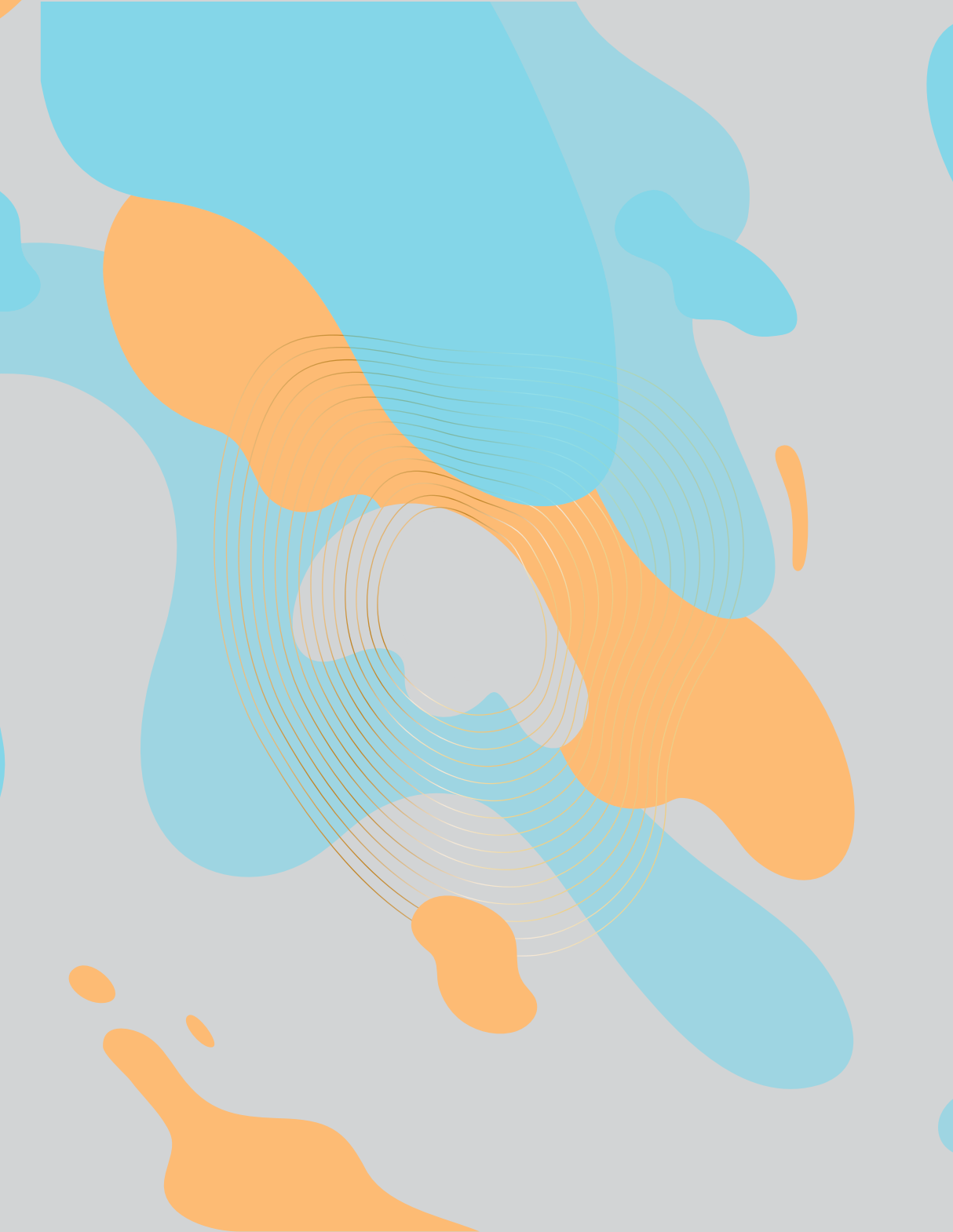

Time Complexity

Introduction to Problem Solving I

Additional Problems

Q1. Find Time Complexity - 4

Solved

Using hints except Complete Solution is Penalty free now

What is the time complexity of the following code snippet

```
int func(int n){
    int s = 0;
    for(int i = 1 ; i*i*i <= n ; i++){
        s = s + i;
    }
    return s;
}
```

$\Rightarrow \quad i^3 = n \quad \Rightarrow \quad i = (n)^{1/3}$

$\Rightarrow$ Since `i` starts from 1

$\Rightarrow$ Number of iterations from $1 \rightarrow n^{1/3}$

$= [1, \sqrt[3]{n}] = (n)^{1/3}$

$\therefore O((n)^{1/3})$

Q2. Complexity of Single Loop

Solved

Using hints except Complete Solution is Penalty free now

Use Hint

What is the complexity of the following code snippet ?

```
int ans = 0;
for (int i = 0; i < n; i++) {
    ans += i * i;
}
return ans;
```

Clearly loop runs $[0, n) = n$ times

$\Rightarrow O(n)$

Q3. Find Time Complexity - 9

Solved

Using hints except Complete Solution is Penalty free now — Use Hint

What is the time complexity of the following code snippet

```
for(int i = 1 ; i <= n ; i++){
    for(int j = 1 ; j <= 3^i ; j++){
        print(i + j);
    }
}
```

i	j	iterations
1	$[1, 3]$	3
2	$[1, 3^2]$	9
3	$[1, 3^3]$	27
⋮	⋮	⋮
n	$[1, 3^n]$	3^n

$\sum$ iterations $= 3^1 + 3^2 + 3^3 + \ldots + 3^n$ → GP, $a = 3$, $CR = 3$

$= \frac{a(r^N - 1)}{r - 1} = \frac{3(3^n - 1)}{3 - 1}$

$= \frac{3}{2}(3^n - 1) \Rightarrow O(3^n)$

Q4. Identifying O(1) Complexity

Solved

Using hints except Complete Solution is Penalty free now — Use Hint

If an algorithm has a time complexity of O(1), then the complexity of it is ?

