TDDD95 Algorithmic Problem Solving

Le 3 – Arithmetic

Fredrik Heintz
Dept of Computer and Information Science
Linköping University

Overview

- **Arithmetic**
- **Arbitrarily big integers (BigInt)**
- **Integer multiplication** with Karatsuba (*Lab 1.6*)
- **Multiplication of polynomials** with FFT (*Lab 1.6*)
- **Linear equations** – Gaussian Elimination (*Lab 1.7-1.8*)
- Other methods
  - **Segment tree** for finding all intervals that contain a query point

Arithmetic

- Range of default integer data types (C++)
  - unsigned int = unsigned long: 2^{32} (9-10 digits)
  - unsigned long long: 2^{64} (19-20 digits)
  - uint128_t (almost 40 digits)
- Operations on Big Integer
  (free in e.g. Java and Python, has to be implemented in C++)
  - Basic: add, subtract, multiply, divide, etc.
  - Use "high school methods".

```
 1  ← carry
218
 45
--- +
263
```

Arithmetic

- Greatest Common Divisor (Euclidean Algorithm)
  - GCD(a, 0) = a
  - GCD(a, b) = GCD(b, a mod b)
    - // Exercise: Prove this!
  - `int gcd(int a, int b) { return (b == 0 ? a : gcd(b, a % b)); }`
- Least Common Multiplier
  - LCM(a, b) = (a*b) / GCD(a, b)
  - `int lcm(int a, int b) { return (a / gcd(a, b)) * b; }`
    - // Why is it good practice to write the lcm code this way?
- GCD/LCM of more than 2 numbers:
  - GCD(a, b, c) = GCD(a, GCD(b, c))

Arithmetic

- Representing rational numbers.
  - Pairs of integers a,b where GCD(a,b) = 1.
- Representing rational numbers modulo m.
  - The only difficult operation is inverse, ax = 1 (mod m), where an inverse exists if and only if a and m are co-prime (gcd(a,m)=1).
  - Can be found using the Extended Euclidean Algorithm
    ax = 1 (mod m) => ax – 1 = qm => ax – qm = 1
    (d, x, y) = EGCD(a,m) => x is the solution iff d = 1.

Karatsuba's algorithm (Lab 1.6)

- Using the classical pen and paper algorithm two *n* digit integers can be multiplied in $O(n^2)$ operations. Karatsuba came up with a faster algorithm.
- Let A and B be two integers with
  - $A = A_1 10^k + A_0,\ A_0 < 10^k$
  - $B = B_1 10^k + B_0,\ B_0 < 10^k$
  - $C = A*B = (A_1 10^k + A_0)(B_1 10^k + B_0)$

    $= A_1 B_1 10^{2k} + (A_1 B_0 + A_0 B_1) 10^k + A_0 B_0$

  Instead this can be computed with 3 multiplications
  - $T_0 = A_0 B_0$
  - $T_1 = (A_1 + A_0)(B_1 + B_0)$
  - $T_2 = A_1 B_1$
  - $C = T_2 10^{2k} + (T_1 - T_0 - T_2) 10^k + T_0$

Karatsuba's algorithm (Lab 1.6)

- Compute 1234 * 4321
- Subproblems:
  - $a_1 = 12 * 43$
  - $d_1 = 34 * 21$
  - $e_1 = (12 + 34) * (43 + 21) – a_1 – d_1 = 46 * 64 – a_1 – d_1$
- Need to recurse…
- First subproblem: $a_1 = 12 * 43$
  - $a_2 = 1 * 4 = 4$; $d_2 = 2 * 3 = 6$; $e_2 = (1+2)(4+3) – a_2 – d_2 = 11$
  - Answer: $4 * 10^2 + 11 * 10^1 + 6 = 516$
- Second subproblem $d_1 = 34 * 21$
  - Answer: $6 * 10^2 + 11 * 10^1 + 4 = 714$
- Third subproblem: $e_1 = 46 * 64 – a_1 – d_1$
  - Answer: $4 * 10^2 + 52 * 10^1 + 24 - 714 - 516 = 1714$
- Final Answer:
  - $1234 * 4321 = 516 * 10^4 + 1714 * 10^2 + 714 = 5{,}332{,}114$

