AULA 18
ESTRUTURA DE DADOS

Árvores Binárias de Pesquisa

Norton T. Roman & Luciano A. Digiampietri

Árvores Binárias de Pesquisa

Para nossa árvore binária de pesquisa, veremos as seguintes funções:

Inicialização da árvore

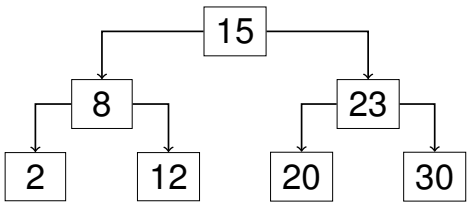
Inserção de um elemento

Busca de um elemento

Contagem do número de elementos

Leitura da árvore

Remoção de um elemento



ABP – Remoção

Problemas na remoção de um nó:

ABP – Remoção

Problemas na remoção de um nó:

- Temos que lidar com as subárvores desse nó

ABP – Remoção

Problemas na remoção de um nó:

- Temos que lidar com as subárvores desse nó
- A árvore resultante deve continuar sendo de busca

ABP – Remoção

Problemas na remoção de um nó:

- Temos que lidar com as subárvores desse nó
- A árvore resultante deve continuar sendo de busca
  - Nós da subárvore da esquerda têm chave menor que a do nó raiz

ABP – Remoção

Problemas na remoção de um nó:

- Temos que lidar com as subárvores desse nó
- A árvore resultante deve continuar sendo de busca
  - Nós da subárvore da esquerda têm chave menor que a do nó raiz
  - Nós da subárvore da direita têm chave maior que a do nó raiz

ABP – Remoção

Como fazer?

ABP – Remoção

Como fazer?

Se o nó a ser retirado possui no máximo um descendente, substitua-o por este

ABP – Remoção

Como fazer?

- Se o nó a ser retirado possui no máximo um descendente, substitua-o por este
- Se o nó possuir 2 descendentes, substituímos o nó a ser retirado pelo nó mais à direita da subárvore da esquerda

ABP – Remoção

Como fazer?

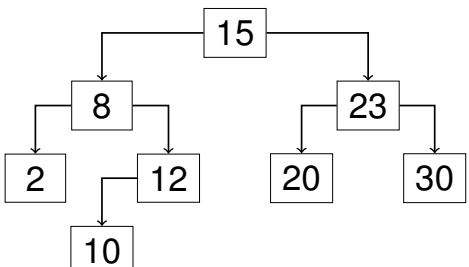
Se o nó a ser retirado possui no máximo um descendente, substitua-o por este

Se o nó possuir 2 descendentes, substituímos o nó a ser retirado pelo nó mais à direita da subárvore da esquerda

Alternativamente, substituímos o nó a ser retirado pelo nó mais à esquerda da subárvore da direita

ABP – Remoção

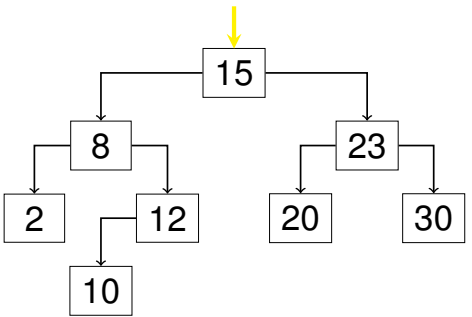
Ou seja...



ABP – Remoção

Ou seja...

Para removermos o 15



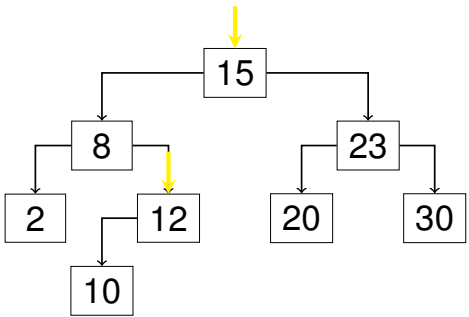
ABP – Remoção

Ou seja...

Para removermos o 15

Ou substituímos pelo 12

E 10 passa a ser filho de 8



ABP – Remoção

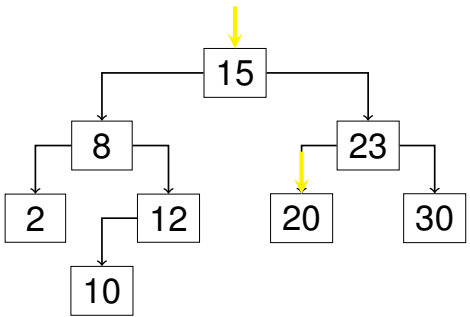
Ou seja...

