Estruturas de Dados Avançadas (INF1010)

Árvores Binárias

Percurso de árvores binárias

- **pré-ordem**: raiz, sae, sad

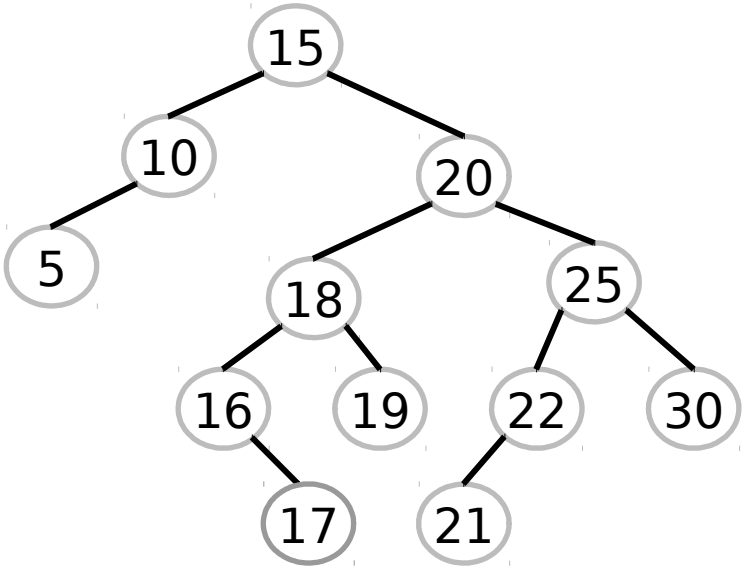
15	10	5	20	18	16	17	19	25	22	21	30

- **simétrica**: sae, raiz, sad

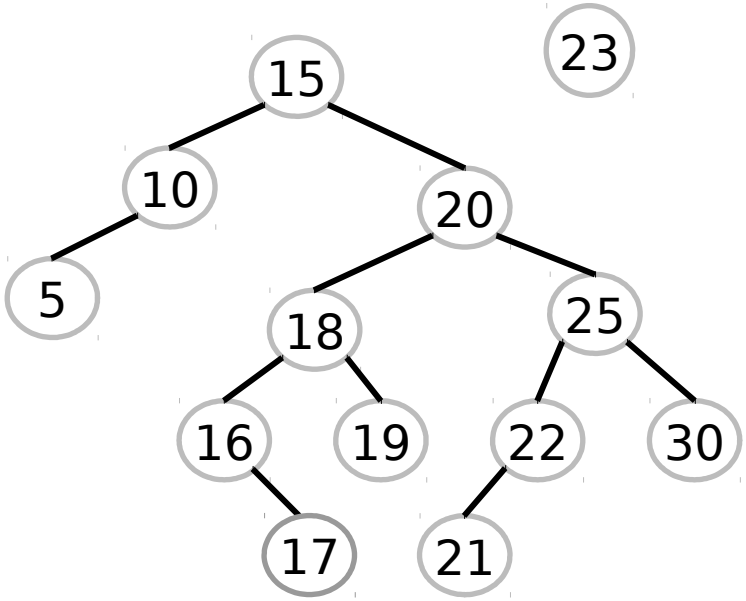
5	10	15	16	17	18	19	20	21	22	25	30

- **pós-ordem**: sae, sad, raiz

5	10	17	16	19	18	21	22	30	25	20	15



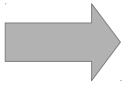
Inserção em uma ABB



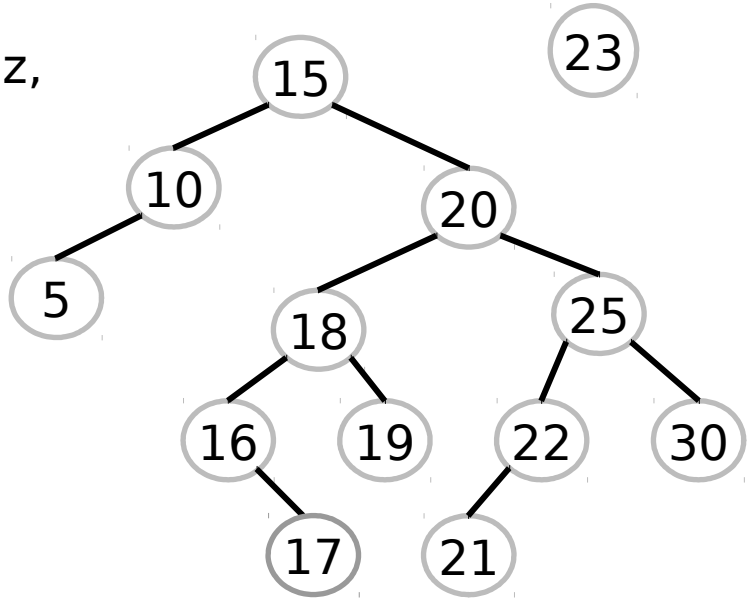
DEPARTAMENTO
DE INFORMÁTICA
PUC·RIO

Inserção em uma ABB

1. Se a árvore for **vazia** então crie um novo nó e retorne este nó
2. Se a chave for **menor** que a chave da raiz, insira na **sub-árvore à esquerda**
3. Se a chave for **maior** que a chave da raiz, insira na **sub-árvore à direita**



