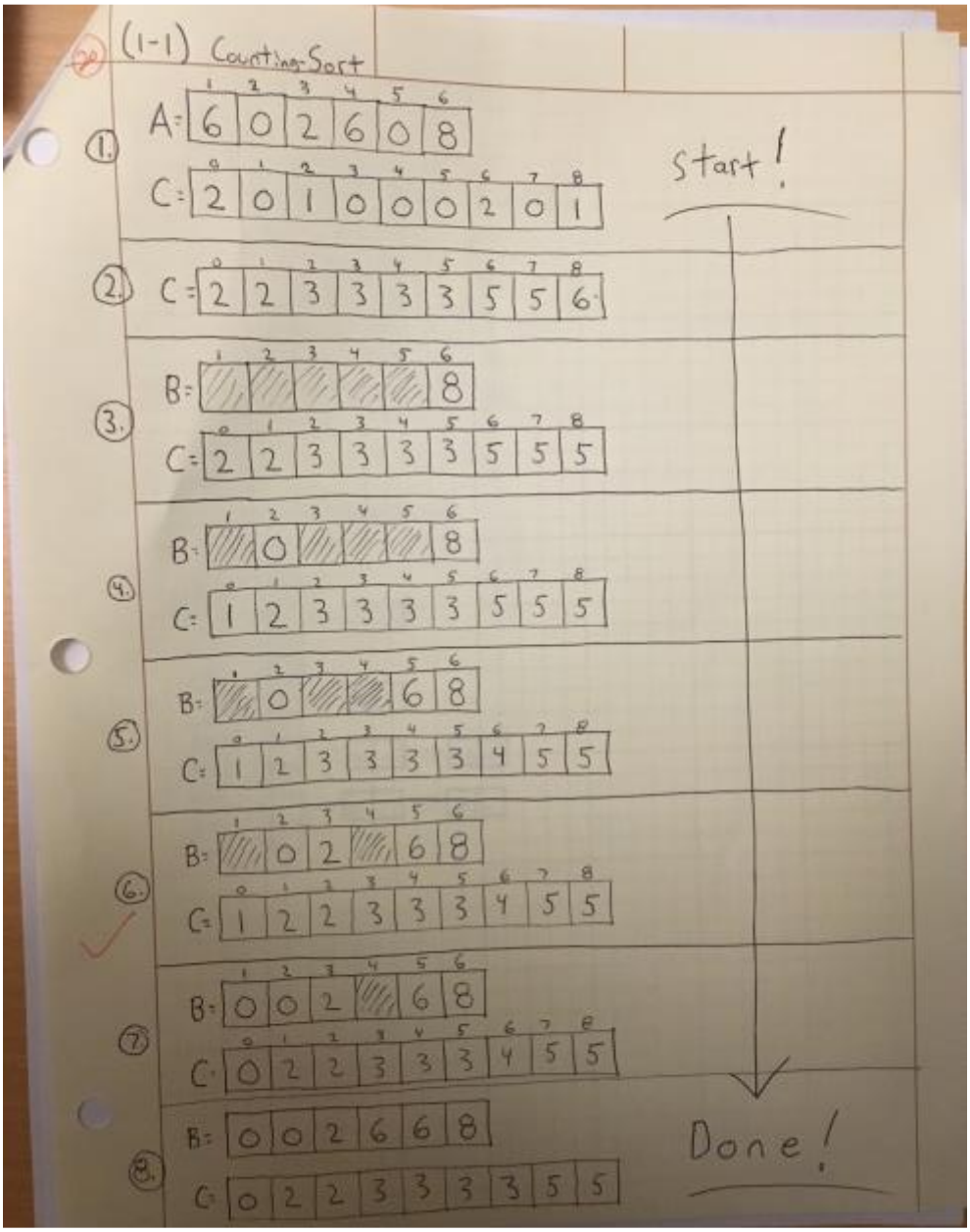
Prof. Benyuan Liu
Fall - 2017

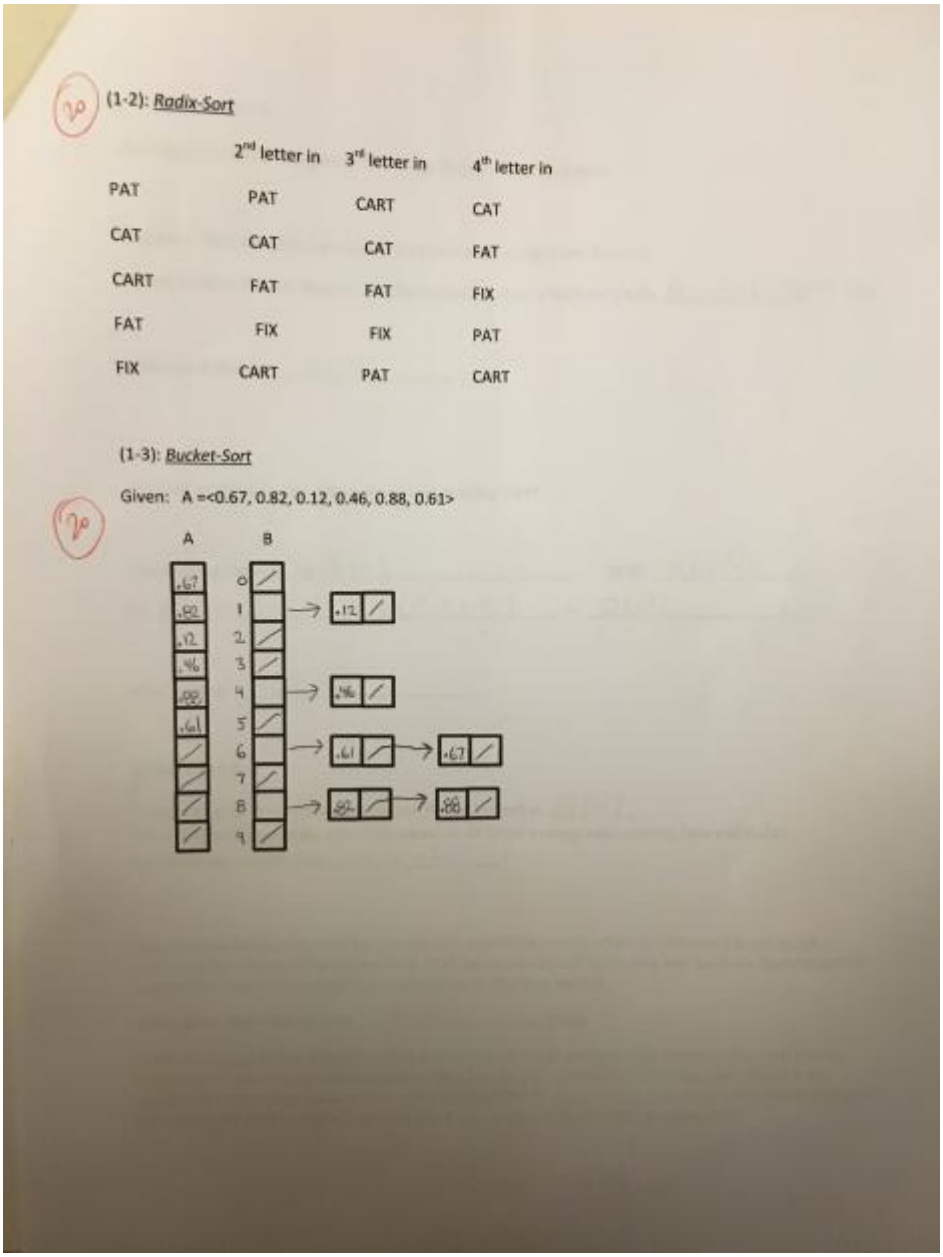
ANALYSIS OF ALGORITHM - HW -5 SOLUTIONS

1. (1)Credits: Ryan Cauble



Prof. Benyuan Liu
Fall - 2017

1 (2)&(3)Credits: Ryan Cauble



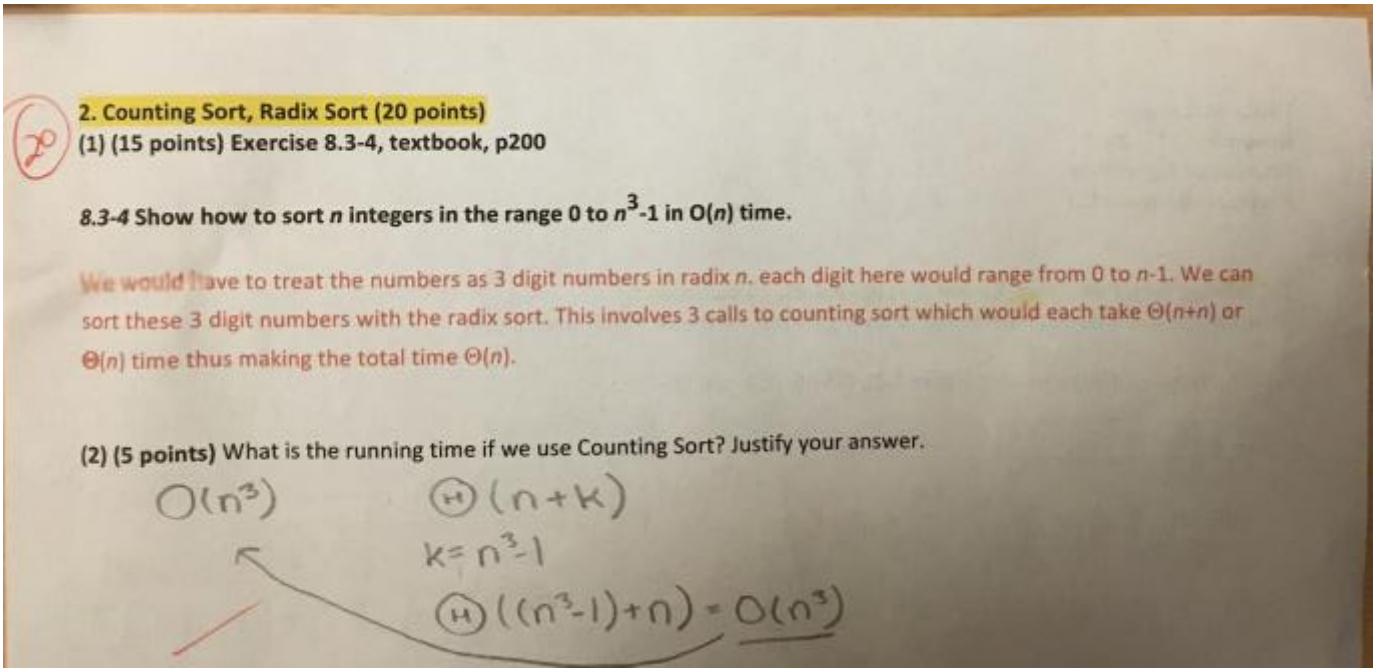
(1-2): Radix-Sort

	2nd letter in	3rd letter in	4th letter in
PAT	PAT	CART	CAT
CAT	CAT	CAT	FAT
CART	FAT	FAT	FIX
FAT	FIX	FIX	PAT
FIX	CART	PAT	CART

(1-3): Bucket-Sort

Given: A =<0.67, 0.82, 0.12, 0.46, 0.88, 0.61>

2. Credits: Taylor M. Langlois



2. Counting Sort, Radix Sort (20 points)
(1) (15 points) Exercise 8.3-4, textbook, p200

8.3-4 **Show how to sort *n* integers in the range 0 to n^3-1 in O(*n*) time.**

