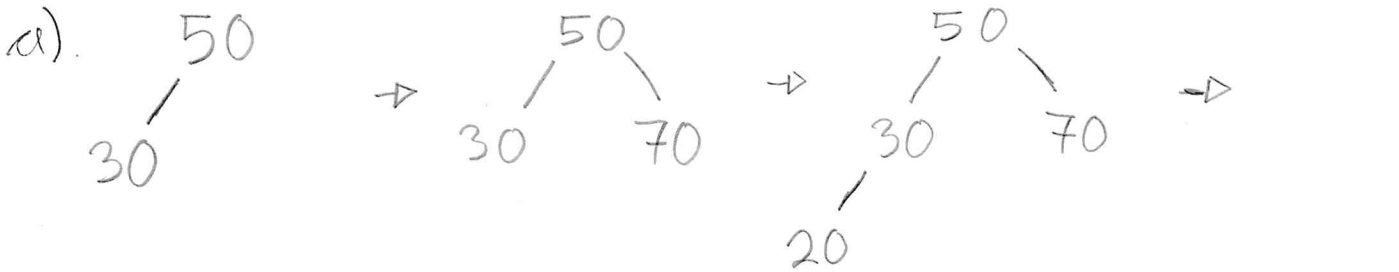
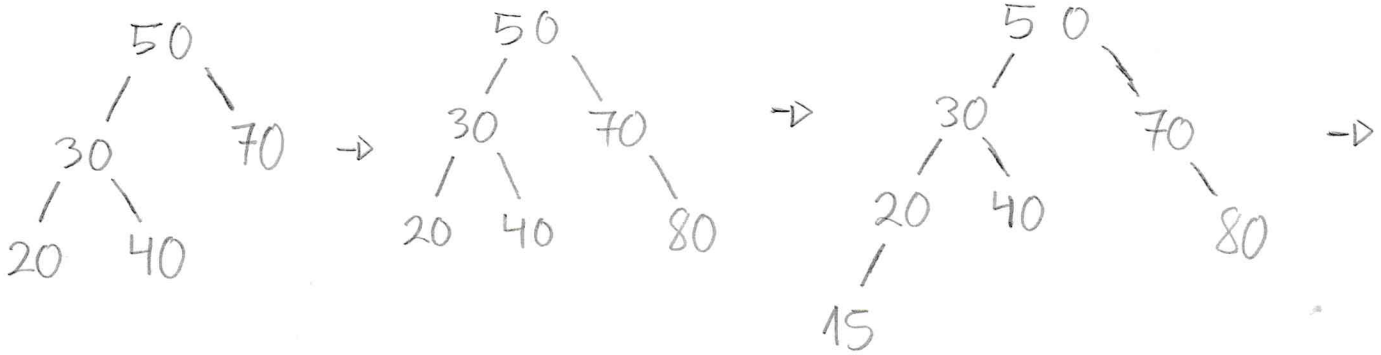
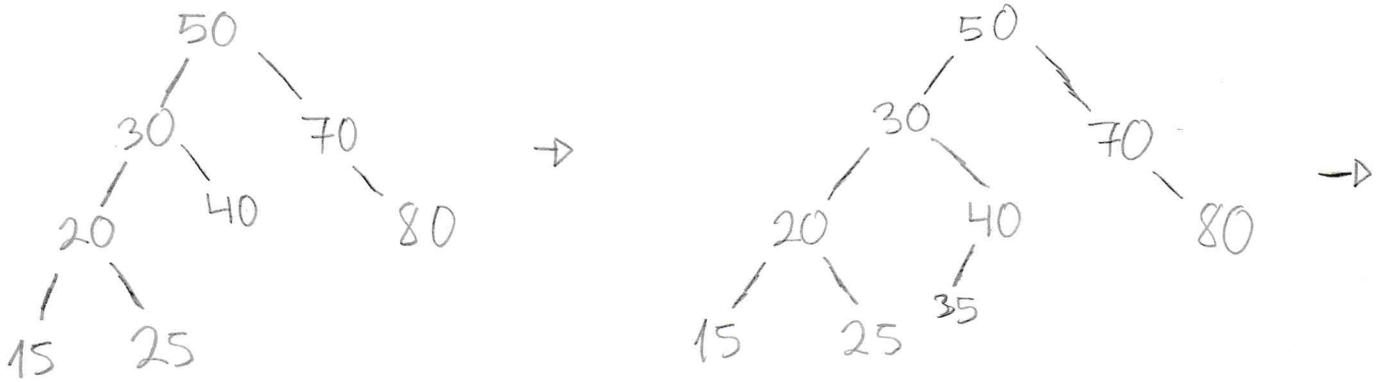
Lista 6 - AED II -

