Combinatoria

October 24, 2019

1 Combinatoria

1.1 Números de Fibonacci

$$\begin{aligned} f(0) &= 0 \\ f(1) &= 1 \\ f(n) &= f(n-1) + f(n-2) \end{aligned}$$

Teorema de Zeckendorff: cualquier entero positivo puede ser escrito de **manera única** como una suma de uno o más términos de la secuencia de Fibonacci, tal que la suma no incluye términos consecutivos.

UVA 763

The standard interpretation of the binary number 1010 is $8 + 2 = 10$. An alternate way to view the sequence 1010 is to use Fibonacci numbers as bases instead of powers of two. For this problem, the terms of the Fibonacci sequence are:

$$1; 2; 3; 5; 8; 13; 21; ...$$

where each term is the sum of the two preceding terms (note that there is only one 1 in the sequence as defined here). Using this scheme, the sequence 1010 could be interpreted as $1x5 + 0x3 + 1x2 + 0x1 = 7$. This representation is called a Fibinary number. Note that there is not always a unique Fibinary representation of every number. For example the number 10 could be represented as either $8 + 2$ (10010) or as $5 + 3 + 2$ (1110). To make the Fibinary representations unique, larger Fibonacci terms must always be used whenever possible (i.e. disallow 2 adjacent 1's). Applying this rule to the number 10, means that 10 would be represented as $8 + 2$ (10010).

Input and Output: Write a program that takes two valid Fibinary numbers and prints the sum in Fibinary form. These numbers will have at most 100 digits. In case that two or more test cases had to be solved, it must be a blank line between two consecutive, both in input and output files.

Adaptar la regla de adición al caso:

$$f(n) + f(n) = f(n-1) + f(n-2) + f(n) = f(n+1) + f(n-2)$$

- Guardar los términos resultando de los $f(n-2)$ en un nuevo número.

- Iterar las sumas con el nuevo número, mientras está no nulo. Por construcción, este número está inferior de al menos un dígito cada vez.
- En cada paso, aplicar corrección a los números para que cada par de 1 consecutivos se cambie a un 1 posterior.

Fibonacci: para calcular el término n

- Método trivial por **DP**, complejidad?
- f(93) es el último que se puede representar en uint64 (usar **Biginteger** o equivalente si necesario).
- Para el cálculo de un sólo término, ¿algo mejor?

Por **exponenciación de matriz**, se puede obtener el resultado de $f(n)$ en $(O(\log(n)))$!
Da una forma **cerrada**:

$$f(n) = \frac{1}{\sqrt{5}}\phi^n - \frac{1}{\sqrt{5}}\varphi^n$$

donde $\phi = \frac{1+\sqrt{5}}{2}$ y $\varphi = \frac{1-\sqrt{5}}{2}$.

- Aproximación útil: para n grande, $f(n) \approx \frac{1}{\sqrt{5}}\phi^n$.
- ¿Cómo llegar a esa forma cerrada?

Usar la versión matricial de la recursión:

$$\begin{pmatrix} f(n) \\ f(n-1) \end{pmatrix} = \begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} f(n-1) \\ f(n-2) \end{pmatrix}$$

y

$$\begin{pmatrix} f(n) \\ f(n-1) \end{pmatrix} = \begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix}^{n-1} \begin{pmatrix} 1 \\ 0 \end{pmatrix}$$

Usar luego la eigen-decomposición de $\begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix}$:

$$\begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix} = \frac{1}{\sqrt{5}} \begin{pmatrix} \phi & \varphi \\ 1 & 1 \end{pmatrix} \begin{pmatrix} \phi & 0 \\ 0 & \varphi \end{pmatrix} \begin{pmatrix} 1 & -\varphi \\ -1 & \phi \end{pmatrix}$$

$$\begin{pmatrix} f(n) \\ f(n-1) \end{pmatrix} = \frac{1}{\sqrt{5}} \begin{pmatrix} \phi & \varphi \\ 1 & 1 \end{pmatrix} \begin{pmatrix} \phi^{n-1} & 0 \\ 0 & \varphi^{n-1} \end{pmatrix} \begin{pmatrix} 1 & -\varphi \\ -1 & \phi \end{pmatrix} \begin{pmatrix} 1 \\ 0 \end{pmatrix}$$