Para removermos o 15

Ou substituímos pelo 12

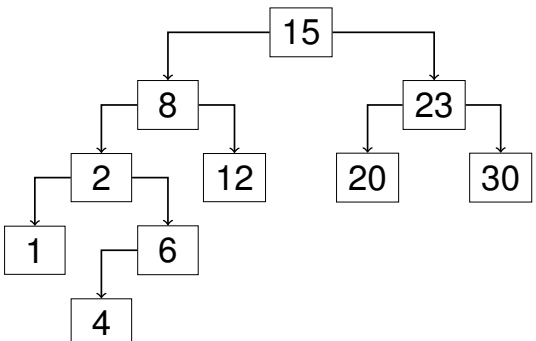
E 10 passa a ser filho de 8

Ou substituímos pelo 20



ABP – Remoção

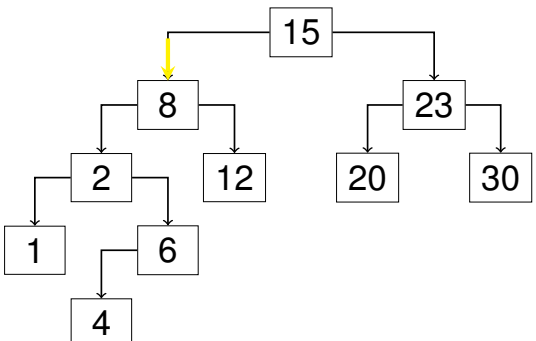
Para remover, precisamos então saber:



ABP – Remoção

Para remover, precisamos então saber:

O nó a ser removido

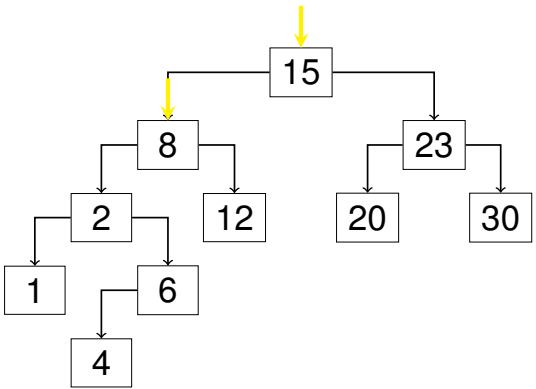


ABP – Remoção

Para remover, precisamos então saber:

O nó a ser removido

Seu pai



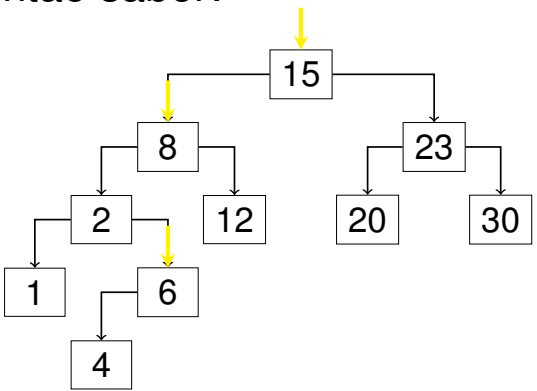
ABP – Remoção

Para remover, precisamos então saber:

O nó a ser removido

Seu pai

O nó substituto



ABP – Remoção

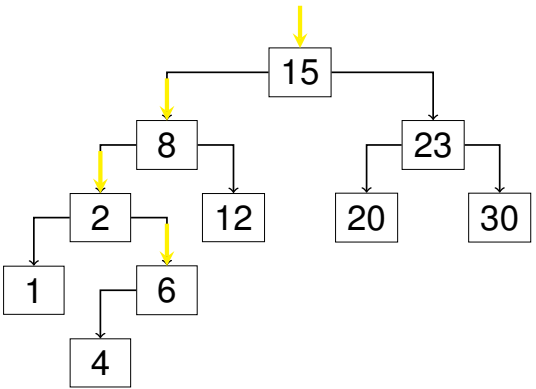
Para remover, precisamos então saber:

O nó a ser removido

Seu pai

O nó substituto

Seu pai



ABP – Remoção

Vamos então fazer um método auxiliar:

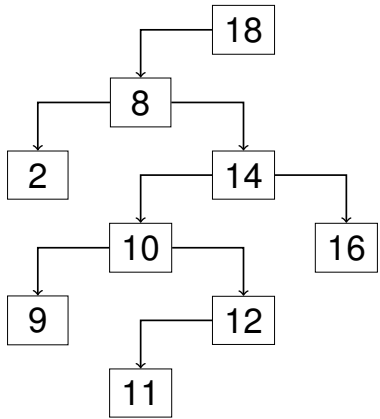
```
/*
  Busca binária não recursiva. Devolve o ponteiro do nó
  buscado. Abastece pai com o ponteiro para o nó pai deste
*/
PONT buscaNo(PONT raiz, TIPOCHAVE ch, PONT *pai){
  PONT atual = raiz;
  *pai = NULL;
  while (atual) {
    if(atual->chave == ch) return(atual);
    *pai = atual;
    if(ch < atual->chave) atual = atual->esq;
    else atual = atual->dir;
  }
  return(NULL);
}
```

ABP – Remoção

```
PONT removeNo(PONT raiz, TIPOCHAVE ch){
  PONT pai, no, p, q;
  no = buscaNo(raiz,ch,&pai);
  if (no==NULL) return(raiz);
  if (!no->esq || !no->dir ) {
    if (!no->esq) q = no->dir;
    else q = no->esq;
  }
  else {
    p = no;
    q = no->esq;
    while (q->dir) {
      p = q;
      q = q->dir;
    }
    if (p != no) {
      p->dir = q->esq;
      q->esq = no->esq;
    }
    q->dir = no->dir;
  }
  if (!pai) {
    free(no);
    return(q);
  }
  if (ch < pai->chave) pai->esq = q;
  else pai->dir = q;
  free(no);
  return(raiz);
}
```

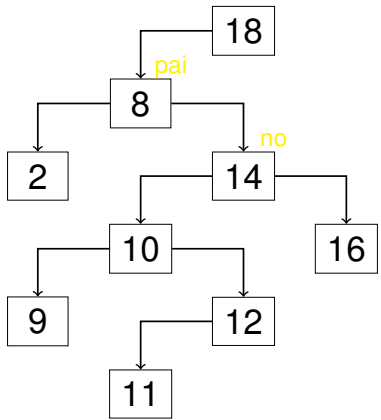
ABP – Remoção (Exemplo)

```
PONT removeNo(PONT raiz, TIPOCHAVE ch){
  PONT pai, no, p, q;
  no = buscaNo(raiz,ch,&pai);
  if (no==NULL) return(raiz);
  if (!no->esq || !no->dir ) {
    if (!no->esq) q = no->dir;
    else q = no->esq;
  }
  else {
    p = no;
    q = no->esq;
    while (q->dir) {
      p = q;
      q = q->dir;
    }
```



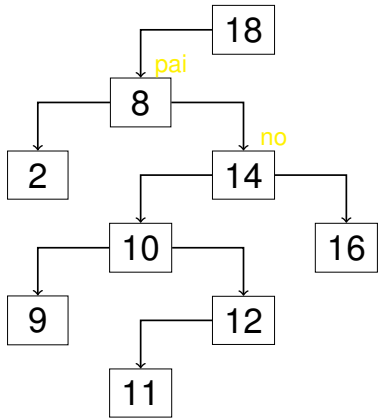
ABP – Remoção (Exemplo)

```
PONT removeNo(PONT raiz, TIPOCHAVE ch){
  PONT pai, no, p, q;
  no = buscaNo(raiz,ch,&pai);
  if (no==NULL) return(raiz);
  if (!no->esq || !no->dir ) {
    if (!no->esq) q = no->dir;
    else q = no->esq;
  }
  else {
    p = no;
    q = no->esq;
    while (q->dir) {
      p = q;
      q = q->dir;
    }
```



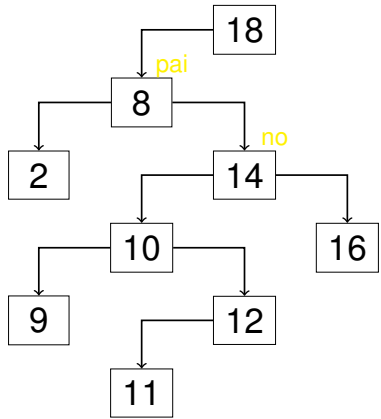
ABP – Remoção (Exemplo)

```
PONT removeNo(PONT raiz, TIPOCHAVE ch){
  PONT pai, no, p, q;
  no = buscaNo(raiz,ch,&pai);
  if (no==NULL) return(raiz);
  if (!no->esq || !no->dir ) {
    if (!no->esq) q = no->dir;
    else q = no->esq;
  }
  else {
    p = no;
    q = no->esq;
    while (q->dir) {
      p = q;
      q = q->dir;
    }
```



