Lab4

# Problem 1

1. BubbleSort is stable sorting algorithms because this block of code

if(arr[j]> arr[j+1]){  
 swap(j,j+1);  
}

show us that BubbleSort only swap current element with next element if current element is greater than next element.

1. SelectionSort is not stable because this block of code

int nextMinPos = minpos(i,len-1);  
swap(i,nextMinPos);

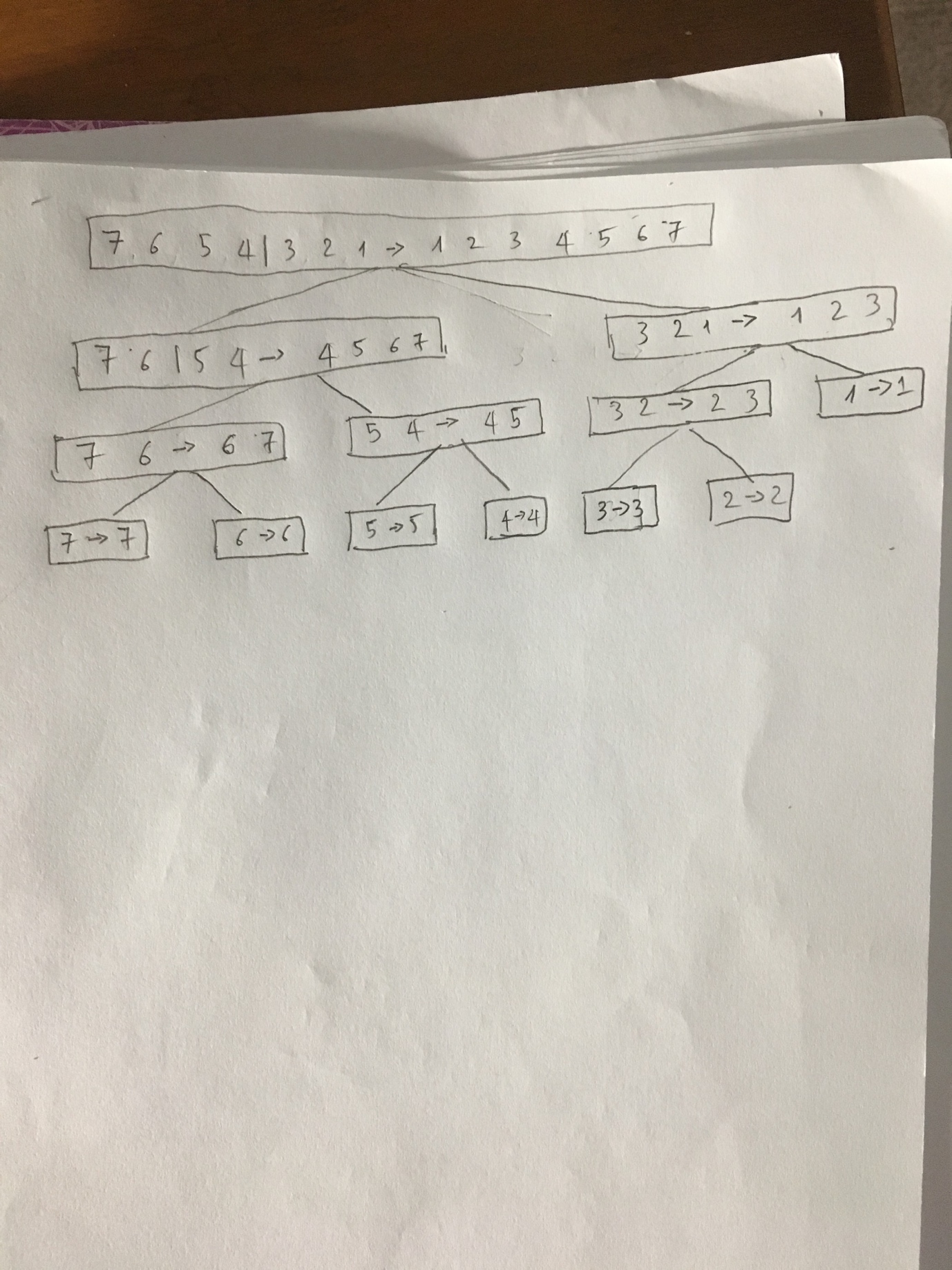
show us that SelectionSort will swap current element with the next minimum element even next minimum element is equal to current element.

1. InsertionSort is stable sorting algorithms because this block of code

while(j>0 && temp < anArray[j-1]){  
 anArray[j] = anArray[j-1];  
 j--;  
}

show us that only if the *temp* will be stuck if it meets an element less than it and won’t do anything.

# Problem 2



# Problem 3

A.

