Resumo -> AQUI <-

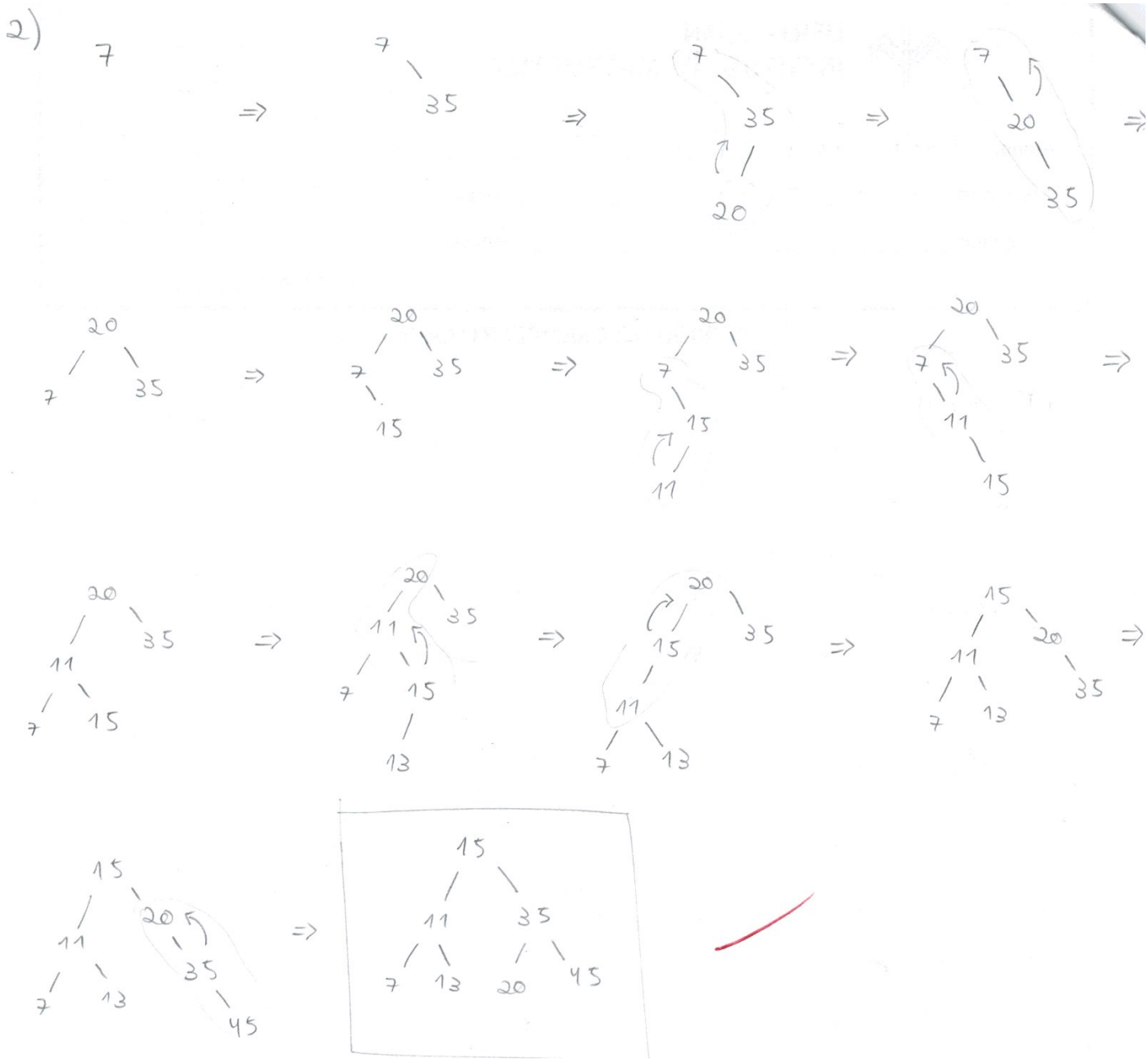
Prova 1 - 2013/2 - Estrutura de Dados

1 - Considere uma árvore binária (não necessariamente uma árvore binária de busca). Escreva um algoritmo de tempo O(n) que preencha, para cada nó da árvore, um campo no->SumDir, com o valor da soma das chaves na sub-arvore direita do nó.

