L01.01
Complexity, Asymptotic notation

50.004 Introduction to Algorithm

Gemma Roig
(slides adapted from Dr. Simon LUI)
ISTD, SUTD

Basic definitions

- What is a **Computational problem**:
- Map an input (e.g. x) to an output (e.g. f(x))

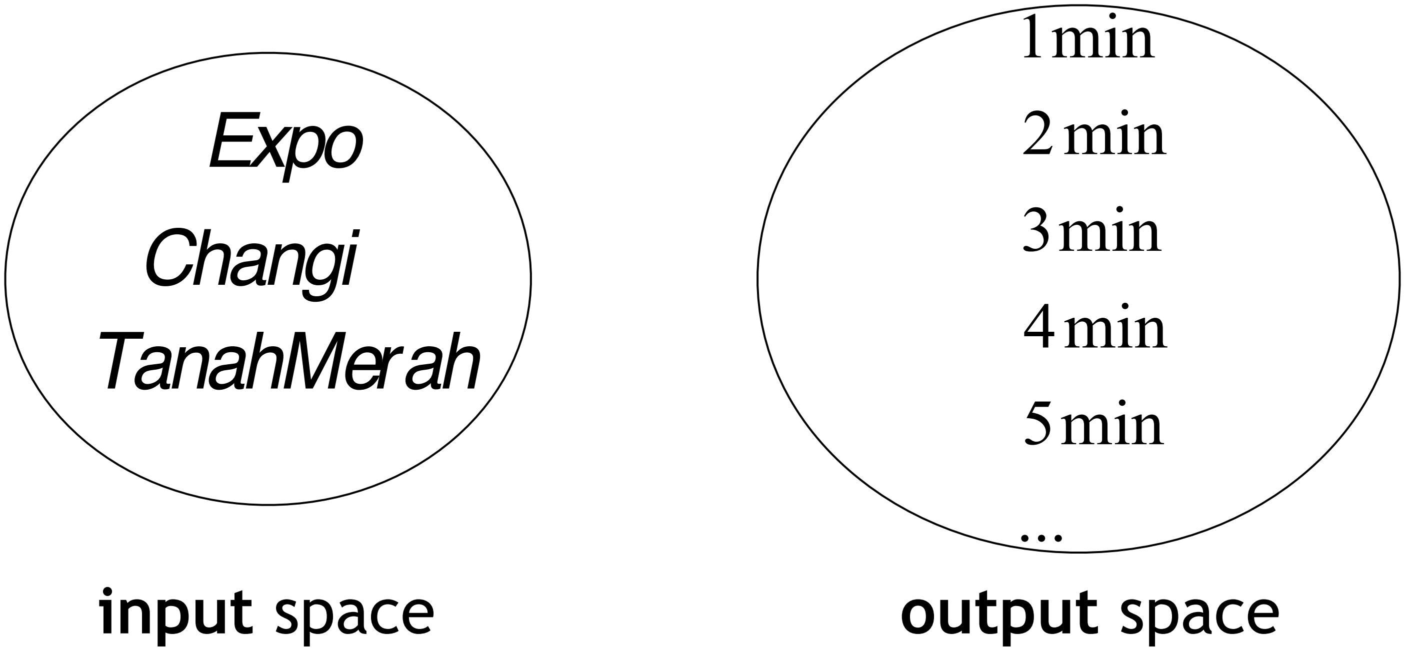
Basic definitions

- e.g. f(x) = shortest distance from Orchard to x
  - If x = “Expo”, f(x) = shortest distance from Orchard to Expo
  - If x = “Changi”, f(x) = shortest distance from Orchard to Changi
- So, given x, we want to COMPUTE f(x)
- A good algorithm can compute f(x) quickly

