Examples

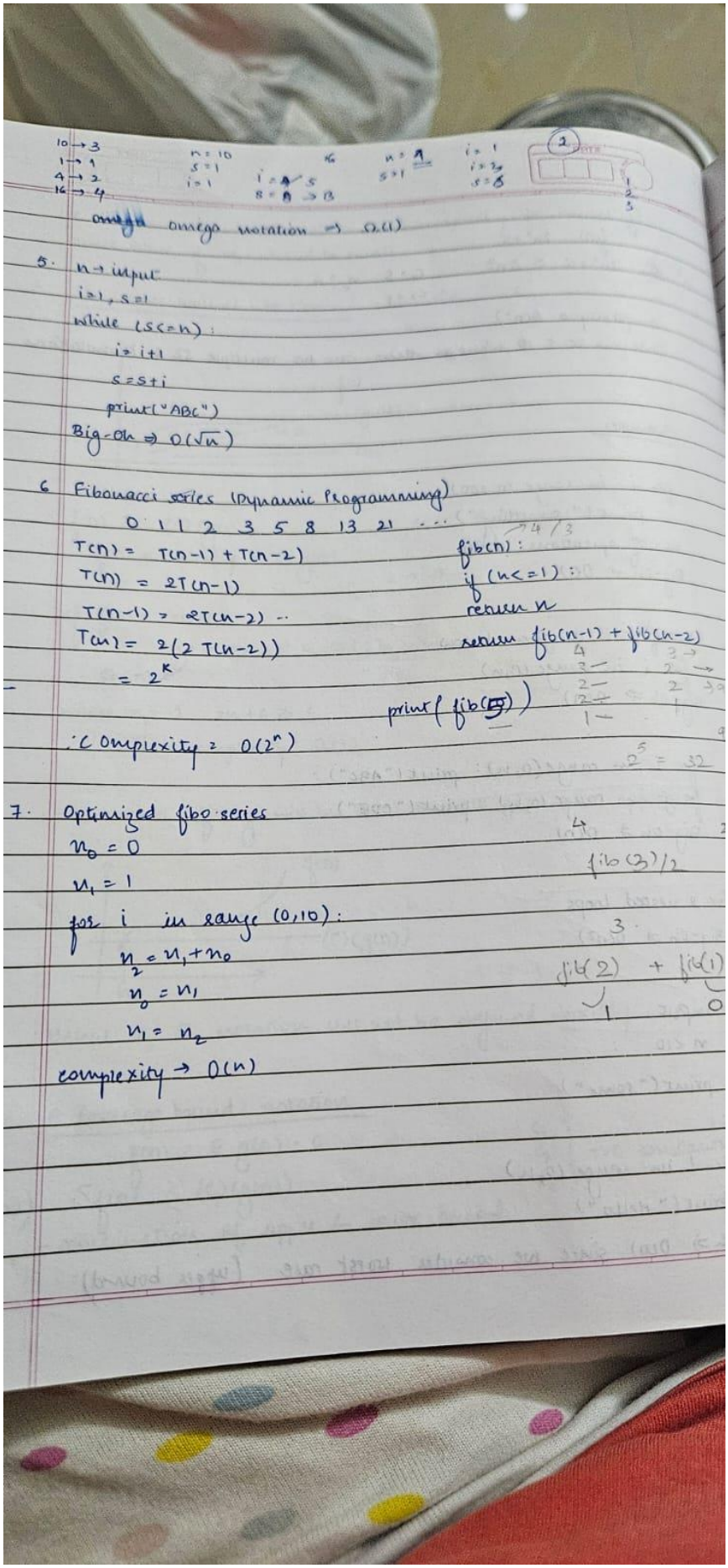
1. for i in range (0, 200)
   print ("something")
   no. of operations ⇒ 200
   Big-Oh ⇒ O(1)

   but if it is as given below
   for i in range (0, n)
   Big-Oh ⇒ O(n)

2. for i in range (0, n): print ("ABC")
   for j in range (0, n): print ("CDE")
   Big-Oh ⇒ O(n)

3 For 2 nested loops
   Big-Oh ⇒ $O(n^2)$

4 n = input
   if n < 10 :
      print ("some")
   else :
      for i in range (0, n)
         Print ("Hello")
   Big-Oh ⇒ O(n) since we consider worst case (upper bou



omega notation → $\Omega(1)$

5. n → input
i = 1, s = 1
while (s <= n):
    i = i+1
    s = s+i
    print("ABC")
Big-Oh ⇒ $O(\sqrt{n})$

