HW4

Q1

Credits: Ganesh Ramani

1 (i) Using the Array <1, 2, 3, 4, 5> As an Example, the q that is Always returned is $q = r$, in which r is the Last Index of Each Sub Array shown below.

20

The call of Quicksort (A, q+1, r) will not Execute More than the conditions and wont Surpass if statement in Quick Sort $q+1 > r$. We continue to recurse through the Quick Sort (A, P, q-1) until $q = 2$

Quicksort (A, 1, 5) ⇒ partition (A, 1, 5) returns $q = 5$
Quicksort (A, P, q-1) ⇒ partition (A, 1, 4) returns $q = 4$
Quicksort (A, P, q-1) ⇒ partition (A, 1, 3) returns $q = 3$
Quicksort (A, P, q-1) ⇒ partition (A, 1, 2) returns $q = 2$

1 (ii) In this case, further partitioning occurs when $p < q-1$ or $q+1 < r$.

Quicksort (A, 1, 5) ⇒ PARTITION (A, 1, 5) ; returns $q = 1$, P & r are Switched

Quicksort (A, q+1, r) ⇒ Quicksort (A, 2, 5) ⇒ partition (A, 2, 5) ; returns $q = 5$

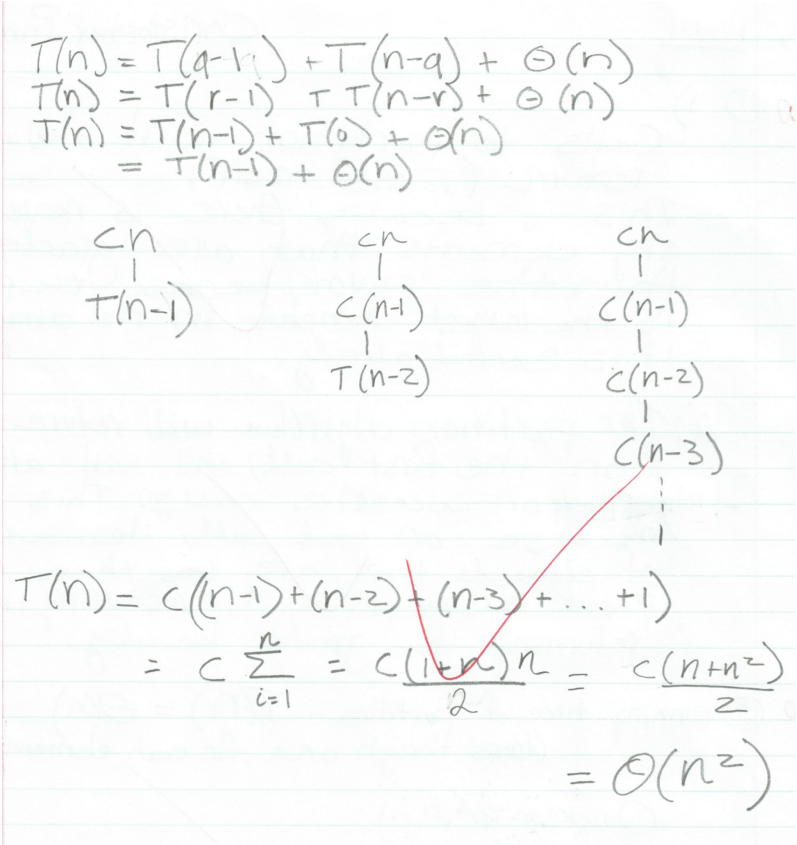
Quicksort (A, P, q-1) ⇒ Quicksort (A, 2, 4) ⇒ partition (A, 2, 4) ; returns $q = 2$
P & rth term switched.