valor de retorno é o eventual novo nó raiz!

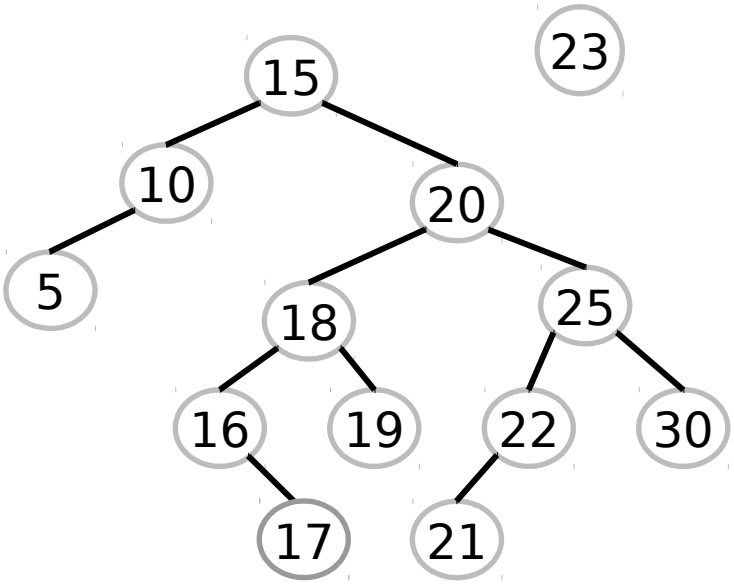


Função auxiliar para criar um nó

```
static Mapa* cria_no (int c, tdados* d) {
    Mapa* no = (Mapa*) malloc(sizeof(Mapa));
    if (no != NULL) {
      no->chave = c;
      no->dados = d;
      no->esq =  no->dir = NULL;
    }
    return no;
}
```

Implementando inserção recursiva

```
Mapa* insere (Mapa* r, int c, tdados *dados) {
  if (r==NULL)
      return cria_no(c, dados);
  else if (c < r->chave)
      r->esq = insere(r->esq,c,dados);
  else  /* if (c > r->chave) */ {
      r->dir = insere(r->dir,c,dados);
  }
  return r;
}
```

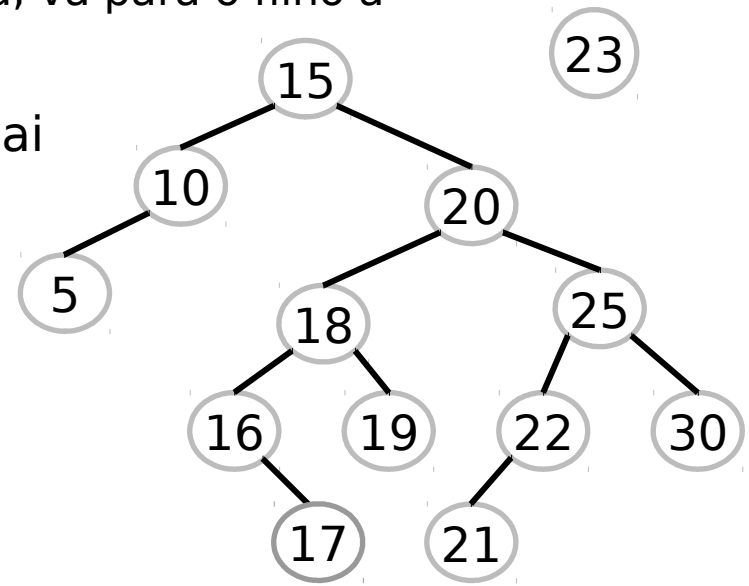


valor de retorno é o eventual novo nó raiz!

