



Dash



All



Articles



Videos



Problems

Courses

Tutorials

Jobs

Practice

Contests



Bucket Sort

Bucket sort is mainly useful when input is uniformly distributed over a range. For example, consider the following problem.

Sort a large set of floating point numbers which are in range from 0.0 to 1.0 and are uniformly distributed across the range. How do we sort the numbers efficiently?

A simple way is to apply a comparison based sorting algorithm. The lower bound for Comparison based sorting algorithm (Merge Sort, Heap Sort, Quick-Sort .. etc) is $\Omega(n \log n)$, i.e., they cannot do better than $n \log n$.

Can we sort the array in linear time? Counting sort can not be applied here as we use keys as index in counting sort. Here keys are floating point numbers.

The idea is to use bucket sort. Following is bucket algorithm.

```
bucketSort(arr[], n)
```

```
.....a) insert arr[i] into bucket[n*array[i]]
3) Sort individual buckets using insertion sort.
4) Concatenate all sorted buckets.
```

<< Prev

Next >>

>

Dash

All

Articles

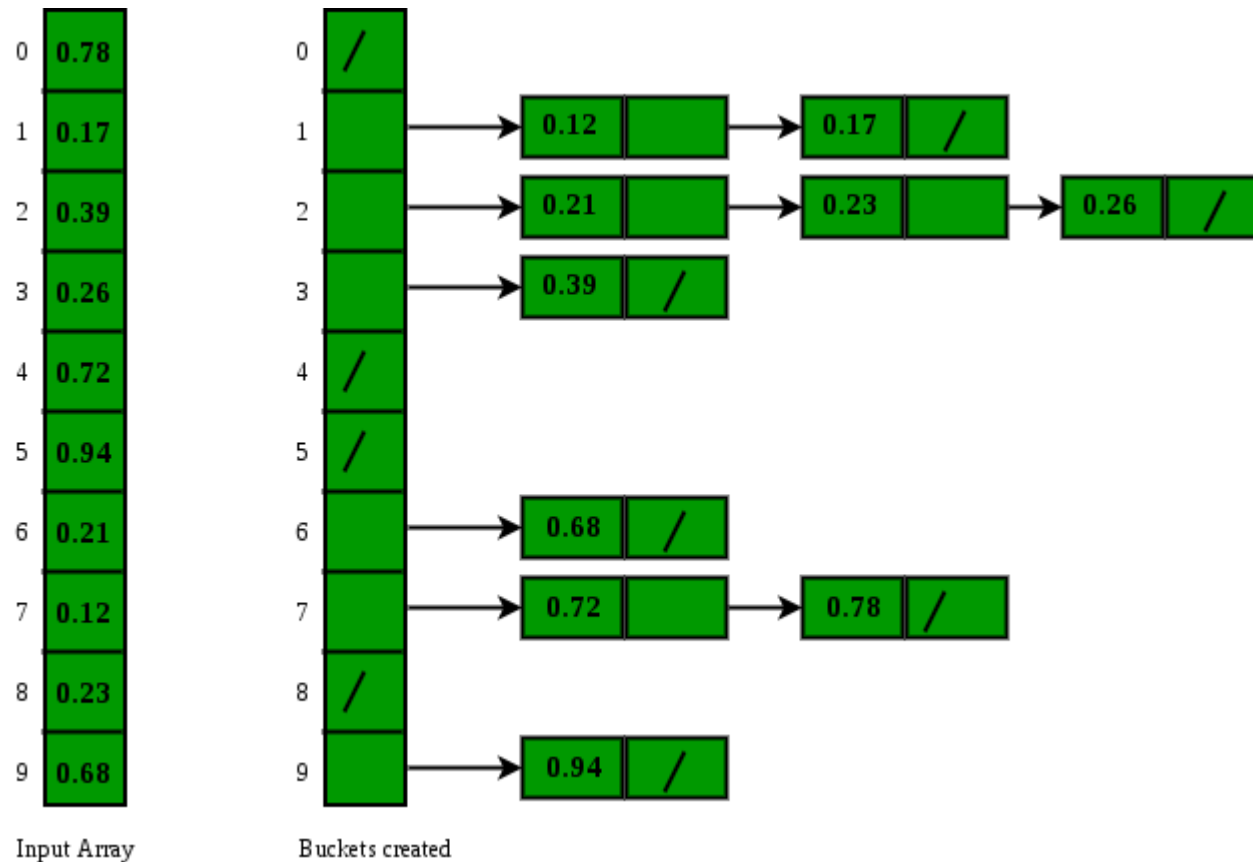
Videos

Problems

Quiz

<<

>>

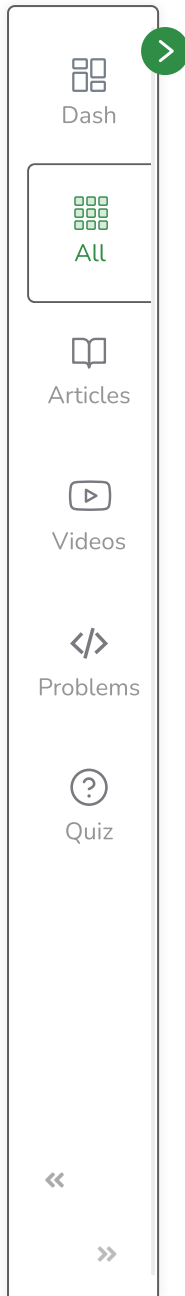


Time Complexity: If we assume that insertion in a bucket takes $O(1)$ time then steps 1 and 2 of the above algorithm clearly take $O(n)$ time. The $O(1)$ is easily possible if we use a linked list to represent a bucket (In the following code, C++ vector is used for simplicity). Step 4 also takes $O(n)$ time as there will be n items in all buckets. The main step to analyze is step 3. This step also takes $O(n)$ time on average if all numbers are uniformly distributed (please refer [CLRS book](#) for more details) Following is the implementation of the above algorithm.

C++

Java

```
// C++ program to sort an
// array using bucket sort
```



```
#include <algorithm>
#include <iostream>
#include <vector>
using namespace std;

// Function to sort arr[] of
// size n using bucket sort
void bucketSort(float arr[], int n)
{
    // 1) Create n empty buckets
    vector<float> b[n];

    // 2) Put array elements
    // in different buckets
    for (int i = 0; i < n; i++) {
        int bi = n * arr[i]; // Index in bucket
        b[bi].push_back(arr[i]);
    }

    // 3) Sort individual buckets
    for (int i = 0; i < n; i++)
        sort(b[i].begin(), b[i].end());

    // 4) Concatenate all buckets into arr[]
    int index = 0;
    for (int i = 0; i < n; i++)
        for (int j = 0; j < b[i].size(); j++)
            arr[index++] = b[i][j];
}
```





Dash



All



Articles



Videos



Problems



Quiz



}

/* Driver program to test above function */

int main()

{

float arr[]

= { 0.897, 0.565, 0.656, 0.1234, 0.665, 0.3434 };

int n = sizeof(arr) / sizeof(arr[0]);

bucketSort(arr, n);

cout << "Sorted array is \n";

for (int i = 0; i < n; i++)

cout << arr[i] << " ";

return 0;

}



Output

Sorted array is

0.1234 0.3434 0.565 0.656 0.665 0.897

[Mark as Read](#) [Report An Issue](#)

