Lecture 2

Introduction to Asymptotic Notations

Recap:

- RAM Model
- Compute the time required by an algorithm

Exercise 1

For each of the following problems, give a metric for input size

1. Computing the sum of n numbers in an array
2. Computing n!
3. Merging two arrays.
4. Finding the shortest path between two cities in a state with n cities

For each of the following problems, give a metric for input size

1. Computing the sum of n numbers in an array

The number of elements in the array

2. Computing n!

The number of bits required to encode the given number n

3. Merging two arrays.

The total number of elements (from both arrays)

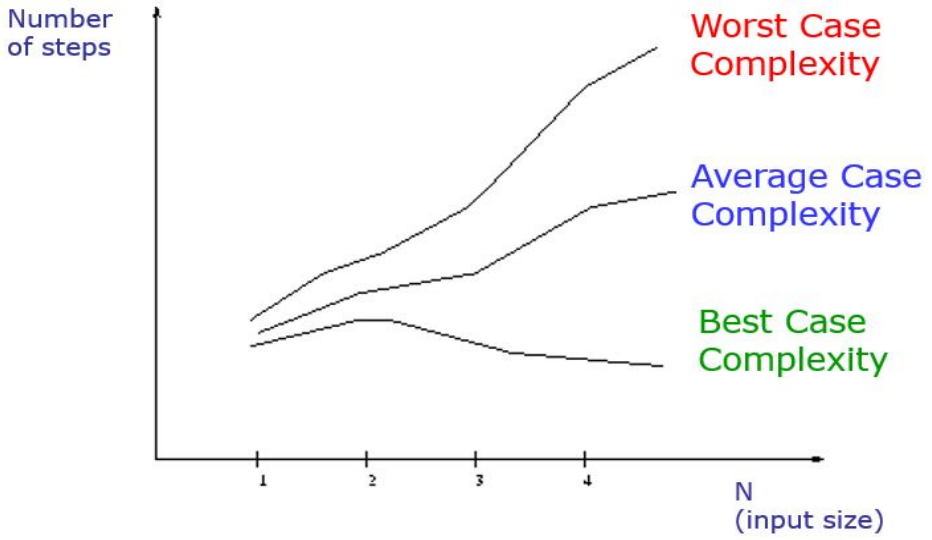
4. Finding the shortest path between two cities in a state n cities

The total number of cities in the state.

Algorithm Complexity

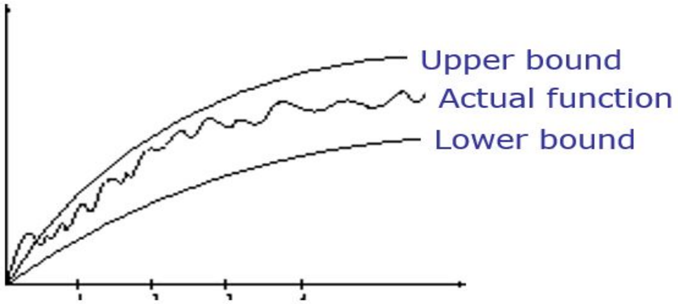
- Worst Case Complexity:
  - the function defined by the *maximum* number of steps taken on any instance of size *n*
- Best Case Complexity:
  - the function defined by the *minimum* number of steps taken on any instance of size *n*
- Average Case Complexity:
  - the function defined by the *average* number of steps taken on any instance of size *n*

Best, Worst, and Average Case Complexity



Doing the Analysis

- It's hard to estimate the running time exactly
  - Best case depends on the input
  - Average case is difficult to compute
  - So we usually focus on worst case analysis
    - Easier to compute
    - Usually close to the actual running time
- Strategy: find a function (an equation) that, for large n, is an upper bound to the actual function (actual number of steps, memory usage, etc.)



