# Prob1:

**Bubble sort: stable**

if(arr[j]> arr[j+1]){

    swap(j,j+1);

}

⬄ with i < j only swap when a[i]>a[j]

**Insertion Sort: stable**

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

    anArray[j] = anArray[j-1];

    j--;

}

Only shift to the right when item need to be inserted < item in sorted array

**Selection Sort: stable**

if(arr[i]<m){

    m = arr[i];

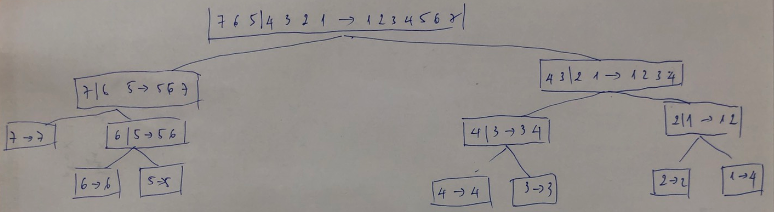
    index = i;

}

The first min item will be swap first and never be swapped again

# Prob2:

a.



# Prob3:

**a. pseudo-code**

Algorithm **mergeSort**(S)

Input sequence S with n integers

Output sequence S sorted

If S.size() <=20 then

return insertionSort(S)

(S1 , S2 ) ← partition(S, n/2)

S1 ← mergeSort(S1 )

S2 ← mergeSort(S2 )

S ← merge(S1 , S2 )

return S

**b. Implement in Java**

void mergeSort(int[] tempStorage, int lower, int upper) {

if (upper - lower <= 20) {

insertionSort(theArray, lower, upper);

return;

} else {

int mid = (lower + upper) / 2;

mergeSort(tempStorage, lower, mid); // sort left half

mergeSort(tempStorage, mid + 1, upper); // sort right half

merge(tempStorage, lower, mid + 1, upper); // merge them

}

}

public void insertionSort(int[] a, int from, int to) {

if (a == null || to - from < 1) {

return;

}

int temp = 0;

int j = 0;

for (int i = 1 + from; i <= to; ++i) {

temp = a[i];

j = i;

while (j > 0 && temp < a[j - 1]) {

a[j] = a[j - 1];

j--;

}

a[j] = temp;

}

}

**c. Result**

Use sortTester in

ARRAY\_SIZES = { 50000, 60000, 70000, 200000 };

1st

415 ms -> MergeSortPlus

447 ms -> MergeSort

2nd

449 ms -> MergeSort

457 ms -> MergeSortPlus

3rd

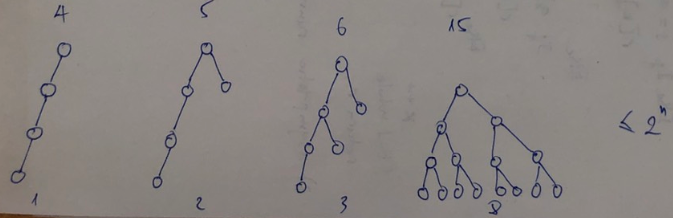
424 ms -> MergeSortPlus

502 ms -> MergeSort

Almost the same

# Prob4:

a.



b. TRUE

c. <=2n

# Prob5:

static void reverse(int[] a, int to) {

if (to == 1) {

int temp = a[0];

a[0] = a[1];

a[1] = temp;

return;

}

reverse(a, to - 1);

// swap the last with the rest

int t = a[to];

for (int i = to; i > 0; i--) {

a[i] = a[i - 1];

}

a[0] = t;

}

Number of self-call: n

Number of work for each call: n

Running time: O(n2)