Data Structures and Algorithms
(ES221)

Counting and Radix Sort

Dr. Zubair Ahmad

Counting Sort

- Suppose that there are N integers to be sorted, A[1⋯N], such that the range of the integers is from 1 to M
- Start by defining an array B[1⋯M] and initializing all its elements to 0. → (O(M))
- Scan through A[i], for i = 1,⋯,N, and copy each A[i] into B[A[i]]. → (ON))
- Scan through B[i], for i = 1,⋯,M , and retrieve all the non-zero elements of B[i]. → (O(M))
- Complexity: O(M + N)
  - If M ~ N → Complexity: O(N)
  - If M ~ N^2 → Complexity: O(N^2)
- Example
  - Sort 5 7 9 2 4 6 1

Counting Sort

A

5 (0)	7 (1)	9 (2)	2 (3)	4 (4)	6 (5)	1 (6)

B[A[i]] = A[i]

B

(0)	(1)	(2)	(3)	(4)	5 (5)	(6)	(7)	(8)	(9)

B[A[0]] = A[0]
B[5] = 5

(0)	(1)	(2)	(3)	(4)	5 (5)	(6)	7 (7)	(8)	(9)

B[7] = 7

(0)	(1)	(2)	(3)	(4)	5 (5)	(6)	7 (7)	(8)	9 (9)

B[9] = 9

(0)	(1)	2 (2)	(3)	(4)	5 (5)	(6)	7 (7)	(8)	9 (9)

B[2] = 2

(0)	(1)	2 (2)	(3)	4 (4)	5 (5)	(6)	7 (7)	(8)	9 (9)

B[4] = 4

(0)	(1)	2 (2)	(3)	4 (4)	5 (5)	6 (6)	7 (7)	(8)	9 (9)

B[6] = 6

(0)	1 (1)	2 (2)	(3)	4 (4)	5 (5)	6 (6)	7 (7)	(8)	9 (9)

