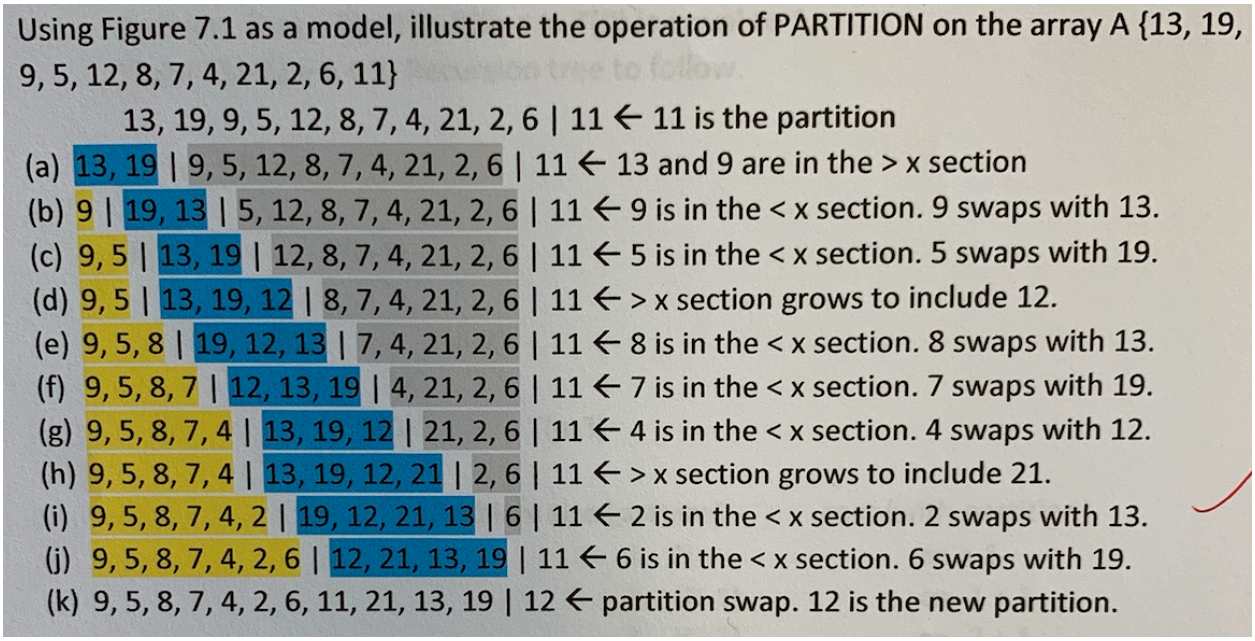
HW 4

1. From Adam Keen



Using Figure 7.1 as a model, illustrate the operation of PARTITION on the array A {13, 19, 9, 5, 12, 8, 7, 4, 21, 2, 6, 11}

13, 19, 9, 5, 12, 8, 7, 4, 21, 2, 6 | 11 ← 11 is the partition

(a) 13, 19 | 9, 5, 12, 8, 7, 4, 21, 2, 6 | 11 ← 13 and 9 are in the > x section

(b) 9 | 19, 13 | 5, 12, 8, 7, 4, 21, 2, 6 | 11 ← 9 is in the < x section. 9 swaps with 13.

(c) 9, 5 | 13, 19 | 12, 8, 7, 4, 21, 2, 6 | 11 ← 5 is in the < x section. 5 swaps with 19.

(d) 9, 5 | 13, 19, 12 | 8, 7, 4, 21, 2, 6 | 11 ← > x section grows to include 12.

(e) 9, 5, 8 | 19, 12, 13 | 7, 4, 21, 2, 6 | 11 ← 8 is in the < x section. 8 swaps with 13.

(f) 9, 5, 8, 7 | 12, 13, 19 | 4, 21, 2, 6 | 11 ← 7 is in the < x section. 7 swaps with 19.

(g) 9, 5, 8, 7, 4 | 13, 19, 12 | 21, 2, 6 | 11 ← 4 is in the < x section. 4 swaps with 12.

(h) 9, 5, 8, 7, 4 | 13, 19, 12, 21 | 2, 6 | 11 ← > x section grows to include 21.

(i) 9, 5, 8, 7, 4, 2 | 19, 12, 21, 13 | 6 | 11 ← 2 is in the < x section. 2 swaps with 13.

(j) 9, 5, 8, 7, 4, 2, 6 | 12, 21, 13, 19 | 11 ← 6 is in the < x section. 6 swaps with 19.

(k) 9, 5, 8, 7, 4, 2, 6, 11, 21, 13, 19 | 12 ← partition swap. 12 is the new partition.