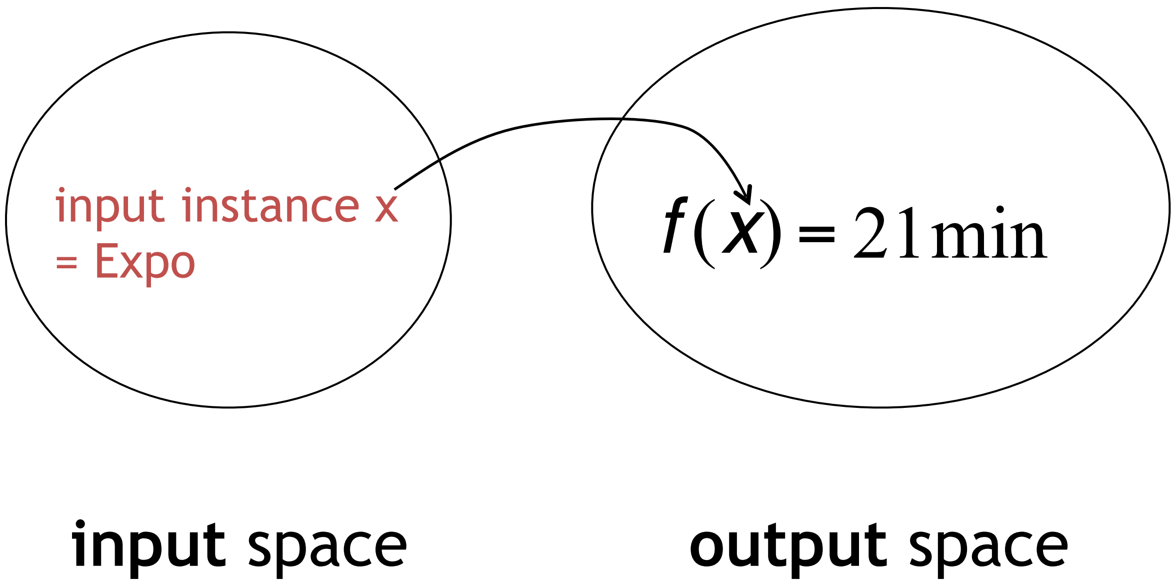
Basic definitions

- What is an input space
  - Set of possible input (e.g. input space = {expo, changi, Tanah Merah})
- What is an input instance
  - A particular input of a problem instance (e.g. x = expo)
- What is an output space
  - Set of possible output (e.g. output space = any positive number)

Basic definitions



Basic definitions



Exercise 1

- Define the input + output space, and give an example of an input instance for the following problems
  1. Integer multiplication
  2. Find if a given integer k is in a list of n integers $[k_1,k_2,\ldots,k_n]$
  3. Sort a list of n integers in increasing order

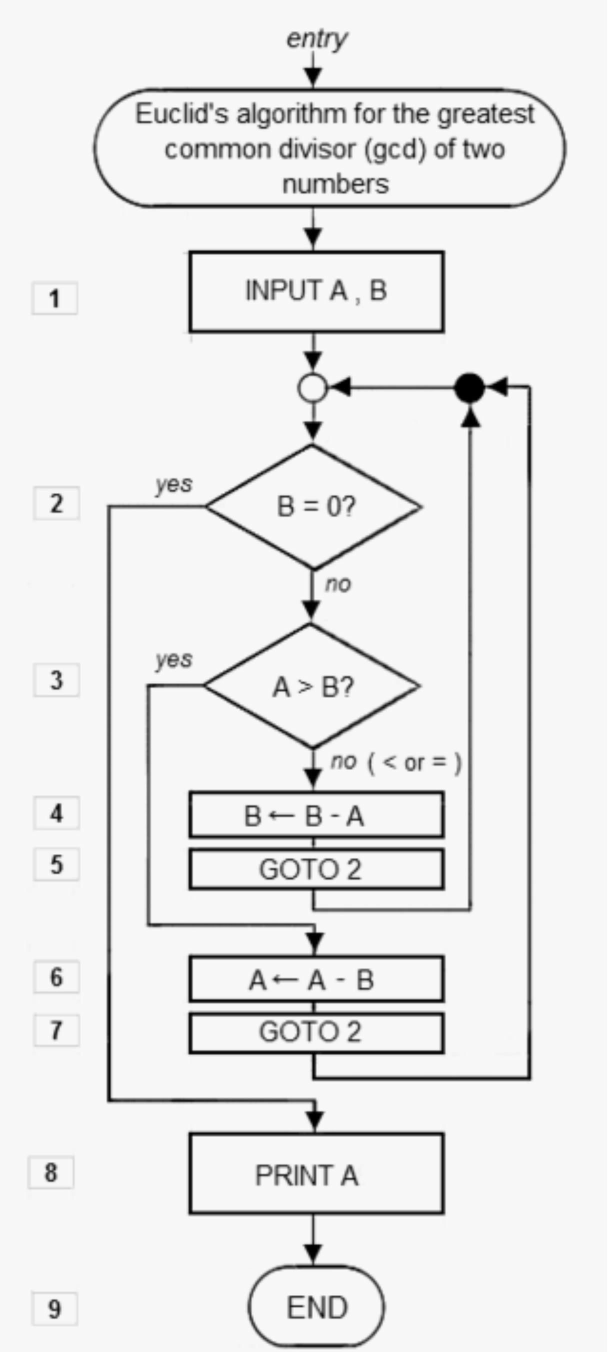
- What is the “size” of the input instance in each case? “Bigger” question = more work required to answer