Quicksort (A, q+1, r) ⇒ Quicksort (A, 3, 4) ⇒ partition (A, 3, 4), returns $q = 4$

HW4

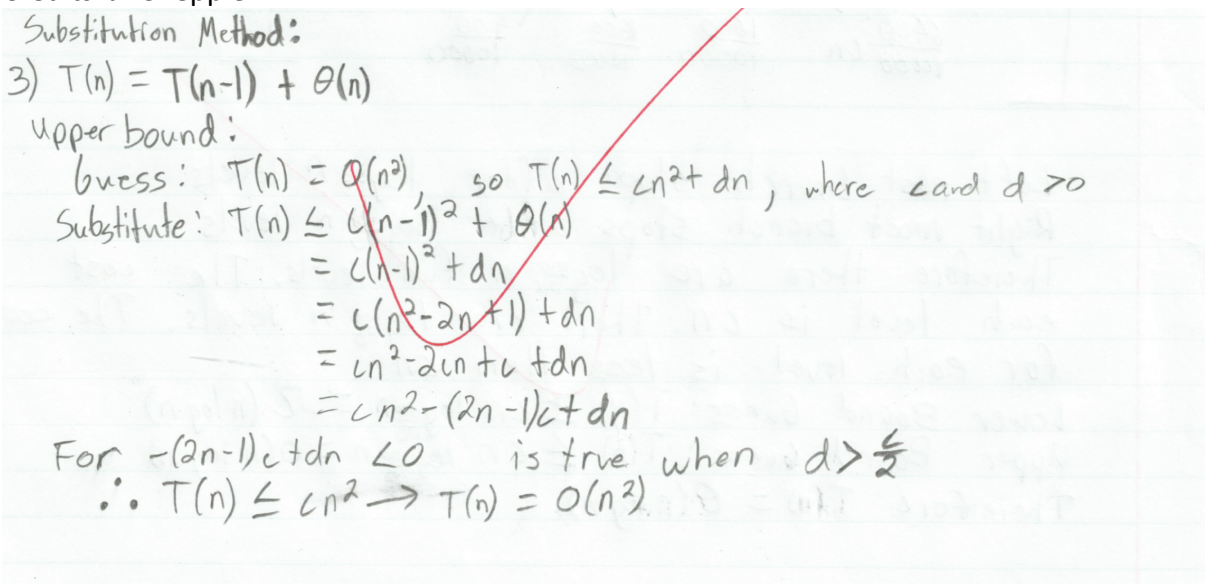
Q2.

Credits: Christopher primes



Q3.

