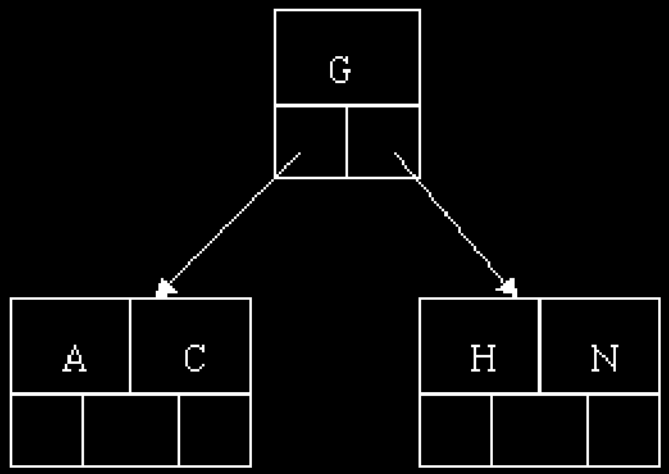
Inserção dos elementos C, N, G e A

Os elementos são inseridos no mesmo nodo de forma ordenada.



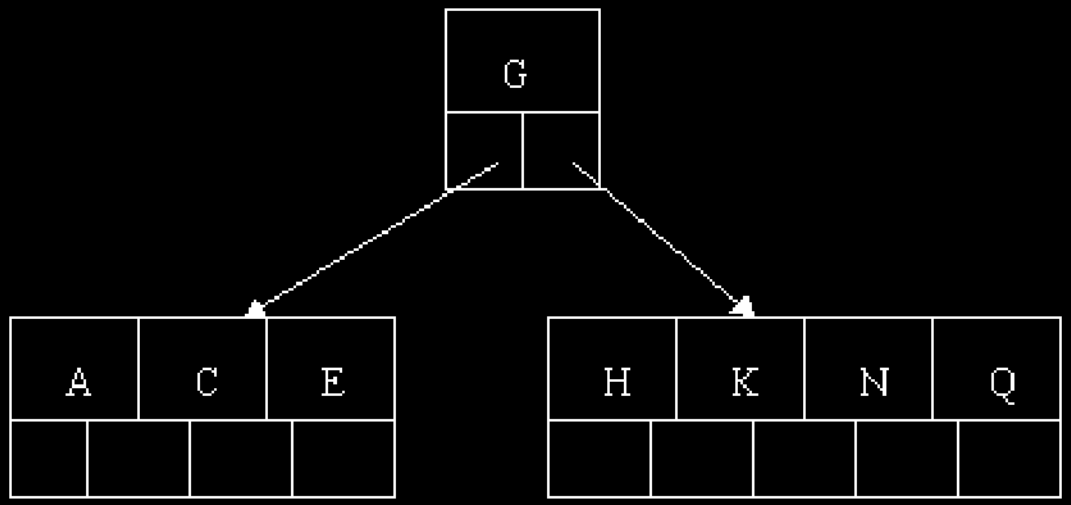
Inserção dos elementos C, N, G e A

Não há espaço disponível no nodo. Ocorre a divisão do nodo em 2, com a movimentação do item G (localizado no meio do conjunto de chaves) para um novo nodo raiz.



