COMPUTATIONAL MATHEMATICS
TOPIC III: BASES

PAUL L. BAILEY

1. Numbers and Numerals

Consider the difference between the words *number* and *numeral*, as they are used by mathematicians.

Webster's New World dictionary defines number as *a symbol or word, or a group of either of these, showing how many or which one in a series.* This is clearly not what we mean when we refer to rational or real numbers. Yet, the alternate definitions are even further from our usage. Perhaps closer would be *an idea corresponding to a quantity.*

Webster's does a better job with the second word, defining numeral as *a figure, letter, or word, or a group of any of these, expressing a number.* So if a number is an idea, a numeral is an expression of an idea.

Our standard way of writing numbers is positional, and depends on the choice of ten as a base. However, the choice of ten is arbitrary, and other cultures have made other choices.

Computers naturally use base two, which is easy to convert to base eight or sixteen. So, computer professionals are also conversant in these bases. We describe more fully what bases are, and how to convert between them.

2. Base b

Suppose we wish to write numbers using a fixed number of symbols. Say we have b symbols, and each of these represents a different number between zero and $b-1$. We call each of these symbols a *digit.*

Let n be any natural number and let k be the largest integer such that $n \geq b^k$. We can n in a unique way as a sequence of digits as follows:

$$n = (a_k a_{k-1} \dots a_1 a_0)_b = a_k b^k + a_{k-1} b^{k-1} + \cdots + a_1 b + a_0,$$

where $a_0, a_1, \dots, a_k$ are digits, each representing a number which is less than b. This is called the *base b representation* of the number n, and we call b the *base.*

Similarly, if q is a real number, we can express an approximation of q using j negative powers of b as follows:

$$q = (a_k a_{k-1} \dots a_1 a_0 . a_{-1} a_{-2} \dots a_{-j})_b = a_k b^k + \cdots + a_0 + a_{-1} b^{-1} + \cdots + a_{-j} b^{-j}.$$

Date: August 16, 2018.

3. Converting from Base b to Base 10

We typically write numbers using base ten, for example,

$$(25283)_{10} = 2 \times 10^4 + 5 \times 10^3 + 2 \times 10^2 + 8 \times 10 + 3.$$

However, the choice of ten as a base is arbitrary. We give some examples of writing numbers is different bases.

- $(1011)_2 = 1 \times 2^3 + 0 \times 2^2 + 1 \times 2 + 1 = 8 + 0 + 2 + 1 = 11$
- $(1234)_5 = 1 \times 5^3 + 2 \times 5^2 + 3 \times 5 + 4 = 125 + 50 + 15 + 4 = 194$
- $(1357)_8 = 1 \times 8^3 + 3 \times 8^2 + 5 \times 8 + 7 = 512 + 192 + 40 + 7 = 651$

Example 1. Convert $(582)_7$ to decimal.

Solution. Let $b = 7$; this is the original base. Let k be one less then the number of digits; since there are three digits, $k = 2$. Thus, $b^k = 7^2 = 49$ is the highest power of seven which is less than the number.

For each digit, write that digit times the appropriate power of 7, and add together the results:

$$(582)_7 = 5 \times 7^2 + 8 \times 7 + 2 = 5 \times 49 + 8 \times 7 + 2 = 345 + 56 + 2 = 403.$$

□

Consider the set of digits $\{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F\}$. There are sixteen of these, so we can write numbers in base sixteen with them. For example,

- $(0A5F)_{16} = 0 \times 16^3 + 10 \times 16^2 + 5 \times 16 + 15 = 0 + 2560 + 80 + 15 = 2655$

Example 2. Convert $(1AF8)_{16}$ to decimal.

Solution. In this case, there are four digits to $k = 3$ and $b = 16$. Since $A = 10$ and $F = 15$, we have

$$(1A3F)_{16} = 1 \times 16^3 + 10 \times 16^2 + 3 \times 16 + 15 = 4096 + 2560 + 48 + 15 = 6719.$$

□

4. Converting from Base 10 to Base b

When we add, subtract, or multiply two integers, we obtain one integer as the result. But when we divide one integer into another, we produce two integers.