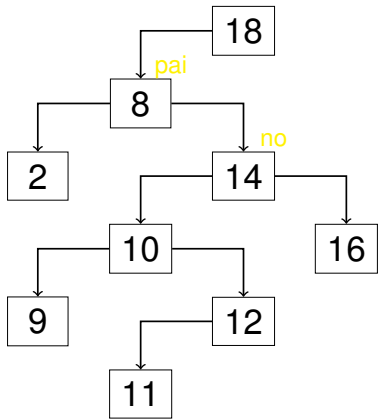
ABP – Remoção (Exemplo)

```
PONT removeNo(PONT raiz, TIPOCHAVE ch){
  PONT pai, no, p, q;
  no = buscaNo(raiz,ch,&pai);
  if (no==NULL) return(raiz);
  if (!no->esq || !no->dir ) {
    if (!no->esq) q = no->dir;
    else q = no->esq;
  }
  else {
    p = no;
    q = no->esq;
    while (q->dir) {
      p = q;
      q = q->dir;
    }
```



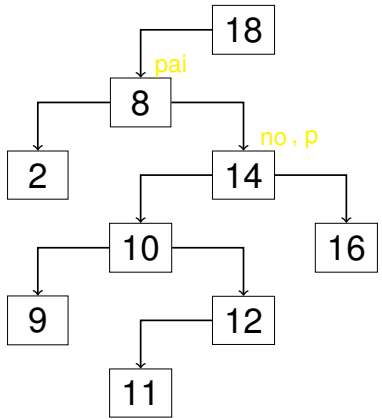
ABP – Remoção (Exemplo)

```
PONT removeNo(PONT raiz, TIPOCHAVE ch){
  PONT pai, no, p, q;
  no = buscaNo(raiz,ch,&pai);
  if (no==NULL) return(raiz);
  if (!no->esq || !no->dir ) {
    if (!no->esq) q = no->dir;
    else q = no->esq;
  }
  else {
    p = no;
    q = no->esq;
    while (q->dir) {
      p = q;
      q = q->dir;
    }
```



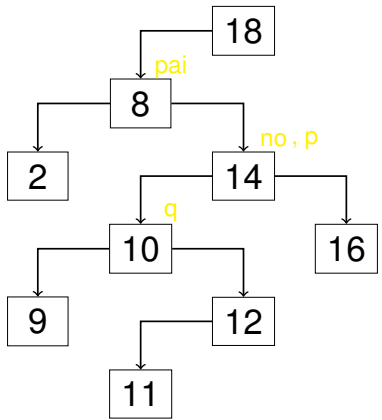
ABP – Remoção (Exemplo)

```
PONT removeNo(PONT raiz, TIPOCHAVE ch){
  PONT pai, no, p, q;
  no = buscaNo(raiz,ch,&pai);
  if (no==NULL) return(raiz);
  if (!no->esq || !no->dir ) {
    if (!no->esq) q = no->dir;
    else q = no->esq;
  }
  else {
    p = no;
    q = no->esq;
    while (q->dir) {
      p = q;
      q = q->dir;
    }
```



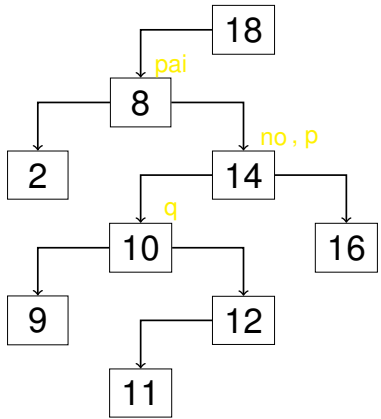
ABP – Remoção (Exemplo)

```
PONT removeNo(PONT raiz, TIPOCHAVE ch){
  PONT pai, no, p, q;
  no = buscaNo(raiz,ch,&pai);
  if (no==NULL) return(raiz);
  if (!no->esq || !no->dir ) {
    if (!no->esq) q = no->dir;
    else q = no->esq;
  }
  else {
    p = no;
    q = no->esq;
    while (q->dir) {
      p = q;
      q = q->dir;
    }
```



