**Day 33**

**Bucket Sort**

**Bucket sort** is a sorting technique in which elements are first uniformly divided into several groups called **buckets**. After that, elements are sorted by any sorting algorithm, and finally, it gathered the elements in a sorted manner. In this section, we will learn **how bucket sort works, its algorithm, complexity, example,** and **its implementation in a Java program**.

Bucket Sort

**Bucket sort** or **bin sort** is a sorting algorithm that works by distributing the elements into a number of buckets, homogenously. Each bucket is then sorted individually. In order to sort the bucket, we use the sort() method of the Arrays class. It is usually used to [sort floating-point numbers](https://www.javatpoint.com/daa-bucket-sort).

The basic idea to perform the bucket sort is:

* Partition the range into a fixed number of buckets.
* Toss each element into its appropriate bucket.
* Sort each bucket individually using insertion sort.
* Concatenate all the sorted buckets.

Pros

* It is asymptotically fast because of uniform distribution.
* It reduces the number of comparisons.
* It is fast in comparison to bubble sort.

Cons

* It is not an **in-place sorting** because we need some extra space to sort the buckets.
* It may or may not be the **stable** sorting algorithm.
* It is not useful if we have large array because it increases the cost.

Let's see the algorithm.

Algorithm

Bucket Sort(A[])

1. Let B[0….n-1] be a new array
2. n=length[A]
3. for i=0 to n-1
4. make B[i] an empty list
5. for i=1 to n
6. do insert A[i] into list B[n a[i]]
7. for i=0 to n-1
8. do sort list B[i] with insertion-sort
9. Concatenate lists B[0], B[1],..……, B[n-1] together in order

Complexity

|  |  |
| --- | --- |
| **Time Complexity** | |
| Worst Case | O(n2) |
| Best Case | O(n+k) |
| Average Case | O(n+k) |

Bucket Sort Example

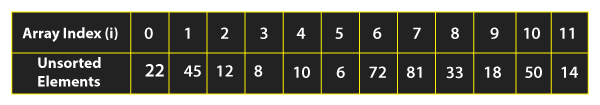
**Sort the following array in ascending order by using the bucket sort.**

**arr[]=22, 45, 12, 8, 10, 6, 72, 81, 33, 18, 50, 14**

Total number of elements in the given array (N) = 12

Max element in array = 81

Min element in array = 6



We need **10** buckets to sort the array. Suppose, these 10 buckets are represented as **B.** After that, we need to find a divider that will be used to put the elements in the bucket. In order to determine the divider, we use the following formula:

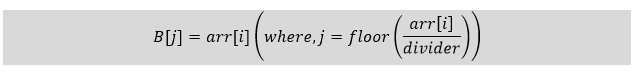
Bucket Sort in Java

Let's put the values in the above formula, we get:

Bucket Sort in Java

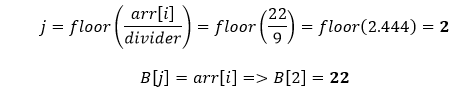
Hence, bucket = 10, divider = 9

Let's put the element arr[i] in the correct bucket, we will use the following formula:



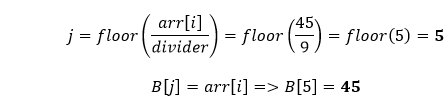
Let's see how it works by putting elements in the buckets. We will start from the first index.

**For i=0:**



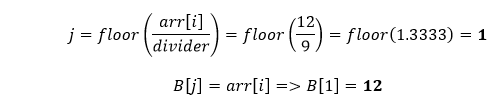
We will insert the **zeroth** element **(22)** in the **2nd** bucket and increment the array index (i) by 1.

**For i=1:**



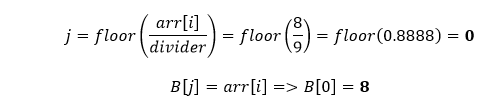
We will insert the **first** element **(45)** in the **5th** bucket and increment the array index (i) by 1.

**For i=2:**



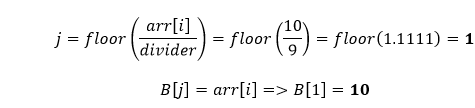
We will insert the **second** element **(12)** in the **1st** bucket and increment the array index (i) by 1.

**For i=3:**



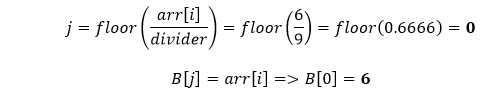
We will insert the **third** element **(8)** in the **0th** bucket and increment the array index (i) by 1.

**For i=4:**



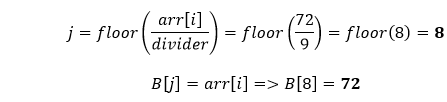
We will insert the **fourth** element **(10)** in the **1st** bucket and increment the array index (i) by 1.