Algorithm

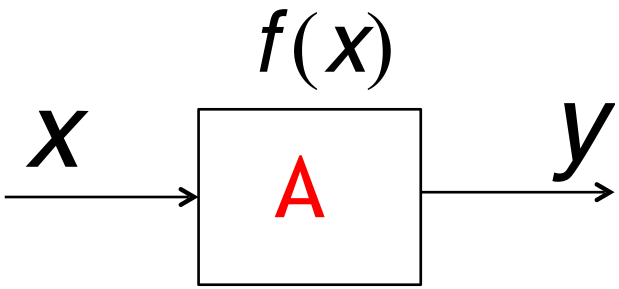
- Procedure for solving a computational problem
- Usually a finite sequence of operations
  - described in structured English
  - pseudocode or real code

Algorithm

- Example



Property of algorithm



$|x| = $ "size" of problem instance $= n$

$T(n) =$ number of steps to solve the problem as a function of its size

A Good Algorithm

- Correct
- Fast
  - For a good algorithm, $T(n)$ should increase “slowly” as n grows

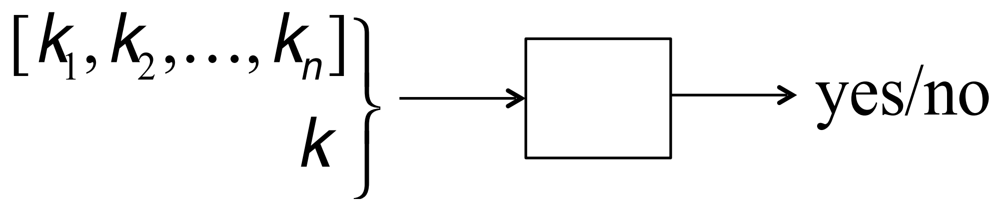
Algorithmic time complexity $T(n)$

Example P: Is the number k in a list of n numbers?

$$\left.\begin{array}{r}[k_1, k_2, \ldots, k_n] \\ k\end{array}\right\} \longrightarrow \boxed{} \longrightarrow \text{yes/no}$$

Algorithmic time complexity $T(n)$

Example P: Is the number k in a list of n numbers?



- Algorithm A1: check for each element k_i of the list if $k_i = k$
- How much time does A1 takes? (i.e. What is T(n)?)

- T(n) is the exact number of steps it takes to run A1 (e.g. T(n) = 1002 times. e.g. T(n)=3045 times)

Asymptotic complexity

- The asymptotic complexity describes $T(n)$, as n grows to infinity

- In this course, we talk about THREE types of Asymptotic complexity
  - Θ (Big Theta)
  - O (Big O)
  - Ω (Big Omega)

Asymptotic complexity

– Θ (Big Theta) means “grows asymptotically = ”

– O (Big O) means “grows asymptotically <= ”

– Ω (Big Omega) means “grows asymptotically >= ”

Asymptotic complexity

- Θ (Theta) means "grows asymptotically equal "
- For example

$$n^2 = \Theta(n^2)$$

- For example

$$0.1n^2 - 100n^{1.9} + 5 = \Theta(n^2)$$

- F(x) = Θ (G(x)) means "F grows as G, when x grows to infinity"

