**What is Competitive Programming?**

Competitive programming is a sport where contestants solve algorithmic problems within a time limit using a programming language of their choice.

Pre-Requisites of Competitive Programming…

* *Programming Language Proficiency*
* *Basic Data Structures*
* *Algorithms*
* *Mathematics*
* *Problem-Solving Skills*
* *Debugging Skills*
* *Efficient Coding Techniques*
* *Online Judges and Platforms*
* *Time Management*
* *Practice and Consistency*
* *Community Involvement*
* *Competitive Spirit*

Basic terminologies:

* *Time limit exceed*
* *Memory limit exceed*
* *Runtime error*
* *Compilation error*
* *Optimization*
* *Template*
* *Editorials*

**Endl vs Back n \*important**

Endl not only inserts a newline character but also flushes the output buffer. Whereas in case of \n used to insert a line break but it does not flush the output buffer.

Endl ; = ‘\n’ << flush;

Difference is seen in case of bulky programs.

C++ standard streams (cin, cout, cerr)

C standard streams (stdin, stdout, stderr).

**ios\_base::sync\_with\_stdio(false);**

is used in C++ programming with the Standard Library's input and output streams to control the synchronization between C++ standard streams (cin, cout, cerr) and C standard streams (stdin, stdout, stderr).

#include <iostream>

int main() {

std::ios\_base::sync\_with\_stdio(false); // Turn off synchronization

// Now you can use C++ and C-style I/O operations without synchronization issues

std::cout << "Hello, ";

printf("world!\n");

return 0;

}

**Decreasing rate of growth…**

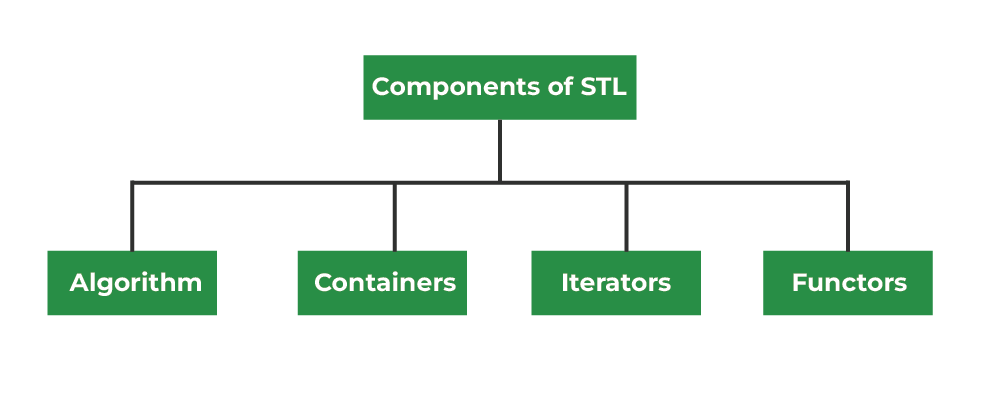
1 < loglogn < sqrt(log(n)) < logn < log2n < 2log(n) < n < log(n!) < nlog(n) < n2 < n3 < 2n < n!

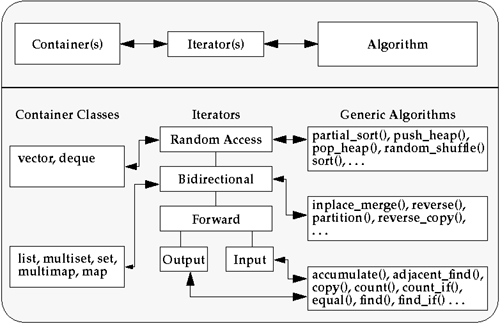
NOTE: To access any element in array constant time is needed because other elements scanning is not required.

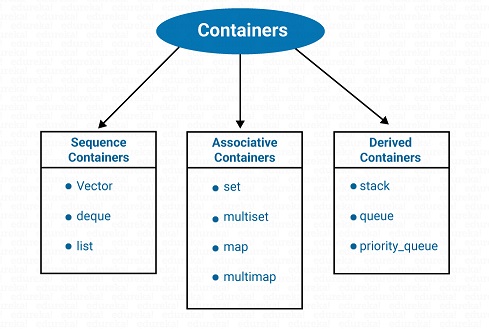


#include <bits/stdc++.h>

**Standard Template Library(20.09.2004)**







**Standard array (**#include <array>**)**

#include <iostream>

#include <array>

int main() {

// Define an array of integers

std::array<int, 5> numbers = {10, 20, 30, 40, 50};

// Calculate the sum of the elements

int sum\_of\_numbers = 0;

for (int num : numbers) {

sum\_of\_numbers += num;

}

// Calculate the average

double average = static\_cast<double>(sum\_of\_numbers) / numbers.size();

// Display the array, sum, and average

std::cout << "Array: [";

for (int i = 0; i < numbers.size(); ++i) {

std::cout << numbers[i];

if (i < numbers.size() - 1) {

std::cout << ", ";

