**3ª Avaliação Parcial**
Curso: Engenharia da Computação
Disciplina: Estruturas de Dados
Prof. Jarbas Joaci de Mesquita Sá Junior
Universidade Federal do Ceará – UFC/Sobral

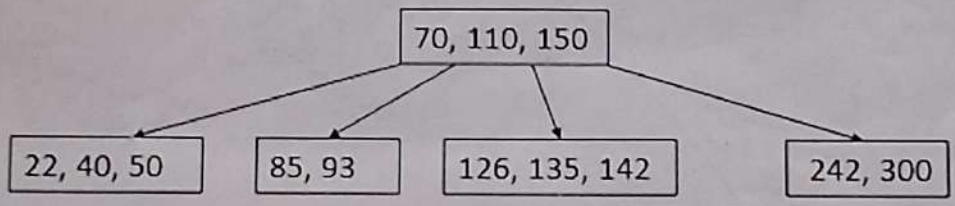
Nome: [redacted] Data 24/09/2024

1ª) Explique detalhadamente o algoritmo *Bubblesort*. (2,5 pontos)

2ª) Explique detalhadamente o algoritmo *Quicksort*. (2,5 pontos)

3ª) Construa uma árvore B de ordem $t = 3$ para a seguinte sequência de chaves: 19, 22, 23, 24, 25, 26, 2, 3, 4, 5, 11, 12, 13, 14, 15, 6, 7, 8, 9, 10, 1, 16, 17, 18, 20, 21. (2,5 pontos)

4ª) Retire da árvore B de ordem $t = 3$ abaixo a seguinte sequência de elementos: 40, 50, 70, 142, 300. Desenhe a árvore resultante de **cada** remoção. (2,5 pontos)




**Obs.**

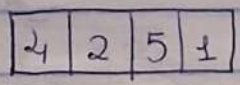
1. todos os nós podem armazenar no **máximo** $2t$-1 elementos. A raiz pode armazenar no **mínimo** 1 elemento e os outros nós no **mínimo** $t$-1 elementos.
2. se a retirada ocorrer em um nó interno, o substituto da "chave" deverá ser buscado na subárvore esquerda;
3. se determinado nó em déficit devido a uma remoção tiver dois nós irmãos e estes puderem emprestar uma chave, o nó da esquerda deverá ser escolhido para o *empréstimo*;
4. se determinado nó em déficit devido a uma remoção tiver dois nós irmãos e estes **não** puderem emprestar uma chave, o nó da esquerda deverá ser escolhido para a *concatenação*.

Nome: Caitlma Maria Melo Fainhares.

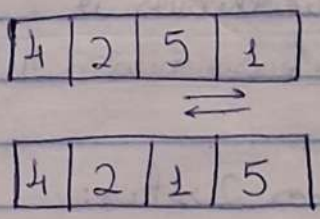
1-) Bubblesort: O algoritmo bubblesort, ou ordenação bolha, realiza multiplas passagens pela lista, identificando e trocando elementos com o objetivo de levar os elementos de maior valor para o final da lista, portanto a cada passagem leva o elemento de maior valor para o final. Sendo assim, os elementos se deslocam como uma bolha.

Complexidade: No melhor caso a complexidade é $O(n)$.
No pior caso a complexidade é $O(n^2)$.

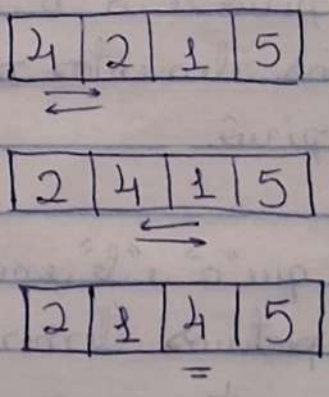
Temos aqui um exemplo de vetor desordenado →



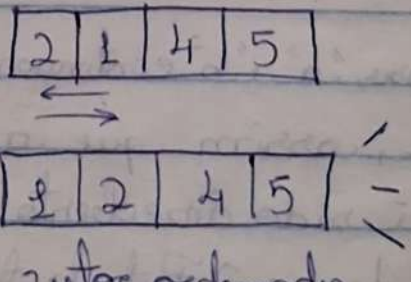

Passando pelo vetor, vemos que o 5 é o elemento de maior valor, portanto, podemos trocá-lo de lugar com o 1.




