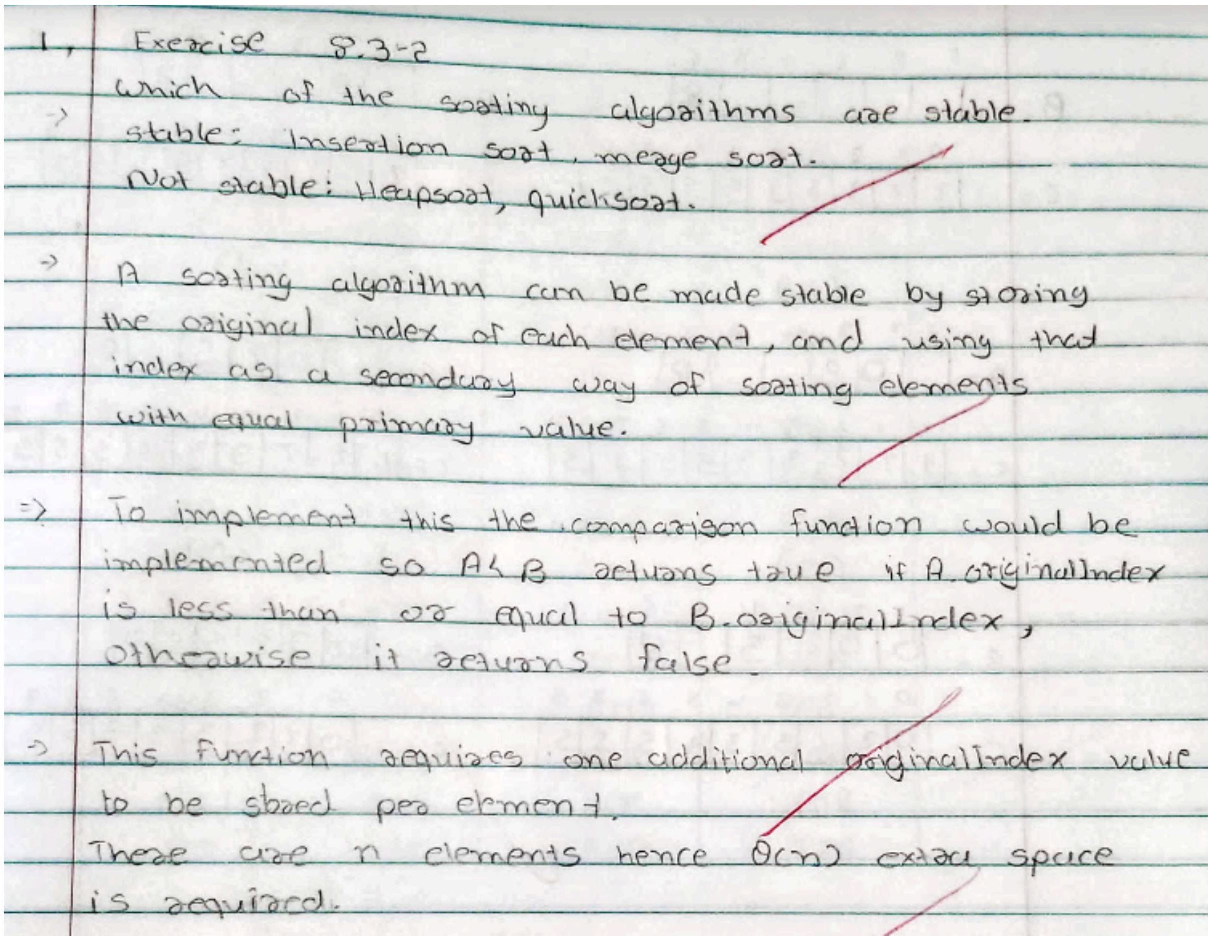
1. Prachi Patel



1, Exercise 8.3-2

Which of the sorting algorithms are stable.

-> stable: Insertion sort, merge sort.

Not stable: Heapsort, quicksort.

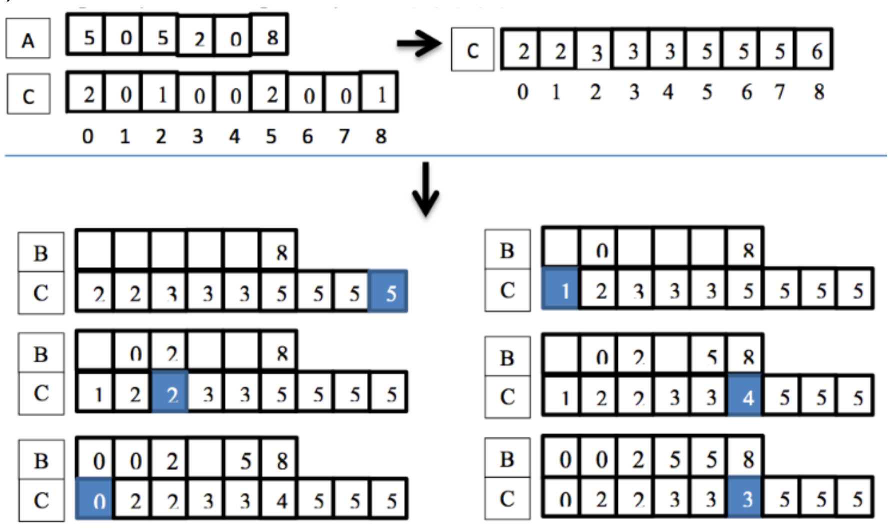
-> A sorting algorithm can be made stable by storing the original index of each element, and using that index as a secondary way of sorting elements with equal primary value.