}

}

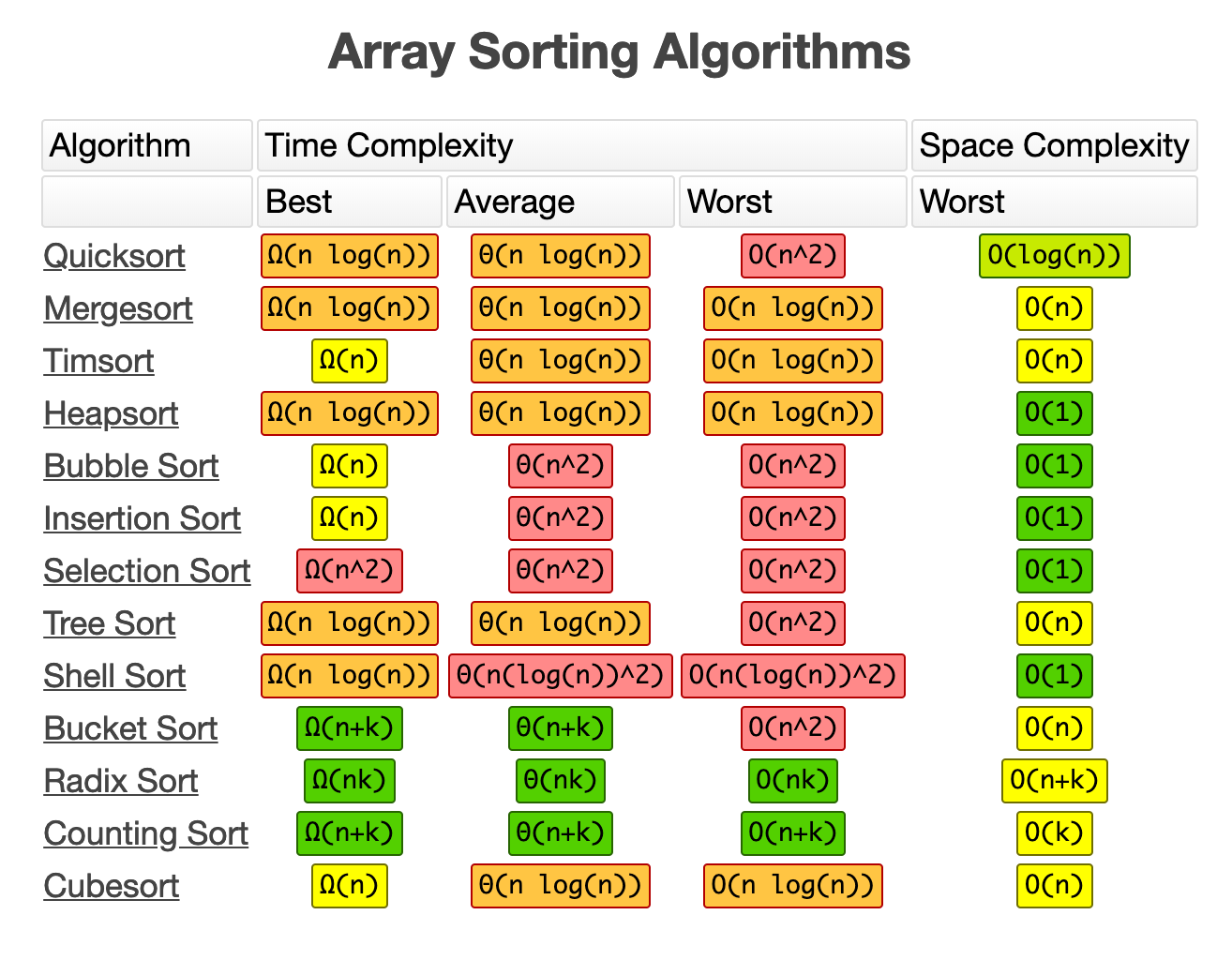
std::cout << "]" << std::endl;

