UFSC-CTC-INE

INE5384 - Estruturas de Dados

Métodos de Pesquisa de Dados (II)

Prof. Ronaldo S. Mello

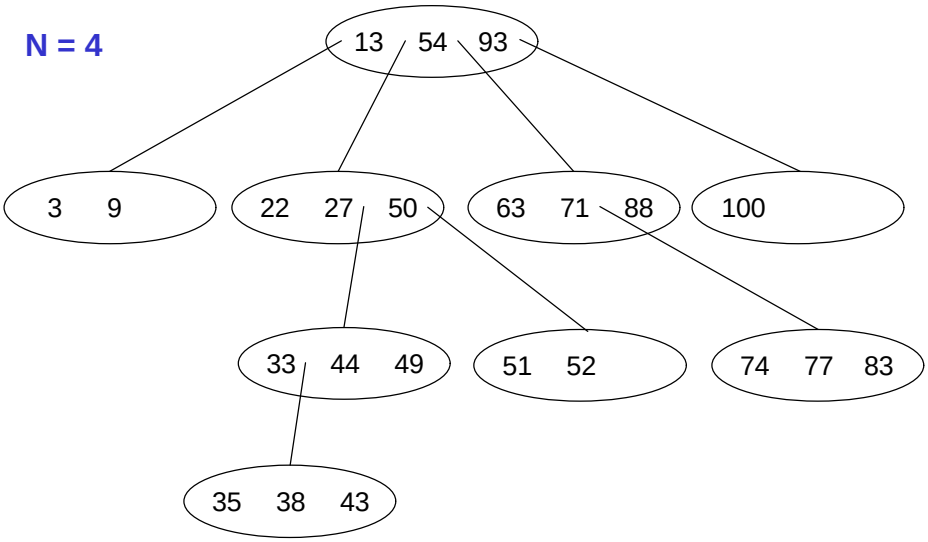
2002/2

Árvore N-ária de Pesquisa

- Uma Árvore N-ária de Pesquisa (ANP) é uma árvore que:
  - contém *m* subárvores e *n* chaves, sendo *n* = *m* -1 e 2 <= *m* <= ***N***
  - todas as chaves estão ordenadas, ou seja, dado um conjunto de chaves ch_1, ch_2, ..., ch_i, ..., ch_n, nesta ordem, tem-se: $ch_i < ch_{i+1}$

Exemplo de ANP

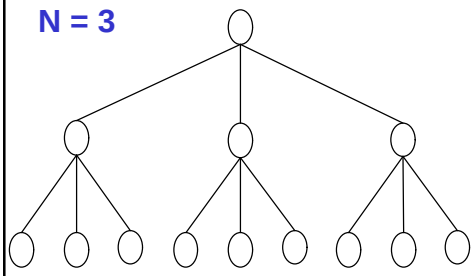
N = 4



Vantagens de uma ANP

- Indexação de um grande volume de dados

N = 3



h(A) = 0 → 2 chaves

h(A) = 1 → 2 + 3.2 = 8 chaves

h(A) = 2 → 2 + 3.2 + 3.2.2 = 26 chaves

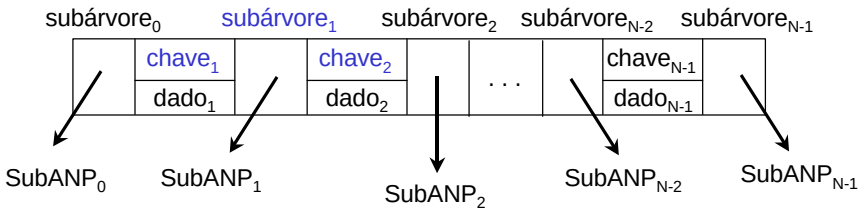
h(A) = x ® N^{x+1} - 1 chaves

"Quanto maior o N, maior é o número de chaves que se pode indexar e conseqüentemente, encontra-se uma chave com menos acessos à arvore"

Vantagens de uma ANP

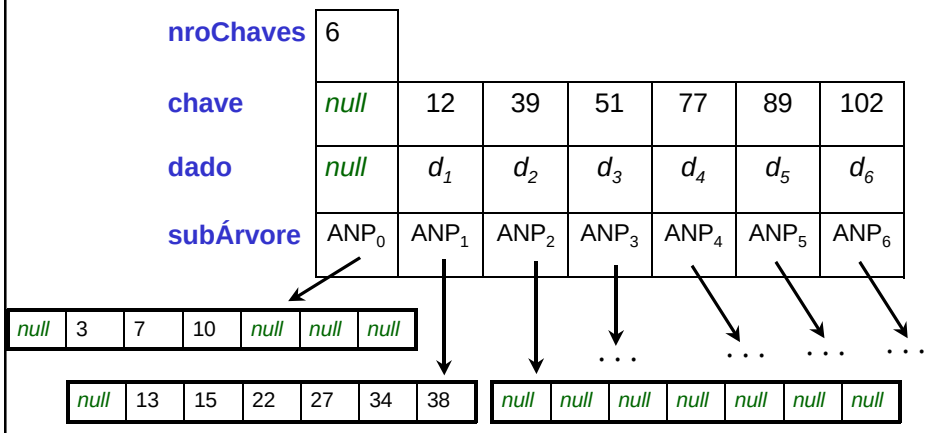
- N pode ser equivalente ao fator de bloco de disco
  - torna mais eficiente o número de acessos a disco para a carga de dados e busca de chave de/em um arquivo de índices de um BD
  - exemplo:
    - fator de bloco = x bytes ≈ nodo de ANP c/127 chaves
    - se N = 128 ⇒ cada acesso traz um nodo da ANP
    - assim: 1º acesso ao arquivo de índice → 127 chaves
      2º acesso → 127^2-1 chaves
      nº acesso → 127^n-1 chaves

Modelagem Física



Subárvore$_i$ mantém todas as chaves ***maiores*** que ***chave$_i$*** e ***menores*** que ***chave$_{i+1}$***

Implementação



Implementação

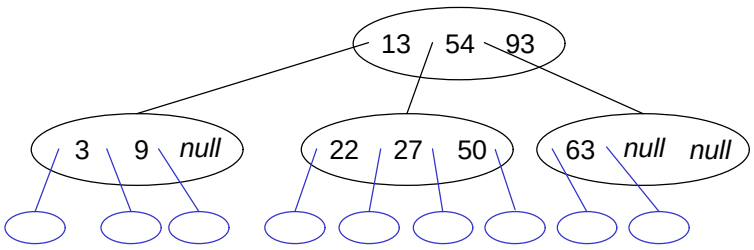
```
Classe ÁrvoreNáriaPesquisa
início
    nroChaves inteiro;
    chave Object[ ];
    dado Object[ ];
    subÁrvore ÁrvoreNáriaPesquisa[ ];

    construtor ÁrvoreNáriaPesquisa (N inteiro);
    início
          se N < 2 então Exceção GrauInválido();
          nroChaves ← 0;
          chave ← NOVO Object[N];
          dado ← NOVO Object[N];
          subÁrvore ← NOVO ÁrvoreNáriaPesquisa[N];
    fim;
fim;
```

Implementação

```
Classe ÁrvoreNáriaPesquisa
início
    nroChaves inteiro;
    chave Object[ ];
    dado Object[ ];
    subÁrvore ÁrvoreNáriaPesquisa[ ];

    método obtémN() retorna inteiro;
    início
          retorna subÁrvore.lenght;
    fim;

fim;
```