Passando pelo vetor novamente, vemos que o 4 é o segundo elemento de maior valor, portanto trocamos ele de lugar com o 2 e depois com o 1, até que o 4 chega à sua posição correta.




E por fim, trocamos o 2 com o 1. Assim, temos o vetor ordenado.



2-) Quicksort: O algoritmo quicksort começa escolhendo um número arbitrário x (geralmente o escolhido é o primeiro do vetor) para ser o piruê. O piruê estará na sua posição correta quando os números à sua esquerda forem menores ou iguais a ele e os números à sua direita forem maiores que ele. É assim recursivamente até o vetor estar ordenado.

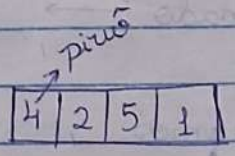
piruê

Complexidade: Melhor caso e Médio caso = $O(n \log(n))$
Pior caso = $O(n^2)$

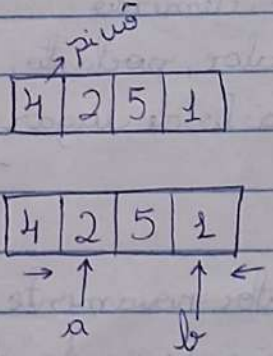
Vamos ver um exemplo: | 4 | 2 | 5 | 1 |

Aqui temos um vetor desordenado.

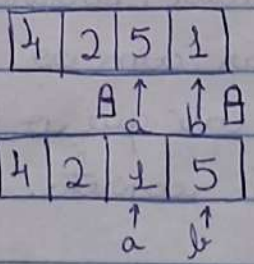
Primeiro, escolhemos o piruê, nesse caso será o 4.



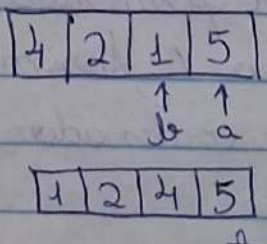

Agora, teremos 2 setas: A "a" e a "b". A "a" irá procurar elementos maiores que o piruê e a "b" irá procurar elementos menores que o piruê.




Assim, que "a" e "b" encontrarem seus respectivos elementos eles param e trocam os 2 de lugar.

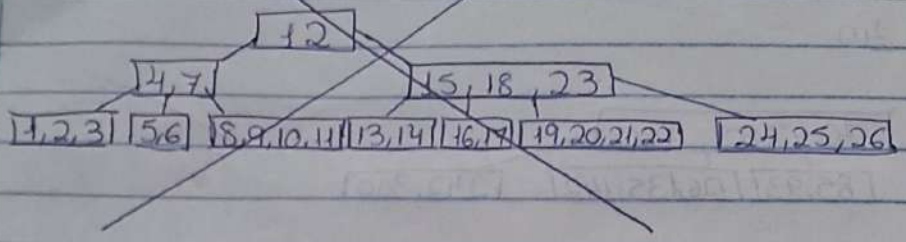



Após isso, "a" e "b" continuam percorrendo o vetor, assim que o "a" for maior que "b", terá sido descoberto a posição correta do piruê. Portanto, troca o piruê de posição.

