Bucket Sort (Implementation)

(Reading time: 4 minutes)

We create a function that receives the array that we want to sort as an argument. We can immediately declare 3 variables: n, which is the length of the buckets (which is the length of the array), allBuckets, which is a new array of all the buckets, and the new array sortedArray, to which we will push the elements to in ascending order. If the length of the array is smaller than 2, we don’t need to sort it at all. We can then return the array right away.

```
function bucketSort(array) {
  const n = array.length;
  const allBuckets = new Array(n);
  const sortedArray = [];

  if (n < 2) return array;
}
```

For every bucket, we initialize an empty array. We will later push the elements that belong to that bucket to its corresponding array.

```
for (let i = 0; i < n; i++) {
   allBuckets[i] = [];
}
```

In order to push elements to these bucket arrays, we loop over the array we want to sort. For every element, we calculate to which bucket they should be pushed. In this example, I use `Math.floor(n * array[i] / 10)` in order to calculate this, however this function can be different.

```
for (let i = 0; i < n; i++) {
   const index = Math.floor(n * array[i] / 10);
   allBuckets[index].push(array[i]);
};
```

Now, we have all the **buckets**. If we logged the buckets now, it would look like:

```
[
 [],
 [ 1.8, 2.3, 2.2 ],
 [],
 [ 5.2, 4.8 ],
 [ 5.9, 6.5 ],
 [],
 []
]
```

Now, we want to sort every individual array using insertion sort. We loop over the buckets array, and invoke the `insertionSort` function on every array. Then, once the bucket array has been sorted, we push it to the `sortedArray` array! We do this for all the arrays in the `allBuckets` array.

```
allBuckets.forEach(bucket => {
   insertionSort(bucket);
   bucket.forEach(element => sortedArray.push(element))
});
```

The entire function would look like :

```
function bucketSort(array) {
  const n = array.length;
  const allBuckets = new Array(n);
  const sortedArray = [];

  if (n < 2) return array;

  for (let i = 0; i < n; i++) {
    allBuckets[i] = [];
  }

  for (let i = 0; i < n; i++) {
    const index = Math.floor(n * array[i] / 10);
    allBuckets[index].push(array[i]);
  };

  allBuckets.forEach(bucket => {
    insertionSort(bucket);
    bucket.forEach(element => sortedArray.push(element))
  });

  return sortedArray;
}
```

In the next lesson, I will talk about the time complexity of this algorithm.