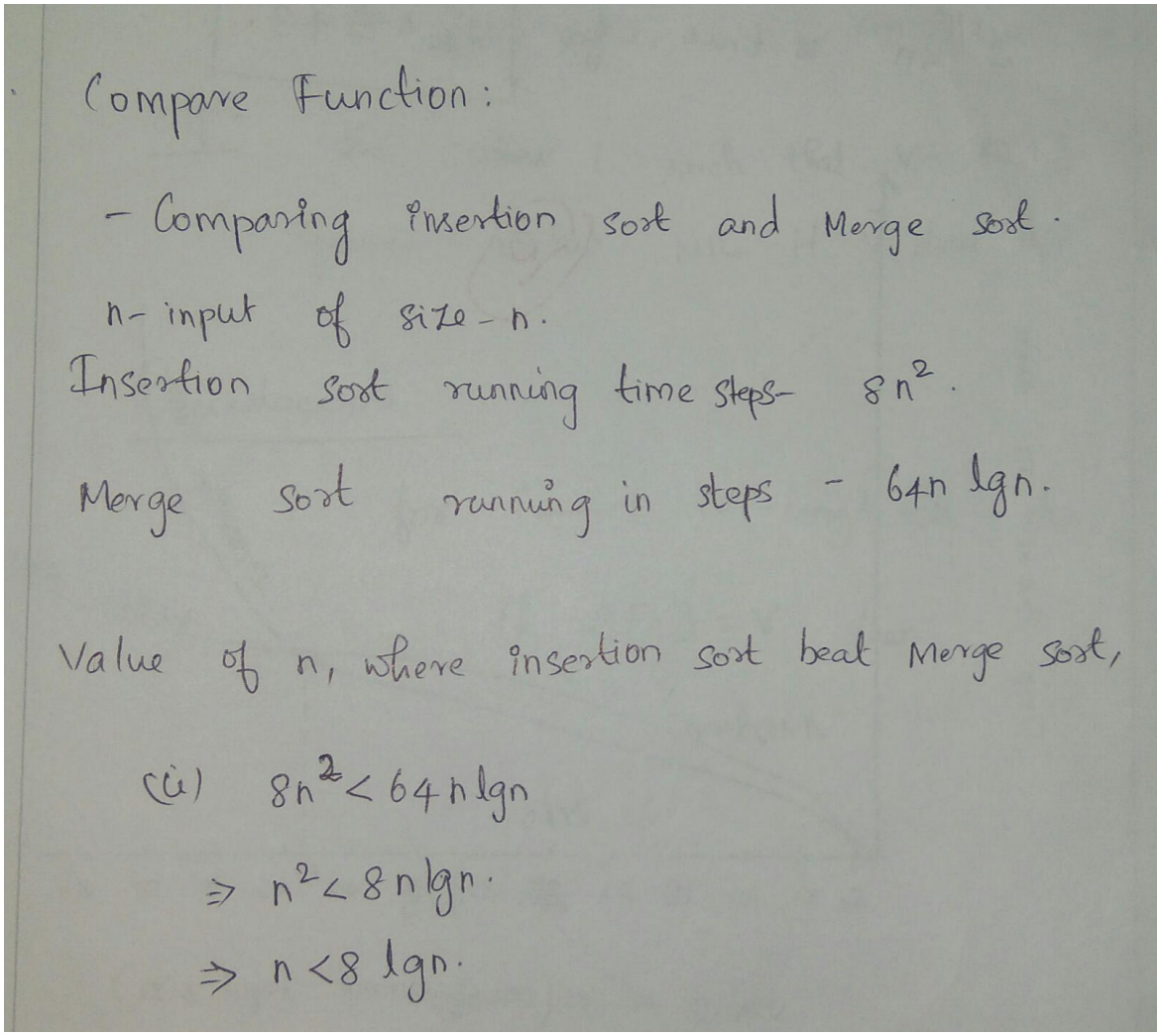
Q1.

Credit : Navaneeth Chandrasekaran



Compare Function:

- Comparing insertion sort and Merge sort.

n- input of size - n.

Insertion sort running time steps- $8n^2$.

Merge sort running in steps - $64n \lg n$.

Value of n, where insertion sort beat Merge sort,

(i) $8n^2 < 64n \lg n$

$\Rightarrow n^2 < 8n \lg n$.