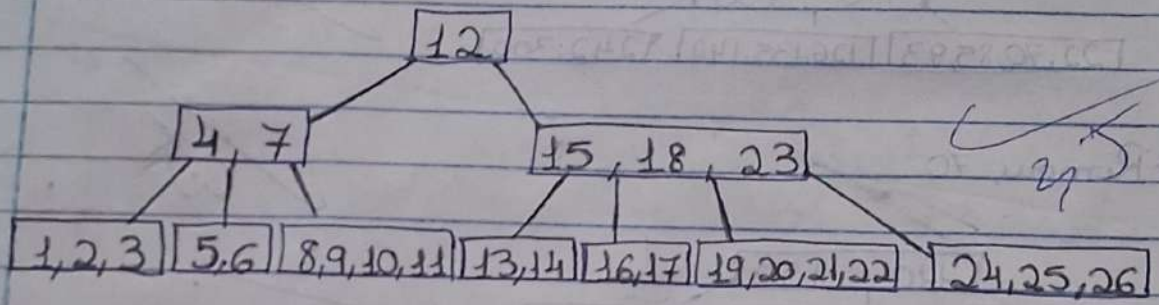



→ vetor ordenado

3-) $t = 3$ máximo = 5 mínimo = 2



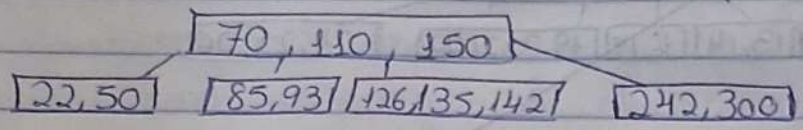

3-) $t = 3$ máximo = 5 mínimo = 2



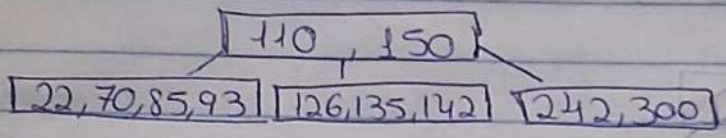

* Desenvolvimento nos rascunhos *

4-) t=3 máx: 5 mín: 2

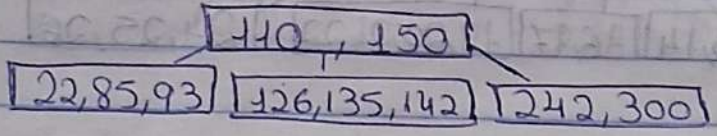
* Remove 40



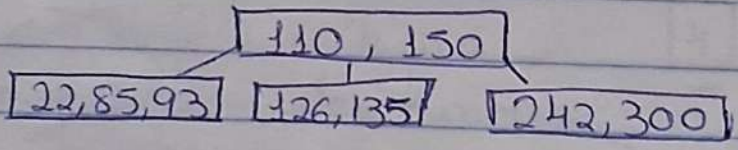

* Remove 50



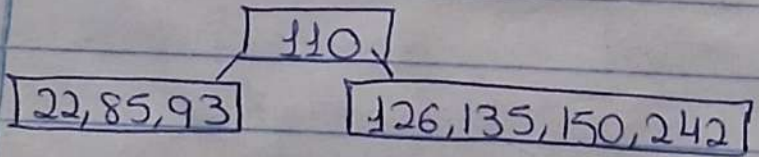

* Remove 70



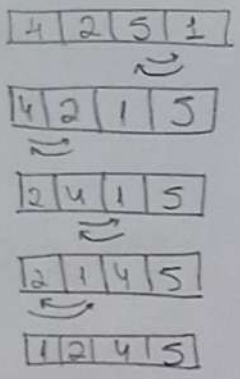

* Remove 142




* Remove 300



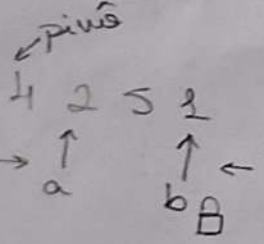
1- O algoritmo bubblesort realiza múltiplas passagens pela lista, identificando e trocando elementos, levando os maiores elementos para o final da lista. Portanto, os elementos se deslocam como uma bolha. Complexidade: Melhor → O(n); Pior O(n²)



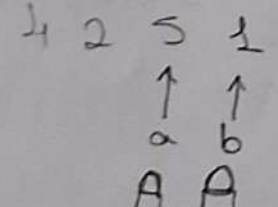
a cada passagem leva o elemento maior para o final.

i-ésimo maior valor.

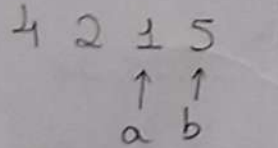
2- O algoritmo quicksort escolhe um número arbitrário x, que será o pivô, esse número terá que ter elementos menores ou iguais a ele à sua esquerda e elementos maiores que ele à sua direita, quando ele estiver na posição correta. E assim recursivamente, até o vetor estar ordenado.