Linear Time

- We may not be able to predict to the nanosecond how long a Java program will take, but do know *some* things about timing:
  - ```
    for (i = 0, j = 1; i < n; i++) {
        j = j * i;
    }
    ```
  - This loop takes time k*n + c, for some constants k and c
  - k : How long it takes to go through the loop once
    (the time for j = j * i, plus loop overhead)
  - n : The number of times through the loop
    (we can use this as the "size" of the problem)
  - c : The time it takes to initialize the loop
  - The total time k*n + c is *linear in n*
    ```

Asymptotic Complexity

- Running time of an algorithm as a function of input size n **for large n**.
- Expressed using only the **highest-order term** in the expression for the exact running time.
  - Instead of exact running time, say $\Theta(n^2)$.
- Describes behavior of function in the limit.
- Written using ***Asymptotic Notation***.

Asymptotic Notation

- To examine an algorithm’s running time by expressing its performance as the input size, *n*, of an algorithm increases.
- There are *three* common asymptotic notations: **Big O**, **Big Theta** and **Big Omega**.

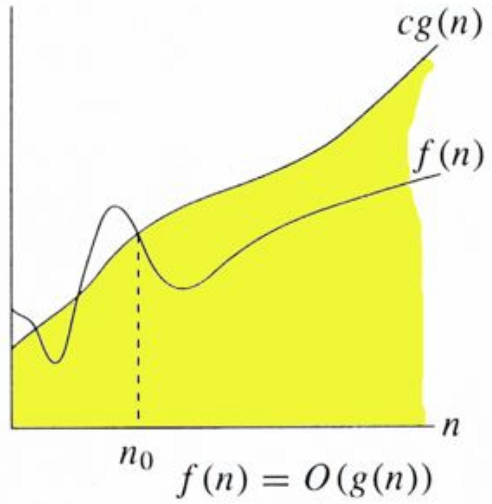
Big Oh O notation

For function $g(n)$, we define $O(g(n))$, big-O of n, as the set:

$O(g(n)) = \{f(n)$:
$\exists$ **positive constants** c **and** n_0, **such that** $\forall n \geq n_0$,

we have $0 \leq f(n) \leq cg(n)$ $\}$

Intuitively: Set of all functions whose *rate of growth* is the same as or lower than that of $g(n)$.



$g(n)$ is an *asymptotic upper bound* for $f(n)$.

$f(n) = \Theta(g(n)) \Rightarrow f(n) = O(g(n))$.
$\Theta(g(n)) \subset O(g(n))$.

Qn:1

$f(n) = n$ and $g(n) = 2n$.

Is $f(n) = O(g(n))$?

$f(n) \leq c.g(n)$ for some n_0

$n \leq c.2n$ //if $n_0 = 1$ and $c = 1$

ie this function grows linearly.

Qn:2

Prove that:

$5n^2 - 3n + 20 = O(n^2)$

$5n^2 - 3n + 20 \leq c(n^2)$ for some c

eliminating - term

$25\ n^2 \leq c\ n^2$

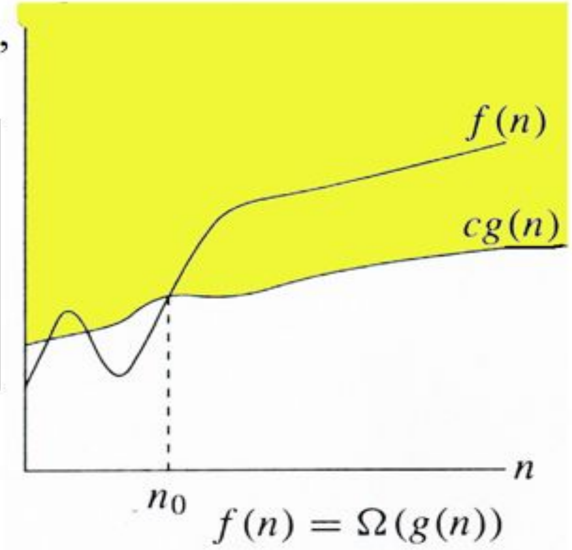
ie c=25 Assuming $n_0=1$

Ω Notation

For function $g(n)$, we define $\Omega(g(n))$, big-Omega of n, as the set:

$\Omega(g(n)) = \{f(n)$:
$\exists$ **positive constants c and n_0, such that $\forall n \geq n_0$,**
we have $0 \leq cg(n) \leq f(n)\}$

Intuitively: Set of all functions whose *rate of growth* is the same as or higher than that of $g(n)$.



$g(n)$ is an *asymptotic lower bound* for $f(n)$.

$f(n) = \Theta(g(n)) \Rightarrow f(n) = \Omega(g(n))$.
