Pattern Problems: Steps to Solve

1. **Find how many times the outer loop will run.**

Number of Lines = Number of Rows

**=>** Number of Times Outer Loop will Run

1. **Identify For Every Row number,**

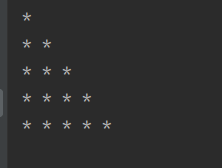
* How many Columns are there?
* Types of Elements in Column

1. **What Do you Need to print?**

* if number needs to be printed, start loop with row = 1

in general: always try to find relationship between no of rows, no of columns and no spaces etc.

**Problem 1: Print Below Pattern**



Solution :

import java.util.Scanner;

public class Pattern1 {

public static void main (String [] args)

{

Scanner sc = new Scanner (System.in);

int n1 = sc.nextInt();

pattern1(n1);

}

static void pattern1(int n)

{

for (int row = 1; row <= n; row++)

{

//for every row, find no of column

for (int col = 1; col <=row; col++)

{

// for every column, what need to be printed

System.out.print("\* ").

}

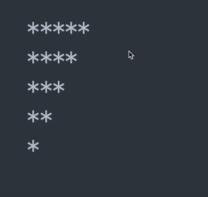
System.out.println();

}

}

}

Problem : Print Below Pattern



Solution:

static void pattern3(int n)

{

//n=5

for ( int row = 1; row <= n; row++ )

{

//for every row ,find no of column

for (int col = 1; col <= n-row+1 ; col++)

{

// for every column, what need to be printed

System.out.print("\* ");

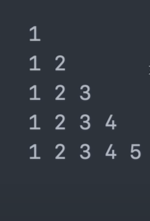
}

System.out.println();

}

}

**Poblem : Print Below Pattern**



Solution:

static void pattern4(int n)

{

//n=5

for ( int row = 1; row <= n; row++ )

{

//for every row ,find no of column

for (int col = 1; col <= row ; col++)

{

// for every column, what need to be printed

System.out.print(col +" ");

}

System.out.println();

}

}

**Problem : Print Below Pattern**



Hint : Find relation between row, n and column ,when row < =n and row >n

Solution :

total no of lines = 2n-1

static void pattern5(int n)

{

//n=5

int noOfColumn;

for ( int row\_no = 1; row\_no < 2\*n; row\_no++ )

{

//for every row\_no ,find no of column

noOfColumn = (row\_no <=n) ? row\_no : (2\*n - row\_no);

for (int col = 1; col <= noOfColumn ; col++)

{

// for every column, what need to be printed

System.out.print("\* ");

}

System.out.println();

}

}

Problem : Print Below Pattern



Solution:

Hint : Find Relation among n, row, no of spaces , no of column.

static void pattern28(int n)

{

//n=5

int noOfColumn;

for ( int row\_no = 1; row\_no < 2\*n; row\_no++ )

{

//for every row\_no ,find no of column

noOfColumn = (row\_no <=n) ? row\_no : (2\*n - row\_no);

// For every row, print no of spaces

for (int space = 1; space <= n-noOfColumn ; space++) {

System.out.print(" ");

}

for (int col = 1; col <= noOfColumn ; col++)

{

// for every column, what need to be printed

System.out.print("\* ");

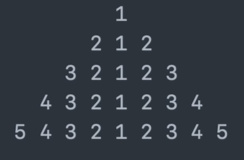
}

System.out.println();

}

}

**Problem: Print Below Pattern**



Solution:

static void pattern29(int n)

{

//n=5

for ( int row\_no = 1; row\_no <=n; row\_no++ )

**{**

//for every row\_no

// For every row, print no of spaces

for (int space = 1; space <= n-row\_no ; space++)

{

//two spaces will be printed for each

//since at each step ,by 1 extra space, first print is being shifted to right

System.out.print(" ");

}

//print row\_no to 1

for (int r = row\_no; r >=1; r--)

{

// for every column, what need to be printed

System.out.print(r+" ");

}

//print 2 to row\_no

for (int tr = 2; tr <=row\_no; tr++)

{

System.out.print(tr+" ");

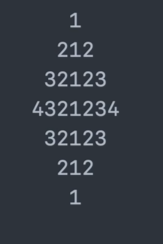
}

System.out.println();

**}** // End Outer loop

}

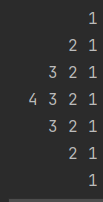
**Problem: Print Below Pattern**



Solution:

Total n = 2n-1;

Hint : Focus on below part first



try to find relation between row\_no and n ,derive no of columns for each half.