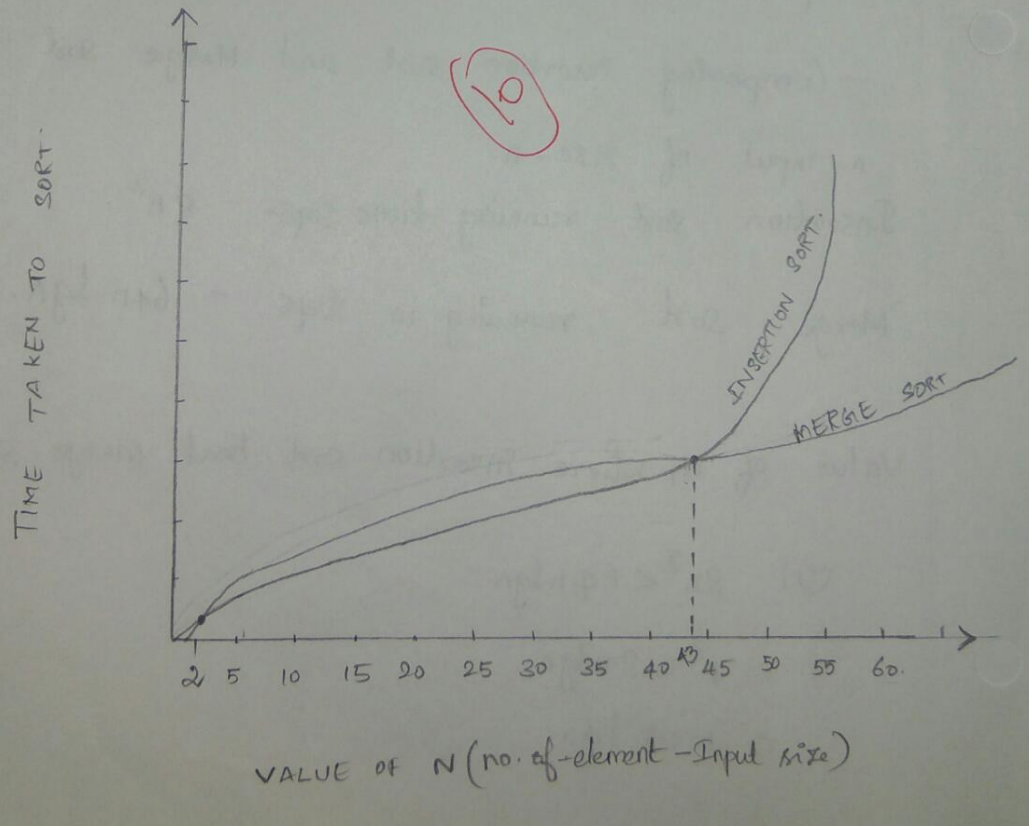
$\Rightarrow n < 8 \lg n$.

$2^{n/8} < n$ $\quad [\because \lg_{10} x = y \Rightarrow 10^y = x.]$

After doing trial and error method in the above function by substituting value of n from 1→44, it is finally proved that.