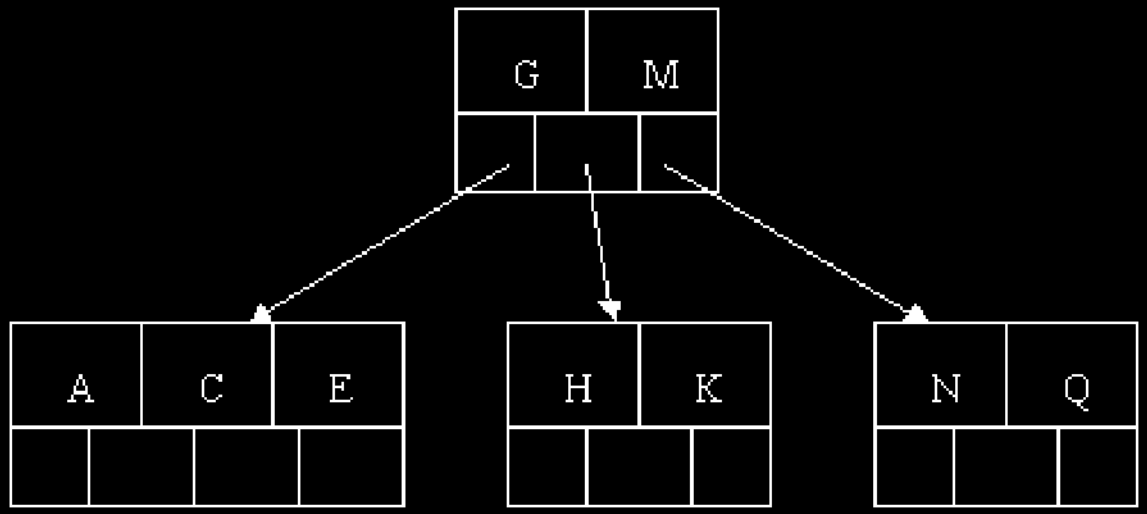
Inserção dos elementos E, K e Q

Os elementos são inseridos sem a necessidade de dividir os nodos existentes.



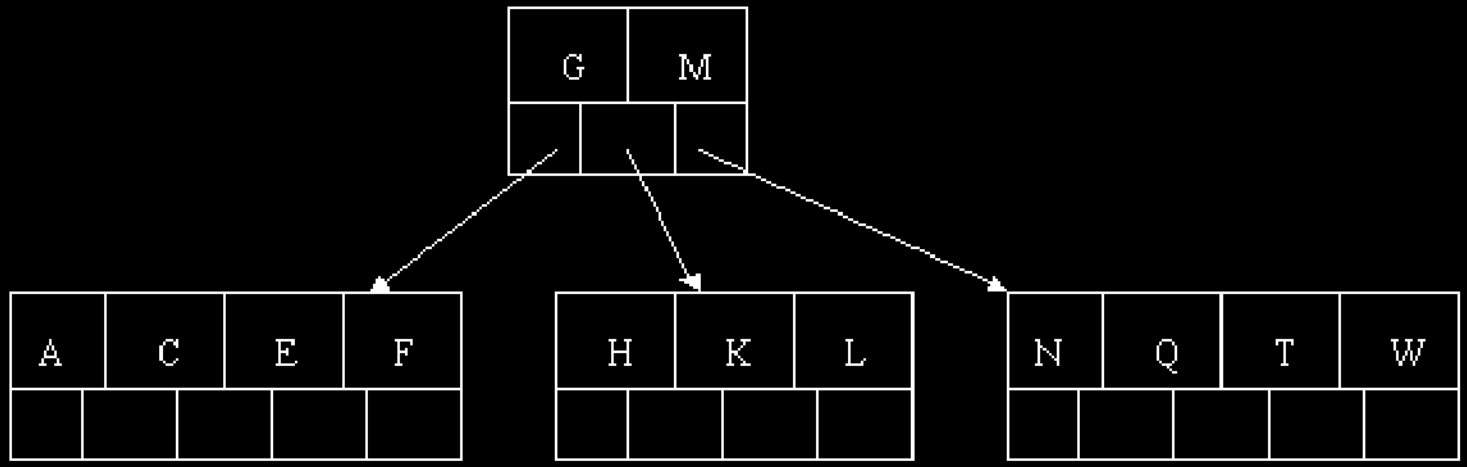
Inserção do elemento M

A inserção do elemento requer uma divisão do nodo. Por ser o elemento central, M é movido para o nodo pai.



