How to best exploit Radix Sort to sort natural numbers whose size is bounded, but not by $O(n)$ [hence counting sort does not apply]? View inputs in terms of their bit representation. Suppose the inputs can be stored using $b$ bits.

- could apply Radix Sort directly with $d = b$ and $R=2$ (options 0, 1), getting $O(b \cdot n)$ time.

- Alternatively, can group bits into blocks to create digits taking a larger range of options. Suppose we choose "block size" $w$ (we can choose) ..... then each block can take on $2^w$ possible values ..... get $R = 2^w$ and $d = (b/w)$

The value $R$ only affects the "counting sort" runtime $O(n+R) \Rightarrow$ we might as well make $R$ as big as possible up to $n$ (this gets the $O(n)$)

$$\Rightarrow \text{Make "block size" } w = \lceil \lg(n) \rceil \Rightarrow O(n+R) = O(n).$$

$$\lceil \lg(n) \rceil \text{ block size } \Rightarrow \frac{b}{\lceil \lg(n) \rceil} \text{ digits, } d = \frac{b}{\lceil \lg(n) \rceil}$$

$$\Rightarrow \text{ Radix Sort is } O(n \times \frac{b}{\lceil \lg(n) \rceil}) \text{ better than }$$

numbers $\leq n^k$

$b$ is $\lceil k \lg(n) \rceil \Rightarrow$ with $w = \lceil \lg(n) \rceil$, get $O(kn)$ for Radix Sort, which is $O(n)$ for constant $k$: -)