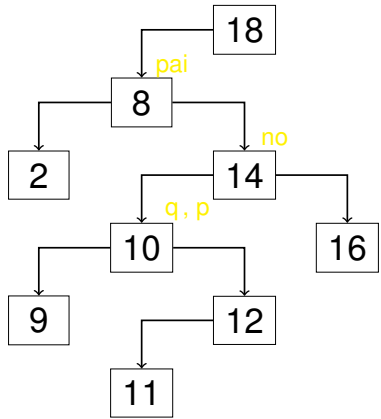
ABP – Remoção (Exemplo)

```
PONT removeNo(PONT raiz, TIPOCHAVE ch){
  PONT pai, no, p, q;
  no = buscaNo(raiz,ch,&pai);
  if (no==NULL) return(raiz);
  if (!no->esq || !no->dir ) {
    if (!no->esq) q = no->dir;
    else q = no->esq;
  }
  else {
    p = no;
    q = no->esq;
    while (q->dir) {
      p = q;
      q = q->dir;
    }
```



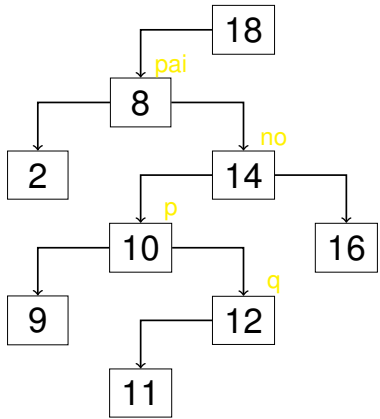
ABP – Remoção (Exemplo)

```
PONT removeNo(PONT raiz, TIPOCHAVE ch){
  PONT pai, no, p, q;
  no = buscaNo(raiz,ch,&pai);
  if (no==NULL) return(raiz);
  if (!no->esq || !no->dir ) {
    if (!no->esq) q = no->dir;
    else q = no->esq;
  }
  else {
    p = no;
    q = no->esq;
    while (q->dir) {
      p = q;
      q = q->dir;
    }
```



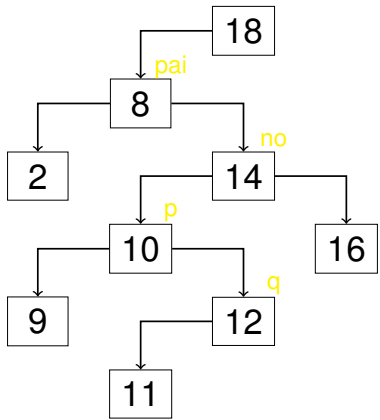
ABP – Remoção (Exemplo)

```
PONT removeNo(PONT raiz, TIPOCHAVE ch){
  PONT pai, no, p, q;
  no = buscaNo(raiz,ch,&pai);
  if (no==NULL) return(raiz);
  if (!no->esq || !no->dir ) {
    if (!no->esq) q = no->dir;
    else q = no->esq;
  }
  else {
    p = no;
    q = no->esq;
    while (q->dir) {
      p = q;
      q = q->dir;
    }
```



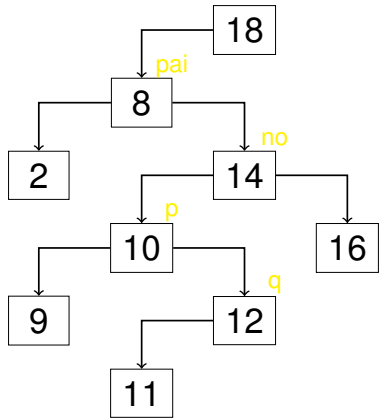
ABP – Remoção (Exemplo)

```
PONT removeNo(PONT raiz, TIPOCHAVE ch){
  PONT pai, no, p, q;
  no = buscaNo(raiz,ch,&pai);
  if (no==NULL) return(raiz);
  if (!no->esq || !no->dir ) {
    if (!no->esq) q = no->dir;
    else q = no->esq;
  }
  else {
    p = no;
    q = no->esq;
    while (q->dir) {
      p = q;
      q = q->dir;
    }
```



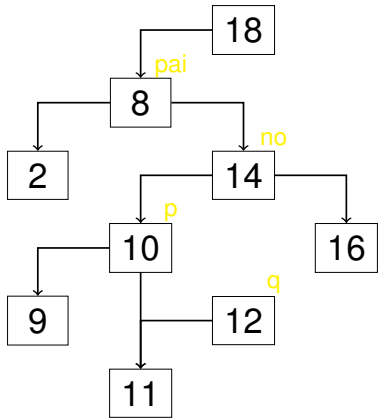
ABP – Remoção (Exemplo)

```
    if (p != no) {
      p->dir = q->esq;
      q->esq = no->esq;
    }
    q->dir = no->dir;
  }
  if (!pai) {
    free(no);
    return(q);
  }
  if (ch < pai->chave) pai->esq = q;
  else pai->dir = q;
  free(no);
  return(raiz);
}
```