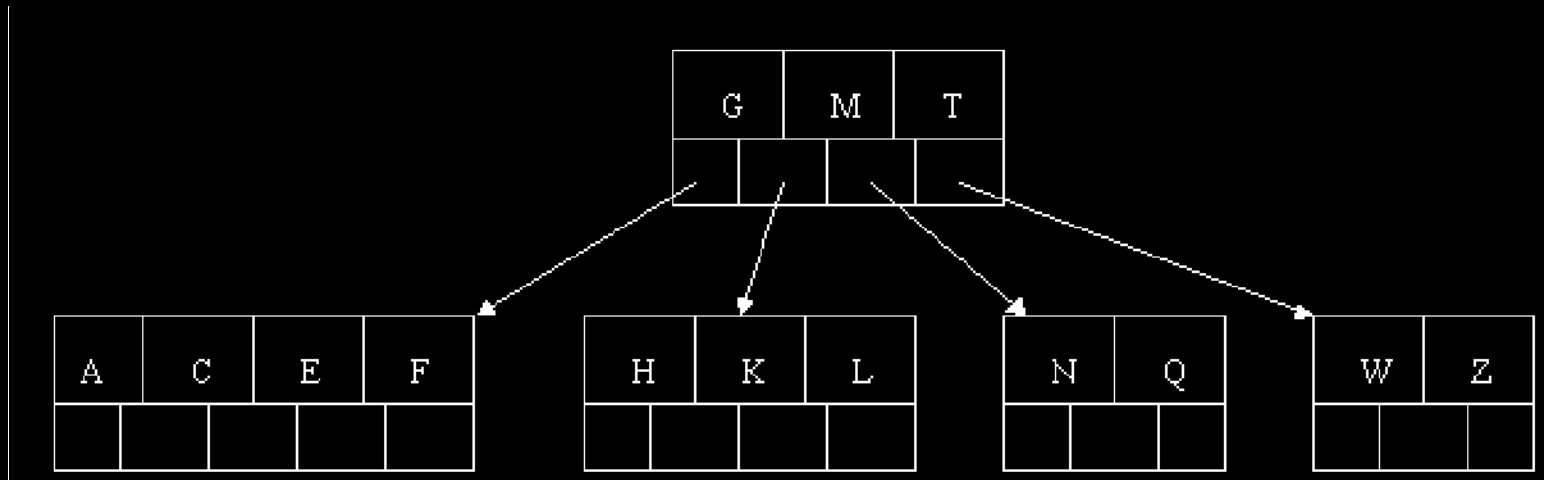
Inserção dos elementos F, W, L e T

Os elementos são inseridos sem a necessidade de dividir os nodos existentes.



Inserção do elemento Z

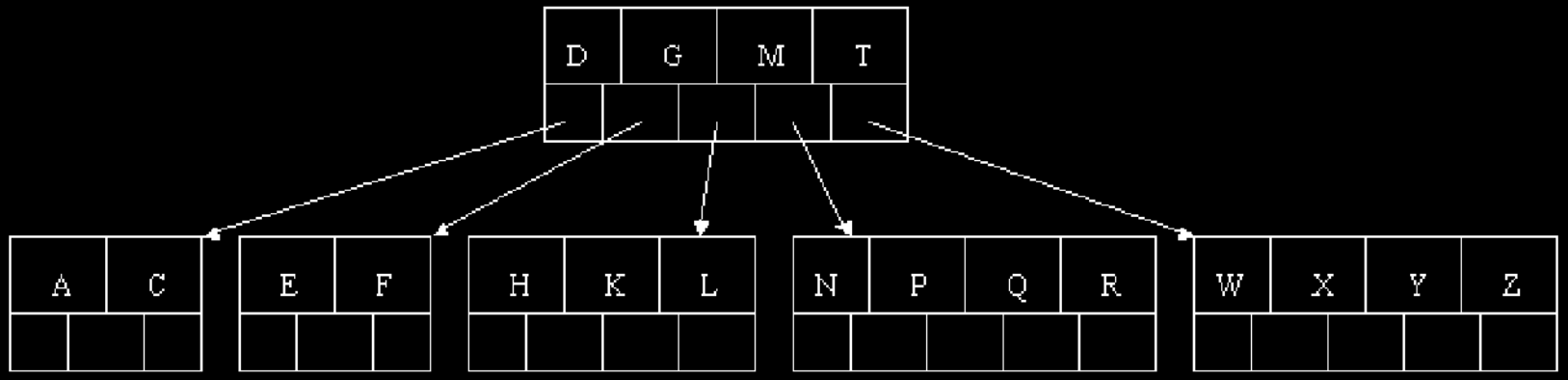
A inserção do elemento requer uma divisão do nodo. Por ser o elemento central, T é movido para o nodo pai.