6 Fibonacci series (Dynamic Programming)
0 1 1 2 3 5 8 13 21 ...

$T(n) = T(n-1) + T(n-2)$
$T(n) = 2T(n-1)$
$T(n-1) = 2T(n-2)$ ..
$T(n) = 2(2T(n-2))$
$= 2^k$

∴ Complexity = $O(2^n)$

```
fib(n):
if (n<=1):
return n
return fib(n-1) + fib(n-2)
print( fib(5) )
```

7. Optimized fibo series
$n_0 = 0$
$n_1 = 1$
for i in range (0,10):
    $n_2 = n_1 + n_0$
    $n_0 = n_1$
    $n_1 = n_2$
complexity → $O(n)$

$= \log\left(\frac{n(n+1)}{2}\right)$

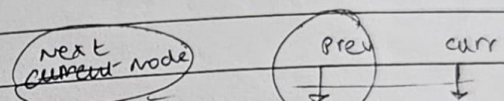
$= \log(n(n-1)(n-2)(n-3)\cdots 1)$

$= \log n!$

Complexity = $O(\log n!)$

```
def fibo(n, k):
    print(k)
    if n <= 1
        return n
    else :
        return fibo(n-1, 'left') + fibo(n-2, 'right')

print(fibo(5, 'c'))
```



8. for (i=1; i*i<n; i++)
$O(\sqrt{n})$

9. for (i=n/2; i<n; i++)
  for (j=1; j+n/2<n; j++)
    for (k=1; k<n; k=2*k)
      ↳ $2^p > n$
      $p = \log_2(n)$

$O(n^2 \log n)$

n=100
1 -
2 -
4 -
8 -
16 -
32 -
64 -
128

10. for (i=1; i<n; i++) → O(n)
  for (j=1; j<n; j=j+i)

Adding no. of times inner loop runs = $n + \frac{n}{2} + \frac{n}{3} + \cdots + \frac{n}{n}$

$= n\left[1 + \frac{1}{2} + \frac{1}{3} + \cdots + \frac{1}{n}\right]$

$= n \log n$

$\therefore O(n^2 \log n)$

n = 10

↳ 1 → 10 (n)
2 → 1,3,5,7,9 (5) (n/2)
3 → 1,4,7,(3) (n/3)
4 → 1,5,9 (3)
5 → 1,6,(2)
6 → 1,7 (2)
7 → 1,8 (2)
8 → 1,9 (2)
9 → 1,10 (1) (1)

j+2
j+3

11

test(n)
if(n>0)
  print("")
  test(n-1)
O(n)

n = 5
5, 4, 3, 2, 1

12 test(n)
if n>0
  for i=0; i<n; i++ → O(n^3)
  {
  }
test(n-1) $O(n^2)$

n = 5

n	-n	n
4	-n-1	n+n-2+n-3
3	-n-2	
2	-n-3	
1	-1	

$\frac{n(n+1)}{2}$

13.

test(n)
if n > 0
for (i = 1 ; i < n ; i++) → 2/3
{
}
test(n-1)

$\frac{2n}{2} + \frac{(n-2)}{2} + \frac{(n-4)}{2} + \cdots \sqrt{n}$

$\frac{1}{2}[2n + \frac{n(n+1)}{2}]$

$T(n) = \sqrt{n} + T(n-1)$
$= \sqrt{n} + \sqrt{n-1} + \cdots \sqrt{n}$

$O(n\sqrt{n})$

n = 10
n = 5
1, 2 ⇒ 2
n = 4
n = 3
1, 2
n = 10
1, 2, 3, 4 (4)
1, 2, 3 (3)
n = 20
1, 2, 3, 4, 5, 6, 7, 8, 9 (9)
n = 5
1, 4 (2)

14 test(n)
if n > 0
for (i = 1 ; i < n ; i = i × 2)
{
}
test(n-1)
$2^k = n$
$k = \log_2 n$

$O(\log_2 n)$

Iteration
1 → 1
2 → 2^2
3 → 2^3
4 → 2^4
⋮
k → 2^k

Stopping condn ⇒ $2^k = n$
$k = \log_2 n$

$T(n) = \log n + \log_2(n-1) + \log_2(n-2) + \cdots + \log_2(n-k)$
$T(n-k)$