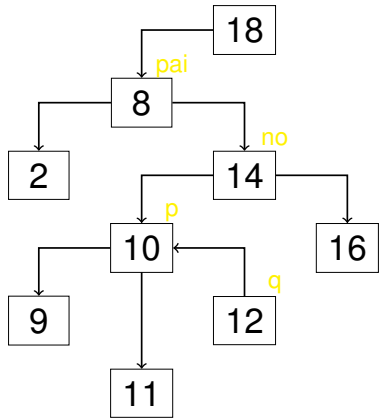
ABP – Remoção (Exemplo)

```
    if (p != no) {
      p->dir = q->esq;
      q->esq = no->esq;
    }
    q->dir = no->dir;
  }
  if (!pai) {
    free(no);
    return(q);
  }
  if (ch < pai->chave) pai->esq = q;
  else pai->dir = q;
  free(no);
  return(raiz);
}
```



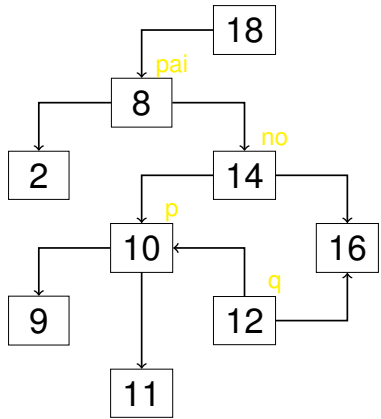
ABP – Remoção (Exemplo)

```
    if (p != no) {
      p->dir = q->esq;
      q->esq = no->esq;
    }
    q->dir = no->dir;
  }
  if (!pai) {
    free(no);
    return(q);
  }
  if (ch < pai->chave) pai->esq = q;
  else pai->dir = q;
  free(no);
  return(raiz);
}
```



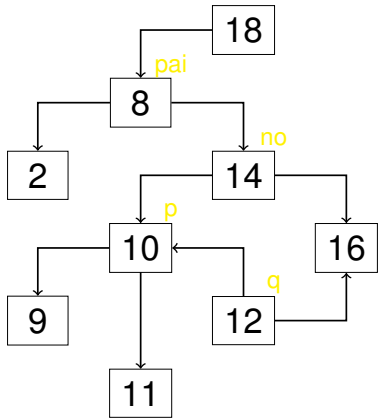
ABP – Remoção (Exemplo)

```
    if (p != no) {
      p->dir = q->esq;
      q->esq = no->esq;
    }
    q->dir = no->dir;
  }
  if (!pai) {
    free(no);
    return(q);
  }
  if (ch < pai->chave) pai->esq = q;
  else pai->dir = q;
  free(no);
  return(raiz);
}
```



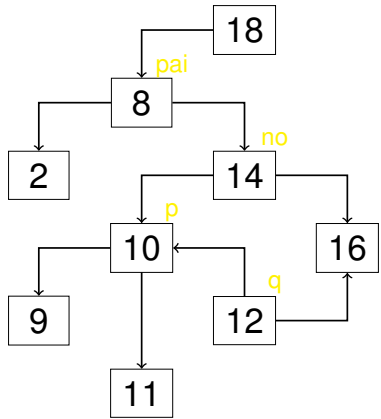
ABP – Remoção (Exemplo)

```
    if (p != no) {
      p->dir = q->esq;
      q->esq = no->esq;
    }
    q->dir = no->dir;
  }
  if (!pai) {
    free(no);
    return(q);
  }
  if (ch < pai->chave) pai->esq = q;
  else pai->dir = q;
  free(no);
  return(raiz);
}
```



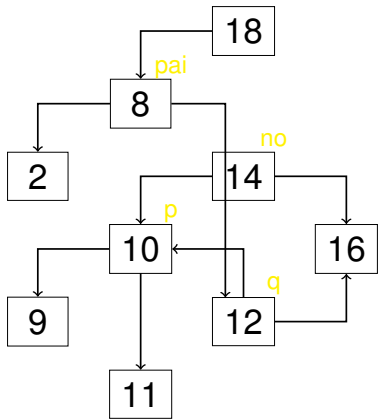
ABP – Remoção (Exemplo)

```
    if (p != no) {
      p->dir = q->esq;
      q->esq = no->esq;
    }
    q->dir = no->dir;
  }
  if (!pai) {
    free(no);
    return(q);
  }
  if (ch < pai->chave) pai->esq = q;
  else pai->dir = q;
  free(no);
  return(raiz);
}
```