Observação

- Para fins de simplificação dos algoritmos de manipulação da ANP, pressupõe-se que toda chave existente em um nodo folha possui subárvores vazias em ambos os lados (nodos com *nroChaves* = 0)

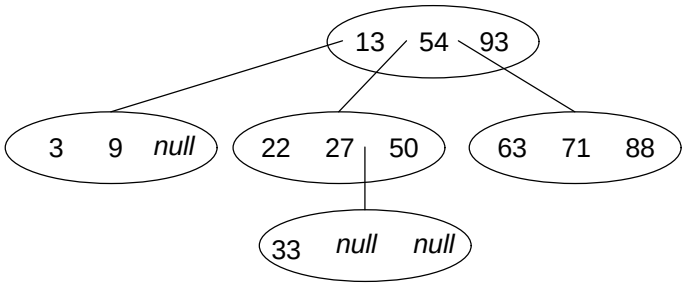


Operações em uma ANP

- Pesquisa
  - pesquisa todos os nodos
  - pesquisa por valor de chave
- Incluir(chave, dado)
- Excluir(chave)

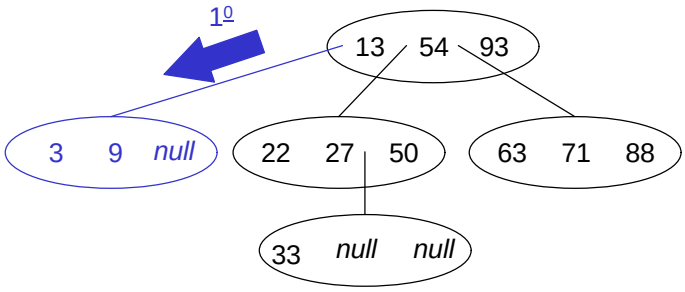
Pesquisa Todos os Nodos

- Busca *in-ordem* em profundidade
  - retorna ordenadamente todas as chaves



Pesquisa Todos os Nodos

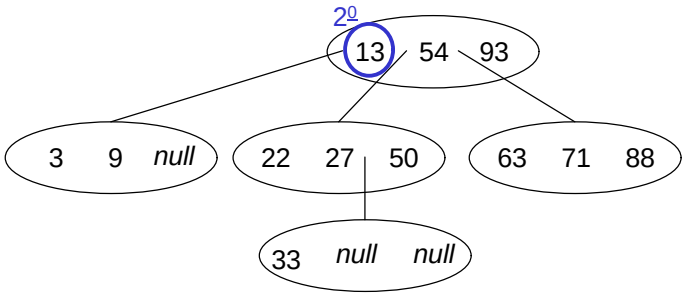
- Busca *in-ordem* em profundidade
  - retorna ordenadamente todas as chaves



3-9

Pesquisa Todos os Nodos

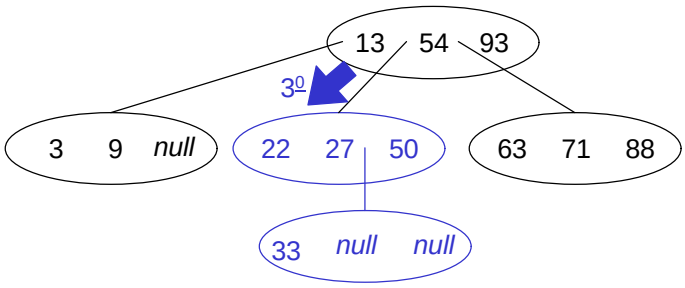
- Busca *in-ordem* em profundidade
  - retorna ordenadamente todas as chaves



3-9-13

Pesquisa Todos os Nodos

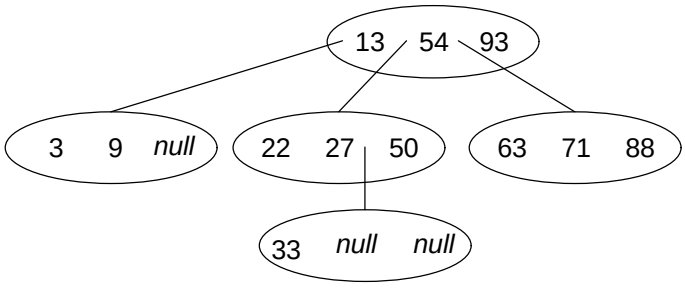
- Busca *in-ordem* em profundidade
  - retorna ordenadamente todas as chaves



3-9-13-22-27-33-50

Pesquisa Todos os Nodos

- Busca *in-ordem* em profundidade
  - retorna ordenadamente todas as chaves



3-9-13-22-27-50-54-63-71-88-93

Complexidade: *O(n)*
n = número de chaves

Pesquisa Todos os Nodos

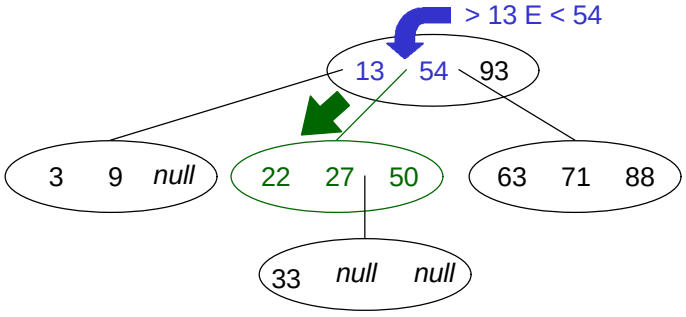
```
Método buscaProfundidade();
início
   i inteiro;

   se this ≠ NULL e nroChaves ≠ 0 então
         para i de 1 até nroChaves faça
         início
                  subÁrvore[i-1].buscaProfundidade();
                  visita(chave[i]);
                  visita(dado[i]);
                  subÁrvore[i].buscaProfundidade();
         fim;
fim;
```

Pesquisa Por Chave

- Mesmo princípio da pesquisa na ABP
  - chave < chave pesquisada $\rightarrow$ pesquisa ESQ
  - chave > chave pesquisada $\rightarrow$ pesquisa DIR
  - complexidade adicional
    - varredura das chaves em cada nodo da ANP
- Tipos de pesquisa
  - Busca Seqüencial(chave)
  - Busca Binária(chave)