$\Theta(g(n)) \subset \Omega(g(n))$.

Qn: 3

Prove that $5n^2 = \Omega(n)$

i.e $0 \leq cn \leq f(n)$

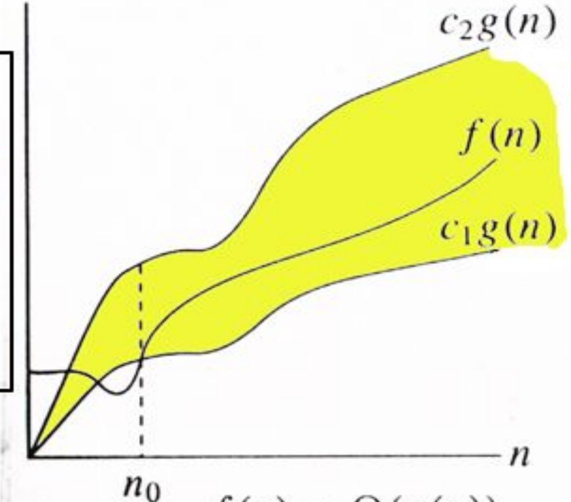
Put $c= 1$ and $n_0=1$

Θ-notation

For function $g(n)$, we define $\Theta(g(n))$, big-Theta of n, as the set:

$\Theta(g(n)) = \{f(n)$:
$\exists$ **positive constants** c_1, c_2, **and** n_0,
such that $\forall n \geq n_0$,
we have $0 \leq c_1 g(n) \leq f(n) \leq c_2 g(n)$
$\}$

Intuitively: Set of all functions that have the same *rate of growth* as $g(n)$.



$f(n) = \Theta(g(n))$

$g(n)$ is an *asymptotically tight bound* for $f(n)$.

Qn: 4

Prove that : $f(n) = 8n^2+2n-3 = \Theta(n^2)$.

To show that $f(n) \in \Theta$ (n2)

– We need to find the following three values.

– **c1, c2 and n_o**

- To find Lower bound we need **c1** and n_o
- To find Upper bound we need **c2** and n_o

Finding c1 and n_o

Lower bound: $0 \le c_1 g(n) \le f(n) \le c_2 g(n)$

$f(n) = 8n^2 + 2n - 3,\ f(n) \in \Theta\,(n^2)$

- $c_1 n^2 \le 8n^2 + 2n - 3$??
  - $7n^2 \le 8n^2 + 2n - 3$
  - $c_1 = 7$
  - $N_o = 1$

c_1 can be anything lesser than the constant with n^2 of the expression

$n_o = 1$

$7(1)^2 \le 8(1)^2 + 2(1) - 3$

$7 \le 8+2-3$

$7 \le 7$

Finding c2 and n_o

<u>Upper Bound:</u> $0 \le c_1 g(n) \le f(n) \le c_2 g(n)$

$f(n) = 8n^2 + 2n - 3, \; f(n) \in \Theta(n^2)$

$8n^2 + 2n - 3 \le c_2 n^2$

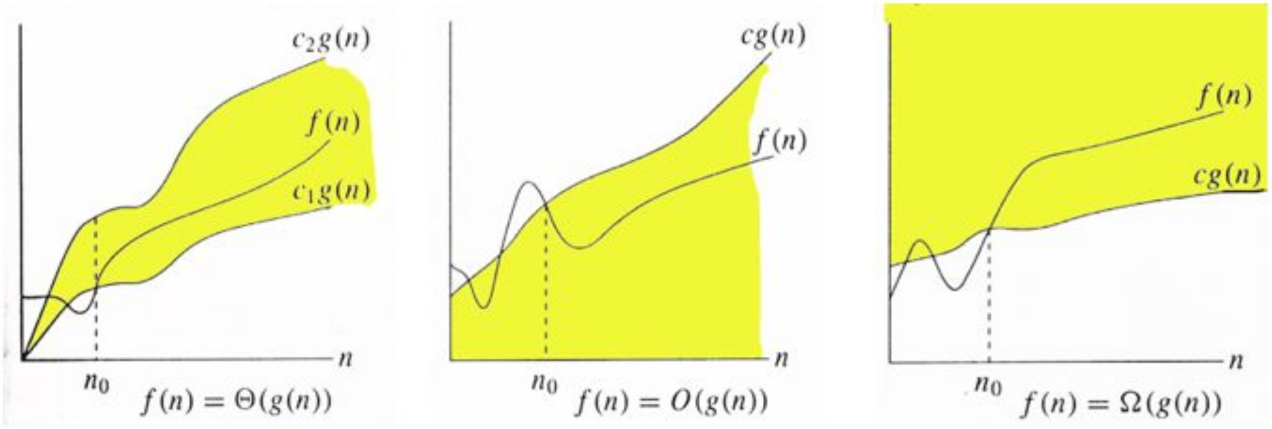
$8n^2 + 2n - 3 \le 9n^2$

$\Rightarrow 8n^2 + 2n - 3 \le 9n^2$

c_2 can be anything greater than the constant with n^2 of the expression

$c_2 = 9$

$N_o = 1$



Intuitively:

O(g(n)) contains functions whose dominant term is *at most* that of g(n) .

Ω(g(n)) contains functions whose dominant term is *at least* that of g(n).

Θ(g(n)) contains functions whose dominant term is *equal to* that of g(n).

Properties

Transitivity

$f(n) = (g(n))$ & $g(n) = (h(n))$ $f(n) = (h(n))$

Reflexivity:

If f(n) is given then

$f(n) = O(f(n))$

Symmetry:

$f(n) = \Theta(g(n))$ if and only if $g(n) = \Theta(f(n))$

Transpose Symmetry:

$f(n) = O(g(n))$ if and only if $g(n) = \Omega(f(n))$

Observations:

$O(f(n)) + O(g(n)) = O(\max(f(n), g(n)))$

$\max(f(n), g(n)) = \Theta(f(n) + g(n))$

Homework

Are each of the following true or false?

- (a) $3\, n^2 + 10\, n \log n = O(n \log n)$
- (b) $3\, n^2 + 10\, n \log n = \Omega(n^2)$
- (c) $3\, n^2 + 10\, n \log n = \Theta(n^2)$

** Submit the image of handwritten solution in teams.

Refer to the video for more worked out examples