Inserção dos elementos D, P, R, X e Y

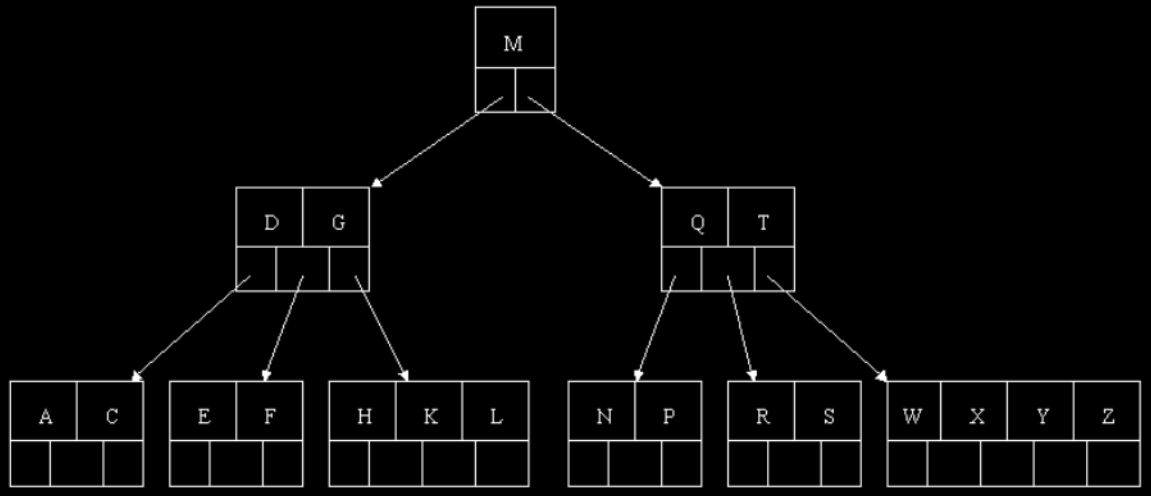
A inserção do elemento D requer uma divisão do nodo. Por ser o elemento central, D é movido para o nodo pai.

Em seguida, os elementos P, R, X e Y podem ser inseridos sem a necessidade de novas divisões.



Inserção do elemento S

A inserção do elemento S requer duas divisões. Por ser o elemento central, Q é movido para o nodo pai (que está completo). Em seguida, o elemento M (o elemento central do nodo pai) é movido para formar a nova raiz da árvore.



Deleção em uma árvore B

A deleção é a operação mais complexa em uma árvore B.

Passos gerais:

1. desça a árvore em busca da chave, deletando-a quando a encontrar;
2. verifique se haverá *underflow* (t-1). Se ocorrer, lembre-se que será necessário fundir o nodo com algum irmão, seguindo regras bem definidas.

Deleção em uma árvore B

Regras de Deleção (remover a chave K da subárbvore X):

Regra 1: se K está em X e X é folha, remova.

Deleção em uma árvore B

Regra 2: se K está em X e X é um nodo interno.

a) se a subárvore Y predecessora de X possui pelo menos t chaves:
- determine a chave predecessora K' em Y
- substitua K por K'
- recursivamente exclua K' da subárvore Y

b) se a subárvore Z sucessora de X possui pelo menos t chaves:
- determine a chave sucessora K' em Z
- substitua K por K'
- recursivamente exclua K' da subárvore Z

c) se as subárvores Y e Z possuem apenas t-1 chaves:
- funda K e todo o nodo Z em Y
- retire K e o apontador para Z de X
- libere o nodo Z
- recursivamente exclua K da subárvore Y