... y la forma cerrada sale inmediatamente:

$$f(n) = \frac{1}{\sqrt{5}}\phi^n - \frac{1}{\sqrt{5}}\varphi^n.$$

Observación: Este proceso se puede aplicar a un gran rango de secuencias.

UVA 948

The well known Fibonacci sequence is obtained by starting with 0 and 1 and then adding the two last numbers to get the next one. For example the third number in the sequence is 1 ($1 = 1 + 0$), the fourth is 2 ($2 = 1 + 1$), the fifth is 3 ($3 = 2 + 1$) and so on.

The sequence appears on many things in our life, in nature, and has a great significance. Among other things, do you know that all positive integer numbers can be represented as a sum of numbers in the Fibonacci sequence? More than that, all positive integers can be represented as a sum of a set of Fibonacci numbers, that is, numbers from the sequence, without repetition.

For example: 13 can be the sum of the sets $\{13\}$, $\{5, 8\}$ or $\{2, 3, 8\}$ and 17 is represented by $\{1, 3, 13\}$ or $\{1, 3, 5, 8\}$. Since all numbers have this property (do you want to try to prove this for yourself?) this set could be a nice way to use as a "base" to represent the number. But, as we have seen, some numbers have more than one set whose sum is the number. How can we solve that? Simple! If we add the constraint that the sets cannot have two consecutive Fibonacci numbers, than we have a unique representation for each number! This restriction is because the sum of any two consecutive Fibonacci numbers is just the following Fibonacci number. Now that we know all this we can prepare a nice way to represent any positive integer. We will use a binary sequence (just zeros and ones) to do that. For example, $17 = 1 + 3 + 13$ (remember that no two consecutive Fibonacci numbers can be used). Let's write a zero for each Fibonacci number that is not used and one for each one that is used, starting at the right. Then, $17 = 100101$. See figure 2 for a detailed explanation. In this representation we should not have zeros at the left, this is, we should only write starting with the first one. In order for you to understand better, note that in this scheme, not using two consecutive Fibonacci numbers means that the binary sequence will not have two consecutive ones. When we use this representation for a number we say that we are using the Fibonaccimal base, and we write it like $17 = 100101$ (fib). Given a set of numbers in decimal base, your task is to write them in the Fibonaccimal base.

Input: The first line of input contains a single number N, representing the quantity of numbers that follow ($1 \leq N \leq 500$). Than follow exactly N lines, each one containing a single positive integer smaller than 100000000. These numbers can come in any order.

Output: You should output a single line for each of the N integers in the input, with the format 'DEC BASE = FIB BASE (fib)'. DEC BASE is the original number in decimal base and FIB BASE is its representation in Fibonaccimal base. See the sample output for an example.

Si $f(k)$ es el **mayor término de Fibonacci** tal que $f(k) \leq n$: * $n - f(k)$ es positivo : * $n - f(k) = f(k - p) + ...$

entonces:

$$n = f(k) + f(k - p) + ...$$

con los términos siguientes no consecutivos.

Observar que necesariamente $p > 1$; sino, tendríamos un término más en la suma, es decir $f(k + 1)$, lo que es imposible por la hipótesis de arriba.

Entonces, tenemos la **representación (única) de Zeckendorff**.

Se deduce una estrategia **glotona**: para n, buscar en cada iteración el mayor $f(k)$ inferior a n y seguir de la misma manera con $n - f(k)$.

