Lesson:

BUCKET SORT

PW SKILLS

Pre-Requisites

- Cpp loops
- Cpp array

List of Concepts Involved

- Bucket Sort

What is bucket sort?

Bucket Sort as the name suggests is a sorting algorithm in which we divide our array elements into various buckets where each bucket is a group of numbers and then sort these buckets to obtain the final sorted array.

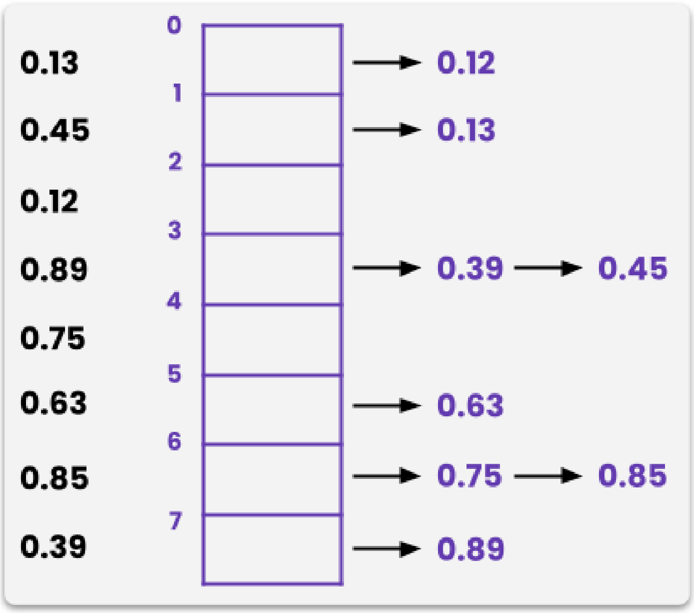
- It is a sorting algorithm which is used to sort an array of numbers which is uniformly distributed over a range.
- It is mostly used with floating point values.

We cannot use counting sort in place of bucket sort as in counting sort the keys that we use are integer indexes but in an array containing floating point numbers the keys will be floating point numbers.

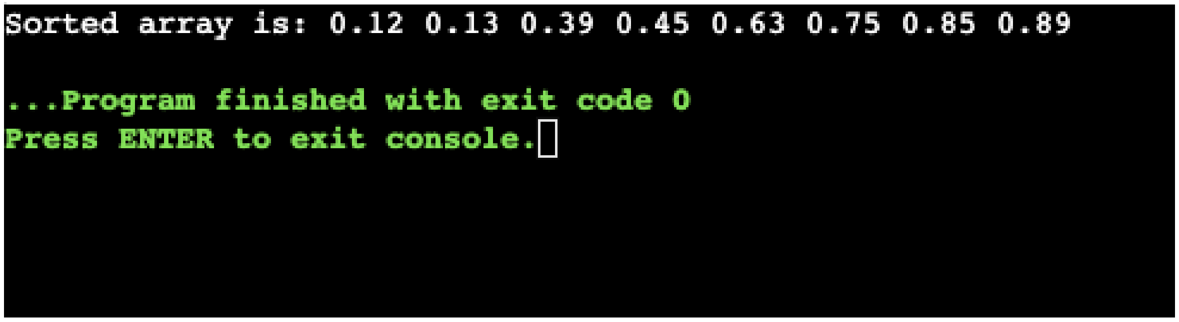
ALGORITHM:

```
1. We create n empty bucket.
2. We will traverse the array and for each array element a[i] we will
    a. Insert a[i] in the bucket[n*a[i]]
3. We will then sort all the buckets individually.
4. Combine all the sorted buckets
```

Let us try to understand this with the help of an example:
Consider the following array a[] : 0.13,0.45,0.12,0.89,0.75,0.63,0.85,0.39



Thus, the code of bucket sort will be: **https://pastebin.com/LrxY66ES**



```
Sorted array is: 0.12 0.13 0.39 0.45 0.63 0.75 0.85 0.89

...Program finished with exit code 0
Press ENTER to exit console.
```

Explanation : The code contains 4 major steps which are used to perform bucket sort, we first create an empty n sized bucket to store the elements. In the next step we will add the element arr[i] in the bucket at index size*arr[i] i.e. we will add arr[i] into bucket[size*arr[i]]. We will then sort each bucket individually and then combine the sorted buckets together to form our sorted array.

