QUESTÃO 1: PROVA1 ISA QUE / 11904702 8

```
Typedef struct _Tno {
  struct _Tno * ant;
  int char;
  struct _Tno prox; } Tno;
```

```
// Inicializa
init (Tno ** pTpTcab
(*pTpTcab) = (Tno *) malloc(sizeof (Tno))
(*pTpTcab) -> prox = (*pTpTcab) -> ant =
                    (*pTpTcab)
```

```
Tno * pTcab;
init (&pTcab)
init2(Tno * &pTcab)
pTcab = malloc(...)
pTcab -> prox = pTcab -> ant = pTcab
```

```
// Algoritmo de busca que retorna pT!
busca (pTcab, x)
pT = pTcab -> prox;
enquanto pT != pTcab & pT -> chave < x
    pT = pT -> prox
  return pT;
// Algoritmo que insere!
Insere (pTcab, x)
 pT = busca (pTcab, x),
  ant = pT -> ant
Se pT != pTcab & pT -> chave == x
      ckhui, já Tem
novo = malloc(...)
Se novo == null
   erro - acabou a memória
   retorna null
novo -> chave = x
novo -> prox = pT
novo -> ant = ant
ant -> prox = novo
pT -> ant = novo   return novo
```

A função busca retorna o ponteiro para o elemento que será o próximo da chave inserida.

A função Insere() retorna nula em caso de erro, ou o endereço do novo elemento inserido.

QUESTÃO 2:

2.1

```
Calcular(Tno** no, int ant)
  Se no == nulo
      retorna 0

  Se no.esq != nulo
      no.soma_prefixo = no.chave + Calcular(no.esq, ant)

  Se não
      no.soma_prefixo = no.chave + ant

  Total = no.soma_prefixo

    Se no.dir
        Total = Calcular(no.dir, Total)

  retorna Total;
```

(segundo)
ant → anterior
esq → esquerda
dir → direita

2.2

Não pode estar certo. A análise O(h) em circunstâncias normais leva em conta o pior caso. Se inserirmos uma chave que seja menor que todas as chaves da árvore, nós precisa atualizar soma_prefixo para todos os nós. Sendo assim, como tem-se que percorrer todos os n nós da árvore, o algoritmo deveria ser O(n).

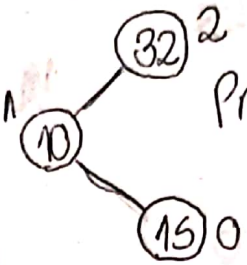
QUESTÃO 3

3.1

Inserindo 32

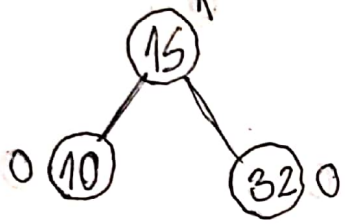
Inserindo 10

Inserindo 15

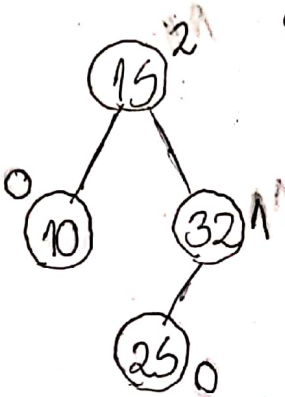


Problema encontrado
Não é AVL.

