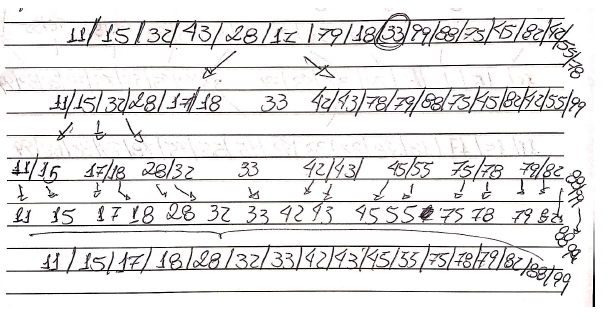
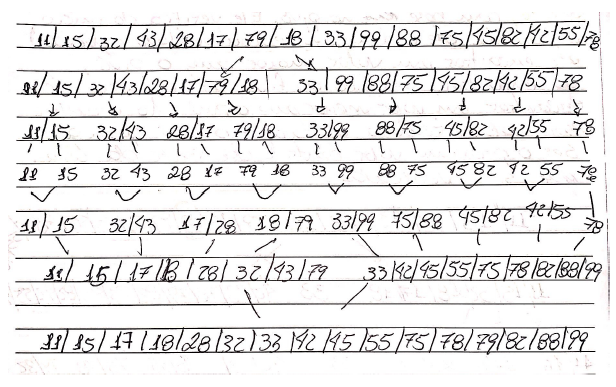
Exercício – 1

O método “quickSort” utiliza a recursividade realizando a chamada dele mesmo e vai reorganizando a lista à esquerda e à direita do pivô, já o método “partição” realiza o particionamento, com base em no pivô. Ele verifica do início dessa lista até o fim até encontrar um valor maior que o pivô. Ao encontrar, verifica do fim até o início até encontrar um valor menor que o pivô. Ao encontrar estes valores realiza a troca, e o pivô passa a ser o próximo valor central da parte da lista que será verificada.



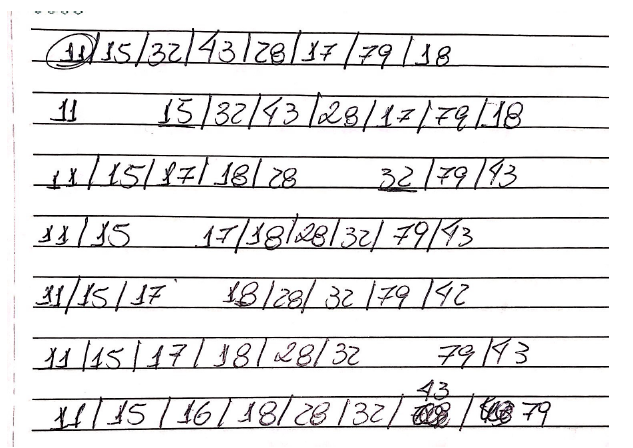
Exercício – 2

O método mergeSortRecursivo trabalha com recursividade realizando chamadas dele mesmo, e utiliza o método “mesclar” porque é um algoritmo que trabalha sua ordenação intercalando os elementos. Primeiro divide o vetor em partes menores até chegar no caso base. Quando finaliza essa operação, desempilha os valores e vai intercalando as partes até ter novamente uma lista de elementos, porém totalmente ordenada pois realiza esse procedimento ao retornar e desempilhar os elementos.

