# AM10C Lecture 3

Note Title 9/10/2016

TODAY

- Sets, Relations, Functions
- ALGORITHMS
  - PROBLEMS
  - ALGORITHMS: Definition + Example
  - RUNNING TIME: $O(\cdot)$ notation
  - EUCLID'S ALGORITHM
  - ARITHMETIC OPERATIONS - State-of-art

---

**Relations:**

- $R \subseteq S \times T$
- if $\forall a \in S \ \exists! \ b \in T$ s.t. $(a,b) \in R$

  then $R$ is a **function**.
- We will study relations on $S$ ($R \subseteq S \times S$)

  $R$ is **equivalence** relation if

① $R$ is reflexive : $(a,a) \in R$ $\forall a \in S$

AND ② $R$ is symmetric : $(a,b) \in R \Leftrightarrow (b,a) \in R$

AND ③ $R$ is transitive : $(a,b) \in R$ &

$(b,c) \in R \Rightarrow (a,c) \in R$

Examples in PS2

- Equivalence Relations $\rightsquigarrow$ Equivalence Classes

if $R$ is an equivalence relation then

it partitions $S$ into equivalence classes.

i.e. $S = \bigcup_{\theta} S_\theta$ "$S_\theta$" equivalence class

& $S_\theta \cap S_{\theta'} = \phi$

- Equivalence Classes important in algebra.

Example:

- $S = \mathbb{Z}$
- $(a,b) \in R$ if $(a-b) = 0 \pmod 5$
- Equivalence Classes:

$$S_0 = \{a \mid a = 0 \bmod 5\}$$

$$S_1 = \{\quad 1 \quad\}$$

$$\vdots$$

$$S_4 = \{\quad 4 \quad\}$$

---

Think about

- $<$ on $\mathbb{Z}$ $((a,b) \in <$ if $a < b)$
- $\leq$ ..
- $=_{1:1}$ on $\mathbb{Z}$ $((a,b) \in =_{1:1}$ if $|a| = |b|)$

# Main Topic: Algorithms

- Algorithms solve computational problems.
- But what are problems?
- Often: function : input $\longmapsto$ output

  (e.g. $+ : (a,b) \longmapsto a+b$)
- More generally relation between input & output

  (e.g. factor: $n \longmapsto (a,b)$

  s.t. $a, b \geq 2$

  $a \cdot b = n$)

Not a function

$24 \longmapsto (2,12)$

$24 \longmapsto (4,6)$

} both acceptable

⋮

# Common Algebraic Problems

- ADDITION : $(a,b) \longmapsto a+b$
- MULTIPLICATION : $(a,b) \longmapsto a * b$
- Division : $(a,b) \longmapsto (q,r)$ s.t.

  $a = q \cdot b + r$
- GCD : $(a,b) \longmapsto g \triangleq \gcd(a,b)$
- Extended GCD : $(a,b) \longmapsto (g,s,t)$

  s.t. $g = sa + tb$

  & $g|a$, $g|b$
- PRIMALITY : $a \longmapsto 1$ if $a$ prime

  $0$ o.w.
- FACTORING : $a \longmapsto (b,c)$

  $2 \leq b, c$

  $b \cdot c = a$

# Algorithm:

- Sequence of steps (transforming input → output)
- Terminates in finite # steps
- Each Step determined completely by current state/view

---

Key parameter: Running time = # steps.

---

(We define only informally. For formal study take CS 121 / 124 / 125 ....

---

## Example: Addition

- Input = $x_0 \dots x_{n-1} \in \{0,1\}$
  $y_0 \dots y_{n-1}$
  $\left[ x = \sum x_i 2^i \quad y = \sum y_i 2^i \right]$
- Output = $z_0 \dots z_n \in \{0,1\}$ s.t. $x+y = \sum z_i 2^i$

$c_0 = 0$

- Algorithm: for $i = 0$ to $n-1$ do

$(c_i, z_i) \leftarrow$ binary sum of $x_i, y_i, c_{i-1}$

end

$z_n \leftarrow c_{n-1}$

Q: is decimal addition faster than binary addition?

Both run in time $\sim n$ for $n$ bit numbers.

Use $O(n)$ to focus on highest order growth.

## Measure Running Time

- Worst-case Complexity
- Asymptotics ( $O(\cdot)$ notation)

## Worst-case complexity

- Expect runtime to depend on input?
- How to compare?
- Partition inputs into different lengths
- Measure max-time over inputs of given length [Worst-case]
- Prefer algorithm that are faster for longer lengths. [Asymptotic]

- Algorithm A runs in time $T(n)$
  if $\forall$ input of length $n$,
  running time $\leq T(n)$

[Reduces complexity to univariate measure]

- $T(n) = O(S(n))$ if
  $\exists\ n_0$ & $c$ s.t.
  $\forall\ n \geq n_0$ $\quad T(n) \leq c \cdot S(n)$

[Asymptotic Comparison]

- Examples : $T(n) = 3n^2 - 10n + 22.3$
  $= O(n^2)$

# GCD algorithm?

- GCD $(x, y)$ // requires $x \geq y$ */

  if $y = 0$ output $x$

  else output GCD $(y, x \bmod y)$

- Assume $x \bmod y$ takes $O(n^2)$ time for $n$ bit $x$ & $y$.

- Run time analysis:

  $x, y \rightarrow y, x \bmod y \rightarrow x \bmod y,$

  $y \bmod (x \bmod y)$

Claim: $\forall\, x \geq y \quad x \bmod y \leq \frac{x}{2}$

$x$ n bits $\Rightarrow x \bmod y$ $n-1$ bits.

$\Rightarrow T(n) = c \cdot n^2 + T(n-1)$

$\Rightarrow T(n) \leq c \cdot n^3$

- Extended GCD

```
EGCD(x,y)    /* x ≥ y ; returns
                (g,s,t) */
if y=0 then return (x,1,0)
else let (q,r) ← DIVIDE(x,y)
     (g,s',t') ← EGCD(y,r)
     return (g, t', s'-t'q)
```

$\longrightarrow$

$[\; s' \cdot y + t' \cdot (x - qy) = g$

$\Leftrightarrow t'x + (s' - t'q)y = g \;]$

# Basic Problems & State-Of-Art Run Time

| Problem | Naïve | Best-Known |
|---|---|---|
| Addition | $O(n)$ | $O(n)$ |
| Multiplication | $O(n^2)$ | $O(n \log^2 n)$ |
| Division | " | " |
| GCD | $O(n^3)$ | $O(n \log^c n)$ $[c \approx 2,3]$ |
| Primality | $O(2^{n/2})$ | $n^c$ $[c \approx 6,7]$ [AKS 2003] |
| Factoring | $O(2^{n/2})$ | $2^{n^{.5}}$ ↑ Better "heuristic" times exist. |