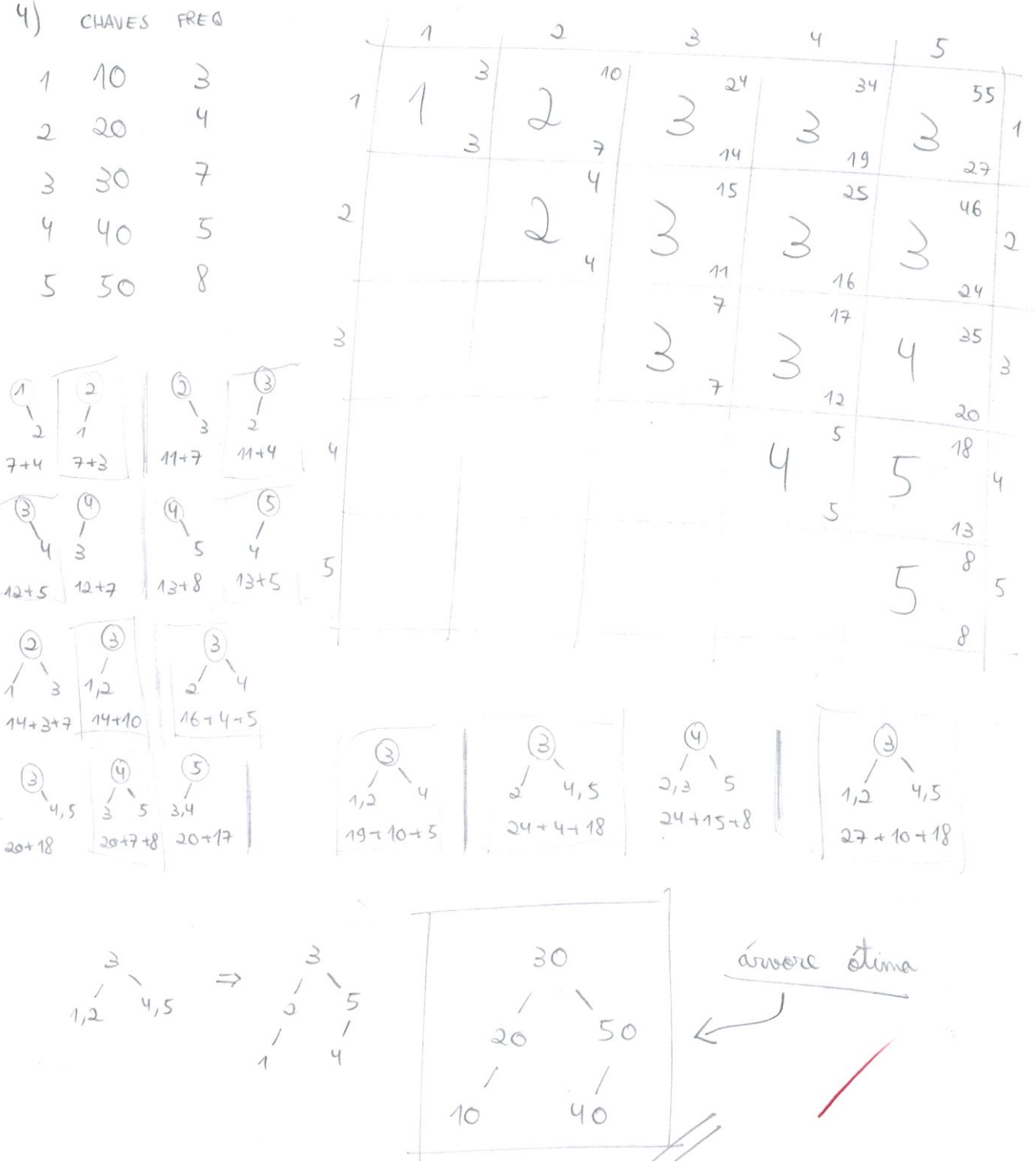
```
int soma(tno* no) {

   int sumesq, sum dir;

   if(no==NULL)
         return 0;

   sumdir=soma(no->dir);
   sumesq=soma(no->esq);
   no->SumDir=sumdir;

   return (sumdir + no-> valor + sumesq);

}
```

Chamou a função soma acima passando o ponteiro para a raiz da árvore. Dessa forma, o campo SumDir de todos os nós será preenchido.

