ADS 7

Abhilekh Pandey

March 2020

1 Problem 7.1

1.1 (c)

A is the array to check with size n

C is the count array with size k

First element of an array is index 0

For a range [a, b], it is assumed that b ¿ a

```
For i in A.Length
C [ A [ i ] ] += 1
```

```
For i from 1 to C.length
C [ i ] = C [ i ] + C [ i - 1 ]
```

```
return C [ b ] - C [ a ]
```

This returns the number of elements between a and b

Because it is a modified version of count sort, complexity is $\Theta(n + k)$

1.2 (e)

Worst case of a bucket sort is when every element falls in the same bucket. Then, the complexity in the bucketsort algorithm mainly comes from the one sorting individual buckets.

Example:

Bucketsort on [0.1, 0.11, 0.111, 0.1111, 0.11111]

Here, all the elements fall under the same bucket. Thus the only sorting occouring is the internal sorting.

2 Problem 7.2

2.1 (b)

My implementation of Radix sort on an array of size n is like the example provided by the professor.

I preform a count sort on the entire array using the individual digits at the ones, tenth ... positions in base 10.

Thus the time complexity of the count sort is $\Theta(n)$

The Radix sort calls count sort $\lfloor log10(Max\_Array\_Element) \rfloor$ number of times.

The time complexity of the Radix sort is then $\Theta(n \cdot d$.

Where $d = \lfloor log10(Max\_Array\_Element) \rfloor$