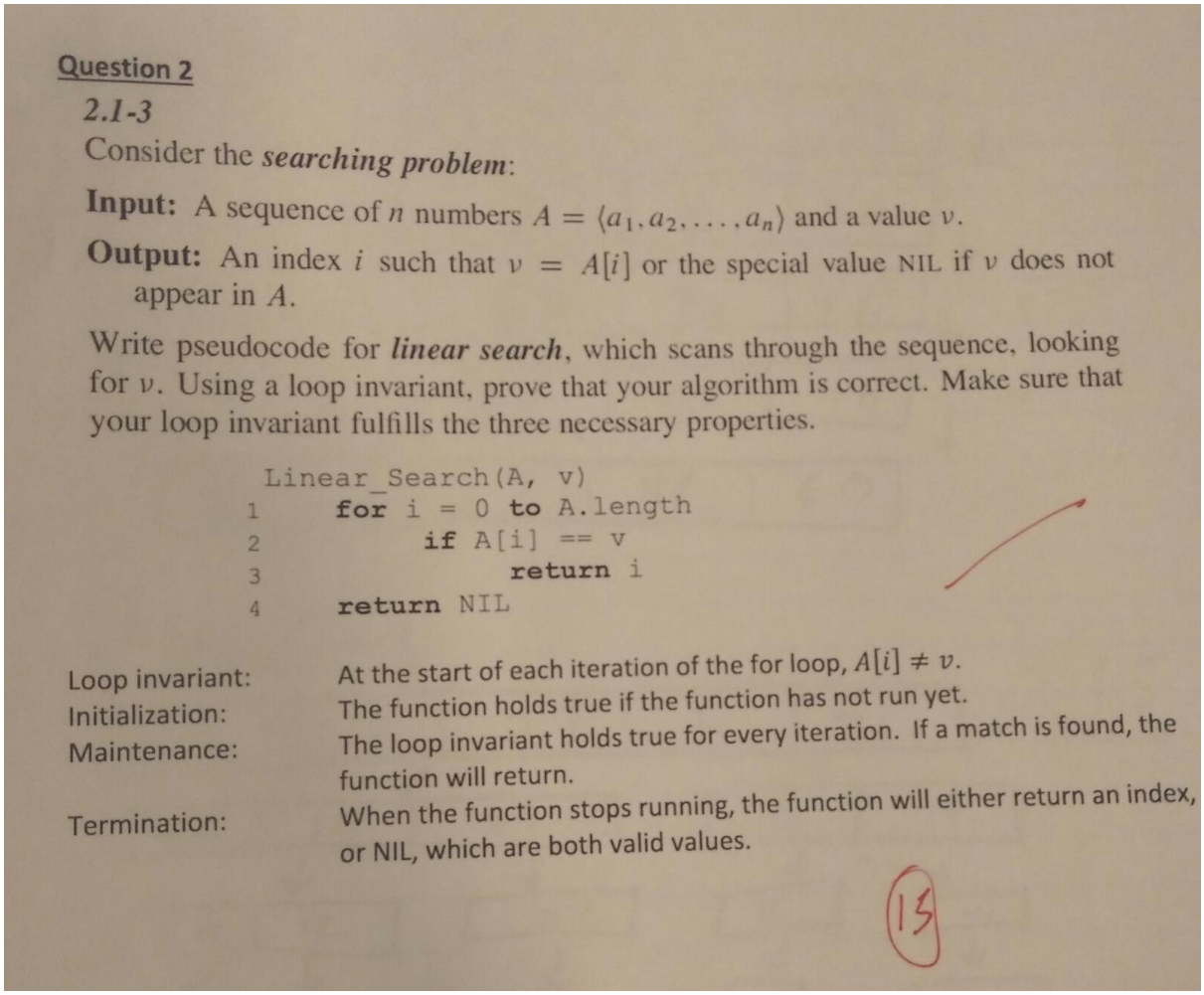
$2^{n/8} < n$ is true for $\boxed{2 \leq n \leq 43.}$

10



Q2.

Credit: Christopher Primes



Question 2

2.1-3

Consider the ***searching problem***:

Input: A sequence of n numbers $A = \langle a_1, a_2, \ldots, a_n \rangle$ and a value v.

Output: An index i such that $v = A[i]$ or the special value NIL if v does not appear in A.

Write pseudocode for ***linear search***, which scans through the sequence, looking for v. Using a loop invariant, prove that your algorithm is correct. Make sure that your loop invariant fulfills the three necessary properties.

```
Linear_Search(A, v)
1    for i = 0 to A.length
2         if A[i] == v
3              return i
4    return NIL
```

Loop invariant: At the start of each iteration of the for loop, $A[i] \neq v$.

Initialization: The function holds true if the function has not run yet.

Maintenance: The loop invariant holds true for every iteration. If a match is found, the function will return.

