* Important Sorting Algos for ECE/EIE/EEE/Robotics

* NCA → Non-Comparison Algorithms

* Count Sort Algorithm

* Single digit whole numbers (0 - 9)

c = c + p

n = 9

① Find the max :→ max = 6

② Create a count array 0 – max, 0 – 6

③ Calculate the freq. of each element in the array

④ Cumulative count → L→R

⑤ Create O/P array R→L

Input → | 1 | 3 | 2 | 4 | 1 | 4 | 6 | 3 | arr[i]

Count → | 0 | 2 | 1 | 3 | 2 | 0 | 1 |
0 1 2 3 4 5 6

Cumulative Count → | 0 | 2 | 3 | 6 | 8 | 8 | 9 | → size
0 1 2 3 4 5 6

output | 1 | 1 | 2 | 3 | 3 | 3 | 4 | 4 | 6 |
0 1 2 3 4 5 6 7 8

count [arr[i]]

arr i/p → | 4 | 0 | 4 | 2 | 1 | count [arr[i]] ++

count → | | ++ | | ++ | ++ |
0 1 2 3 4

| 0 | 1 2 | 1 | 1 | 1 |
0 1 2 3 4

```
for (int i = 0; i < size ; i++)  {
    count [arr[i]] ++ ;
}
```

i = 0, i = 1, i = 2, i = 4

Radix Sort :→ (Non-Comparison Algorithm)

Places → 1's, 10's, 100's and so on

* Multiple Digits

* Constant Length Strings

Rahul Samar

1. Find max → 325

* How many digits? 325 → 3 digits ③ passes

2. 0 – 9 10 Buckets

O(n * max)

325	224	032	008	091	
091	032	224	325	008	P1
008	224	325	032	091	P2
008	032	091	224	325	P3

| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |

008, 032, 091 × 224 325

P1, P2, P3 → countSort

How are the no of passes or iterations or rounds in Radix Sort controlled depending on the max value? max = 325? 3 passes?

1, 2, 3 ✗

L&T, TCS, CG, Bosch

```
for (int exp = 1; max/exp > 0 ; exp *= 10)
{
    countSort (arr, size, exp);
}
```

max = 325
exp = 1, 10, 100, 1000
325 / 1 = 325
325 / 10 = 32
325 / 100 = 3
325 / 1000 = 0

Linear Data Structures :→

Stack → LIFO → FILO

↓ Reversal

① → ② → ③ → ④ → null

4 3 2 1

1 — 2 — 1
1 2 1

| 4 |
| 3 |
| 2 |
| 1 |

1 2 1

Class

top = -1

top++ 5 ↓top

array 0 99

100

MAX-SIZE = 99

top--

top == -1

push ( ), pop( )

top == MS - 1

size( ), empty( )

peek( )

top

Stacks / Queues / Linked Lists :→

Trees → STL → Standard Template Library

#include <stack>

Graphs

queue → FIFO

addition

front rear

removal

front++ rear++

empty f = r = -1

f = r

0

f = r = -1

f++

empty → f = r = 0

f → f++

arr[rear] = x

f = r = -1

r++

f r

display front

f = -1
r = -1

r++ = 0

arr[front] =