

NPTEL MOOC, JAN-FEB 2015  
Week 1, Module 6

# DESIGN AND ANALYSIS OF ALGORITHMS

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



# 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**



# Input size ...

- \* How do we fix input size?
- \* Typically a natural parameter
  - \* For sorting and other problems on arrays: array size
  - \* For combinatorial problems: number of objects
  - \* For graphs, two parameters: number of vertices and number of edges



# Input size ...

- \* Input size for numbers
  - \* Is  $n$  a prime?
- \* What should be the input size? Magnitude of  $n$ ?
- \* Arithmetic operations are performed digit by digit
  - \* Addition with carry, subtraction with borrow, multiplication, long division ...
  - \* Number of digits is input size
  - \* Same as  $\log_b n$  when we write  $n$  in base  $b$



# 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**



# Choice of basic operations

- \* Flexibility in identifying “basic operations”
- \* Swapping two variables involves three assignments
  - \*  $\text{tmp} \leftarrow x$
  - $x \leftarrow y$
  - $y \leftarrow \text{tmp}$
- \* Number of swaps is 3 times number of assignments
- \* If we ignore constants,  $t(n)$  is of the same order of magnitude even if swapping values is treated as a basic operation



# Worst case complexity

- \* Running time on input of size  $n$  varies across inputs
- \* Search for  $K$  in an unsorted array  $A$

```
i ← 0
while i < n and A[i] != K do
    i ← i+1
if i < n return i
else return -1
```



# Worst case complexity

- \* For each  $n$ , worst case input forces algorithm to take the maximum amount of time
  - \* If  $K$  not in  $A$ , search scans all elements
- \* Upper bound for the overall running time
  - \* Here worst case is proportional to  $n$  for array size  $n$
- \* Can construct worst case inputs by examining the algorithm



# Average case complexity

- \* Worst case may be very rare: pessimistic
- \* Compute average time taken over all inputs
- \* Difficult to compute
  - \* Average over what?
  - \* Are all inputs equally likely?
  - \* Need probability distribution over inputs



# Worst case vs average case

- \* Worst case can be unrealistic ...
- \* ... but average case is hard, if not impossible, to compute
- \* A good worst case upper bound is useful
- \* A bad worst case upper bound may be less informative
  - \* Try to “classify” worst case inputs, look for simpler subclasses