Linear-time Sorting

Last time: Any comparison-based sorting alg has worst-case complexity $\Omega(n \log n)$

Can we sort in a way not based on comparing elements?
Consider sorting $n$ elements that are all integers $\{0, 1, \ldots, k-1\}$

E.g. $k=10$

Idea: keep 10 lists and put all elements of value $i$ into list $i$

Using an array: Count # of occurrences of each element and then you can put them in order
Counting Sort

basic idea we've just described.

We'll need one more property

Stable sort - any two equivalent elements are kept in same relative order

d, a, b, c, b  \rightarrow  a, b, b, c, d  \text{ stable}
Radix

CountingSort(input, output, k) {
    for (d = 0 to #digits - 1)
        int n = input.length;
        for (i = 0; i < k; i++)
            count[i] = 0;
        for (j = 0; j < n; j++)
            count[input[j]]++;
        for (i = 1; i < k; i++)
            count[i] += count[i - 1];
        for (j = n - 1; j >= 0; j --)
            output[--count[input[j]]] = input[j];
    RadixSort digitizer = getDigit(input[j], d); current digit
}
Kansas State University

$k = 3$

Input: 2 1 0 1 1 0 0 2

Output:

<table>
<thead>
<tr>
<th>Count</th>
<th>after 1st loop</th>
<th>after 2nd loop</th>
<th>after 3rd loop</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>3</td>
<td>3 X 1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>3</td>
<td>6 X 5</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>2</td>
<td>8 X 7</td>
</tr>
</tbody>
</table>
Time complexity

$\Theta(n+k)$

two loops over counters $\Theta(k)$
two loops over elements $\Theta(n)$

When $k=O(n)$ then this

is a $\Theta(n)$ sort.

linear time
Suppose I want to sort social security number

9 digits

What would "k" be if we wanted to use counting sort? \(10^9\)