**Pascal`s Triangle**

Q: Write a program to print Pascal`s triangle using Iteration, Recursion, Memoization

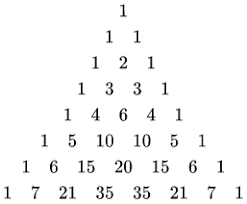
**Theory:**

Before going towards coding part lets understand what is Pascal`s triangle

Pascal`s triangle is basically a triangle of the binomial coefficient which is used in combinatorics (binomial theorem).

it was named on the mathematician 'Blaise Pascal', but first time it was discovered by the 'Umar bin Khiyam'

**Pascal`s Triangle:**



The idea start with 1 and also end at in every row, then in second after 1 write 2, then 1,

from third row, first position is fixed for 1, after 1 add first two numbers from preceding row

1

/ \

1 2 1

/ \/ \/ \

1 3 3 1

/ \/ \/ \/ \

1 4 6 4 1

**1- Pascal`s triangle using iteration**

**Tip:**

* Whenever we print any type of pattern in programming, in most of the time we print the number of rows equal to the number of times user want.
* As pattern is of 2-D, and for 2-D we use nested loops.

**Program Explanation:**

1. To print the pattern first print the blank spaces before and after the number in the triangle.
2. As we discussed the number are not more than just a binomial coefficient, therefore we calculate the binomial coefficient using the formula nCr = n! / (r! \* (n-r)! )
3. after printing the number we are changing the row to do so print a blank line
4. Now the remaining thing is to calculate the how many times the loop will execute, the first loop will execute equal to the number of rows as I mentioned earlier.
5. Secondly we are printing the blank space, but the question is how many times we print the blank space, see first row, we are printing the blank space three times, in second row it is about 2 times, in third row it is 1 => 3, 2, and 1. The from the diagram we can see that the blank space is equal to the (number of rows - number of iteration). In first row will print only 1 means all other will be blank space, for this number of iteration is 0 and number of row is 4 (in our case), therefore blank space = 4 - 0 = 3 i = 0, j = 0, row = 4. i is zero, then we iterate through the loop using j (nested loop), j will iterate till (noOfRows - i).
6. After printing the space we print the coefficient, to calculate the coefficient are using the formula of nCr, In our case the n=i, r=j. To calculate coefficients we are using formula, as formula is all about factorials: (number \* the number preceding to it but greater than 1). Here we are calculating it using recursion. Recursion: Calling itself, Recursive function: The function which calls itself.
7. We print the blank line, after every row. NOTE: While printing the number make sure you are not using println in Java, because it will print new line which will not print the pattern.

**2- Pascal`s triangle using Recursion**

**Theory:**

We have talked about recursion in previous section now let’s understand the program to print the pattern. In recursion we will just call the function over and over again.

The point to be noted here is that we have to take care of the infinite function calls which leads to stack overflow, we can restrict it using the check which we call it as base condition. In recursion the important part is to find the base condition.

**Program Explanation:**

1. In our program we declared a recursive function to calculate the binomial coefficient.
2. This function (prev = triangleUsingRecursion(i-1, j-1)) calculate the just above row and column is previous column value of the triangle independently and (next = triangleUsingRecursion(i-1, j)) calculate just above row and the column is also same The parameters are values of n and c in nCr.
3. **Note:** Recursion is costlier in terms of computation, for large values it will through an error overflow' we can enhance it using Memoization which we will cover it in next session of this post

**3- Pascal`s triangle using Memoization**

**Theory:**

Memoization is a way to reduce computational cost or it is technique to improve computation.

In recursion, if we carefully observe the computation of coefficients it calculate it even it is calculated before.

e.g. if we want to calculate the 8C5, what is the formula nCr => n! / (r! \* (n-r)!) In numerator it is n! => 5!, in denominator it r! and multiplied by (n-r)! r!, means if we apply for 5 take a look

8! / (5! \* (8-5)!) => in numerator we calculate 8! it will goes to

8!, 7!, 6!, 5!, 4!, 3!, 2!, 1. In denominator for 5! it will go through

5!, 4!, 3!, 2!, 1 and similarly for 8-5 = 3 it is

3!, 2!, 1

but as we calculated 3! in (8-5)! then why do we calculate it in 5! and 8!, similarly in 5! we will calculate it, but using recursion we calculate it in 8!, To overcome this repetition we use Memoization We can achieve Memoization using creating a cache memory or temporary storage, it stores the computed values, every time when compute nCr first will check nCr for the respective n and r, means is nCr is present in our memory, if it is present then we will not calculate it just return the value from memory. When we didn’t find the nCr in memory we computer it and store it in the memory for future use.

**Program Explanation:**

1. Program is similar to the previous program (Pascal Triangle using recursion), but the small difference is that we are storing the calculated values. Before calculating the coefficient first we check it in the memory, it is present return that value.
2. The values of n and r is rows and columns.
3. We are using HashMap to implement the cache memory ==> public static HashMap<Integer, HashMap<Integer, Integer>> cache = new HashMap<> (); key is of type Integer and values is of type HashMap which further have the key integer and value integer.
4. As we know HashMap provide many inbuilt function for operation like get the values, get the keys and put the keys and values, we are using them to operate on HashMap.
5. putIfAbsent function insert the provided value if the key is not present for the provided key.