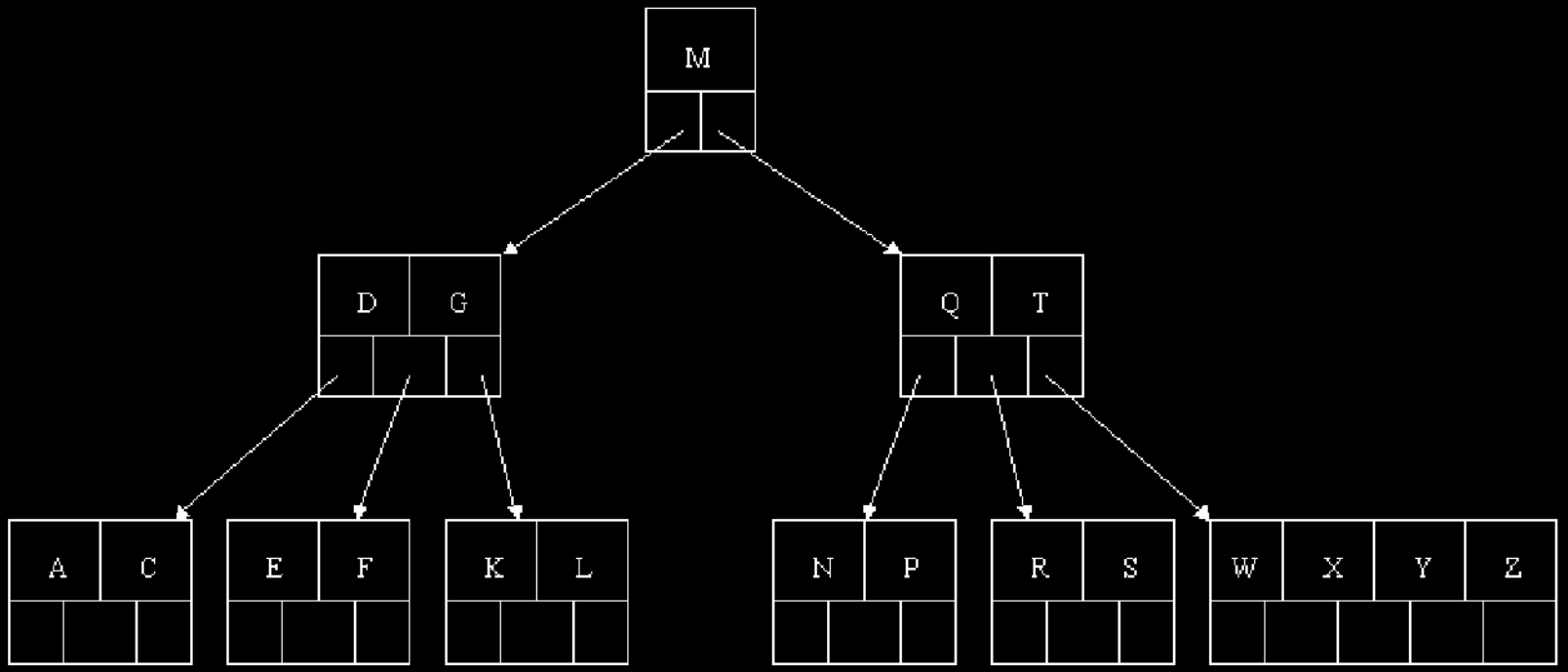
Deleção em uma árvore B

Regra 3: se K não está em X e X é um nodo interno.

1. determine a raiz $P_i[X]$ da subárvore onde ***K*** deve estar
2. exclua recursivamente ***K*** de $P_i[X]$
3. SE $P_i[X]$ tem apenas t-1 chaves:
   a) se um dos vizinhos de $P_i[X]$ tem pelo menos t chaves:
      - mova uma chave de X descendo para $P_i[X]$
      - mova uma chave do vizinho de $P_i[X]$ subindo para X
      - mova o apontador apropriado do vizinho para $P_i[X]$
   b) se $P_i[X]$ e seus vizinhos têm apenas t-1 chaves:
      - funda $P_i[X]$ com um dos seus vizinhos e com a chave K_i de X
      - remova o par K_i e P_i de X
      - libere o nodo apontado por Pr_i

