Time and Space complexity and other Pre-Requisites for Competitive Programming

Generally, the time limit for running your code on platforms like Hackerrank, Atcoder, Codeforces, etc. is 1-2 seconds. So, we need to write an efficient code that passes this time limit constraint.

Big Oh (O)

- It represents the **upper bound of a function**
- Used to approximate time complexity of a code

If you have function f(x), then consider a function g(x) such that
$f(x) <= c.g(x)$
For all values of $x>=x_0$ for some value of x_0
Then we say, $f(x) = O(g(x))$

Eg. 1:
If $f(n) = n - 2$
$f(n) <= 8.n$
$f(n) <= 8.g(n)$
Where $g(n) = n$
By definition, $f(n) = O(n)$

Eg. 2: If $f(n) = 3n^2 + 5n + 8$
By definition of Big Oh, $f(n) = O(n^2)$
[**In polynomial functions, only see the highest degree term to find Big Oh**]

Eg. 1 Consider an array A of size n

```
for(int i=0; i<n; i++)
{
   cout<<A[i];
}
```

Find time complexity of this code.

i=0; will run only 1 time
i<n; will be run n+1 times
i++ will be run n times
cout<<A[i]; will be run n times

Time complexity = 1 + n+1 + n + n
= 3n + 2
= **O(n)**

Eg. 2 Find time complexity of this code.

```
int val;
bool found=false;
for(int i=0; i<n; i++)
{
    if ( A[i] == val )
    {
        found=true;
        break;
    }
}
```

We always consider the worst case scenario in finding time complexity of a code.
Here, the worst case is when the array A doesn't contain the value val. So, loop will run n times.

Thus, Time complexity: **O(n)**

Eg. 3 Find time complexity of this code.

```
int b;
int a = 2*b;
cout<<b;
```

Time complexity: **O(1) [constant]**

Eg. 4 Find time complexity of this code.

```
for(int i=0; i<n; i++)
{
    for(int j=0; j<i; j++)
    {
       cout<<2;
       ...
    }
}
```

i=0; inner loop will run 0 times
i=1; inner loop will run 1 times
i=2; inner loop will run 2 times
….
i=n-1 ; inner loop will run n-1 times

Total steps = 0 + 1 + 2 + …. + n-1
  = (n * (n-1)) /2
  = **O(n^2)**

Eg. 4: Find time complexity of this code.

```
for(int i=1; i<=n; i=i*2)
{
sum=sum+i;
}
```

For i=1, 2, 4, 8, 16, ….., 2^k
Let us assume that it breaks out of loop in k steps

1.(2^k) > n
2^k > n
k approximately log2 (n)

Time complexity: O(log2 (n))

Eg. 5 Find time complexity of this code.

```
for(int i=n; i>0; i--)
{
    for(int j=0 ; j<i; j=j+2)
    {
       ......
       cout<<1;
    }
}
```

For i=n, inner loop will execute n/2 times
i= n-1, inner loop will execute (n-1)/2 times
i=n-2, inner loop will execute (n-2)/2 times
…..
i=1, inner loop will execute 1 time

Time complexity: n/2 + (n-1)/2 + (n-2) / 2+..........
= **O(n^2)**

HW-1: Find time complexity of this code.
(Find answer at the end of this doc)

```
for(int i=1; i*i<=n; i++)
{
cout<<2;
}
```

<u>Important:</u> In 1 second, only 10^7 - 10^8 operations can be performed.

Eg. 6 : Suppose, n<=10^5
(i) You have written a code with time complexity O(n^2). Find whether your code will pass time limit of 1 second.

Sol: In worst case, code takes 10^{10} operations to perform. But, this is greater than 10^7- 10^8 . So, it is slow. It will not pass the time limit of 1 second.

(ii) If you a write code with time complexity O(n log n) Find whether your code will be able to pass time limit of 1 second.

Sol: In worst case, N log N = 10^5.log(10^5) <= 10^8

So, It will pass the time limit of 1 second.

Important: In general, $O(1) < O(\log N) < O(\text{root } N) < O(N) < O(N \log N) < O(N^2) < O(N^3) < \ldots.. O(N^{100}) < O(2^N)$

Common errors in online platforms

1. **Compiler error (CE)**

- Indicated by compiler itself with the line number in which there is error

2. **Wrong Answer (WA)**

eg. Yes not = YES

- Read the input and output format in the question very carefully

3. **Time Limit Exceeded (TLE)**

Time limit is generally 1 second. And if your code is slow to pass this time limit, you will get this error.

- You can also use Fast input / output with cin, cout:

```
int main()
{
ios_base::sync_with_stdio(false);
cin.tie(NULL);
cout.tie(NULL);
....
// All your code after this
}
```

4. Runtime error

(a) If you are accessing an invalid element of an array.

```
int arr[100];
cout<<arr[1000]; // runtime-error
cout<<arr[-1]; // runtime-error
```

(b) When you divide by 0

```
cout<<a/0;
```

(c) Overflow [Important to prevent such errors]
Try running this code:

```
#include <bits/stdc++.h>
using namespace std;

int32_t main()
{

```

```cpp
    int a=1000000000;
    int b=1000000000;
    int ans=a*b;
    cout<<ans;
    return 0;
}
```

// Output: -1486618624 (something like this)
// Surprising, right ?
Why this happens?
int can store integers only upto 10^9 approximately.
Numbers greaters than this, can't be stored in an int variable.
For bigger integers , upto 10^{18}, use long long variable.

```cpp
    int a=1000000000;
    int b=1000000000;
    long long ans=a*b;
    cout<<ans;
```

// Output: -1486618624 (something like this)
Still, it will give the same wrong answer
Because you need to convert the integer to long long, during the multiplication also.

Now, try running this code:

```
    int a=1000000000;
   int b=1000000000;

   long long ans = (long long)a * b;
    cout<<ans;
```

Now, you will get correct answer

One more method, is **always use long long variables**.

```
    long long a=1000000000;
   long long b=1000000000;

   long long ans = a * b;
   cout<<ans;
```

// This is also correct

Checking equality of floating point numbers (decimal numbers)

Never compare floating point numbers with == sign

```
float a=1.00000001;
float b=1.00000000;
```

```
if (a==b)
{
cout<<"equal";
}
else
{
cout<<"Not equal";
}
```

// The above code may give wrong answer in some places due to lack of precision in float operations

We use this method for comparison:

```
const float eps = 0.000001; // 1e-6
float a=1.00000001;
float b=1.00000000;
if ( abs(a-b) < eps )
{
   cout<<"Equal";
}
else
{
cout<<"Not equal":
}
```

Space Complexity:

It denotes the Big Oh of space taken by variables in a program, etc.

Eg. 1

```
int arr[n];
```

This takes O(n) space

Eg. 2

```
int arr[n][m];
```

This takes O(n*m) space

You can't declare an integer global array of size > 10^7 or 10^6. So, while declaring an array, take care of the size.
Otherwise, you get a **MLE (Memory limit exceeded) error** on platforms like hackerrank, atcoder, etc.

Ans of HW-1: **O(sqrt(N))**
Where sqrt(N) = square root of N