Bucket sort algorithm

Source: ***https://github.com/rusevrosen/codehub.github.io/tree/main/Algorithms/Sorting_Algorithms***

1. *What is the Bucket Sort algorithm?*

Bucket sort is an efficient sorting algorithm that works by distributing elements of an array into a number of buckets. Each bucket is then sorted individually, either using a different sorting algorithm or by recursively applying the bucket sort algorithm. Once all the buckets have been sorted, the elements are recombined back into the original array in the order of their buckets. This algorithm has a time complexity of **O(n)** when the elements are uniformly distributed. C++ implementation of the Bucket Sort algorithm would involve creating an array of buckets, iterating through the input array and placing each element into its appropriate bucket, sorting the elements within each bucket, and finally iterating through the buckets and recombining the elements back into the original array.

2. *Code example.*

```cpp
#include <iostream>
#include <algorithm>
#include <vector>

void bucketSort(float arr[], int n) {
    // Create n empty buckets
    std::vector<float> b[n];

    // 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]);
    }

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

    // 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];
}

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);

    std::cout << "Sorted array is \n";
    for (int i=0; i<n; i++)
       std::cout << arr[i] << " ";

    return 0;
}
```

This program will output the sorted array:

```
0.1234 0.3434 0.565 0.656 0.665 0.897
```

3. Code explanation.

The example above demonstrates how to implement the Bucket Sort algorithm in C++. The input array is an array of floating-point numbers called "**arr**". The function "**bucketSort**" takes two parameters, the array to be sorted and the number of elements in the array.

A short explanation of what the code does:

1. Create an array of **n** empty vectors (buckets) called "**b**". These buckets will be used to store the elements of the input array.
2. Iterate through the input array and place each element into its corresponding bucket based on its value. The index of the bucket is calculated by multiplying the element's value by the number of elements in the array.
3. Sort the elements within each bucket using the STL sort function.
4. Concatenate all the elements in all the buckets back into the original array in the order of the buckets.
5. The main function uses the **bucketSort** function to sort the input array and print the sorted array.

The output of the program is the sorted array in ascending order, this is achieved by using the STL sort function which is a comparison sort algorithm. It's worth noting that this implementation has a time complexity of **O(n log n)** which is slower than the **O(n)** time complexity that can be achieved by using a linear-time sorting algorithm such as radix sort.

4. Time complexity.

The time complexity of the Bucket Sort algorithm implemented in the example above is **O(n log n)** where **n** is the number of elements in the input array.

The reason for this is that the example uses the STL sort function which is a comparison sort algorithm and has a time complexity of **O(n log n)** to sort the elements within each bucket. Since there are n buckets and each bucket is sorted in **O(n log n)** time, the overall time complexity of the algorithm is **O(n log n).**

This implementation using the STL sort function is slower than the **O(n)** time complexity that can be achieved by using a linear-time sorting algorithm such as radix sort.

It's worth noting that bucket sorting is efficient when the input is uniformly distributed ☺

I wish you happy learning, Your Rosen Rusev