Complexity of Karatsuba's Algorithm

- Let T(n) be the time to compute the product of two n-digit numbers using Karatsuba's algorithm.
  Assume $n = 2^k$. $T(n) = \Theta(n^{\lg(3)})$, $\lg(3) \approx 1.58$

- $T(n) \leq 3T(n/2) + cn$

$$\leq 3(3T(n/4) + c(n/2)) + cn = 3^2T(n/2^2) + cn(3/2 + 1)$$

$$\leq 3^2(3T(n/2^3) + c(n/4)) + cn(3/2 + 1)$$

$$= 3^3T(n/2^3) + cn(3^2/2^2 + 3/2 + 1)$$

…

$$\leq 3^iT(n/2^i) + cn(3^{i-1}/2^{i-1} + \ldots + 3/2 + 1)$$

…

$$\leq c3^k + cn[((3/2)^k - 1)/(3/2 - 1)] \quad \text{--- Assuming } T(1) \leq c$$

$$\leq c3^k + 2c(3^k - 2^k) \leq 3c3^{\lg(n)} = 3cn^{\lg(3)}$$

Fast Fourier Transform (Lab 1.6)

- See separate presentation.

A system of linear equations can be presented in different forms

$$\left.\begin{aligned} 2x_1 + 4x_2 - 3x_3 &= 3 \\ 2.5x_1 - x_2 + 3x_3 &= 5 \\ x_1 \qquad\qquad - 6x_3 &= 7 \end{aligned}\right\} \Leftrightarrow \begin{bmatrix} 2 & 4 & -3 \\ 2.5 & -1 & 3 \\ 1 & 0 & -6 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 3 \\ 5 \\ 7 \end{bmatrix}$$

Standard form $\qquad\qquad\qquad\qquad$ Matrix form

Solutions of Linear Equations

$$\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 1 \\ 2 \end{bmatrix} \text{ is a solution to the following equations:}$$

$$x_1 + x_2 = 3$$

$$x_1 + 2x_2 = 5$$

Solutions of Linear Equations

- A set of equations is **inconsistent** if there exists no solution to the system of equations:

$$x_1 + 2x_2 = 3$$

$$2x_1 + 4x_2 = 5$$

$$\text{These equations are inconsistent}$$

Solutions of Linear Equations

- Some systems of equations may have **infinite number of solutions**

$$x_1 + 2x_2 = 3$$
$$2x_1 + 4x_2 = 6$$

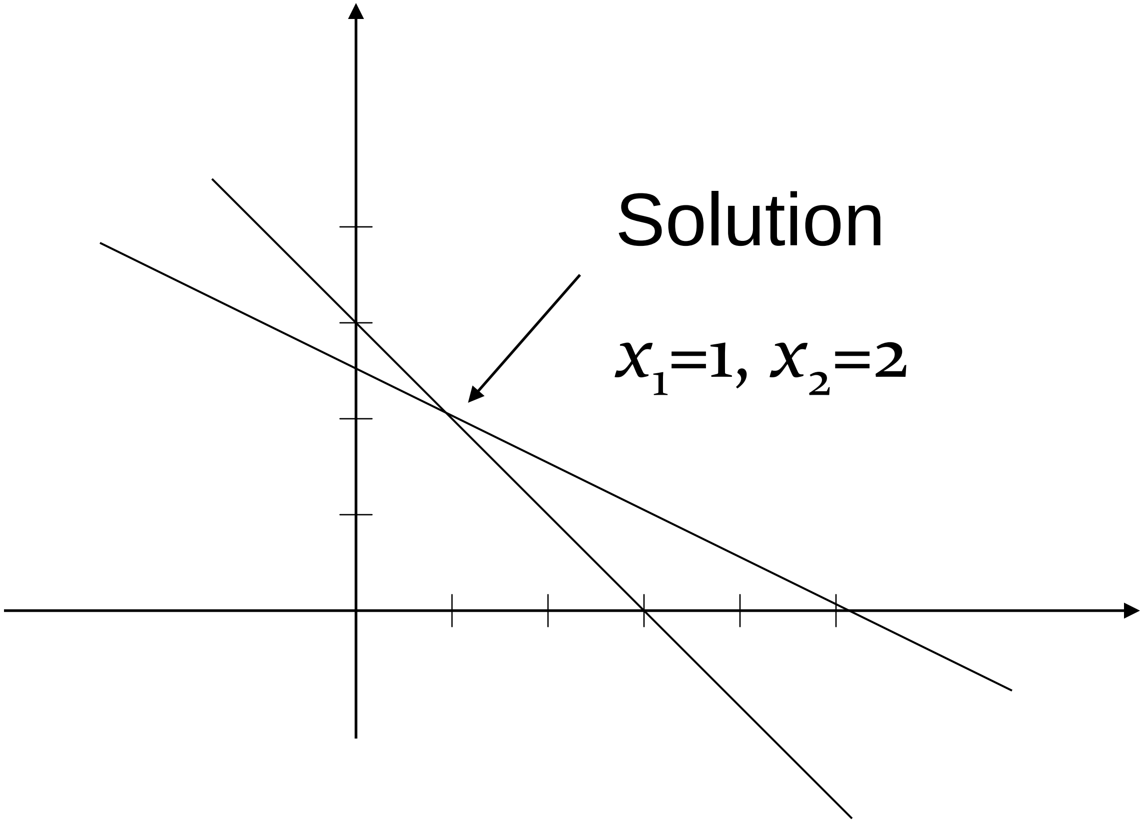
have infinite number of solutions

$$\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} a \\ 0.5(3-a) \end{bmatrix} \text{ is a solution for all } a$$

Graphical Solution of Systems of Linear Equations

$$x_1 + x_2 = 3$$

$$x_1 + 2x_2 = 5$$



Cramer's Rule is Not Practical

Cramer's Rule can be used to solve the system

$$x_1 = \frac{\begin{vmatrix} 3 & 1 \\ 5 & 2 \end{vmatrix}}{\begin{vmatrix} 1 & 1 \\ 1 & 2 \end{vmatrix}} = 1, \qquad x_2 = \frac{\begin{vmatrix} 1 & 3 \\ 1 & 5 \end{vmatrix}}{\begin{vmatrix} 1 & 1 \\ 1 & 2 \end{vmatrix}} = 2$$

Cramer's Rule is not practical for large systems.

To solve N by N system requires $(N+1)(N-1)N!$ multiplications.

To solve a 30 by 30 system, 2.38×10^{35} multiplications are needed.

It can be used if the determinants are computed in efficient way

Naive Gaussian Elimination

- The method consists of two steps:
  - **Forward Elimination**: the system is reduced to upper triangular form. A sequence of elementary operations is used.
  - **Backward Substitution**: Solve the system starting from the last variable.

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix} \Rightarrow \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ 0 & a_{22}' & a_{23}' \\ 0 & 0 & a_{33}' \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2' \\ b_3' \end{bmatrix}$$

Elementary Row Operations

- Adding a multiple of one row to another.
- Swap two rows.
- Multiply any row by a non-zero constant.

Example: Forward Elimination

$$\begin{bmatrix} 6 & -2 & 2 & 4 \\ 12 & -8 & 6 & 10 \\ 3 & -13 & 9 & 3 \\ -6 & 4 & 1 & -18 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} 16 \\ 26 \\ -19 \\ -34 \end{bmatrix}$$

Part 1: Forward Elimination

Step 1: Eliminate x_1 from equations 2, 3, 4

$$\begin{bmatrix} 6 & -2 & 2 & 4 \\ 0 & -4 & 2 & 2 \\ 0 & -12 & 8 & 1 \\ 0 & 2 & 3 & -14 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} 16 \\ -6 \\ -27 \\ -18 \end{bmatrix}$$

