

Algorithm

- [MIT 6.006 Introduction to Algorithms, Spring 2020](#)
 - [1. Algorithms and Computation](#)
 - [2. Data Structures and Dynamic Arrays](#)
- [Word RAM Model](#)

MIT 6.006 Introduction to Algorithms, Spring 2020

<https://www.youtube.com/playlist?list=PLUI4u3cNGP63EdVPNLG3ToM6LaEUuStEY>

1. Algorithms and Computation

What?

- input, output

“What is the input, and what is the output?”

How?

- How?

What is the input? What is the output?

What is the input? What is the output?

- What is the input?
- Inductive Hypothesis: if first k students contain match alg returns a match before interviewing student $k + 1$
- What is the output?
 - base case: $n=0$ $n=0$ $n=1$ $n=1$
 - Inductive Step:
 - k : $k+1$

What is the input? What is the output?

- What is the input?
- What is the output?
 - (IBM's research computer vs my computer)
- Don't measure time, instead count ops
- Expect performance to depend on size of our input

- $O(n)$ upper bound $\Omega(n)$ lower bound θ both

32-bit CPU can only handle 32-bit integers. If you try to store a value greater than 2^{32} , it will overflow and wrap around. For example, if you store 4GB, it will wrap around to 0, and if you store 4GiB, it will wrap around to 0.

2. Data Structures and Dynamic Arrays

What data can store (What data can store)

Interface (API / ADT)

- What data can store (API, ADT)
- What data can store

Specification (What data can store)

- What data can store
- What data can store

What data can store (What data can store)

- What data can store
- What data can store

What operations are supported (What data can store)

- What data can store
- What data can store (add, remove, find)

What data can store

- What data can store
- What data can store

Representation (如何表示)

- 如何表示一个字符串
- 如何表示一个数组
 - 如何表示一个数组
 - 如何表示一个数组

How to store data (如何存储)

- 如何存储一个字符串
- 如何存储一个数组
 - 如何存储一个数组
 - 如何存储一个数组

Algorithms to support operations (如何支持操作)

- 如何支持一个字符串
- 如何支持一个数组

Word RAM(Random Access Machine) Model

- 如何支持一个字符串
- [Word RAM Model](#) 如何支持

Word RAM Model

This may seem simple, but we're really going to need this model and really rely on this model increasingly as we get to more interesting data structures. By Erik Demaine

The Word RAM model is a computational model used to analyze the performance of algorithms and data structures. It assumes a random-access memory (RAM) where the memory is organized as an array of w -bit words.

Key Concepts

- **Array:**
 - An array is a contiguous block of memory.
 - Accessing an element in the array can be done in $O(1)$ time.
- **Memory:**
 - The memory is an array of w -bit words.
 - An "array" refers to a consecutive chunk of memory.
 - Accessing an element in the array can be done in $O(1)$ time.
 - `array[i] = memory[address(array) + i]`.

w -bit Words

- **Definition:**
 - Each word in the memory is w bits wide.
 - " w " represents the word size, which is the number of bits in each word.
- **Examples:**
 - In a 32-bit system, each word is 32 bits (4 bytes).
 - In a 64-bit system, each word is 64 bits (8 bytes).

Importance

- **Data Representation:**
 - The size of w determines how much data can be stored in each word.
- **Memory Addressing:**
 - The word size affects the memory address space. For example, a 64-bit system can address more memory than a 32-bit system.
- **Performance:**

- Algorithms assume $O(1)$ time complexity for accessing memory locations. This is crucial for the efficiency of many data structures and algorithms.

Usage

- The Word RAM model is fundamental in computer science for designing and analyzing algorithms.
- It simplifies the theoretical analysis by assuming constant time for basic operations like reading and writing to memory.

By understanding the Word RAM model and the significance of w -bit words, one can better appreciate the design and efficiency of modern data structures and algorithms.