Inserção iterativa em uma ABB

1. Se a árvore for vazia, então crie um nó e retorne este nó
2. Senão, começe pelo nó raiz, e desça na árvore mantendo o nó anterior (pai)
3. Enquanto o nó não for **NULL** ou não contiver a chave dada:
   1. Se a chave do nó for **maior** que a chave dada, vá para o filho à **direita**
   2. Se a chave do nó for **menor** que a chave dada, vá para o filho à **esquerda**
4. Crie um novo nó e atualize a sub-árvore do pai

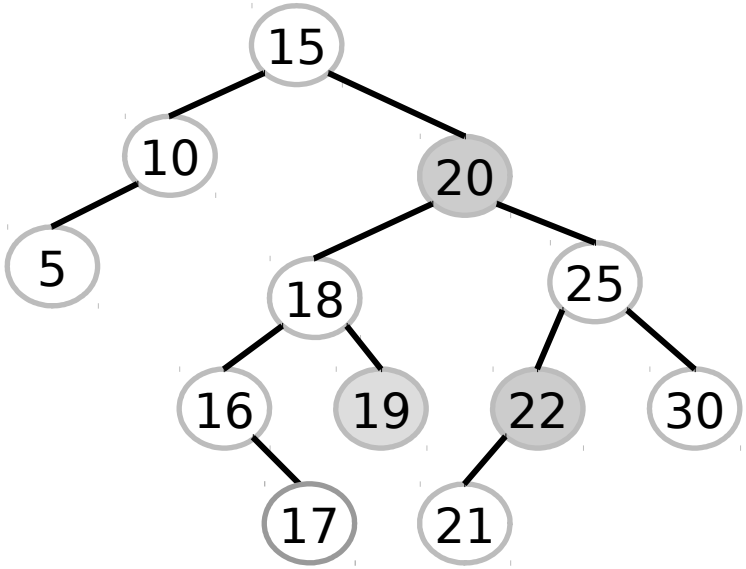
   (a raiz não mudou, retorne a raiz)



DEPARTAMENTO DE INFORMÁTICA
PUC·RIO

Remoção em uma ABB

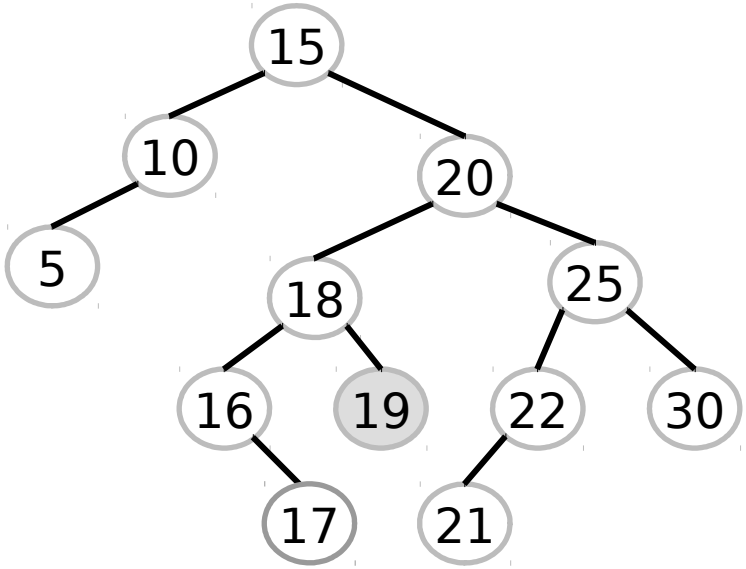
1. Se a árvore for **vazia** nada a ser feito
2. Se a chave for **menor** que a chave da raiz, remova da **sub-árvore à esquerda**
3. Se a chave for **maior** que a chave da raiz, remova da **sub-árvore à direita**
4. Se a chave for **igual** à chave da raiz