Let m and n be integers. Divide m into n to obtain the *quotient* q, which is the largest integer such that $n \geq mq$, and the *remainder* r. We have

$$n = mq + r \qquad \text{and} \qquad 0 \leq r < m.$$

We may convert from base 10 to any base b by successively dividing by b and taking the remainder. To see this, let n be the number given in base b as $(a_k a_{k-1} \dots a_1 a_0)_b$. Then

$$\begin{aligned} n &= (a_k a_{k-1} \dots a_1 a_0)_b \\ &= a_k b^k + a_{k-1} b^{k-1} + \cdots + a_1 b + a_0 \\ &= b(a_k b^{k-1} + a_{k-1} b^{k-2} + \cdots + a_1) + a_0. \end{aligned}$$

Since $a_0 < b$, it is clear that if we divide n by b, the quotient is $(a_k a_{k-1} \dots a_1)_b$ and the remainder is a_0. By the same reasoning, if we now divide this quotient by b, the remainder will be a_1. In this way, we can pick the base b digits out of n.

To summarize, if we wish to convert a number n to base b, we divide by b, then divide the quotient by b, and continue until the quotient is zero. The digits of the base b expression of n are the remainders we obtain along the way. The first remainder is the least significant, and the last remainder is the most significant.

Example 3. Convert $(72)_{10}$ to base five.

Solution. Repeatedly divide by 5.
Divide 73 by five to get $73 = 5 \times 14 + 3$, so $q = 14$ and $r = 3$.
Divide 14 by five to get $14 = 5 \times 2 + 4$, so $q = 2$ and $r = 4$.
Divide 2 by five and get $2 = 5 \times 0 + 2$, so $q = 0$ and $r = 2$.
Thus $(72)_{10} = (243)_5$.
Check this: $(243)_5 = 2 \times 5^2 + 4 \times 5 + 3 = 50 + 20 + 3 = 73$. □

Example 4. Convert $(463)_{10}$ to binary.

Solution. Repeatedly divide by two.
Divide 463 by 2 and get $q = 231$ and $r = 1$.
Divide 231 by 2 and get $q = 115$ and $r = 1$.
Divide 115 by 2 and get $q = 57$ and $r = 1$.
Divide 57 by 2 and get $q = 28$ and $r = 1$.
Divide 28 by 2 and get $q = 14$ and $r = 0$.
Divide 14 by 2 and get $q = 7$ and $r = 0$.
Divide 7 by 2 and get $q = 3$ and $r = 1$.
Divide 3 by 2 and get $q = 1$ and $r = 1$.
Divide 1 by 2 and get $q = 0$ and $r = 1$. We stop here; this is the leftmost bit.
Thus $(563)_{10} = (111001111)_2$. □

Example 5. Convert $(2607)_{10}$ to hexadecimal.

Proof. Divide 2607 by 16 to get $q = 162$ and $r = 15$, where $(15)_{10} = (F)_{16}$.
Divide 162 by 16 and get $q = 10$ and $r = 2$.
Divide 10 by 16 and get $q = 0$ and $r = 10$, where $(10)_{10} = (A)_{16}$.
Thus $(2607)_{10} = (A2F)_{16}$. □

5. Rapid Polynomial Evaluation

Consider the polynomial

$$f(x) = a_0 + a_1x + a_2x^2 + \cdots + a_nx^n.$$

The naive way to evaluate this polynomial at a given value for x involves evaluating each monomial separately and adding the values together. This requires n additions and $\sum_{i=1}^{n} n = \frac{n(n+1)}{n}$ multiplications.

However, we may factor the polynomial thusly:

$$f(x) = a_0 + x(a_1 + x(a_2 + \cdots + x(a_{n-1} + x(a_n))\ldots)).$$

Evaluating this at the same x requires n additions and n multiplications.

6. Integer Base Algorithm

Select a positive integer $b \geq 2$ to use as a base.

Let $n \in \mathbb{Z}$; without loss of generality assume n is positive. The division algorithm states that

$$n = bq + r \text{ for some } q, r \in \mathbb{Z} \text{ with } 0 \leq r < b.$$

The last statement, that $r < b$, is critical in what follows. It states that r is a digit in base b.

