

```

1 NestedLoops.java
2 public class NestedLoops {
3
4
5     public static void main(String[] args) {
6
7
8         //for -- outer
9         //for --inner
10
11
12         //00 01 02 03 04 05
13         //10 11 12 13 14 15
14         //20 21 22 23 24 25
15         //30 31 32 33 34 35
16         //40 41 42 43 44 45
17         //50 51 52 53 54 55
18
19
20         //i--> 0 to 5
21         //j--> 0 to 5
22         for(int i=0; i<=5; i++) {
23
24             for(int j=0; j<=5; j++) {
25                 System.out.println(i+" "+j);
26             }
27         }
28     }
29 }

```

00

01

02

And so on.

```

16          //40 41 42 43 44 45
17          //50 51 52 53 54 55
18
19
20          //i--> 0 to 5
21          //j--> 0 to 5
22          for(int i=0; i<=5; i++) {
23
24              for(int j=0; j<=5; j++) {
25                  System.out.print(i+""+j+" "); //10 01 02 03
26              }
27
28              System.out.println();
29
30          }
31
32
33
34

```

00 01 02 03

10 11 12 13

And so on.

```

17          // i--> 0 to 5
18          // j--> 0 to 5
19          // 6*6 --> m*n
20          for (int i = 0; i <= 5; i++) {
21
22              for (int j = 0; j <= 5; j++) {
23                  System.out.print(i + "" + j + " "); //00
24                  break;
25
26              }
27
28              System.out.println();
29
30          }
31

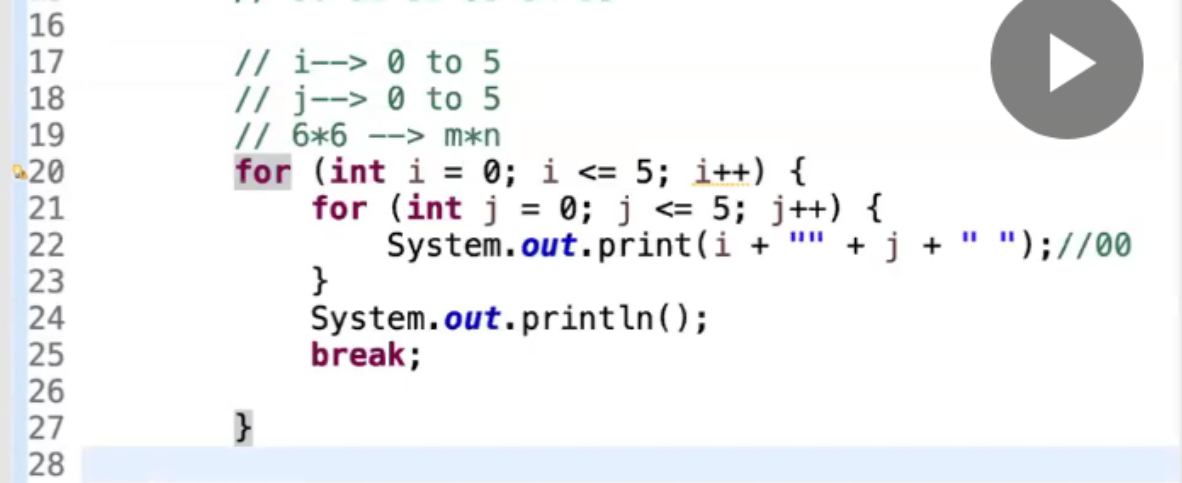
```

00

10

20

And so on.

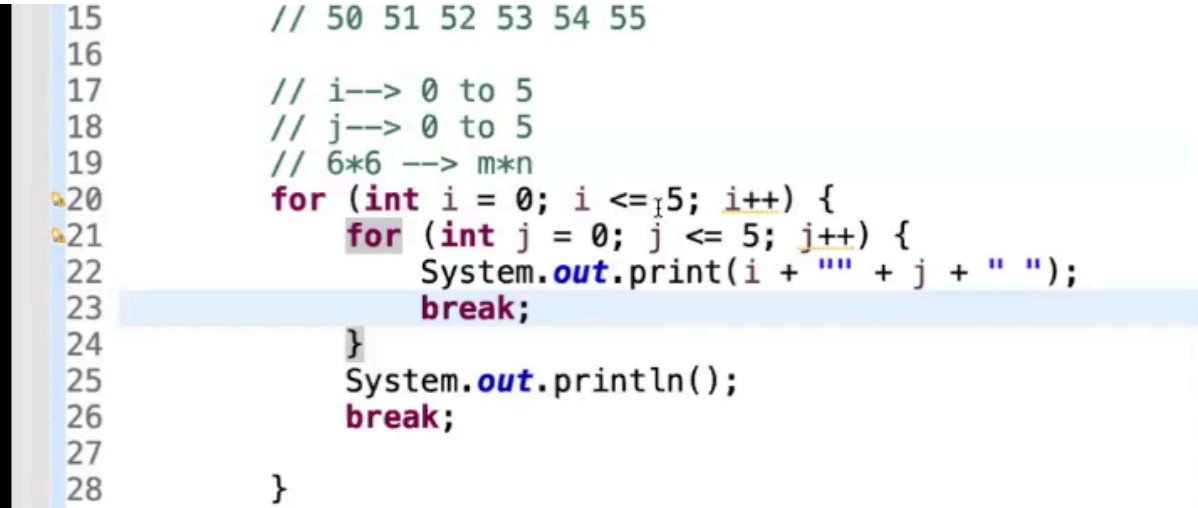


```

16
17 // i--> 0 to 5
18 // j--> 0 to 5
19 // 6*6 --> m*n
20 for (int i = 0; i <= 5; i++) {
21     for (int j = 0; j <= 5; j++) {
22         System.out.print(i + "" + j + " "); //00
23     }
24     System.out.println();
25     break;
26
27 }
28