2 - Insira em uma árvore AVL inicialmente vazia, as seguintes chaves, nesta ordem: **7, 35, 20, 15, 11, 13 e 45. Mostre a árvore antes e após qualquer rotação!**



4 - Calcule a árvore ótima para as chaves 10, 20, 30, 40 e 50, com frequências de acesso respectivamente 3, 4, 7, 5 e 8. Mostre a tabela que o algoritmo preenche para calcular a árvore, bem como a árvore obtida.



OBS: Questões 3 e 5 dessa prova falavam, respectivamente, de árvore rubro-negra e heaps binárias, coisa que ele não comentou ainda.

Prova 1 - 2012/2 - Estrutura de Dados

1.1- Escreva o algoritmo de remoção de um nó em uma lista duplamente encadeada, ordenada, circular com nó cabeça.

```
Remove(ptcab,x){
        ptcab -> chave=x;
        pt  = ptcab -> prox;
        Enquanto(pt->chave>x){
                pt=pt->prox;
                Se(pt->chave<x) e (pt!=ptcab){
                        ptant=pt-> ant;
                        ptprox=pt->prox;
                        ptant->prox=ptprox;
                        ptprox->ant=ptant;
                        free(pt);
                }
                Senão{
                        printf("Erro!");
                }
        }
}
```

1.2- Considerando a lista do item 1.1, mostre como fica a lista com a inserção de chaves 2,4,1 e 3, nesta ordem (desenhe a representacao da lista).

2- Considere uma árvore binaria com raíz raiz, onde cada no contem campos esq, dir, chave, e prod.

2.1 - Escreva um algoritmo eficiente que preencha, para cada no' da arvore, o campo no->prod (inicialmente não preenchido), com o valor do produto da chave do no' pelo produto das chaves menores que a chave do no, que são descententes do no' na arvore. (peça ao professor um exemplo se você não entendeu).

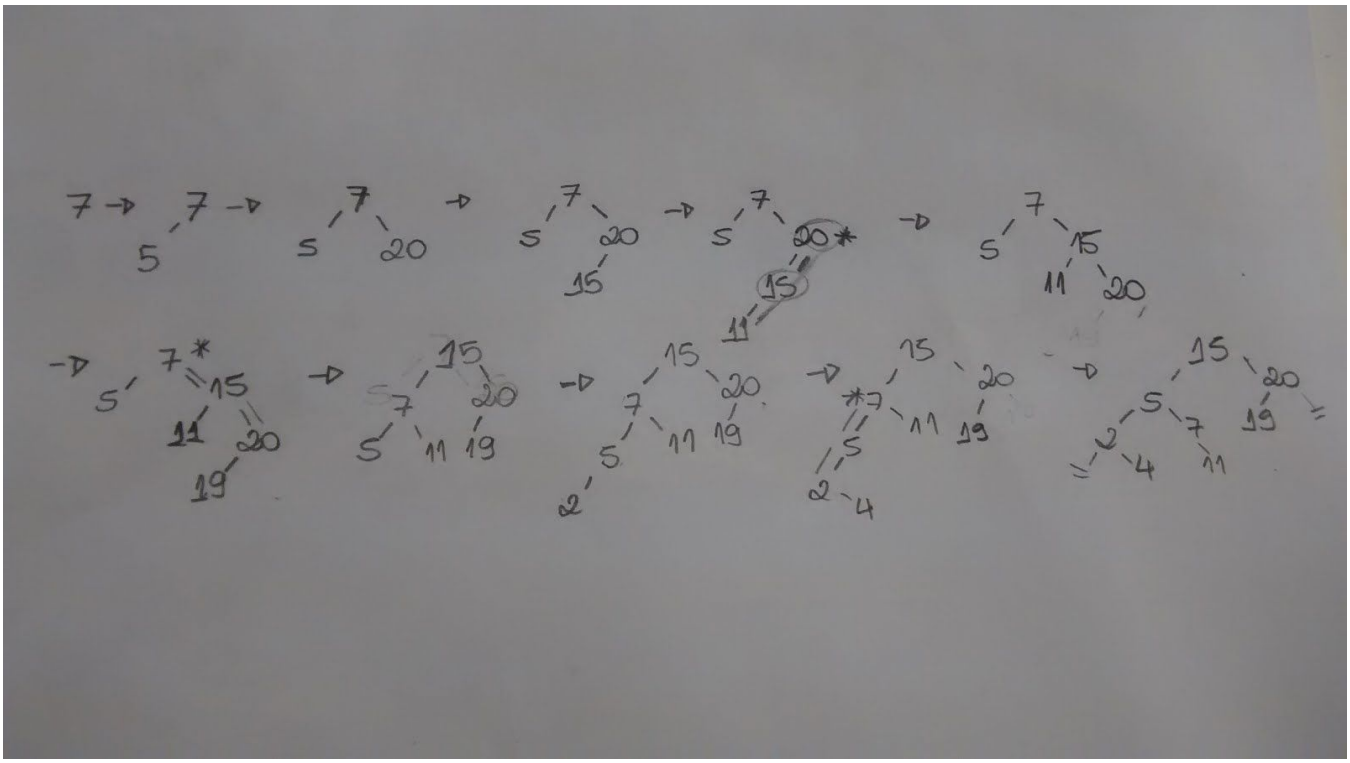
NÃO SEI SE TA CERTO 'SA PORRA.

```
Prod (no) {
  If (no == null) { return 0 }
  else {
    prodesq = Prod (no->esq);
    proddir = Prod (no->dir);
    no->prod = prodesq;}
```