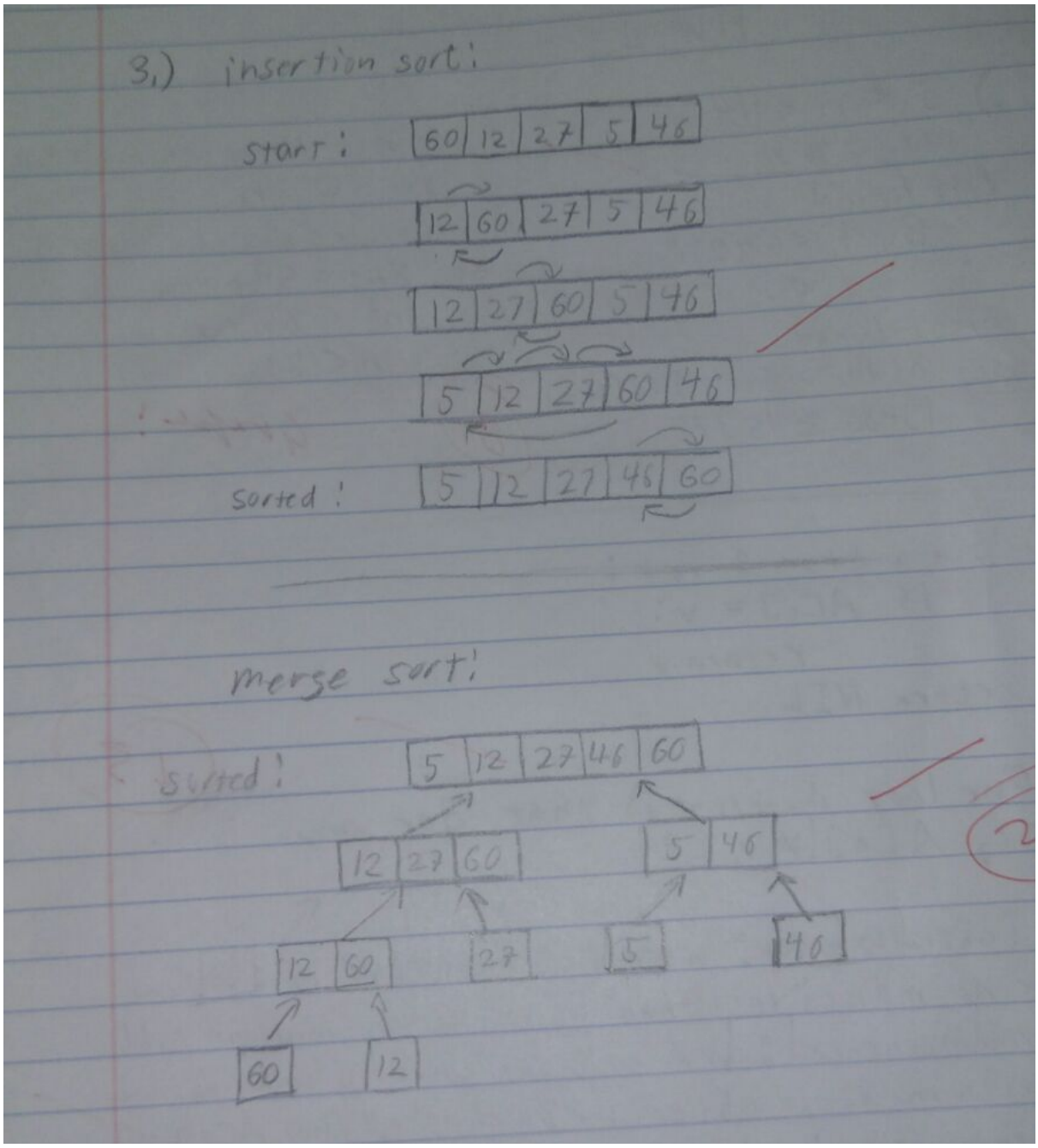
Termination: When the function stops running, the function will either return an index, or NIL, which are both valid values.

