

CS3201 Algorithm Design Term Project

MADE BY :

NAVIN SINGH 5935221

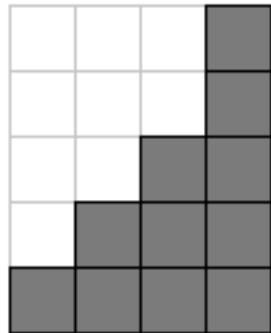
WILLIAM POCH 5938122

ACM.Timus .ru1017. Staircases

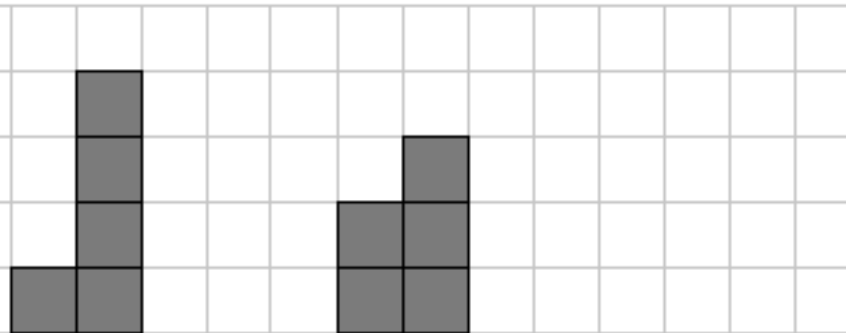
Difficulty:157

One curious child has a set of N little bricks ($5 \leq N \leq 500$). From these bricks he builds different staircases. Staircase consists of steps of different sizes in a strictly descending order. It is not allowed for staircase to have steps equal sizes. Every staircase consists of at least two steps and each step contains at least one brick. Picture gives examples of staircase for $N=11$ and $N=5$:

$N = 11$



$N = 5$



Your task is to write a program that reads the number N and writes the only number Q — amount of different staircases that can be built from exactly N bricks.

Input and Output

► **Input**

Number N

► **Output**

Number Q

Sample

input	output
212	995645335

How did we solve this problem

This problem asks us to build a staircase with N number of bricks.

The staircases must consist of steps of different sizes in decreasing order.

To solve this problem we will use dynamic programming.

One method to solve this problem is to utilize a bottom-up approach.

This method consists of looping through a multi-dimensional array (2D or 3D), starting from the base cases.

Bottom Up Approach

- ▶ Going **bottom-up** is a way to avoid recursion, saving the **memory cost** that recursion incurs when it builds up the **call stack**.
- ▶ Put simply, a bottom-up algorithm "starts from the beginning," while a recursive algorithm often "starts from the end and works backwards."
- ▶ Even though in bottom up all states are visited but recursion eats a lot of memory using top down

This is the Bottom Up Approach

```
from time import time
n = int(input())
m = [[0 for i in range(n + 1)] for j in range(n + 1)]
m[0][0] = 1 # base case
start = time()
for last in range(1, n + 1):
    for left in range(0, n + 1):
        m[last][left] = m[last - 1][left]
        if left >= last:
            m[last][left] += m[last - 1][left - last]
end = time()
print(m[n][n] - 1)
print(end-start)
```

Output using Bottom Up Approach

```
C:\Codes>python Staircase.py  
212  
995645335  
0.03390908241271973
```

Reference

- ▶ Code by Jeremy Tuloup: https://jtp.io/2016/07/26/dynamic-programming-python.html?fbclid=IwAR0q5eQR4zeWoSAQ59t711yCmVZSwY_XkyukvMVG_G9xYIYvLIjZ8u_J2R9Q
- ▶ Timus Problem: <http://acm.timus.ru/problem.aspx?space=1&num=1017>

