

## HW4

Q1

Credits: Ganesh Ramani

1 (i) using the Array  $\langle 1, 2, 3, 4, 5 \rangle$  as an example, the  $q$  that is always returned is  $q=5$ , in which  $q$  is the last index of each sub array shown below.

The call of Quicksort  $(A, q+1, \text{ })$  will not execute more than the conditions and won't surpass if statement in quick sort  $q+1 > r$ . We continue to recurse through the Quicksort  $(A, p, q-1)$  until  $q=2$

Quicksort  $(A, 1, 5) \Rightarrow$  partition  $(A, 1, 5)$  returns  $q=5$

Quicksort  $(A, p, q-1) \Rightarrow$  partition  $(A, 1, 4)$  returns  $q=4$

Quicksort  $(A, p, q-1) \Rightarrow$  partition  $(A, 1, 3)$  returns  $q=3$

Quicksort  $(A, p, q-1) \Rightarrow$  partition  $(A, 1, 2)$  returns  $q=2$

Quicksort  $(A, p, q-1) \Rightarrow$  partition  $(A, 1, 2)$  returns  $q=2$

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

Quicksort  $(A, 1, 5) \Rightarrow$  PARTITION  $(A, 1, 5)$ ; returns  $q=1$ ,  $p, r$  are switched

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

Quicksort  $(A, p, q-1) \Rightarrow$  Quicksort  $(A, 2, 4) \Rightarrow$  partition  $(A, 2, 4)$ ; returns  $q=2$   
 $p, r$  terms switched.

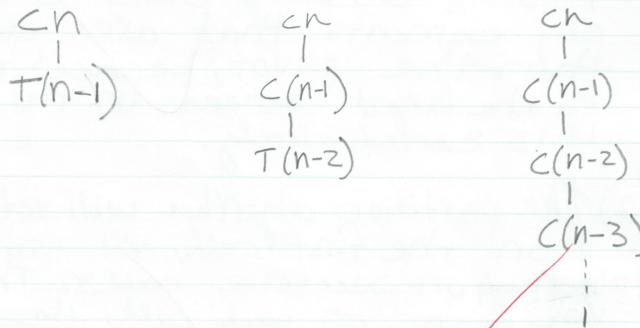
Quicksort  $(A, q+1, r) \Rightarrow$  Quicksort  $(A, 3, 4) \Rightarrow$  partition  $(A, 3, 4)$ ; returns  $q=4$ .

## HW4

Q2.

Credits: Christopher primes

$$\begin{aligned} T(n) &= T(n-1) + T(n-n) + \Theta(n) \\ T(n) &= T(n-1) + T(n-n) + \Theta(n) \\ T(n) &= T(n-1) + T(0) + \Theta(n) \\ &= T(n-1) + \Theta(n) \end{aligned}$$



$$\begin{aligned} T(n) &= c((n-1) + (n-2) + (n-3) + \dots + 1) \\ &= c \sum_{i=1}^n = c \frac{(1+n)n}{2} = c \frac{(n+n^2)}{2} \\ &= \Theta(n^2) \end{aligned}$$

Q3.

Credits: JT Shepple

Substitution Method:

$$3) T(n) = T(n-1) + \Theta(n)$$

Upper bound:

Guess:  $T(n) = O(n^3)$  so  $T(n) \leq cn^2 + dn$ , where  $c$  and  $d > 0$

$$\text{Substitute: } T(n) \leq (n-1)^2 + \Theta(n)$$

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

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

$$= n^2 - 2n + 1 + dn$$

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

For  $-(2n-1)c + dn < 0$  is true when  $d > \frac{c}{2}$

$$\therefore T(n) \leq cn^2 \rightarrow T(n) = O(n^2)$$

## HW4

Lower Bound

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

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

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

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

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

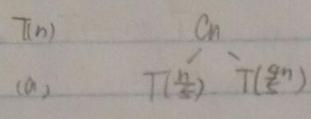
$$\therefore T(n) \geq cn^2 \rightarrow T(n) = \sqrt{2}(n^2)$$

$$\text{so } T(n) = \Theta(n^2)$$

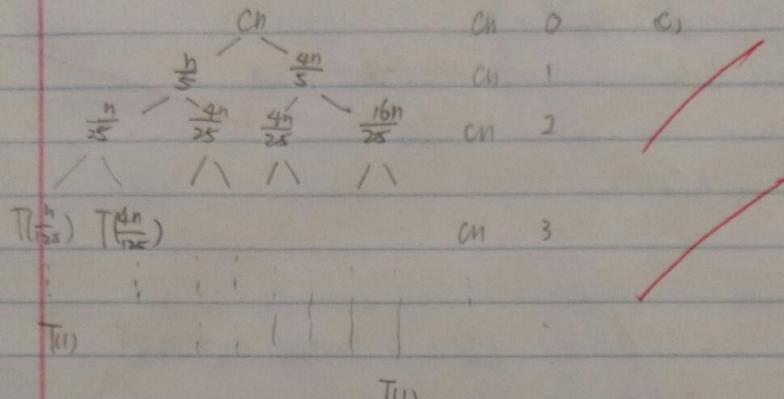
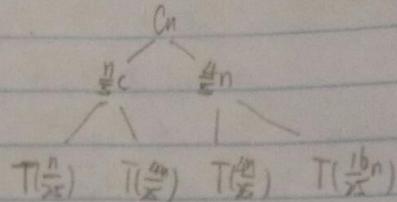
Q4.

Credits: Ruifeng

$$4, \quad T(n) = T\left(\frac{80}{100}n\right) + T\left(\frac{20}{100}n\right) + cn \\ = T\left(\frac{n}{5}\right) + T\left(\frac{4n}{5}\right) + cn$$



(b)



The leftmost peter out after  $\log_5 n$  level

The rightmost peter out after  $\log_{4/5} n$  level

There are  $\log_5 n$  full level cost:  $\log_5 n \cdot cn$

From  $\log_5 n$  to  $\log_{4/5} n$  cost for each level  $\leq cn$

∴ upper bound:  $\log_5 n \cdot cn$

lower bound:  $\log_{4/5} n \cdot cn \geq \sqrt{2}(\log_5 n)$

upper bound:  $\log_{4/5} n \cdot cn \leq O(\log_5 n)$

$$T(n) = \Theta(\log_5 n)$$

## 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 (p 178)

- 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  $\alpha$ . One iterator reduces the number of element from  $n$  to  $\alpha n$ , 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

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

Similar with the minimum, 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.

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