CONVERT BASE: THE MYSTERY BEHIND THE ALGORITHM

ERIC ROTHSTEIN

Some of you appear to be confused by what the convertBase algorithm does. This document aims to bring a bit more of insight into what is going on when we execute the convertBase algorithm.

1. Preliminaries

Consider any number n in any base b. For example, $n_1 = 1111$ and $b_1 = 2$, or $n_2 = f$ and $b_2 = 16$. Although the numbers n_1 and n_2 look different (they are, in fact, written differently), they both represent the same *abstract quantity*: 1111_2 apples is the same as f_{16} apples. What is this *abstract quantity* precisely? This is what we will be defining next using a base that is familiar to us: base 10.

We can represent numbers with arrays of *integer digits* in base 10. For example, $n_1 = [1, 1, 1, 1]$ and $n_2 = [15]$ (note that, since f is not an integer, we have to use its base 10 equivalent *number*, in this case 15). We define $AQ([d_0, \ldots, d_{m-1}], b)$, the *abstract quantity of an array of digits of size m $[d_0, \ldots, d_{m-1}]$ in a base b*, by

$$\begin{aligned} AQ([d_0, \ldots, d_{m-1}], b) &= d_0 b^{m-1} + d_1 b^{m-2} + \ldots + d_{m-1} b^0 \\ &= b^m \left(d_0 \frac{1}{b} + d_1 \frac{1}{b^2} + d_2 \frac{1}{b^3} + \ldots + d_{m-1} \frac{1}{b^m} \right) \\ &= b^m \underbrace{\sum_{i=0}^{m-1} d_i \left(\frac{1}{b}\right)^{i+1}}_{\text{Remember this}}. \end{aligned}$$

For example,

$$\begin{aligned} AQ([1, 1, 1, 1], 2) &= 1 \times 2^3 + 1 \times 2^2 + 1 \times 2^1 + 1 \times 2^0 \\ &= 15 \text{ (in base 10)} \end{aligned}$$

$$\begin{aligned} AQ([15], 16) &= 15 \times 16^0 \\ &= 15 \text{ (in base 10)} \end{aligned}$$

Now, we are not going to work with integers, but with fractional numbers between 0 and (strictly less than) 1. For example, $n_3 = 0.1$ with $b_3 = 3$. How do we represent fractional numbers? We also use arrays! However, to n_3, we will now have to talk about precision: the array $[1]$ and the array $[1, 0, 0, 0]$ both represent the *fractional* part of n_3, but they have different precision; the former has precision 1, and the latter has precision 4.

Let us now talk about the abstract quantity of a fractional number n, with $0 \leq n < 1$, whose fractional part is represented with an array $\vec{n}$ with precision $\vec{n}$`.length` in a base b. Let us call it $AQ_f(\vec{n}, b)$. For $n_3 = 0.1$, let $\vec{n_3} = [1, 0, 0]$. We want $AQ_f([1, 0, 0], 3)$ to be equal to $\frac{1}{3}$. Thus, we need to slightly modify the formula for $AQ_f(\vec{n}, b)$, as follows: $AQ_f(\vec{n}, b)$

$$\begin{aligned} AQ_f([d_0, \ldots, d_{m-1}], b) &= d_0\frac{1}{b} + d_1\frac{1}{b^2} + d_2\frac{1}{b^3} + \ldots + d_{m-1}\frac{1}{b^m} \\ &= \underbrace{\sum_{i=0}^{m-1} d_i \left(\frac{1}{b}\right)^{i+1}}_{\text{Looks familiar?}} \end{aligned}$$

$$\begin{aligned} AQ_f([1, 0, 0, 0], 3) &= 1 \times 3^{-1} + 0 \times 3^{-2} + 0 \times 3^{-3} + 0 \times 3^{-4} \\ &= \frac{1}{3} \text{ (in base 10).} \end{aligned}$$

Representing $\frac{1}{3}$ as an array in base 10 would require us to have infinite digits, since $\frac{1}{3} = 0.3333\ldots$. We will need to restrict this representation to some precision p. Let us consider the case where $p = 3$

$$\begin{aligned} AQ_f([3, 3, 3], 10) &= 3 \times 10^{-1} + 3 \times 10^{-2} + 3 \times 10^{-3} \\ &= 0.333 \\ &< \frac{1}{3} \text{ (in base 10).} \end{aligned}$$

We now take a look at what the `convertBase` is supposed to do.

