do - while loop

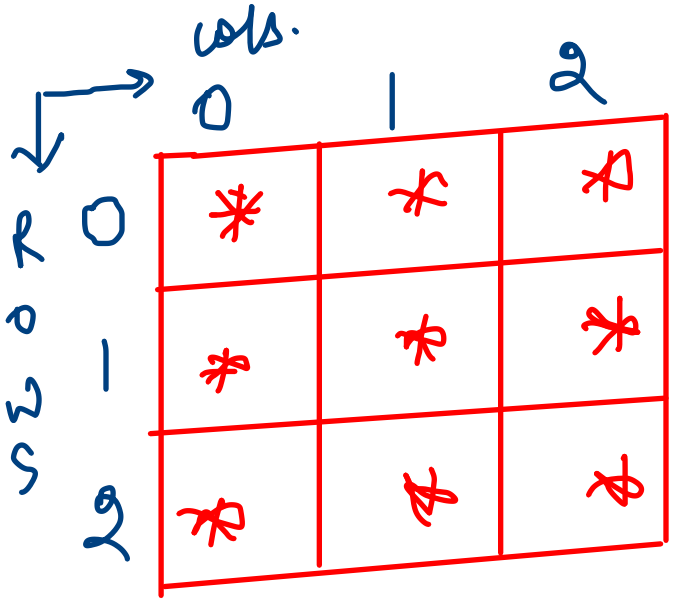
Syntax

initialize
do {
// statement
upgrade
} while (condition);

```
int i = 0;
do {
System.out.println(i);
  i++;
}
while (i < 5);
```

loop will execute once before checking the condition, whether condition is true or not.

Patterns → Sequence of objects.



Rows → (i)

Columns → (j)

n = 3

```
    0   1   2
0   *   *   *
1   *   *   *
2   *   *   *
```

```
for(int i=0; i<n; i++){
    for(int j=0; j<n; j++){
        sysout("*");
    }
    sysout();
```

rows (outer loop)

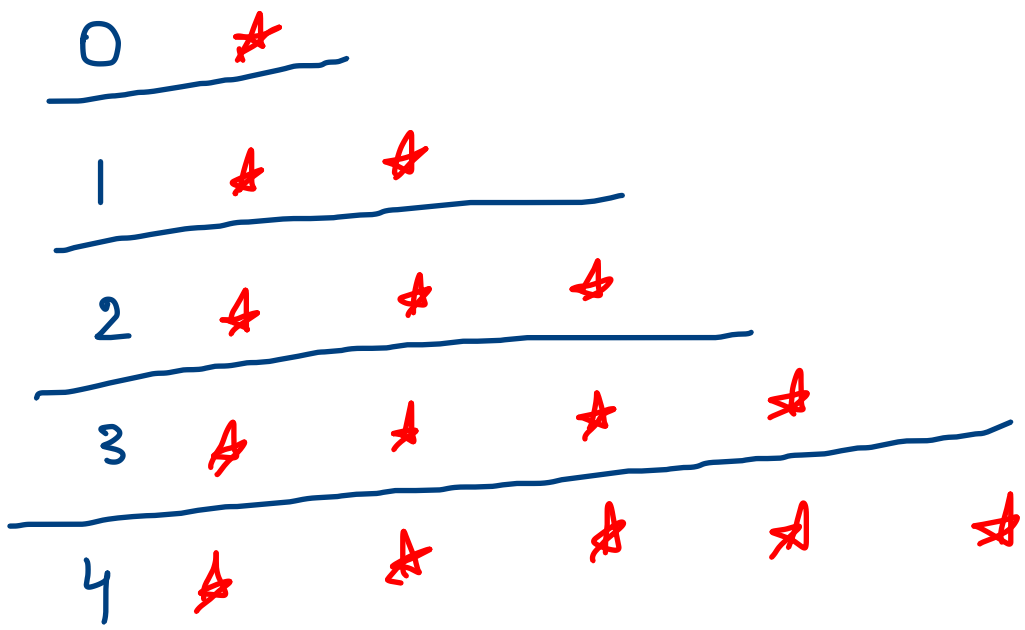
columns (inner loop)

'i' will represent current row

'j' will represent current col.

n=5

0 1 2 3 4




```
int i=0; i<n; i++
int j=0; j<i; j++
    cout("*");
}
cout<<endl;
```

Pattern 1 - Print Stars in same line

```java
Scanner s = new Scanner(System.in);
int n = s.nextInt();

for(int i = 0; i < n; i++){
    System.out.print("*");
}
```

i = 0 < n [0 – n]

n work → n

2 3 5 10 ''