Asymptotic complexity

- O (Big O) means "grows asymptotically <= "
- For example

$$n^2 = O(n^{1000})$$

- For example

$$2n^3 + 100n^2 + 5 = O(n^{3000000})$$

- F(x) = O(G(x)) means "F grows at most as fast as G, when x grows to infinity"

Asymptotic complexity

- Ω (Omega) means "grows asymptotically >="
- For example

$$n^{9999} = \Omega(1)$$

- For example

$$2n^{999999} + 100n^{33} + 5 = \Omega(n^2)$$

- F(x) = Ω (G(x)) means "F grows at least as fast as G, when x grows to infinity"

Quick Exercises

$$(1.01)^x = ?(x^{10} \log x)$$

$$x \log \log x = ?(x^{1.5})$$

$$x^2 + x(\log x)^2 = ?(x^2)$$

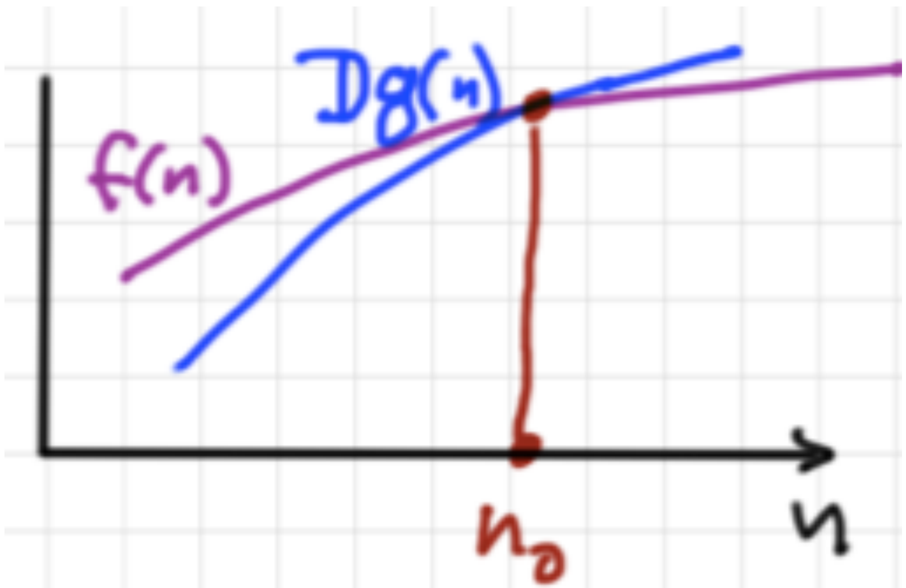
Hints, asymptotic increasing order (slow to quick)
$\log(\log x)$, $\log x$, x, x^k, k^x, x^x

