고속 역제곱근 알고리즘

한남대학교 수학과

20172581 김남훈

1. 고속 역제곱근 알고리즘

고속 역 제곱근 알고리즘은 UC버클리 수학과 교수인 윌리엄 카한이 1986년 아이디어를 제시하고, 그 아이디어에 기반하여 실리콘 그래픽스 사가 90년대 초 개발한 알고리즘으로, 간단히 설명하면 실수 a 에 대해 $\frac{1}{\sqrt{a}}$ 를 빠르게 계산하는 알고리즘이다. 우선 코드를 보면 다음과 같다.

```
float Q_rsqrt( float number )
{
    long i;
    float x2, y;
    const float threehalfs = 1.5F;

    x2 = number * 0.5F;
    y = number;
    i = * ( long * ) &y;                       // evil floating point bit level hacking
    i = 0x5f3759df - ( i >> 1 );               // what the fuck?
    y = * ( float * ) &i;
    y = y * ( threehalfs - ( x2 * y * y ) ); // 1st iteration
//  y = y * ( threehalfs - ( x2 * y * y ) ); // 2nd iteration, this can be removed

    return y;
}
```

Figure 1: 출처 : https://en.wikipedia.org/wiki/Fast_inverse_square_root

위 코드는 1999년 발매된 게임 **퀘이크 3 아레나** 의 소스코드 중 일부로, 유명 개발자인 존 카멕이 작성했다고 알려져 있으며, 주석에는 이해할 수 없는 코드에 대한 프로그래머의 당혹감이 담겨 있다. 이 코드는 다음과 같이 작동한다.

1. 주어진 실수 number 을 2 로 나눈 것을 x_2 라 한다.
2. number 을 컴퓨터에게 정수로 인식시킨 것을 i 라 한다.
3. 16진수 5f3759df(10진수로는 1597463007) 에서 $\frac{i}{2}$ 를 뺀 값을 다시 i 로 저장한다.
4. 이제 다시 i 를 컴퓨터에게 실수로 인식시킨 것을 y 라 한다.
5. $y\left[\frac{3}{2} - (x_2 y^2)\right]$ 를 반환한다.

위 함수가 반환하는 값이 어째서 $\frac{1}{\sqrt{\text{number}}}$ 인지 이해하기 어렵지만, 이 알고리즘에는 신기한 수학적 원리가 숨어 있다. 원리를 알아보자.

2. 컴퓨터가 숫자를 저장하는 방법

컴퓨터에 숫자가 저장되는 방법은 크게 두 가지, 정수형과 실수형으로 나눌 수 있다. 컴퓨터 메모리는 0 과 1 만을 저장할 수 있으므로, 정수형과 실수형 모두 기본적으로 이진법으로 저장된다.

정수형

정수형은 주어진 정수를 다음과 같이 32 자리 또는 64 자리의 이진법으로 저장한다. 이 때, 첫 번째 비트는 수의 부호(0 이면 양수, 1 이면 음수) 를 나타내며, 뒤의 비트들은 이진수의 각 자릿수를 나타낸다.

$$537 = 0\ \ 0000000\ \ 00000000\ \ 00000010\ \ 000011001_{(2)}$$

실수형

실수형은 다른 말로 부동소수점(floating point)이라고 한다.

수학과 학생들은 다음과 같은 표현에 익숙할 것이다.

$$373.884 = 3.73884 \times 10^2$$

여기서 3141592 을 **유효숫자**, 0 을 **지수** 라고 한다. 컴퓨터가 실수를 저장하는 방법도 동일하다.

$$\begin{aligned} 373.884 &= 101110101.111000100100111011101_{(2)} \\ &= 1.011101011110001001001110111 01_{(2)} * 2^8 \end{aligned}$$

현대의 표준 규격은 부호를 나타내는 1 비트, 지수를 나타내는 8 비트, 유효숫자를 나타내는 23 비트로 총 32 비트로 실수를 저장한다. 표준 규격대로 위의 수를 저장하면

$$373.884 \simeq 0\ \underbrace{10000111}_{8}\ \underbrace{01110101111000100100110}_{23}{}_{(2)}$$

가 된다. 여기서, 지수가 $00001000_{(2)}$ 이 아니라 $10000111_{(2)}$ 인 이유는, 지수에 $2^8 - 1 = 127$ 을 더해 저장하기 때문이다. 이러한 방식의 장점은 1 보다 작은 실수도 저장할 수 있다는 것이다. 실제로 32 비트 부동소수점 방식은 10^{-38}~10^{38} 사이의 실수를 저장할 수 있다.

2. 저장된 수를 다른 형식으로 인식시켰을 때 일어나는 일

고속 역제곱근 알고리즘에는 실수형 데이터를 정수형으로 인식시키는 과정과, 반대로 정수형 데이터를 실수형으로 인식시키는 과정이 존재한다. 이 과정은 이 알고리즘에서 가장 이해하기 어려운 부분 중 하나로, 수학적으로는 다음과 같이 설명할 수 있다.

실수형 수를 다음과 같이 s, a, b 로 나누어 생각하자.

$$x = s\ \underbrace{y_1 \ldots y_8}_{a}\ \underbrace{x_1 \ldots x_{23}}_{b}\ {}_{(2)}$$

이제 이 데이터를 실수로 인식시킨 값을 y 라고 하면, 다음과 같다.

$$\begin{aligned} x &= \left[1 + b \times 2^{-23}\right] \times 2^{a-127} \\ y &= a \times 2^{23} + b \end{aligned}$$

반대로 x 로부터 a, b 를 계산하면

$$\begin{aligned} a &= \lfloor \log_2 x \rfloor + 127 \\ b &= 2^{\log_2 x - \lfloor \log_2 x \rfloor} \end{aligned}$$

이다. 또한, a, b 는 각각 y 를 2^{23} 으로 나눈 몫과 나머지이다. 따라서 x 와 y 를 서로에 대한 함수로 나타낼 수 있다.

$$\begin{aligned} x &= \left[1 + (y \bmod 2^{23}) \times 2^{-23}\right] \times 2^{y \div 2^{23} - 127} \\ y &= \left[\lfloor \log_2 x \rfloor + 127\right] * 2^{23} + 2^{\log_2 x - \lfloor \log_2 x \rfloor} \end{aligned}$$