```

00 01 02 03 04 05



```

15 // 50 51 52 53 54 55
16
17 // i--> 0 to 5
18 // j--> 0 to 5
19 // 6*6 --> m*n
20 for (int i = 0; i <= 5; i++) {
21     for (int j = 0; j <= 5; j++) {
22         System.out.print(i + "" + j + " ");
23         break;
24     }
25     System.out.println();
26     break;
27
28 }
29

```

00

Time complexity-

```

1 package javasessions;
2
3 public class TimeComplexity {
4
5     public static void main(String[] args) {
6
7         //Tc: Big Oh() $\rightarrow$ Big O(n)
8
9
10        //1.
11        int i = 1;
12        System.out.println(i);
13
14        //O(1)  $\rightarrow$  constant time
15

```

For loop time complexity-

```

15
16        //2. 100 secs  $\rightarrow$  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  $\Rightarrow$  3n+1  $\rightarrow$  Linear Equation
25        //3n+1  $\Rightarrow$  3n  $\rightarrow$  n  $\rightarrow$  O(n)
26

```

P runs only once so it is 1.

$P \leq 10 \rightarrow$ runs as per the number of inputs.

$P++ \rightarrow$ 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.

```

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

```



```

24      //
25      int k = 1;
26      while(k<=10) {
27          System.out.println(k);
28          k++;
29      }
30
31      //1+n+n+n==>3n+1==>3n==>n==>O(n)
32

```

```

33      //m*n
34      for (int m = 0; m <= 5; m++) {
35          for (int n = 0; n <= 5; n++) {
36              System.out.print(m + " " + n + " ");
37          }
38          System.out.println();
39          System.out.println("Hi");
40      }
41
42
43      //(1+n+n+n+n)(1+n+n+n+n)==>(1+4n)(1+3n)==>1+4n+3n+12n^2 ==> 12n^2+7n+1 ==> quadratic equation
44      //12n^2+7n+1 ==>12n^2+7n ==> n(12n+7)==>12n^2 ==> n^2 ==> O(n^2)


```

=

```

45
46 //n*n*n==>n^3==>O(n^3) --> cubic equation
47 for (int m = 0; m <= 5; m++) {
48     for (int n = 0; n <= 5; n++) {
49
50         for(int v=0; v<=5; v++) {
51             System.out.print(m + " " + n + v + " ");
52
53         }
54
55     }
56     System.out.println();
57
58 }
59

```



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

=

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

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

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

$$N^3 + n^2 + n =$$

$$N(n^2 + 1)=$$

$$N(n^2) =$$

$$N^3$$

Divide and search for element- binary search.

To search for 26 we did three iterations.

Binary Search Animation by Y. Daniel Liang

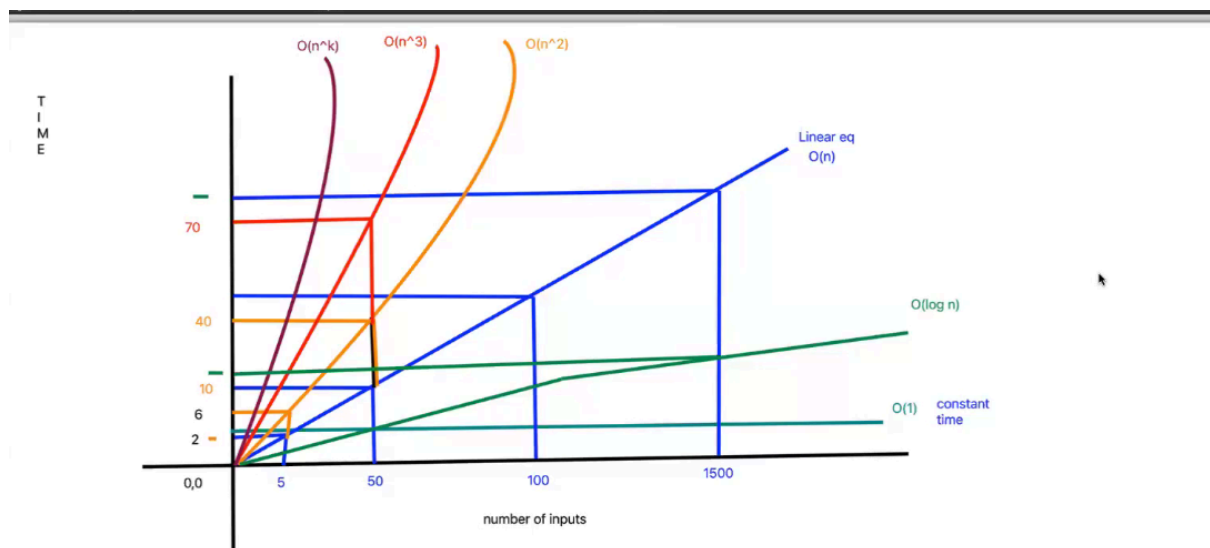
Enter a key as a number. Click the Step button to perform one comparison. Click the Reset button to start over with a new random list of integers. You may enter a new key for a new search.

Key: 24 Step Reset

A new random list is created

Handwritten annotations: Red arrows and letters (L, R) indicating the binary search process on the array. The array is: 0, 25, 26, 29, 30, 32, 37, 46, 58, 62, 63, 64, 73, 73, 75, 77. The value 26 is circled, and 46 is also circled. A red arrow points from the key input field to the array.

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 $O(n^2)$.