1) Insertion Sorting

* Array = [1, 3, 5, 2, 0]

j	x	array	locked ele.
~~1~~	~~1~~	(i) 1, 3, 5, 2, 0	~~3~~
~~2~~	~~2~~	(i) 1, 3, 5, 2, 0	~~5~~
~~3~~	~~3~~	(i) 1, 3, 2, 5, 0	~~2~~
	~~2~~	(ii) 1, 2, 3, 5, 0	~~2~~
	~~1~~		~~1~~
4	~~4~~	(i) 1, 2, 3, 0, 5	~~0~~
	~~3~~	(ii) 1, 2, 0, 3, 5	~~0~~
	~~2~~	(iii) 1, 0, 2, 3, 5	~~0~~
	~~1~~	(iv) 0, 1, 2, 3, 5	~~0~~
	0		0

5
X

[0, 1, 2, 3, 5]

In case of worst case ~~scenario~~ scenario, while loop will be executed till x becomes 0 as in case of [5, 4, 3, 2, 1]

=

5, 4, 3, 2, 1 n = 5

j	x	key	array
1	1	4	(i) 4, 5, 3, 2, 1
0		4	
2	2	3	4, 5, 3, 2, 1
			(i) 4, 3, 5, 2, 1
	1	3	(ii) 3, 4, 5, 2, 1
0		2	
3	3	2	(i) 3, 4, 2, 5, 1
2	2	2	(ii) 3, 2, 4, 5, 1
1	1	2	(iii) 2, 3, 4, 5, 1
0			
4) 4	4	1	(i) 2, 3, 4, 1, 5
3	3	1	(ii) 2, 3, 1, 4, 5
2	2	1	(iii) 2, 1, 3, 4, 5
1	1	1	(iv) 1, 2, 3, 4, 5
0			

$1 + 2 + 3 + 4$

$\approx \frac{n(n+1)}{2} = O(n^2)$

2)) Quick Sort

This algorithm assumes a pivot element and finds its location ~~the~~ in the array such that every element to its left are less than that pivot element and all the elements to the right are bigger than the pivot element assuming that we want to sort the ~~element~~ array in increasing order,

Array = [1, 2, 4, 3, 8, 10, 10, 7]

i) sorting (array, 0, 7)

