NestedLoops.java

```
public class NestedLoops {

    public static void main(String[] args) {


        //for -- outer
            //for --inner


        //00 01 02 03 04 05
        //10 11 12 13 14 15
        //20 21 22 23 24 25
        //30 31 32 33 34 35
        //40 41 42 43 44 45
        //50 51 52 53 54 55


        //i--> 0 to 5
        //j--> 0 to 5
        for(int i=0; i<=5; i++) {

            for(int j=0; j<=5; j++) {
                System.out.println(i+""+j);
            }

        }
```

00

01

02

And so on.

```java
        //40 41 42 43 44 45
        //50 51 52 53 54 55


        //i--> 0 to 5
        //j--> 0 to 5
        for(int i=0; i<=5; i++) {

            for(int j=0; j<=5; j++) {
                System.out.print(i+""+j+" ");//00 01 02 03

            }

            System.out.println();

        }
```

00 01 02 03

10 11 12 13

And so on.

```java
        // i--> 0 to 5
        // j--> 0 to 5
        // 6*6 --> m*n
        for (int i = 0; i <= 5; i++) {

            for (int j = 0; j <= 5; j++) {
                System.out.print(i + "" + j + " ");//00
                break;

            }

            System.out.println();

        }
```

00

10

20

And so on.

```java
// i--> 0 to 5
// j--> 0 to 5
// 6*6 --> m*n
for (int i = 0; i <= 5; i++) {
    for (int j = 0; j <= 5; j++) {
        System.out.print(i + "" + j + " ");//00
    }
    System.out.println();
    break;

}
```

00 01 02 03 04 05

```java
// 50 51 52 53 54 55

// i--> 0 to 5
// j--> 0 to 5
// 6*6 --> m*n
for (int i = 0; i <= 5; i++) {
    for (int j = 0; j <= 5; j++) {
        System.out.print(i + "" + j + " ");
        break;
    }
    System.out.println();
    break;

}
```

00

Time complexity-

```java
NestedLoops.java    TimeComplexity.java
1 package javasessions;
2
3 public class TimeComplexity {
4
5     public static void main(String[] args) {
6
7         //Tc: Big Oh()--Big O(n)
8
9
10        //1.
11        int i = 1;
12        System.out.println(i);
13
14        //O(1) --constant time
```

For loop time complexity-

```java
15
16        //2. 100 secs -->0.00001
17        for(int p = 1; p<=10; p++) {
18            System.out.println(p);
19            System.out.println(p);
20            System.out.println(p);
21
22        }
23
24        //1+n+n+n => 3n+1 --> Linear Equation
25        //3n+1 ==> 3n --> n --> O(n)
```

P runs only once so it is 1.

P<=10 -> runs as per the number of inputs.

P++ -> runs as per the number of inputs.

Since p=1 is very small, we can neglect it.

Also in 3n, we can remove constant as tomorrow the lines of code can change to 100 so it becomes 100n.

N □ number of times code runs / number of elements to run.

```
//0(1) --constant time

//2. 100 secs -->0.00001
for(int p = 1; p<=10; p++) {
    System.out.println(p);
}

//1+n+n+n => 3n+1 --> Linear Equation: n+c
//3n+1 ==> 3n --> n --> O(n)
```

```
//
int k = 1;
while(k<=10) {
    System.out.println(k);
    k++;
}

//1+n+n+n==>3n+1==>3n==>n==>O(n)
```

```
//m*n
for (int m = 0; m <= 5; m++) {
    for (int n = 0; n <= 5; n++) {
        System.out.print(m + "" + n + " ");
    }
    System.out.println();
    System.out.println("Hi");

}

//(1+n+n+n+n)(1+n+n+n)==>(1+4n)(1+3n)==>1+4n+3n+12n^2 ==> 12n^2+7n+1 ==> quadratic equation
//12n^2+7n+1 ==>12n^2+7n ==> n(12n+7)==>12n^2 ==> n^2 ==> O(n^2)
```

=

```java
//n*n*n==>n^3==>O(n^3) --> cubic equation
for (int m = 0; m <= 5; m++) {
    for (int n = 0; n <= 5; n++) {

            for(int v=0; v<=5; v++) {
                System.out.print(m + "" + n + v+ " ");

            }


    }
    System.out.println();

}
```

