

Objective: Implementation of Radix Sort

Radix Sort

Radix sort is a sorting technique that sorts the elements by first grouping the individual digits of the same place value. Then, sort the elements according to their increasing/decreasing order.

Radix Sort Algorithm:

```
radixSort(array)

    d <- maximum number of digits in the largest element

    create d buckets of size 0-9

    for i <- 0 to d
        sort the elements according to ith place digits using countingSort
    countingSort(array, d)

    max <- find largest element among dth place elements

    initialize count array with all zeros

    for j <- 0 to size
        find the total count of each unique digit in dth place of elements and
        store the count at jth index in count array

    for i <- 1 to max
        find the cumulative sum and store it in count array itself

    for j <- size down to 1
        restore the elements to array

    decrease count of each element restored by 1
```

How Radix Sort Works?

1. Find the largest element in the array, i.e. `max`. Let `X` be the number of digits in `max`. `X` is calculated because we have to go through all the significant places of all elements.
2. Now, go through each significant place one by one. Use any stable sorting technique to sort the digits at each significant place. We have used counting sort for this. Sort the elements based on the unit place digits (`X=0`).
3. Now, sort the elements based on digits at tens place.
4. Finally, sort the elements based on the digits at hundreds place.

Code:

```
def RadixSort(nums):  
    r = 10  
    place = 1  
    maximum = max(nums)  
    while place < maximum:  
        buckets = [list() for _ in range( r )]  
        for i in nums:  
            tmp = int((i / place) % r)  
            buckets[tmp].append(i)  
        a = 0  
        for b in range( r ):  
            bucket = buckets[b]  
            for i in bucket:  
                nums[a] = i
```

```
        a += 1

    place *= r

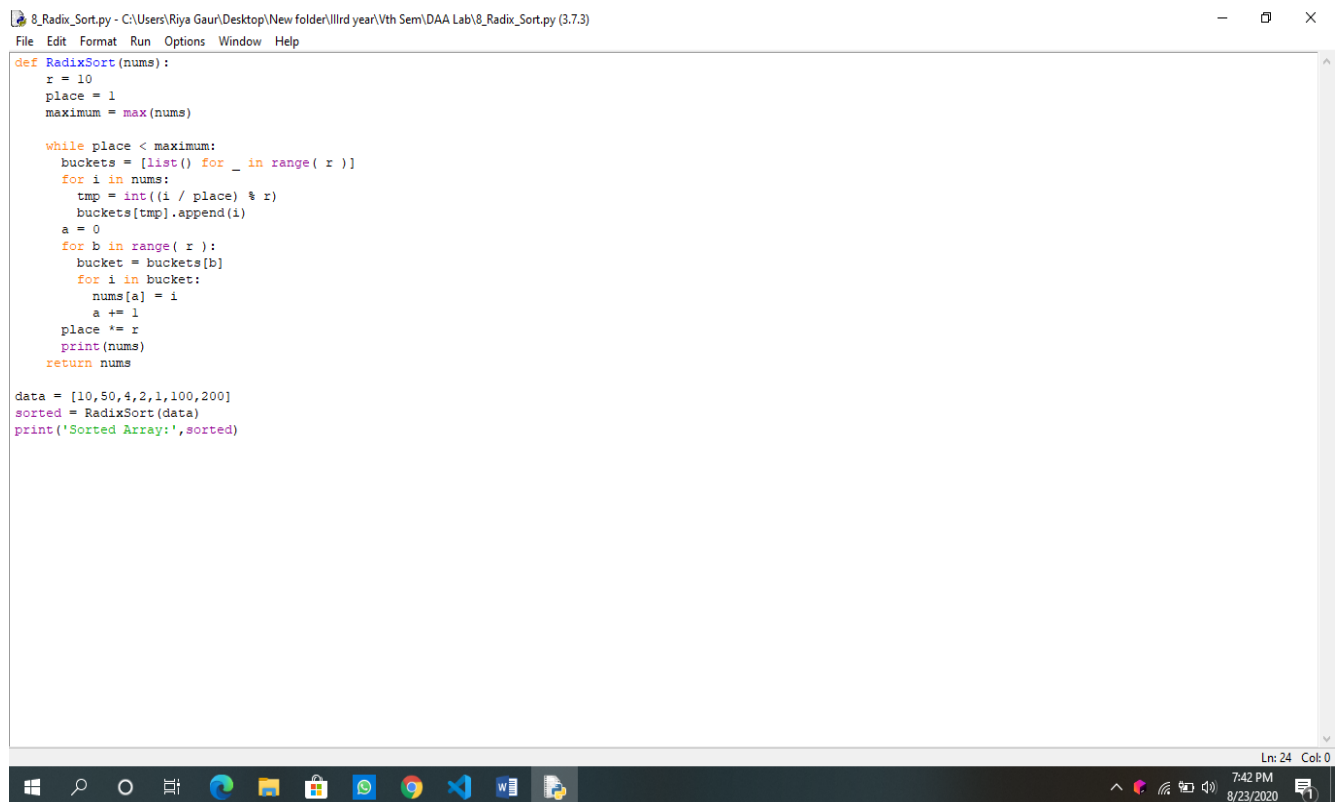
    print(nums)

    return nums

data = [10,50,4,2,1,100,200]

sorted = RadixSort(data)

print('Sorted Array:',sorted)
```