UVA 10689

Let's define another number sequence, given by the following function:

$$\begin{array}{rcl} f(0) & = & a \\ f(1) & = & b \\ f(n) & = & f(n-1)+f(n-2) \end{array}$$

When $a = 0$ and $b = 1$, this sequence gives the Fibonacci Sequence. Changing the values of a and b, you can get many different sequences. Given the values of a, b, you have to find the last m digits of $f(n)$.

Input: The first line gives the number of test cases, which is less than 10001. Each test case consists of a single line containing the integers a, b, n, m. The values of a and b range in $[0, 100]$, value of n ranges in $[0, 1000000000]$ and value of m ranges in $[1, 4]$.

Output: For each test case, print the last m digits of $f(n)$. However, you should NOT print any leading zero.

La técnica DP directa genera TLE para este problema (n es demasiado grande).

Usar la exponenciación:

$$\begin{pmatrix} f(n) \\ f(n-1) \end{pmatrix} = \begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix}^{n-1} \begin{pmatrix} b \\ a \end{pmatrix}$$

Complejidad?

Puede ser mejorado, siguiendo la estrategia:

$$\begin{aligned} A^{2p} &= (A^p)^2 \\ A^{2p+1} &= (A^p)^2.A \end{aligned}$$

permite realizar la exponenciación en $\log_2(p)$.

Para fines de este problema en particular: Usar todas las operaciones modulo m.

```
In [ ]: #include <iostream>
        using namespace std;
        Mat pow(Mat b,int p,int mod) {
            Mat ans;
            for (int i=0;i<2;i++)
                for (int j=0;j<2;j++) {
                    ans.data[i][j] = (i==j);
            }
            while (p) {
                if (p&1) ans = mult(ans,b,mod);
                b = mult(b,b,mod);
                p >>=1;
            }
            return ans;
        }

        int main() {
            ios::sync_with_stdio(false);
            Mat fib;
```

```
    fib.data[0][0] = 1;
    fib.data[1][1] = 0;
    fib.data[1][0] = 1;
    fib.data[0][1] = 1;
    int nCases,n,m,a,b;
    cin >> nCases;
    for (int l=0;l<nCases;l++) {
        cin >> a >> b >> n >> m;
        int p = 1; for (int k=0;k<m;k++) p *= 10;
        Mat f = pow(fib,n,p);
        int h = f.data[1][0]*b+f.data[1][1]*a;
        cout << h%p << endl;
    }
}
```

Periodo de Pisano: los últimos $1, 2, 3, 4$ dígitos de los números de Fibonacci se repiten con periodos respectivos de $60, 300, 1500, 15000$. Se podría simplificar el problema calculando los últimos dígitos de $f(n\%k_m)$, donde k_m es el periodo correspondiendo a m.

1.2 Coeficientes binomiales

Número de maneras en que se puede tomar k elementos (distintos) entre n.

$$C(n,k) = \frac{n!}{k!(n-k)!}$$

Problemas de **overflow** posibles con grandes n o k.

Para evitar overflow por uno de los factoriales en un cálculo individual $C(n,k)$: * Comparar k y n − k para elegir de calcular la razón $\frac{n!}{k!}$ o $\frac{n!}{(n-k)!}$ por productos. * Utilizar BigInteger si necesario.

Para muchos valores C(n, k):

- Si no son todas las C(n, k), usar DP top-down (recursión con memoization)

$$\begin{array}{rcl} C(n,0) & = & C(n,n) = 1 \\ C(n,k) & = & C(n-1,k-1) + C(n-1,k). \end{array}$$

- Si se necesitan todas, mejor usar el triangulo de Pascal (DP bottom-up).
- Si k es acotado (por $K < n$), usar el triángulo de Pascal sólo hasta K.
- Usar la simetría $C(n,k) = C(n,n-k)$.

UVA 11955

John likes mathematics a lot. His main passion is the binomial theorem. However it is rather hard to calculate binomial coefficients, so he decided to write a computer program that can expand any power of a sum into a sum of powers. Mathematically it can be written like this:

$$(a+b)^k = x_1a^k + x_2a^{k-1}b + x_3a^{k-2}b^2 + ... + x_{k+1}b^k$$

where $x_1 ... x_{k+1}$ are binomial coefficients $x_i = C(k, i)$.