**For i=5:**



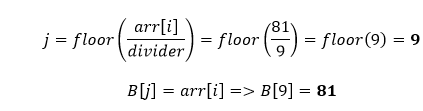
We will insert the **fifth** element **(6)** in the **0th** bucket and increment the array index (i) by 1.

**For i=6:**



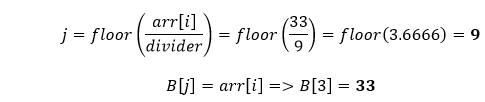
We will insert the **sixth** element **(72)** in the **8th** bucket and increment the array index (i) by 1.

**For i=7:**



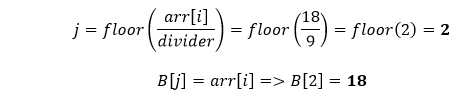
We will insert the **seventh** elements **(81)** in the **8th** bucket and increment the array index (i) by 1.

**For i=8:**



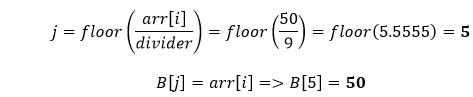
We will insert the **eighth** element **(33)** in the **3rd** bucket and increment the array index (i) by 1.

**For i=9:**



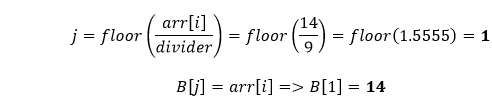
We will insert the **ninth** elements **(18)** in the **2nd** bucket and increment the array index (i) by 1.

**For i=10:**

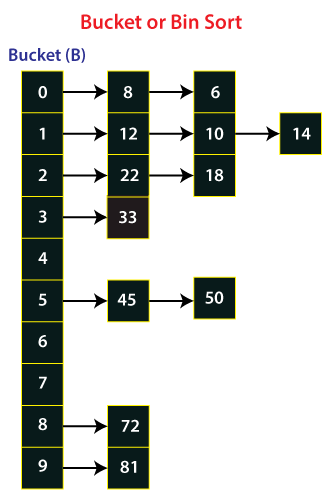


We will insert the **tenth** elements **(50)** in the **5th** bucket and increment the array index (i) by 1.

**For i=11:**

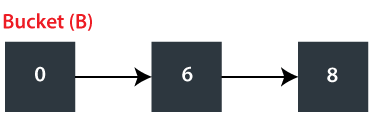


We will insert the **eleventh** elements **(14)** in the **1st** bucket and increment the array index (i) by 1.



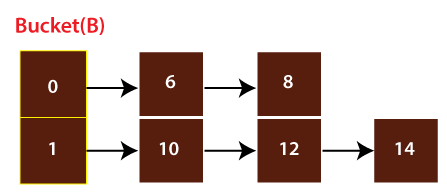
Now, will perform **insertion** sort on the individual buckets to sort the elements. Let's start from the first bucket (**0th**).

Is ? Yes, swap their positions.



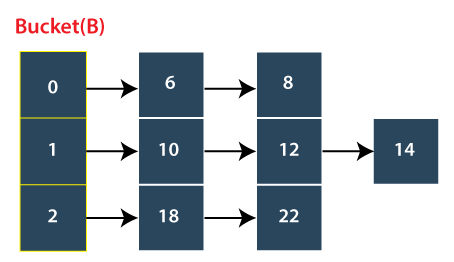
Now, move to the next bucket (**1st**) and compare each element to the other.

Is ? Yes, swap their positions and compare the next pair. Is ? No, elements are already in a sorted manner, so we will not swap their positions.



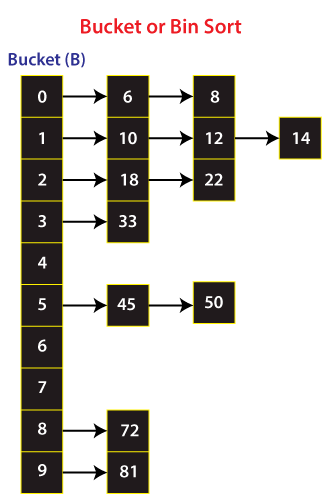
Now, move to the next bucket (**2nd**) and compare their elements.

Is ? Yes, swap their positions.



Now, we will move to the next bucket. Here, a point to note that the bucket that has only one element is already sorted and the bucket that has no element, we will skip them. Therefore, we will move to the **fifth** bucket and compare their elements.

Is ? No, elements are already sorted. Similarly, trace the buckets until we reach the last bucket. So, we will stop here as we have got a sorted array.



At last, we will take out all the elements from each bucket. Therefore, we get a sorted array.

