백준 15992번 - 1,2,3 더하기 7

시간제한	메모리 제한	제출	정답	맞은 사람	정답 비율
0.25초	512MB	912	482	383	55.267%

문제

정수 4를 1, 2, 3의 합으로 나타내는 방법은 총 7가지가 있다. 합을 나타낼 때는 수를 1개 이상 사용해야 한다.

- 1+1+1+1
- 1+1+2
- 1+2+1
- 2+1+1
- 2+2
- 1+3
- 3+1

정수 n과 m이 주어졌을 때, n을 1, 2, 3의 합으로 나타내는 방법의 수를 구하는 프로그램을 작성하시오. 단, 사용한 수의 개수는 m개 이어야 한다.

입력

첫째 줄에 테스트 케이스의 개수 T가 주어진다. 각 테스트 케이스는 한 줄로 이루어져 있고, 정수 n과 m이 주어진다. n은 양수이며 1,000보다 작거나 같다. m도 양수이며, n보다 작거나 같다.

출력

각 테스트 케이스마다, n을 1, 2, 3의 합으로 나타내는 방법의 수를 1,000,000,009로 나눈 나머지를 출력한다. 단, 사용한 수의 개수는 m개 이어야 한다.

예제 입력1

```
3
4 2
7 5
10 6
```

예제 출력1

```
3
15
90
```

출처

출처

알고리즘 분류

- 다이나믹 프로그래밍

접근 방법

우리가 찾고자 하는 정답이 {1,2,3}이라는 숫자를 m개 이용하여 n이라는 수를 만드는 것이므로 1, 2, 3으로 시작하는 State들을 그려본다면 쉽게 규칙성을 발견할 수 있다.

만약, 규칙이 눈에 들어오지 않는다면 Top-bottom 방식의 DP도 시도해볼만하다. 하지만, 위 문제에서는 시간 제한이 0.25초이므로 최대한 함수 호출에 따른 오버헤드를 줄이고자 규칙성을 이용한 bottom-top 방식을 이용하여 메모이제이션 테이블을 채워주었다.

위 문제 풀이에 핵심이 되는 부분은 아래와 같다.

`DP[i][j] : i라는 수를 j개의 숫자의 조합으로 만들 수 있는 경우의 수`

위와 같이 메모이제이션 테이블을 정의한다면 `DP[i][j]`에서 확장되는 새로운 상태는 아래와 같이 3가지가 된다.

`1. DP[i+1][j+1], 2. DP[i+2][j+1], 3. DP[i+3][j+1]`

규칙성을 이용하여 Recurrence Relation을 다음과 같이 정의할 수 있다.

$$dp[i][j] = (dp[i-1][j-1] + dp[i-2][j-1] + dp[i-3][j-1])$$

이 후 Base Case를 고려하여 점화식을 구성한다면 아래와 같은 완전한 형태의 점화식을 얻을 수 있다.

$$\begin{aligned} dp[1][1] = dp[2][1] = dp[3][1] = 1 \quad &- base\ condition \\ dp[i][j] = dp[i-1][j-1] + dp[i-2][j-1] + dp[i-3][j-1] \quad &- relation \end{aligned} \tag{1}$$

초반에 1000 x 1000 테이블을 채워둔다면 이 후에 주어지는 T개의 Query에 대하여 $O(1)$에 처리가 가능하므로 전체 소스코드의 시간 복잡도는 $O(K^2)$ 가 된다. (k = 1000)

소스코드

```cpp
#include <bits/stdc++.h>
using namespace std;

typedef long long ll;
const ll MOD = 1000000009LL;
ll dp[1001][1001] = {0, };
int main(void){
    int T;
    cin >> T;
    dp[1][1] = dp[2][1] = dp[3][1] = 1;
    // dp[i][j] : i를 j개의 수를 이용하여 만드는 경우의 수
    for(int j=2; j<=1000; j++){
        for(int i=2; i<=1000; i++){
            dp[i][j] += dp[i-1][j-1], dp[i][j] %= MOD;
            if(i-2 >= 0) {
                dp[i][j] += dp[i-2][j-1];
                dp[i][j] %= MOD;
            }
            if(i-3 >= 0){w
                dp[i][j] += dp[i-3][j-1];
                dp[i][j] %= MOD;
            }
        }
    }
    while(T--){
        ll n, m;
        cin >> n >> m;
        cout << dp[n][m] << '\n';
    }
    return 0;
}
```