

Stable algorithms: Insertion sort, merge sort
Not stable algorithms: Heapsort, quicksort.

To make algorithm stable, we should store the element with its index (original order) as a secondary way of sorting elements with equal primary value. This scheme takes $O(n \log n)$ additional space.

2. By Dangnhi Ngo

2/ Counting Sort, Radix Sort, Bucket Sort

(1) COUNTING - SORT: $A = \langle 6, 0, 2, 6, 0, 8 \rangle$

A

6	0	2	6	0	8
1	2	3	4	5	6

C

2	0	1	0	0	0	2	0	1
0	1	2	3	4	5	6	7	8

C

2	2	3	3	3	3	5	5	6
0	1	2	3	4	5	6	7	8

B

					8
1	2	3	4	5	6

↓

	0				8
1	2	3	4	5	6

↓

	0			6	8
1	2	3	4	5	6

↓

	0	2		6	8
1	2	3	4	5	6

↓

0	0	2		6	8
1	2	3	4	5	6

↓

0	0	2	6	6	8
1	2	3	4	5	6

C

2	2	3	3	3	3	5	5	5
0	1	2	3	4	5	6	7	8

1	2	3	3	3	3	5	5	5
---	---	---	---	---	---	---	---	---

1	2	3	3	3	3	4	5	5
---	---	---	---	---	---	---	---	---

1	2	2	3	3	3	4	5	5
---	---	---	---	---	---	---	---	---

0	2	2	3	3	3	4	5	5
---	---	---	---	---	---	---	---	---

0	2	2	3	3	3	3	5	5
---	---	---	---	---	---	---	---	---

0	2	2	3	3	3	3	5	5
---	---	---	---	---	---	---	---	---

0	2	2	3	3	3	3	5	5
---	---	---	---	---	---	---	---	---

0	1	2	3	4	5	6	7	8
---	---	---	---	---	---	---	---	---

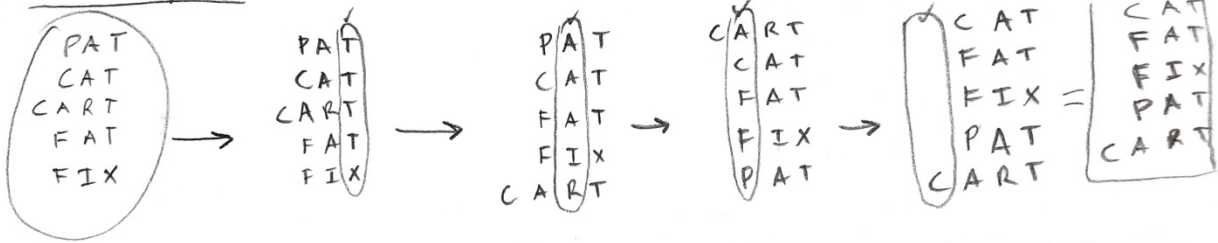
Answer B

0	0	2	6	6	8
---	---	---	---	---	---

By Vlad Gordiyevsky

RADIX-SORT:

Given Input:



By Shraddha Kharche

Q.3) BUCKET-SORT

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

A	B	B
1 0.67	✓	→ 0.12 /
2 0.82	✓	✓
3 0.12	✓	→ 0.46 /
4 0.46	✓	→ 0.61 /
5 0.88	✓	→ 0.67 → 0.82 /
6 0.61	✓	→ 0.88 /

$$\text{Floor}(0.67 \times 6) = \text{Floor}(4.02) = 4$$

$$\text{Floor}(0.82 \times 6) = \text{Floor}(4.92) = 4$$

$$\text{Floor}(0.12 \times 6) = \text{Floor}(0.72) = 0$$

$$\text{Floor}(0.46 \times 6) = \text{Floor}(2.76) = 2$$

$$\text{Floor}(0.88 \times 6) = \text{Floor}(5.28) = 5$$

$$\text{Floor}(0.61 \times 6) = \text{Floor}(3.66) = 3$$

3. By Duyen Tran

(1)

using radix sort we have 3 digits
in base n so we call counting
sort three times

4.

$O(n^3)$ because the range of the input is
 $n^3 - 1$

$$\begin{aligned} O(n + K) \quad K &= \text{range of input} \\ &= O(n + (n^3 - 1)) \\ &= O(n^3) \end{aligned}$$

4. **Sorting** – 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 bucket sort occurs when a single bucket contains all n elements of the original array. After placing the elements into their appropriate bucket, Insertion-Sort is called to sort them in the bucket which has a worst-case running time of $O(n^2)$. The dominating cost of Bucket-Sort is in sorting each bucket so that can be easily fixed by replacing Insertion-Sort with a different sorting algorithm that has a better worst-case running time. For instance, merge sort has a worst-case running time of $O(n \lg n)$ and can be called to sort each bucket to give Bucket-Sort a worst-case running time also of $O(n \lg n)$.