Credits: JT Shepple



HW4

Lower Bound

Guess: $T(n) = \Omega(n^2)$, so $T(n) \geq cn^2 + dn$ where c and $d > 0$

Substitute: $T(n) \geq c(n-1)^2 + \Theta(1)$

$= c(n^2 - 2n + 1) + dn$

$= cn^2 - 2cn + c + dn$

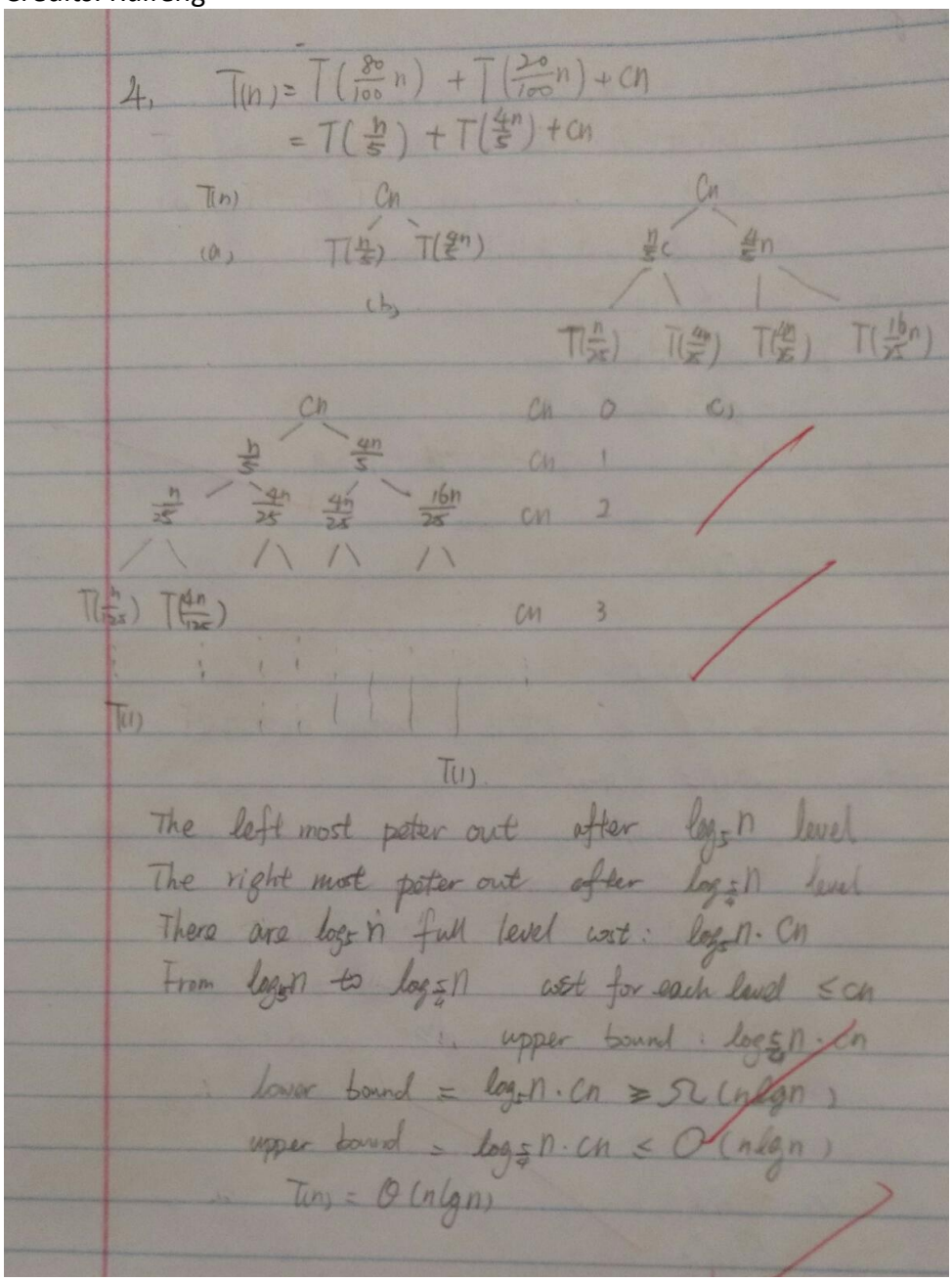
$= cn^2 - c(2n - 1) + dn$ is true when $d \leq \frac{c}{2}$

$\therefore T(n) \geq cn^2 \rightarrow T(n) = \Omega(n^2)$

so $T(n) = \Theta(n^2)$

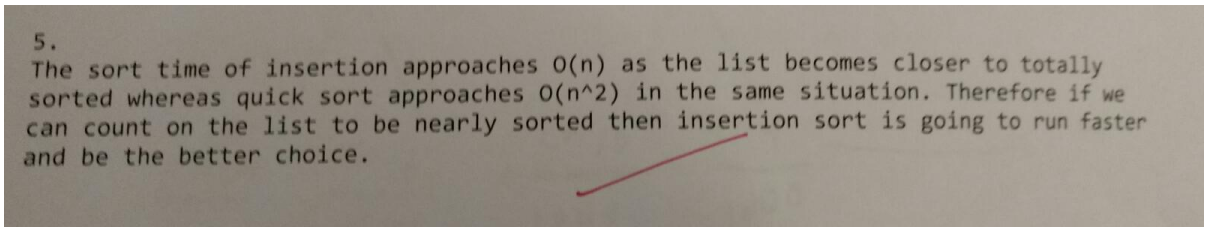
Q4.