1017. Staircases



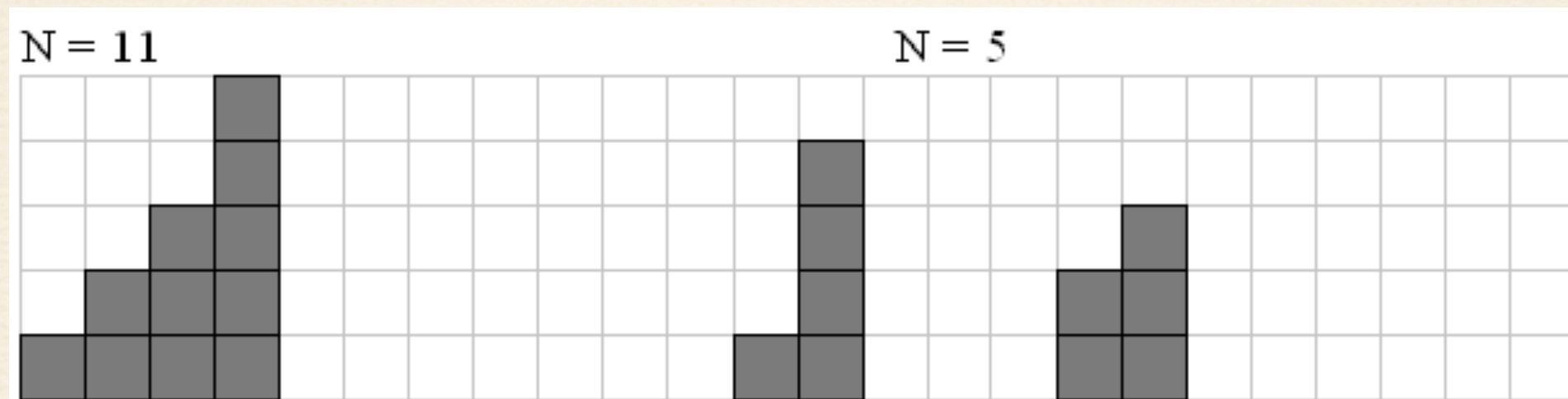
Smith S. 5613456

Pitpiboon P. 5815887

Question

- ❖ One curious child has a set of $\mathcal{N}$ little bricks ($5 \leq \mathcal{N} \leq 500$). From these bricks he builds different staircases. Staircase consists of steps of different sizes in a strictly descending order. It is not allowed for staircase to have steps equal sizes. Every staircase consists of at least two steps and each step contains at least one brick.
- ❖ Time limit: 1.0 second
- ❖ Memory limit: 64 MB

Example



- ❖ Picture gives examples of staircase for $N=11$ and $N=5$:

input	output
212	995645335


```

1 //https://tausiq.wordpress.com/2011/06/16/timus-1017-staircases/
2 //shahab
3 #include <iostream>
4 #define LL long long
5 LL memo [501] [501];
6
7 void reset ()
8 {
9     for ( int i = 0; i < 501; i++ )
10
11         for ( int j = 0; j < 501; j++ ) memo [i] [j] = -1;
12 }
13
14 LL dp (int prev, int remain) A
15 {
16     if ( remain == 0 ) return 1;
17     if ( remain <= prev ) return 0; B
18     if ( memo [prev] [remain] != -1 ) return memo [prev] [remain];
19
20     LL ret = 0;
21     for ( int i = prev + 1; i <= remain; i++ )
22     {
23         ret += dp (i, remain - i); C
24     }
25     //printf("%d prev\n",prev );
26     //printf("%d remain\n",remain );
27     return memo [prev] [remain] = ret;
28 }
29
30 int main ()
31 {
32     int n;
33     while ( scanf ("%d", &n) != EOF ) {
34         reset ();
35         printf ("%lld\n", dp (0, n) - 1); D
36     }
37     return 0;
38 }

```

- ❖ prev = number of bricks you have set on previous column

A

- ❖ remain = number of remaining bricks

- ❖ if remain = 0, then you have got a solution

- ❖ if remain <= prev, then you cant get a solution

B

- ❖ if you had come to this point before then don't calculate one more time, just return the previous calculated result

- ❖ Now you are trying to fill up the current column with every possible number of bricks

C

- ❖ Obviously started from prev + 1 to <= remain

- ❖ prev = 0, remain = n

- ❖ dp (0, n) gives the final result

D

- ❖ but solutions with one column not acceptable so we subtracted 1 from the final result

How It works

```

peet13601 — 1017 — 80x24
Last login: Thu Nov 29 00:27:43 on ttys000
[Smiths-MacBook-Air-2:~ peet13601$ /Users/peet13601/Desktop/1017
5
1 prev
4 remain
2 prev
3 remain
0 prev
5 remain
2 result

```

- ❖ $n=5$
- ❖ 1 and 4
- ❖ 2 and 3
- ❖ 0 and 5 (unacceptable)
- ❖ $3 - 1 = 2$ result

Result

```

[Smiths-MacBook-Air-2:~ peet13601$ /Users/peet13601/Desktop/1017
212
995645335

```

input	output
212	995645335

8153356	19:45:05 28 Nov 2018	peet13601	1017. Staircases	G++ 7.1	Accepted		0.046	2 372 KB
---------	-------------------------	---------------------------	----------------------------------	---------	----------	--	-------	----------

✓	1017	<u>Staircases</u>	USU Local Contest. 1999	<u>8402</u>	157
---	------	-------------------	-------------------------	-------------	-----

Reference

- ❖ <http://acm.timus.ru/problem.aspx?space=1&num=1017>
- ❖ <https://tausiq.wordpress.com/2011/06/16/timus-1017-staircases/> by SHAHAB

Code

```
//https://tausiq.wordpress.com/2011/06/16/timus-1017-staircases/  
//shahab  
#include <iostream>  
#define LL long long  
LL memo [501] [501];  
  
void reset ()  
{  
    for ( int i = 0; i < 501; i++ )  
  
        for ( int j = 0; j < 501; j++ ) memo [i] [j] = -1;  
}  
  
LL dp (int prev, int remain)  
{  
    if ( remain == 0 ) return 1;  
    if ( remain <= prev ) return 0;  
    if ( memo [prev] [remain] != -1 ) return memo [prev] [remain];  
  
    LL ret = 0;  
    for ( int i = prev + 1; i <= remain; i++ )  
    {  
        ret += dp (i, remain - i);  
    }  
    //printf("%d prev\n",prev );  
    //printf("%d remain\n",remain );  
    return memo [prev] [remain] = ret;  
}  
  
int main ()  
{  
    int n;  
    while ( scanf ("%d", &n) != EOF ) {  
        reset ();  
        printf ("%lld\n", dp (0, n) - 1);  
    }  
    return 0;  
}
```