# YOUR FIRST STEP TO THE **PROBLEM SOLVING** WORLD

# HELLO!

## ABDERRAZZAQ LAANAOUI

Software Engineering and Distributed Information Systems Student

You can find me at **@abderrazzaq-laanaoui**

# HELLO!

## ACHRAF TAFFAH

Software Engineering and Distributed Information Systems Student

You can find me at **@achraf-taffah**

# Objectifs

1. BENEFITS OF CP FOR AN IT STUDENT
2. WHERE TO FIND PROBLEMS
3. HOW CAN I SOLVE MY FIRST PROBLEM?
4. COMMUN ERRORS
5. COMPLEXITY

# Benefits of CP for an IT student

1

Problem solving in programming helps to gain more knowledge over coding and programming.

2

These problem solving skills also help to develop more skills in a person and build a promising career.

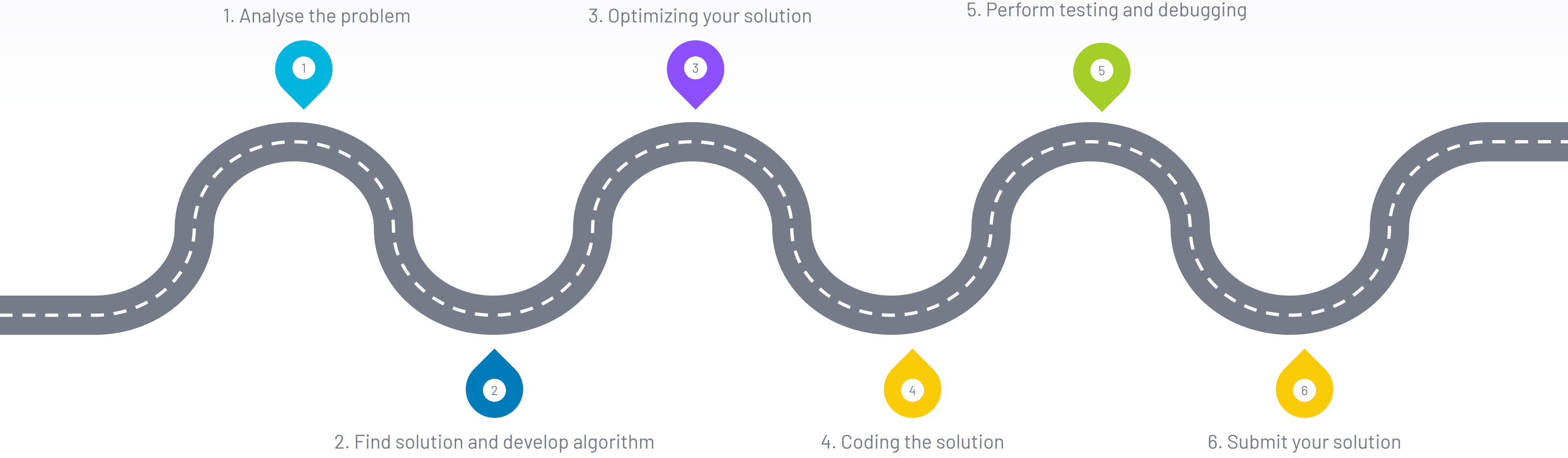
3

These skills also help to find the solutions for critical and complex problems in a perfect way.

4

Most of the companies are looking for people with good problem solving skills.

# How can i solve my first problem?


1. Analyse the problem
1
3. Optimizing your solution
3
5. Perform testing and debugging
5
2
2. Find solution and develop algorithm
4
4. Coding the solution
6
6. Submit your solution

## Online judges

HackerRank

CODEFORCES

{M</>I ARENA}

LeetCode

# Complexity

1

## Types of complexity

# Cognitive complexity

Is it important in problem solving?

Reference:

https://www.sonarsource.com/docs/CognitiveComplexity.pdf

## Space complexity

How to evaluate space complexity?