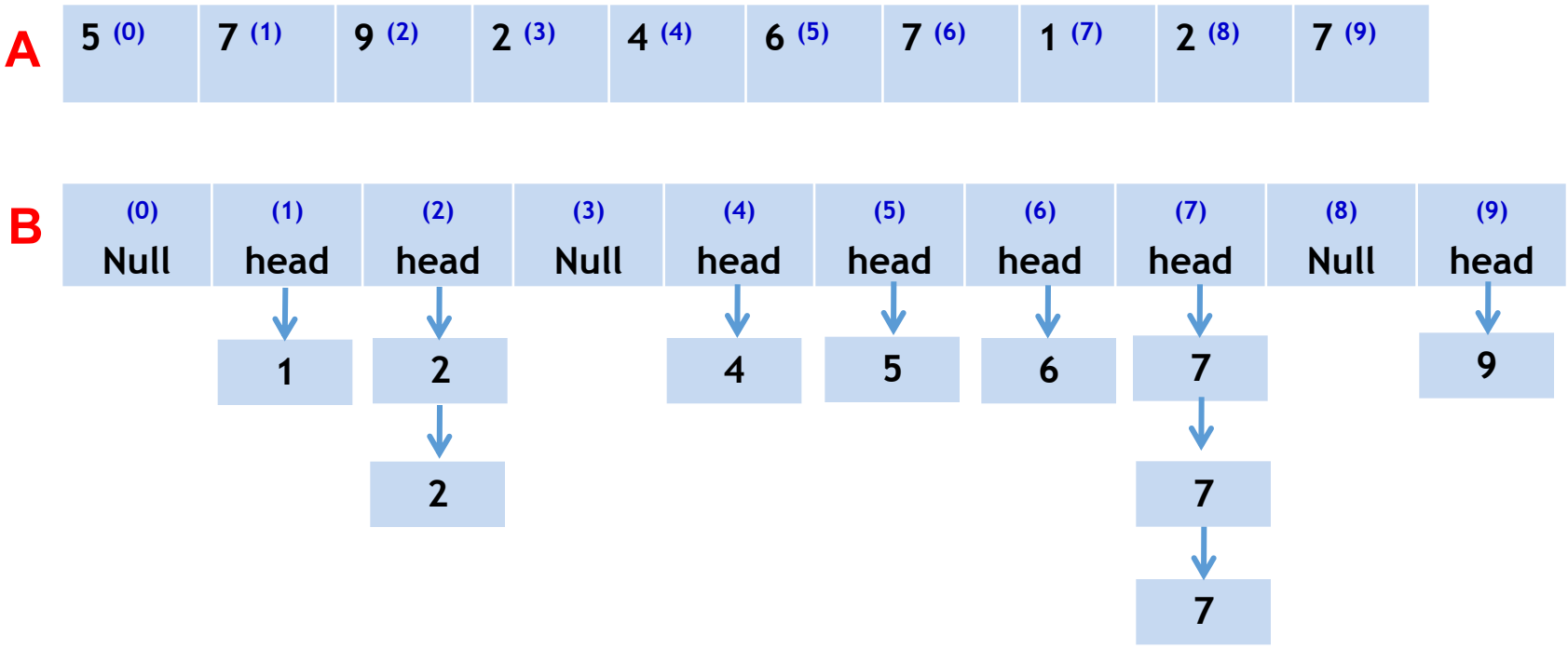
B[1] = 1

Final Output: 1 2 4 5 6 7 9

Counting Sort

- How to cope up with duplicates?
- B becomes a an array of pointers, (rather than an array of integers). Each element has a
  - head pointer (points to the head of a linked list)
  - tail pointer (points to the tail of the linked list)
- A[j] is added at the tail of the list B[A[j]]
- Finally, the array B is sequentially traversed and each nonempty list retrieved.
- Complexity: O(M + N)

Counting Sort – Duplicate keys



Final Output: 1 2 2 4 5 6 7 7 7 9

Radix Sort

- Counting sort becomes expensive if **M** is large compared to **N**
- Radix sort may be built upon counting sort
- **Basic Idea**: every integer can be represented by at most k digits
  - $d_1 d_2 \cdots d_k$ where d_i are digits in base r

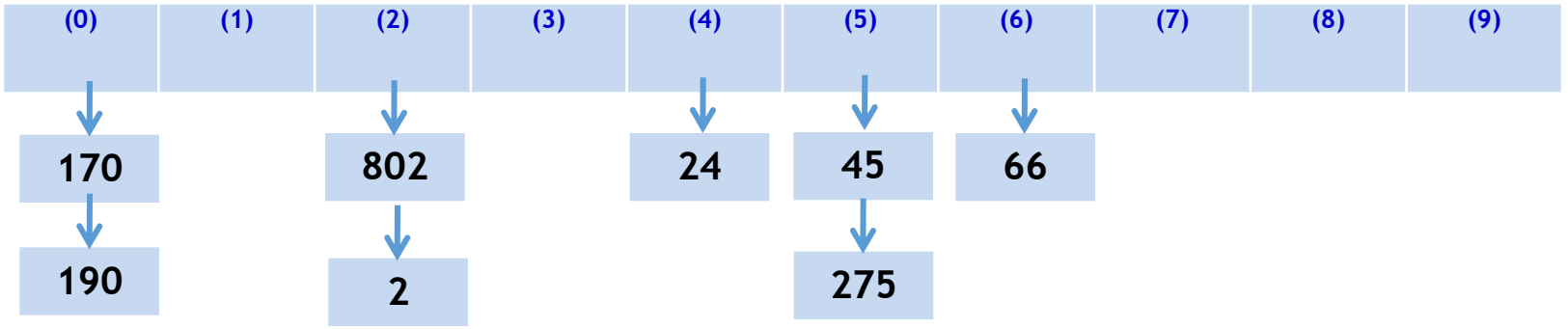
Radix Sort

- Algorithm
  - Take the least significant digit (or group of bits) of each key.
  - Group the keys based on that digit, but otherwise keep the original order of keys
  - Repeat the grouping process with each more significant digit.
  - The sort in step 2 is usually done using **bucket sort** or **counting sort**, which are efficient in this case since there are usually only a small number of digits.