15

the array index should start from 1, not 0

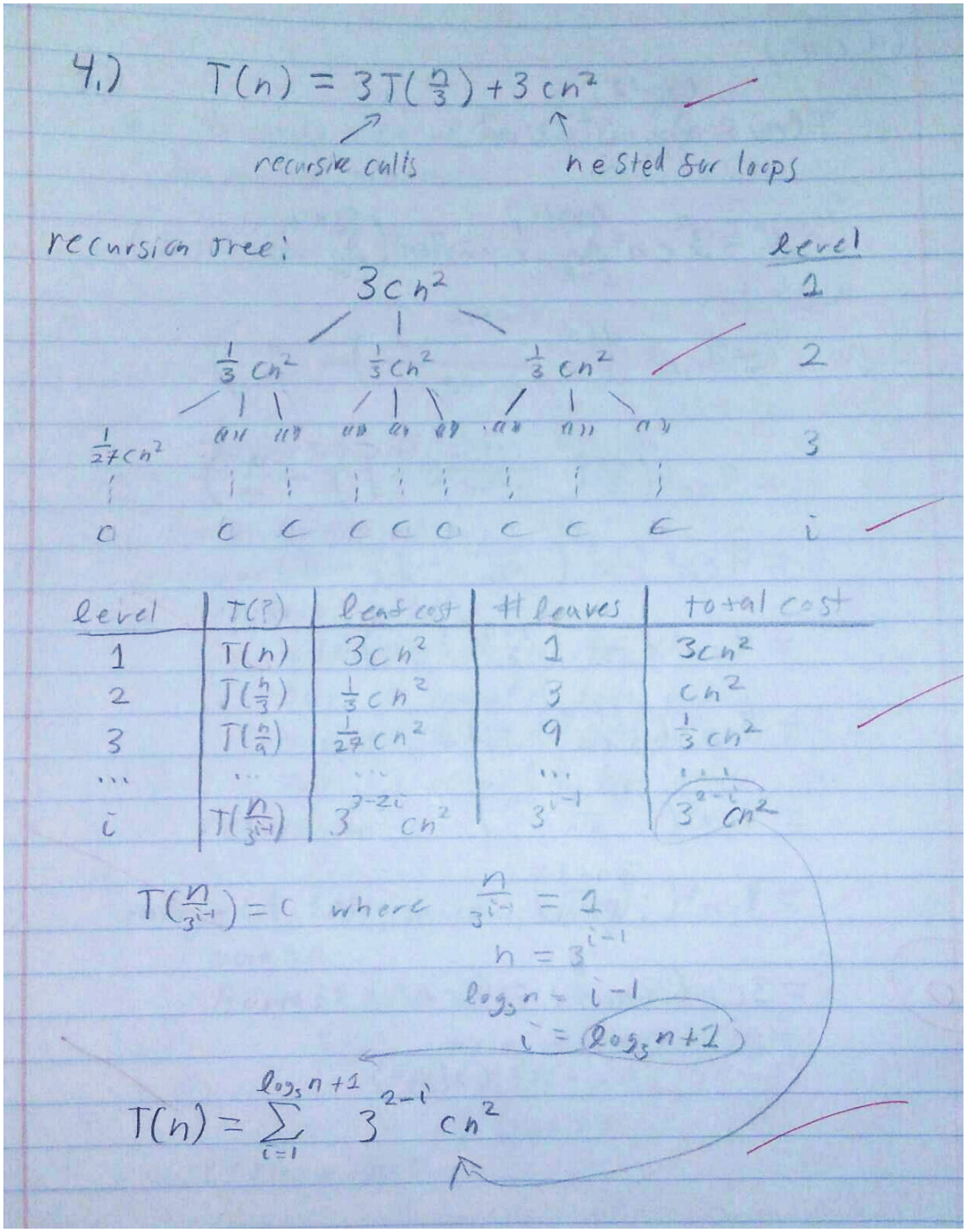
Q3.

