Radix Sort

Radix sort sorts numbers by sequential digit given an input of n numbers with a maximum of d digits.

```
def RadixSort:
   for i in [0, d-1]:
      StableSort(A, key=digit i)
```

A good sort to use to implement Radix sort is Counting sort because it is stable and we can pick the radix r of the numbers to optimize the efficiency of the algorithm. The runtime complexity of Radix sort for radix r is $\Theta(\log_r S(n+r))$, where S is the range of values in the input array ($\log_r S = d$).

Optimal Radix Take the derivative of the runtime complexity to optimize r.

$$\frac{d}{dr}\left(\frac{c\ln S}{\ln r}(n+r)\right) = 0 \tag{1}$$

$$c\ln S\left(\frac{\ln r - n/r - 1}{(\ln r)^2}\right) = 0 \tag{2}$$

The nontrivial solution to this equation is

$$r = \frac{n}{\ln r - 1} \tag{3}$$

Guess $r \stackrel{?}{=} n$, which gives an inconsistent result

$$r = \frac{n}{\ln n - 1} \tag{4}$$

Substituting this value for r and evaluating again yields

$$r = \frac{n}{\ln n - \ln\ln n - 1} \tag{5}$$

The trend continues, and because $\ln\ln n$ is almost negligible, it's enough to say that the best radix to pick for the sort is

$$r = \frac{n}{\ln n - 1} \tag{6}$$

Note that in practice, it's often best to pick a radix that is a power of 2 to take advantage of hardware. If a power of 2 is close to $n/(\ln n - 1)$, then it's the best base to pick.