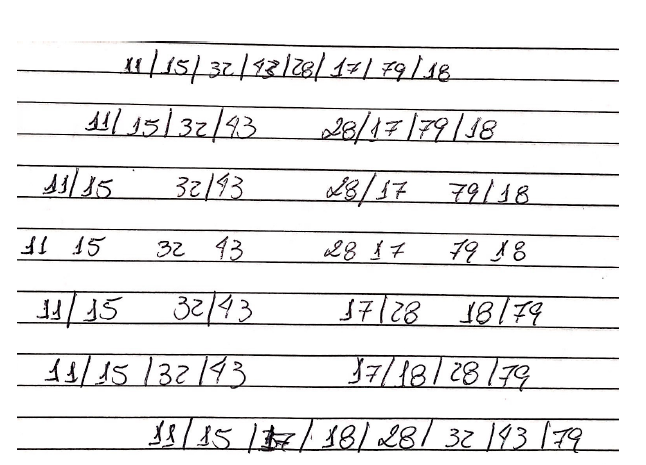


Exercício – 3

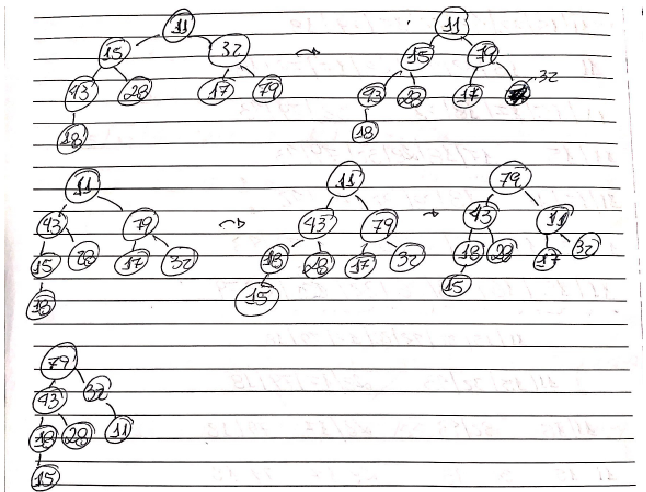
O QuickSort particiona uma lista de elementos e define um pivô, utilizando o método de dividir e conquistar, onde os elementos a esquerda do pivô serão menores do que ele e os elementos a direita serão maiores do que ele. Este pivô pode ser definido aleatoriamente, no início, no meio ou no fim da lista. O algoritmo verifica da esquerda para direita valores maiores ou iguais, e da direita para esquerda, valores menores ou iguais, ao encontrar realiza a troca dos elementos.



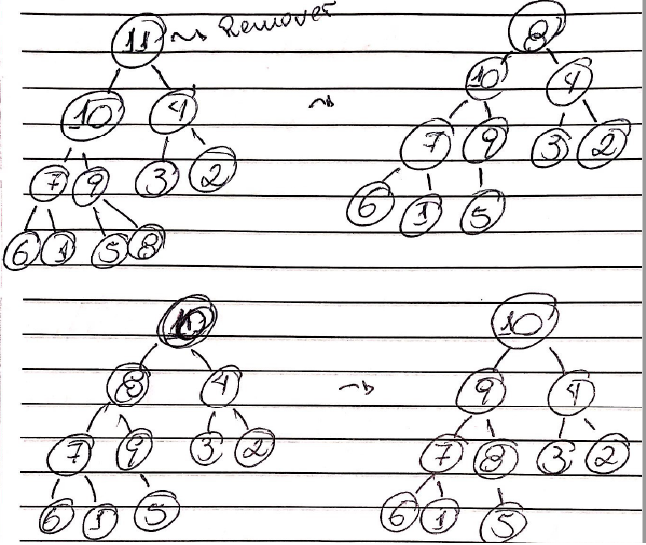
O MergeSort também utiliza o método de dividir e conquistar, porém, diferente do QuickSort não define pivô, trabalha com método de intercalação dos elementos. Na fase da divisão, separa os elementos em parte menores até o seu caso trivial (apenas um elemento). Feito isso, faz a parte da conquista, onde recursivamente vai intercalando os elementos e realizando a ordenação.



HeapSort enxerga o vetor como uma árvore binária e é possível dois tipo de estrutura, através do heap-máximo ou heap-mínimo. Caso seja construído através do heap-máximo, o elemento de maior valor ficará no “nó” raiz, que é o nó que fica no topo do algoritmo. Para cada nó, todos os elementos descendentes são menores do que o nó raiz.



Exercício 4



Exercício 5

O algoritmo em questão é o QuickSort, que utiliza o método de dividir e conquistar e temos que definir um “pivô”, este será um determinado ponto da lista que é definido para que as comparações sejam realizadas. Por exemplo, se definimos o primeiro elemento da lista como sendo o pivô, o algoritmo percorre a lista do seu inicio para o fim, até encontrar um elemento maior que o pivô, e do fim para o início até encontrar um elemento menor que o pivô. Quando essas duas condições são atendidas, esses números são trocados de lugar.