Radix Sort

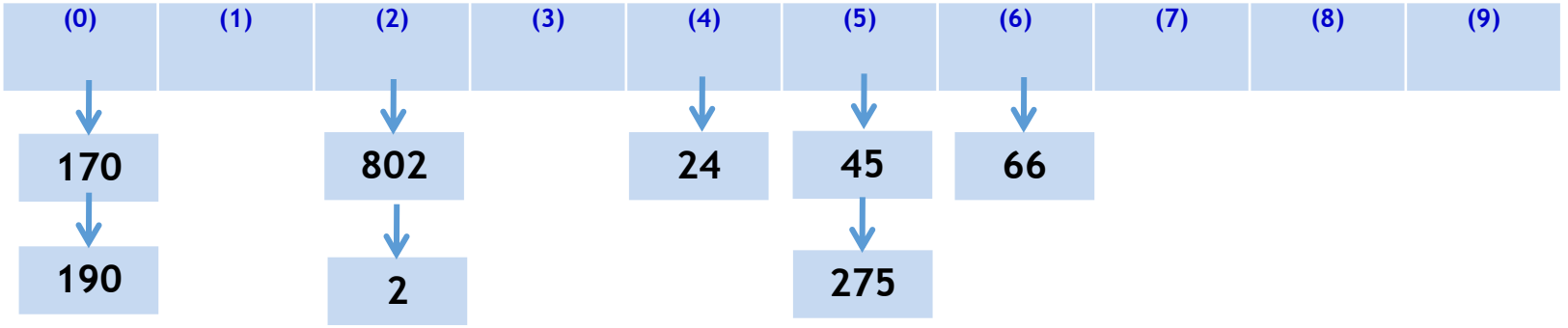
- Example
- Sort: 170, 45, 275, 190, 802, 24, 2, 66

- **First pass**

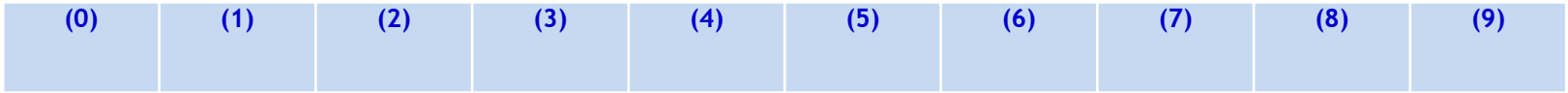


Radix Sort

- **First pass**

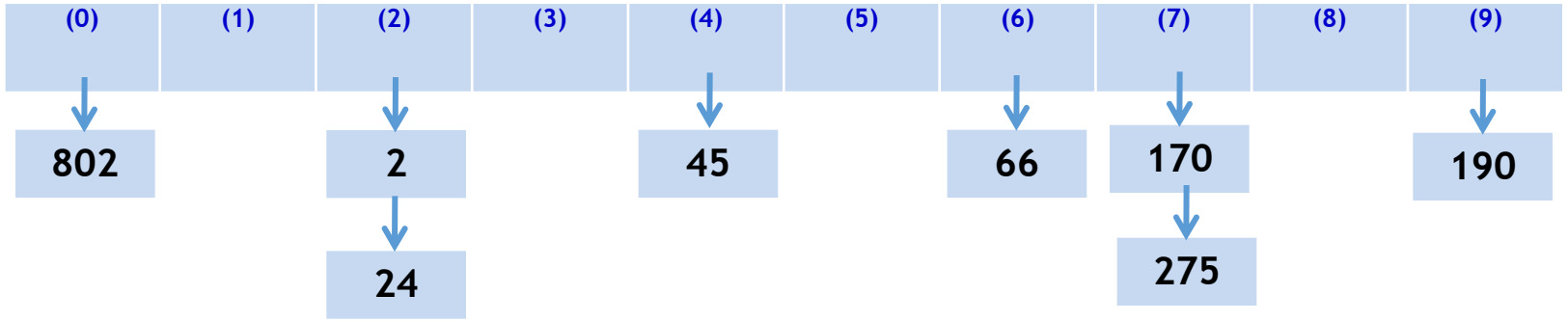


- **Second pass**



Radix Sort

- **Second pass**

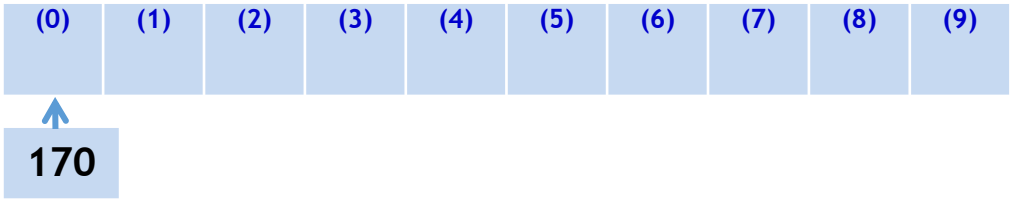


- **Third pass**

Final Output: **2 24 45 66 170 190 275 802**

Radix Sort

170, 45, 275, 190, 802, 24, 2, 66



- Algorithm RadixSort(A, N, d)