Retornar a multiplicação entre prodesq, proddir e no QUE NÃO FOR(EM) 0.}

2.2- Qual a complexidade de seu algoritmo.

3 - Insira em uma árvore AVL inicialmente vazia, as seguintes chaves, nesta ordem: 7, 5, 20, 15, 11, 19, 2 e 4. **Mostre a arvore antes e após qualquer rotacao!**



4 – Calcule a árvore ótima para as chaves 10, 20, 30, 40 e 50, com frequências de acesso respectivamente 3, 14, 7, 8 e 10. Mostre a tabela que o algoritmo preenche para calcular a arvore, bem como a arvore obtida.

Prova 1 - 2012/1 - Estrutura de Dados

1 - Escreva o algoritmo de remoção de uma chave x de uma lista duplamente encadeada, ordenada, circular e com nó cabeça.

```
remover(x, ptcab){
        ptcab->chave=x
        pt=ptcab->prox
        enquanto pt->chave>x{
                pt=pt->prox
                se
        }
}
```

2 - Construa a árvore ótima com chaves 1, 2, 3, 4 e 5 e frequências de acesso respectivamente iguais a 3,7,11,19 e 30. Mostre a tabela preenchida pelo algoritmo de construção da árvore ótima, e a árvore obtida.

3 - Insira as chaves nesta ordem, em uma árvore AVL inicialmente vazia: 20, 40, 67, 45, 55, 60, 10 e Mostre a árvore antes e após cada rotação efetuada.

Prova 1 - 2011/1 - Estrutura de Dados

Nas questões 1 e 2 considere uma árvore binaria com raíz raiz, onde cada no contem campos esq, dir, chave, e sumesq.

1 - Escreva um algoritmo eficiente que preencha, para cada no' da arvore, o campo no->sumesq (inicialmente não preenchido), com o valor da soma das chaves na sub-arvore esquerda do no'. (peça ao professor um exemplo se você não entendeu). Qual a complexidade de seu algoritmo.

2 - Escreva um algoritmo de insercao de uma chave nesta arvore, que atualize adequadamente o campo sumesq de cada nó.

4 – Calcule a árvore ótima para as chaves 10, 20, 30, 40 e 50, com frequências de acesso respectivamente 2, 5, 3, 4 e 10. Mostre a tabela que o algoritmo preenche para calcular a arvore, bem como a arvore obtida.

Prova 1 - 2010/2 - Estrutura de Dados

1.1- Escreva o algoritmo de inserção de um nó em uma lista duplamente encadeada, ordenada com nó cabeça.

1.2 – Qual a complexidade de seu algoritmo ?

2- Considere uma árvore binaria com raíz raiz, onde cada no contem campos esq, dir, chave, e

sum.

2.1 - Escreva um algoritmo eficiente que preencha, para cada no' da arvore, o campo no->sum (inicialmente não preenchido), com o valor da soma das chaves na sub-arvore do no'. (peça ao professor um exemplo se você não entendeu).

2.2- Qual a complexidade de seu algoritmo.

