

Non-Comparison-Based Sorting: Counting Sort

- An input array $A[]$ of n integers.
- Assume all elements (keys) are between 1 and k .
- Uses an array $count[1 \cdots k]$ to determine the frequency of each element.
- Ask class for ideas!

CountingSort(A, n, k)

```
for  $j = 1$  to  $k$ 
     $count[j] = 0$ ;
for  $i = 0$  to  $n - 1$ 
     $count[A[i]] ++$ ;
for  $j = 1$  to  $k$ 
    print  $j$   $count[A[j]]$  times;
```

- Time Complexity: $\Theta(n + k)$.
- This is an $O(n)$ time algorithm as long as $k = O(n)$.
- What happened to the $\Omega(n \log n)$ lower bound?
- Wasteful if the range is large: e.g. SSN.

Radix Sort

- Gets around large ranges by sorting one digit at a time.
- Ancient Punched-Card-Sorting algorithm.
- Ask class for ideas.
- What if we sort from MSB to LSB? (Too many intermediate piles.)

- Sorts from least to most significant digits.
- Use Counting Sort when sorting on digit d_i .
- Important: Sorting needs to be stable—two equal elements keep their original order. (Counting sort is stable.)

329	720	720	329
457	355	329	355
657	436	436	436
839	457	839	457
436	657	355	657
720	329	457	720
355	839	657	839

- Good for sorting when number of digits is small. E.g.

Radix Sort

RadixSort(A, d)

1. for $i = 1$ to d do
2. Use a stable sort to sort A on digit i .

- Proof of correctness by induction on columns.
 - Assuming each digit is in the range $[1, k]$, then counting sort for each digit takes $O(n + k)$ time.
 - Thus, the total time is $O(nd + kd)$, which is $O(n)$ assuming d is a constant and $k = O(n)$.
1. What if k is small and d is large? (e.g. 64-bit numbers)
 2. Group multiple bits to form “digits”. Use 8-digit numbers in the range $[1, 2^8]$.
 3. Suppose 10^6 64-bit numbers. We can radix-sort using 4-digits in radix- 2^{16} system, in four passes. The $\Theta(n \log n)$ algorithms have running time about $20n$.

Summary of Sorting Algorithms

- Comparison-Based:

1. $O(n^2)$: Insertion, Bubble, Selection, Shell

2. $O(n \log n)$: Mergesort, Quicksort.

- Non-Comparison-Based:

1. $O(n + k)$: Counting sort.

2. $O(nd + kd)$: Radix sort.

3. $O(n)$ expected: Bucket sort.

4. $\Theta\left(\frac{bn}{\log n}\right)$: Radix Sort.