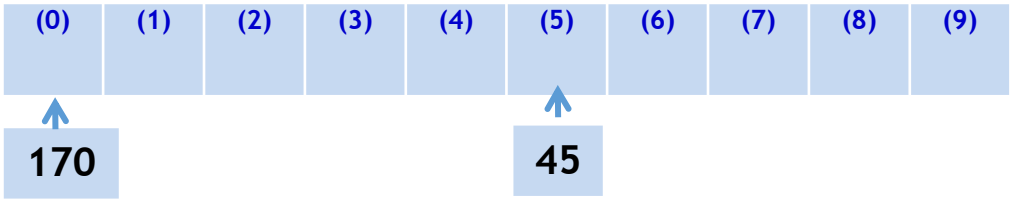
```
for  p = 0 to 9
     Q[p] =Null
D = 1
for  k = 1 to d
     D = D * 10
     for  i = 0 to N
          t = (A[i] mod D) div (D/10)
          enqueue(A[i], Q[t])
     j = 0
     for  p = 0 to N
          do while Q[p] is not empty
               A[j] = dequeue(Q[p])
               j = j +1
```

D = 10

i = 0
t = (A[0] mod 10) div (1)
= 170 mod 10 = 0
enqueue(170, Q[0])

Radix Sort

170, 45, 275, 190, 802, 24, 2, 66



- Algorithm RadixSort(A, N, d)

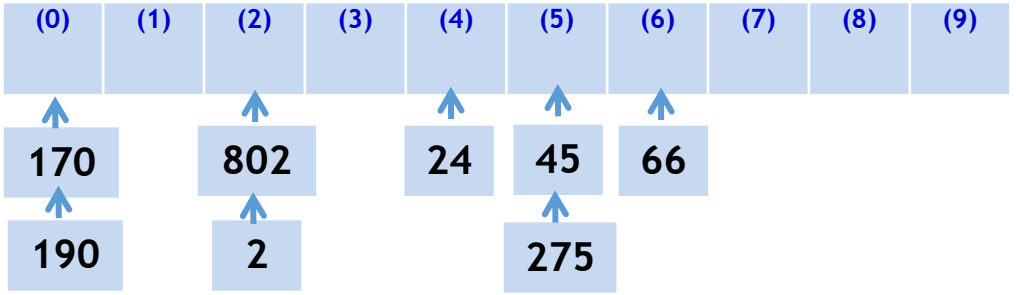
```
for  p = 0 to 9
     Q[p] =Null
D = 1
for  k = 1 to d
     D = D * 10
     for  i = 0 to N
          t = (A[i] mod D) div (D/10)
          enqueue(A[i], Q[t])
     j = 0
     for  p = 0 to N
          do while Q[p] is not empty
               A[j] = dequeue(Q[p])
               j = j + 1
```

D = 10

i = 1
t = (A[1] mod 10) div (1)
= 45 mod 10 = 5
enqueue(45, Q[5])

Radix Sort

170, 45, 275, 190, 802, 24, 2, 66



- Algorithm RadixSort(A, N, d)

```
for  p = 0 to 9
     Q[p] =Null
D = 1
for  k = 1 to d
     D = D * 10
     for  i = 0 to N
          t = (A[i] mod D) div (D/10)
          enqueue(A[i], Q[t])
     j = 0
    for  p = 0 to N
          do while Q[p] is not empty
               A[j] = dequeue(Q[p])
               j = j +1
```

D = 10

i = 2, 4, 5, 6, 7, 8, 9

t = (A[2] mod 10) div (1)

= 275 mod 10 = 5

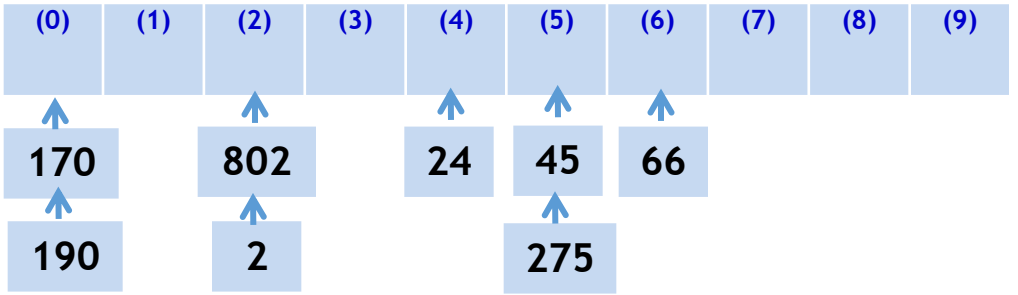
enqueue(245, Q[5])