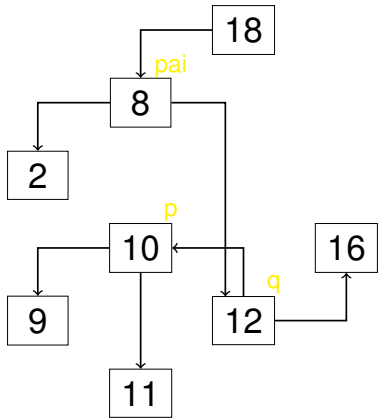
ABP – Remoção (Exemplo)

```
    if (p != no) {
      p->dir = q->esq;
      q->esq = no->esq;
    }
    q->dir = no->dir;
  }
  if (!pai) {
    free(no);
    return(q);
  }
  if (ch < pai->chave) pai->esq = q;
  else pai->dir = q;
  free(no);
  return(raiz);
}
```



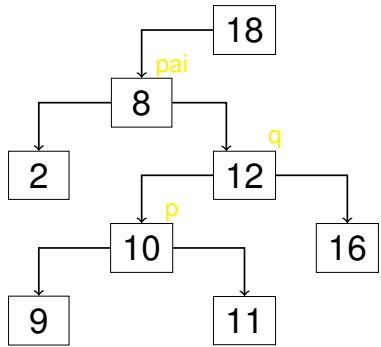
ABP – Remoção (Exemplo)

```
    if (p != no) {
      p->dir = q->esq;
      q->esq = no->esq;
    }
    q->dir = no->dir;
  }
  if (!pai) {
    free(no);
    return(q);
  }
  if (ch < pai->chave) pai->esq = q;
  else pai->dir = q;
  free(no);
  return(raiz);
}
```



ABP – Remoção (Exemplo)

```
    if (p != no) {
      p->dir = q->esq;
      q->esq = no->esq;
    }
    q->dir = no->dir;
  }
  if (!pai) {
    free(no);
    return(q);
  }
  if (ch < pai->chave) pai->esq = q;
  else pai->dir = q;
  free(no);
  return(raiz);
}
```



ABP – Remoção

```
PONT removeNo(PONT raiz, TIPOCHAVE ch){
  PONT pai, no, p, q;
  no = buscaNo(raiz,ch,&pai);
  if (no==NULL) return(raiz);
  if (!no->esq || !no->dir ) {
    if (!no->esq) q = no->dir;
    else q = no->esq;
  }
  else {
    p = no;
    q = no->esq;
    while (q->dir) {
      p = q;
      q = q->dir;
    }
```

Tratamos o caso do nó removido ter no máximo um filho

ABP – Remoção

```
PONT removeNo(PONT raiz, TIPOCHAVE ch){
  PONT pai, no, p, q;
  no = buscaNo(raiz,ch,&pai);
  if (no==NULL) return(raiz);
  if (!no->esq || !no->dir ) {
    if (!no->esq) q = no->dir;
    else q = no->esq;
  }
  else {
    p = no;
    q = no->esq;
    while (q->dir) {
      p = q;
      q = q->dir;
    }
```

Tratamos o caso do nó removido ter no máximo um filho

Ou de ter 2 filhos

ABP – Remoção

```
    if (p != no) {
      p->dir = q->esq;
      q->esq = no->esq;
    }
    q->dir = no->dir;
  }
  if (!pai) {
    free(no);
    return(q);
  }
  if (ch < pai->chave) pai->esq = q;
  else pai->dir = q;
  free(no);
  return(raiz);
}
```

ABP – Remoção

Do nó removido ser a raiz

```
    if (p != no) {
      p->dir = q->esq;
      q->esq = no->esq;
    }
    q->dir = no->dir;
  }
  if (!pai) {
    free(no);
    return(q);
  }
  if (ch < pai->chave) pai->esq = q;
  else pai->dir = q;
  free(no);
  return(raiz);
}
```

ABP – Remoção

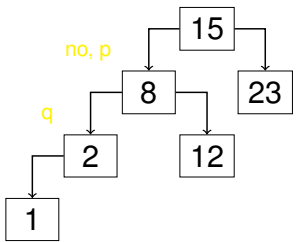
Do nó removido ser a raiz

Ou de não ser a raiz

```
    if (p != no) {
      p->dir = q->esq;
      q->esq = no->esq;
    }
    q->dir = no->dir;
  }
  if (!pai) {
    free(no);
    return(q);
  }
  if (ch < pai->chave) pai->esq = q;
  else pai->dir = q;
  free(no);
  return(raiz);
}
```