Exact definitions

$$f(n) = O(g(n)) \iff$$

$$\exists D > 0,\ n_0 \text{ such that}$$

$$|f(n)| \le D|g(n)| \text{ for } n \ge n_0$$

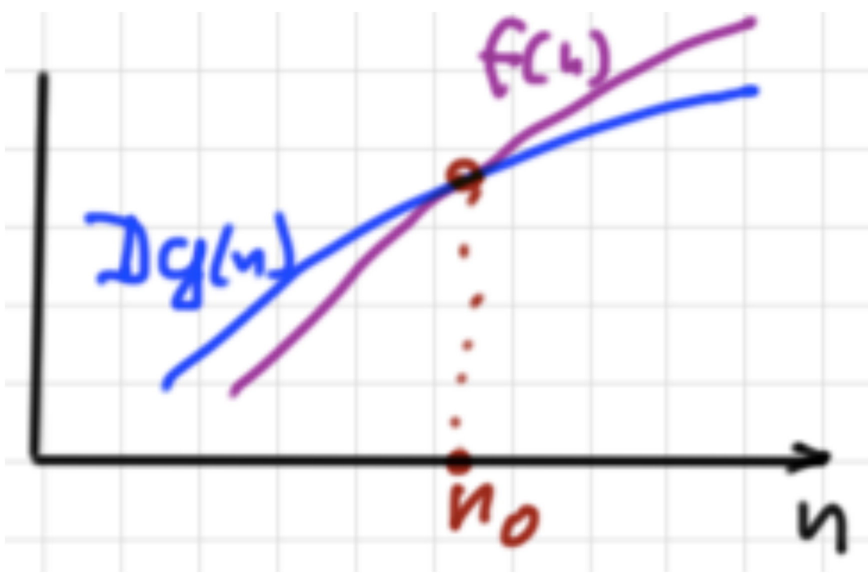


Exact definitions

$$f(n) = \Omega(g(n)) \Leftrightarrow$$

$$\exists D > 0,\ n_0 \text{ such that}$$

$$|f(n)| \geq D|g(n)| \text{ for } n \geq n_0$$

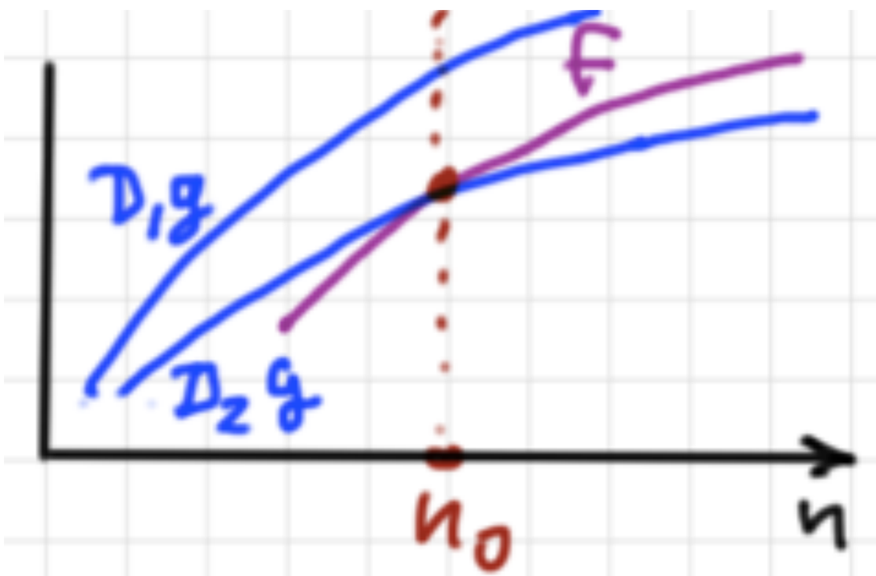


Exact definitions

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

$$\exists D_1, D_2 > 0,\ n_0 \text{ such that}$$

$$D_1 |g(n)| \geq |f(n)| \geq D_2 |g(n)| \text{ for } n \geq n_0$$



Exact definitions

Examples

$f(n) = 100n + 1000$

$g(n) = n^2$

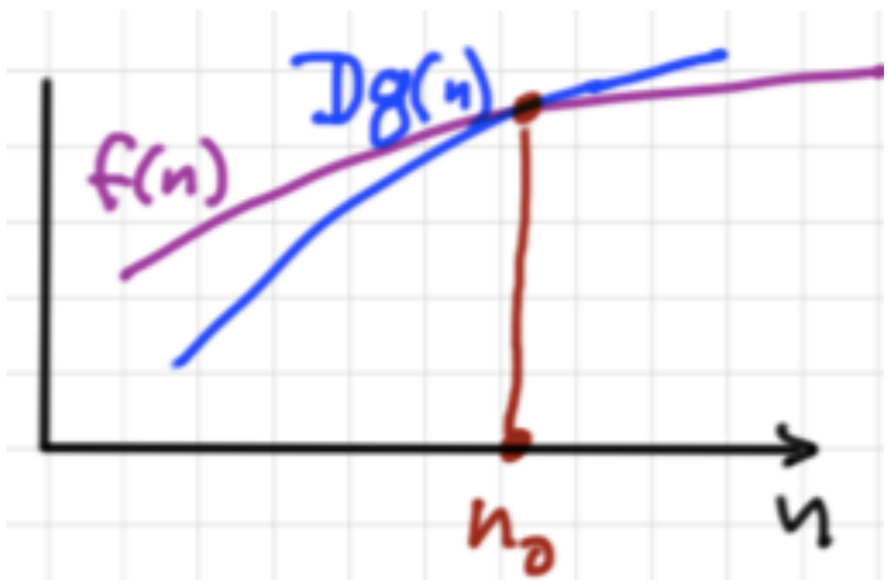
$so,\ f(n) = O(g(n))$

$f(n) = O(g(n)) \Leftrightarrow$