O(n)

Pattern 2 - Print n x 12 star rectangle

```java
Scanner s = new Scanner(System.in);
int n = s.nextInt();

for(int i = 0; i < 12; i++){
    for(int j = 0; j < n; j++){
        System.out.print("*");
    }
    System.out.println();
}
```

0 – 12 constant – neglect

→ 12

→ 0 – n – n times

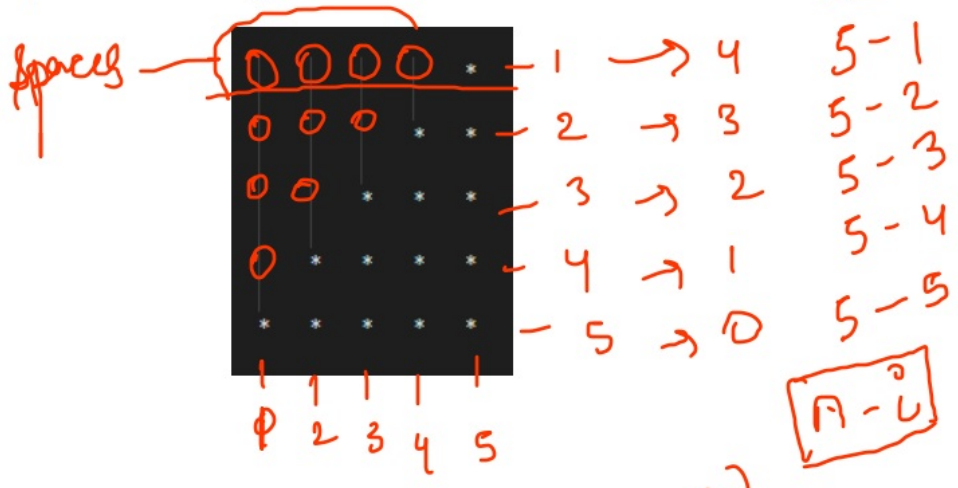
loop within loop → 12 × n = 12n

O(n)

n × n = n^2

$O(n^2)$

and print the following pattern.



n = 5

5-1
5-2
5-3
5-4
5-5

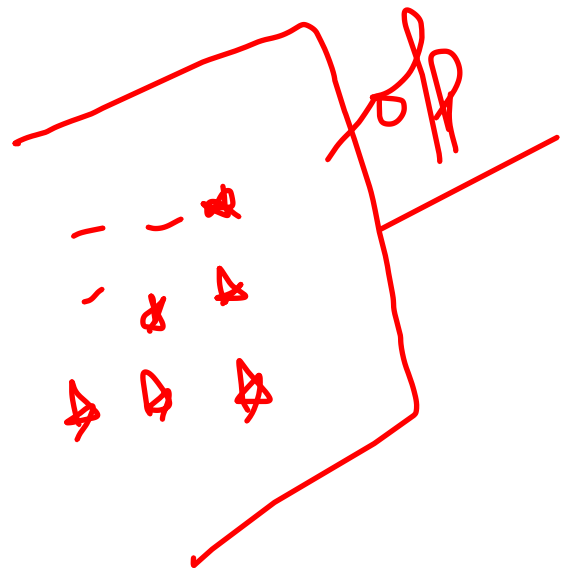
n - i

n × n = n²

(spaces, *)

```java
Scanner s = new Scanner(System.in);
int n = s.nextInt();

for(int i = 1; i <= n; i++){
    // spaces
    for(int j = 1; j <= n - i; j++){
        System.out.print(" ");
    }
    // stars
    for(int k = 1; k <= i; k++){
        System.out.print("*");
    }
    // move to next line
    System.out.println();
}
```



```
for(int i = 1; i <= n; i++){
for(int j = 1; j <= n - i; j++){
```

n * n/2

n = 3

i = 1 ≤ 3 T

j = 1 ≤ 3 - 1 = 2 T
2 ≤ 2 T
3 ≤ 2 x

k = 1 ≤ 1 T

i = 2 ≤ 3 T

j = 1 ≤ 3 - 2 = 1 T
2 ≤ 3 F

k = 1 ≤ 2 T → 1
2 ≤ 2 T → 2
3 ≤ 2 x

i = 3 ≤ 3 T

j = 1 ≤ 3 - 3 = 0 F

k = 1 ≤ 3 T → 1
2 ≤ 3 T → 2
3 ≤ 3 T → 3
4 ≤ 3 x

Hw_Print Spaced Right-angled whole numbers

```java
Scanner s = new Scanner(System.in);
int n = s.nextInt();

for(int i = 1; i <= n; i++){
    // spaces
    for(int j = 1; j <= n - i; j++){
        System.out.print("  ");
        // two spaces for alignment
    }
    // stars
    for(int k = 1; k <= i; k++){
        System.out.print(k + " ");
    }
    // move to next line
    System.out.println();
}
```

tab spaces → 4 spaces

("\t") j

TC → n^2

rows = n

n × n = n^2

SC = n - 6 + 6 [n]

m = 7 | 4 → |L-2| 2 0 1 2

rint the final output.

