# 강의 12: 수 II

## 강의 개요

- 복습:
  - 고정밀도 계산
  - 곱셈
- 나눗셈
  - 알고리즘
  - 오차 분석
- 결론

## 복습:

$\sqrt{2}$ 를 백만 자릿수까지 계산 :

$$\lfloor\sqrt{2\cdot 10^{2d}}\rfloor \qquad d = 10^6$$

$\lfloor\sqrt{a}\rfloor$ 를 뉴튼 방법을 이용하여 계산

$$\begin{aligned}\chi_0 &= 1 \quad \text{(initial guess)}\\ \chi_{i+1} &= \frac{\chi_i + a/\chi_i}{2} \quad \leftarrow \text{division!}\end{aligned}$$

## 뉴턴법의 오차 분석

Suppose $X_n = \sqrt{a}\cdot(1+\epsilon_n)$ $\quad \epsilon_n$ may be + or -

Then,

$$\begin{aligned}X_{n+1} &= \frac{X_n + a/X_n}{2}\\ &= \frac{\sqrt{a}(1+\epsilon_n) + \frac{a}{\sqrt{a}(1+\epsilon_n)}}{2}\\ &= \sqrt{(a)}\frac{\left((1+\epsilon_n) + \frac{1}{(1+\epsilon_n)}\right)}{2}\\ &= \sqrt{(a)}\left(\frac{2+2\epsilon_n+{\epsilon_n}^2}{2(1+\epsilon_n)}\right)\\ &= \sqrt{(a)}\left(1+\frac{{\epsilon_n}^2}{2(1+\epsilon_n)}\right)\end{aligned}$$

Therefore,

$$\epsilon_{n+1} = \frac{{\epsilon_n}^2}{2(1+\epsilon_n)}$$

매 단계마다 정확도가 두 배가 되므로 이차적 수렴이라고 할 수 있다.

뉴튼 방법은 고정밀 나눗셈을 필요로 한다. 12강에서 곱셈을 다뤘다.

## 곱셈 알고리즘:

1. 기본 분할 정복 알고리즘: $\Theta(d^2)$ 시간
2. 카라추바 알고리즘: $\Theta(d^{\log_2 3}) = \Theta(d^{1.584...})$
3. Toom-Cook 알고리즘이 카라추바 알고리즘을 일반화함. (k $\geq$ 2 인 k 부분으로 쪼개는 경우)

$T(d) = 5T(d/3) + \Theta(d) = \Theta(d^{\log_3 5}) = \Theta(d^{1.465...})$

4. Schönhage-Strassen - 거의 선형 시간에 가까움! $\Theta(d \lg d \lg\lg d)$ FFT(고속 푸리에 변환)을 사용. 여기까지는 모두 gmpy 패키지에 구현되어 있음.
5. Furer (2007): $\Theta(n \log n\, 2^{2\log^* n})$ log* n은 반복 로그라고 부르는데, 로그값의 결과가 1보다 작거나 같을 때까지 반복해서 로그를 계산해야 하는 횟수를 의미한다.

## 고정밀 나눗셈

### $\frac{a}{b}$ 의 고정밀 표현을 계산해야 하는 경우

- $\frac{1}{b}$ 의 고정밀 표현을 먼저 계산
- $\frac{1}{b}$ 의 고정밀 표현은 R이 큰 값인 경우에 $\left[\frac{R}{b}\right]$ 을 r로 나눈 결과라고 할 수 있다.

  예시: $R = 2^k$ 인 경우 2진수의 나눗셈을 쉽게 할 수 있다

## 나눗셈

### $\frac{R}{b}$ 을 계산하기 위한 뉴튼 방법

$$\begin{aligned} f(x) &= \frac{1}{x} - \frac{b}{R} \quad \left(\text{zero at } x = \frac{R}{b}\right) \\ f'(x) &= \frac{-1}{x^2} \\ \chi_{i+1} &= \chi_i - \frac{f(\chi_i)}{f'(\chi_i)} = \chi_i - \frac{\left(\frac{1}{\chi_i} - \frac{b}{R}\right)}{-1/\chi_i{}^2} \\ \chi_{i+1} &= \chi_i + \chi_i{}^2\left(\frac{1}{\chi_i} - \frac{b}{R}\right) = 2\chi_i - \frac{b\chi_i{}^2 \rightarrow \text{ multiply}}{R \rightarrow \text{ easy div}} \end{aligned}$$

## 예시

$$\text{Want } \frac{R}{b} = \frac{2^{16}}{5} = \frac{65536}{5} = 13107.2$$

$$\text{Try initial guess } \frac{2^{16}}{4} = 2^{14}$$

$$\begin{aligned} \chi_0 &= 2^{14} = 16384 \\ \chi_1 &= 2 \cdot (16384) - 5(16384)^2/65536 = \underline{12}288 \\ \chi_2 &= 2 \cdot (12288) - 5(12288)^2/65536 = \underline{13}056 \\ \chi_3 &= 2 \cdot (13056) - 5(13056)^2/65536 = \underline{13107} \end{aligned}$$