Radix Sort

- Algorithm RadixSort(A, N, d)

```
for  p = 0 to 9
     Q[p] =Null
D = 1
for  k = 1 to d
     D = D * 10
     for  i = 0 to N
          t = (A[i] mod D) div (D/10)
          enqueue(A[i], Q[t])
     j = 0
     for  p = 0 to N
          do while Q[p] is not empty
               A[j] = dequeue(Q[p])
               j = j +1
```

170, 45, 275, 190, 802, 24, 2, 66

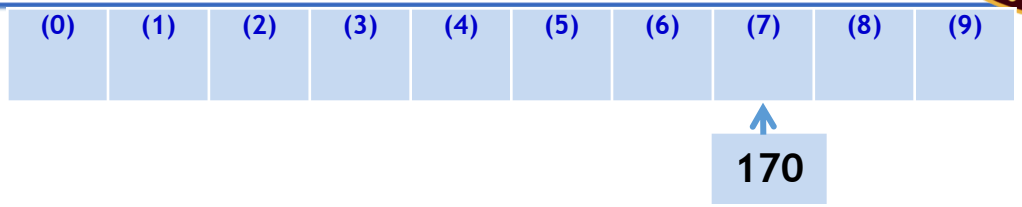


D = 10

A = { 170, 190, 802, 2, 24, 45, 275, 66}

Radix Sort

A = { 170, 190, 802, 2, 24, 45, 275, 66 }



- Algorithm RadixSort(A, N, d)

```
for  p = 0 to 9
     Q[p] =Null
D = 1
for  k = 1 to d
     D = D * 10
     for  i = 0 to N
          t = (A[i] mod D) div (D/10)
          enqueue(A[i], Q[t])
     j = 0
     for  p = 0 to N
          do while Q[p] is not empty
               A[j] = dequeue(Q[p])
               j = j +1
```

D = 100

i = 0

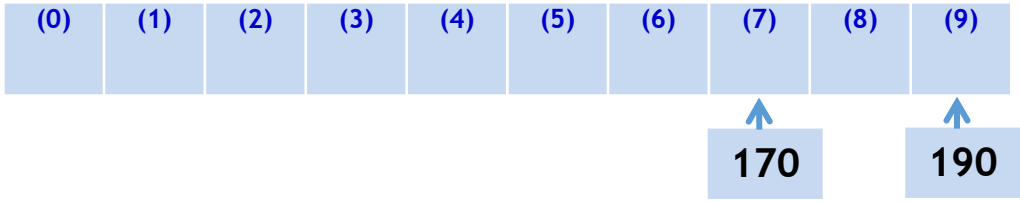
t = (A[0] mod 100) div (10)

= (170 mod 100) div (10) =70/10=7

enqueue(170, Q[7])

Radix Sort

A = { 170, 190, 802, 2, 24, 45, 275, 66 }



- Algorithm RadixSort(A, N, d)

```
for  p = 0 to 9
     Q[p] =Null
D = 1
for  k = 1 to d
     D = D * 10
     for  i = 0 to N
          t = (A[i] mod D) div (D/10)
          enqueue(A[i], Q[t])
     j = 0
    for  p = 0 to N
          do while Q[p] is not empty
               A[j] = dequeue(Q[p])
               j = j +1
```

D = 100

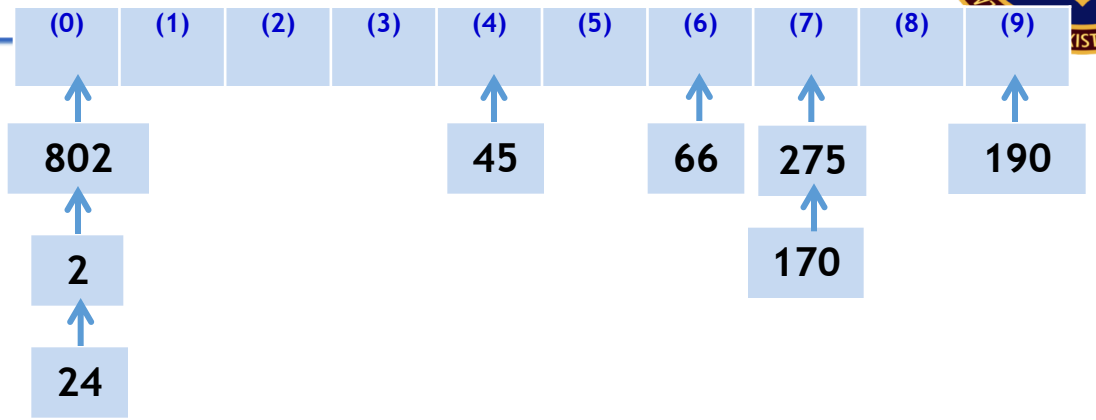
i = 1

t = (A[1] mod 100) div (10)
= (190 mod 100) div (10) = 90/10 = 9

enqueue(190, Q[9])