lines = (m/2 + 1) → 2 parts divide

	0	1	2	3	4	5	6
0	★	★	★	★	★	★	★
1	–	★	★	★	★	★	–
2	–	–	★	★	★	–	–
3	–	–	–	★	–	–	–

7 - 2×0 → 0 → 7

7 - 2×1 → 1 → 5

7 - 2×2 → 2 → 3

7 - 2×3 → 3 → 1

```
for(int i=0; < lines ; i++){
    for(int j=0; < i ; j++){
        sysо("(t");
    }
    for(int j=0; < m -2xi ; j++){
        sysо("*");
    }
```

if(j < [m - 2×i] - 1)

0 - 6

j < 4 ?

1)

```java
Scanner s = new Scanner(System.in);
int m = s.nextInt();

// total no pf lines in first half include middle one
int lines = m / 2 + 1;

// top half
for(int i = 0; i < lines; i++){
    // initial tab spaces
    for(int j = 0; j < i; j++){
        System.out.print("\t");
    }
    // stars
    for(int j = 0; j < m - 2 * i; j++){
        System.out.print("*" + "\t");
        // if(j < m - 2 * i - 1){
        //     System.out.print("\t");
        // }
    }
    System.out.println();
}
```

rows. lines + lines $\neq$ n

col. → ~~n~~ n

n x n

$\Rightarrow TC = O(n^2)$

2^n

2)

Delhi-Mumbai → train → 1-2 days → n^2

Delhi-Mumbai → flight → 1-2 hours → n

```java
// bottom half

for(int i = lines - 2; i >= 0; i--){
    // initial tab spaces
    for(int j = 0; j < i; j++){
        System.out.print("\t");
    }
    // stars
    for(int j = 0; j < m - 2 * i; j++){
        System.out.print("*" + "\t");
        // if(j < m - 2 * i - 1){
        //     System.out.print("\t");
        // }
    }
    System.out.println();
}
```

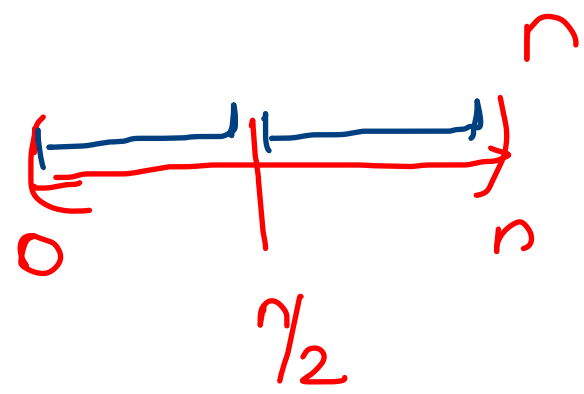
Time complexity

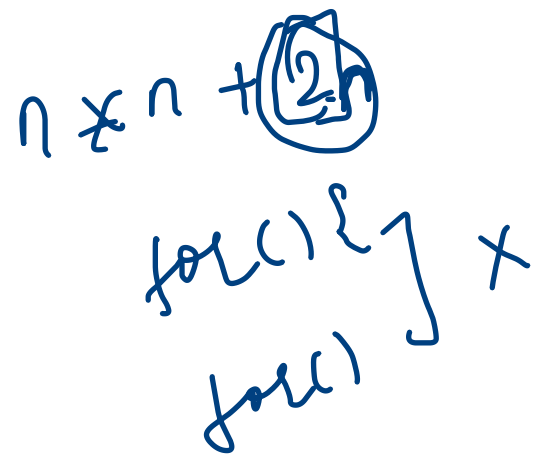
time taken to execute a program.

n rows m cols. $O(n \times m)$

```
for (i=0 ; i<n/2 ; i++)
    for (j=n/2 ; j<n ; j++)
```

$\rightarrow O(n)$





3 time complexity

$O(n)$ → linear

$O(n^2)$ → square

$O(n^3)$ → cube

$O(n^2 + n)$ ≈ $O(n^2)$

https://harish303.medium.com/va-10-star-patterns-in-java-a-must-know-interview-question-4d97731c8d35

Patterns Questions