- nó folha
- nó possui uma sub-árvore
- nó possui duas sub-árvores

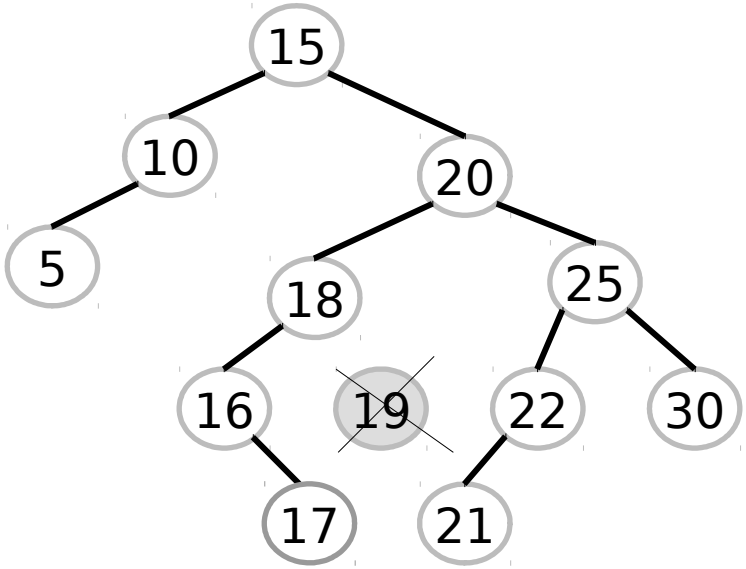
Remoção em uma ABB

1. Nó folha



Remoção em uma ABB

1. Nó folha
   - simplesmente elimine o nó

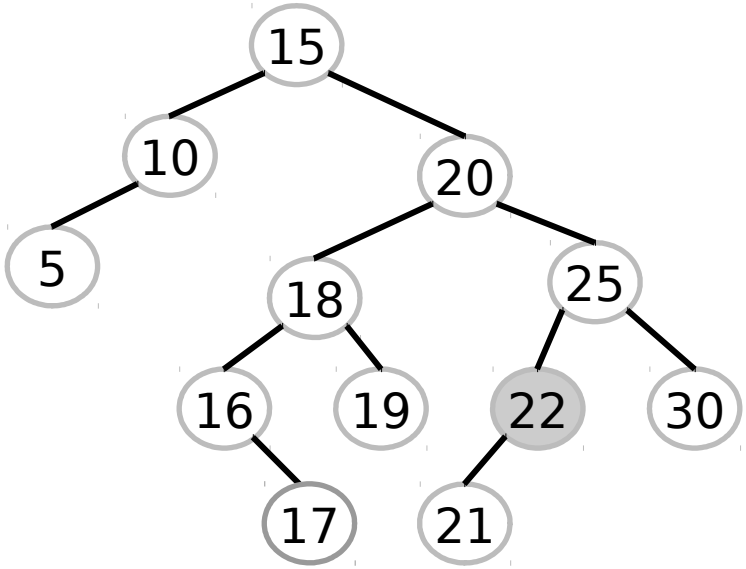


valor de retorno é o eventual novo nó raiz!