2. From Victor Espaillat

Let $A = \langle 9, 8, 7, 6, 5, 4, 3, 2, 1, 0 \rangle$, an array of $n = 10$ distinct elements sorted in descending order. Let's see what happens when we call QUICKSORT$(A, p = 1, r = 10)$.

An initial call to PARTITION would be made. i would be initialized to $i = p - 1 = 0$. Then the rightmost, and therefore smallest element of A, 0, would be picked as the pivot. The *if* statement inside the *for loop* would always fail resulting in i never incrementing. Therefore the $\leq x$ subarray would remain empty, while the $\geq x$ subarray would contain all other elements in the array. Then element 0 would be swapped with the leftmost (largest) element, 9. Resulting in $A = \langle 0, 8, 7, 6, 5, 4, 3, 2, 1, 9 \rangle$ and $i + 1 = 0 + 1 = 1$. So the first call to PARTITION would return index 1.

Returning to QUICKSORT, we would have $q = 1$. Then there are two recursive calls: QUICKSORT$(A, p = 1, q - 1)$ and QUICKSORT$(A, p = (q + 1), r = 10)$. The first would lead to nothing since it would be calling an instance of QUICKSORT with $p = 1, r = 0$ resulting in $p \not< r$. The second would work on subarray $A[q + 1..r] = A[2..10] = \langle 8, 7, 6, 5, 4, 3, 2, 1, 9 \rangle$.

This second call to QUICKSORT will again call PARTITION and work on subarray $A[2..10] = \langle 8, 7, 6, 5, 4, 3, 2, 1, 9 \rangle$. Instead of the rightmost element being the smallest, we now have it being the largest, so the pivot will be element 9. The $\leq x$ subarray will contain all other elements and the $\geq x$ subarray will be empty; element 9 will stay in place. The return value will equal element 9's index, which is 10.

Returning to QUICKSORT, we would have $q = 10$. For the two recursive calls, now the second will lead to nothing since QUICKSORT$(A, p = (q + 1), r = 10)$ would lead to $p \not< r$. The first call however, QUICKSORT$\big(A, p = 2, r = (q - 1)\big)$ will work on subarray $A[2..9] = \langle 8, 7, 6, 5, 4, 3, 2, 1 \rangle$.

Now the input is similar to how we started, with the rightmost element being the smallest element, to be taken as the pivot. The result is the same; the rightmost (smallest) element being swapped with the leftmost (largest) element, resulting in $A[2..9] = \langle 1, 7, 6, 5, 4, 3, 2, 8 \rangle$ and index 2 being returned.

∴ The pattern for the return value of PARTITION would continue in this manner : first return 1, then n, then 2, then $n - 1$, then 3, then $n - 2$, and so on, until all indices have been returned once. The indices would indicate smallest element, then largest element, then second smallest, then second largest, and so on.

3. From Victor M Espaillat

	QUICKSORT(A, p, r)	Cost	Times
1.	**if** $p < r$	c_1	1
2.	$q =$ PARTITION(A, p, r)	c_2	1
3.	QUICKSORT$(A,\ p,\ q-1)$	$\mathrm{T}(n-1)$	1
4.	QUICKSORT$(A,\ q+1,\ r)$	$\mathrm{T}(n-1)$	1

Since the recursive call on an array of size 0 just returns, $\mathrm{T}(0) = \Theta(1)$. The recurrence for the worst case running time of QUICKSORT becomes

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

$$\begin{aligned}\mathrm{T}(n) &= cn + c(n-1) + c(n-2) + c(n-3) + \cdots + c \cdot 1\\ &= c(n + (n-1) + (n-2) + (n-3) + \cdots + 1)\\ &= c\sum_{i=0}^{n}(n-i) = c\sum_{i=0}^{n} i = \frac{c\,(n+1)n}{2}\\ &= \Theta(n^2)\end{aligned}$$

4. From Phong Vo

$T(n) = T(n-1) + cn$ (c: positive const.)

* Upper bound: $T(n) \leq T(n-1) + cn$

Guess $T(n) = O(n^2)$

$T(n) \leq dn^2$ (d: pos. const.)

$T(n-1) \leq d(n-1)^2$

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

Substitution: $T(n) \leq d(n^2 - 2n + 1) + cn$

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

$\leq dn^2$ if $-n(2d-c) + d \leq 0$

$\Rightarrow 2d - c \geq 0 \Rightarrow c \leq 2d$

Hence: upper bound of $T(n) = O(n^2)$

* Lower bound: $T(n) \geq T(n-1) + cn$ (c: pos. const.)

Guess $T(n) = \Omega(n^2)$

$\geq dn^2$ (d: pos. const.)