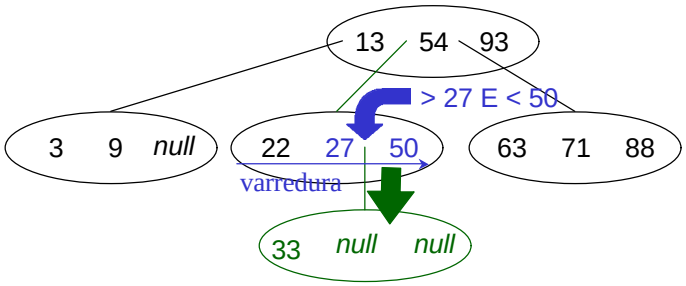
Busca Seqüencial

- Como proceder?
  - exemplo: chave = 33



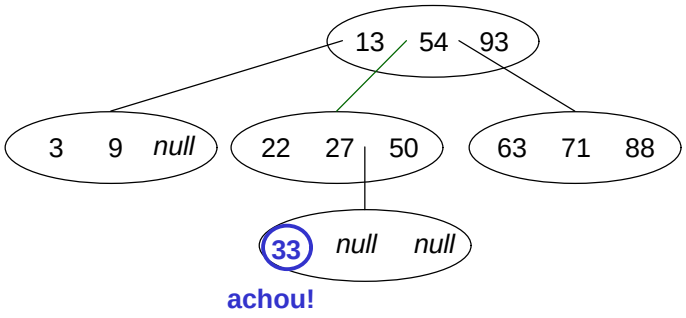
Busca Seqüencial

- Como proceder?
  - exemplo: chave = 33



Busca Seqüencial

- Como proceder?
  - exemplo: chave = 33



Busca Seqüencial

```
Método buscaSeqüencial(ch object) retorna dado;
início
    i, subArv inteiro;

    se nroChaves = 0 então
                Exceção ChaveInexistente();
    para i de 1 até nroChaves faça /* varredura */
    início
        se chave[i] = ch então retorna dado[i];
        se chave [i] > ch então /* deve ir para a subArv à ESQ */
        início
                subArv ← i – 1;
                i ← nroChaves + 1; /* sai do para-faça */
        fim;
    fim;
    retorna subÁrvore[subArv].buscaSeqüencial(ch);
fim;
```

Busca Seqüencial

- Complexidade:
  - cada varredura em um nodo:
    - pesquisa N-1 chaves $\rightarrow O(N-1) \approx O(N)$
  - para encontrar o nodo na ANP:
    - *pior caso*: navega no maior caminho na árvore = $h(A)$
    - Propriedade (P4): $h(A) = \lfloor log_N n \rfloor \rightarrow O(\lfloor log_N n \rfloor) \approx O(log_N n)$
  - complexidade:
    - h(A). (N-1) $\rightarrow$ $\boldsymbol{O(N\ log_N n)}$

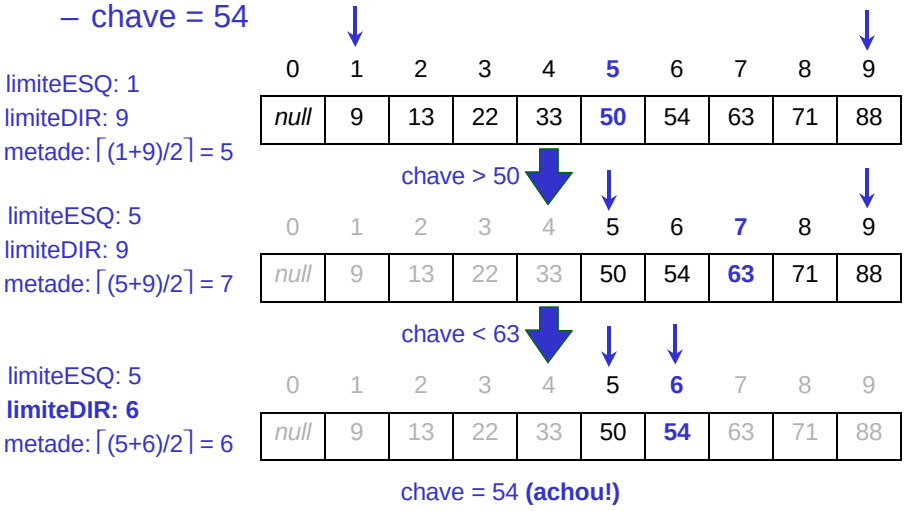
Busca Binária

- Trata de forma mais eficiente a varredura das chaves em um nodo, considerando o fato das chaves estarem ordenadas
- Exemplo:
  - N = 10 e chave = 54

	0	1	2	3	4	5	6	7	8	9
chave:	*null*	9	13	22	33	50	54	63	71	88

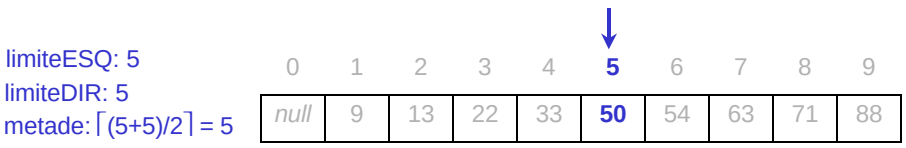
Busca Binária

- Princípio de funcionamento:
  - chave = 54



Busca Binária

- Características:
  - a cada iteração:
    - se chave > metade → busca em [metade;limDIR]
    - se chave < metade → busca em [limESQ;metade-1]
  - se não achou a chave:
    - pára sempre no nodo anterior mais próximo
    - busca continua na subárvore DIR (SubÁrvore[metade])



exemplo: **chave = 52 (não achou!)**
buscaBinária em **Subárvore[5]**

Busca Binária

```
Método buscaBinária(ch object) retorna dado;
início
    índice inteiro;

    se this = NULL ou nroChaves = 0 então
                Exceção ChaveInexistente();
    índice ← obtémÍndice(ch);
    se índice > 0 e chave[índice] = ch então retorna dado[índice];
    retorna subÁrvore[índice].buscaBinária(ch);
fim;
```

Busca Binária

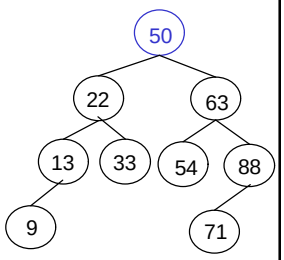
```
Método obtémÍndice(ch object) retorna inteiro;
início
    esq, dir, metade inteiro;

    /* se chave menor que todas as chaves no nodo, ir para Subárv ESQ */
    /* índice = 0 */
    se ch < chave[1] então retorna 0;
    esq  ← 1;
    dir ← nroChaves;
    enquanto (esq < dir) faça  /* executa até que ambos apontem para a
    início                          mesma chave */
          metade  ← ⌈(esq + dir) / 2⌉;
          se ch < chave[metade] então dir ← metade – 1
          senão                          esq ← metade;
    fim;
    retorna esq;
fim;
```

Busca Binária

- Complexidade:
  - varredura em um nodo:
    - pesquisa $\lfloor \log_2 N-1 \rfloor$ chaves $\rightarrow$ $O(\lfloor \log_2 N-1 \rfloor) \approx O(\log_2 N)$

0	1	2	3	4	5	6	7	8	9
null	9	13	22	33	**50**	54	63	71	88

pior caso: h(A) = $\lfloor \log_2 N-1 \rfloor$



  - complexidade:
    - h(ANP). (N-1) $\rightarrow$ $O(\log_2 N \log_N n)$

Comparação

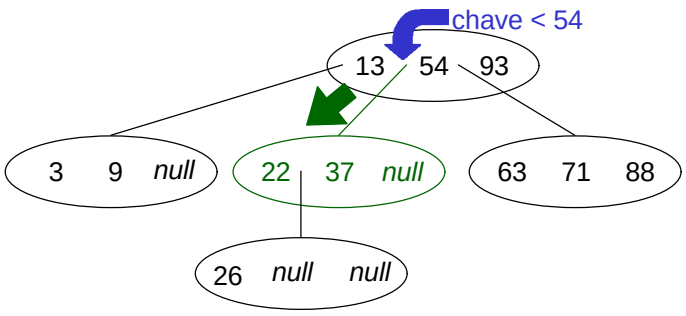
- Busca seqüencial
  - complexidade: $O(N \log_N n)$
- Busca binária
  - complexidade: $O(\log_2 N \log_N n)$

