Honors Analysis of Algorithms

Instructor: Subhash Khot, off: 416

Class: T Th 9:30-10:45, WWH 317.

Office hours: TBA

TA: T. Devanathan. TA hours: TBA

Webpage

www.cs.nyu.edu/~khot/CSCI-GA.3520-001-2016.htm

Text Algorithm Design, Kleinberg-Tardos

Introduction to Algorithms, 2nd Ed, Cormen, Leiserson, Rivest, Stein.

Grading - Problem Sets 50% (6-7 sets) → Available from 2008

- Final exam 50%

CS comprehensive exam in Algorithms (PhD requirement) → Past PhD, MS exams available

NO Automata Theory, Theory of computation.

Algorithm: A systematic procedure to solve computational problem/task. E.g. sorting. Better be fast.

Computability and computational complexity

- What is "problem", "solving", "running time"?
- Is every problem solvable
  - in finite time?
  - "efficiently"/"fast"?
- How about Travelling Salesperson?

— x —

Course syllabus

① Basic algorithmic techniques + relevant data structures.

- Divide & conquer.
- Greedy algorithms.
- Dynamic Programming.
- Amortized analysis.

Specific / Advanced algorithms

- Shortest paths
- MAX. FLOW
- Randomization: hashing.

② Computability Theory.

- Turing machines
- Decidability E.g. Hilbert's 10^{th} problem, 1900.
  Negative answer in 1970.

③ Computational Complexity.

- P, NP, NP-completeness.
- Cook-Levin Theorem.

<u>Pre-requisites</u>

- Basic math
- Proof techniques
  - induction
  - proof by contradiction etc.
- 1^{st} level / introductory / breadth.
- Emphasis on <u>proofs</u>.

Asymptotic Running Time

- We are typically interested in solving problems on larger and larger instances.
- n = "size" of problem instance.

  <u>sorting</u>: n = size of list.

  <u>TSP</u> : n = number of cities

- n could be very large, e.g. data arising from genomics, astronomy, internet statistics etc.
- so useful to study behavior of algorithms on large instances.

	ALGO 1	ALGO 2
	$100\,n$	n^2
$n=10$	10^3	10^2
$n=100$	10^4	10^4
$n=10^6$	10^8	10^{12}
on computer w/ 10^6/sec steps	100 sec	$\approx$ 10 days

- $100\,n \ll n^2 \ll n^3 \cdots \ll 2^n \ll 2^{2^n} \cdots$

- "constants are not very important"
  - $100\,n$ vs n^2
  - "step" undefined, depends on model
  - on Turing m/c, "speed" can be increased by any constant factor

O, Ω, Θ notations

Assume all functions take $\mathbb{R}^+$-values.

Asymptotic upper bound : O-notation

$T(n)$ = Running time of an algo. on problem instance of size n.

$f(n)$ = A typical/concrete/standard function eg. n^2, $n \log_2 n$, 2^n etc.

Def $T(n) = O(f(n))$ ($T(n)$ is $\overset{\text{big-O of}}{\text{order}}$ $f(n)$)

if there is a constant $c > 0$ and sufficiently large integer n_0 s.t.

$$\forall n \geq n_0, \quad T(n) \leq c \cdot f(n).$$

Examples

① $100n$ is $O(n)$ "hide constants."

② $n^2+5n+100$ is $O(n^2)$.

Proof. Suppose $n \geq 1$ so that, $n \leq n^2$, $1 \leq n^2$

$$n^2+5n+100 \leq n^2+5n^2+100n^2$$
$$= 106\, n^2.$$

Take $c = 106$.

③ For any polynomial $T(n) = a_k n^k + a_{k-1} n^{k-1} + \cdots + a_1 n + a_0$

where $a_k, a_{k-1}, \ldots, a_0$ are coefficients,

$T(n)$ is $O(n^k)$.

④ n^3 is $O(2^n)$: Exercise.

⑤ $\log_2 n$ is $O(\sqrt{n})$: Exercise

O-notation is used to state upper bound on running time of an algorithm.

Theorem Sorting has an algorithm that runs in time $O(n \log n)$.

⑥ $\log_b n$ is $O(\log_2 n)$ for any $b > 1$.

Fact If $\lim_{n\to\infty} \frac{T(n)}{f(n)} \leq C$ then $T(n)$ is $O(f(n))$

Asymptotic lower bound. Ω-notation.

Def $T(n)$ is $\Omega(f(n))$ if

$\exists\ \varepsilon > 0$ and sufficiently large n_0 s.t.

$\forall\ n \geq n_0$, $T(n) \geq \varepsilon \cdot f(n)$.

Examples

① $100n$ is $\Omega(n)$.

② $n^2 + 5n + 100$ is $\Omega(n^2)$.

Lower bound.

Theorem Any (comparison based) sorting algorithm takes $\Omega(n \cdot \log n)$ time.

Def: $T(n) = \Theta(f(n))$ if

$T(n) = O(f(n))$ and $T(n) = \Omega(f(n))$,

Example ① $n^2 + 5n + 100$ is $\Theta(n^2)$

② $\sum_{i=0}^{k} a_i n^i$, $a_k > 0$ is $\Theta(n^k)$.

$\forall b > 1$, ③ $\log_b n$ is $\Theta(\log_2 n)$.

Fact If $\exists\ c > 0$ s.t. $\lim_{n \to \infty} \frac{T(n)}{f(n)} = c$.

then $T(n) = \Theta(f(n))$.

Theorem. Time complexity of (comparison based) sorting is $\Theta(n \log n)$.

Properties of asymptotic growth functions

1. If $f = O(g)$, $g = O(h)$ then $f = O(h)$.
2. If $f = \Omega(g)$, $g = \Omega(h)$ then $f = \Omega(h)$.
3. If $f = \Theta(g)$, $g = \Theta(h)$ then $f = \Theta(h)$.
4. If $f = O(h)$, $g = O(h)$ then $f+g = O(h)$.

Some common functions

$\log n \ll \log^2 n \ll n^{0.01} \ll n \ll n^2$

$\ll n^3 \cdots \ll 2^n \ll 3^n \cdots \ll 2^{n^2} \cdots \ll 2^{2^n} \cdots$