 풀이 과정

문제는 뭔가 많이 어렵게 설명을 해 놨는데, 그림을 그려서 선을 이어보면 dp[i]=dp[i-1]+dp[i-2](i>2)이 된다. i=1이면 당연히 1개이고, i=2이면 양옆을 서로 직선으로 잇는 경우와 엇갈려서 잇는 경우 2가지가 된다. 따라서 해당 점화식을 사용하여 dp를 진행하면 된다. 이때 시간제한이 0.5초이기 때문에 빠르게 실행하기 위해서 메모이제이션을 추가하였다. 또한 메모이제이션 만으로는 시간초과가 떠서 인터넷에 알아보니 ios::sync\_with\_stdio(false)와 cin.tie(NULL);이 컴파일 속도를 빠르게 해준다는 사실을 알게 되어 추가해보니 성공하였다!

**1. ios::sync\_with\_stdio(false);**

C 표준 stream과 C++ 표준 stream의 동기화를 끊습니다.

기본적으로, 모든 표준 stream들은 동기화 되어있습니다. 그래서 우리는 C와 C++의 입출력방식을 자유롭게 혼용할 수 있습니다. 동기화를 끊는다면, C++ stream들은 독립적인 버퍼를 갖게되며, C와 C++의 입출력방식을 혼용해서 쓰는 것이 굉장히 위험해집니다, 또한 동기화된 C++ stream은 thread-safe합니다. (다른 thread의 output이 동시에 액세스해도 충돌하지 않습니다.)

동기화를 끊으면 사용하는 버퍼의 수가 줄어들기 때문에 실행속도 자체는 향상됩니다.

2. cin.tie(NULL);

cin을 cout으로부터 untie합니다. stream을 tie하면 다른 stream에서 입출력요청이 오기전에 stream을 flush시킵니다.:

#include <iostream>

#include <string.h>

using namespace std;

int dp[191229] = {1,2};

int main(void) {

ios::sync\_with\_stdio(false); //통과의 주요 요소

cin.tie(NULL); cout.tie();

int t;

cin >> t;

for (int i = 2; i < 191229; i++) { //메모이제이션

dp[i] = (dp[i - 1] + dp[i - 2]) % 1000000007;

}

while (t-- > 0) {

int n;

cin >> n;

cout << dp[n - 1] <<"\n";

}

return 0;

}