$T(n-1) \geq d(n-1)^2$

Substitution: $T(n) \geq d(n-1)^2 + cn$

$= d(n^2 - 2n + 1) + cn$

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

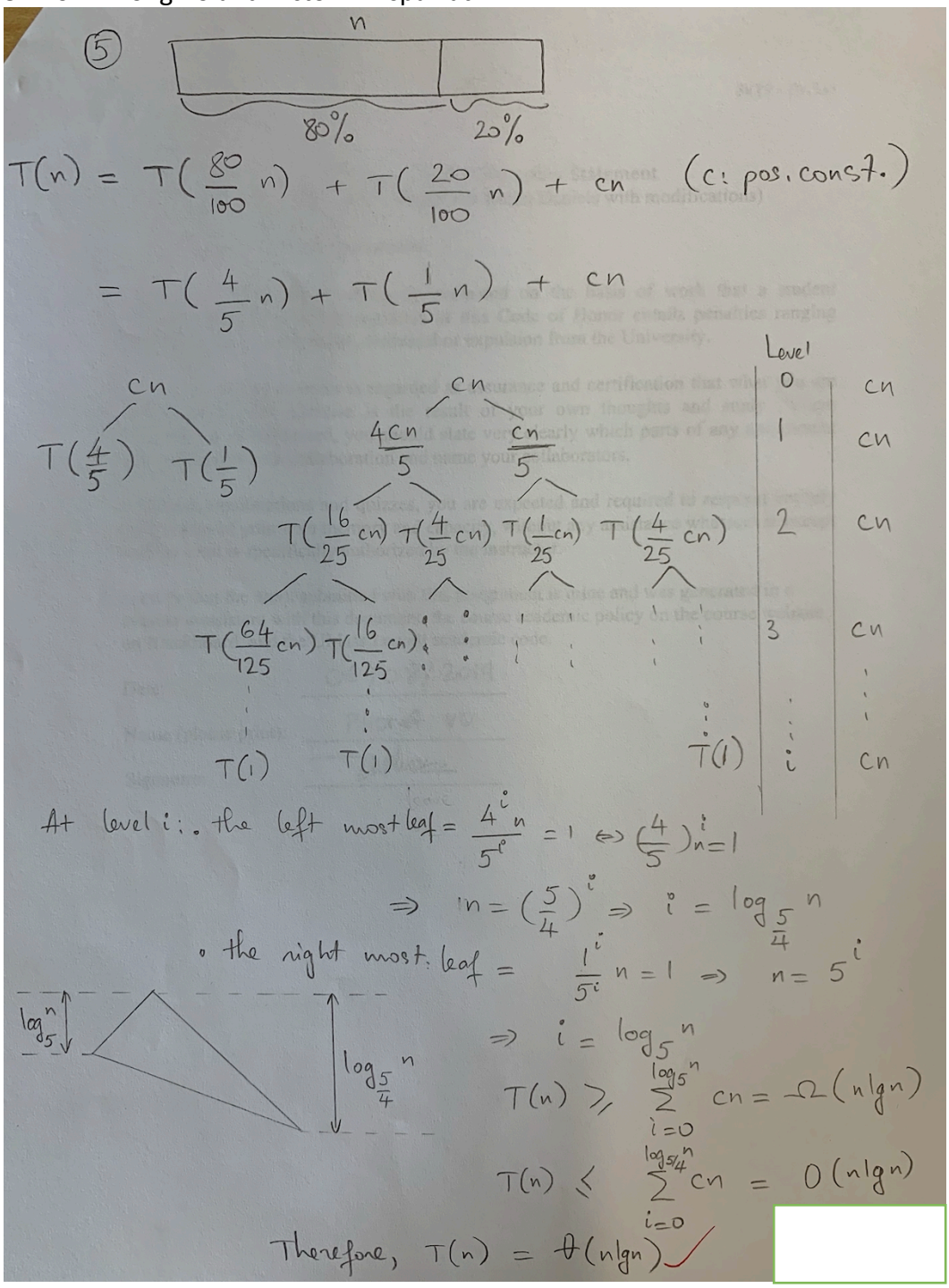
$\geq dn^2$ if $(c-2d)n + d \geq 0$

$\Rightarrow c - 2d \geq 0 \Rightarrow c \geq 2d$

Hence, lower bound of $T(n) = \Omega(n^2)$

We have $T(n) = O(n^2)$ and $T(n) = \Omega(n^2)$, so $\boxed{T(n) = \Theta(n^2)}$

5. From Phong Vo and Victor M Espaillat



Theorem:

If $T(n) = T(an) + T((1-a)n) + \Theta(n)$ where $0 < a < \frac{1}{2}$

then $T(n) = \Theta(n \lg n)$

In this case $T(n) = T\left(\frac{n}{5}\right) + T\left(\frac{4n}{5}\right) + \Theta(n)$

$a = \frac{1}{5}$, $(1-a) = \frac{4}{5}$, and $0 < a < \frac{1}{2}$

$\therefore T(n) = \Theta(n \lg n)$