06-Linear-time Sorting

No linear-time sort can rely on comparisons between items.

**Counting Sort**

Sorts items in range 0 to k-1

Steps:

1) Make array of ints of length k (zeros)

2) Increment $C[\text{In}[i]]$ for $0 \leq i \leq n$

3) $C[i] = C[i] + C[i-1]$ for $1 \leq i \leq k$

4) Make output array of length n

5) $\text{Out}[C[\text{In}[i]]] = \text{In}[i]$, $C[i]--$, $n \geq i \geq 0$

**Example**

**In:** 3 6 8 4 1 5 9 5 4 1 2 3 1 2 7 9

**C:** 0 3 2 2 2 1 1 1 2

**Out:** 1 1 1 2 2 3 3 4 4 5 5 6 7 8 9 9

$\Theta(n+k) = \Theta(n)$ for $k = O(n)$
Stability
A sort is stable if elements with the same key have the same relative position after the sort.

Why is this good?
1) Helpful w/ multiple keys
2) Keeps satellite data in same relative order
Radix Sort

\[ \begin{array}{cccc}
355 & 720 & 720 & 329 \\
720 & 355 & 329 & 355 \\
436 & 436 & 436 & 436 \\
657 & 657 & 839 & 839 \\
657 & 457 & 457 & 457 \\
457 & 839 & 839 & 839 \\
329 & 329 & 329 & 329 \\
\end{array} \]

\[ \Theta(d(n+k)) = \Theta(n) \text{  if  } k=O(n) \text{  and  } d=O(1) \]
What about other bases?

We used base 10 in Radix sort above. Can we use base 2?

Can we use Radix Sort to sort $n$ numbers in the range $0$ to $n^2 - 1$? Sure. Set $k = n$, $d = 2$.

Max is if $n$ numbers range from $0$ to $n^2 - 1$. 
Disadvantages:

1) Constants get huge
2) Need extra space
3) Sometimes we don't know the range
Bucket Sort

n input elements over interval \([0, 1)\).

\[0.31, 0.42, 0.18, 0.39, 0.90, 0.81, 0.63, 0.58, 0.12, 0.98\]

1) make array len \(n\)

\[0, 0.1, 0.2\]

\[0.18 \rightarrow 0.12\]

2) put elements in appropriate bucket

\[0.31 \rightarrow 0.39\]

\(n_i = \#\) elements in \(i^{th}\) list

3) sort each bucket with insertion sort

\[0.42, 0.58, 0.63\]

\[0.81\]

\[0.90 \rightarrow 0.98\]
Running Time

\[ E\left[ O(n + \sum_{i=1}^{n} (n_i)^2) \right] \]
\[ O(n + \sum_{i=1}^{n} E[n_i^2]) \]
\[ O(n + \sum_{i=1}^{n} 2) = O(3n) = O(n) \]