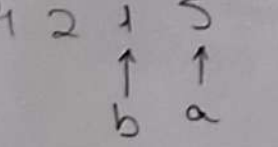



a → procura elementos maiores que o pivô
b → procura elementos menores que o pivô




Complexidade → Melhor e médio caso: O(n log n)
Pior: O(n²)



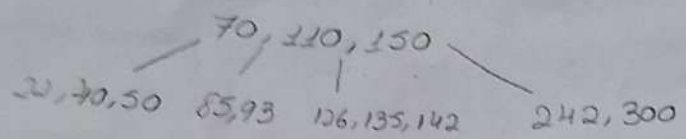




b está antes de a, portanto a posição de b é a posição certa para o pivô, então troca os dois.

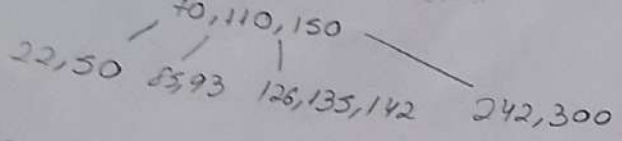
| 1 | 2 | 4 | 5 |
|---|---|---|---|

Nome: Guilhina

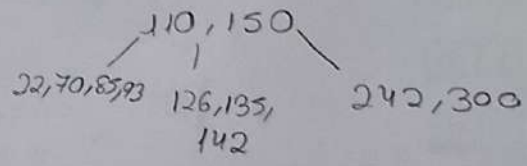
2-) $t=3$ máx: 5 mínimo 2 (mín: 2)
2·3-1=5 3-1=2