Input: There is a number of tests T (T $\leq$ 100) on the first line. After T test follows. Each test is written on a single line in form of '(a+b)^k'. Where a and b are same variables names. Variables names are strings constructed from 'a'–'z' characters. And k ($1 \leq k \leq 50$) is a power that you need to raise the sum. You can assume that there are no lines longer than 100 characters.

Output: For each test output a single line 'Case N: T'. Where N is the test number (starting from 1) and T is an expanded expression (see examples for clarification). By the way, you shouldn't output coefficients and powers equal to one.

En este problema (como en muchos otros): **precalcular todos los coeficientes binomiales por el triángulo de Pascal**.

1.3 Applicaciones de los coeficientes binomiales

Teorema

Para cualquier par de enteros $n, k > 0$, el número de k-tuplas (es decir, **conjunto ordenado**) de enteros positivos cuya suma es igual a n es igual al número de sub-conjuntos de $k - 1$ elementos elegidos entre $n - 1$.

- Se puede ver la $k-$tupla como una "barra" de n unidades en total separada en k partes.
- Para generarla, hay que elegir $k - 1$ separadores entre $n - 1$ posibles.
- Entonces son en total $C(n - 1, k - 1)$ maneras para generar una.

Teorema

Para cualquier par de enteros n, $k > 0$, el número de k-tuplas de enteros no-negativos cuya suma es igual a n es igual al número de multisets de cardinal $k - 1$, con elementos elegidos en un conjunto de $n + 1$ elementos. Este número es $C(n + k - 1, n)$.

Exactamente el mismo razonamiento que arriba: la posibilidad de tener 0 ahora simplemente modifica el problema, en el sentido que se puede poner:

- dos separadores en el mismo espacio (o sea elegir dos o más veces el mismo espacio-separador)
- los espacios de inicio y final.

Por eso equivale a contar los **multisets de cardinal** $k - 1$ (posiciones de los espacios) entre $n + 1$ valores posibles (posiciones posibles entre datos).

Además, el problema es equivalente a elegir n lugares donde se pondrán datos, entre $n + k - 1$ posibles posiciones de dato o separador; entonces el $C(n + k - 1, n)$ (o, equivalentemente, los $k - 1$ lugares donde poner los espacios).

Ejemplo

Tenemos un grupo de 16 alumnos de la clase de Programación eficiente. Si las calificaciones posibles del examen están entre 0 y 10, cuántas distribuciones (histogramas) posibles de calificaciones hay?

Aquí, $k = 11$ y $n = 16$.

Entonces, el número buscado es $C(26, 16) = 5311735$.

UVA 10541

You are given a rectangle $1 \times N$, where its 1×1 squares can be painted as white or black. So, one can build a "code" of this rectangle — this will be a sequence of numbers, the number of consequent black squares from left to right. For example, the code of this rectangle is 2 3 2 8 1.

However, the number of white squares is not counted anywhere (but two groups of black squares must be separated by at least one white square). That is why, there are a few rectangles which can satisfy the same code. For example, the following rectangle satisfies the given code as well. The problem is to calculate the number of rectangles, which can satisfy the given code.

Input: There can be multiple test cases. The first line of the input gives you the number of test cases T ($1 < T < 20$). In the next T lines you would have the input for each of the test cases. Each test case consists of N the length of a rectangle ($1 \leq N \leq 200$). Then K – the number of numbers in a code ($0 \leq K \leq\leq (N+1)/2$). Then K numbers, which represent the code itself.

Output: The output consists of one number — the number of rectangles, which can satisfy the given code. You can assume that the output will always fit in a 50 digit integer.

El problema se convierte facilmente en encontrar el número de posibilidades de colocar $N - S - K + 1$ elementos blancos en $K + 1$ grupos eventualmente nulos (entre cada par de grupos negros, o al inicio, o al final), donde S es el número total de negros.

