The RAM Computation Model

Yufei Tao

Department of Computer Science and Engineering
Chinese University of Hong Kong
This is not a programming course.

**Take-away message 1 from this course**

Programming is the last step in software development, which occurs only after the algorithms are clear.

**Take-away message 2**

Computer science is a branch of mathematics with its art reflected in the beauty of algorithms.

- Programming knowledge is not necessary to study algorithms.

**Many people believe that this branch holds the future of mankind.**
In mathematics (and hence, computer science) everything—including every term and symbol—must be rigorous.

Computer science is a subject where we

1. first define a **computation model**, which is a **simple** yet **accurate** abstraction of a computing machine;

2. then slowly build up a theory in this model from scratch.
The Random Access Machine (RAM) model

A machine has a memory and a CPU.

**Memory**

- An infinite sequence of cells, each of which contains the same number $w$ of bits.
- Every cell has an address: the first cell of memory has address 1, the second cell 2, and so on.
The Random Access Machine (RAM) model

CPU

- Contains a fixed number—8 in this course—of registers, each of which has \( w \) bits (i.e., same as a memory cell).

![Diagram of CPU with 8 registers, each with \( w \) bits and an address line.](image)
The Random Access Machine (RAM) model

CPU

Can do the following atomic operations:

1. **(Register (Re-)Initialization)**
   Set a register to a fixed value (e.g., 0, −1, 100, etc.), or to the content of another register.
The Random Access Machine (RAM) model

CPU

- Can do the following **atomic operations**:
  2. **(Arithmetic)**
     Take the integers $a, b$ stored in two registers, calculate one of the following and store the result in a register:
     - $a + b$, $a - b$, $a \cdot b$, and $a/b$.

Note: $a/b$ is “integer division”, which returns an integer. For example, $6/3 = 2$ and $5/3 = 1$. 
The Random Access Machine (RAM) model

CPU

- Can do the following **atomic operations**: 

  3. **(Comparison/Branching)**
  Take the integers \( a, b \) stored in two registers, compare them, and learn which of the following is true:
  - \( a < b \), \( a = b \), \( a > b \).
The Random Access Machine (RAM) model

CPU

- Can do the following **atomic operations**:

  4. **(Memory Access)**
     Take a memory address \( A \) currently stored in a register. Do one of the following:
     - Read the content of the memory cell with address \( A \) into another register (overwriting the bits there).
     - Write the content of another register into the memory cell with address \( A \) (overwriting the bits there).
The Random Access Machine (RAM) model

CPU

Can do the following atomic operations:

5. **(Randomness)**
   - RANDOM(x, y): Given integers x and y (satisfying $x \leq y$), this operation returns an integer chosen uniformly at random in $[x, y]$, and places the random integer in a register.
An **execution** is a sequence of atomic operations.

Its **cost** (also called its **running time**, or simply, **time**) is the **length** of the sequence, namely, the number of atomic operations.
The Random Access Machine (RAM) model

A *word* is a sequence of $w$ bits, where $w$ is called the *word length*.

- In other words, each memory cell and CPU register store a word.

Unless otherwise stated, you do not need to pay attention to the value of $w$ in this course.
An **input** refers to the initial state of the registers and the memory before an execution starts.

An **algorithm** is a piece of description that, given an input, can be utilized to **unambiguously** produce a sequence of atomic operations, namely, the **execution** of the algorithm.

- In other words, it should be always clear what the next atomic operation should be, given the outcome of all the previous atomic operations.

The **cost** of an algorithm on an input is the length of its execution on that input (i.e., the number of atomic operations required).
Deterministic Algorithms vs. Random Algorithms

An algorithm is **deterministic** if it never invokes the atomic operation `RANDOM`. Otherwise, the algorithm is **randomized**.

On the same input, the cost of a deterministic algorithm is a fixed integer—it remains the same every time you execute the algorithm.

The cost of a randomized algorithm, however, is a random variable. Even on the same input, the cost may change each time the algorithm is executed.
Example

1. do
2. \( r = \text{RANDOM}(0, 1) \)
3. until \( r = 1 \)

How many times would Line 2 be executed? The answer is—“we don’t know” (in fact, the line may be executed an infinite number of times)! Every time the above “algorithm” is executed, it may produce a new sequence of atomic operations.
Let $X$ be a random variable that equals the cost of an algorithm on an input. The \textbf{expected cost} of the algorithm on the input is the expectation of $X$. 