ABP – Remoção

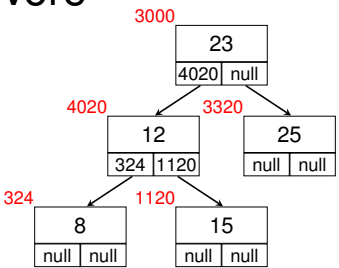
Além de tratarmos do caso do pai do substituto ser ou não o nó removido

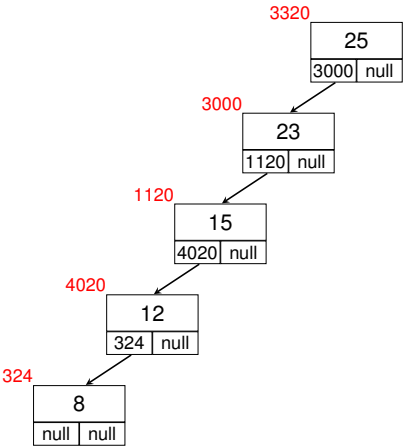



```
    if (p != no) {
      p->dir = q->esq;
      q->esq = no->esq;
    }
    q->dir = no->dir;
  }
  if (!pai) {
    free(no);
    return(q);
  }
  if (ch < pai->chave) pai->esq = q;
  else pai->dir = q;
  free(no);
  return(raiz);
}
```

ABP – Balanceamento

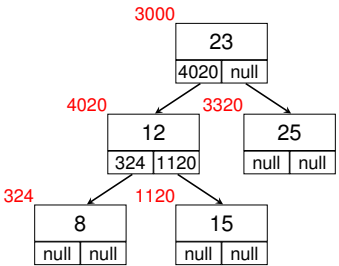
Vimos na Aula 16 que a ordem de inserção determina a forma da árvore

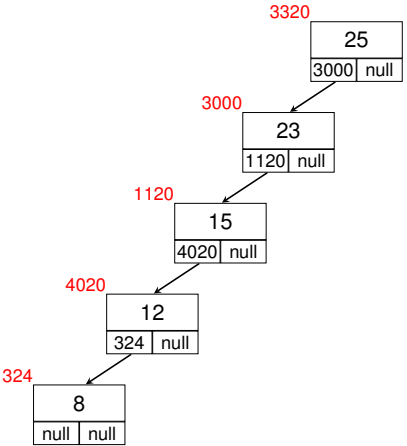




ABP – Balanceamento

E isso determina quão eficientes serão buscas na árvore

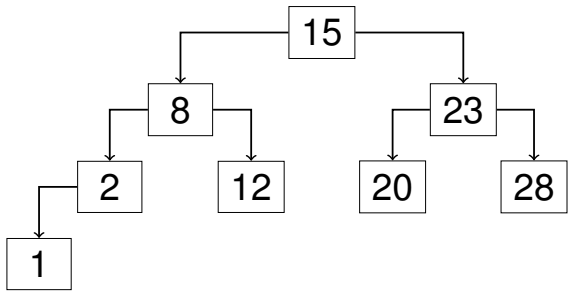




ABP – Balanceamento

Podemos ter então a eficiência de uma busca binária, caso a árvore esteja balanceada

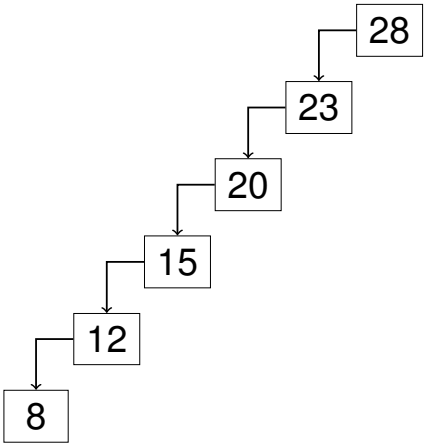
Com a vantagem de ser uma estrutura dinâmica



ABP – Balanceamento

Ou voltamos à busca sequencial, como em uma lista ligada

- Só que usando mais memória, pelo ponteiro extra



ABP – Balanceamento

Não veremos balanceamento agora...

ABP – Balanceamento

Não veremos balanceamento agora...

A boa notícia é que se os elementos que compõem a árvore forem obtidos aleatoriamente, espera-se um desempenho apenas 39% pior do que a árvore completamente balanceada

- Ou seja, a árvore em que as folhas aparecem no mesmo nível ou, no máximo, em dois níveis adjacentes

AULA 18
ESTRUTURA DE DADOS

Árvores Binárias de Pesquisa

Norton T. Roman & Luciano A. Digiampietri