# R1 Intro about Algorithm

3.22 https://github.com/GUMI-21/MIT6.006_note

•A problem is a binary relation connecting problem inputs to correct outputs.
•A (deterministic) algorithm is a procedure that maps inputs to single outputs.
•An algorithm solves a problem if for every problem input it returns a correct output.

## Correctness

often proof via induction.

## Efficiency

storage space or running time.
use *asymptotic performance* and ignore constant factor differences in haedware performance.

## Asymptotic Notation

*upper bound*

- O Notation: Non-negative function g(n) is in O(f(n)) if and only if there exists a positive real number c and positive integer n0 such that $g(n) \leq c \cdot f(n)$ for all $n \geq n0$.
  *lower bounds*
- Ω Notation: Non-negative function g(n) is in Ω(f(n)) if and only if there exists a positive real number c and positive integer n0 such that $c \cdot f(n) \leq g(n)$ for all $n \geq n0$.
  *tight bound*
- Θ Notation: Non-negative g(n) is in Θ(f(n)) if and only if $g(n) \in O(f(n)) \cap \Omega(f(n))$.

## Mod of Computation

we will use the *w-bit WordRAM* model of computation.
If a machine word contains only w bits, the processor will only be able to read and write from at most 2w addresses in memory2.
we will always assume our Word-RAM has a word size of at least w > log2 n bits,or else the machine would not be able to access all of the input in memory. a Word-RAM model of a byte-addressable 64-bit machine allows inputs up to ~ 1010 GB in size.

## Data Structure

The set of operations supported by a data structure is called an *interface*

ex: static array

- StaticArray(n): allocate a new static array of size n initialized to 0 in $\Theta(n)$ time
- StaticArray.get at(i): return the word stored at array index i in $\Theta(1)$ time
- StaticArray.set at(i, x): write the word x to array index i in $\Theta(1)$ time