CS 218 Design and Analysis of Algorithms

Nutan Limaye

Indian Institute of Technology, Bombay
nutan@cse.iitb.ac.in

Module 1: Basics of algorithms

Credit Structure

Course credit structure

assignments and vivas	$50 \pm 5\%$
mid-sem	15%
end-sem and viva	30%
class participation	5 ± 5 %

Office hours:	with prior appointment

- Course related announcements on Moodle and MS Teams.
- Course related discussion on MS Teams.
- Code for joining MS Teams: amqp5c6.

Course Outline

Module I Basic techniques

- Greedy algorithms.
- Divide and Conquer.
- Dynamic programming.

Module II Combinatorial optimization

- Max-flow and min-cut.
- Applications of max-flow and min-cut.
- Optimization problems, LP formulation and duality.

Module III NP: a roadblock for algorithm design?

- NP-completeness and reductions.

Module IV Mitigating NP-hardness (If time permits.)

- Approximation algorithms.
- Better-than-brute-force algorithms.

Algorithms are everywhere!

Navigation apps on our phones.

Medical imaging and disease diagnosis.

Communication apps such as Signal, Telegram, WhatsApp.

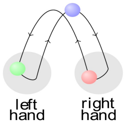
App for searching the internet such as Google Search, Bing, DuckDuckGo, etc.

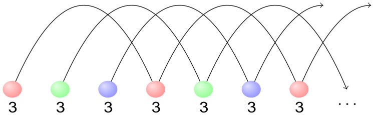
Basically we are surrounded by them!

How to study algorithms?

A short course on juggling.

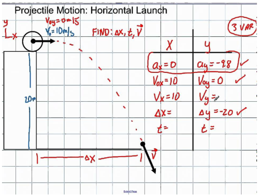
Do you know juggling?





- Cascade is a type of juggling pattern in which hands alternate throwing balls at each other resulting in a figure 8.

- The equations for projectile motion can tell us how long the juggler has to catch the ball, how high it will rise and about how far apart to keep the hands.



Now do you know juggling?

Understanding algorithms

Here input = log n

Given: a number n

Output: if n prime then output `True` else `False`.

Hence,
$O(\sqrt{n}) = 2^{O(N)}$
= expo

Consider the following simple algorithm.

flag ← `True`
for $i = 2$ to $\sqrt{n}$ **do**
  if i divides n **then**
    flag ← `False`
  end if
end for
output flag

Correctness: if a number is not a prime then it has at least one factor $\leq$ its square root.

Time complexity: Poly?

NO!

Input length $N = \log n$

Running time $= O(\sqrt{n}) = 2^{O(N)}$

It is known that the above problem can be solved in time poly$(\log n)$ = poly(N).

[Agrawal, Kayal, Saxena 2002]

Asymptotic upper bounds: Big-Oh notation

Big-Oh notation, $O(.)$.

> $T(n)$ is said to be $O(f(n))$ if there exist constants $c > 0$ and $n_0 \geq 0$ such that for all $n \geq n_0$, $T(n) \leq c \cdot f(n)$.

Examples

> Let $T(N) = N^2 + 10N$. Then $T(N) \in O(N^2)$. Also, $T(N) \in O(N^3)$.
> Is $T(N) \in O(N)$? No.
>
> Let $T(N) = 2^{\lg N}$. Is $T(N) \in O(N)$? Yes.
>
> Let $T(N) = \sqrt{N}$. Is $T(N) \in 2^{O(\log N)}$? Yes.

Asymptotic lower bounds: Omega notation

Omega notation, $\Omega(.)$.

> $T(n)$ is said to be $\Omega(f(n))$ if there exist constants $c > 0$ and $n_0 \geq 0$ such that for all $n \geq n_0$, $T(n) \geq c \cdot f(n)$.

Examples

> Let $T(N) = N^2 + 10N$. Then $T(N) \in \Omega(N^2)$. But, $T(N) \notin \Omega(N^3)$. Is $T(N) \in \Omega(N)$? Yes.

Asymptotically tight bounds: Theta notation

Theta notation, $\Theta(.)$.

> $T(n)$ is said to be $\Theta(f(n))$ if there exist constants $c, c' > 0$ and $n_0 \geq 0$ such that for all $n \geq n_0$, $c \cdot f(n) \leq T(n) \leq c' \cdot f(n)$.

Examples

> Let $T(N) = N^2 + 10N$. Then $T(N) \in \Theta(N^2)$. But, $T(N) \notin \Theta(N^3)$. Is $T(N) \in \Theta(N)$? No.