



# LECTURE 1 – COMPLEXITY & MEMORY, PRIME FACTORIZATION, GCD AND BASIC DATA STRUCTURES

**Prepared by:** Izbassar Assylzhan

**Course:** Algorithms & Data Structures – Fall 2025

# LECTURE 1

---



In this lecture we'll understand **Complexity & Memory** usage of algorithms, start writing C++ code in **VSCode**, and look at the algorithms for **Prime Factorization**, finding **Greatest Common Divisor** using **Sieve Eratosthenes algorithm**, and data structures like **Stack, Queue, Deque**

## TOPICS WE'LL COVER:

Complexity & Memory

Prime Factorization

Greatest Common Divisor

Stack, Queue, Deque

## GOALS FOR THIS LECTURE:

- Learn how to analyze algorithm efficiency using time and memory complexity (Big-O)
- Understand and implement prime factorization, GCD, and the Sieve of Eratosthenes
- Get comfortable using stack and queue data structures with examples

# COMPLEXITY & MEMORY

Complexity &  
Memory

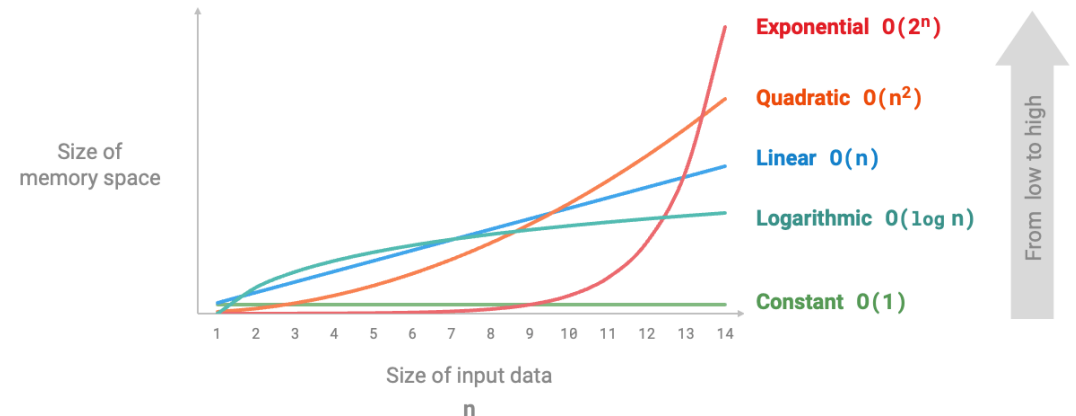
Prime  
Factorization

Greatest Common  
Divisor

Stack, Queue,  
Deque

**Algorithm Complexity** is a measure of how the time or memory usage of an algorithm grows as the input size increases

- **Time Complexity** is the amount of time an algorithm takes to run, expressed as a function of input size  $n$  (using Big-O notation)
- **Memory Complexity** is the amount of memory an algorithm uses, including variables, data structures, recursion stack, etc.
- **Tradeoff** is in sometimes we use more memory to gain faster execution (e.g., precomputing primes with a sieve)



# PRIME FACTORIZATION

A **prime number** is a natural number greater than 1 that has exactly two distinct divisors: 1 and itself.

- Given an integer  $n$ , find **all prime numbers** in the **range 1 to  $n$** .

```
7  vector<bool> is_prime (n+1, true);
8  is_prime[0] = is_prime[1] = false;
```

```
10  for (size_t i = 2; i <= n; ++i) {
11      if(is_prime[i]) {
12          if (i * 1ll * i <= n) {
13              for (size_t j = i * i; j <= n; j += i) {
14                  is_prime[j] = false;
15              }
16          }
17      }
18  }
```

Sieve of Eratosthenes

```
20
2 3 5 7 11 13 17 19
```

Complexity &  
Memory

Prime  
Factorization

Greatest Common  
Divisor

Stack, Queue,  
Deque

# GREATEST COMMON DIVISOR

**Greatest Common Divisor (GCD)** is the largest positive integer that divides two (or more) integers without leaving a remainder.

- Examples:  $\text{GCD}(12, 18) = 6$ ;  $\text{GCD}(9, 28) = 1$  (coprime numbers).

$$\text{gcd}(a, b) = \begin{cases} a, & \text{if } b=0 \\ \text{gcd}(b, a \bmod b), & \text{otherwise} \end{cases}$$

```

5  int gcd(int a, int b) {
6      if (b == 0)
7          return a;
8      else
9          return gcd(b, a % b);
10 }
```

65 35  
5

Referenced from [http://e-maxx.ru/algo/euclid\\_algorithm](http://e-maxx.ru/algo/euclid_algorithm)

Complexity &  
Memory

Prime  
Factorization

Greatest Common  
Divisor

Stack, Queue,  
Deque

# STACK, QUEUE, DEQUE

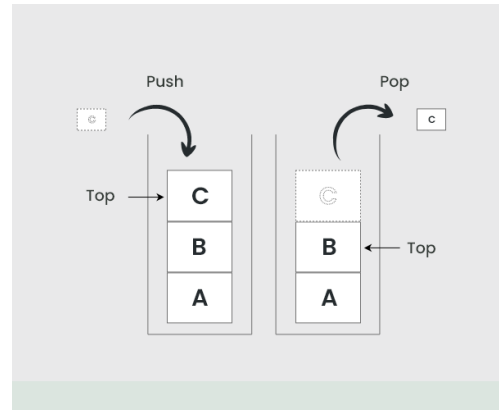
Complexity & Memory

Prime Factorization

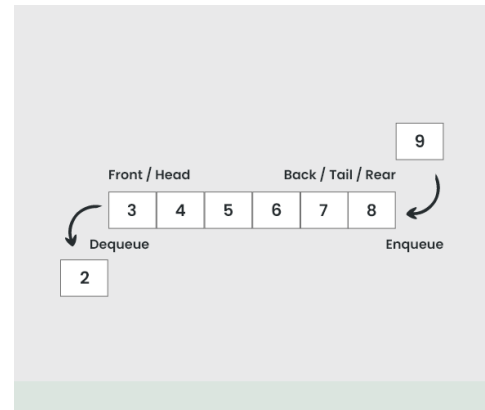
Greatest Common Divisor

Stack, Queue, Deque

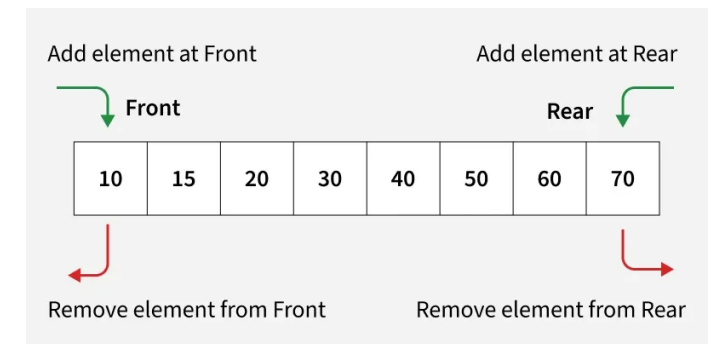
**Stack** is a linear data structure where elements are added and removed from one end only.



**Queue** is a linear data structure where elements are added at the rear and removed from the front.



**Deque** is a generalized queue where insertion and deletion are allowed at both the front and the rear.



Referenced from <https://geeksforgeeks.org>

Q & A