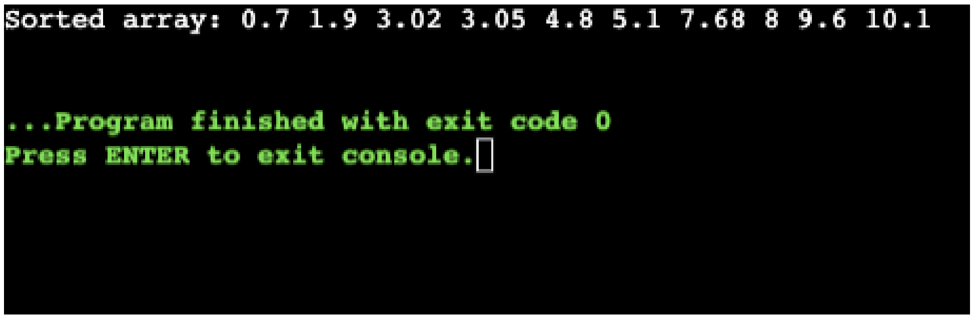
Time complexity: The first and the second step clearly takes O(n) time as we are inserting the elements in the bucket and the same goes with step 4 in which we are traversing each element of each bucket which will again take O(n) time. Now talking about step 3, if the elements are uniformly distributed then sorting will take an average time of O(n) thus the time complexity will be O(n).

Now this method to divide elements in bucket will work only if size*arr[i] is less than size i.e. arr[i] is less than 1, the question arises what should we do if arr[i] is greater than 1? How should we divide elements in a bucket? What will be the range of the bucket ?

Algorithm:

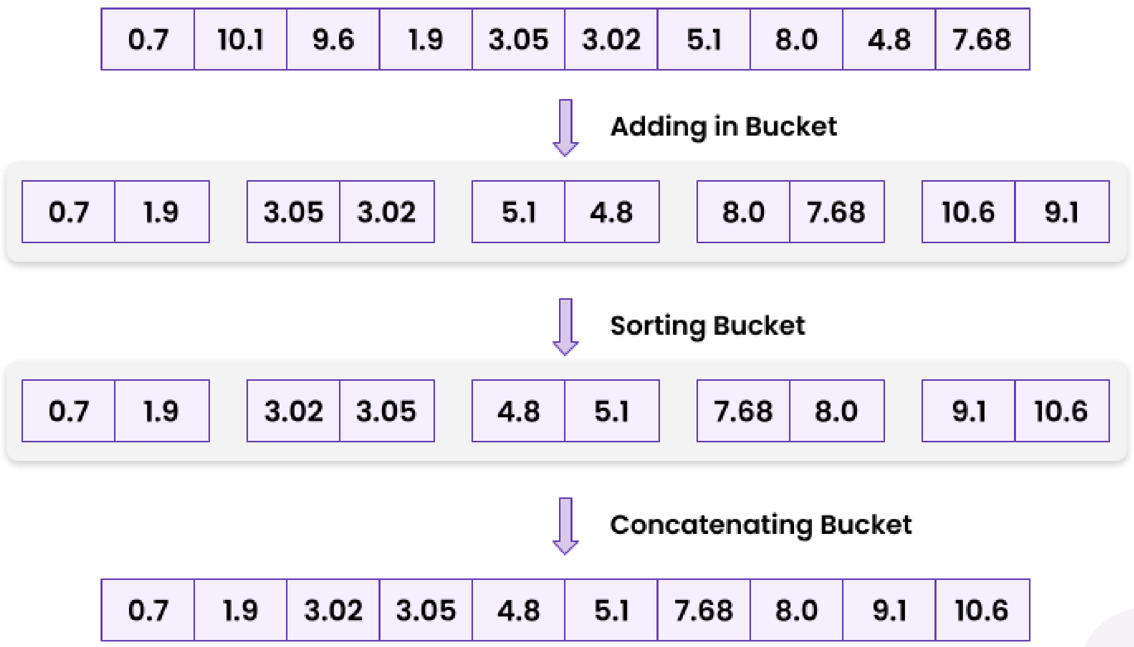
1. We will first find the range of each bucket and for that we will find the minimum and maximum element of the array and thus the range will be equal to range=(max_element - min_element) / n where n is the size of array.
2. We will then create n buckets of this range where n is the size of the array.
3. We will now find the bucket index for each element of the array and that will be equal to bi= (arr[i] - min_element) / range
4. Sort each bucket individually
5. Concatenate the buckets.

Code link: https://pastebin.com/MxRpxhhR



```
Sorted array: 0.7 1.9 3.02 3.05 4.8 5.1 7.68 8 9.6 10.1


...Program finished with exit code 0
Press ENTER to exit console.
```



Explanation: We first calculate the range of each bucket by finding the min and max element of the array using inbuilt functions, after finding the range we then create n empty buckets. We then start with filling these buckets by finding the bucket index for each arr[i] using the formula index=(arr[i]-min_elmt)/range
Here we take care of the fact that if the index turns out to be greater than the size of the bucket array then we subtract 1 from it to make it the index in range. We then sort each of these buckets individually and concatenate them.

Time complexity: Since we are just traversing each of the buckets and the array to fill the buckets, thus the time complexity will be O(n+k) where k is the number of buckets.

Upcoming Class Teasers

- Problems based on sorting algorithms.