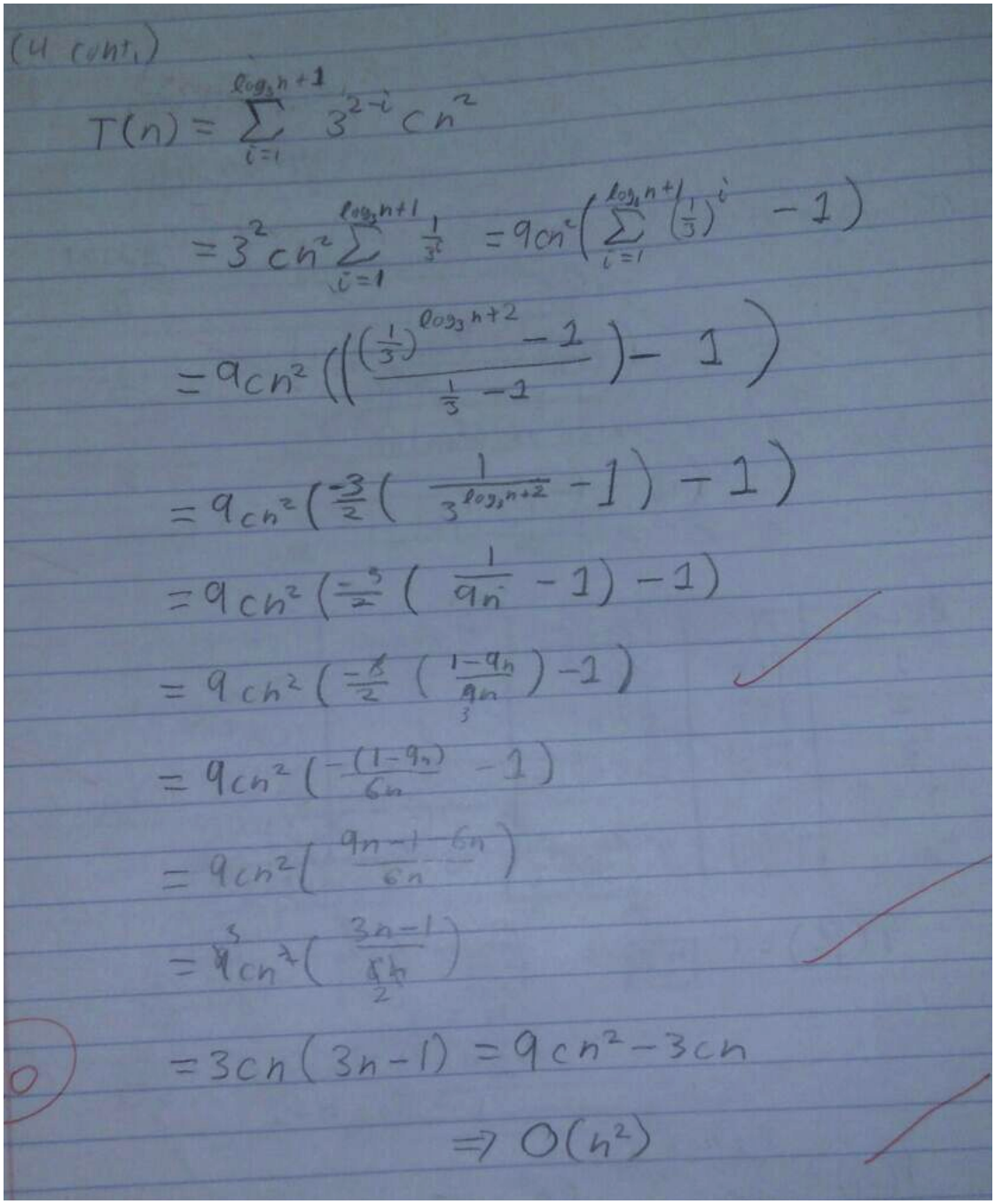
Credit: Wesley Wuzzo



Q4.