- Corresponde al teorema de arriba.
- De manera general el problema de contar las distribuciones posibles de q objetos idénticos entre p grupos se puede ver como el de contar la manera de elegir $p-1$ "separadores" en un grupo de $q+p-1$ elementos (los que no son separadores serán los objetos idénticos, agrupados). Entonces: $C(q+p-1, p-1)$ maneras.
- En este problema: $q = N - S - K + 1$, y $p = K + 1$ por lo que la solución es:

$$C(N - S + 1, K)$$

- BigInteger o implementación similar...

```
In [ ]: int main() {
            ios::sync_with_stdio(false);
            fillBinomials(256);
            int nCases;
            cin >> nCases;
            for (int i=0;i<nCases;i++) {
                int G,N,S=0;
                cin >> N >> G;
                for (int j=0;j<G;j++) {
                    int L;
                    cin >> L;
                    S+=L;
                }
                if (N-S+1>=0 && N-S+1-G>=0) {
                    // With my BigInteger implementation
```

```
                string &ss = binomials[N-S+1][N-S+1-G].s;
                int j=0;
                if (ss.size()) {
                    cout << ss << endl;
                } else {
                    cout << "0" << endl;
                }
            } else {
                cout << "0" << endl;
            }
        }
    }
```

1.4 Números de Catalan

Una pregunta clásica (cf. **UVA 10303**) es el cálculo del **número de árboles binarios** que se puede formar con n **nodos identificados**.

Un análisis simple (cuántos nodos se ponen a la izquierda de la raiz) lleva a la recurrencia:

$$\begin{cases} C_1 & = & 1 \\ C_{n+1} & = & \sum_{i=0}^{n} C_i C_{n-i}. \end{cases}$$

Son los **números de Catalan** $(1, 1, 2, 5, 14, 42...)$.

Otros problemas de combinatoria resueltos por los números de Catalan:

- Número de palabras de Dyck de tamaño 2n (en estas palabras, hay siempre, en cualquier sub-cadena empezando en el primer caracter, un número de 'A' superior o igual al de 'B').
- Número de expresiones válidas con n **pares de parentesis** ('A'='(' y 'B=')').
- En una rejilla $n \times n$, número de caminos que **van de una extremidad de la diagonal a la otra (por elementos horizontal/vertical) que no cruzan la diagonal** ('A'='-' y 'B'='|').
- Número de maneras de calcular **una secuencia de adiciones/multiplicaciones entre** $n+1$ **elementos** (colocando paréntesis).
- Número de maneras de particionar un **poligono de** $n+2$ **vértices en** n **triángulos** a partir de los vertices del poligono (un triángulo es un nodo de arból binario).

Se calculan también con otras formulas equivalentes:

$$C_n = \frac{1}{n+1}\frac{(2n)!}{n!}.$$

$$C_{n+1} = 2\frac{(2n+1)}{(n+2)}C_n.$$

Asíntotaticamente:

$$C_n \approx \frac{4^n}{n^{3/2}\sqrt{\pi}}$$

Importante para determinar si se tiene que usar BigInteger...

UVA 10312

In this problem you will have to find in how many ways n letters can be bracketed so that the bracketing is non-binary bracketing. For example 4 letters have 11 possible bracketing: xxxx, (xx)xx, x(xx)x, xx(xx), (xxx)x, x(xxx), ((xx)x)x, (x(xx))x, (xx)(xx), x((xx)x), x(x(xx))

Of these the first six bracketing are not binary. Given the number of letters you will have to find the total number of non-binary bracketing.

Input: The input file contains several lines of input. Each line contains a single integer n $(0 < n \leq 26)$. Input is terminated by end of file.

Output: For each line of input produce one line of output which denotes the number of non binary bracketing with n letters.

La diferencia entre esta organización de las parentesis es que **no es para manejar 2 operandos exactamente, sino 2 o mas** (lo que da una combinatoria mucho más grande).

