**6960. Сумма различными числами**

Натуральное число *n* можно представить в виде суммы различных натуральных чисел несколькими способами. Например:

* для *n* = 5 существует 3 способа: 5, 2 + 3, 1 + 4;
* для *n* = 6 существует 4 способа: 6, 1 + 5, 1 + 2 + 3, 2 + 4.

При этом перестановки одних и тех же слагаемых считаются одним и тем же способом. То есть, например, 1 + 2 + 3, 2 + 1 + 3 и 3 + 1 + 2 считаются одним способом.

**Вход.** Первая строка содержит количество тестов *t* (1 ≤ *t* ≤ 20).

В следующих *t* строках заданы сами тесты – по одному числу *n* (1 ≤ *n* ≤ 2000) в строке.

**Выход.** Для каждого числа *n* в отдельной строке выведите количество различных способов его представления в виде суммы различных натуральных чисел, как описано выше.

Так как ответ может быть очень большим, выведите его по модулю 100999.

|  |  |
| --- | --- |
| **Пример входа** | **Пример выхода** |
| 4  5  6  10  200 | 3  4  10  50568 |

## РЕШЕНИЕ

**динамическое программирование**

**Анализ алгоритма**

Пусть dp[*i*][*j*] – количество способов представить число *i* в виде суммы различных чисел, каждое из которых не превышает *j* (0 ≤ *i*, *j* ≤ 2000).

Например:

|  |  |
| --- | --- |
| * dp[6][0] = dp[6][1] = 0 | * dp[6][4] = 2: 5 + 1, 4 + 2 |
| * dp[6][2] = 0 | * dp[6][5] = 3: 5 + 1, 4 + 2, 3 + 2 + 1 |
| * dp[6][3] = 1: 3 + 2 + 1 | * dp[6][6] = 4: 6, 5 + 1, 4 + 2, 3 + 2 + 1 |

Для суммы 0 существует ровно один способ – пустая сумма (ничего не складывать). Поэтому для всех *j* присвоим dp[0][*j*] = 1.

Значение dp[*i*][*j*] пересчитываем следующим образом:

* Если *j* > *i*, значит текущее максимальное число *j* больше, чем сумма *i*, и мы не можем его использовать. Тогда количество способов равно количеству способов представить *i* при помощи чисел от 1 до *j* – 1:

dp[*i*][*j*] = dp[*i*][*j* – 1]

* Если *j* ≤ *i*, то мы можем либо не использовать число *j* (тогда способов dp[*i*][*j* – 1]), либо использовать *j* (тогда мы ищем количество способов представить сумму *i* – *j* при помощи чисел от 1 до *j* – 1, чтобы избежать повторного использования числа *j*):

dp[*i*][*j*] = dp[*i*][*j* – 1] + dp[*i* – *j*][*j* – 1]

Для заданного числа *n* ответом будет значение dp[*n*][*n*].

**Реализация алгоритма**

Объявим константы.

#define MAX 2001

#define MOD 100999

Объявим массив *dp*.

vector<vector<int>> dp(MAX, vector<int>(MAX, 0));

Базовый случай. Единственный способ получить сумму 0 – это пустое множество.

for (j = 0; j < MAX; j++)

dp[0][j] = 1;

Вычисляем значения массива *dp*.

for (i = 1; i < MAX; i++)

for (j = 1; j < MAX; j++)

{

if (j > i)

dp[i][j] = dp[i][j - 1];

else

dp[i][j] = (dp[i][j - 1] + dp[i - j][j - 1]) % MOD;

}

Читаем количество тестов *tests*.

scanf("%d", &tests);

Последовательно обрабатываем *tests* тестов. Для каждого входного значения *n* выводим ответ dp[*n*][*n*].

while (tests--)

{

scanf("%d", &n);

printf("%d\n", dp[n][n]);

}