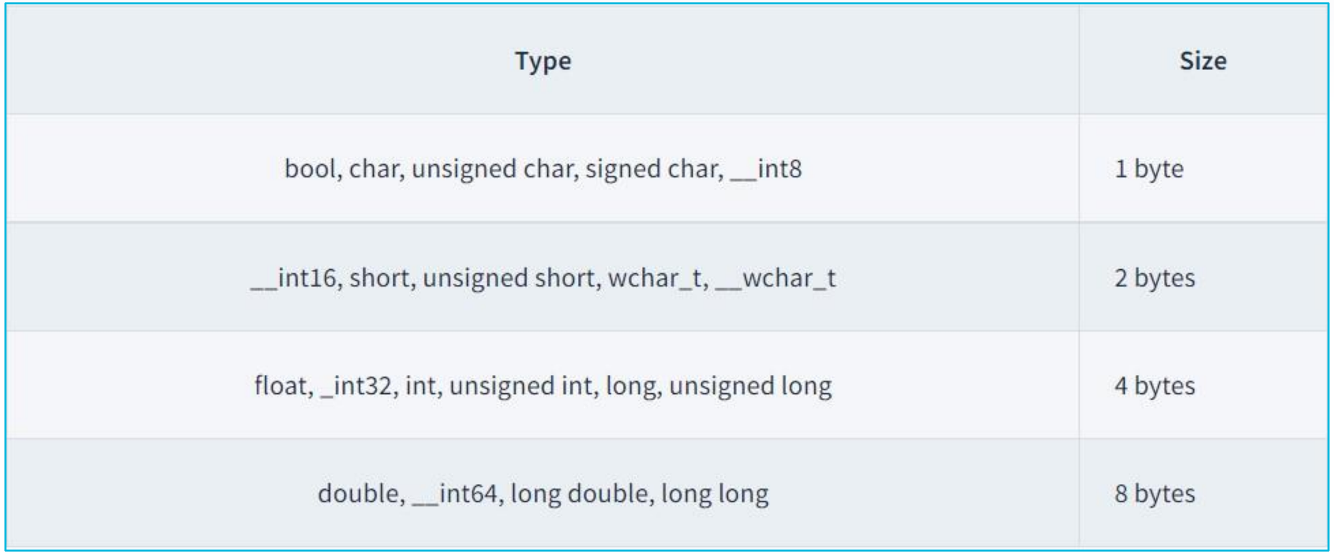
Reference:

https://www.scaler.com/topics/data-structures/space-complexity-in-data-structure/

## How to evaluate space complexity?

We need to know the amount of memory used by different types of datatype variables,program instructions, constant values and few other things like function call, recursion stack(in recursion case) in order to calculate the space complexity. The amount of memory used by different types of datatype variables varies by os, but the way of calculating the space complexity continues to remain the same.

## Language C compiler takes the following space:



| Type | Size |
|---|---|
| bool, char, unsigned char, signed char, __int8 | 1 byte |
| __int16, short, unsigned short, wchar_t, __wchar_t | 2 bytes |
| float, _int32, int, unsigned int, long, unsigned long | 4 bytes |
| double, __int64, long double, long long | 8 bytes |

## Example : Addition of Numbers

```
{
    int a = x + y + z;
    return (a);
}
```

So in the above example, there are 4 integer variables those are a, x, y, z so they will take 4 bytes(as given in the table above) space for each variable, and extra 4-byte space will also be added to the total space complexity for the return value that is a.

Hence, the total space complexity = 4*4 + 4 = 20 bytes

## Time complexity

How to evaluate time complexity?

Reference:

https://www.scaler.com/topics/data-structures/time-complexity-in-data-structure/

## How to evaluate space complexity?

We need to know the amount of memory used by different types of datatype variables,program instructions, constant values and few other things like function call, recursion stack(in recursion case) in order to calculate the space complexity. The amount of memory used by different types of datatype variables varies by os, but the way of calculating the space complexity continues to remain the same.

# Constant time O(1)

When an algorithm is not reliant on the input said to have constant time with order O(1).

The runtime will always be the same, regardless of the input size n.

```
main( ){
    int a;
    printf("Enter value of a=");
    scanf("%d",&a);
    printf("a=%d",a);
}
```

## Linear time O(n)

When the running time of an algorithm rises linearly with the length of the input, it is said to have linear time complexity.

When a function checks all the values in an input data set, it is said to have Time complexity of order O(n)

```
main( ){
        int a[5]={2,3,5,9};
        printf("Elements of a=\n");
        for(int i=0;i<5;i++)
                printf("a= %d",a);
}
```

## Logarithmic time O(Log(n))

When an algorithm lowers the amount of the input data in each step, it is said to have a logarithmic time complexity.

Binary trees and binary search functions are some of the algorithms with logarithmic time complexity.

```
int binarySearch(int arr[ ]. int l, int r, int x){
  if (r >=1){
    int mid = l + (r-1)/2;
      if (arr[mid] == x)
       return mid;
      if (arr[mid] > x)
       return binarySearch(arr, l, mid-1, x);
       return binarySearch(arr, mid + 1, r. x);
}}
```

## Quadratic time O(n2)

When the execution time of an algorithm rises non-linearly (n2) with the length of the input, it is said to have a quadratic time complexity.

When the execution time of an algorithm rises non-linearly (n2) with the length of the input, it is said to have a quadratic time complexity.

```
for(int i=0; i<n; i++){
        for(int j=0; j<n; j++){
                printf("%d",arr[i][j]);
        }
}
```

# THANKS!

## Any questions?

You can find us at:

- abderrazzaq.laanaoui@gmail.com
- taffahachraf184@gmail.com