Inclusão em uma ANP

- Mesmo princípio da inclusão em uma ABP
  - busca a posição na qual a chave deve ser inserida (nodo mais próximo com espaço para a colocação da chave)
  - no nodo em que a chave for inserida, o vetor deve ser rearranjado (deslocamento de chaves, dados e subárvores) para a sua correta colocação

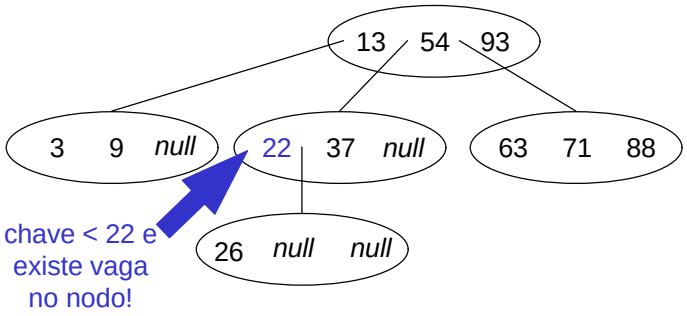
Inclusão em uma ANP

- Exemplo: chave = 18



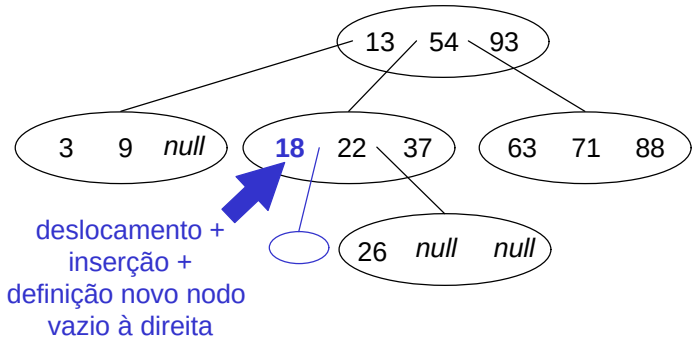
Inclusão em uma ANP

- Exemplo: chave = 18



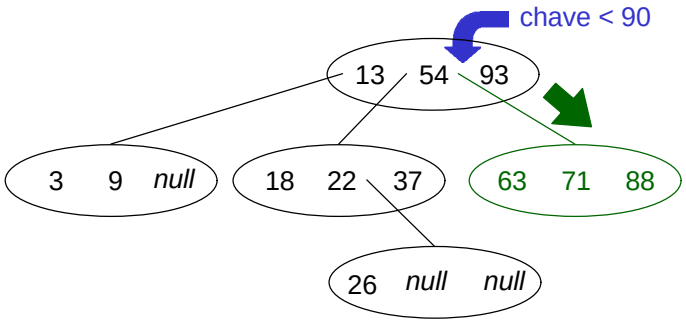
Inclusão em uma ANP

- Exemplo: chave = 18



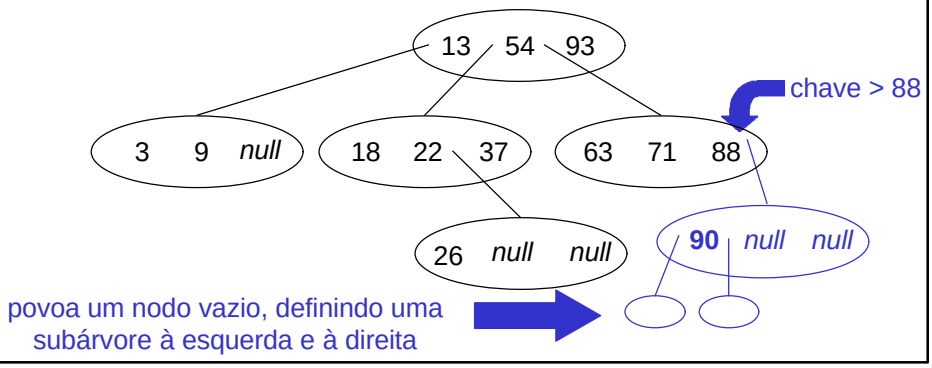
Inclusão em uma ANP

- Exemplo2: chave = 90



Inclusão em uma ANP

- Exemplo2: chave = 90



Inclusão

```
Método inclui(ch object, d object);
início
    índice, i inteiro;

    se nroChaves = 0 então /* chegou em um nodo vazio */
    início
        subÁrvore[0] ← NOVO ÁrvoreNáriaPesquisa(obtémN());
        chave[1] ← ch;
        dado [1] ← d;
        subÁrvore[1] ← NOVO ÁrvoreNáriaPesquisa(obtémN());
        nroChaves ← 1;
    fim
    senão início /* chegou em um nodo com chaves */
    . . .
fim;
```

Inclusão

```
Método inclui(ch object, d object);
início
    . . .
    senão início /* chegou em um nodo com chaves */
            índice ← obtémÍndice(ch); /* busca posição para inserção */
            se índice ≠ 0 e ch = chave[índice] então
                    Exceção ChaveDuplicada();
            se nroChaves < obtémN() – 1 então /* existe espaço no nodo */
            início
                    para i de nroChaves até índice + 1 faça /* deslocamento */
                    início
                      chave[i+1] ← chave[i];
                      dado [i+1] ← dado[i];
                      subÁrvore[i+1] ← subÁrvore[i];
                    fim;
                    chave[índice+1] ← ch; /* índice aponta para chave anterior */
                    dado[índice+1] ← d;    /* por isso, nova chave fica em índice+1 */
                    subÁrvore[índice+1] ← NOVO ÁrvoreNáriaPesquisa(obtémN());
                    nroChaves ← nroChaves + 1;
            fim
           senão /* não existe espaço no nodo: vai p/ a subárvore DIR da chave anterior */
                    subÁrvore[índice].inclui(ch,d);
          fim;
fim;
```

Inclusão em uma ANP

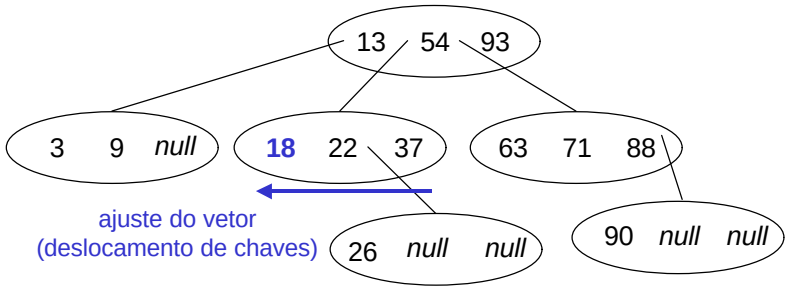
- Etapas do algoritmo:
  - busca da chave: $O(log_2 N \; log_N n)$
  - ajuste vetor chave: $O(N)$
- Complexidade:
  - $O(log_2 N \; log_N n + N)$

Exclusão em uma ANP

- Mesmo princípio da exclusão em uma ABP
  - se a chave **não** possui subárvores ESQ e DIR vazias, ela é removida e ocorre deslocamento no vetor para ajustar as chaves e dados restantes
  - se a chave possui subárvores ESQ e/ou DIR com chaves, ela é trocada com a maior chave da subárvore ESQ ou a menor chave da subárvore DIR (processo recursivo – até que a chave não contenha subárvores!)