We would have to treat the numbers as 3 digit numbers in radix *n*. each digit here would range from 0 to *n*-1. We can sort these 3 digit numbers with the radix sort. This involves 3 calls to counting sort which would each take $\Theta(n+n)$ or $\Theta(n)$ time thus making the total time $\Theta(n)$.

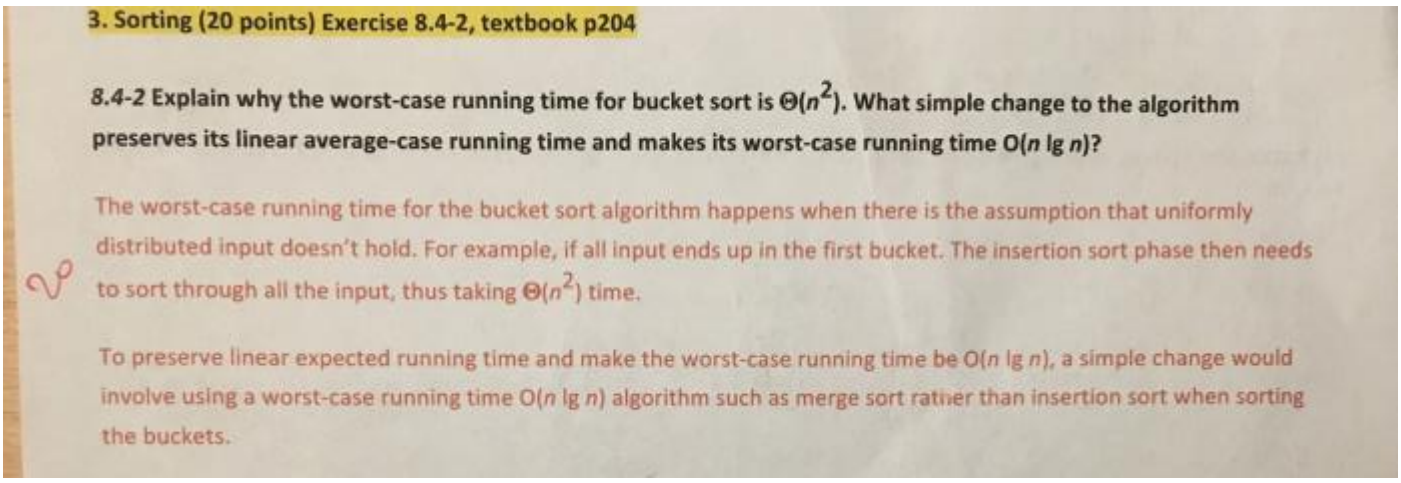
(2) (5 points) What is the running time if we use Counting Sort? Justify your answer.

$O(n^3)$ $\Theta(n+k)$

$k = n^3-1$

$\Theta((n^3-1)+n) = O(n^3)$

3. Credits: Taylor M. Langlois



3. Sorting (20 points) Exercise 8.4-2, textbook p204

8.4-2 **Explain why the worst-case running time for bucket sort is $\Theta(n^2)$. What simple change to the algorithm preserves its linear average-case running time and makes its worst-case running time O(*n* lg *n*)?**

The worst-case running time for the bucket sort algorithm happens when there is the assumption that uniformly distributed input doesn't hold. For example, if all input ends up in the first bucket. The insertion sort phase then needs to sort through all the input, thus taking $\Theta(n^2)$ time.

To preserve linear expected running time and make the worst-case running time be O(*n* lg *n*), a simple change would involve using a worst-case running time O(*n* lg *n*) algorithm such as merge sort rather than insertion sort when sorting the buckets.