pivt = 7

i = -1 j = 0 Array

~~i~~ ~~0~~ (1 < 7) => 1, 2, 4 7

~~0~~

~~1~~ ~~1~~ (2 < 7) => 1, 2, 4 7

~~2~~ ~~2~~ (4 < 7) => 1, 2, 4 7

3 ~~3~~ (3 < 7) => 1, 2, 4, 3 7

~~4~~ (8 < 7) X

~~5~~ (10 < 7) X

~~6~~ (10 < 7) X

7

// swap(a[i+1] & a[end])
Array = 1, 2, 4, 3, 7, 11, 10, 8

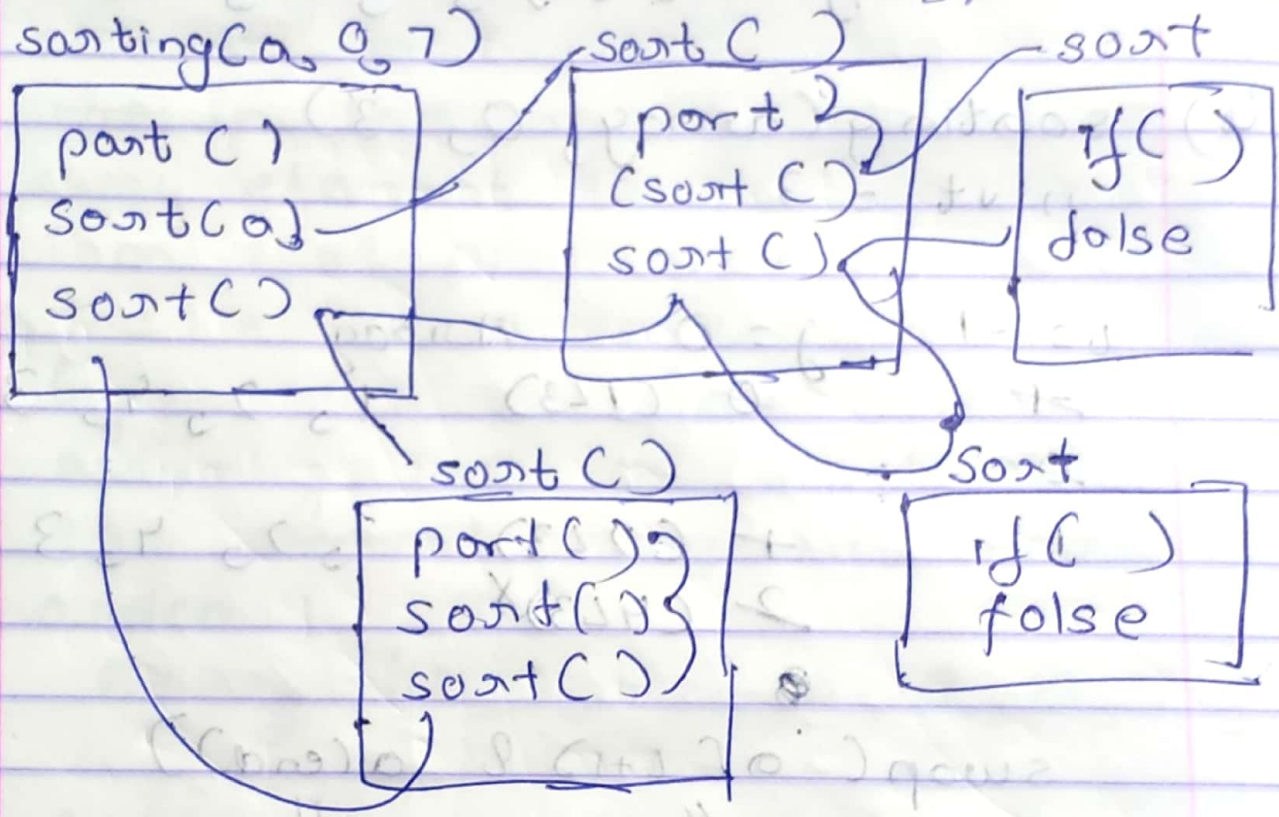
(ii) sorting(array, 0, 3)
pivt = 3

i = -1 j = 0 Array
~~+~~ ~~0~~ (1<3) 1, 2, 4, 3
~~0~~
1 + (2<3) 1, 2, 4, 3
2 (4<3)X

swap(a[i+1] & a[end])
// //
a(2) a[3]
Array = 1, 2, 3, 4, 7, 10, 10, 8

(iii) sorting (array, 5, 7)
pivt = 8
~~i = 5-1=4~~
i = 5-1=4 j = ~~5~~ Array
(11<8)X 11, 10, 8
6 (10<8)X
swap(a[i+1] & a[end])
// //
11 8

Array = 1, 2, 3, 4, 7, 6, 8, 10, 11



Total 7 activation records

It is an in place sorting algorithm since no new arrays are required

Bubble sort

Array = 5, 4, 3, 6, 1

n = 5

i = 0	j = 0	Array
	(5 > 4)	4, 5, 3, 6, 1
	(5 > 3)	4, 3, 5, 6, 1
	(5 > 6) X	
	(6 > 1)	4, 3, 5, 1, 6
1	(4 > 3)	3, 4, 5, 1, 6
	(4 > 5) X	
	(5 > 1)	3, 4, 1, 5, 6
2	(3 > 4) X	
	(4 > 1)	3, 1, 4, 5, 6
3	(3 > 1)	1, 3, 4, 5, 6