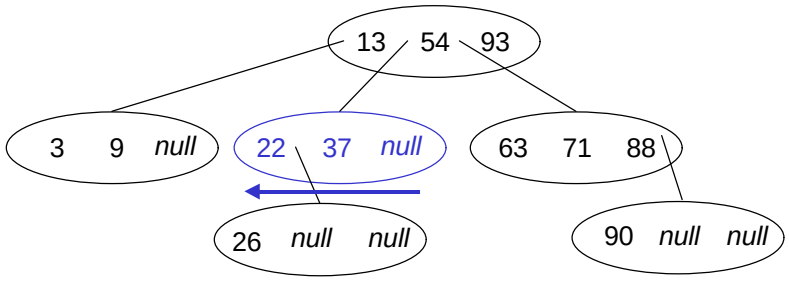
Exclusão em uma ANP

- Exemplo1: chave = 18
  - não há subárvores ESQ e DIR



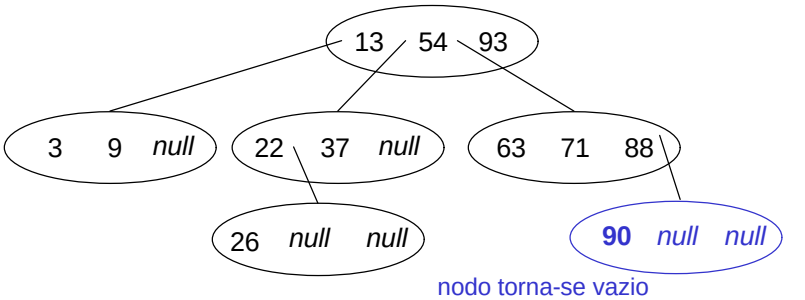
Exclusão em uma ANP

- Exemplo1: chave = 18
  - não há subárvores ESQ e DIR



Exclusão em uma ANP

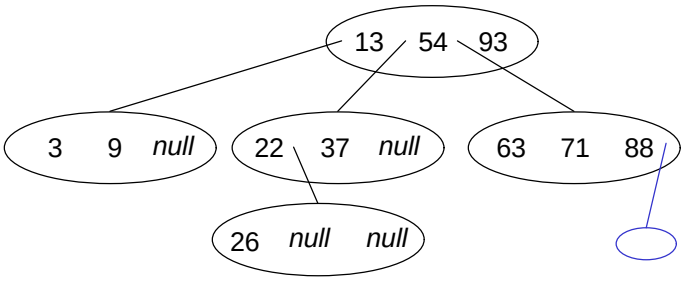
- Exemplo2: chave = 90
  - não há subárvores ESQ e DIR e a chave é a única do nodo



nodo torna-se vazio
(subárvores ESQ e DIR de 90 tornam-se NULL!)

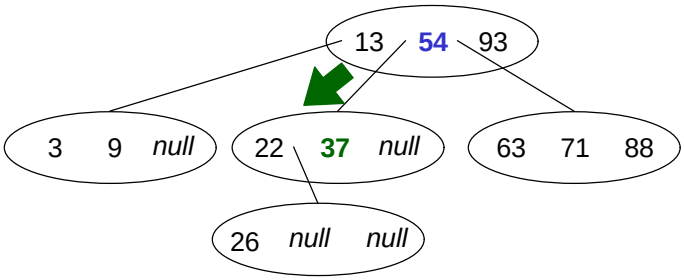
Exclusão em uma ANP

- Exemplo2: chave = 90
  - não há subárvores ESQ e DIR e a chave é a única do nodo



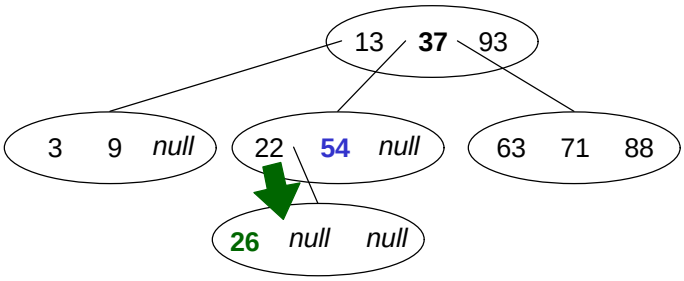
Exclusão em uma ANP

- Exemplo3: chave = 54
  - existem subárvores ESQ e DIR: a chave é trocada com o maior chave na ESQ



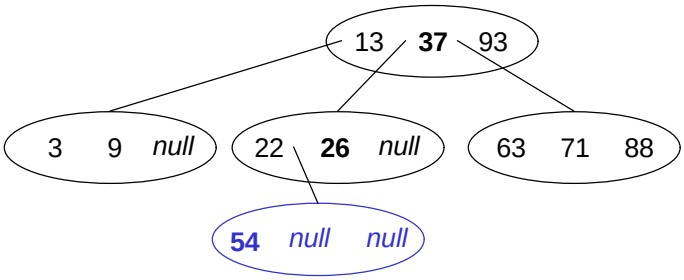
Exclusão em uma ANP

- Exemplo3: chave = 54
  - a chave ainda possui subárvore ESQ não vazia: nova troca!



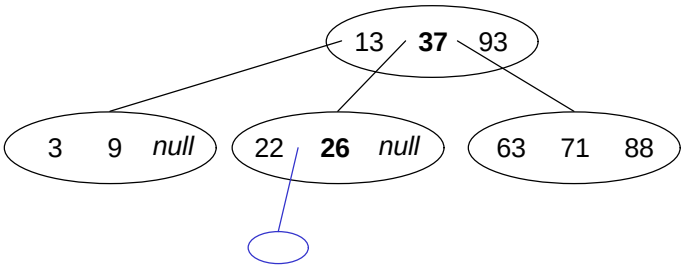
Exclusão em uma ANP

- Exemplo3: chave = 54
  - a chave agora pode ser removida!



Exclusão em uma ANP

- Exemplo3: chave = 54
  - chave removida!



Exclusão

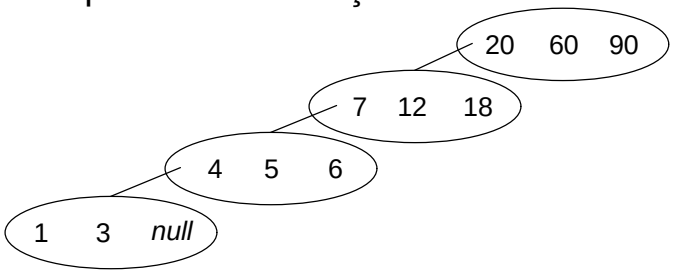
```
Método exclui(ch object);
início
    índice, i inteiro;
    arvAux ÁrvoreNáriaPesquisa;

    se nroChaves = 0 então /* chegou em um nodo vazio */
          Exceção ChaveInexistente();
    índice ← obtémÍndice(ch);
    se índice ≠ 0 e ch = chave[índice] então  /* achou a chave! */
    início
          se subÁrvore[índice-1].nroChaves >  0 então /* se existe ESQ */
          início
                aux ← subÁrvore[índice-1].maior();
                trocaValoresMaior(this, índice, aux);
                subÁrvore[índice-1].exclui(ch);
          fim
          senão  /* verifica se existe DIR */ . . .
fim;
```