DEPARTAMENTO DE INFORMÁTICA
PUC·RIO

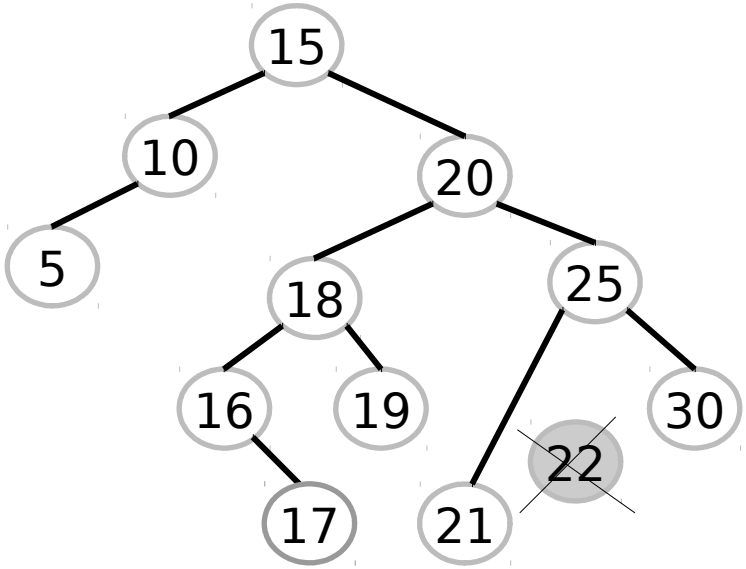
Remoção em uma ABB

2. Nó possui uma sub-árvore



Remoção em uma ABB

2. Nó possui uma sub-árvore
   - promove a sub-árvore

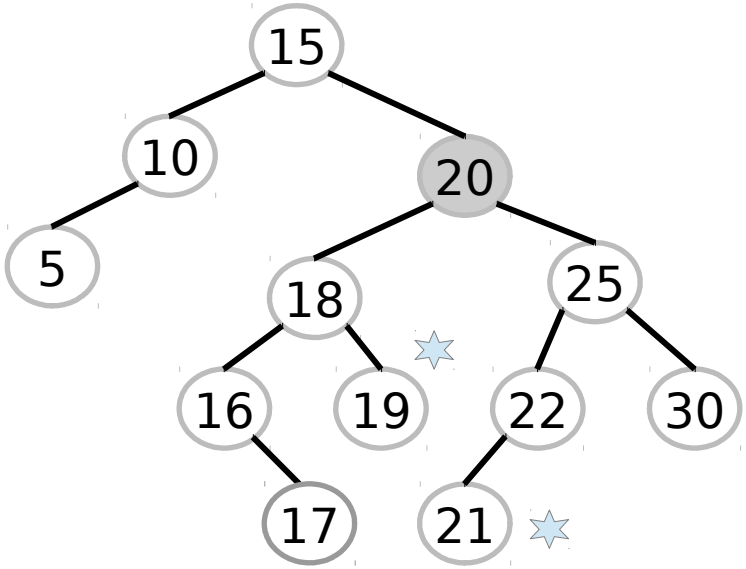


valor de retorno é o eventual novo nó raiz!

DEPARTAMENTO DE INFORMÁTICA
PUC·RIO

Remoção em uma ABB

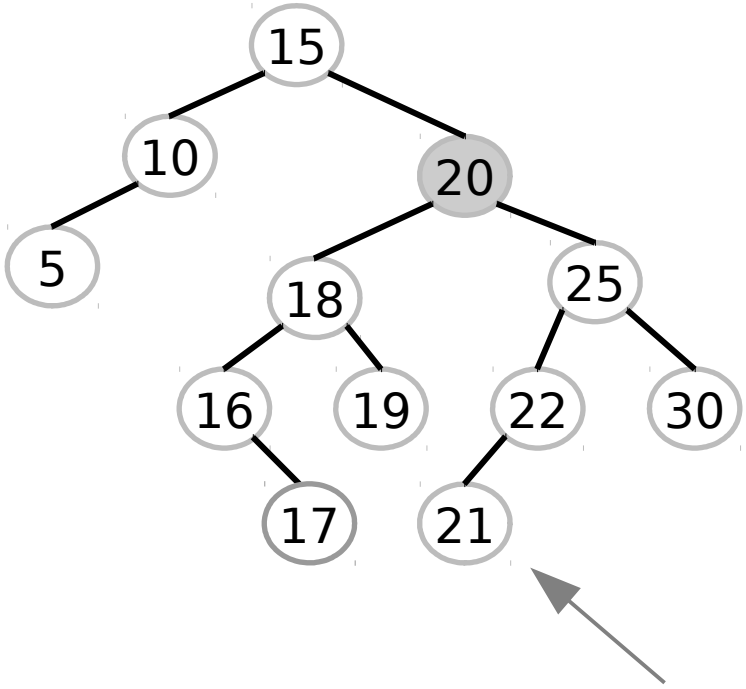
3. Nó possui duas sub-árvores
   - nó predecessor
   - nó sucessor



