Informe de 568B(div - 1)

Ana Karla Caballero González C-213

14 de julio de 2021

B. Simetric and Transitive

Dado un conjunto ρ de pares $< a, b >$ de elementos de un conjunto A se define una relación binaria en el set A: para dos elementos a y b del conjunto A se dice que están en relación ρ si el par $< a, b > \in \rho$. La relación binaria es una relación de equivalencia. Se quiere contar el número de relaciones binarias sobre un conjunto de tamaño n de manera que sean simétricas, transitivas pero no reflexivas(o sea, no llegar a ser una relación de equivalencia)

Entradas y salidas

* Como entrada se tiene un entero n tal que $1 \leq n \leq 4000$
* Como salida se imprime un entero n

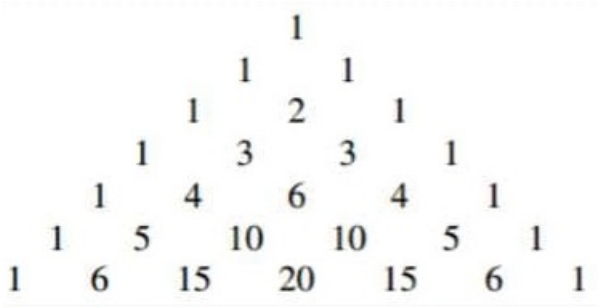
Solución al problema

Para poder ver la solución dada lo separaremos por casos:

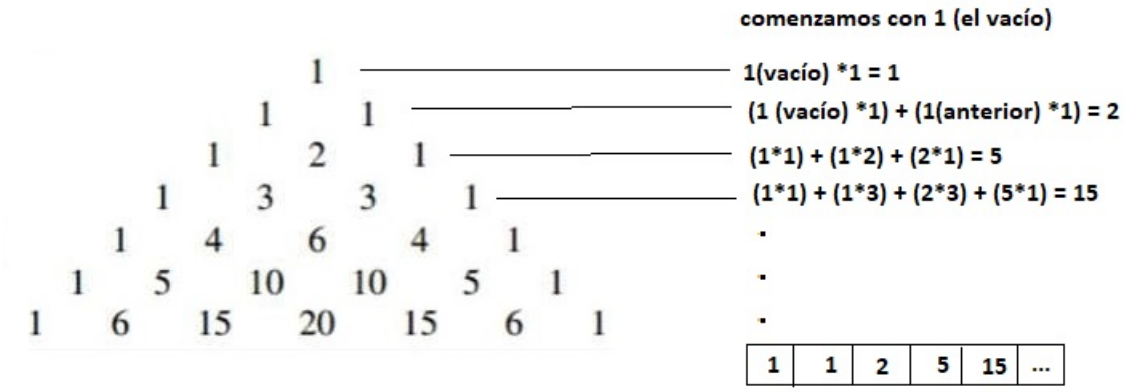
* Para n = 0, en este caso no tenemos ningún par perteneciente a el conjunto A, por lo que solamente tendríamos el vacío como relación, la cual no cumple con la relación binaria que se desea hallar.
* Para n = 1, significaría que tenemos un único par $\{< x, x >\}$ en el conjunto A, y como se quieren hallar la cantidad de relaciones binarias que existen que son simétricas, transitivas pero no reflexivas se obtiene como único par perteneciente a ρ el ϕ
* Para n = 2, significaría que tenemos $\{< x, x >\}$, $\{< y, y >\}$, $\{< x, x >, < x, y >, < y, x >, < y, y >\}$ en el conjunto A, de estas tres relaciones la tercera no cumple con las condiciones de relación definida, a lo que queda resultante $2 + 1$ (el vacío) $= 3$ relaciones binarias pertenecientes al conjunto ρ.
* Para n = 3, se tendría en el conjunto A $\{< x, x >\}$, $\{< y, y >\}$, $\{< z, z >\}$, $\{< x, x >, < x, y >, < y, x >, < y, y >\}$, $\{< x, x >, < x, z >, < z, x >, < z, z >\}$, $\{< y, y >, < y, z >, < z, y >, < z, z >\}$, $\{< x, x >, < x, y >, < y, x >, < y, y >, < z, z >\}$, $\{< x, x >, < x, z >, < z, x >, < z, z >, < y, y >\}$, $\{< y, y >, < y, z >, < z, y >, < z, z >, < x, x >\}$, $\{< x, x >, < y, y >, < z, z >, < x, y >, < y, z >, < x, z >, < z, x >, < y, x >\}$, estas últimas 4 relaciones, como contienen a todos los pares simétricos del conjunto A, se pueden decir que son relaciones de equivalencia, por lo tanto no cumple con las condiciones de la relación binaría dada. Entonces existen $3+3+3+1$ (el vacío) $= 10$ relaciones binarias pertenecientes al conjunto ρ.