Credits: Ruifeng



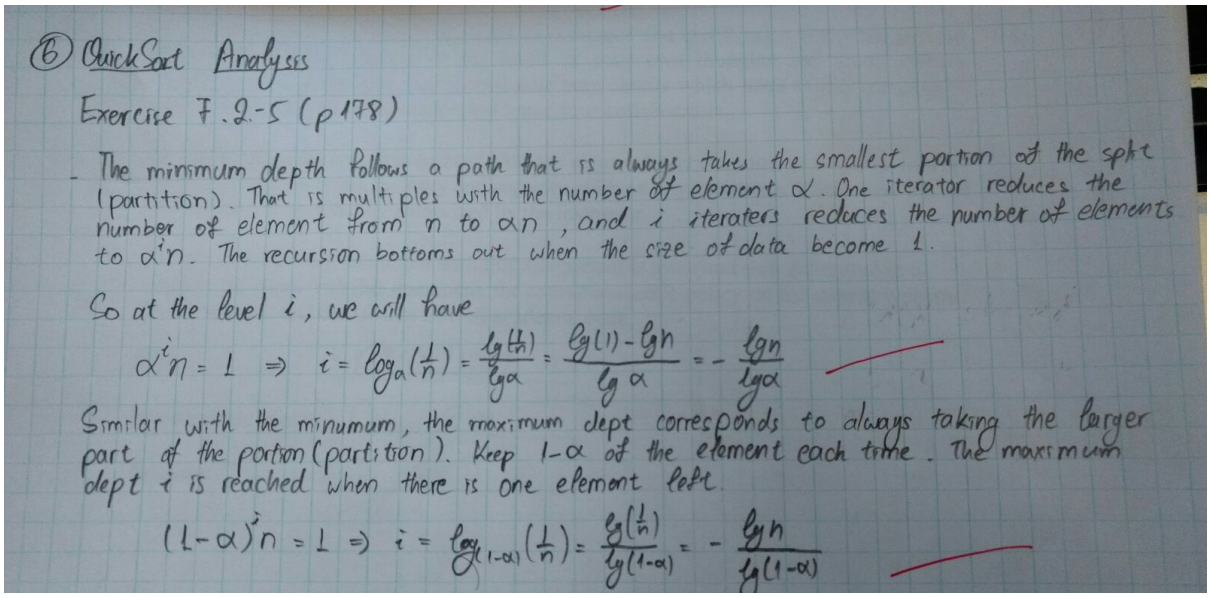
HW4

Q5.
Credits: Chris Marino



5.
The sort time of insertion approaches O(n) as the list becomes closer to totally sorted whereas quick sort approaches O(n^2) in the same situation. Therefore if we can count on the list to be nearly sorted then insertion sort is going to run faster and be the better choice.

Q6.
Credits: Chunn Kim



⑥ QuickSort Analysis

Exercise 7.2-5 (p178)

- The minimum depth follows a path that is always takes the smallest portion of the split (partition). That is multiples with the number of element α. One iterator reduces the number of element from n to αn, and i iterators reduces the number of elements to $\alpha^i n$. The recursion bottoms out when the size of data become 1.

So at the level i, we will have

$$\alpha^i n = 1 \Rightarrow i = \log_\alpha\left(\frac{1}{n}\right) = \frac{\lg\left(\frac{1}{n}\right)}{\lg\alpha} = \frac{\lg(1) - \lg n}{\lg\alpha} = -\frac{\lg n}{\lg\alpha}$$

Similar with the minumum, the maximum dept corresponds to always taking the larger part of the portion (partition). Keep $1-\alpha$ of the element each time. The maximum dept i is reached when there is one element left.

$$(1-\alpha)^i n = 1 \Rightarrow i = \log_{(1-\alpha)}\left(\frac{1}{n}\right) = \frac{\lg\left(\frac{1}{n}\right)}{\lg(1-\alpha)} = -\frac{\lg n}{\lg(1-\alpha)}$$