CODE

Competitive programming 101
POINTERS & LINKED LISTS

By: Wafae Boumajjane & Hamza Bouali

POINTERS

Time complexity

Time complexity measures how the running time of an algorithm grows as the input size increases. We use Big O notation to express this.

Space complexity

Space complexity measures the additional memory space needed by an algorithm as the input size grows.

HOW TO CALCULATE ?

Time complexity

- each elementary operation, constant process are counted as 1 operation(input,output, mathematical operations, accessing element in indexed data structure, creating variables)
- for loops, we count the product of the number of the maximum repetitions possibles if it’s not constant

Space complexity

- how much of the memory the algorithm use.
- we count each bit as 1 unit (each data type has it’s own size)
- it’s less complicated than time complexity.

CODE

HOW TO CALCULATE ?

the math behind stu ff

Asymptotic Notation

Big O Notation (Upper Bound)

- **let's consider f(n) the number of operation for a program such as**

  f(n) = O(g(n)) if there exist positive constants c and n_0 such that:

  $$0 \leq f(n) \leq c \cdot g(n) \text{ for all } n \geq n_0$$

- the more the information grow, the more we neglect all the weak factor in the expression
- constant complexity means that the number of operation stay the same as we vary the amount of data.
- linear complexity means that the number of operations depends on some N linear.

CODE

TIME COMPLEXITY(FAMOUS ONE)

```python
def binary_search(arr, target):
    left, right = 0, len(arr) - 1
    while left <= right:
        mid = (left + right) // 2
        if arr[mid] == target:
            return mid
        elif arr[mid] < target:
            left = mid + 1
        else:
            right = mid - 1
```

O(log n) –Logarithmic Time

```python
def get_first(arr):
    return arr[0]  #
```

O(1) – Constant Time

```python
def find_max(arr):
    max_val = arr[0]
    for num in arr:
        if num > max_val:
            max_val = num
    return max_val
```

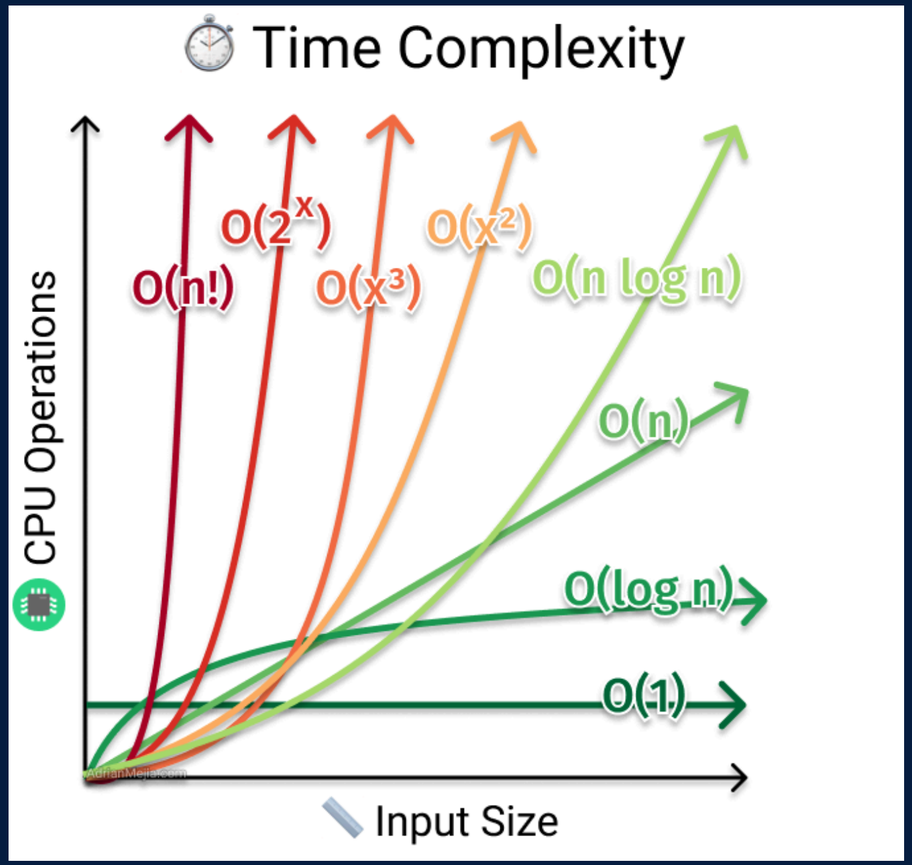
O(n log n) – Linearithmic Time

```python
def bubble_sort(arr):
    n = len(arr)
    for i in range(n):
        for j in range(n - 1):
            if arr[j] > arr[j + 1]:
                arr[j], arr[j + 1] = arr[j + 1], arr[j]
```

O(n²) – Quadratic Time

CODE

TIME COMPLEXITY



TIPS

Time complexity

- the processeur can do 10^9 operations per second
- Single loop through data → O(n)
- Nested loop → O(n²)
- dividing the each time per x → O(log_x (n))

Space complexity

Always depend on the algorithm implementation, so there isn't a specific rule for each one

CODE

C++ >>>>

what ?