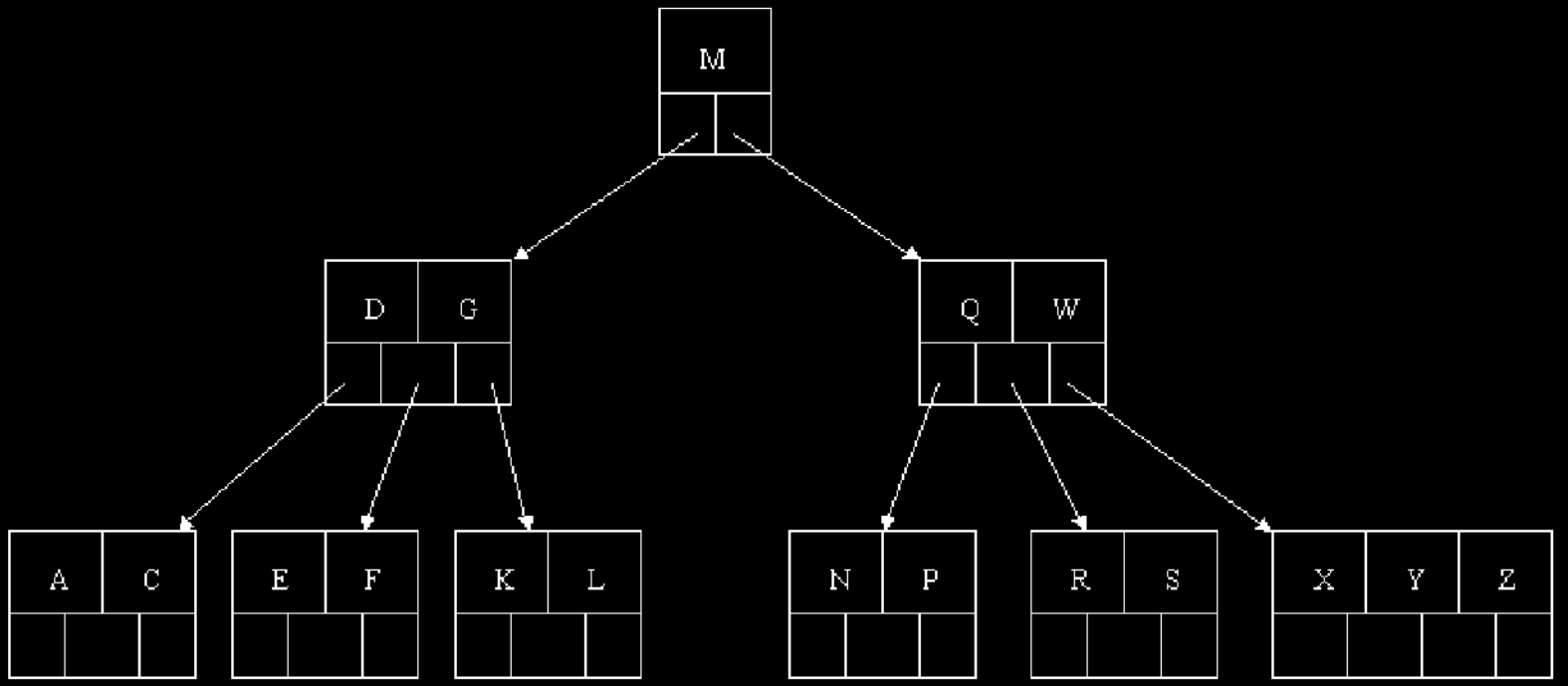
Deleção do elemento H

Aplicação da regra 1.