Exclusão

```
Método exclui(ch object);
início
. . .senão se subÁrvore[índice].nroChaves >  0 então /* se existe DIR */
           início
                aux ← subÁrvore[índice].menor();
                trocaValoresMenor(this, índice, aux);
                subÁrvore[índice].exclui(ch);
           fim
           senão início /* chave não possui ESQ e DIR e pode ser removida */
                nroChaves ← nroChaves – 1;
                para i de índice até nroChaves faça
                início
                      chave[i] ← chave[i+1];
                      dado[i] ← dado[i+1];
                      subÁrvore[i] ← subÁrvore[i+1];
                fim;
                chave[i] ←NULL; dado[i] ←NULL;    /* anula a posição mais a */
                subÁrvore[i] ←NULL;               /* direita do vetor */
                se nroChaves = 0 então subÁrvore[0] ←NULL;
               fim
    senão subÁrvore[índice].exclui(ch); /* se não achou, vai p/ DIR do */
                                        /* nodo imediatamente anterior */
fim;
```

Problema com ANP

- Uma ANP também pode ficar "desbalanceada"!
- Exemplo:
  - N = 4
  - seqüência de inserção: 20-60-90-12-7-18-5-4-6-1-3



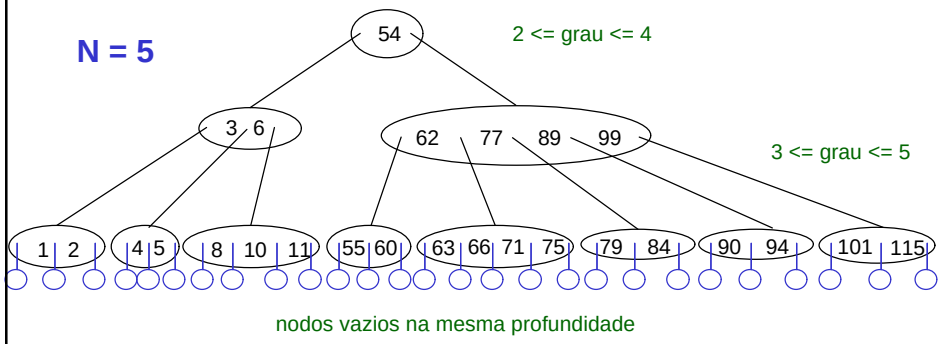
- Uma solução: usar árvores B!

Árvore B

- Uma árvore B é uma ANP com as seguintes características:
  - raiz com 2 <= grau <= N
  - outros nodos têm ⌈N/2⌉ <= grau <= N
  - nodos vazios estão todos na mesma profundidade

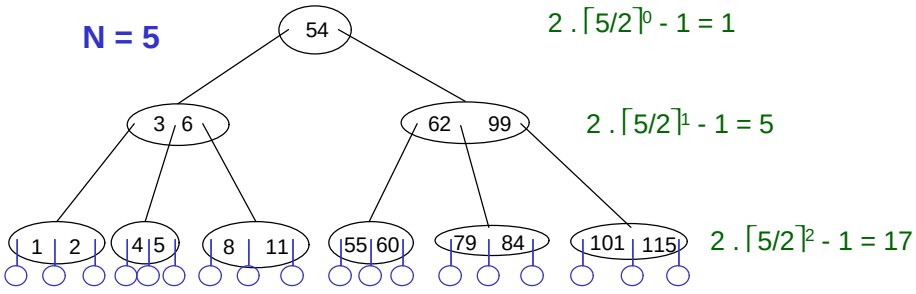
grau: *número de subárvores não vazias*

Exemplo



Propriedade (P5)

- O número mínimo de chaves em uma Árvore B *A* com N >= 2 é $\mathbf{2\lceil N/2 \rceil^{h(A)} - 1}$



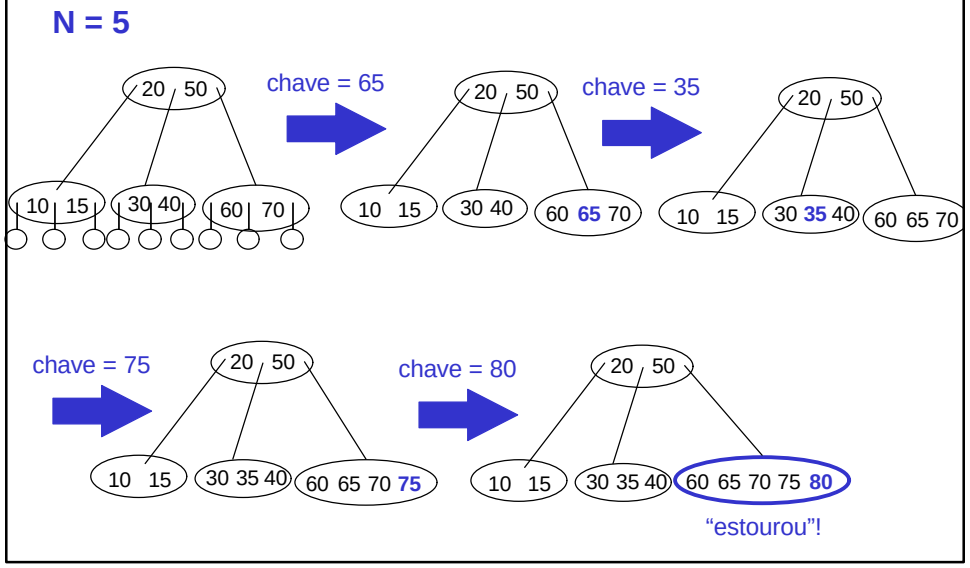
Implementação

```
Classe ÁrvoreB
Subclasse de ÁrvoreNáriaPesquisa
início

    pai ÁrvoreB [ ];

    construtor ÁrvoreB (N inteiro);
    início
        se N < 2 então Exceção GrauInválido();
        nroChaves ← 0;
        chave ← NOVO Object[N];
        dado ← NOVO Object[N];
        subÁrvore ← NOVO ÁrvoreNáriaPesquisa[N];
        pai ← NULL;
    fim;
fim;
```

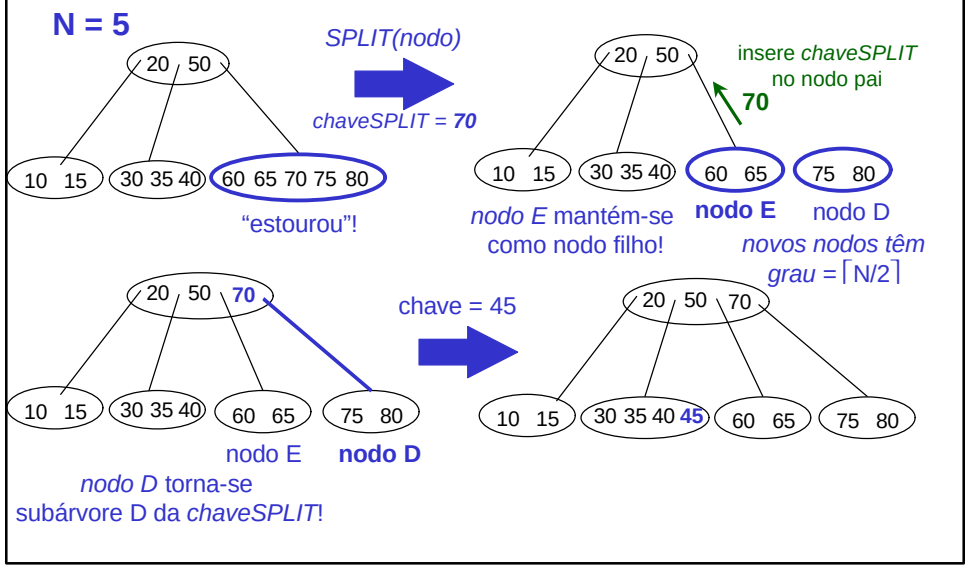
Inserção em uma Árvore B

- Princípio de funcionamento:
  - inserção sempre em nodo folha
  - se há espaço no nodo ela é ali inserida
- Caso o nodo exceder a sua capacidade:
  - divisão do nodo (*Split*)
  - a chave central (chave *split*) do nodo "estourado" é enviada ao nodo pai para ser lá inserida
  - o processo se repete até que não ocorram mais divisões ou que seja criada uma nova raiz (árvore cresce para cima)