Algorithm mergeSort(S)  
 Input sequence S with n integers  
 Output sequence S sorted  
 if S.size() > 1 then  
 if S.size() <= 20 then  
 insertionSort(S)  
 else  
 (S1, S2) <- partition(S, n/2)  
 mergeSort(S1)  
 mergeSort(S2)  
 S <- merge(S1, S2)  
 return S

B. See MergeSortPlus.java

C.

The test result:

66 ms -> MergeSort

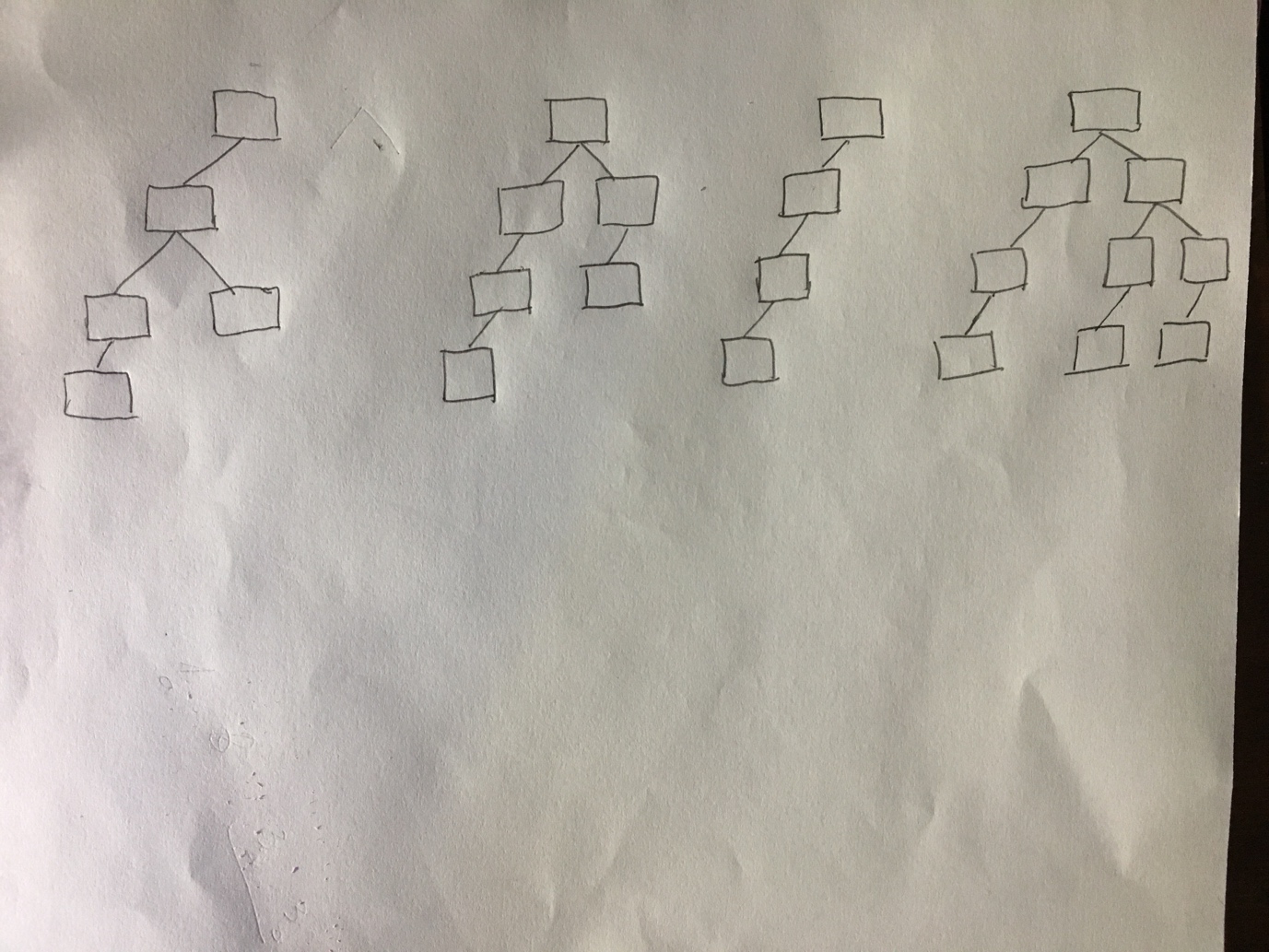
104 ms -> MergeSortPlus

MergeSort run faster than MergeSortPlus.

I test these sort algorithms by copy the MergeSortPlus to sortroutines package and run SortTester.java

# Problem 4

A



B. Every binary tree of height 3 has at most 23=8 leaves => True

C. Number of leaves = 2n , n = height of tree