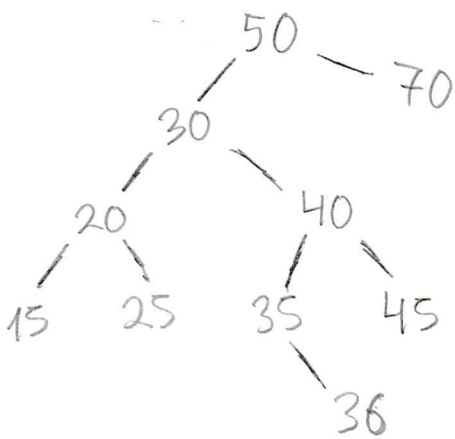
1- {50, 30, 70, 20, 40, 60, 80, 15, 25, 35, 45, 36}.

a)



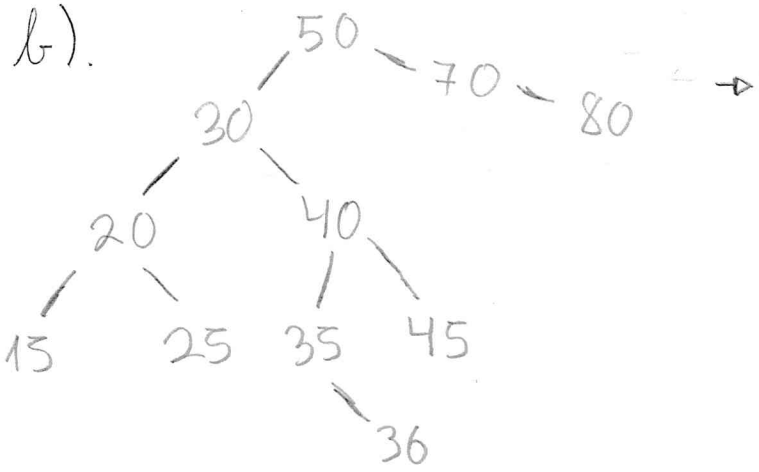




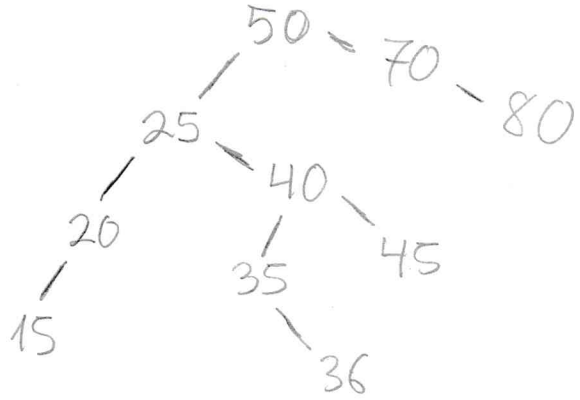


→ Que é a árvore resultante.

