Q4.3) Let A be an array, where each of the n elements is a randomly chosen integer between 1 and n.
For example, if n=12, this array could be A = [3,5,1,10,5,7,9,12,2,8,8,6].
Determine whether Bubble Sort or Bucket Sort sorts this array faster.

Given array A = [3,5,1,10,5,7,9,12,2,8,8,6].

Algorithm and Explanation - Bubble Sort

Bubble Sort is an algorithm that works by repeatedly swapping the adjacent elements if they are in the wrong order, that is, the elements seem to bubble up to their correct positions in the list and thus the name of bubble sort. It is also referred to as comparison or sinking sort.

```
function bubbleSort(A):
    n = len(A)
    for i in range(n):
        swapped = False
        # Last i elements are already in place
        for j in range(0, n-i-1):
            # traverse the array from 0 to n-i-1.
            #Swap if the element found is greater than the next element
            if A[j] > A[j+1] :
                A[j], A[j+1] = A[j+1], A[j]
                swapped = True
        if swapped == False:
            break
```

In the given question, n array elements are randomly chosen integer between 1 and n. This means that the array elements are arranged in any order and the number of swaps of two elements is equal to the number of comparisons in this case, as every or some elements can be out of place. The number of comparisons is constant in Bubble Sort so in average case, there is $O(n^2)$ comparisons. This is because irrespective of the arrangement of elements, the number of comparisons C(n) is same.
For the number of swaps, consider the following points:

- If an element is in index I_1 but it should be in index I_2, then it will take a minimum of I_2-I_1 swaps to bring the element to the correct position.
- An element E will be at a distance of I_3 from its position in sorted array. Maximum value of I_3 will be (n-1) for the edge elements and it will be n/2 for the elements at the middle.
- The sum of maximum difference in position across all elements will be (n-1) + (n-3) + (n-5) ... + 0 + ... + (n-3) + (n-1)
  = n x n - 2 x (1 + 3 + 5 + ... + n/2)
  = n^2 - 2 x n^2 / 4
  = n^2 - n^2 / 2
  = n^2 / 2

Therefore, in average, the number of swaps = $O(n^2)$.
The recurrence relation will be T(n) = T(n-1) + $\Theta(n)$ will result in time complexity $\Theta(n^2)$.

So, for the given question, the running time of Bubble Sort is $\Theta(n^2)$.

Algorithm and Explanation - Bucket Sort

Bucket sort is a comparison sort algorithm that works by distributing the elements of an array into a number of buckets and then each bucket is sorted individually using a separate sorting algorithm or by applying the bucket sort algorithm recursively. This algorithm is mainly useful when the input is uniformly distributed over a range.
BucketSort algorithm follows the following steps,

- Create an empty array of size n(n empty buckets).
- Loop through the original array and put each array element in a “bucket”.
- Sort each of the non-empty buckets using insertion sort.
- Visit the buckets in order and put all elements back into the original array

function bucketSort(A[]):
step 1: create as many empty buckets (lists or arrays) as the length of the input array
step 2: store the array elements into the buckets based on their values
step 3: for i from 0 to length of A do

(a) bucket_index = int((n * A[i]) / 10) //here n is the length of the input array

(b) store the element A[i] at bucket[bucket_index]

step 4: for i from 0 to length of bucket do:

(a) sort the elements stored inside bucket[i], either by applying recursion or by some other sorting method

step 5: push the values stored in each bucket starting from lowest index to highest index to a new array/list

step 6: return the sorted array/list

In our bucket sort algorithm, we have four loops (step 1, step 3, step 4 and step 5) which iterate over k buckets and n elements.
In the given question, n array elements are randomly chosen integer between 1 and n. This means that the array elements are arranged in any order. The average case occurs when the elements are distributed randomly in the list. Bucket sort runs in linear time in all cases until the sum of the squares of the bucket sizes is linear in the total number of elements. It takes O(n) time in order for us to assign and store all the elements of the input array into the bucket list. As our final step, we have to concatenate our elements back together in the form of an array to produce our final sorted array. Since there are k buckets, this step will cost us a total of O(k) time. It happens when the array's elements are distributed at random as given in the question. Bucket sorting takes linear time, even if the elements are not distributed uniformly. Also, if insertion sort is used to sort bucket elements, the overall complexity will be linear, that is, O(n+k).

Hence, we come to the conclusion that the average case time complexity for this algorithm will be O(n+k) where k is the number of buckets when the elements are distributed randomly in the array.

Hence, for the given question, where A consists of n array elements are randomly chosen integer between 1 and n, Bucket Sort works better and sorts this array faster in $\Theta(n+k)$ time as compared to Bubble Sort which takes $\Theta(n^2)$ time.