Asymptotic Analysis

1. Definitions

- Algorithm: A sequence of steps to produce a solution to a problem, given a set of inputs
- Data Structure: A representation of data in the computer's main memory (so that it can be used efficiently)
- Abstract Data Type (ADT): A collection of data and a set of operations that can be performed on the collection so that the data behave in a well-defined way

2. Suppose we have a collection of people at a concert. How are we going to find a specific person, given their name?

```
T(n) = Running time
n = input size

* The input size is indicated with the number n; sometimes there can be multiple inputs.
```

- Ask them to stand up and wave: T(n) = 60
- Ask everyone in the audience, one at a time, if their name is the same as the given: T(n) = 7n + 4

3. Asymptotic Analysis

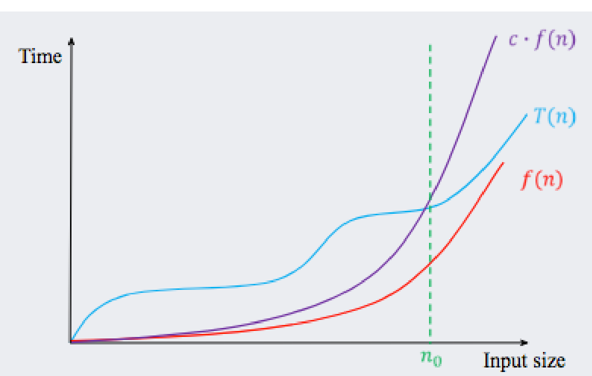
There are different metrics of algorithms:

- Time complexity: How long does it take to complete the given task?
- Space complexicity: How much memory does it take to execute the task?

Running time is a function of n such as:

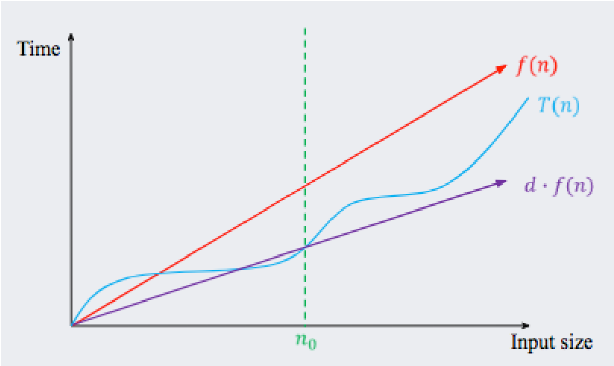
- T(n) = 4n + 5
- T(n) = 0.5nlogn - 2n + 7
- T(n) = $2^n + n^3$ + 3n

O-Notation (Classification for running time)



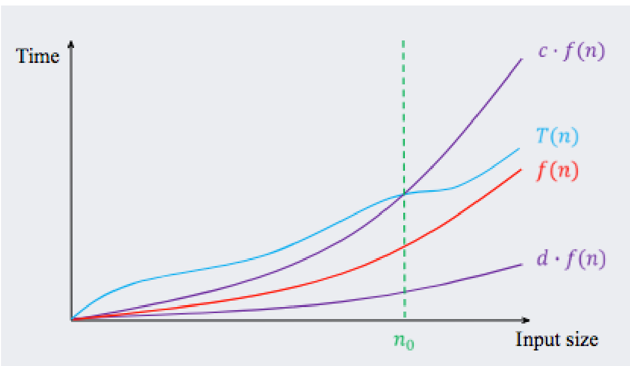
T(n) ∈ O(f(n)) if there are constants c and n_0 such that T(n) ≤ c·f(n) for all n ≥ n_0

- T(n) is bounded from above by c·f(n)
- i.e. the growth of T(n) is no faster than f(n)



$T(n) \in \Omega(f(n))$ if there are constants d and n_0 such that $T(n) \leq d \cdot f(n)$ for all $n \geq n_0$

- T(n) is bounded from below by d·f(n)
- i.e. the growth of T(n) is no slower than f(n)



$T(n) \in \Theta(f(n))$ if $T(n) \in O(f(n))$ and $T(n) \in \Omega(f(n))$

- T(n) is bounded from above and below by f(n)
- i.e. T(n) grows at the same rate as f(n)

Asymptotic Analysis hacks:

Eliminate low order terms, and then constant coefficients.

T(n)	**Eliminate low order**	f(n)
4n + 5	4n	n
0.5nlogn - 2n + 7	0.5nlogn	nlogn
$2^n + n^3 + 3n$	2^n	2^n
$n\log(n^2)$	2nlogn	nlogn

Examples:

- $10{,}000n^2 + 25n \in \Theta(n^2)$
- $10^{-10}n^2 \in \Theta(n^2)$
- $n\log n \in O(n^2)$
- $n\log n \in \Omega(n)$
- $n^3 + 4 \in O(n^4)$, but not $\Theta(n^4)$
- $n^3 + 4 \in \Omega(n^2)$, but not $\Theta(n^2)$

Typical growth rates in order

Name	Big-O Notation	Remarks
Constant	$O(1)$	
Logarithmic	$O(\log n)$	$\log_k n$, $\log(n^2) \in O(\log n)$
Poly-log	$O((\log n)^k)$	
Sublinear	$O(n^c)$	c is a constant, $0 < c < 1$
Linear	$O(n)$	
Log-linear	$O(n \log n)$	
Superlinear	$O(n^{1 + c})$	c is a constant, $0 < c < 1$
Quadratic	$O(n^2)$	
Cubic	$O(n^3)$	
Polynomial	$O(n^k)$	k is a constant; "tractable"
Exponential	$O(c^n)$	c is a constant > 0; "intractable"

Dominance

We can look at the dominant term to guess at a big-O growth rate.

e.g. $T(n) = 2n^2 + 600n + 60000$

- Up to $n = 100$, the constant term dominates
- Between $n = 100$ and $n = 300$, the linear term dominates
- Beyond $n = 300$, the quadratic term dominates, $T(n) \in O(n^2)$

Analyzing Code

Types of analysis

- Bound flavour
  - Upper bound (O)
  - Lower bound (Ω)
  - Asymptotically tight (Θ)
- Analysis case
  - Worst case (adversary)
  - Average case
  - Best case / "lucky" case
  - "common" case
- Analysis quality
  - Loose bound (any true analysis)
  - Tight boud (no better "meaningful" bound that is asymptotically different)