Con este problema aparece otro tipo de números, relacionados a los números de Catalan: los números de **Schroder-Hipparchus** (o también llamados super números de Catalan).

- Número de maneras de dividir un poligono de n + 1 vértices en poligonos (no necesariamente triangulos).
- Número de maneras de formar un arbol plano con n nodos con los nodos internos habiendo al menos dos hijos.
- Número de maneras de insertar parentesis en una secuencia de n simbolos, las parentesis alrededor de **2 o mas simbolos o parentesis, sin parentesis alrededor de la expresion entera**.

Satisfacen:

$$S_n = \frac{1}{n}((6n-9)S_{n-1} - (n-3)S_{n-2})$$

Se deduce la solución del **UVA 10312** restando C_n (que cuenta los binarios) a S_n .

UVA 10843

A little girl whose name is Anne Spetring likes to play the following game. She draws a circle on paper. Then she draws another one and connects it to the first cicrle by a line. Then she draws another and connects it to one of the first two circles by a line. She continues this way until she has n circles drawn and each one connected to one of the previously drawn circles. Her circles never intersect and lines never cross. Finally, she numbers the circles from 1 to n in some random order. How many different pictures can she draw that contain exactly n circles? Two pictures are different if one of them has a line connecting circle number i to circle number j, and the other picture does not.

Input: The first line of input gives the number of cases, N. N test cases follow. Each one is a line containing n $(0 < n \leq 100)$.

Output: For each test case, output one line containing 'Case #x:' followed by X, where X is the remainder after dividing the answer by 2000000011.

Por construcción, la nina está constuyendo un **arból**. La pregunta es **cuántos árboles distintos se puede formar con n nodos**.

1.5 Formula de Cayley

El número de árboles que se pueden construir con n vértices etiquetados es n^{n-2} (formula de **Cayley**).

Prueba por Jim Pitman: contar de dos maneras distintas el **número de secuencias diferentes de inserciones de aristas dirigidas en un grafo con n vértices para generar un arból enraizado**.

Para generar una de esa secuencia, multiplicar:

- T_n el **número de árboles posibles** (lo que estamos buscando).
- n el **número de raices posibles** (una raiz impone una dirección a cada arista)
- $(n-1)!$ la **secuencia de aristas** que insertar (entre $n-1$ aristas). La orientación de cada arista está definida únicamente una vez que la raíz y el arból están definidos.

Tenemos entonces $T_n n!$ secuencias.

Ahora, podemos contar este número de otra manera: observar que **después de haber agregado $n-k$ aristas dirigidas, tenemos necesariamente un bosque de k arboles enraizados**.

Para la inserción siguiente, podemos elegir la arista dirigida saliendo de un nodo entre n, y llegando a un nodo entre las $k-1$ raices que no son la raiz en la que está el nodo selecionado. Entonces $n(k-1)$ elecciones posibles.

Al realizar las n $-$ 1 inserciones:

$$\prod_{k=n}^{2} n(k-1) = n^{n-1}(n-1)! = n^{n-2}n!.$$

Se deduce de las dos formulas:

$$T_n = n^{n-2}.$$

UVA 12024

John Hatman, the honest cloakroom attendant of the Royal Theatre of London, would like to know the solution to the following problem. When the show finishes, all spectators in the theatre are in a hurry to see the Final of the UEFA Championship. So, they run to the cloakroom to take their hats back. Some of them take a wrong hat. But, how likely is that everyone take a wrong hat?

Input The first line of the input contains an integer, t, indicating the number of test cases. For ach test case, one line appears, that contains a number n, $2 \leq n \leq 12$, representing the number of people and hats.

Output: For each test case, the output should contain a single line with the number representing the number of favourable cases (i.e., the number of cases where all people take a wrong hat), followed by a bar, '/', and followed by a number representing the total number of possible cases.