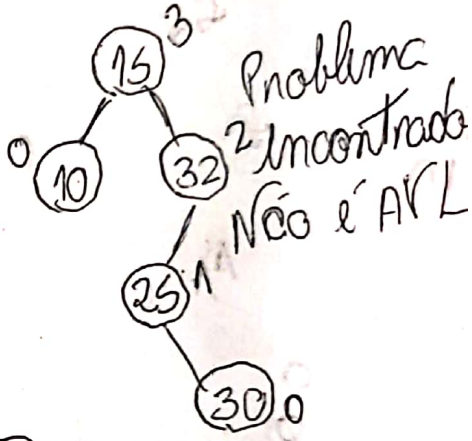
Realizando uma rotação dupla direita



Inserindo 25

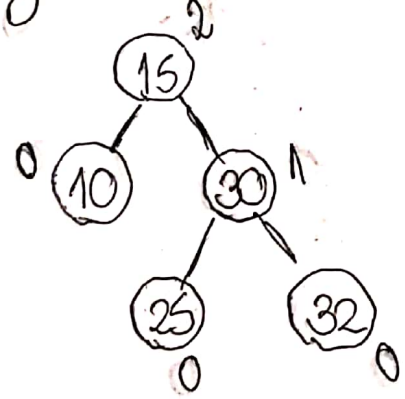


Inserindo 30



Problema Encontrado
Não é AVL

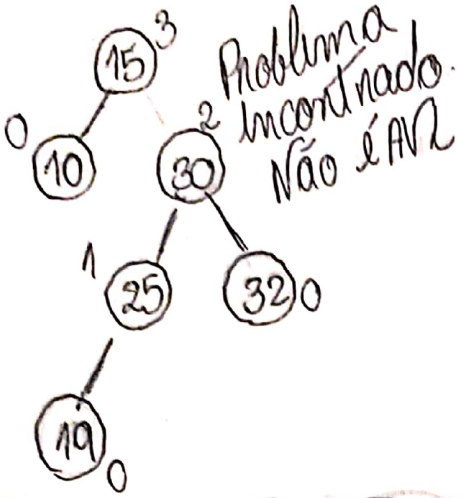
Realizando uma rotação dupla esquerda.



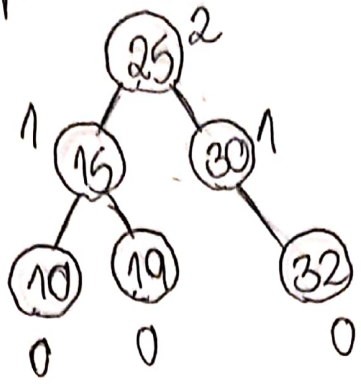
FINAL!

3.2:

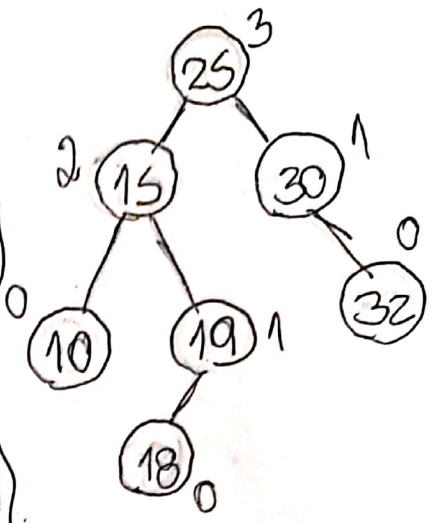
Inserindo 19



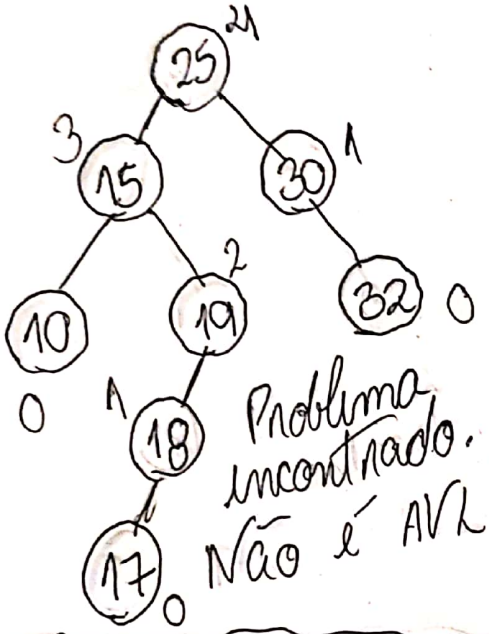
Realizando rotação dupla esquerda



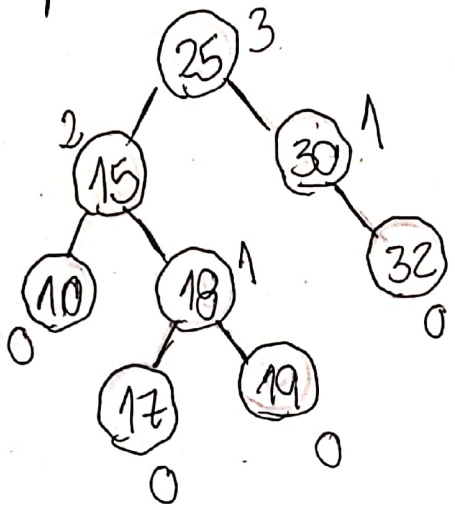
Inserindo 18



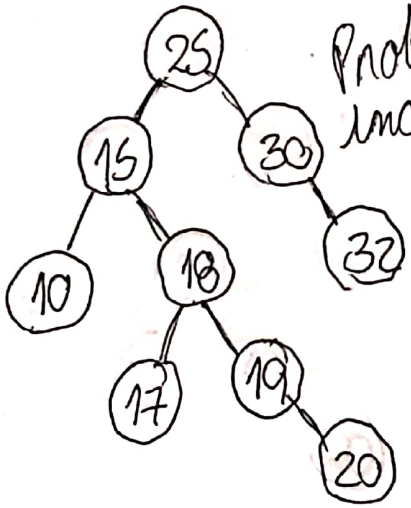
Inserindo 17



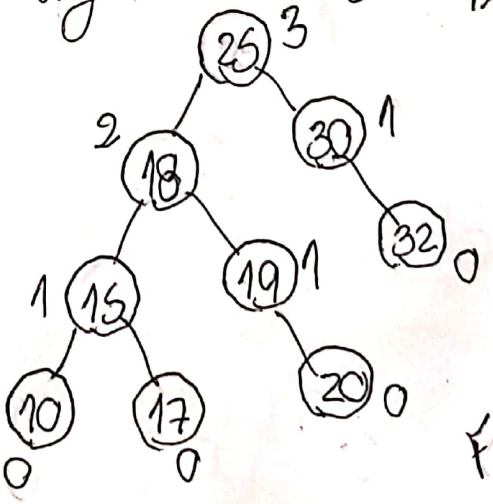
Realizando notação dupla Direita



Inserindo 20



Realizando notação simples



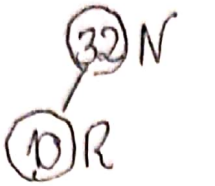
FINAL!

QUESTÃO 4: 32 / 10 / 15 / 25 / 30

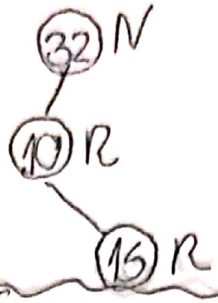
4.1

Inserir 32

Inserir 10

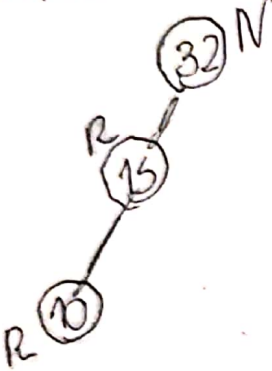


Inserir 15

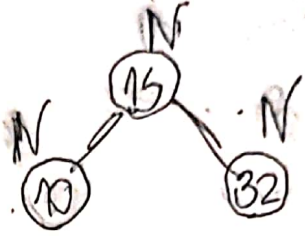


Problema encontrado 2 RUBROS!

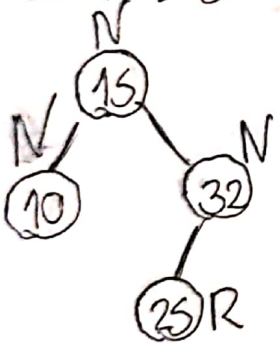
Rotaciona 10 à esquerda



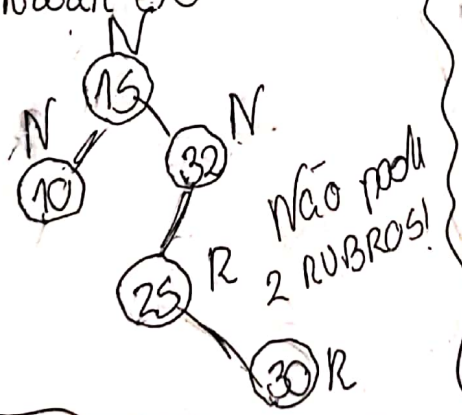
Rotaciona 32 à direita e troca cor com 15 raiz fica preta.