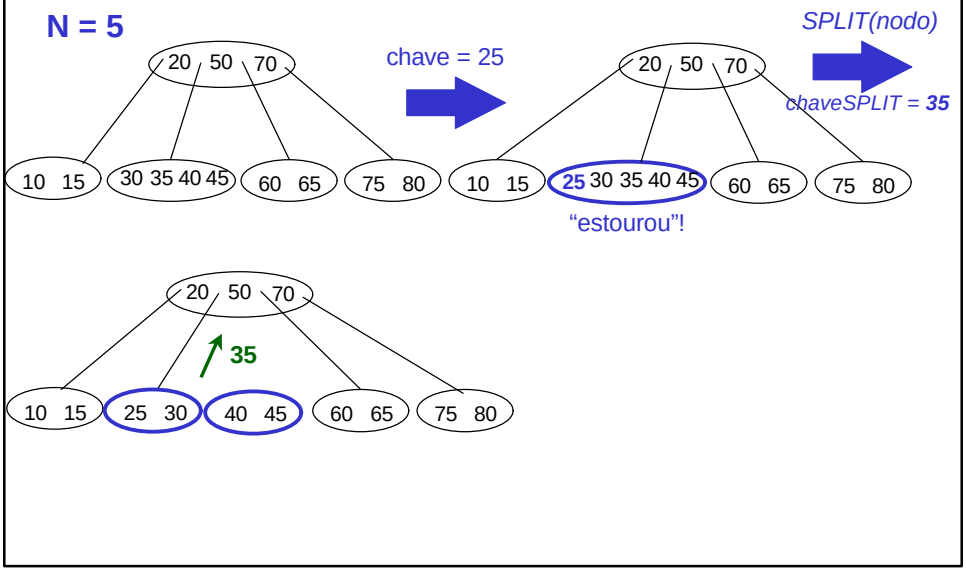
Exemplos de Inclusão



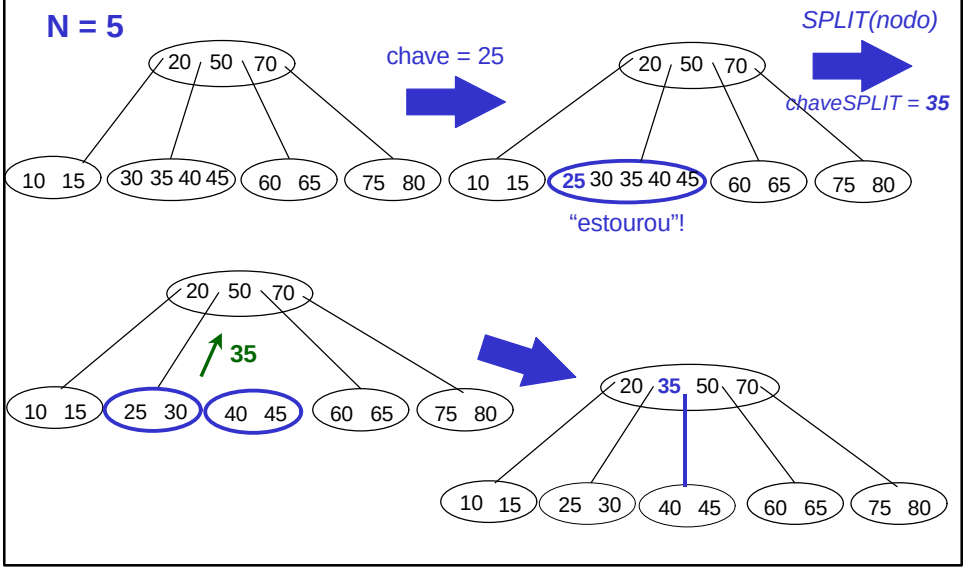
Exemplos de Inclusão



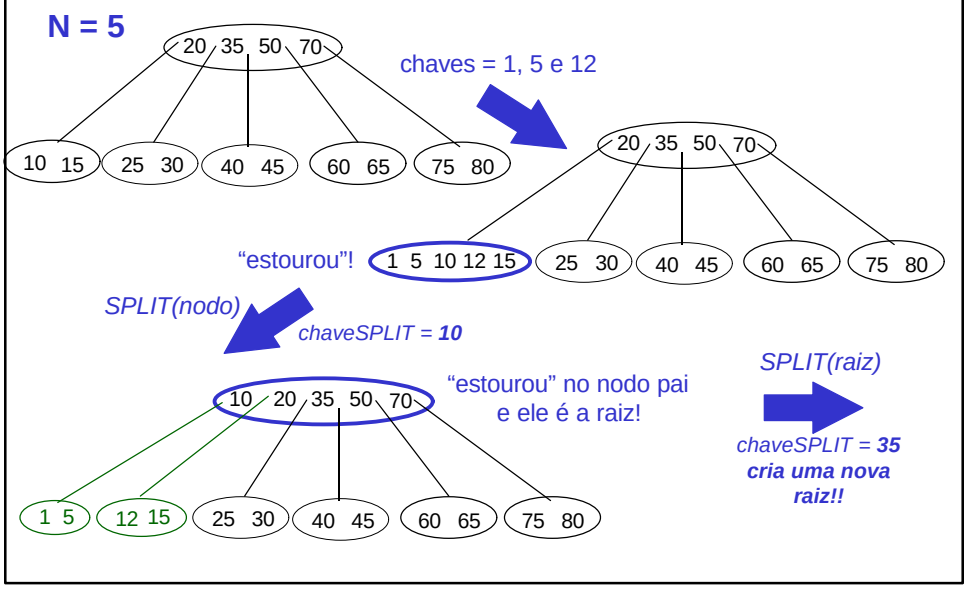
Exemplos de Inclusão



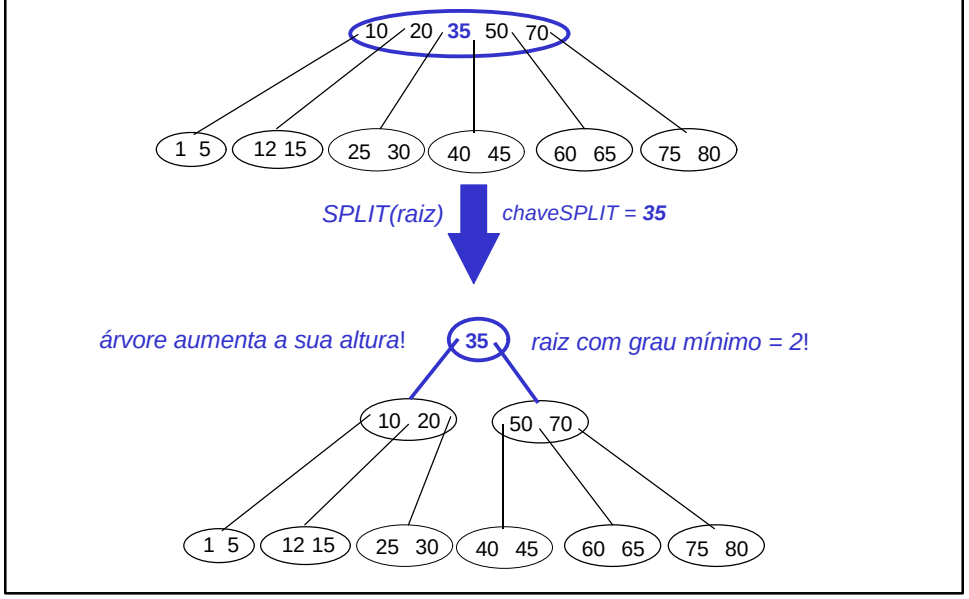
Exemplos de Inclusão



Exemplos de Inclusão



Exemplos de Inclusão



Exercícios

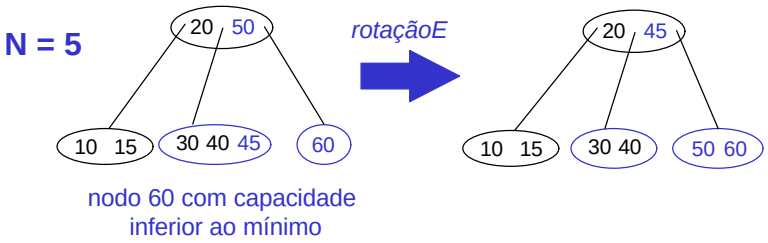
1. Dado N = 5, mostre como fica uma árvore B após cada uma das seguintes inserções de chaves: 20, 10, 40, 50, 30, 55, 3, 11, 4, 28, 36, 33, 52, 17, 25, 13, 45, 9, 43, 8, 48
2. Implementar na classe ANP:
   - Método maior() retorna ANP;
   - Método menor() retorna ANP;
   - Método trocaValoresMaior(arv1 ANP, índice inteiro, arv2 ANP);
   - Método totalChaves() retorna inteiro;

Exclusão em uma Árvore B

- Princípio inicial de funcionamento:
  - similar à exclusão na ANP
  - pesquisa na árvore pela chave informada
  - se a chave **não** possui subárvores ESQ e DIR então **remove** a chave e faz deslocamento das chaves no vetor
  - senão, **troca** a chave pela maior chave na subárvore ESQ ou pela menor chave na subárvore DIR e recursivamente exclui a chave na subárvore para a qual ela foi enviada

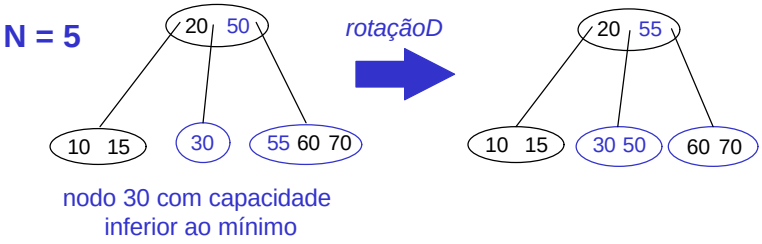
