CRACK A HACK

POPSICLE STICK MOUNTAINS

COURSE: ALGORITHMIC PROBLEM SOLVING

COURSE CODE: 17ECSE309

By:

Prathmesh Sambrekar

USN:01FE15BEC133.

1.**INTRODUCTION**:

Pospsicle stick mountains is a arrangement of ‘n’ sticks such that there are ‘n’ upstrokes and ‘n’ downstrokes. The number of possible ways for formation of such an arrangement is given by catalan numbers. The Catalan numbers are a sequence of positive integers that appear in many counting problems in combinatorics. They satisfy the following equation:

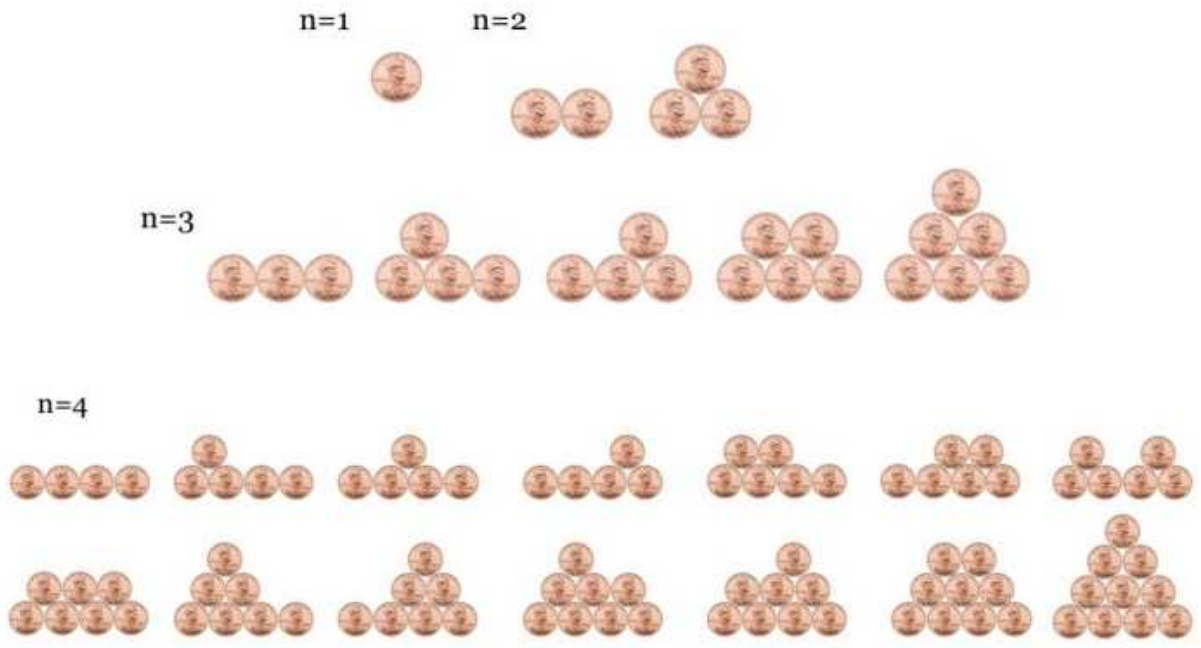
$$C(n)=(2n)!/(n+1)!*(n)!$$

Where, C(n) is the nth catalan number.

2.**EXAMPLE**:

Stacking coins:

We are going to stack ‘n’ coins on a bottom row that consists of n consecutive coins. It is not allowed to put the coins on the two sides of the bottom coins. So the number of ways to stack coins on the ‘n’ coins is given by the catalan of ‘n’.



3.**HACKER RANK QUESTION LINK**:

https://www.hackerrank.com/contests/walmart-codesprint-algo/challenges/popsicle-stick-mountains.

4.**CODE (In C++)**:

```
#include<cstdio>
#include<fstream>
#include<algorithm>
#include<cmath>
#include<string>
#include<cstring>
#include<vector>
#include<cmath>
#include<map>
#include<iostream>
#include<cstdlib>
#include<set>
#include<sstream>
#include<stack>
#include<list>
```

```
#include<cstdlib>
#include<queue>
#include<iterator>
using namespace std;

typedef long long ll;
typedef pair<long long,long long> lll;
typedef pair<long long,long long> lli;
typedef pair<long long,long long> ii;

#define EL printf("\n")
#define OK printf("OK")
#define pb push_back
#define mp make_pair
#define ep emplace_back
#define X  first
#define Y  second
```

```
#define fillchar(a,x) memset(a, x, sizeof(a))

#define FOR(i,l,r) for (long long i=l;i<=r;i++)

#define FORD(i,r,l) for (long long i=r;i>=l;i--)

#define Get getchar()

Int getInt() { int a=0,s=1; char c=0; while(c<33)
c=Get; if(c=='-') {s=-1; c=Get;} while(c>33)
{a=(a<<3)+(a<<1)+c-'0'; c=Get;} return a*s; }



const int maxn = 3e5+7;


long long mod = 1000000007;


int main() {
```

```
int t;
t = getInt();
//t=1;
long long catalan[2000+2];

 // Initialize first two values in table
 catalan[0] = catalan[1] = 1;

// Fill entries in catalan[] using recursive formula
for (int i=2; i<=2001; i++)
{
	catalan[i] = 0;
	for (int j=0; j<i; j++)
                                        catalan[i] =
		(catalan[i]+(catalan[j] * catalan[i-j-
		1]))%mod;
}
```

```
        while(t--)
 {
    int n;
    n = getInt();
    n /=2;
    long long ans = 0;
    for(int i=1;i<=n;i++)
    {
        ans = (ans+catalan[i])%mod;
    }
    printf("%lld\n",ans);
}
return 0;
}
```

5.**EXPLANATION**:

- Divide the number of sticks ‘n’ given by 2.So that there are ‘n/2’ upstrokes and ‘n/2’ downstrokes.
- Find the catalan numbers of all the numbers ranging from ‘1’ to ‘n/2’ and add all the catalan numbers. The sum will the number of possible ways to form the arrangement.
- To find the catalan numbers we follow the dynamic approach.
- Create a array of size ‘m+1’,where ‘m’ is the number whose catalan number is to be found.
- Assign the first two elements of the array as ‘1’.
- Now multiply the first element of the array with the last element(preceding the index value of the number whose catalan number is to be found). This is done by maintaining two ‘for’ loops.
- Again multiply the second element with the second last element and so on.
- Sum all the products obtained and store it at the index value of the number whose catalan equivalent is to be obtained.

6.**TIME COMPLEXITY**:

Time complexity of the above implementation is O(n^2).

7.**APPLICATIONS**:They have applications in most of the Counting problems in combinatorics.

- Balanced paranthesis.
- Stacking coins.
- Mountain ranges.
- Polygon Triangulation.
- Binary Trees.
- Binary Paths.

8.**REFERENCES**:

- http://mathforum.org/mathimages/index.php/Catalan_Numbers
- https://en.wikipedia.org/wiki/Catalan_number