⇒ Constant i.e the algorithm will always run a specific number of times and does not depend on the input size

Q5. Identifying O(log n) Complexity

Solved

Using hints except Complete Solution is Penalty free now — Use Hint

If for an algorithm time complexity is given by O(log2n) then complexity will:

- constant
- polynomial
- exponential
- none of the mentioned

polynomials are sums of terms of the form ax^n where a is constant and n is a non negative integer.

∴ Expressions in the form of $\log_x x$ can not be considered polynomial

Q6. Algorithm Complexity Classification

Solved

Using hints except Complete Solution is Penalty free now — Use Hint

If an algorithm has a time complexity of O(n), then the complexity of it is ?

⇒ Linear

Q7. Time-Complexity-12

Solved

Using hints except Complete Solution is Penalty free now

Use Hint

If for an algorithm time complexity is given by O((3/2)^n) then complexity will:

- constant
- quardratic
- exponential
- none of the mentioned

No—an expression like 2^x is *not* a polynomial in x.

A polynomial in x is an expression of the form

$a_n x^n + a_{n-1}x^{n-1} + \ldots + a_1 x + a_0,$

where a_i are constants and n is a nonnegative integer. Notice that in a polynomial, x must appear only as an integer power.

However, in 2^x, the *exponent* is x rather than a fixed integer. That makes 2^x an *exponential function*, not a polynomial.

Q8. NESTED_CMPL

Solved

Using hints except Complete Solution is Penalty free now

Use Hint

What is the time, space complexity of following code :

```
int a = 0, b = 0;
for (i = 0; i < N; i++) {
    for (j = 0; j < N; j++) {
        a = a + j;
    }
}
for (k = 0; k < N; k++) {
    b = b + k;
}
```

This loop has roughly N^2 iterations

This loop has N iterations

$\Rightarrow O(N * N)$ or $O(N^2)$

$\Rightarrow$ Space : $O(1)$

Q9. Time Complexity - M4

Solved

Using hints except Complete Solution is Penalty free now — Use Hint

Find the Time Complexity of the following function solve :

```
public void solve(int N) {
    for(int i = 0; i < Math.pow(2,N); i++) {
        int j = i;
        while(j > 0){
            j -= 1;
        }
    }
}
```

i	j	iterations
0	(0, 0)	0
1	(0, 1]	1
2	(0, 2]	2
3	(0, 3]	3
⋮	⋮	⋮
2^N	(0, 2^N]	2^N

$\sum$ iterations $= 1 + 2 + 3 + \dots + 2^N \Rightarrow$ Sum of first 2^N natural numbers

$= \frac{2^N(2^N+1)}{2} = 2^{N-1}(2^N+1) = 2^{2N-1} + 2^{N-1}$

$\Rightarrow O(2^N)$

Q10. Find Time Complexity

Solved

Using hints except Complete Solution is Penalty free now — Use Hint

What will be the time complexity of the above function where n is a positive integer?

```
public void function (int n) {
    while (n > 0) {
        n++;
        n -= 2;
    }
}
```

n++; n -= 2; can be simplified into n = n-1;

→ say n = 2

↘ n = 1 - 1 = 0 ↘ exit = 1 iteration

↘ say n = 2

① n = 2 - 1 = 1

② n = 1 - 1 = 0 ↘ exit = 2 iterations

→ Say n = 3

① n = 3 − 1 = 2

② n = 2 − 1 = 1

③ n = 1 − 1 = 0 → exit loop

→ So for n = N iterations, the loop will run N times

⇒ O(N)

Q11. Time Complexity-iii Solved

Using hints except Complete Solution is Penalty free now — Use Hint

What is the time complexity of the following code snippet?

```
for(int i = 0; i < n; i++){

    for(int j = i - 1; j >= 0; j++){

            ans += i + j ;

    }

}
```

I think these two expressions gave away the trick!

→ for i = 0 → nested loop won't run

i = 1 → j = 0

j++ → j = 1

⋮

∞ → there is no relevant exit condition

Code will run indefinitely

Q12. Time Complexity - 8 Solved

Using hints except Complete Solution is Penalty free now Use Hint

What is the time complexity of the following code snippet?

```
int sum = 0;
for(int i = 0; i <= N; i ++){
   for(int j = i; j <= N && j > i; j++){
      sum += i;
   }
}
```

Again, I think this just gave away the trick!

→ The nested - inner loop will never run

→ it starts from $j = i$, but the condition $j > i$ must also hold true to enter inside the loop!

→ Therefore, only the outer loop will run i.e N+1 times

⇒ $O(N)$