Jose Franco Baquera

April 29, 2018

CS 273

Homework #12

Read Section 18 of 273 booklet (page 45 to 47). Answer the following questions.

1. Describe the representation of IEEE single precision floating point numbers.

IEEE devised a representation of floating point numbers at a universal level, which allowed all computers to use the same format for floating point representation. In essence, the representation of IEEE single precision floating point numbers can be evaluated with the following equation:

$$\textbf{Number} = (-1)^{s} * 1.\text{mmmmmmmmmmmmmmmmmmmmmmm} * 2^{(\text{eeeeeeee} - 127)}$$

We note the following:

s = signed bit (0 = positive, 1 = negative)

e = exponent bits that represent what exponent ***base 2*** we will use

m = mantissa bits that represent the significant digits of the number

Single precision floating-point numbers in IEEE format have 1 signed bit, 8 exponent bits, and 23 mantissa bits "in that order, from left to right." In other words, if we were to look at any 32 bits that represent a single precision point number, it would have the following structure in memory: *seeeeeeeemmmmmmmmmmmmmmmmmmmmmm*. We note the sign bit is for the sign of the overall mantissa and that the exponent stored in the bits is the actual exponent +127. It is important to also note that the exponent is a power of 2 (***NOT*** 10) and that there is a hidden "1" bit before the mantissa in order to allow us to "free" one bit, thus producing more accuracy in single precision floating point numbers (i.e. we always assume that a binary mantissa always begins with 1). Lastly, exponents of all 1's and all 0's are *reserved*.

2. How many different single precision numbers are represented with the IEEE standard?

We note that single precision numbers with the IEEE standard use a total of 32 bits. Therefore, we can represent a total of 2^{32} = 4294967296 different numbers! Among these, we have:

- Negative infinity (s = 1, all 23 mantissa bits = 0, all exponent bits = 1).
- Positive infinity (s = 0, all 23 mantissa bits = 0, all exponent bits = 1).
- NaN ("Not a Number") (all 23 mantissa bits ≠ 0, all exponent bits = 1).
- Negative Zero (s = 1, all 23 mantissa bits = 0, all exponent bits = 0).
- Positive Zero (s = 0, all 23 mantissa bits = 0, all exponent bits = 0).

- $(2^8 - 1) * 2^{23}$-1 = 2,139,095,039 positive numbers
- $(2^8 - 1) * 2^{23}$-1 = 2,139,095,039 negative numbers

3. What is the largest single precision number?

To find the largest, set the sign bit to 0, set all the mantissa bits to 1, and the exponent bits are set to 11111110 (i.e. the bit string 01111111-01111111-11111111-11111111= 0x7F7FFFFF). By using the above equation, we get the following largest precision number:

$(-1)^0 * 1.9999998808 * (2)^{254-127} = 1.9999998808 * 2^{127} = 3.402823466 * 10^{38}$

The largest single precision number is: $\mathbf{1.9999998808 * 2^{127} = 3.402823466 * 10^{38}}$

4. What is the smallest single precision number?

To find the smallest single precision number, we use a similar method done in question 3, but instead of using 0 as a signed bit, we use 1. To find the smallest, set the sign bit to 1, set all the mantissa bits to 1, and the exponent bits are set to 11111110 (i.e. the bit string 11111111-01111111-11111111-11111111= 0xFF7FFFFF).

By using the above equation, we get the following largest precision number:

$(-1)^1 * 1.9999998808 * (2)^{254-127} = -1.9999998808 * 2^{127} = -3.402823466 * 10^{38}$

The largest single precision number is: $\mathbf{-1.9999998808 * 2^{127} = -3.402823466 * 10^{38}}$

5. What single precision numbers have the smallest non-zero absolute value?

There are "two" smallest non-zero absolute values: a "normal" and "denormal."

- The minimum positive **denormal** value occurs with the following:

Number $=(-1)^0 * 0.00000000000000000000001 * 2^{-126} = 2^{-149}$
The "minimum positive **(denormal)** value" is $2^{-149} \approx 1.401298 * 10^{-45}$.

- The minimum positive **normal** value occurs with the following:

This value occurs when have a zero signed bit, all mantissa bits to 0, and the exponent bits are set to 00000001 (i.e. the bit string 00000000-10000000-00000000-00000000= 0x00800000).
$(-1)^0 * 1.0000000000 * (2)^{1-127} = 2^{-126} \approx 1.1754943508 * 10^{-38}$
The "minimum positive **(normal)** value" is $2^{-126} \approx 1.1754943508 * 10^{-38}$

6. Explain why the highest precision occurs around the value of 1 (and -1)?

The highest precision occurs around the value of 1 (and -1) because 1 is the "base" used, meaning that to get a smaller number we need to decrease the exponent, which will distribute the step size. In contrast, to get a larger number, we need to increase the exponent, which will also distribute the step size. In other words, since we either increase or decrease the exponent to get a

bigger or smaller number, the step size is distributed, thus we achieve better precision around the base. Therefore, high precision occurs around the values of 1 and -1.