NUMERICAL ANALYSIS TOPIC II
BASE CONVERSION ALGORITHM

PAUL L. BAILEY

1. 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:

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

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

Date: January 25, 2004.

2. Integer Base Algorithm

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

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

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

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

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

$$q_0 = \beta q_1 + r_0.$$

Then inductively compute

$$q_i = \beta q_{i+1} + r_i.$$

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

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

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:

$$q_{k-2} = r_{k-2} + \beta(r_{k-1} + \beta r_k);$$
$$q_{k-3} = r_{k-3} + \beta(r_{k-2} + \beta(r_{k-1} + \beta r_k));$$

and so forth until finally

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

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

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

Program 1. Write a program which includes the following:

```
char *D="0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz";
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`.

3. Rational Base Algorithm

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

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

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 $0 \leq \beta z_0 < \beta$. Write

$$p_1 = \beta z_0 - z_1 \quad \text{ where } 0 \leq z_1 < 1.$$

Repeat this: $p_2 = \beta z_1 - z_2$, $p_3 = \beta z_2 - z_3$, etc. Eventually, either $z_i = 0$ or i exceeds some predetermined constant k:

$$p_{k+1} = \beta z_k - z_{k+1}.$$

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

$$p_{k+1} \approx \beta z_k.$$

Solve each of these equation for the earlier z_i term:

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

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

$$z_k \approx \beta^{-1} p_{k+1};$$

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

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

and so forth, until eventually

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

This can be rewritten using summation notation as

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

Program 2. Write a program which includes the following:

```
char *D="0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz";
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 & CSCI, Southern Arkansas University
E-mail address: `plbailey@saumag.edu`