Revision.

1. Double occ.
2. Mirror of me (aditive inverse).

1. Double occ.

A → 1 2 3 4 5   occ. == 2. A[i].

B → 1 1 2 3 3 | 1 3

2. Mirror Image.

A → 1 2 -2 3 -5

B → -1 2 -3 4 5

$1 + (x) = 0$

$x = -1$

| 1 -2 3

$-2 + x = 0$

$x = 2$

Time. Comp.   i/p v/s oper.

$n^2 + 2n + K$

O(k) ≈ O(1) → const   100.

≠1

int a = 10;

+ - * ÷   O(1)

scn. nextInt → O(1).   O(100)

Syso -   O(1)

if

n → n

Scanner scn = new Scanner

O(n) → linear

time ∝ operations.

1. i/p → for 'n' sized array.
2. print the entire array. (1D).
3. find a number in an array using linear O(n)
4. iterating entire array.
5. for (i=0 → i<n ; i++).

$O(n^2)$

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

    }
}
```

i=0 n

i=1 n

i=2 n

n times

⋮

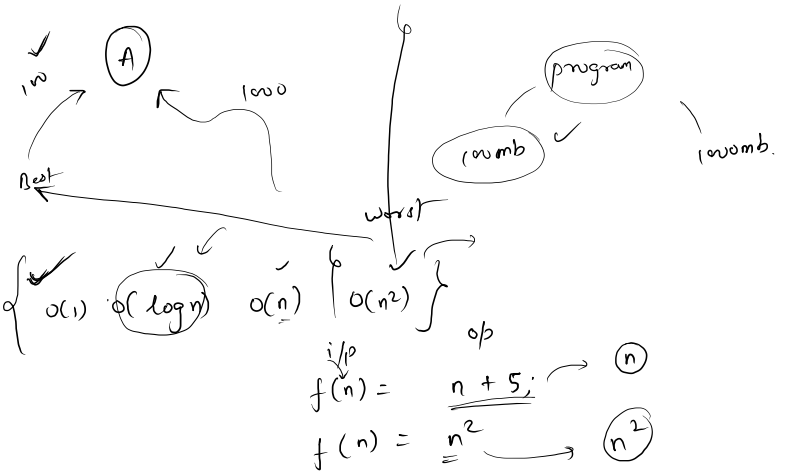
n-1 n

1 → n

$n \to n^2$.

Time Comp.

Space Complexity




```java
public class Main
{
    public static int linearSearch(int [] arr, int key){
        for(int i = 0; i < arr.length; i++){
            if(arr[i] == key)
            {
                return i;
            }
        }
        return -1;
    }
    public static void main(String[] args) {
        int n = scn.nextInt();
        //arr
        int key = 3;
        int ans = linearSearch(arr, key);
        system.out.println(ans);
```

→ O(n).

```java
public class Main
{
    public static int linearSearch(int [] arr, int key){
        for(int i = 0; i < arr.length; i++){
            if(arr[i] == key)
            {
                return i;
            }
        }
        return -1;
    }
    public static void main(String[] args) {
        int n = scn.nextInt();
        //arr
        for(int i = 0; i < n; i++){
            int key = 3;     //random
            int ans = linearSearch(arr, key);
            System.out.println(ans);
        }
```

O(n)

O(n * k)

= O(n).

k + n

O(n)

1 → n

2 → n

3 → n

4 → n

.

.

.

nth → n

1 → n

n → n²

O(n²).

```java
public static void loopCalling(n){
    for(int i = 0; i < n; i++){
        for(int j = 0; j < n; j++){

        }
    }
}

public static void main(String[] args) {
    int n = scn.nextInt();
    //arr
    for(int i = 0; i < n; i++){
        loopCalling(n);
    }
```

$O(n^2)$

$i=0 \rightarrow n$
$i=1 \rightarrow n$
$\vdots$
$n \rightarrow n$

$i=0 \rightarrow O(n^2)$
$i=1 \rightarrow n^2$
$i=2 \rightarrow n^2$
$\vdots$
$n \rightarrow n^2$

$1 \rightarrow n^2$
$n \rightarrow n \cdot n^2 = n^3$

$O(n^3)$

```java
public static void loopCalling(){
    for(int i = 0; i < n; i++){
        for(int j = 0; j < 10; j++){

        }
    }
}
```

$n = 5,000$

1 → 10 (k)
2 → (k)
3 →
4
5

$1 \rightarrow k$
$n \rightarrow n.k = 10n$

$O(n)$

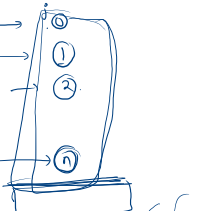
```java
public static void loopCalling(){
    for(int i = 0; i < n; i++){
        for(int j = 0; j < i; j++){

        }
    }
}
```

n^2

$n = 500$

$i=0$
$i=1$
$i=2$
$\vdots$
n



$1+2+3+\dots+n$

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

$O(n^2)$

Avg $\frac{1}{n}$ o/p

$\frac{\text{T o/p}}{\text{T iteral}} = \frac{O(n^2)}{n} = O(n)$

$i = 0$ ——— 10

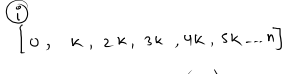
$0 \rightarrow 10$

$n = 5$
$n = 1000.00$

$k.k \Rightarrow k^2 \rightarrow c$

$O(c) \approx O(1)$

```java
public static void loopCalling(){
    for(int i = 0; i < n; i += k){
        System.out.println("Geekster");
    }
}
```

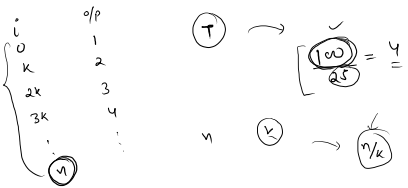


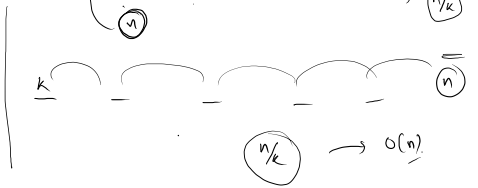
Total of it

$\frac{n}{k}$

1 single → how of times it will work.

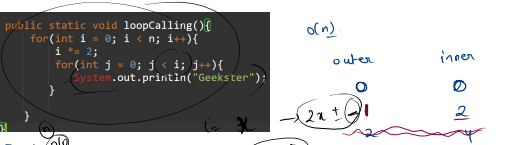
$O(n)$

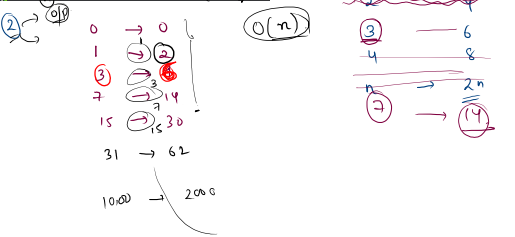




$\frac{n}{k}$ → $O(n)$

```java
public static void loopCalling(){
    for(int i = 0; i < n; i++){
        i += 2;
        for(int j = 0; j < i; j++){
            System.out.println("Geekster");
        }
    }
}
```






```java
class NestedLoop {
  public static void main(String[] args) {
    int n = 10;
    int sum = 0;
    double pie = 3.14;

    for (int var = 0; var < n; var = var + 3) {
      System.out.println("Pie: " + pie);
      for (int j = 0; j < n; j = j + 2) {
        sum++;
        System.out.println("Sum = " + sum);
      }
    }
  }
}
```