En el caso 1 tenemos solamente el elemento vacío, el cual en el triángulo de Pascal es el primer elemento de arriba hacia abajo; en el caso 2 tenemos el vacío y el par $< x, x >$, en el cuál este primero es el primer elemento de la segunda fila del triángulo de Pascal y el segundo es el segundo elemento, en el tercer caso tenemos el vacío, los pares $\{< x, x >\}$, $\{< y, y >\}$ y $\{< x, x >, < x, y >, < y, x >, < y, y >\}$ (los elementos de la tercera fila $\{1, 2, 1\}$. En cuanto a los valores que se devuelven para el caso 1 es 0, caso 2 es 1 (combinación del caso 0 en 1), caso 3 es 3 (sumatoria de la combinación del caso 0 en 1 y el caso 1 en 2), para el caso 4 tendremos 10 (sumatoria del caso 0 en 1, caso 1 en 3 y caso 2 en 3)

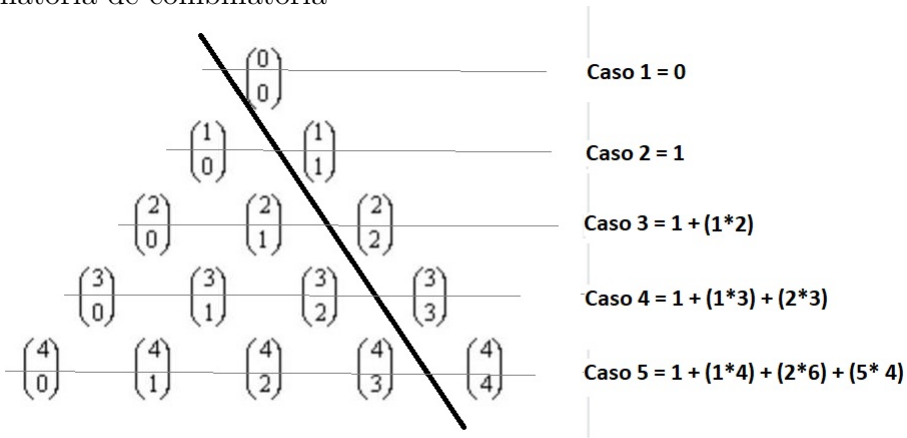
Triángulo de Pascal



Combinatoria



Sumatoria de combinatoria



Solución programada

```
Simetric_and_Transitive.py
1   mod = 10**9 + 7
2   n = int(input())
3   f = [0]*(n+1)
4   f[0] = 1
5   count = 1
6   for i in range(1, n+1):
7       for j in range (0, i):
8           f[(count + j + n + 1)% (n+1)]  = (f[(count - i + j + n + 1) % (n+1)] + f[(count + j +n) % (n+1)]+ mod)% mod
9       count = (count + i + n +1) %(n+1)
10  print(f[(count -1 +n)%(n+1)])
```

Se crea el número por el cual se dividirá en caso de que el resultado se muy grande como se indicó en la orientación del problema. Se reciben los parámetros de entradas. Para ahorrar espacio se crea un array donde se va a poner los valores de la sumatoria de la combinatoria de los factores del triángulo de Pascal. Se indexa

en la primera posición el 1 , ya que este es el vacío, como se muestra en la figura de combinatoria. Se crea un valor (count) que iniciará en 1. Luego se itera en el array por medio de un doble for, dado a que como se muestra en las fotos, el triángulo de Pascal se puede crear en una matriz en la cual se utilizará solamente el triangulo inferior, por eso los indices de los fors serán del 1 al n+1, y del 0 al index por el que se está iterando. Dado que se utiliza un array y no una matriz, en este se indexaran sumaran los valores hallados de manera circular,para no perder la posición que es corresponde. Finalmente se devuelve el valor que se quiere.

Complejidad

En este algoritmo se recorre el array de manera circular; se mantiene un i fijo que estará entre 1 y $n+1$ y se iterará desde 0 a i hasta que se llegué al final. Esto tiene una complejidad temporal máxima de $O(n^2)$

Generador

Generador.py

```python
from math import factorial
import random

input_number = random.randint(1, 4000)
mod = 10**9 + 7
list_fac = [0]*(input_number+1)
for i in range(0, input_number+1):
    list_fac[i] = factorial(i)

dp = [0] * (input_number + 1)
dp[0] = 1
res = 0

for i in range(1, input_number + 1):
    cur = 0
    for j in range(0,i):
        cur += (list_fac[i-1]//(list_fac[j]*list_fac[(i-1)-j])) * dp[j]
        cur %= mod
    dp[i] = cur
    res += (list_fac[input_number]//(list_fac[i-1]*list_fac[input_number+1-i])) * dp[i-1]
    res %= mod
print(str(input_number) + '\n' + str(res))
```

El generador de casos pruebas abarca el problema de una manera un poco diferente. El resultado de lo descrito en la solución del problema se puede escribir como $\Sigma_0^{i=n} \frac{n!}{i!*(n-i)!}$. Por lo que luego de escoger un número al azar entre los parámetros de entrada establecidos, y poner en una variable el número por el cuál se divide, se crea una lista en la cuál se crearán todos los factoriales de los números del 0 hasta el n. luego se procede igual que en la solución dada. Al hallar todos los factoriales del 0 al n, la complejidad del método aumenta por lo que se tiene $O()$