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)
- Final exam 50\%

Available from 2008

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?

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

3 Computational Complexity.
- P, NP, NP-completeness.
- Cook-Levin Theorem.

Pre-requisites
- Basic math
- Proof techniques - induction
  proof by contradiction etc.
- 1st level / introductory / breadth.
- Emphasis on proofs
Asymptotic Running Time

- We are typically interested in solving problems on larger and larger instances.

- \( n \) = "size" of problem instance.
  
  \[ \text{sorting: } n = \text{size of list.} \]
  
  \[ \text{Tsp: } n = \text{number of cities} \]

- \( n \) could be very large, e.g. data arising from genomics, astronomy, internet statistics etc.

- So useful to study behavior of algorithms on large instances.

<table>
<thead>
<tr>
<th>ALGO 1</th>
<th>ALGO 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>100 ( n )</td>
<td>( n^2 )</td>
</tr>
<tr>
<td>( n = 10 )</td>
<td>( 10^3 )</td>
</tr>
<tr>
<td>( n = 100 )</td>
<td>( 10^4 )</td>
</tr>
<tr>
<td>( n = 10^6 )</td>
<td>( 10^8 )</td>
</tr>
</tbody>
</table>

On computer w/ 100 sec \( 10^6 \) sec steps \( \approx 10 \) days

\[ 100 \cdot n \ll n^2 \ll n^3 \ll 2^n \ll 2^n \ldots \]
- "constants are not very important"
  - $100 \cdot n \ vs \ n^2$
  - "step" undefined, depends on model
- On Turing m/c, "speed" can be increased by any constant factor.

**$O$, $\Omega$, $\Theta$ notations**

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

**Asymptotic upper bound: $O$-notation**

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

$f(n) = \text{A typical/concrete/standard function}$

eg. $n^2$, $n \log_n 2^n$, $2^n$ etc.

**Def** $T(n) = O(f(n))$ ( $T(n)$ is order $f(n)$) if there is a constant $C > 0$ and sufficiently large integer $n_0$ s.t.

$\forall n \geq n_0, \ T(n) \leq C \cdot f(n)$. 
Examples

1. \( 100n \) is \( O(n) \) "hide constants."

2. \( n^2 + 5n + 100 \) is \( O(n^2) \).

   Proof. Suppose \( n \geq 1 \) so that \( 1 \leq n^2 \)

   \[
   n^2 + 5n + 100 \leq n^2 + 5n^2 + 100n^2 = 106n^2.
   \]

   Take \( C = 106 \).

3. For any polynomial \( T(n) = q_k n^k + q_{k-1} n^{k-1} + \ldots + q_1 n + q_0 \)
   where \( q_k, q_{k-1}, \ldots, q_0 \) 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. \( \Omega \)-notation.

Definition: \( T(n) \) is \( \Omega(f(n)) \) if

\[
\exists \; \varepsilon > 0 \text{ and sufficiently large } n_0 \text{ s.t. } \forall \; n \geq n_0, \; T(n) \geq \varepsilon \cdot f(n).
\]

Examples

1. \( 100n \) is \( \Omega(n) \).
2. \( n^2 + 5n + 100 \) is \( \Omega(n^2) \).

Lower bound.

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

Definition: \( T(n) = \Theta(f(n)) \) if

\[
T(n) = O(f(n)) \quad \text{and} \quad T(n) = \Omega(f(n))
\]

Example

1. \( n^2 + 5n + 100 \) is \( \Theta(n^2) \).
2. \( \sum_{i=0}^{k} a_i n^i, \quad a_k > 0 \) is \( \Theta(n^k) \).
3. \( \log_b n \) is \( \Theta(\log_2 n) \).
Fact. If $\exists \epsilon > 0$ 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

\[ \log n \ll \log^2 n \ll n^{0.01} \ll n \ll n^2 \ll n^3 \ll 2^n \ll 3^n \ll 2^{n^2} \ll 2^{2^n} \]