1.6 Desarreglos

Es una permutación de n elementos donde el elemento i, para $i \in [1, n]$, **no se va al elemento** i (corresponde al caso enunciado en el problema **UVA 12024**).

El número de desarreglos es acotado por $n!$ (es un subconjunto de las permutaciones).

Teorema

Para cualquier $n > 0$, hay $!n$ desarreglos de $[1, n]$, donde $!n$ es llamado subfactorial y es dada recursivamente por:

$$\begin{cases} !0 &= 1 \\ !1 &= 0 \\ !n &= (n-1)(!(n-1)+!(n-2)) \end{cases}$$

Prueba: Suponer que el elemento 1 va en posición $i \neq 1$ para i. Hay $n-1$ elecciones posibles. Dos casos luego:

- El elemento i va con elemento 1. Entonces nos queda el problema de asociar $n-2$ elementos con su lugar, donde cada elemento tiene 1 prohibido (sí mismo).
- El elemento i no va con elemento 1. Entonces nos queda el problema de asociar $n-1$ elementos con su lugar, donde cada elemento tiene 1 prohibido (el elemento i tiene prohibido el 1, los $j \neq i$ tienen prohibidos el j).

Entonces:

$$!n = (n-1)(!(n-1)+!(n-2)).$$

UVA 10497

Children are always sweet but they can sometimes make you feel bitter. In this problem, you will see how Tintin, a five year's old boy, creates trouble for his parents. Tintin is a joyful boy and is always busy in doing something. But what he does is not always pleasant for his parents. He likes most to play with household things like his father's wristwatch or his mother's comb.

After his playing he places it in some other place. Tintin is very intelligent and a boy with a very sharp memory. To make things worse for his parents, he never returns the things he has taken for playing to their original places. Think about a morning when Tintin has managed to 'steal' three household objects.

Now, in how many ways he can place those things such that nothing is placed in their original place. Tintin does not like to give his parents that much trouble. So, he does not leave anything in a completely new place; he merely permutes the objects.

Input: There will be several test cases. Each will have a positive integer less than or equal to 800 indicating the number of things Tintin has taken for playing. Each integer will be in a line by itself. The input is terminated by a '-1' (minus one) in a single line, which should not be processed.

Output: For each test case print an integer indicating in how many ways Tintin can rearrange the things he has taken.

1.7 Números de Stirling

Los **números de Stirling de primer tipo**, escritos

$$\left[\begin{matrix} n \\ k \end{matrix} \right]$$

corresponden al **número de permutaciones de n elementos con k ciclos exactamente**.

Siguen la relación de recurrencia siguiente:

$$\left[\begin{array}{c} n+1 \\ k \end{array} \right] = n \left[\begin{array}{c} n \\ k \end{array} \right] + \left[\begin{array}{c} n \\ k-1 \end{array} \right]$$

para $k > 0$ y con:

$$\left[\begin{array}{c} 0 \\ 0 \end{array} \right] = 1$$

y

$$\left[\begin{array}{c} n \\ 0 \end{array} \right] = \left[\begin{array}{c} 0 \\ n \end{array} \right] = 0$$

para $n > 0$.

Los **números de Stirling de segundo tipo**, escritos $S(n,k)$ o

$$\left\{ \begin{array}{c} n \\ k \end{array} \right\}$$

corresponden al número de maneras de **particionar un conjunto de n objetos etiquetados en k sub-conjuntos (sin etiqueta) no-vacíos**.

Siguen la relación de recurrencia siguiente:

$$\left\{ \begin{array}{c} n+1 \\ k \end{array} \right\} = k \left\{ \begin{array}{c} n \\ k \end{array} \right\} + \left\{ \begin{array}{c} n \\ k-1 \end{array} \right\}$$

para k > 0 con:

$$\left\{ \begin{array}{c} 0 \\ 0 \end{array} \right\} = 1$$

y

$$\left\{ \begin{array}{c} n \\ 0 \end{array} \right\} = \left\{ \begin{array}{c} 0 \\ n \end{array} \right\} = 0$$

para $n > 0$.