Radix Sort

A = { 170, 190, 802, 2, 24, 45, 275, 66}



- Algorithm RadixSort(A, N, d)

```
for  p = 0 to 9
     Q[p] =Null
D = 1
for  k = 1 to d
     D = D * 10
     for  i = 0 to N
          t = (A[i] mod D) div (D/10)
          enqueue(A[i], Q[t])
     j = 0
     for  p = 0 to N
          do while Q[p] is not empty
               A[j] = dequeue(Q[p])
               j = j +1
```

D = 100

i = 2, 3, 4, 5, 6, 7, 8, 9

A = {802, 2, 24, 45, 66, 275, 170, 190}

Radix Sort

A = {802, 2, 24, 45, 66, 275, 170, 190}

(0)	(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)	(9)
								802	

- Algorithm RadixSort(A, N, d)

```
for  p = 0 to 9
     Q[p] =Null
D = 1
for  k = 1 to d
     D = D * 10
     for  i = 0 to N
          t = (A[i] mod D) div (D/10)
          enqueue(A[i], Q[t])
     j = 0
    for  p = 0 to N
          do while Q[p] is not empty
               A[j] = dequeue(Q[p])
               j = j +1
```

D = 1000

i = 0

t = (A[0] mod 1000) div (1000)

= (802 mod 1000) div (1000) =802/100=8

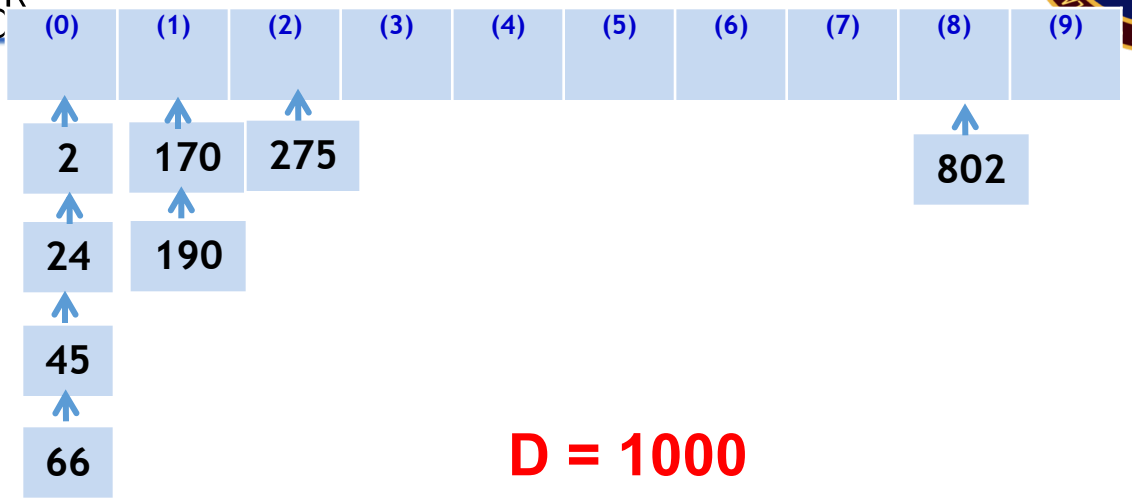
enqueue(802, Q[8])

Radix Sort

A = {802, 2, 24, 45, 66, 275, 170, 190}

- Algorithm RadixSort(A, N, d)

```
for  p = 0 to 9
     Q[p] =Null
D = 1
for  k = 1 to d
     D = D * 10
     for  i = 0 to N
          t = (A[i] mod D) div (D/10)
          enqueue(A[i], Q[t])
     j = 0
     for  p = 0 to N
          do while Q[p] is not empty
               A[j] = dequeue(Q[p])
               j = j +1
```



D = 1000

i = 1, 2, 3, 4, 5, 6, 7, 8, 9

Final Output

A = {2, 24, 45, 66, 170, 190, 275 ,802}

Questions?

zahmaad.github.io