Code:

static void pattern17(int n)

{

//n=4

// Outer Loop will run 2\*n-1 times

for ( int row\_no = 1; row\_no <=2\*n-1; row\_no++ )

{

//for every row\_no, find no of columns

int c = row\_no > n ? 2\*n - row\_no : row\_no;

// For every row, print no of spaces

for (int space = 0; space <= n-c ; space++) {

//two spaces will be printed for each

//since at each step ,by 1 extra space, first print is being shifted to right

System.out.print(" ");

}

//print col to 1

for (int col = c; col >=1; col--)

{

// for every column, what need to be printed

System.out.print(col+" ");

}

//print 2 to col

for (int tc = 2; tc <=c; tc++)

{

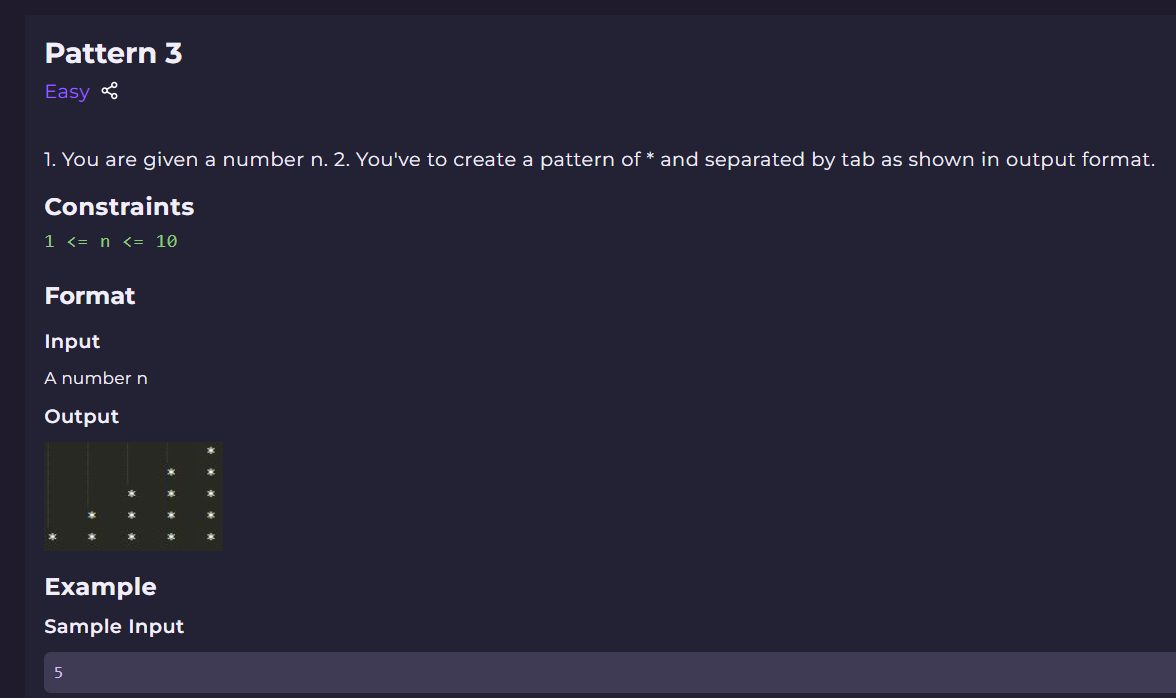
System.out.print(tc+" ");

}

System.out.println();

}

}



import java.util.\*;

public class Main {

public static void main(String[] args)

{

Scanner scn = new Scanner(System.in);

int n = scn.nextInt();

for(int row =1; row <=n ; row++)

{

// for each row ,get the spaces, print it

int totalSpaces = n-row;

for(int space =1; space <= totalSpaces ; space++)

{

System.out.print("\t");

}

// for each row ,print the column

for(int col =1; col <= row ; col++)

{

System.out.print("\*\t");

}

System.out.println();

}// end of loop

}

}



import java.util.\*;

public class Main

{

    public static void main(String[] args)

{

        Scanner scn = new Scanner(System.in);

        int n = scn.nextInt();

        for(int row =1; row <=n ; row++)

        {

            // for each row ,get the spaces

            int totalSpaces = row-1;

            for(int space =1; space <= totalSpaces ; space++)

            {

                System.out.print("\t");

            }

            // for each row ,get the no of column

            for(int col =1; col <= n-row+1 ; col++)

            {

                System.out.print("\*\t");

            }

            System.out.println();

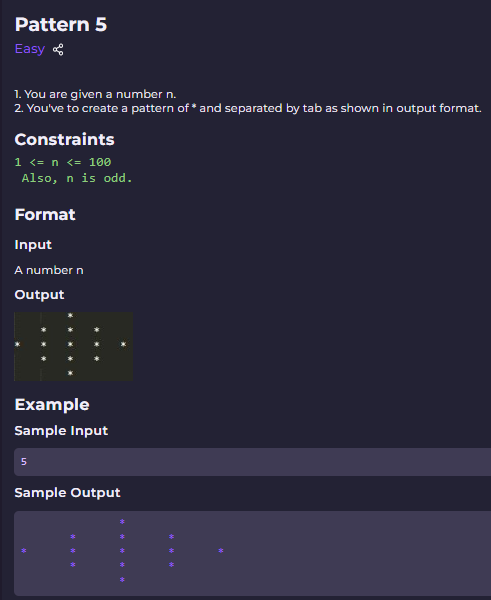
        }

    }

}

Time Complexitiy :

The time complexity of this solution is O(n^2), where n is the given input number since the outer for loop implements for n times and the inner loop for stars also implements for n times and when this loop has to execute only once then the loop for spaces has to implement for n - 1 times. So in terms of big ‘O’ notations, the complexity is O(n^2).



