NPTEL MOOC,JAN-FEB 2015
Week 1, Module 5

DESIGN AND ANALYSIS OF ALGORITHMS

MADHAVAN MUKUND, CHENNAI MATHEMATICAL INSTITUTE
http://www.cmi.ac.in/~madhavan

Analysis of algorithms

- Measuring efficiency of an algorithm
  - Time: How long the algorithm takes (running time)
  - Space: Memory requirement

Time and space

- Time depends on processing speed
  - Impossible to change for given hardware
- Space is a function of available memory
  - Easier to reconfigure, augment
- Typically, we will focus on time, not space

Measuring running time

- Analysis independent of underlying hardware
  - Don’t use actual time
  - Measure in terms of “basic operations”
- Typical basic operations
  - Compare two values
  - Assign a value to a variable
- Other operations may be basic, depending on context
  - Exchange values of a pair of variables

Input size

* Running time depends on input size
    * Larger arrays will take longer to sort
* Measure time efficiency as function of input size
    * Input size n
    * Running time t(n)
* Different inputs of size n may each take a different amount of time
* Typically t(n) is worst case estimate

Example 1: Sorting

- Sorting an array with n elements
  - Naïve algorithms : time proportional to n^2
  - Best algorithms : time proportional to $n \log n$
- How important is this distinction?
- Typical CPUs process up to 10^8 operations per second
  - Useful for approximate calculations

Example 1: Sorting …

- Telephone directory for mobile phone users in India
  - India has about 1 billion = 10^9 phones
- Naïve n^2 algorithm requires 10^{18} operations
  - 10^8 operations per second $\Longrightarrow$ 10^{10} seconds
  - 2778000 hours
  - 115700 days
  - 300 years!
- Smart n log n algorithm takes only about 3 x 10^{10} operations
  - About 300 seconds, or 5 minutes

Example 2: Video game

- Several objects on screen
- Basic step: find closest pair of objects
- Given n objects, naïve algorithm is again n^2
  - For each pair of objects, compute their distance
  - Report minimum distance over all such pairs
- There is a clever algorithm that takes time n log n

Example 2: Video game ...

- High resolution monitor has 2500 x 1500 pixels
  - 3.75 million points
- Suppose we have 500,000 = 5×10^5 objects
- Naïve algorithm takes 25×10^{10} steps = 2500 seconds
  - 2500 seconds = 42 minutes response time is unacceptable!
- Smart n log n algorithm takes a fraction of a second

Orders of magnitude

- When comparing t(n) across problems, focus on orders of magnitude
  - Ignore constants
- $f(n) = n^3$ eventually grows faster than $g(n) = 5000\ n^2$
  - For small values of n, f(n) is smaller than g(n)
  - At n = 5000, f(n) overtakes g(n)
  - What happens in the limit, as n increases : asymptotic complexity

Typical functions

- We are interested in orders of magnitude
- Is t(n) proportional to log n, …, n^2 , n^3 , …, 2^n?
- Logarithmic, polynomial, exponential …

Typical functions t(n)…

Input	log n	n	n log n	n^2	n^3	2^n	n!
10	3.3	10	33	100	1000	1000	10^6
100	6.6	100	66	10^4	10^6	10^{30}	10^{157}
1000	10	1000	10^4	10^6	10^9		
10^4	13	10^4	10^5	10^8	10^{12}		
10^5	17	10^5	10^6	10^{10}			
10^6	20	10^6	10^7				
10^7	23	10^7	10^8				
10^8	27	10^8	10^9				
10^9	30	10^9	10^{10}				
10^{10}	33	10^{10}					