Lecture TC-1

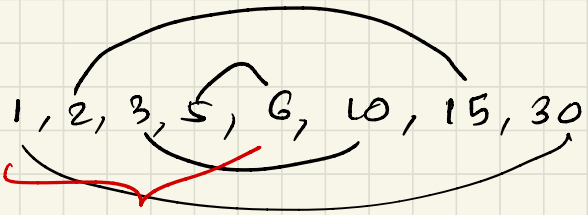
(1)

```
def isPrime(num):
    n = int(num)

    count = 0
    for i in range(1, √n+1)
        if n % i == 0:
            count += 2

    if count == 2
        return True
    else:
        return False
```

num = 30



$\sqrt{n}+1$

$5.x + 1$

range (1, 6.x)

Lecture TC-2

Big O notations

$$\log N < \sqrt{N} < N < N\log N < N\sqrt{N} < N^2 < N^3 < 2^N < 3^N < N!$$

A.P. & G.P.

$T_n = a + (n-1)d$

$T_K = ar^{K-1}$

$S_n = \frac{n}{2}[2a + (n-1)d]$

$S_K = \frac{a(r^K - 1)}{r-1}$

Lecture 3 : Introduction to Arrays

(2) Given n elements in a list. count no. of elements which have at least 1 element greater than itself present in the list

[2, 5, 1, 4, 8, 0, 8, 1, 3, 8]

→ 8 - 3 = 5

Brute Force Approach :

```
cnt = 0
for i in range(len(l)):
    for j in range(len(l)):
        if l[j] > l[i]
            cnt += 0
            break
return cnt
```

$O(N^2)$

Optimized Algorithm :

1. find largest value & no. of times it occurs
2. Return n - cnt ← cnt

```
max = l[0]
for i in range(len(l)):
    if l[i] > max:
        max = l[i]

cnt = 0
for i in range(0, len(l)):
    if l[i] == max:
        cnt += 1

return n - cnt
```

O(N)

③ Given N array elements, check if there exists a pair (i, j) s.t. arr[i] + arr[j] == k

arr : [3, -2, 1, 5, 6]

(0,0) (0,1) (0,2) (0,3) (0,4)

(1,1) (1,2) (1,3) (1,4)

(2,2) (2,3) (2,4)

(3,3) (3,4)

(4,4)

```
for i in range(0, len(l)-1):
    for j in range(i+1, len(l)-1):
        if arr[i] + arr[j] == k:
            return True
return False
```

for i = 0 j = 1, N
i = 1 j = 2, N

$O(N^2)$

iterations = $\frac{n(n-1)}{2}$

④ Given a list, reverse the list

Brute force

```
def reverse_list(l):
    return l[::-1]
```

O(N)

S(N)

Optimized Algorithm

swapping

```
n = len(l)
for i in range(0, n//2):
    l[i], l[n-1-i] = l[n-1-i], l[i]
```

O(N) S(1)

⑤ Given the list, rotate the list (right rotation)

A = 1, 2, 3, 4 → 4, 1, 2, 3 → 3, 4, 1, 2

B = 2

```
B = B % n
part 1 = A[ : (n - B)]
part 2 = A[ -B : ]
for x in part 1 :
    part2 append (x)
```