The screenshot shows a Python script in a text editor window titled "8_Radix_Sort.py - C:\Users\Triya Gaur\Desktop\New folder\Iird year\Vth Sem\DAA Lab\8_Radix_Sort.py (3.7.3)". The script defines a function `RadixSort(nums)` that sorts a list of numbers using radix sort. The function uses a base `r = 10` and iterates through the digits of the numbers, placing them into buckets and then recombining them. The script also includes a test case where the function is applied to the list `[10, 50, 4, 2, 1, 100, 200]`, and the sorted result is printed. The output of the script is shown in the console window at the bottom of the editor.

```
def RadixSort(nums):
    r = 10
    place = 1
    maximum = max(nums)

    while place < maximum:
        buckets = [list() for _ in range( r )]
        for i in nums:
            tmp = int((i / place) % r)
            buckets[tmp].append(i)
        a = 0
        for b in range( r ):
            bucket = buckets[b]
            for i in bucket:
                nums[a] = i
                a += 1
        place *= r
        print(nums)
    return nums

data = [10,50,4,2,1,100,200]
sorted = RadixSort(data)
print('Sorted Array:',sorted)
```

Output:

```
[10, 50, 100, 200, 1, 2, 4]

[100, 200, 1, 2, 4, 10, 50]

[1, 2, 4, 10, 50, 100, 200]

Sorted Array: [1, 2, 4, 10, 50, 100, 200]
```

```
Python 3.7.3 Shell
File Edit Shell Debug Options Window Help
Python 3.7.3 (v3.7.3:ef4ec6ed12, Mar 25 2019, 21:26:53) [MSC v.1916 32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
RESTART: C:\Users\Triya Gaur\Desktop\New folder\IIIrd year\Vth Sem\DAA Lab\8_Radix_Sort.py
[10, 50, 100, 200, 1, 2, 4]
[100, 200, 1, 2, 4, 10, 50]
[1, 2, 4, 10, 50, 100, 200]
Sorted Array: [1, 2, 4, 10, 50, 100, 200]
>>> |
```

Time Complexities:

Radix sort complexity is $O(kn)$ for n keys which are integers of word size k .

For all there cases time i.e best , worst and average time complexity is $O(kn)$.

Radix Sort Applications:

Radix sort is implemented in

- DC3 algorithm (Kärkkäinen-Sanders-Burkhardt) while making a suffix array.
- places where there are numbers in large ranges.