Exclusão em uma Árvore B

- Após a remoção, é possível que o número de chaves em um nodo não raiz seja **menor** que o mínimo permitido (⌈N/2⌉ - 1). Então:
  - se existe nodo irmão à ESQ com nroChaves > ⌈N/2⌉ - 1, faz **rotaçãoE**:



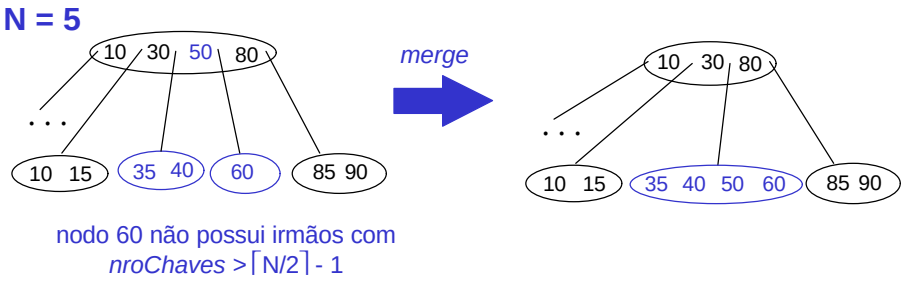
Exclusão em uma Árvore B

  - senão, se existe nodo irmão à DIR com nroChaves > ⌈N/2⌉ - 1, faz **rotaçãoD**:



Exclusão em uma Árvore B

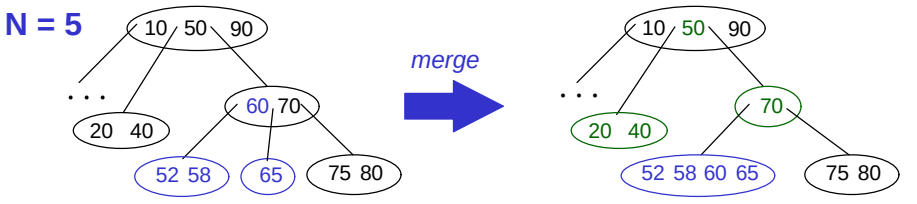
– senão (se não existe nodo irmão à ESQ nem à DIR com nroChaves > ⌈N/2⌉ - 1), faz-se unificação (*merge*) de nodos:



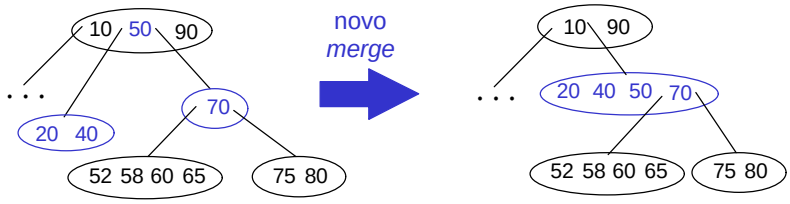
nodo 60 não possui irmãos com *nroChaves* > ⌈N/2⌉ - 1

Exclusão em uma Árvore B

– A unificação (*merge*) pode se propagar para mais de um ancestral:

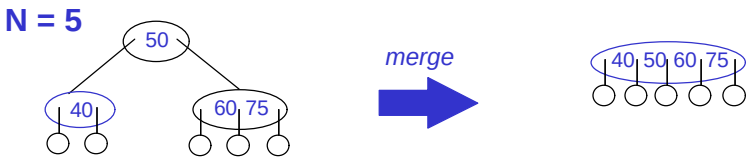


nodo 70 ficou com capacidade inferior ao mínimo!



Exclusão em uma Árvore B

– Quando a unificação (*merge*) se propaga até a raiz, a árvore reduz a sua altura:



Exercício

Considerando a árvore B gerada pelo exercício 1, mostre como ela fica após cada uma das seguintes exclusões de chaves: 20, 33, 4, 50, 30, 55, 11, 40, 28, 36, 10, 52, 17, 25, 13, 45, 9, 43, 8