=> To implement this the comparison function would be implemented so $A < B$ returns true if A.originalIndex is less than or equal to B.originalIndex, otherwise it returns false.

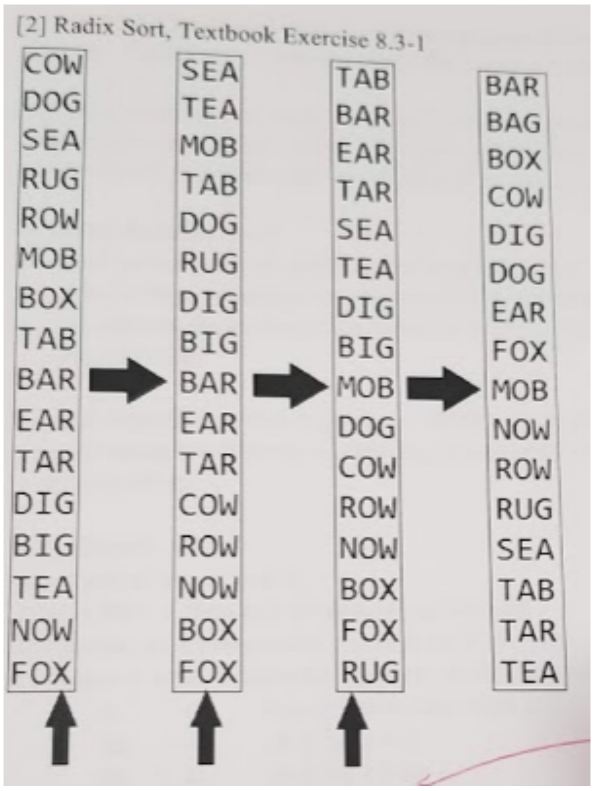
-> This function requires one additional originalIndex value to be stored per element.

There are n elements hence $\Theta(n)$ extra space is required.

2(1)