Example: Forward Elimination

Step2 : Eliminate x_2 from equations 3, 4

$$\begin{bmatrix} 6 & -2 & 2 & 4 \\ 0 & -4 & 2 & 2 \\ 0 & 0 & 2 & -5 \\ 0 & 0 & 4 & -13 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} 16 \\ -6 \\ -9 \\ -21 \end{bmatrix}$$

Step3 : Eliminate x_3 from equation 4

$$\begin{bmatrix} 6 & -2 & 2 & 4 \\ 0 & -4 & 2 & 2 \\ 0 & 0 & 2 & -5 \\ 0 & 0 & 0 & -3 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} 16 \\ -6 \\ -9 \\ -3 \end{bmatrix}$$

Example: Forward Elimination

Summary of the Forward Elimination:

$$\begin{bmatrix} 6 & -2 & 2 & 4 \\ 12 & -8 & 6 & 10 \\ 3 & -13 & 9 & 3 \\ -6 & 4 & 1 & -18 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} 16 \\ 26 \\ -19 \\ -34 \end{bmatrix} \Rightarrow \begin{bmatrix} 6 & -2 & 2 & 4 \\ 0 & -4 & 2 & 2 \\ 0 & 0 & 2 & -5 \\ 0 & 0 & 0 & -3 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} 16 \\ -6 \\ -9 \\ -3 \end{bmatrix}$$

Example: Backward Substitution

$$\begin{bmatrix} 6 & -2 & 2 & 4 \\ 0 & -4 & 2 & 2 \\ 0 & 0 & 2 & -5 \\ 0 & 0 & 0 & -3 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} 16 \\ -6 \\ -9 \\ -3 \end{bmatrix}$$

Solve for x_4, then solve for x_3,... solve for x_1

$$x_4 = \frac{-3}{-3} = 1, \qquad x_3 = \frac{-9+5}{2} = -2$$

$$x_2 = \frac{-6-2(-2)-2(1)}{-4} = 1, \quad x_1 = \frac{16+2(1)-2(-2)-4(1)}{6} = 3$$

Determinant

The elementary operations do not affect the determinant

Example:

$$A = \begin{bmatrix} 1 & 2 & 3 \\ 2 & 3 & 2 \\ 3 & 1 & 2 \end{bmatrix} \xrightarrow{\text{Elementary operations}} A' = \begin{bmatrix} 1 & 2 & 3 \\ 0 & -1 & -4 \\ 0 & 0 & 13 \end{bmatrix}$$

$$\det(A) = \det(A') = -13$$

How Many Solutions Does a System of Equations AX=B Have?

Unique	No solution	Infinite
$\det(A) \neq 0$	$\det(A) = 0$	$\det(A) = 0$
reduced matrix has no zero rows	reduced matrix has one or more zero rows corresponding B elements $\neq 0$	reduced matrix has one or more zero rows corresponding B elements $= 0$

How Good is the Solution?

$$\begin{bmatrix} 1 & -1 & 2 & 1 \\ 3 & 2 & 1 & 4 \\ 5 & -8 & 6 & 3 \\ 4 & 2 & 5 & 3 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 1 \\ -1 \end{bmatrix} \quad solution \quad \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} -1.8673 \\ -0.3469 \\ 0.3980 \\ 1.7245 \end{bmatrix}$$

$$\text{Residues:}\ R = \begin{bmatrix} 0.005 \\ 0.002 \\ 0.003 \\ 0.001 \end{bmatrix}$$

Other algorithms and data structures

- **Segment tree** for finding all intervals that contain a query point in $O(\log n + k)$, where k is the number of intervals.

Overview

- **Arithmetic**
- **Arbitrarily big integers (BigInt)**
- **Integer multiplication** with Karatsuba (*Lab 1.6*)
- **Multiplication of polynomials** with FFT (*Lab 1.6*)
- **Linear equations** – Gaussian Elimination (*Lab 1.7-1.8*)
- Other methods
  - **Segment tree** for finding all intervals that contain a query point