3 - Insira em uma árvore AVL inicialmente vazia, as seguintes chaves, nesta ordem: 17, 35, 20, 15, 11, 19, 5 e 4. Mostre a arvore antes e após qualquer rotacao!

4 – Calcule a árvore ótima para as chaves 10, 20, 30, 40 e 50, com frequências de acesso respectivamente 13, 14, 7, 15 e 10. Mostre a tabela que o algoritmo preenche para calcular a arvore, bem como a arvore obtida.

5 – Rode o seguinte algoritmo em uma lista simplesmente encadeada L com os valores 1, 2 , 4, 3, 7, 8 e 9 nesta ordem. Explique com suas palavras o que o algoritmo faz, e desenhe a estrutura resultante.

```
OqueFaz( ) {
        if (L != NULL) {
                L1 = L;
                L2 = L->prox;
                pt = L;

                enquanto pt != NULL
                        prox = pt->prox;
                        if (prox != NULL){
                                pt->prox = prox->prox;
                        }
        pt = prox;
        }
```

```
else {
        L2 = NULL;
}
```

Prova Antigonas(2003-09) - Estrutura de Dados

Peguei as questões que ele já cobrou, mas que faz alguns anos que não cobra na prova. Mas CASO COBRAR:

1 - Desenhe uma lista vazia, e após a inserção de chaves 15, 42 e 12.

2 - Qual a complexidade da operação de busca em uma árvore AVL ? Explique por que uma árvore AVL pode ser considerada melhor que uma árvore binária de busca comum.

3 - a) Escreva um algoritmo de busca de um elemento em uma lista encadeada, ordenada, sem nó cabeça.

b) Escreva também um algoritmo de inserção de uma chave x nesta lista.

c) Escreva um algoritmo que imprima apenas os elementos em posição par (Contada a partir do fim da lista!). Ou seja, se a lista tem 6 elementos, 1, 2, 3 ,4 ,5 e 6, seu algoritmo deve imprimir os numeros 5, 3 e 1, nesta ordem.

4 - a) Escreva o algoritmo de inserção de elementos em uma pilha com alocação sequêncial. Não esquece que o seu algoritmo deve lidar corretamente com o overflow, de forma a permitir que sempre sejam inseridos novos elementos, se houver memória no sistema.

b) Escreva o algoritmo de inserção de elementos em uma pilha com alocação dinâmica.

```
/*Rotina que faz a inserção de elementos na pilha
O Parâmetro dado recebe um ponteiro para string
A função não retorna valor algum
*/
void push(char *dado) {
   alocar = (struct no *) malloc(sizeof(struct no)); //Faz alocação na memória
   if (!alocar) { //Se não for possível a alocação, sai do programa
      printf("Falta de memória");
      exit(0);
   }
   strcpy(alocar->dado, dado); //Copia o dado para o novo nó alocado
    if (!topo) { //Se não houver elemento ainda na pilha, insere
               //na base, apontando o ponteiro de topo para o único
               //elemento até então inserido
```

```
        topo = alocar;
        topo->proximo = NULL;
    }
    else //se não...
    {
        alocar->proximo = topo; //Aponta o próximo para o "antigo" topo da pilha
        topo = alocar;          //Aponta o ponteiro de topo para o dado que foi alocado

    }
}
```

Fonte: (http://www.vivaolinux.com.br/script/Pilha-algoritmos-push-pop-e-imprimir-explicados)

c) Qual a complexidade dos algoritmos dos itens (a) e (b). Não esqueça de considerar a possibilidade de overflow na lista (a)

5 - Escreva os algoritmos de inserção e remoção de chaves de uma fila implementada como lista encadeada.

5.2 - Qual a complexidade destes algoritmos?

6 - Escreva o algoritmo de busca de uma chave em uma árvore binária de busca.

6.1- Insira, em uma árvore binária de busca, inicialmente vazia, as seguintes chaves, nesta ordem: 10, 20, 30, 15, 45, 25 e 17.

6.2 - Da árvore obtida na questão anterior, remova o nó da chave 20.

7 - Qual a complexidade da operação inserção de chaves em uma árvore AVL?

8 - Escreva um algoritmo recursivo que calcule para cada subárvore de uma árvore o número de nós naquela subárvore. Seu algoritmo deve visitar cada nó apenas um número constante de vezes. Qual a complexidade de seu algoritmo?