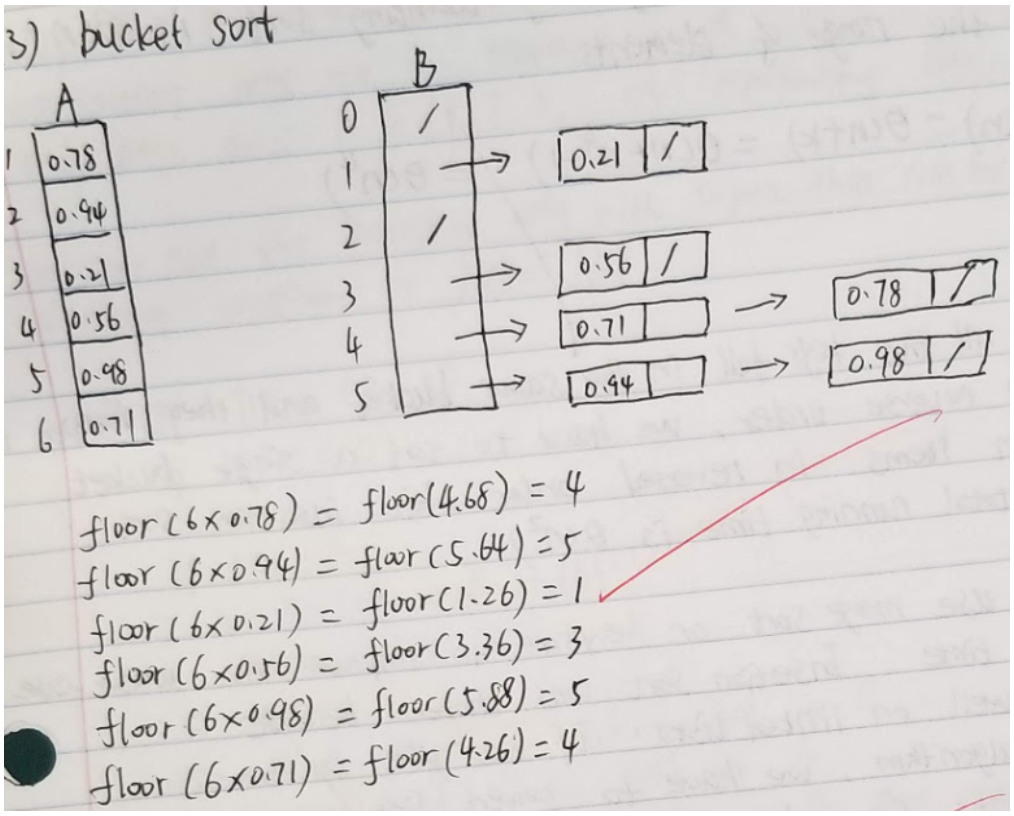


2(2) Charn Supparanya


[2] Radix Sort, Textbook Exercise 8.3-1

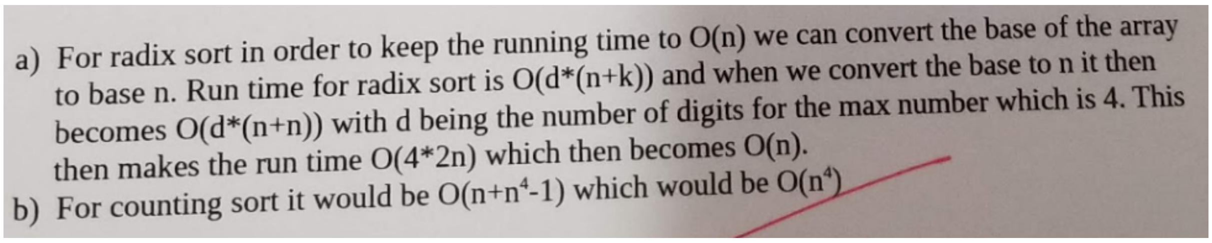
COW	SEA	TAB	BAR
DOG	TEA	BAR	BAG
SEA	MOB	EAR	BOX
RUG	TAB	TAR	COW
ROW	DOG	SEA	DIG
MOB	RUG	TEA	DOG
BOX	DIG	DIG	EAR
TAB	BIG	BIG	FOX
BAR	BAR	MOB	MOB
EAR	EAR	DOG	NOW
TAR	TAR	COW	ROW
DIG	COW	ROW	RUG
BIG	ROW	NOW	SEA
TEA	NOW	BOX	TAB
NOW	BOX	FOX	TAR
FOX	FOX	RUG	TEA

2(3) Guanxin Ye



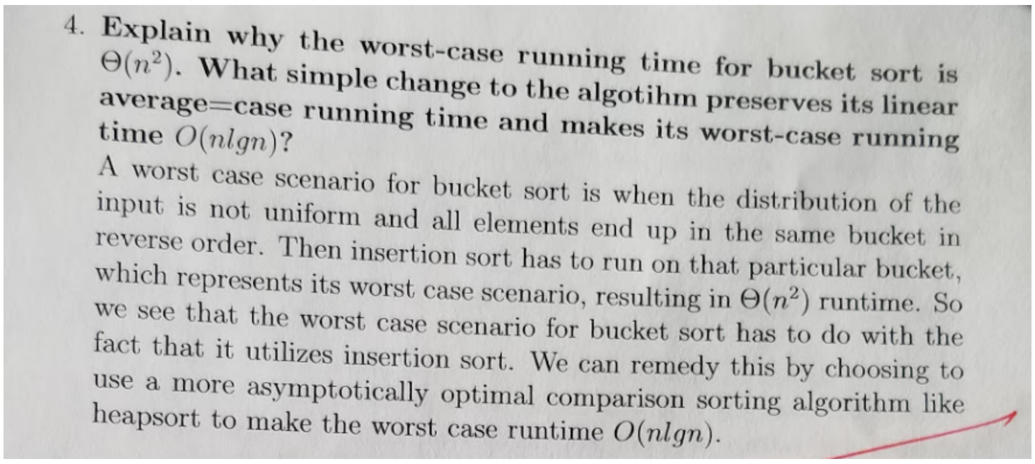
$\text{floor}(6 \times 0.78) = \text{floor}(4.68) = 4$
$\text{floor}(6 \times 0.94) = \text{floor}(5.64) = 5$
$\text{floor}(6 \times 0.21) = \text{floor}(1.26) = 1$ ✓
$\text{floor}(6 \times 0.56) = \text{floor}(3.36) = 3$
$\text{floor}(6 \times 0.98) = \text{floor}(5.88) = 5$
$\text{floor}(6 \times 0.71) = \text{floor}(4.26) = 4$

3. Adrien Fokrum



a) For radix sort in order to keep the running time to O(n) we can convert the base of the array to base n. Run time for radix sort is O(d*(n+k)) and when we convert the base to n it then becomes O(d*(n+n)) with d being the number of digits for the max number which is 4. This then makes the run time O(4*2n) which then becomes O(n).
b) For counting sort it would be O(n+n^4-1) which would be O(n^4)

4. Rafael Megali



4. **Explain why the worst-case running time for bucket sort is $\Theta(n^2)$. What simple change to the algotihm preserves its linear average=case running time and makes its worst-case running time $O(nlgn)$?**
A worst case scenario for bucket sort is when the distribution of the input is not uniform and all elements end up in the same bucket in reverse order. Then insertion sort has to run on that particular bucket, which represents its worst case scenario, resulting in $\Theta(n^2)$ runtime. So we see that the worst case scenario for bucket sort has to do with the fact that it utilizes insertion sort. We can remedy this by choosing to use a more asymptotically optimal comparison sorting algorithm like heapsort to make the worst case runtime $O(nlgn)$.