(1+n+n+n)(1+n+n)(1+n+n+n)=(1+3n)(1+2n)(1+3n)
=

(1+2n+3n+6n^2)(1+3n)= (1+5n+6n^2)(1+3n) =

(1+3n+5n+15n^2+6n^2+18n^3) =

(18n^3 + 21n^2 +8n +1) =

N^3 + n^2 +n =

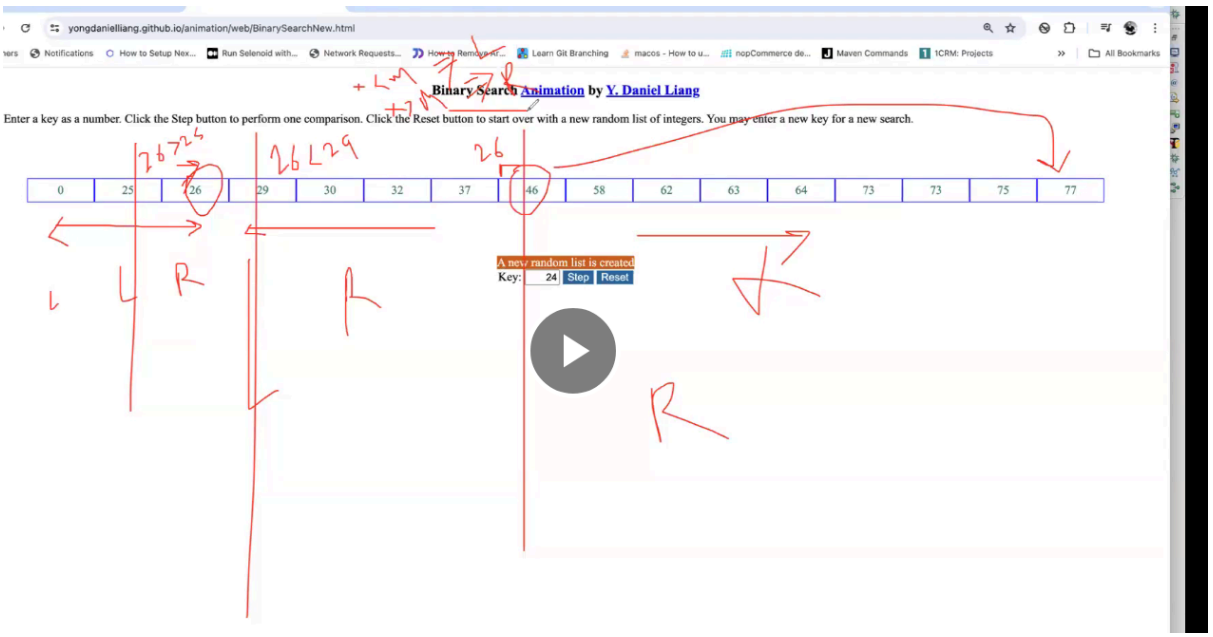
N(n^2 +1)=

N(n^2) =

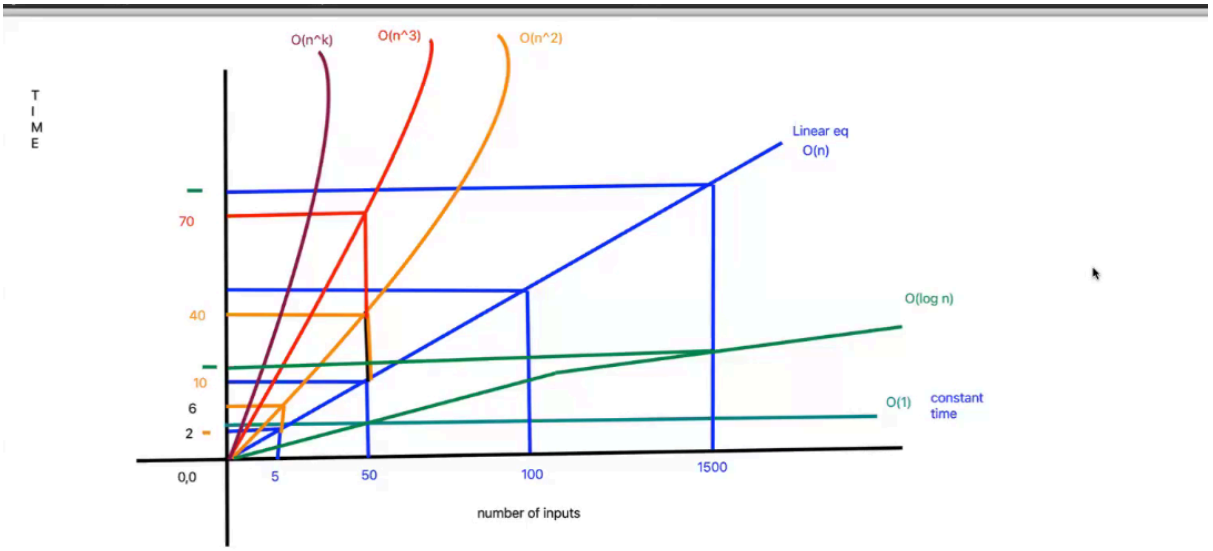
N^3

Divide and search for element- binary search.

To search for 26 we did three iterations.



But data has to be in sorted order.



```
61
62          //log n:
63          //n=32
64          //n/2=16
65          //n/4=4
66          //n/8==2
67
68          //tc = n/k
69          //T = n/k;
70          //log T = log(n/k);
71          //log T = log n - log k
72          //log T = 1-log k
73          //log T = log k;
74          //O(log n)
75
```

Binary search example of log n.

Log n to the base n is 1. So line 72, is 1-log k to the base n.

Note-

If we want to fetch any array element like –

Sysout(a[1]) – time complexity is O(1).

If we want to print all elements of array time complexity is O(n).

For fetching from excel we need rows and columns so two loops so time complexity is O(n^2).