C++ is a high-level, general-purpose programming language that was developed by Bjarne Stroustrup in 1983 as an extension of the C programming language. It introduces object-oriented programming features while maintaining the efficiency and flexibility of C.

C++ >>>>

why ?

- **Performance** : C++ is a compiled language, providing high-speed execution, which is crucial in contests with strict time limits.

- **Standard Template Library (STL) :** C++'s STL offers a wide range of data structures and algorithms, such as vectors, sets, maps, and priority queues, which are essential for solving problems efficiently.

- **Flexibility:** C++ allows low-level manipulation (like C) and supports object-oriented programming, giving you the best of both worlds.

CODE

C++ >>>>

your first c++ program

```cpp
#include <iostream>


int main() {
    std::cout << "Hello, World!" << std::endl;
    return 0;
}
```

`int main()`

the main function of any c++ file/program, its existence is essential for the program to run.
Only the command inside this programm will be execute.

CODE

C++ >>>>

your first c++ program

```cpp
#include <iostream>

int main() {
    std::cout << "Hello, World!" << std::endl;
    return 0;
}
```

```
std::cout<<"Hello, World"<<std::endl;
```

std:: : is namespace templates that provide some help to differentiate between two function with the same name.

cout: pronounced as “see out”, the most used function for printing in c++,

C++ >>>>

your first c++ program

```cpp
#include <iostream>

int main() {
    std::cout << "Hello, World!" << std::endl;
    return 0;
}
```

`#include <iostream>`

a preprocessor responsible for the input & output of c++ program.

CODE

C++ >>>>

Basic data types: numerical

int

- Represents whole numbers.
- Typically 4 bytes in most systems.

short

- Smaller integer, usually 2 bytes.

long

- Larger integer, usually 4 or 8 bytes.

double

- Double-precision floating-point numbers.
- Typically 8 bytes.

C++ >>>>

Basic data types: non-numerical

char

- Represents a single character.
- Typically 1 byte.

string

- usually we use an array of characters but c++ has his own string data type to handle it.
- it takes the number of characters plus one byte

bool

- Represents true or false.
- Typically 1 byte.

void

- Represents no type.
- Used for functions that do not return a value.

C++ >>>>

Basic data types: let’s practice

```cpp
#include <iostream>

using namespace std;

int main()
{
    int n;
    cin>>n;
    double b=(double) n/2;
    cout<<"the mean "<<b<<endl;
    char c;
    cin>>c;
    cout<<"choosed string: "<<c<<endl;
    return 0;
}
```

variable/data-structure:

data_type **variable_name** = *variable_value(optional)*

C++ >>>>

Control Structures: if...else if....else

conditional statement

```cpp
if (condition1) {
    // Code for condition1
} else if (condition2) {
    // Code for condition2
} else {
    // Code if none of the above
}
```

cpp

```python
if condition1:
    # Code for condition1
elif condition2:
    # Code for condition2
else:
    # Code if none of the above
```

python

C++ >>>>

Control Structures: if...else if....else

conditional statement

```cpp
if (condition1) {
    // Code for condition1
} else if (condition2) {
    // Code for condition2
} else {
    // Code if none of the above
}
```

cpp

```python
if condition1:
    # Code for condition1
elif condition2:
    # Code for condition2
else:
    # Code if none of the above
```

python

C++ >>>>

Control Structures: while

the “while” loop

```cpp
for (initialization; condition; update)
{
    // Code to execute in each iteration
}
```

cpp

```python
for i in range(start, stop, step):
    # Code to execute in each iterati
```

python

C++ >>>>

Control Structures: examples

```cpp
for (int i = 0; i < 5; i++) {
    std::cout << i << std::endl;
}
```

for loop

```cpp
int i = 0;
while (i < 5) {
    std::cout << i << std::endl;
    i++;
}
```

while loop

C++ >>>>

functions

```cpp
return_type function_name(parameter_list) {
    // Function body
    return value;  // if return type is not voi
}
```

cpp

```python
def function_name(parameters):
    # Function body
    return value  # Optional,
```

python

C++ >>>>

functions

Examples

```
int add(int a, int b = 5) {
    return a + b;
}
```

```
void increment(int &num) {
    num++;
}
```

C++ >>>>

Your turn : let’s practice

Problemset-1
(check the WhatsApp group)

C++ >>>>

Your turn : let’s practice

Be ready for your first contest

9/11/2024