## 오차 분석

$$\begin{aligned} \chi_{i+1} &= 2\chi_i - \frac{b\chi_i{}^2}{R} \qquad \text{Assume } \chi_i = \frac{R}{b}(1+\epsilon_i) \\ &= 2\frac{R}{b}(1+\epsilon_i) - \frac{b}{R}\left(\frac{R}{b}\right)^2(1+\epsilon_i)^2 \\ &= \frac{R}{b}\left((2+2\epsilon_i) - (1+2\epsilon_i+\epsilon_i{}^2)\right) \\ &= \frac{R}{b}(1-\epsilon_i{}^2) = \frac{R}{b}(1+\epsilon_{i+1}) \text{ where } \epsilon_{i+1} = -\epsilon_i{}^2 \end{aligned}$$

각 단계마다 자릿수가 두 배로 증가하므로 이차적 수렴이다.

나눗셈의 복잡도는 lg d 에 곱셈의 복잡도를 곱한 값이라고 생각할 수 있다. lg d는 d 자릿수 정확도를 만들기 위해서 필요한 곱셈의 반복 횟수이기 때문이다. 하지만

실제로는 나눗셈의 복잡도는 곱셈의 복잡도와 동일하다.

먼저 $\alpha \geq 1$ 에 대해서 n 자릿수 곱셈의 복잡도를 $\Theta(n^{\alpha})$ 이라고 한다면. 나눗셈은 각 반복마다 *다른 크기*의 자릿수를 가지는 숫자를 곱하는 것을 필요로 한다. 초기에는 작은 자릿수의 숫자를 계산하지만, 갈수록 그 숫자는 길어져서 $d$ 자릿수를 계산하게 된다.

따라서 나눗셈에 사용되는 연산의 횟수를 계산하면 다음과 같은 형태가 된다. :

$$c \cdot 1^{\alpha} + c \cdot 2^{\alpha} + c \cdot 4^{\alpha} + \cdots + c \cdot \left(\frac{d}{4}\right)^{\alpha} + c \cdot \left(\frac{d}{2}\right)^{\alpha} + c \cdot d^{\alpha} < 2c \cdot d^{\alpha}$$

## 제곱근 계산의 복잡도

f(x) = $x^2 - a$ 의 해를 찾기 위해서 가장 먼저 뉴턴법의 첫 번째 단계를 적용한다. 첫 번째 단계의 각 반복에서는 나눗셈을 필요로 한다. 처음부터 정확도를 $d$ 자릿수로 설정해 놓으면, 첫 번째 단계에서 필요한 반복은 $lg\ d$번이 된다. 이것은 곱셈의 복잡도가 $\Theta(d^{\alpha})$ 일 때, 우리는 나눗셈의 복잡도가 곱셈의 복잡도와 같다는 것을 보였으므로 제곱근 계산이 가지는 복잡도는 $\Theta(d^{\alpha} lgd)$ 가 된다.

첫 번째 단계에서 뉴튼 방법을 적용할 때 필요로 하는 자릿수의 정확도는 작은 상태에서 점점 커진다는 것을 깨닫는다면, 더 잘할 수 있다. 여기서 $d$자릿수 나눗셈을 계산할 때 복잡도가 $\Theta(d^{\alpha})$ 라고 한다면, 결국 제곱근을 계산하는데 걸리는 시간에 대한 복잡도도 $\Theta(d^{\alpha})$ 라고 할 수 있다.

## 결론

Iteration: $\chi_{i+1} = \lfloor \frac{\chi_i + \lfloor a/\chi_i \rfloor}{2} \rfloor$
Do floors hurt? Does program terminate? ($\alpha$ and $\beta$ are the fractional parts below.)
Iteration is

$$\begin{aligned} \chi_{i+1} &= \frac{\chi_i + \frac{a}{\chi_i} - \alpha}{2} - \beta \\ &= \frac{\chi_i + \frac{a}{\chi_i}}{2} - \gamma \quad \text{where } \gamma = \frac{\alpha}{2} + \beta \text{ and } 0 \leq \gamma < 1 \end{aligned}$$

Since $\frac{a+b}{2} \geq \sqrt{ab}$, $\frac{\chi_i + \frac{a}{\chi_i}}{2} \geq \sqrt{a}$, so subtracting $\gamma$ always leaves us $\geq \lfloor \sqrt{a} \rfloor$. This won't stay stuck above if $\epsilon_i < 1$ (good initial guess).