# RADIX SORT

CS340

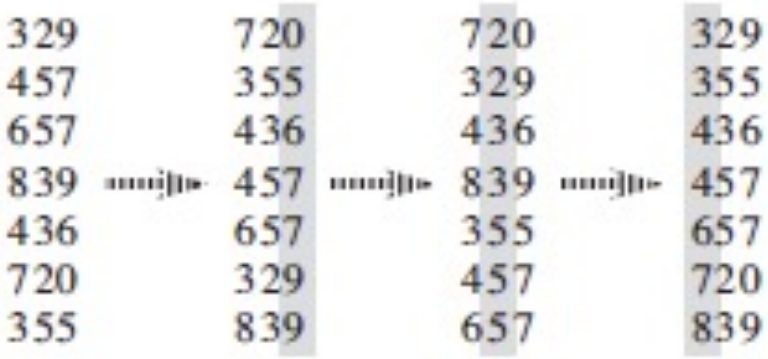
# Radix Sort

- Sorting based on the digits in a number
  - In decimal, there are 10 digits
  - Based on this, you could divide numbers into bins based on their most significant digit, sort the bins, and combine.
  - This means 9/10 bins must be put aside to sort each of the bins.
    - That's a lot of intermediate piles to keep track of.

# Radix Sort

- Radix Sort counterintuitively sorts based on the least significant digit first.
- Then combines all numbers into a single bin.
- Then sorts that bin based on second-least significant digit.
- And so on.
- If the numbers being sorted have d digits, then only d passes through the numbers are required to sort.
- In order for this to work, the sort used at each pass must be stable.

# Radix Sort




RADIX-SORT$(A, d)$

1 **for** $i = 1$ **to** $d$

2 use a stable sort to sort array $A$ on digit $i$

# Radix Sort

- Counting sort is a good choice for the sort on line 2.
- Time complexity of Radix Sort?

```
RADIX-SORT(A, d)
1  for i = 1 to d
2      use a stable sort to sort array A on digit i
```

# Time complexity

- $\Theta(d(n + k))$
- With a computer, the number is binary.
- The binary number consists of b-bits
- We choose how many binary numbers make a digit, r
- Time complexity is then

  $\Theta((b/r)(n + 2^r))$

```
RADIX-SORT(A, d)
1  for i = 1 to d
2      use a stable sort to sort array A on digit i
```

# Example of time complexity

- In the binary world, this time complexity is:
  - $\Theta((b/r)(n + 2^r))$
  - b = number of bits, r = number of bits in a digit
- Eg we are sorting a 32 bit word(b=32), which we consider as having 4 8bit digits (r=8).
- The number of passes was d, d = (b/r) = 4.
- Because there are r bits in a digit, the biggest number is k = $2^r$.

# Example of time complexity

- $\Theta((b/r)(n + 2^r))$
  - b = number of bits, r = number of bits in a digit
  - Eg an ordinary number has b=32, and we choose r=8.
- This formulation is handy because it removes k.
- n and b are given. Choose r to minimize time complexity
- If r=b, the we have
  - $\Theta((b/b)(n + 2^b)) = \Theta(n + 2^b) = \Theta(n)$

# Radix Sort

- Is it preferable to other sorts like Quicksort?
  - Hard to say. $\Theta(n + 2^b)$ vs $\Theta(n \lg n)$
  - Radix sort appears better.
  - Constant factors with Radix Sort may be large.
  - Quicksort makes more passes over the n items, but counting sort passes take longer.
  - Quicksort sorts in place.

```
RADIX-SORT(A, d)
1  for i = 1 to d
2      use a stable sort to sort array A on digit i
```

# How about an example?

- COW
- DOG
- SEA
- RUG
- ROW
- MOB
- BOX

# Radix Sort

- It's a loop
  - What is the loop invariant?
  - Initialization, maintenance, termination proofs?

```
RADIX-SORT(A, d)
1  for i = 1 to d
2      use a stable sort to sort array A on digit i
```

# Sorting in general

- Is Radix Sort the limit?
- Can it be faster than O(n) time?