Computational number theory & algebra

- Computation & algebra have always enriched each other. Egs.

Computation enriching algebra:

- **Euclid's** gcd algorithm for integers & polynomials is a nice algebraic tool.
- **Galois'** attempt to find roots of a polynomial $f(x)$ led to founding abstract algebra.
- **Weil's** attempt to study roots of a polynomial $f(x,y)$, over finite fields, led to founding algebraic-geometry.

Algebra enriching computation:

- Many optimization problems reduce to formula satisfiability, SAT.

- SAT in turn reduces to algebraic equations.

  e.g. $(x_1 \vee x_2 \vee \bar{x}_3) \wedge (\bar{x}_2 \vee x_3)$ is satisfiable iff

$$\left.\begin{aligned} (1-y_1)(1-y_2)y_3 &= 0 \\ y_2(1-y_3) &= 0 \end{aligned}\right\} \text{ has a solution in } \mathbb{F}_2 .$$

- <u>Coding theory</u>:

  Alice wants to send a message to Bob on an erroneous channel. How should Alice efficiently encode her message?

  (The best ways are algebraic!)

- <u>Internet security</u>:

  Alice wants to send a message to Bob on an insecure channel.

  (The best ways use algebra/number theory!)

Course outline

- The above introduction motivates the following topics:

  - Fast algorithms for multiplying (or dividing) integers & polynomials.

  - Fast polynomial factorization.
    (applied in Reed-Solomon codes)

  - Lattices & short vectors.

  - Primality testing. (applied in RSA crypto.)

  - Integer factoring. (breaking RSA!)
  - Discrete logarithm.

  - Advanced topics — Elliptic curves, point counting etc.

- References are listed on the homepage.

- <u>Grading</u>:
  25% - Assignments
  30% - Mid-sem exam
  40% - End-sem exam
  5% + bonus - class participation or extra talk.

- Go to: ...nitin/teaching.html

- You will find this course interesting if:
  (1) you like computational complexity,
  (2) you like algebra or its applications in computer science.

- Basic complexity notation:

- Algorithm –
  Formally, it is a Turing machine.
  Informally, it is a routine that can be implemented on any computer.

- Time/space –
  Obvious resources. We will express them as a function of the input size $|x|$ (the bit-size).

Polynomial time

- P – the set of decision problems L that can be solved by some algorithm A in time $\le |x|^c$, for a constant c.

Exponential time

- EXP – ... problems solvable in time $\le 2^{|x|^c}$.

• <u>Randomized algorithm</u> –
The algorithm could use coin tosses but in the end the error probability should be "small".

Basic <u>algebra notation</u> :

• <u>Fields</u> are algebraic objects with addition, multiplication operations & the natural properties of –
associativity, commutativity, identity $(1,0)$, inverses.
Eg. $\mathbb{Q}$, $\mathbb{R}$, $\mathbb{C}$.

<u>Exercise</u>: $\mathbb{Z}/n\mathbb{Z}$ is a field iff n is prime.
operations are (mod n)

<u>Exercise</u>: (1) Finite fields have size p^m, for a prime p.
(2) $\exists$ <u>unique</u> finite field of size p^m (denoted $\mathbb{F}_{p^m}$).

- Rings are like fields except that we drop commutativity & inverse on the multiplication operation.

  Eg. $\mathbb{Z}$, $\mathbb{Q}[x]$, $\mathbb{H}(\mathbb{Q})$, $M_n(\mathbb{Q})$.

  polynomials quaternions matrices

- Ideal I of a ring R are useful in "dividing" the ring into smaller rings.

  I is a subring & $R \cdot I \subseteq I$, where $R \cdot I := \{r \cdot a \mid r \in R, a \in I\}$.

Exercise: $R/I := \{r + I \mid r \in R\}$ has a ring structure.

- Groups are algebraic objects with a single operation (& natural properties).

  Eg. $(\mathbb{F}, +)$, $(\mathbb{F}^*, \cdot)$, $(GL_n(\mathbb{F}), \cdot)$ for a field $\mathbb{F}$.

Exercise: $(\mathbb{F}_{p^m}^*, \cdot)$ is a cyclic group.