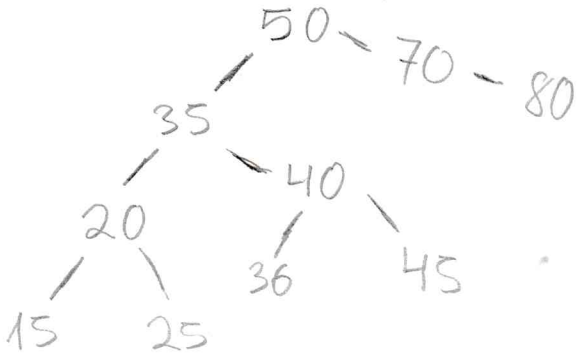
b).



→ Registro mais à direita no sub esquerdo



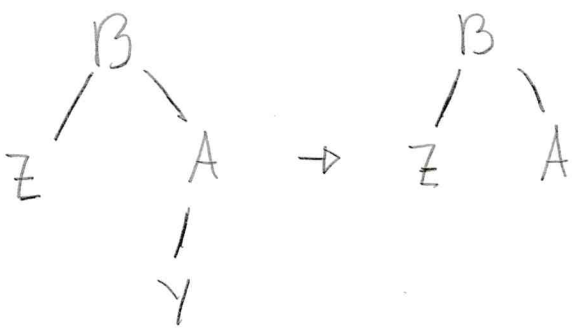
→ Registro mais à esquerdo no sub direito



2- Tomemos uma árvore dada por:

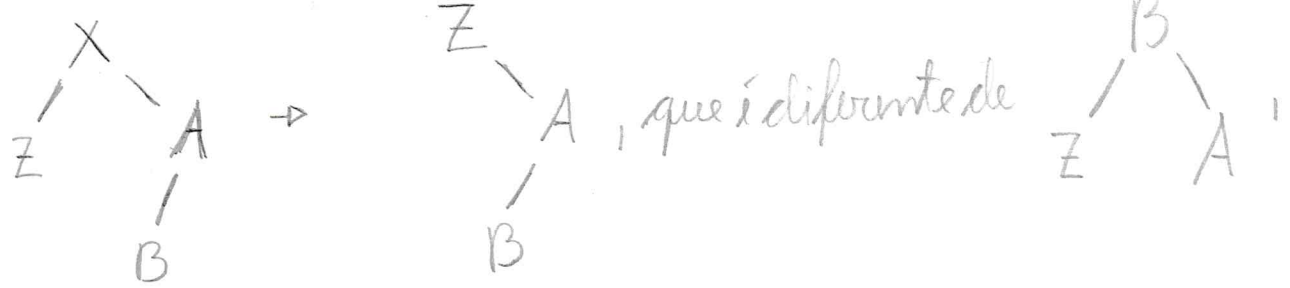
, como podemos buscar o registro mais a direita na subárvore esquerda ou o registro mais a esquerda na subárvore direita, tomaremos primeiro a lógica do registro na sub direita:

→ Dir: X e dps Y



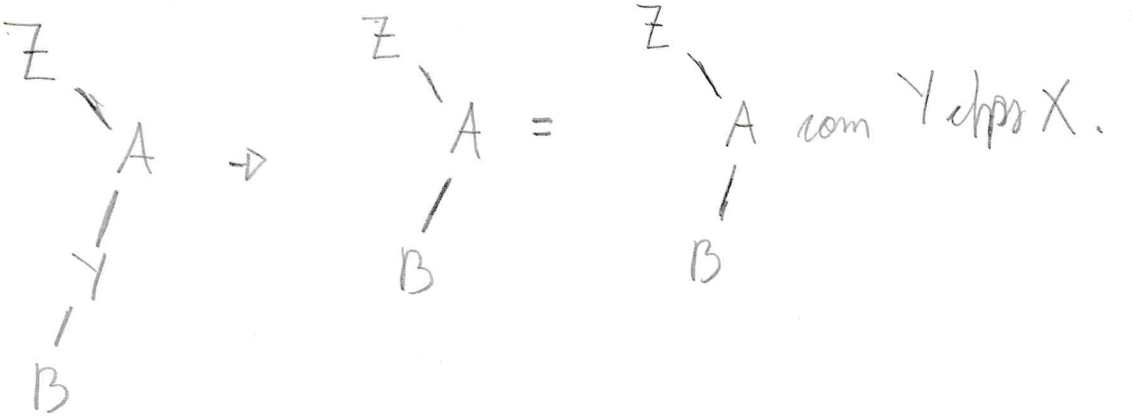
, agora tomaremos a lógica do mais à direita no sub esquerdo:

↳ Esq: Y dps X



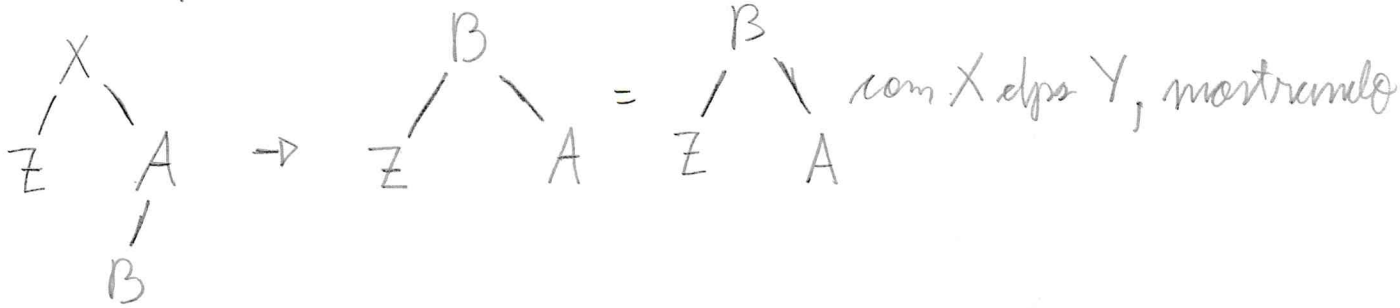
portanto, a operação de remoção não é comutativa. A base da demonstração é a sentença lógica que permite duas operações distintas dentro de remoção. Se só tomarmos a primeira lógica, teremos:

↳ Esq: X dps Y



Para a segunda lógica, temos:

↳ Dir: Y e dps X.



que a mudança de lógica é vital para a demonstração.

3 - Para o algoritmo de remoção, temos três casos de nó:

1º - O nó é uma folha e não possui filhos

2º - O nó possui um filho (Direito ou esquerdo)

3º - O nó possui dois filhos.

Portanto, teremos (usando recursão):

```
Arv retira(Arv r, int v){
  if(r==NULL)  // caso base caso não encontre.
        return NULL
  else if(r->info > v) // Usar a estrutura de dados ao nosso favor.
        r->esq=retira(r->esq, v);
  else if(r->info < v)
        r->dir = retira(r->dir, v);
  else { // Achou o elemento!!
     if(r->esq == NULL && r->dir==NULL){ // 1º caso
       free(r);
       r=NULL; }
     else if(r->esq == NULL) { // 2º caso no direito
       Arv novo = r;
       r=r->dir;
       free(novo); }
     else if(r->dir == NULL) { // 2º caso no esquerdo
       Arv novo = r;
       r=r->esq;
       free(novo); }
```

```
else { // 3º caso
    Arv pai = r, filho = r->esq;
    while(filho->dir != NULL) { // busco o maior da esq no sub
árvore à direita.
        pai = filho;
        filho = filho->dir; }

// troco as infos.
    r->info = filho->info;
    if(pai == r) // quando o pai é o nó que tem que ser removido.
        pai->esq = filho->esq;
    else
        pai->dir = filho->esq;
    free(filho);
    }
    }
    return r;
}
```

o algoritmo escrito foi dado em uma aula da professora Daniela Musa em AED-I e segue o lógico dado pela explicação inicial. Qualquer alteração é devido pela adaptação ao papel, entretanto a lógica segue inalterada.