## 3499. Суммирование подмножеств

Через G(S) обозначим сумму элементов множества S, а F(*n*) – сумму значений G(S) по всем подмножествам множества, состоящего из первых *n* натуральных чисел. Например,

F(3) = (1) + (2) + (3) + (1 + 2) + (1 + 3) + (2 + 3) + (1 + 2 + 3) = 24

Для заданного числа *n* найдите значение суммы F(1) + F(2) + … + F(*n*).

**Вход.** Первая строка содержит количество тестов *t* (*t* ≤ 1000) . Каждая из следующих *t* строк содержит одно целое число *n* (1 ≤ *n* ≤ 109).

**Выход.** Выведите *t* строк – по одному числу в каждой строке для соответствующего теста. Поскольку ответы могут быть очень большими, выводите их по модулю 8388608.

|  |  |
| --- | --- |
| **Пример входа** | **Пример выхода** |
| 3  1  2  3 | 1  7  31 |

**РЕШЕНИЕ**

**комбинаторика**

# Анализ алгоритма

Найдем формулу для F(*n*).

***Теорема.*** Каждый элемент *i* входит в ровно половину всех подмножеств, то есть в 2*n* – 1 подмножеств.

***Доказательство.*** Общее число всех (включая пустое) подмножеств множества {1, 2, …, *n*} равно 2*n*. Фиксируем некоторый элемент *i*. Любое подмножество, в котором присутствует *i*, можно задать однозначно выбором подмножества из оставшихся *n* – 1 элементов. Поэтому число подмножеств, содержащих *i*, равно числу всех подмножеств множества размера *n* – 1, то есть 2*n*-1. Но 2*n*-1 – ровно половина от 2*n*. Значит элемент *i* входит в ровно половину всех подмножеств.

Следовательно:

F(*n*) =  = = 2*n* – 2 \* *n* (*n* + 1)

Нам следует вычислить сумму:

= =

=

Осталось вычислить указанную сумму. Рассмотрим две формулы:

= (*n* – 1) \* 2*n* + 1 + 2,

= (*n2* – 2*n* + 3) \* 2*n* + 1 – 6

Подставляем их в выражение:

= ((*n2* – 2*n* + 3) \* 2*n* + 1 – 6) + ((*n* – 1) \* 2*n* + 1 + 2) =

2*n* + 1 (*n*2 – *n* + 2) – 4

Следовательно:

= (2*n* + 1 (*n*2 – *n* + 2) – 4) = 2*n* – 1 (*n*2 – *n* + 2) – 1

**Пример**

# Вычислим некоторые значения функции F(*n*):

F(1) = (1) = 1,

F(2) = (1) + (2) + (1 + 2) = 6,

F(3) = (1) + (2) + (3) + (1 + 2) + (1 + 3) + (2 + 3) + (1 + 2 + 3) = 24

Таким образом,

F(1) = 1,

F(1) + F(2) = 1 + 6 = 7,

F(1) + F(2) + F(3) = 1 + 6 + 24 = 31

Вычислим эти же суммы используя формулу:

F(1) = 21 – 1 (12 – 1 + 2) – 1 = 1,

F(1) + F(2) = 22 – 1 (22 – 2 + 2) – 1 = 7,

F(1) + F(2) + F(3) = 23 – 1 (32 – 3 + 2) – 1 = 31

# Реализация алгоритма

Объявим константу - модуль.

#define MOD 8388608 // 2^23

Функция ***PowMod*** вычисляет значение *xn* mod *m*.

long long PowMod(long long x, long long n, long long m)

{

if (n == 0) return 1;

if (n % 2 == 0) return PowMod((x \* x) % m, n / 2, m);

return (x \* PowMod(x, n - 1, m)) % m;

}

Основная часть программы. Читаем количество тестов *tests*.

scanf("%d", &tests);

while (tests--)

{

Читаем входное значение *n*.

scanf("%lld", &n);

Вычисляем ответ по формуле.

res = PowMod(2, n - 1, MOD); // 2^(n-1) mod MOD

t = ((n % MOD) \* (n % MOD)) % MOD; // n^2

t = (t - n + MOD + 2) % MOD; // n^2 - n + 2

res = (res \* t - 1 + MOD) % MOD;

Выводим ответ.

printf("%lld\n", res);

}