Data Structures and Algorithms

Name : Mohit Kumar Date : January 10, 2022

SR No. : 04-01-03-10-51-21-1-19825

Assignment 1

Question 1 : Consider these two float approximations for the value of π – (i) 22/7, (ii) 0x40490FDB. In which binary position do they begin to differ? In which decimal digit position do they begin to differ?

Q1.c is the C program for this question.

32 bit IEEE 754 representation corresponding to 22/7 is given by

0 10000000 10010010010010010010010

which in decimal is 3.142857

32 bit IEEE 754 representation corresponding to 0x40490FDB is given by

0 10000000 10010010000111111011011

which in decimal is 3.141593

As can also be seen from the output of Q1.c, these two float approximations for the value of π differ in the 19th binary position from the left and 3rd decimal position to the right of the decimal point.

Question 2 : Assuming that real data is represented using IEEE single (double) precision floating point representation, calculate and explain how many times this C while loop will iterate. Verify your answer experimentally.

float f=1.0;

while (f != 0.0) f = f / 2.0;

Q2.c is the C program for this question.

Assuming that real data real data is represented using IEEE single precision floating point representation for float data type in C, we see through the program in Q2.c that the while loop runs for 150 iterations. The reason for this is as follows:

32 bit IEEE 754 representation of 1.0 is given by

0 01111111 00000000000000000000000

When we iteratively divide 1.0 by 2.0, the exponent field in the above representation will keep on decreasing from 127 to 0 which will account for 127 iterations. After the exponent field becomes 0, the value is a denormalized number, which now has an assumed leading 0 before the binary point. Thus, this represents a number $(-1)^s \times 0.f \times 2^{-126}$, where s is the sign bit and f is the fraction. Continuing to divide by 2.0 now the bits in the fractional part in the above representation will now start to become 0 from the left. As the fractional part contains 23 bits. The while loop will run for an additional 23 iterations after which the fractional part in the above representation will become 0. When this happens the condition of the while loop will become false as f will be equal to 0.0 and the while loop will terminate.

Question 3 : Consider the real valued sum of the harmonic sequence

$$\sum_{i=1}^{n} 1/i$$

Write C functions that compute the float value by iterating (i) forwards (from i=1 up to i=n), and (ii) backwards (from i=n down to i=1). Observe that the sum depends on the order of summation (i.e., backward vs forward) for some values of n? Explain why this happens.

Q3.c is the C program for this question. We see that for some values of n, for example

```
Enter the value of n : 5

Forward sum = 2.283334

Backward sum = 2.283333
```

The reason for this is as follows:

Using IEEE 754 format for representing real values, we can only represent numbers of the form $y + x/2^i$. Other fractions (rationals) have repeating bit representations, for eg, 1/5 is represented as $0.001100110011[0011]_2$ .Floating Point Addition is in general not associative, i.e. (a + b) + c is not equal to a + (b + c) because of overflow and inexactness of rounding. That is why the forward sum and backward sum are not equal.