Credit: Wesley Wuzzo

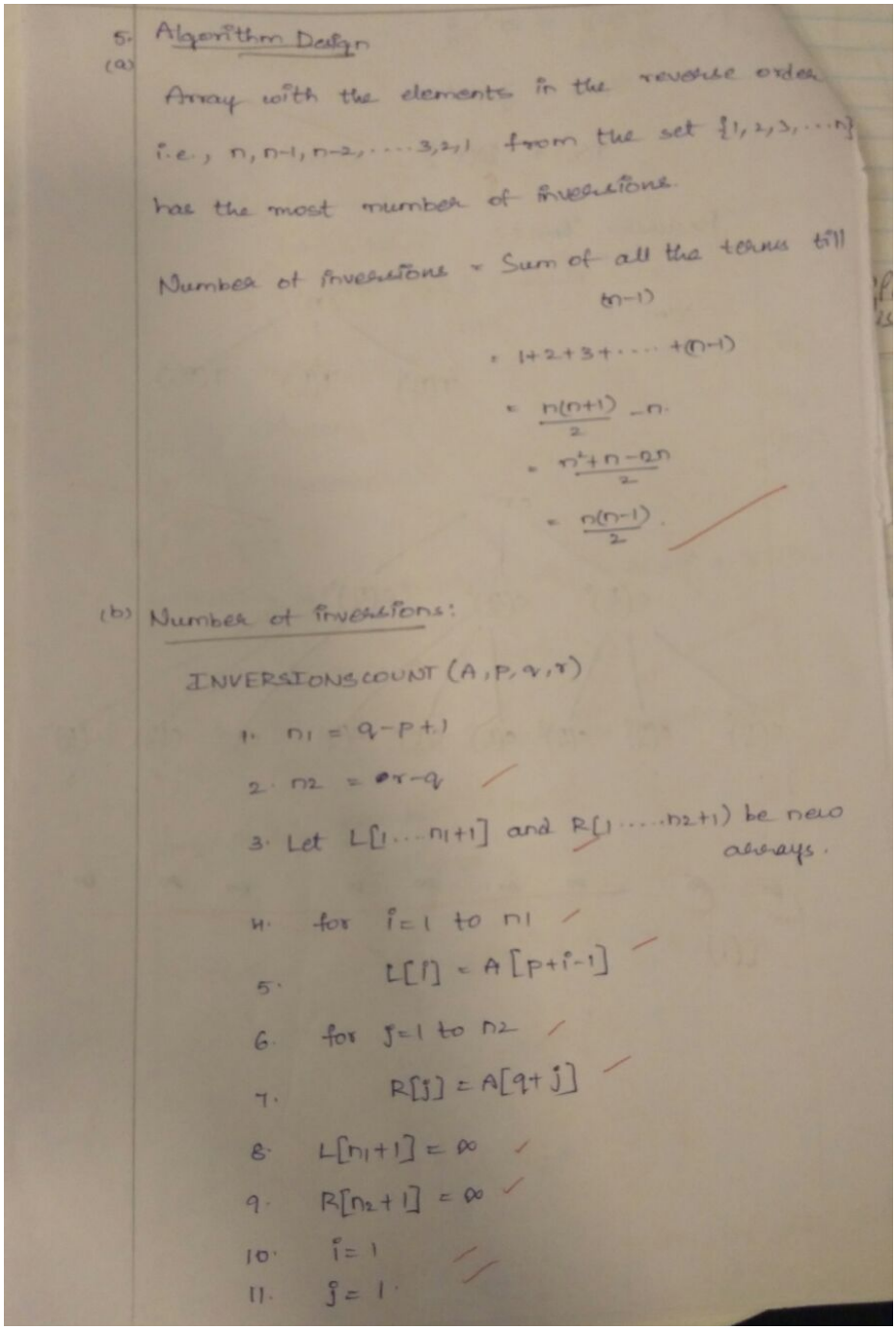




The range of *i* is not very accurate. I will review it in the class.

Q5.

Credit: Sai Krishnan Kiran



5. Algorithm Design

(a)

Array with the elements in the reverse order i.e., $n, n-1, n-2, \ldots 3,2,1$ from the set $\{1,2,3,\ldots n\}$ has the most number of inversions.

Number of inversions = Sum of all the terms till $(n-1)$

$= 1+2+3+\cdots+(n-1)$

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

$= \frac{n^2+n-2n}{2}$

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

(b) Number of inversions:

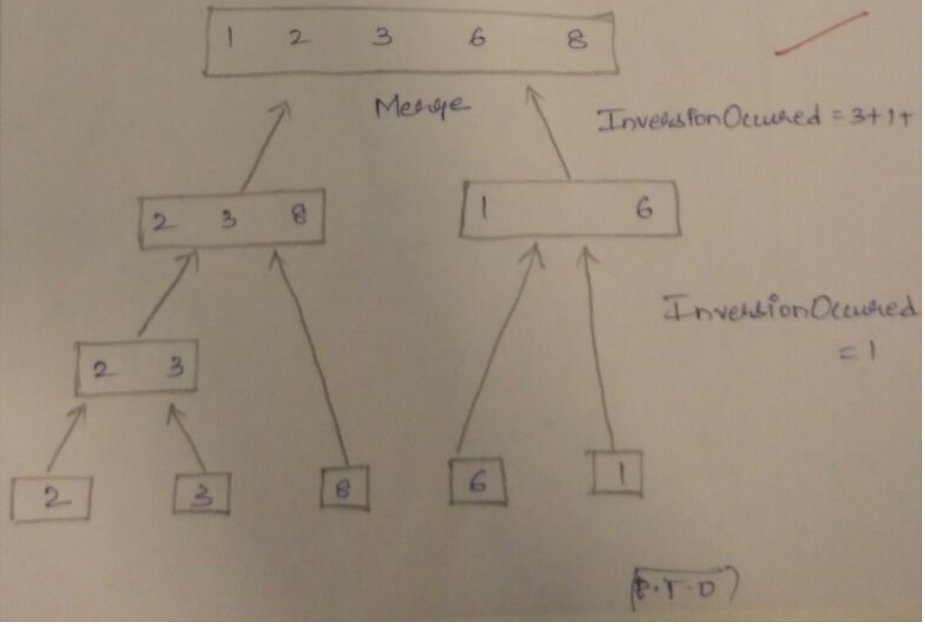
INVERSIONSCOUNT (A, P, q, r)