$\exists D > 0,\ n_0$ such that

$|f(n)| \le D|g(n)|$ for $n \ge n_0$

D=1



Exact definitions

$$f(n) = 0.001n$$

$$g(n) = n^{0.5} + 1000$$

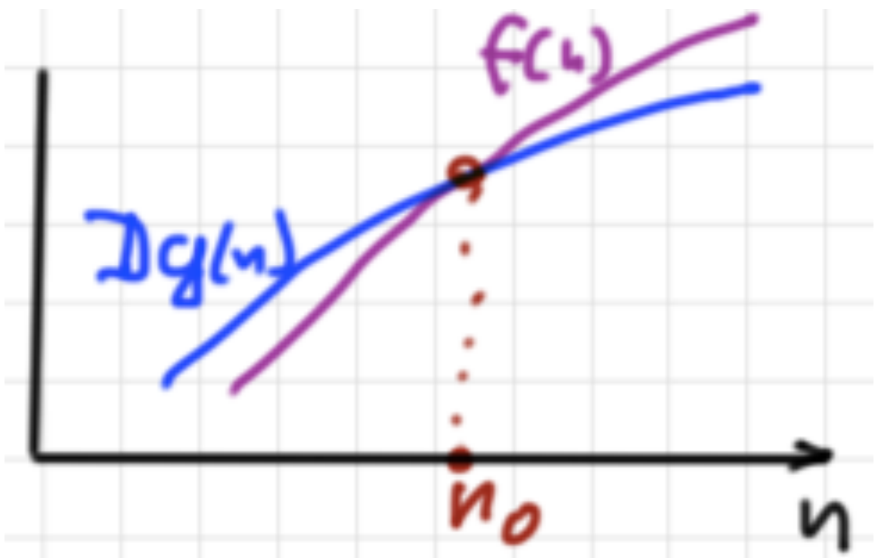
$$so,\ f(n) = \Omega(g(n))$$

$$f(n) = \Omega(g(n)) \Leftrightarrow$$

$$\exists D > 0,\ n_0 \text{ such that}$$

$$|f(n)| \geq D|g(n)| \text{ for } n \geq n_0$$

D=0.001



Exact definitions

$$f(n) = 2n^2 + 100n - 1000$$

$$g(n) = n^2$$

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

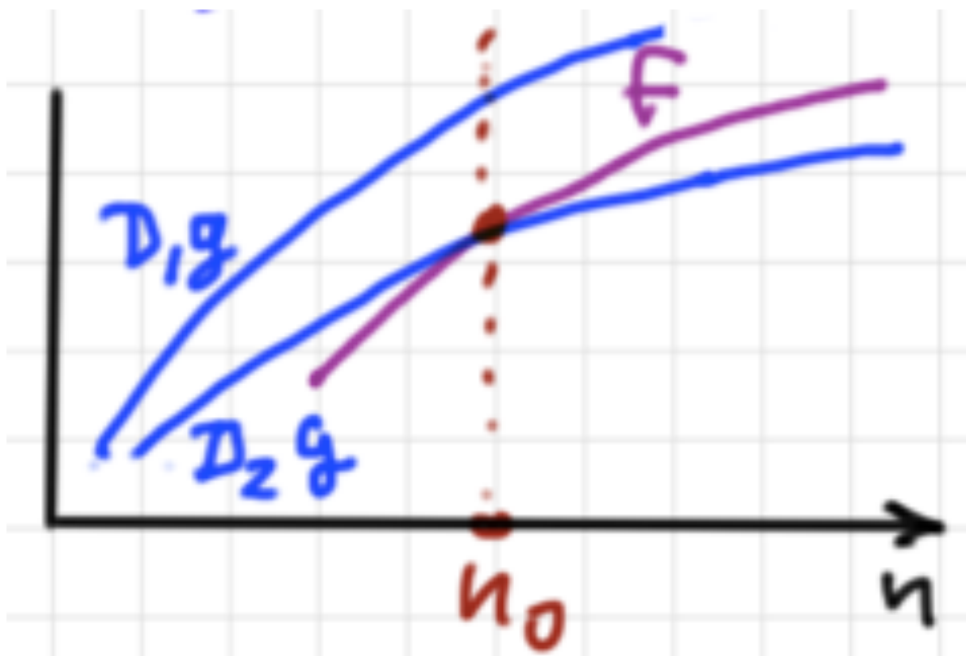
$f(n) = \Theta(g(n)) \Leftrightarrow$

$\exists D_1, D_2 > 0,\ n_0$ such that

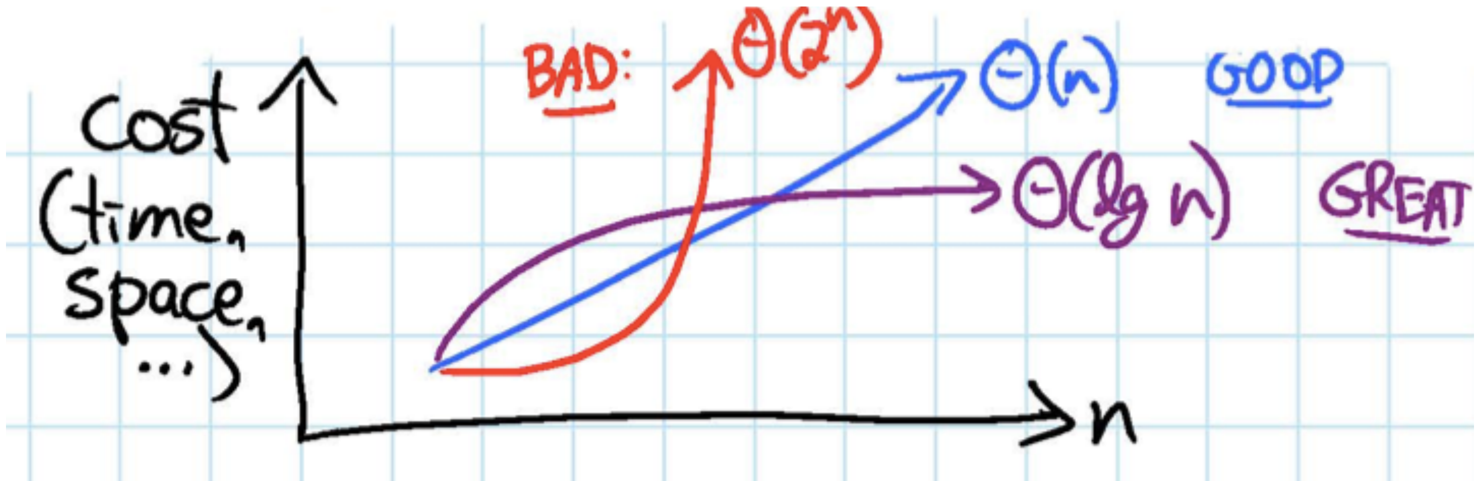
$D_1|g(n)| \geq |f(n)| \geq D_2|g(n)|$ for $n \geq n_0$

$D_1=2$

$D_2=1$



Complexity of algorithms



In simple and practical words used in the industry

$\Theta(\log n)$: great algorithm

$\Theta(n)$, $\Theta(n \log n)$, $\Theta(n^2)$: good algorithm

$\Theta(n^3)$: hard to say....

$\Theta(2^n)$: bad algorithm

Asymptotic complexity in practice

Data Size in μs

"Practical" Algorithm

Practically ineffective algorithm

n	$\log_2 n$	n	$n \log_2 n$	n^2	n^3	2^n
10	3.3 μs	10 μs	33 μs	100 μs	1 ms	1 ms
10^2	6.6 μs	100 μs	664 μs	10 ms	1 s	4×10^{16} years
10^4	13 μs	10 ms	133 ms	1.7 minutes	11.6 days	10^{2997} years
10^6	20 μs	1 s	20 s	11.6 days	32000 years	10^{300000} years

The time needed to solve a problem

Conclusions

- Key concepts:
  - Problem
  - Space and instance
  - algorithm
  - properties of algorithms
  - asymptotic complexity (Θ, O, Ω)