Remoção em uma ABB

3. Nó possui duas sub-árvores

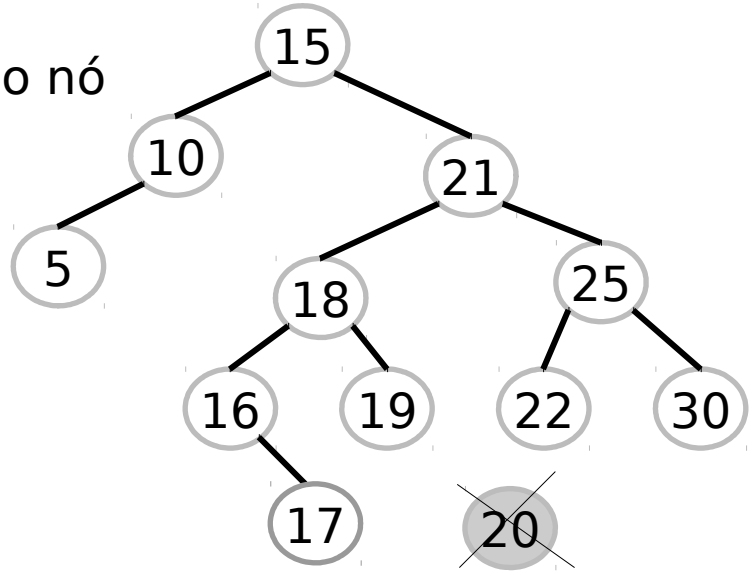
quem é o
sucessor de 20?



menor chave da
sub-árvore direita!

Remoção em uma ABB

3. Nó possui duas sub-árvores
   - coloque a informação do sucessor no nó
   - **remova** o sucessor



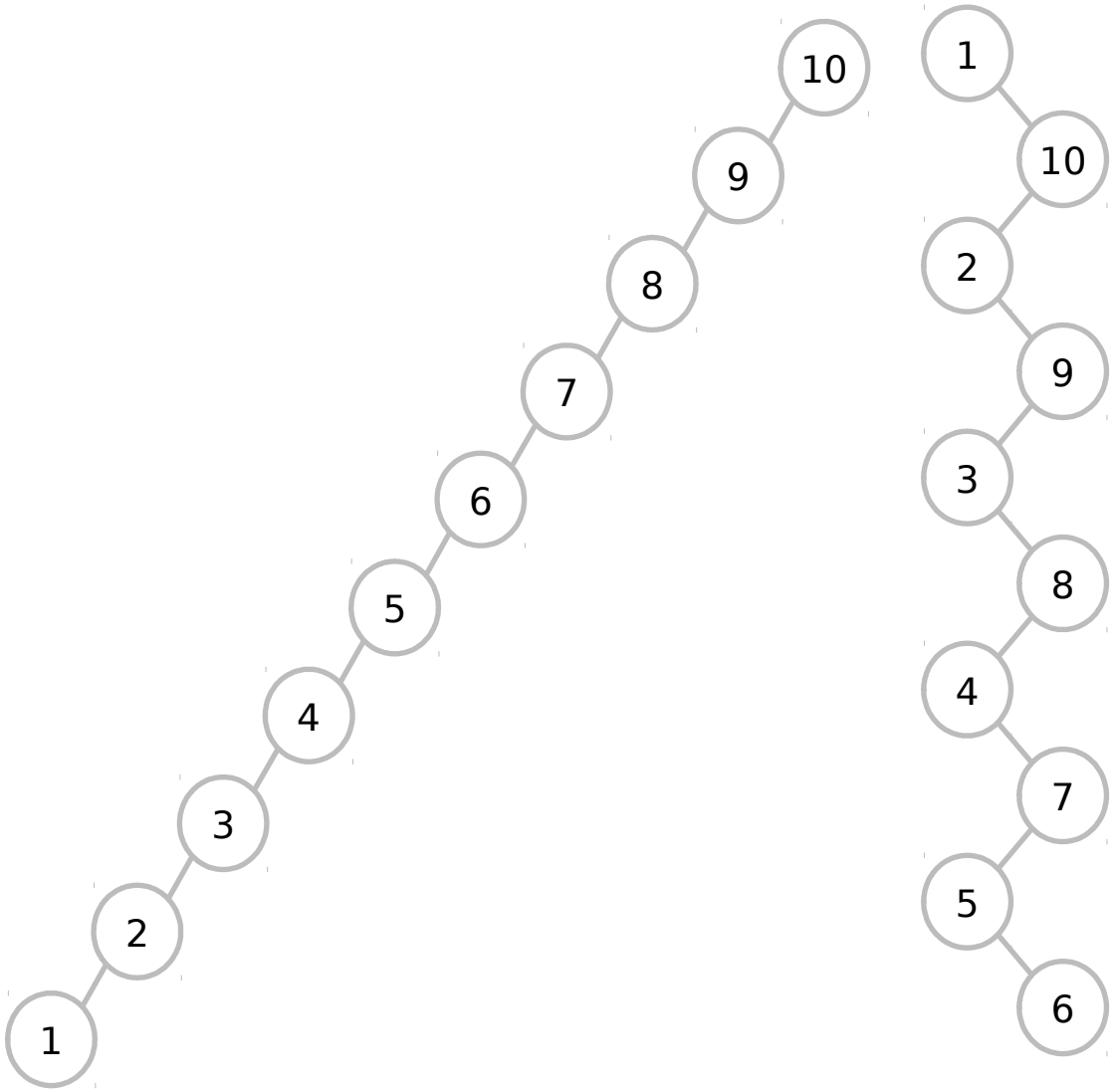
um nó com 0 ou uma sub-árvore!