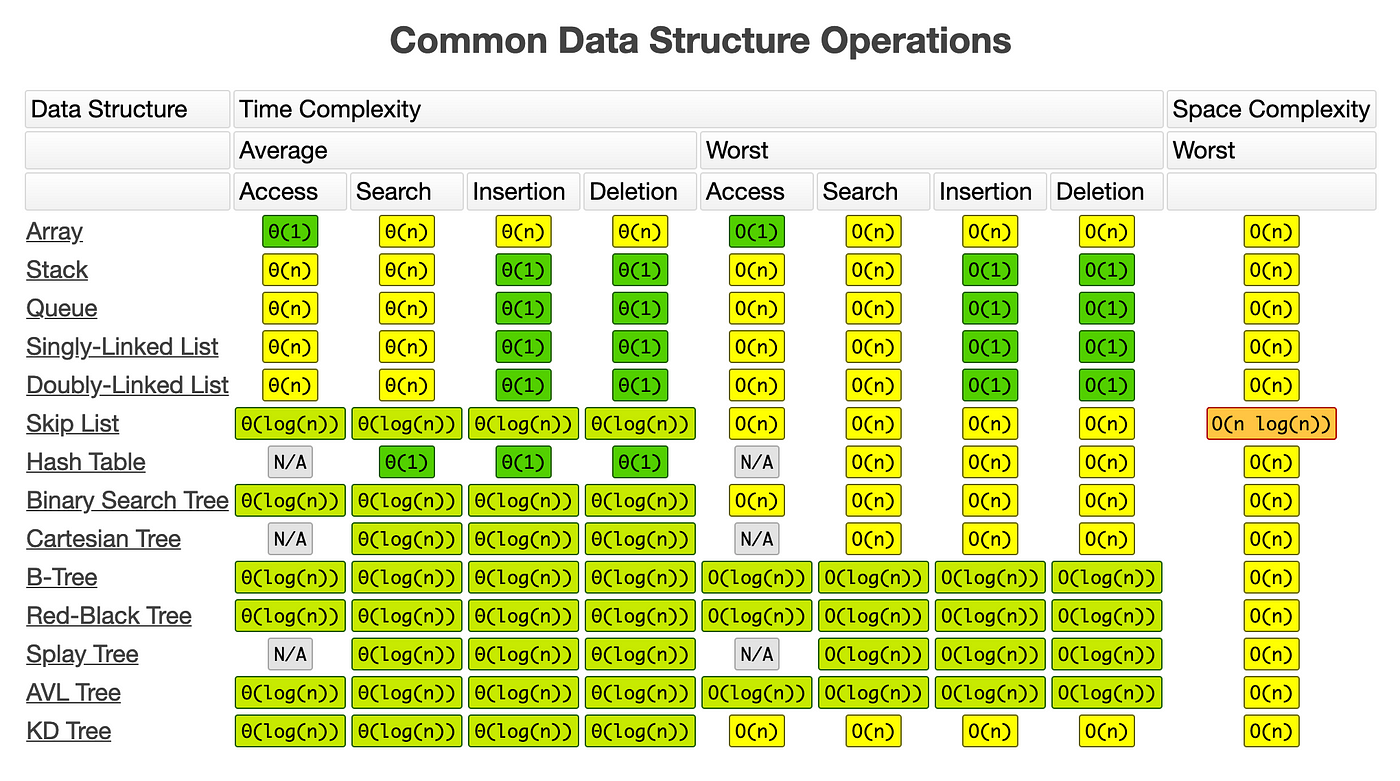
std::cout << "Sum of numbers: " << sum\_of\_numbers << std::endl;

std::cout << "Average: " << average << std::endl;

return 0;

}





n! > 2n > n3 > n2 > nlog(n)

> log(n!) > n > 2logn > log2n

> log(n) > √log(n) >

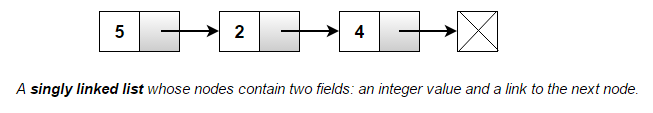
log(logn) > 1

BASICS OF DATA STRUCTURES…

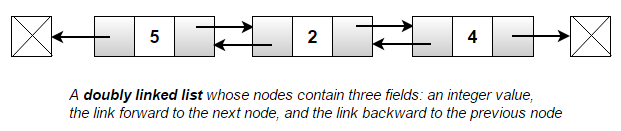
{LINKED LIST}

* *Singly*
* *Doubly*
* *Circular*

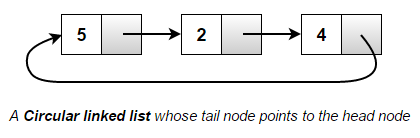
Singly Linked list



Doubly Linked list



Circular Linked list



Q1.WAP to insert an element at the end of the Linked List!

#include<iostream>

using namespace std;

struct Node {

int val;

Node\* next;

};

// Function to insert a new node at the end of the linked list

void insertNode(Node\*& head, int value) {

Node\* newNode = new Node;

newNode->val = value;

newNode->next = nullptr;

if (head == nullptr) {

// If the linked list is empty, make the new node the head

head = newNode;

} else {

// Traverse the list to find the last node

Node\* current = head;

while (current->next != nullptr) {

current = current->next;

}

// Insert the new node at the end

current->next = newNode;

}

}

// Function to print the linked list

void printList(Node\* head) {

Node\* current = head;

while (current != nullptr) {

cout << current->val << " ";

current = current->next;

}

cout << endl;

}

int main() {

Node\* head = nullptr;

int element;

// Input an element to insert into the linked list

cout << "Enter an element to insert into the linked list: ";

cin >> element;

// Call the insertNode function to insert the element

insertNode(head, element);

// Print the updated linked list

cout << "Updated Linked List: ";

printList(head);

return 0;

}