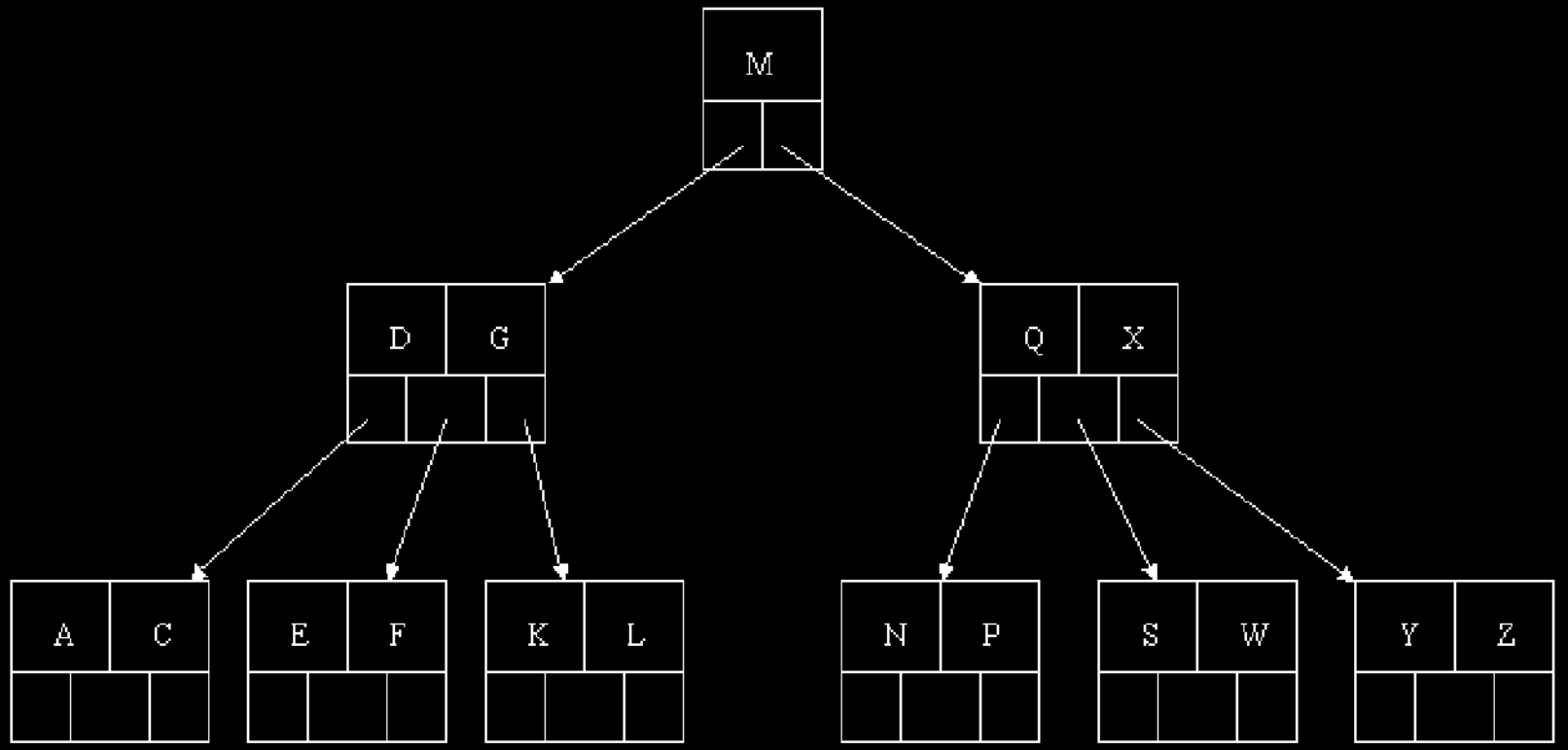
Deleção do elemento T

Aplicação da regra 2(b).



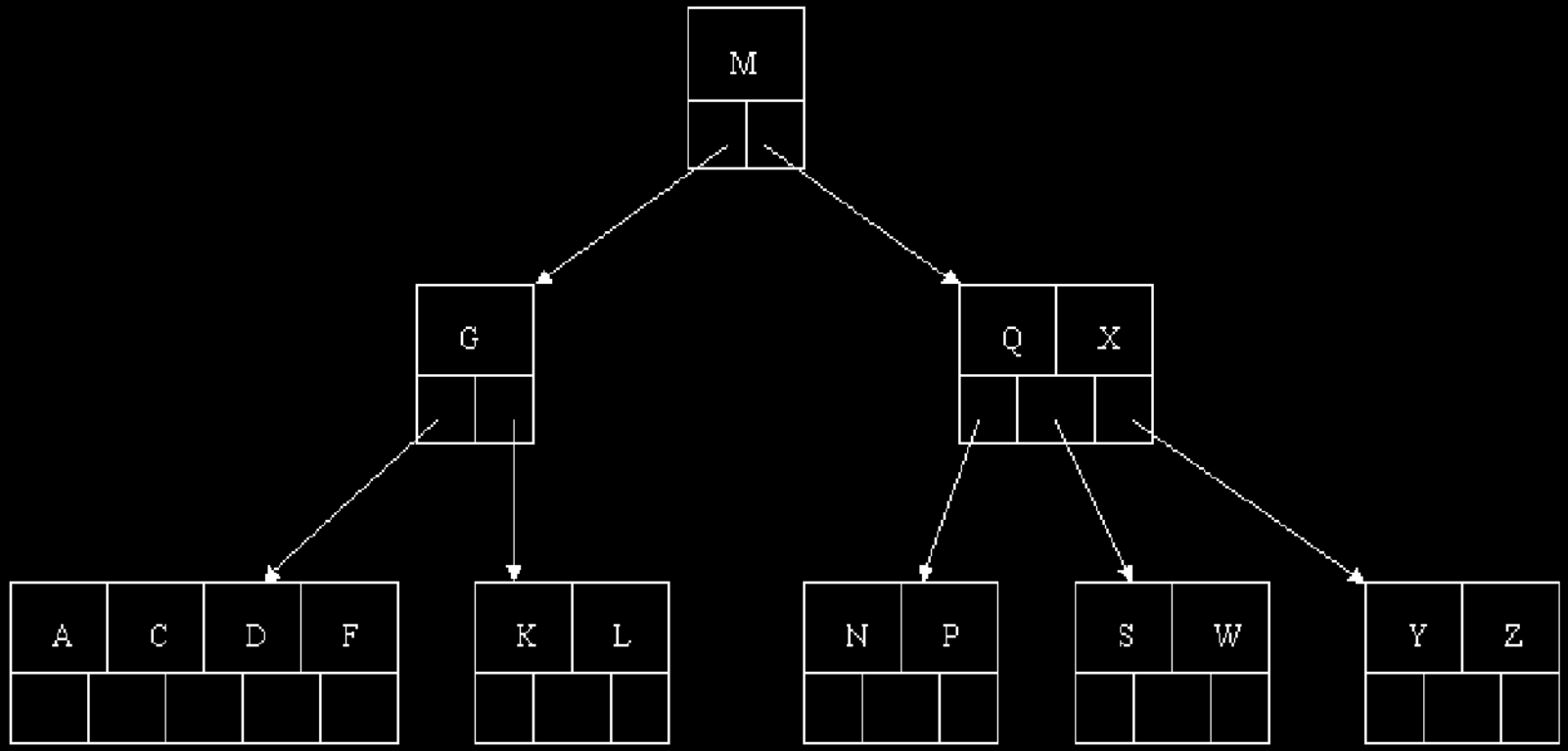
Deleção do elemento R

Aplicação da regra 3(a).