*Solution:*

1.try to find realtion between n ,spaces and no of columns

row \_no spaces no\_of\_col(stars)

|  |  |  |
| --- | --- | --- |
| 1 | 2 | 1 |
| 2 | 1 | 3 |
| 3 | 0 | 5 |
| 4 | 1 | 3 |
| 5 | 2 | 1 |

Code 1:

public static void pattern5(int n)

{

int col =1;

int sp = n/2;

for (int row\_no = 1; row\_no <=n ; row\_no++)

{

// For each row

// First, print spaces

for (int spaces = 1; spaces <=sp ; spaces++) {

System.out.print("\t");

}

// print the column

for (int col\_no = 1; col\_no <=col ; col\_no++) {

// print the content

System.out.print("\*\t");

}

//find No of spaces and columns

if(row\_no <=n/2)

{

//spaces are decreasing by 1

sp--;

// columns are increasing by 2

col = col+2;

}

else

{

//spaces are increasing by 1

sp++;

//col are decreasing by 2

col =col -2;

}

// change the line

System.out.println();

}

}

Code 2:

import java.util.\*;

public class Main {

    public static void main(String[] args) {

        Scanner scn = new Scanner(System.in);

        // write ur code here

         int n = scn.nextInt();

         int n1=2;

        int qutiont = n/2; e.g 5/2 =2 .

        int constant\_one =1;

        int totalTabs =constant\_one;

        int total\_col = constant\_one;

        int partDecider= qutiont+constant\_one;

        // 1. Total no of rows is n

        for(int row =1; row <=n ; row++)

        {

            if( row <= partDecider)

            {

               // 2. total no of tabs

                totalTabs = qutiont -row+1;

                // 3. print the tabs

               for(int tab =1; tab <= totalTabs ; tab++)

                {

                System.out.print("\t");

                }

                // 4. for each row ,get the no of column

                for(int col =1; col <= total\_col; col++)

               {

                System.out.print("\*\t");

               }

                 total\_col = total\_col+2;

               System.out.println();

            }

            // when row > partDecider

            else

            {

                     // total no of tabs

             totalTabs = constant\_one;

              // 3. print the tabs

            for(int tab =1; tab <= totalTabs ; tab++)

            {

                System.out.print("\t");

            }

            totalTabs = constant\_one++;

              // for each row ,get the no of column

                  total\_col = n-n1;

                 for(int col =1; col <= total\_col; col++)

                 {

                System.out.print("\*\t");

                 }

            n1=n1+2;

            System.out.println();

            }

        }

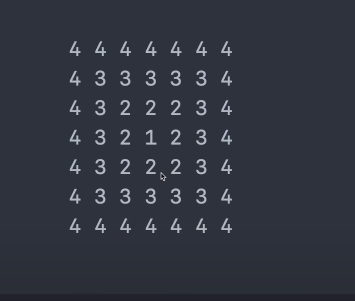
    }

}

Time Complexity: O(n)

We are running a loop from 1 to n, so the time complexity becomes O(n).

***Problem: Print Below Pattern***



Solution:

First try to print this:

A screenshot of a computer screen

Description automatically generated

Every Cell Value at any index = min Distance from {left wall, bottom wall,right wall,upper wall}

=min {col , n-row, n-col,row}

static void pattern31(int n)

{

int originalN =n;

n = 2\*n-2;

for ( int row = 0; row <= n; row++ )

{

//for every row ,find no of column

for (int col = 0; col <=n ; col++)

{

// for every column, what need to be printed

int atEveryIndex = originalN-Math.min(Math.min(row,col),Math.min(n-row,n-col));

//int atEveryIndex = Math.min(Math.min(row,col),Math.min(n-row,n-col));

System.out.print(atEveryIndex +" ");

}

System.out.println();

}

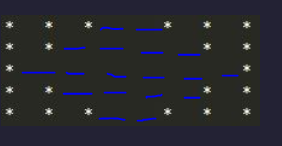
}

Problem: Print Below pattern



Solution:

Due to tab/space being printed at the last of star, one spaces would be less always.



public static void pattern6(int n)

{

int col =n/2+1;

int sp = 1;

for (int row\_no = 1; row\_no <=n ; row\_no++)

{

// For each row

// print the column

for (int col\_no = 1; col\_no <=col ; col\_no++) {

// print the content

System.out.print("\*\t");

}

// print spaces

for (int spaces = 1; spaces <=sp ; spaces++) {

System.out.print("\t");

}

// print the column

for (int col\_no = 1; col\_no <=col ; col\_no++) {

// print the content

System.out.print("\*\t");

}

//find spaces and columns

if(row\_no <=n/2)

{

//spaces are increasing by 2

sp += 2;

// columns are decreasing by 1

col--;

}

else

{

//spaces are decreasing 2

sp -= 2;

//col are increasing by 1

col++;

}

// change the line

System.out.println();

}

}

Time Complexity: O(n^2)