2. Convert Base

The algorithm `convertBase` takes the *fractional part* of a number, represented as an array `digits` in a base `baseA` and returns a representation of the same number as an array `output` in base `baseB` with some `precision`. For example, if `digits` $= [1, 0]$, `baseA` $= 3$, `baseB` $= 10$ and `precision` $= 5$, then `output` $= [3, 3, 3, 3, 3]$.

Now, the idea behind `convertBase` relies on the equality

$$AQ_f(\texttt{digits}, \texttt{baseA}) = AQ_f(\texttt{output}, \texttt{baseB}).$$

The equality above implies the following:

$$\underbrace{AQ_f(\texttt{digits}, \texttt{baseA}) - \texttt{output}[0]\left(\frac{1}{\texttt{baseB}}\right)}_{AQ_f(\texttt{digits}', \texttt{baseA})} = AQ_f(\texttt{tail}(\texttt{output}), \texttt{baseB})$$

where `tail`(`output`) is equal to `output`, except that it does not have the first element (for example, if `output` $= [1, 2, 3, 4]$, then `tail`(`output`) $= [2, 3, 4]$), and `digits`$'$ represents a new number, most likely smaller than the number represented by `digits`.

The algorithm `convertBase` has a systematic way of computing $\texttt{digits}'$ so that the equality above is preserved, yielding an element of `output` during each iteration. Do note that

$$\underbrace{AQ_f(\texttt{digits}', \texttt{baseA}) - \texttt{tail}(\texttt{output})[0]\left(\frac{1}{\texttt{baseB}}\right)}_{AQ_f(\texttt{digits}'', \texttt{baseA})} = AQ_f(\texttt{tail}(\texttt{tail}(\texttt{output})), \texttt{baseB})$$

is equivalent to

$$\underbrace{AQ_f(\texttt{digits}', \texttt{baseA}) - \texttt{output}[1]\left(\frac{1}{\texttt{baseB}}\right)}_{AQ_f(\texttt{digits}'', \texttt{baseA})} = AQ_f(\texttt{tail}^2(\texttt{output}), \texttt{baseB})$$

and we can continue

$$\underbrace{AQ_f(\texttt{digits}'', \texttt{baseA}) - \texttt{output}[2]\left(\frac{1}{\texttt{baseB}}\right)}_{AQ_f(\texttt{digits}''', \texttt{baseA})} = AQ_f(\texttt{tail}^3(\texttt{output}), \texttt{baseB})$$

etc.

Your implementation should focus on finding the values of $\texttt{digits}'$, $\texttt{digits}''$, $\texttt{digits}'''$, etc. (which can be done in the same array). This computation can be achieved by implementing the following instructions:

for ($i <$ `precisionB`){
- (1) initialize a `carry` to 0.
- (2) from RIGHT to LEFT{ // (using an index j)
  - (a) $\texttt{x} = \texttt{digits}[\texttt{j}] \times \texttt{baseB} + \texttt{carry}$
  - (b) $\texttt{digits}[\texttt{j}] = \texttt{x}\%\texttt{baseA}$
  - (c) $\texttt{carry} = \texttt{x}/\texttt{baseA}$
  - (d) }
- (3) $\texttt{output}[\texttt{i}] = \texttt{carry}$

}

IMPORTANT: make sure that you DO NOT mutate your `digits` array, create a copy of it, and work on the copy instead.