Time comp. = 4 + 3 + 2 + 1

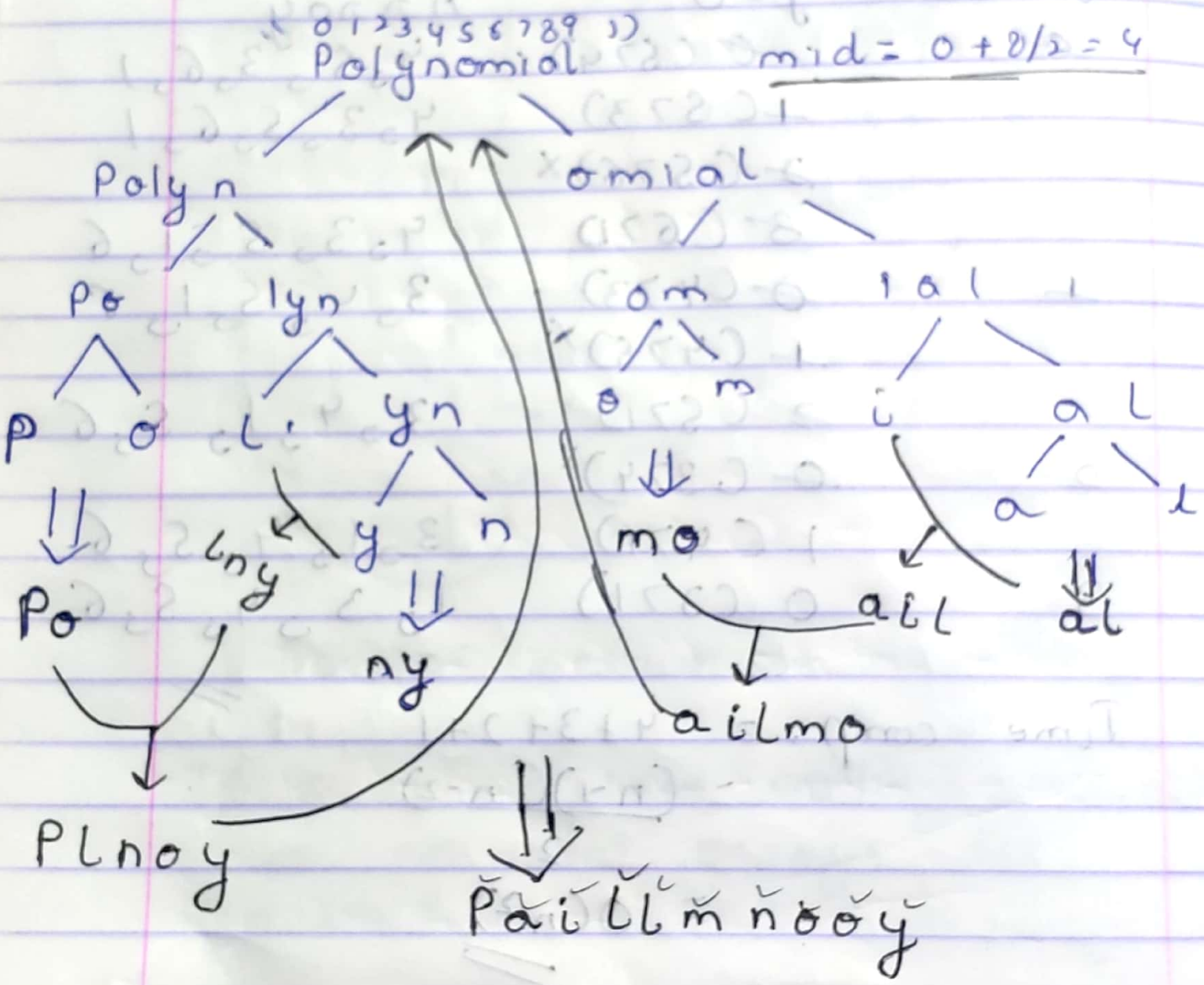
$= \frac{(n-1)(n-2)}{2}$

$= O(n^2)$

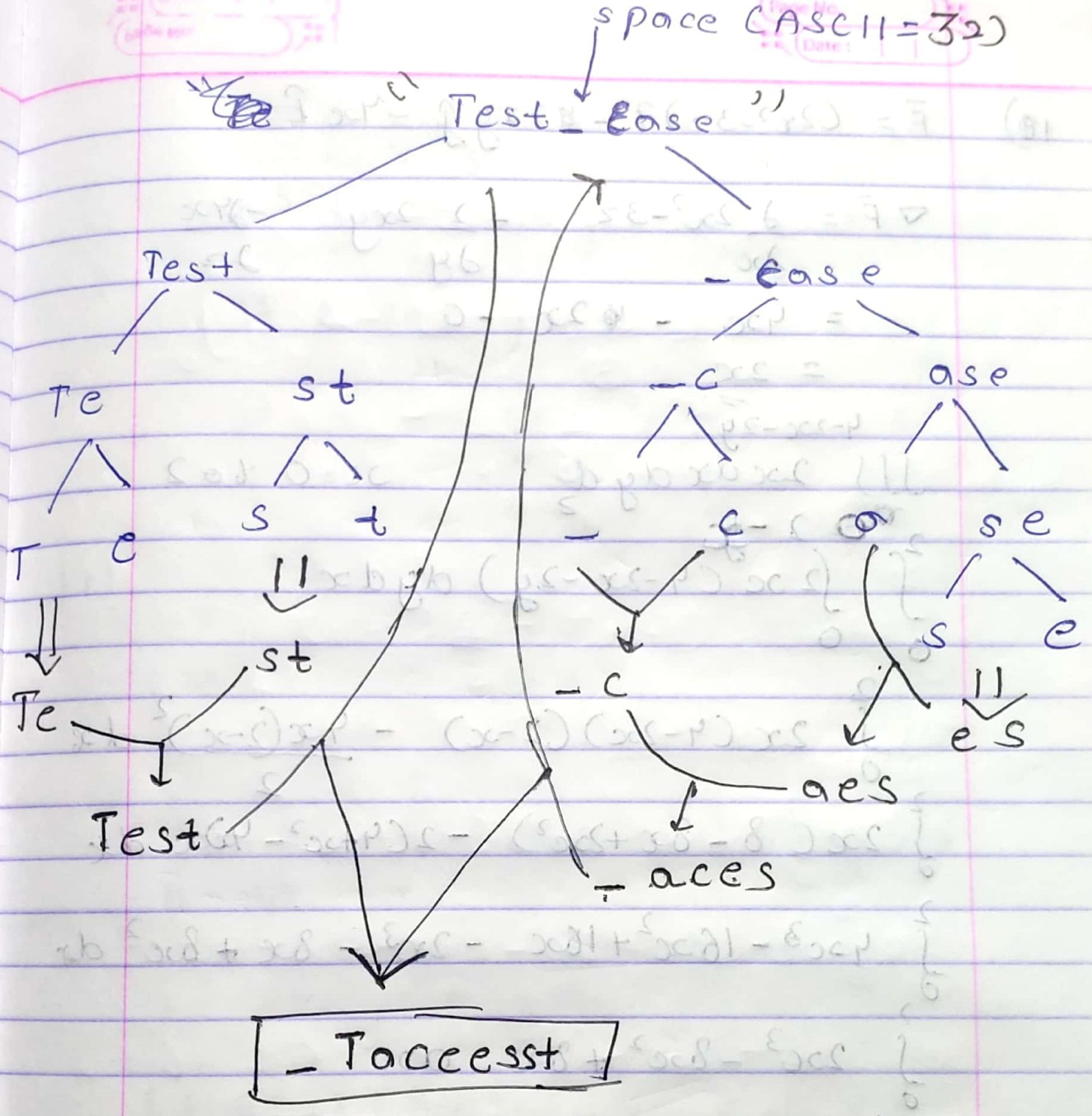
Time complexity of both bubble sort & insertion sort is $O(n^2)$ where as that of quick sort is $n \log n$

Quick sort < Bubble ≈ insertion.

3) Merge Sort



Blue text ⇒ mergeSort ();
Black text ⇒ merge Array ();



Bloct text ⇒ merge Array ()
Blue text ⇒ merge Sort ()