valor de retorno é o eventual novo nó raiz!

Implementando remoção recursiva

```
Mapa* retira (Mapa* r, int chave){  ESBOÇO
   Mapa *sucessor, *t;
   if (r == NULL) return NULL;
   else if (chave < r->chave)
      r->esq = retira(r->esq, chave);
   else if (chave > r->chave)
      r->dir = retira(r->dir, chave);
   else {          /* achou o nó a remover */
      if (r->esq == NULL && r->dir == NULL) { /* nó sem filhos */
          free (r); r = NULL;
      }
      else if (r->esq == NULL) {/* nó só tem filho à direita */
         t = r; r = r->dir; free (t);
      }
      else if (r->dir == NULL) {/* só tem filho à esquerda */
         t = r; r = r->esq; free (t);
      }
      else {/* nó tem os dois filhos: busca o sucessor */
         sucessor = r->dir;
         while (sucessor->esq) sucessor = sucessor->esq;
         r->chave = sucessor->chave; sucessor->chave = chave; …
         r->dir = retira(r->dir, chave);
      }
   }
   return r;
}
```

Árvore Binária de Busca Degenerada



- nós não terminais têm 1 filho
- estrutura hierárquica "degenera" em "linear"

DEPARTAMENTO DE INFORMÁTICA
PUC·RIO

Árvore Binária de Busca Balanceada

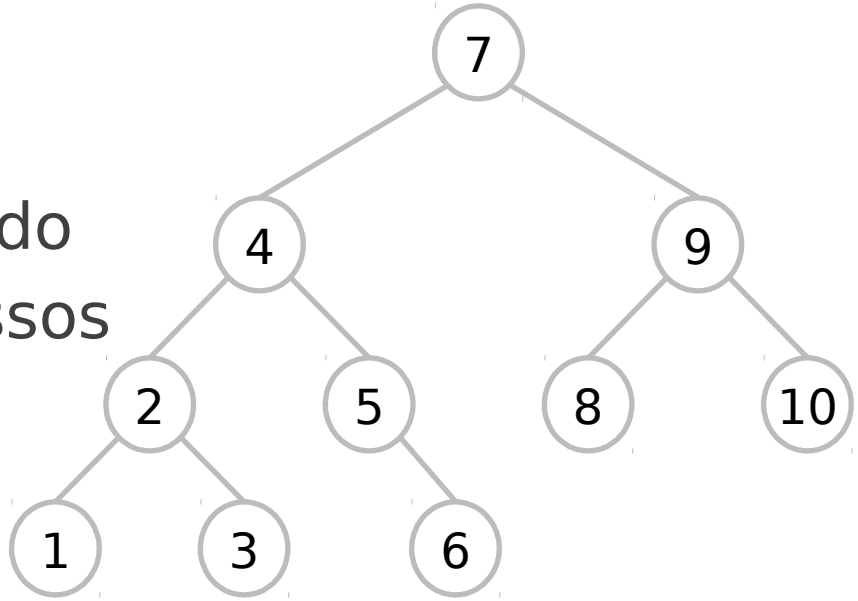
$|he-hd| \leq 1$

- he = altura da sub-árvore esquerda
- hd = altura da sub-árvore direita

→ qualquer nó pode ser alcançado a partir da raiz em O(log(n)) passos

→ (quase) todos os nós não terminais têm dois filhos

→ distribuição de nós próxima à de uma árvore cheia (altura mínima)



Balanceamento

Se há **m** elementos na sae e **n** na sad e **n > m + 1** movemos o valor da raiz para a sae e o menor elemento da sad para a raiz

→ o procedimento para m > n+1 é semelhante

→ procedimento é repetido até | n – m | <= 1

Processo deve ser repetido (recursivamente) para balancear as sub-árvores