Inserir 25

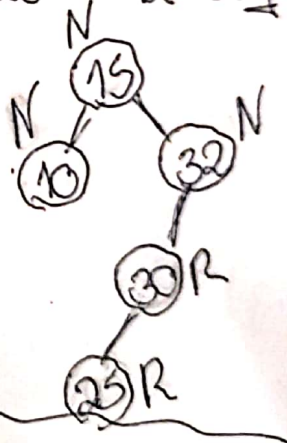


Inserir 30

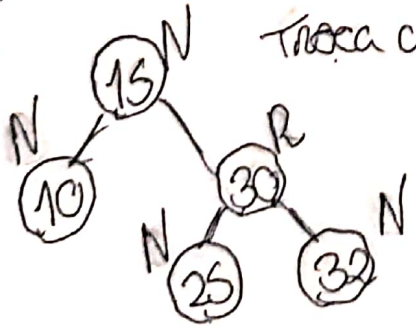


NÃO PODE 2 RUBROS!

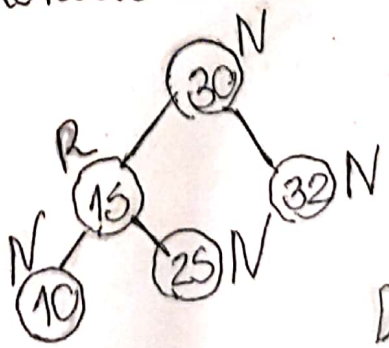
Rotaciona à esquerda



Rotaciona à direita e troca cor!



