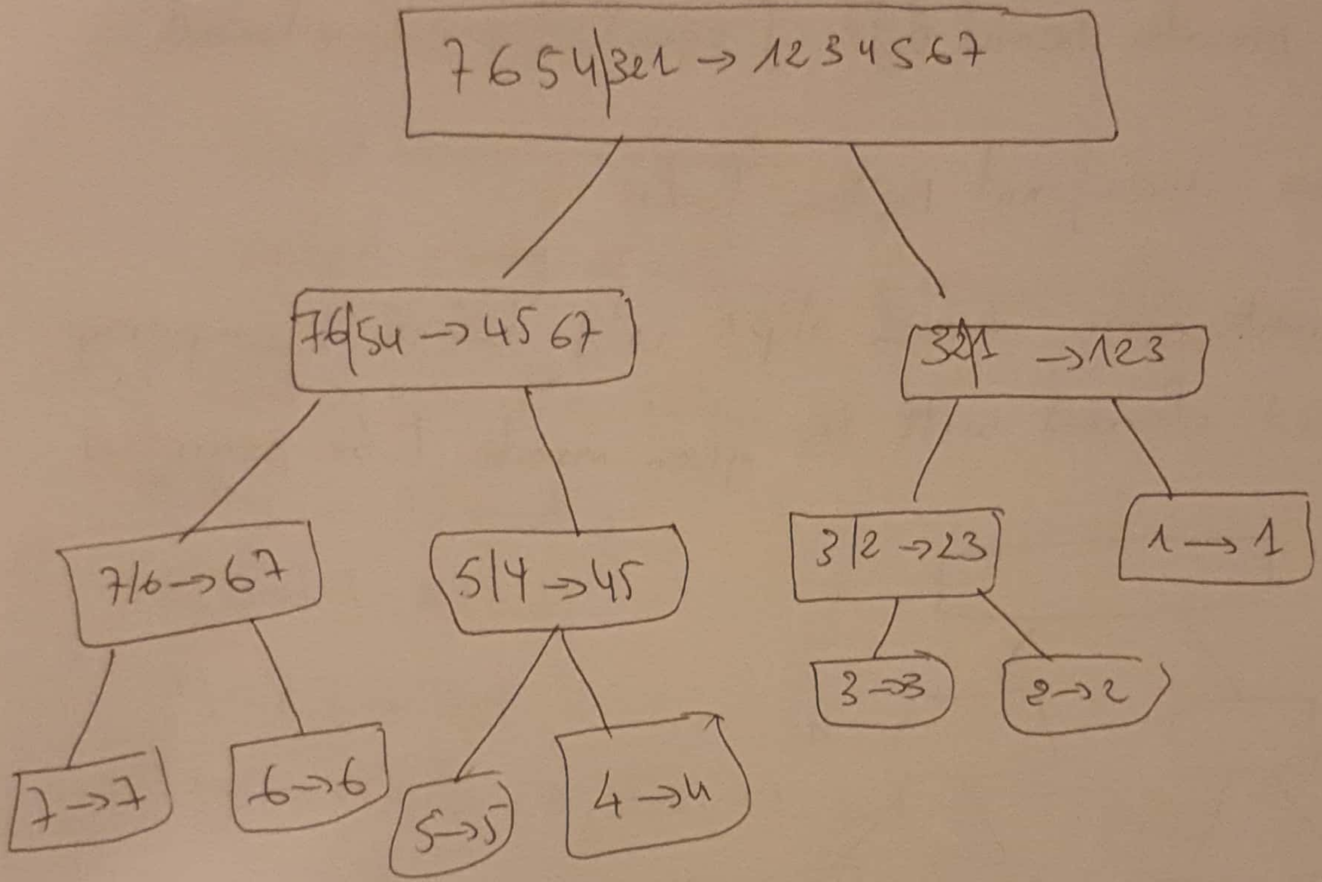
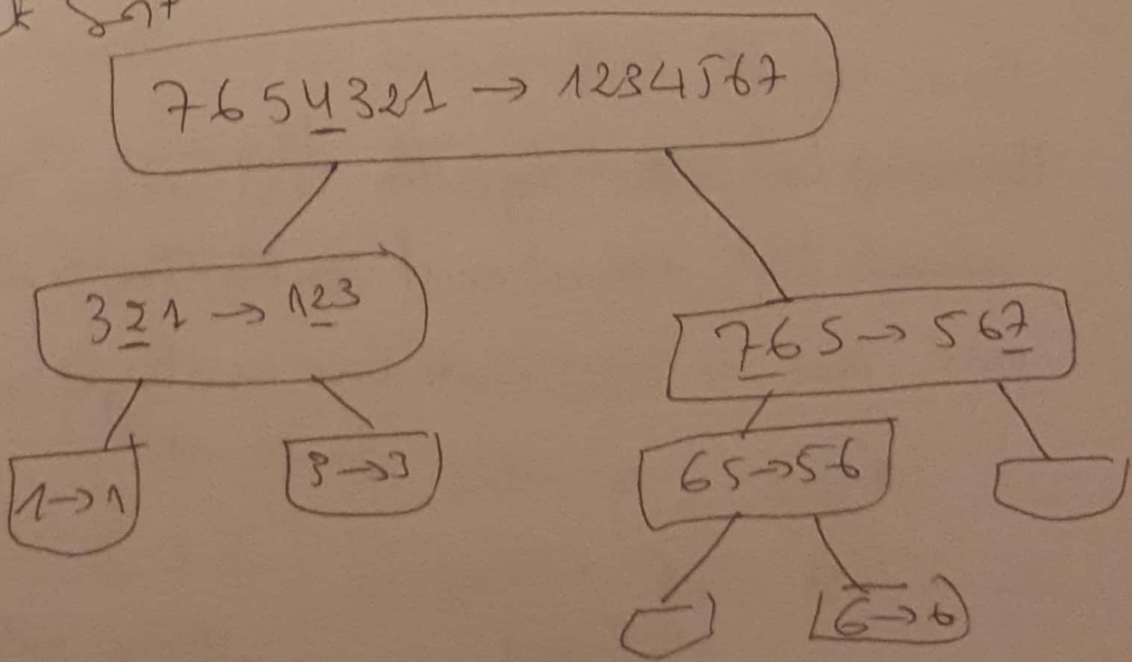
Lab 3B