Repeat this process in a manner similar to the Euclidean algorithm, but crucially different, as follows. Set $q_0 = n$, $q_1 = q$, and $r_0 = r$ so that the above equation becomes

$$q_0 = bq_1 + r_0.$$

Then inductively compute

$$q_i = bq_{i+1} + r_i.$$

Since the q_i's are positive and decreasing, this process eventually ends, say at the k^{th} stage, so that

$$q_{k-1} = bq_k + r_{k-1} \qquad \text{with } 0 \leq q_k < b.$$

Unwind all this in the same manner as the Euclidean algorithm (setting $r_k = q_k$ and commuting the r_i's to the front); something different occurs:

$$q_{k-2} = r_{k-2} + b(r_{k-1} + br_k);$$

$$q_{k-3} = r_{k-3} + b(r_{k-2} + b(r_{k-1} + br_k));$$

and so forth until finally

$$n = q_0 = r_0 + b(r_1 + b(r_2 + b \ldots (r_{k-1} + br_k) \ldots)).$$

This can be rewritten in standard polynomial form, using summation notation, as

$$n = \sum_{i=0}^{k} r_i b^i.$$

7. Rational Base Algorithm

Let x be a positive real number; in practice, x rational, but we won't use that here. Let

$$p = \max\{n \in \mathbb{N} \mid x - n \geq 0\}.$$

Let $z = x - p$; then $0 \leq z < 1$, and $x = p + z$.

Assume that $z = z_0$ is a positive integer with $0 \leq z_0 < 1$. Then $0 \leq bz_0 < b$. Write

$$p_1 = bz_0 - z_1 \quad \text{ where } 0 \leq z_1 < 1.$$

Repeat this: $p_2 = bz_1 - z_2$, $p_3 = bz_2 - z_3$, etc. Eventually, either $z_i = 0$ or i exceeds some predetermined constant k:

$$p_{k+1} = bz_k - z_{k+1}.$$

Discard z_{k+1}; we have and approximation:

$$p_{k+1} \approx bz_k.$$

Solve each of these equation for the earlier z_i term:

$$z_i = b^{-1}(p_{i+1} + z_{i+1}).$$

Rewind this by plugging in z_{i+1}:

$$z_k \approx b^{-1}p_{k+1};$$
$$z_{k-1} \approx b^{-1}(p_k + b^{-1}p_{k+1});$$
$$z_{k-2} \approx b^{-1}(p_{k-1} + b^{-1}(p_k + b^{-1}p_{k+1}));$$

and so forth, until eventually

$$z_0 \approx b^{-1}(p_1 + b^{-1}(p_2 + b^{-1}(\ldots b^{-1}(p_k + b^{-1}p_{k+1})\ldots))).$$

This can be rewritten using summation notation as

$$z = \sum_{i=1}^{k+1} p_i b^{-i}.$$

8. Mathematical Exercises

Exercise 1. Convert the following numbers to decimal.

(a) $(101001)_2$
(b) $(765)_8$
(c) $(4A9)_{16}$
(d) $(1234)_7$

Exercise 2. Perform the indicated conversion.

(a) Convert $(172)_{10}$ to base 5.
(b) Convert $(301)_{10}$ to octal.
(c) Convert $(123)_{10}$ to binary.
(d) Convert $(200)_{10}$ to hexadecimal.

Exercise 3. Convert the roman numeral LXVII to decimal, binary, octal, and hexadecimal.

Exercise 4. Add $(00110011)_2$ to $(10101010)_2$, without converting to decimal. Then convert all numbers to decimal to check your work.

Exercise 5. Multiply $(00110011)_2$ by $(10101010)_2$, without converting to decimal. Then convert all numbers to decimal to check your work.

9. Programming Exercises

Program 1. Write a program which includes the following:

```
char *D="0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZ";
int btoi(char *ext, int base)
void itob(int num, int base, char *ext);
```

The function `btoi` converts the string `ext`, expressed in base `base`, into an internal (computer) integer. The function `itob` converts an internal integer into a string in base `base`, and stores the string in `ext`.

Program 2. Write a program which includes the following:

```
char *D="0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZ";
float btof(char *ext, int base)
void ftob(float num, int base, char *ext);
```

The function `ftoi` converts the string `ext`, expressed in base `base`, into an internal (computer) floating point number. The function `itob` converts an internal floating point into a string in base `base`, and stores the string in `ext`.

Department of Mathematics and CSci, BASIS Scottsdale
Email address: `plbailey@saumag.edu`