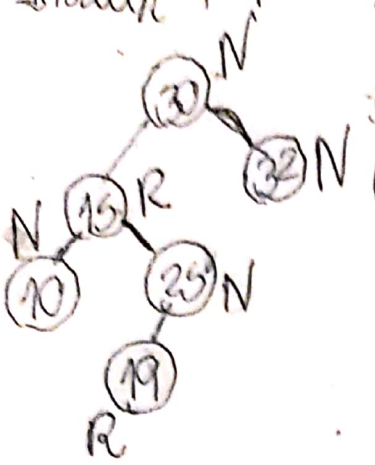
Rotaciona



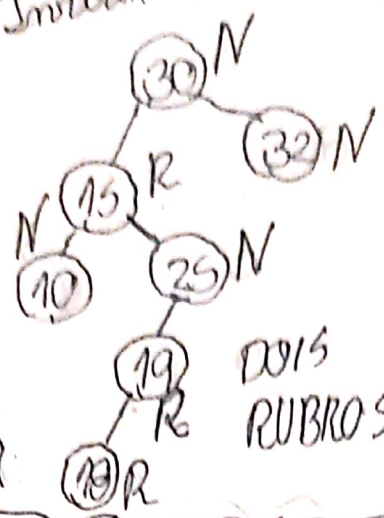
RESPOSTA FINAL

9.2

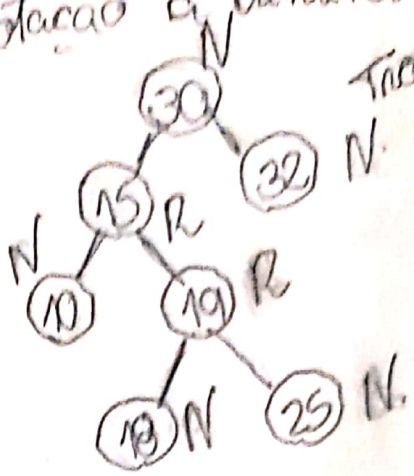
Inserir 19



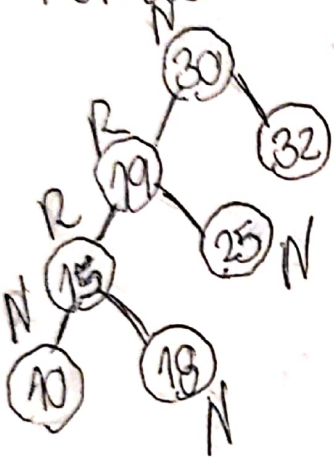
Inserir 18



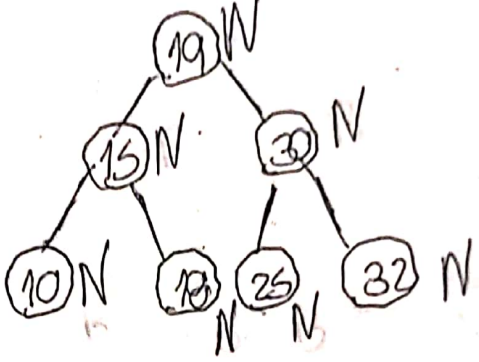
Rotação à direita e troca a cor



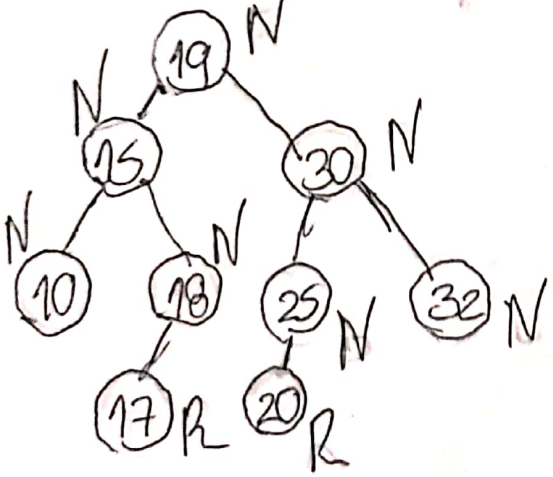
Rotaciona à esquerda e rubro!



Rotaciona para direita e troca a cor!



Inserir 17 e 20

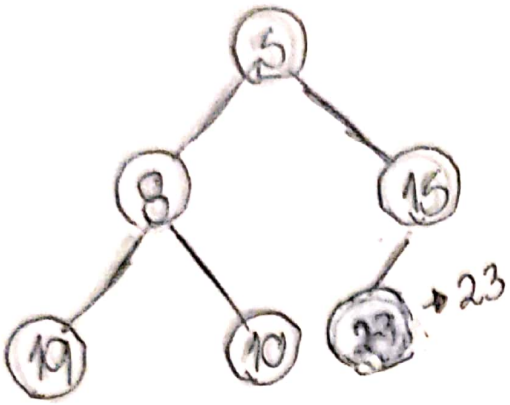


RESPOSTA FINAL.

5- QUESTÃO

5 | 8 | 15 | 19 | 10 | 23 | | | | | | |

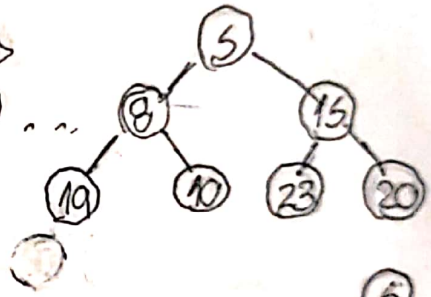
5.1



5.2 Inserir nesta Map as chaves 20, 7, 11, 18:

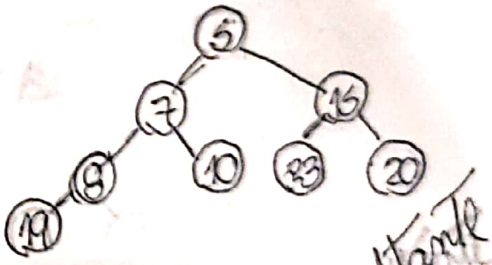
20

5 | 8 | 15 | 19 | 10 | 23 | 20 ...



7

5 | 7 | 15 | 8 | 10 | 23 | 20 | 19 ...



11

5 | 7 | 15 | 8 | 10 | 23 | 20 | 19 | 11 ...

Resultante

18

5 | 7 | 15 | 8 | 10 | 23 | 20 | 19 | 11 | 18 ...

5.3 Remover chave mínima 5.1

~~5~~ 23 | 8 | 15 | 19 | 10 |

↳ 8 | 23 | 15 | 19 | 10 | ...

↳ 8 | 10 | 15 | 19 | 23 ← Depois de remover a chave mínima.