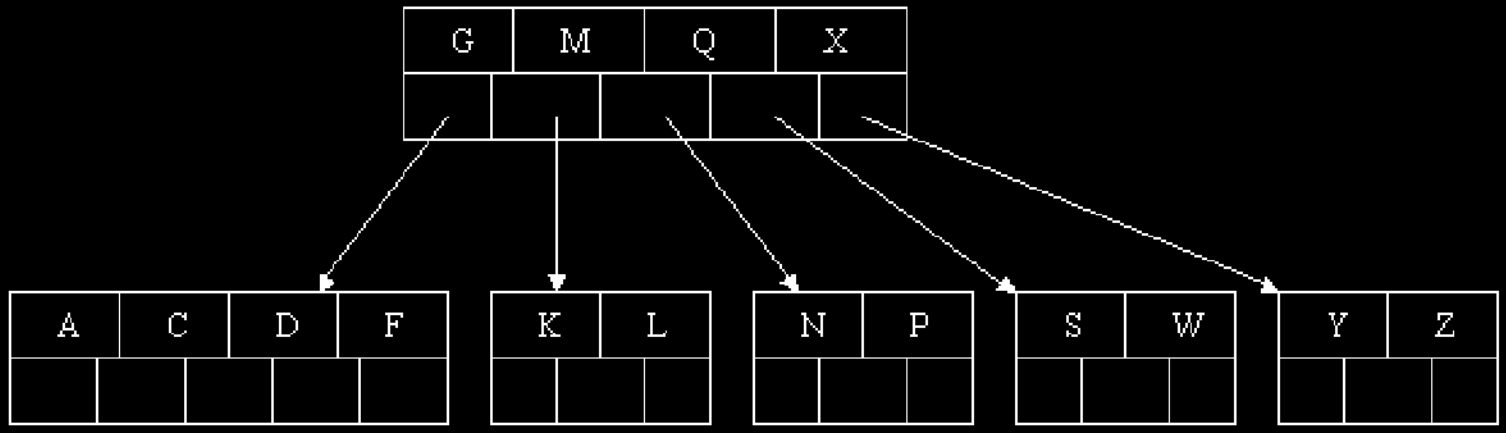
Deleção do elemento E (1)

Aplicação da regra 3(b).



Deleção do elemento E (2)

Aplicação da regra 3(b).



Algoritmo de Deleção em Árvore B

Verificação e ajuste de *underflow* realizado em apenas um lugar, ao retornar das chamadas recursivas.

Vantagens: Não importa se foi a deleção na folha causou underflow ou se há underflow em nodo interno, tratamos ambos em um só lugar.

Algoritmo de Deleção em Árvore B (1)

```
deleta_B(nodoB pai, nodoB nodo, infoB dado)
   variaveis
      nodoB filho;
   inicio
      SE contem_chave_B(nodo, dado) ENTÃO
         SE raiz->folha ENTÃO /* caso 1 */
            deleta_folha_B(nodo, dado);
         SENÃO /* caso 2 */
             deleta_interno_B(pai, nodo, dado);
         FIM SE
      SENÃO /* desce recursivamente */
         filho <- selecionaRamoDescida_B(nodo, dado);
         deleta_B(nodo, filho, dado);
      FIM SE
      /* caso 3 depois de fazer tudo */
      ajusta_nodo_B(pai, nodo, dado);
   fim
```

Observe que se **nodo** é a raiz, então **pai** é nulo. Isso deve ser levado em consideração no ajuste e na deleção interna.

Algoritmo de Deleção em Árvore B (2)

```
deleta_B(nodoB pai, nodoB nodo, infoB dado)
   variaveis
      nodoB filho;
   inicio
      SE contem_chave_B(nodo, dado) ENTÃO
         SE raiz->folha ENTÃO /* caso 1 */
            deleta_folha_B(nodo, dado);
         SENÃO /* caso 2 */
             deleta_interno_B(pai, nodo, dado);
         FIM SE
      SENÃO /* desce recursivamente */
         filho <- selecionaRamoDescida_B(nodo, dado);
         deleta_B(nodo, filho, dado);
         /* caso 3 depois de fazer tudo */
         ajusta_nodo_B(nodo, filho, dado);
      FIM SE
   fim
```

Como a raiz não possui underflow, podemos tratar o ajuste aqui. Se K estiver na raiz, não haverá ajuste.