1. $n_1 = q-p+1$
2. $n_2 = r-q$
3. Let $L[1 \ldots n_1+1]$ and $R[1 \ldots n_2+1]$ be new always.
4. for $i=1$ to n_1
5. $L[i] = A[p+i-1]$
6. for $j=1$ to n_2
7. $R[j] = A[q+j]$
8. $L[n_1+1] = \infty$
9. $R[n_2+1] = \infty$
10. $i=1$
11. $j=1$

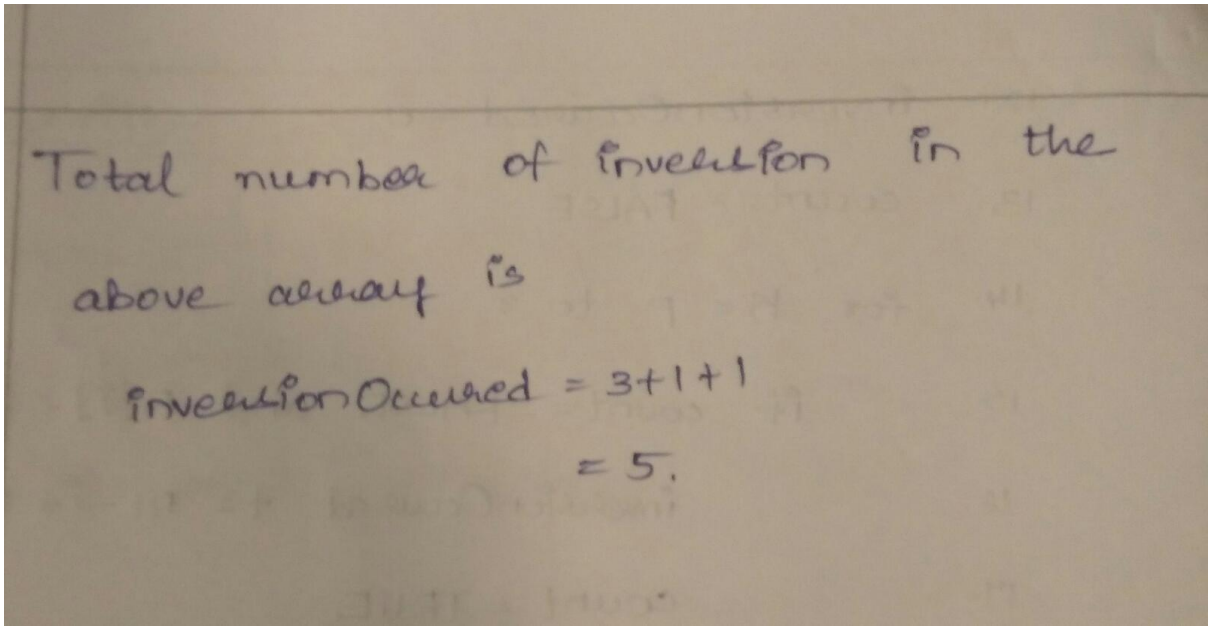
```
12.  inversionOccured = 0
13.  count = FALSE
14.  for k = p to r
15.      if count == FALSE and R[j] < L[i]
16.          inversionOccured += n1 - i + 1
17.          count = TRUE
18.      if L[i] ≤ R[j]
19.          A[k] = L[i]
20.          i = i + 1
21.      else A[k] = R[j]
22.          j = j + 1
23.          count = FALSE
24.  return inversionOccured
```

c. Working of the above algorithm on the array

<2, 3, 8, 6, 1>



P.T.O


Total number of inversion in the
above array is
inversionOccured = 3+1+1
= 5.