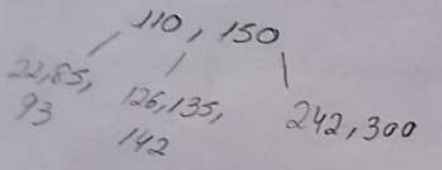

* Remove 40 → 1º



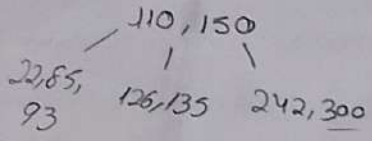

* Remove 50 → 2º



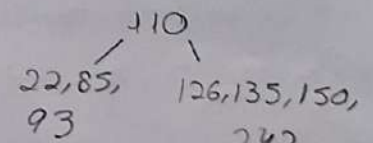

* Remove 70 → 3º



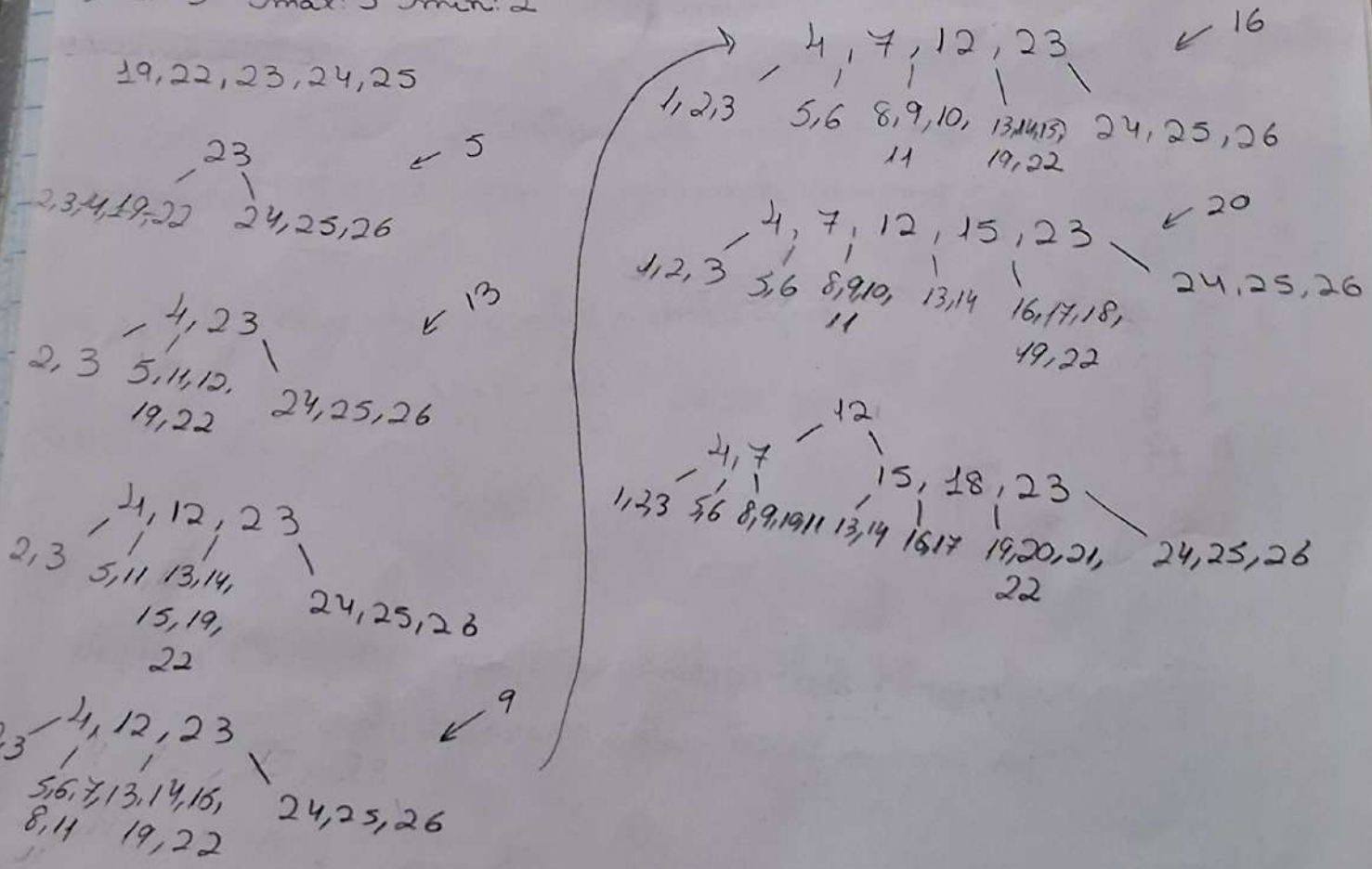

* Remove 142 → 4º



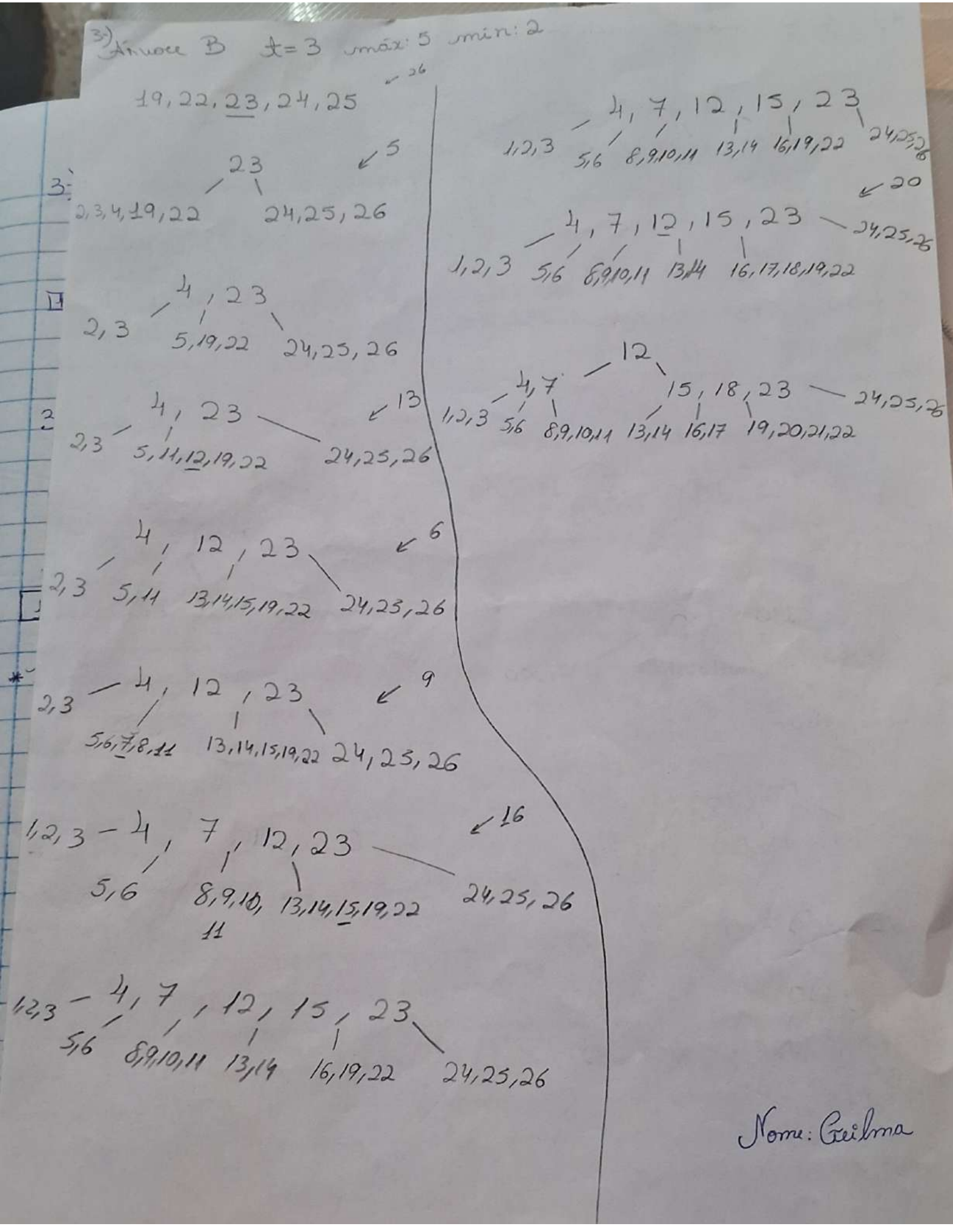

* Remove 300 → 5º




3-) $t=3$ máx: 5 mín: 2