1) using Merge Sort

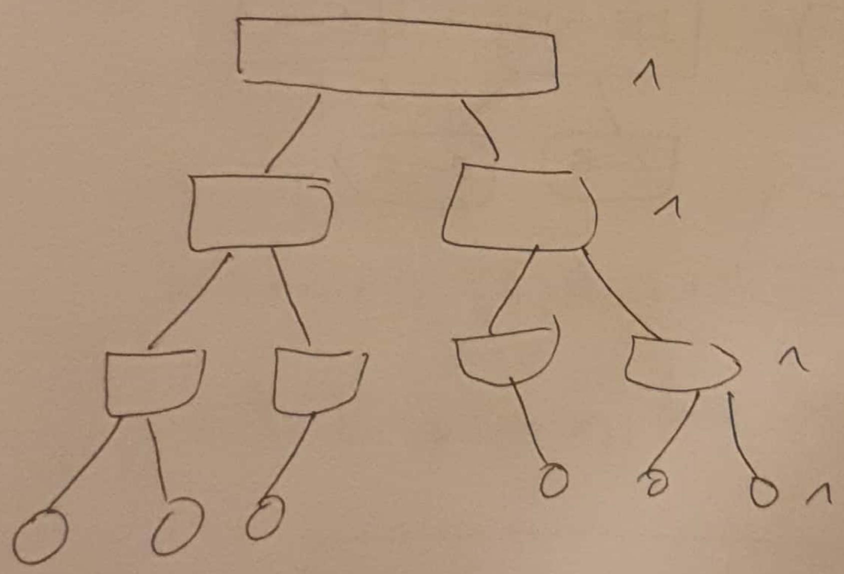


2) using Quick Sort



pb2: binary search algorithm only search at one side of the list
element is compared with the middle element if large search
right if smaller search left if equal to mid element
return true and if not return false.

note: work done at each step i is 1 which is comparing
the mid element with the given number to be searched.



the worst case in this algorithm is if we are looking for
element that does not exist on the tree, at each level
one there is O(1) work done and height of the tree is
(logn)

$T(n) = O(1 * \log n)$
$= O(\log n).$

Scanned with CamScanner

Problem 3: recursive algorithm to reverse order

Algorithm reverse order

Input array of n elements

output : array of n elements in reversed order

if start >= end then return +c

temp ← A[start]

A[start] = A[end]

A[end] ← temp

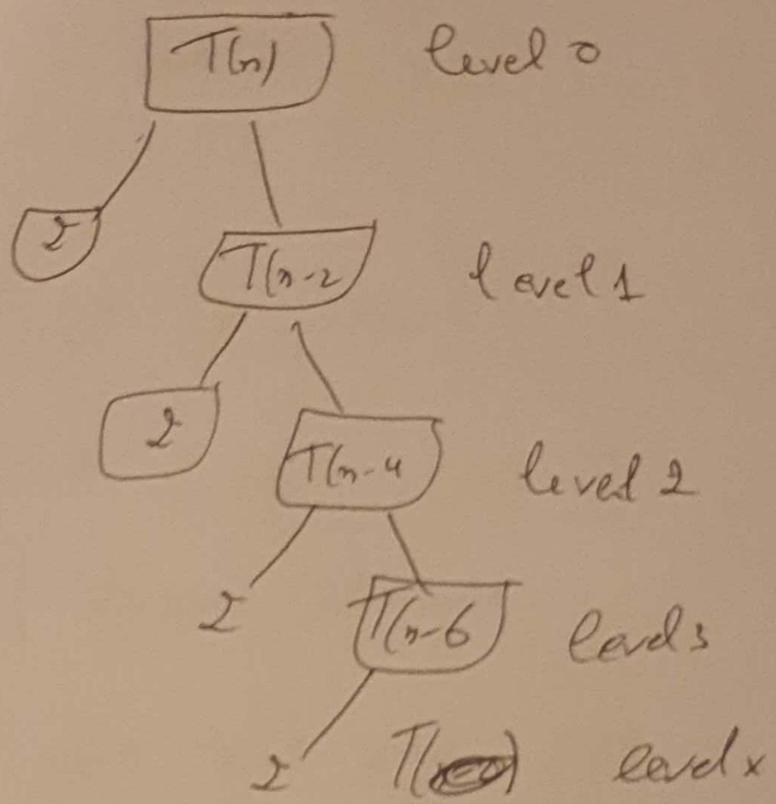
reverseOrder (A, start + 1, end - 1) $T(n-2)$

Compute the running time using Count Self Calls

in the given algorithm we can observe that the next method recursive call is smaller by 2, than the first one because start + 1 and end - 1

$$T(n)\begin{cases} 2 & n=0 \\ T(n-2)+c & \text{otherwise} \end{cases}$$

because we don't have formula to resolve this we will use the tree method



O(n).

problem4 · algorithm fib it (n)

Input : non negative intege (n)

output the fibonacci value at n^{th} sequence

```
fib ← 1
fib ← 1
for i ← 2 to n-1 do
    temp ← fib
    pfib ← fib + pfib
    pfib ← Temp
    return fib
```

$T(n) = 4(n) = O(n)$