Prueba: considerar el elemento $n+1$. Dividiremos las particiones de $[1, n+1]$ entre las particiones que tienen a $n+1$ de singletón, y las otras.

- Las $k-$particiones que contienen a $n+1$ de singletón son:

$$\left\{ \begin{array}{c} n \\ k-1 \end{array} \right\}$$

- Las $k-$particiones en las que $n+1$ no está en un singletón son tales que se elige $k-$particiones con los n elementos restantes, y queda ubicar a $n+1$ dentro de una de las particiones (k posibilidades).

De ahí:

$$\left\{ \begin{array}{c} n+1 \\ k \end{array} \right\} = k \left\{ \begin{array}{c} n \\ k \end{array} \right\} + \left\{ \begin{array}{c} n \\ k-1 \end{array} \right\}.$$

1.8 Lema de Burnside

UVA 10733

All 6 sides of a cube are to be coated with paint. Each side is is coated uniformly with one color. When a selection of n different colors of paint is available, how many different cubes can you make? Note that any two cubes are only to be called "different" if it is not possible to rotate the one into such a position that it appears with the same coloring as the other.

Input: Each line of the input file contains a single integer n ($0 < n < 1000$) denoting the number of different colors. Input is terminated by a line where the value of $n = 0$. This line should not be processed.

Output: For each line of input produce one line of output. This line should contain the number of different cubes that can be made by using the according number of colors.

Sea G un grupo finito (e.g. de transformaciones) que actúen sobre un conjunto X. Sea X^g , para $g \in G$, el subjonjunto de X de los elementos **dejados invariantes por** g. Tenemos:

$$|X/G| = \frac{1}{|G|} \sum_{g \in G} |X^g|.$$

donde $|X/G|$ es el número de órbitas dentro de X por la acción de G (es decir el número de trayectorias **distintas** por aplicaciones de los g).

Por ejemplo, en $\mathbb{R}^3$, con el grupo $SO(3)$ de las rotaciones $3D$, las orbitas son las esferas centradas en la origen (cada punto de una esfera se deduce de otro punto por una rotación).

Consecuencia: para calcular el número de órbitas, puedes promediar el número de los elementos invariantes entre todas las transformaciones de un grupo de transformaciones.

Para regresar al problema **UVA 10733**: Sea n el número de colores posibles.

- X aquí es el conjunto de las n^6 combinaciones de colores en el cubo (con una orientación particular).
- G es el grupo de las rotaciones 3D de $\pm k\pi/2$ que llevan a la orientación deseada.
- Dos elementos de X están en la misma orbita cuando se deducen uno del otro por una de estas rotaciones: El número que buscamos es entonces el número de orbitas $|X/G|$.

Usando el lema, podemos entonces promediar el número de invariantes para todas las rotaciones de G. Las rotaciones son de 4 tipos:

- la identidad ($\times 1$) que deja los n^6 elementos de X invariantes;
- las rotaciones de ± 90 grados alrededor de las caras ($\times 6$): $n^2 \times n = n^3$ elementos invariantes (2 caras del eje de rotación y configuraciones de 4 colores idénticas);
- las rotaciones de 180 grados alrededor de las caras ($\times 3$): $n^2 \times n^2 = n^4$ elementos invariantes (caras del eje de rotación y configuraciones de 2 colores cara a cara idénticas);
- las rotaciones de ± 120 grados por los ejes-diagonales del cubo ($\times 8$): n^2 elementos invariantes (configuraciones de 3 colores consecutivas idénticas ortogonalmente al eje de rotación);

- las rotaciones de 180 grados alrededor de ejes juntando los promedios de aristas opuestas y no presentes en la misma cara ($\times 6$): n^3 elementos invariantes (configuraciones con 3 pares iguales).

En total:

$$\frac{1}{24}(n^6 + 3n^4 + 12n^3 + 8n^2).$$

Da la respuesta al problema.