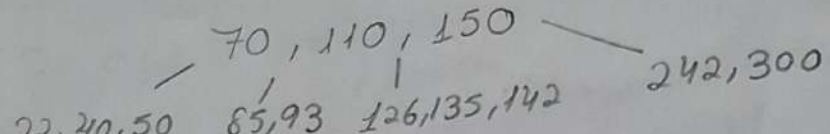


3) Árvore B $t = 3$ máx: 5 mín: 2

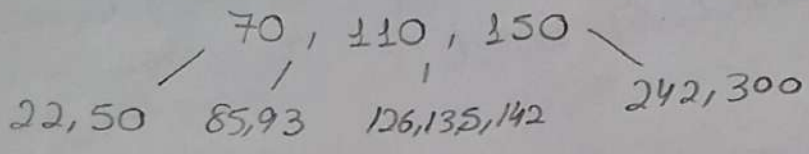



Nome: Geilma

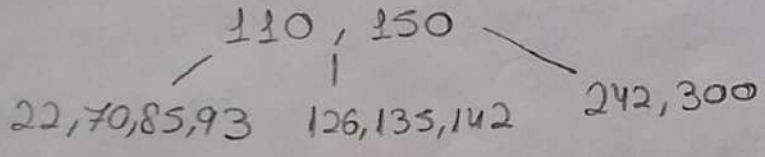
4-) $t = 3$ máx: 5 mín: 2



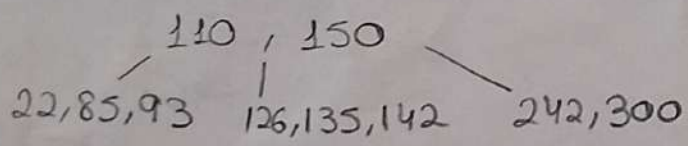

-Remove 40-



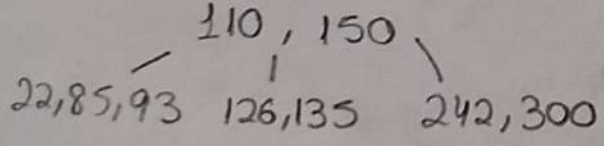

-Remove 50-




-Remove 70-




-Remove 142-




-Remove 300-