

题目描述

一天，小明跟爸爸去从山脚爬上山顶，途中经过一个有个个台阶的阶梯，他想这要爬一个台阶，每次只能跳1步或跳2步，试问要爬这个阶梯有多少种不同的跳法方式？

输入描述

输入只有一个整数N (0<N<50) 此阶梯有多少个台阶。

输出描述

输出有多少种跳法方式（解决方式见）。

用例

输入	30
输出	132100097
说明	无

输入	3
输出	3
说明	无

题目解析

这题是一道经典的 非递归算法 题，以及动态规划解法。

这题除了使用递归求解外还可以通过递归来求解，也可以使用 动态规划 求解这题网上求解。

使用动态规划求解的步骤。

这题和Fibonacci数列很像，通过公式如下：

- jump[1] = 1
- jump[2] = 1
- jump[3] = 2
- jump[n] = jump[n-1] + jump[n-2] n>3

状态转移方程如下：

- 当n=1
- 当n=2
- 当n=3
- 当n=4
- 当n=5
- 当n=6
- 当n=7
- 当n=8
- 当n=9
- 当n=10
- 当n=11
- 当n=12
- 当n=13
- 当n=14
- 当n=15
- 当n=16
- 当n=17
- 当n=18
- 当n=19
- 当n=20
- 当n=21
- 当n=22
- 当n=23
- 当n=24
- 当n=25
- 当n=26
- 当n=27
- 当n=28
- 当n=29
- 当n=30

上篇递归公式和状态转移方程的递推关系如下

阶数	跳法方式	几种跳法方式
1	1	1
2	11	1
3	111 21	2
4	1111 112 211 22	3
5	11111 1112 1121 1211 2111 221	4
6	111111 11112 11121 11211 12111 21111 2211 231	5
7	1111111 111112 111121 111211 112111 121111 211111 22111 2311 241	6

Java算法源码

非递归算法（带缓存优化）

```
1 import java.util.Scanner;
2 import java.util.Scanner;
3
4 public class Main {
5     static int[] cache = new int[50];
6
7     public static void main(String[] args) {
8         Scanner sc = new Scanner(System.in);
9         int n = sc.nextInt();
10
11         Arrays.fill(cache, -1);
12         cache[1] = 1;
13         cache[2] = 1;
14         cache[3] = 2;
15         cache[4] = 3;
16
17         System.out.println(recursive(n));
18     }
19
20     public static int recursive(int n) {
21         if (cache[n] != -1) return cache[n];
22         cache[n] = recursive(n-1) + recursive(n-2);
23         return cache[n];
24     }
25 }
```

动态规划算法源码

```
1 import java.util.Scanner;
2 import java.util.Scanner;
3
4 public class Main {
5     static int[] cache = new int[50];
6     static int n;
7     static Scanner sc = new Scanner(System.in);
8
9     public static void main(String[] args) {
10         n = sc.nextInt();
11         Arrays.fill(cache, -1);
12         cache[1] = 1;
13         cache[2] = 1;
14         cache[3] = 2;
15         cache[4] = 3;
16         cache[5] = 5;
17         cache[6] = 8;
18         cache[7] = 13;
19         cache[8] = 21;
20         cache[9] = 34;
21         cache[10] = 55;
22         cache[11] = 89;
23         cache[12] = 144;
24         cache[13] = 233;
25         cache[14] = 377;
26         cache[15] = 610;
27         cache[16] = 987;
28         cache[17] = 1597;
29         cache[18] = 2584;
30         cache[19] = 4181;
31         cache[20] = 6765;
32         cache[21] = 10946;
33         cache[22] = 17711;
34         cache[23] = 28657;
35         cache[24] = 46368;
36         cache[25] = 75025;
37         cache[26] = 121393;
38         cache[27] = 196418;
39         cache[28] = 317811;
40         cache[29] = 514229;
41         cache[30] = 832040;
42         cache[31] = 1346269;
43         cache[32] = 2178309;
44         cache[33] = 3524558;
45         cache[34] = 5702867;
46         cache[35] = 9227466;
47         cache[36] = 14930355;
48         cache[37] = 24157813;
49         cache[38] = 39088168;
50         cache[39] = 63246021;
51         cache[40] = 102334155;
52         cache[41] = 165580176;
53         cache[42] = 267914281;
54         cache[43] = 433494457;
55         cache[44] = 701408738;
56         cache[45] = 1134903195;
57         cache[46] = 1836311933;
58         cache[47] = 2970215128;
59         cache[48] = 4806527061;
60         cache[49] = 7776736189;
61         cache[50] = 12586864067;
62     }
63 }
```

Python算法源码

非递归算法（带缓存优化）

```
1 import sys
2 import sys
3
4 def main():
5     n = int(input())
6     cache = [0] * 50
7     cache[1] = 1
8     cache[2] = 1
9     cache[3] = 2
10    cache[4] = 3
11    cache[5] = 5
12    cache[6] = 8
13    cache[7] = 13
14    cache[8] = 21
15    cache[9] = 34
16    cache[10] = 55
17    cache[11] = 89
18    cache[12] = 144
19    cache[13] = 233
20    cache[14] = 377
21    cache[15] = 610
22    cache[16] = 987
23    cache[17] = 1597
24    cache[18] = 2584
25    cache[19] = 4181
26    cache[20] = 6765
27    cache[21] = 10946
28    cache[22] = 17711
29    cache[23] = 28657
30    cache[24] = 46368
31    cache[25] = 75025
32    cache[26] = 121393
33    cache[27] = 196418
34    cache[28] = 317811
35    cache[29] = 514229
36    cache[30] = 832040
37    cache[31] = 1346269
38    cache[32] = 2178309
39    cache[33] = 3524558
40    cache[34] = 5702867
41    cache[35] = 9227466
42    cache[36] = 14930355
43    cache[37] = 24157813
44    cache[38] = 39088168
45    cache[39] = 63246021
46    cache[40] = 102334155
47    cache[41] = 165580176
48    cache[42] = 267914281
49    cache[43] = 433494457
50    cache[44] = 701408738
51    cache[45] = 1134903195
52    cache[46] = 1836311933
53    cache[47] = 2970215128
54    cache[48] = 4806527061
55    cache[49] = 7776736189
56    cache[50] = 12586864067
57
58    print(cache[n])
59
60 if __name__ == '__main__':
61     main()
```

动态规划算法源码

```
1 import sys
2 import sys
3
4 def main():
5     n = int(input())
6     cache = [0] * 50
7     cache[1] = 1
8     cache[2] = 1
9     cache[3] = 2
10    cache[4] = 3
11    cache[5] = 5
12    cache[6] = 8
13    cache[7] = 13
14    cache[8] = 21
15    cache[9] = 34
16    cache[10] = 55
17    cache[11] = 89
18    cache[12] = 144
19    cache[13] = 233
20    cache[14] = 377
21    cache[15] = 610
22    cache[16] = 987
23    cache[17] = 1597
24    cache[18] = 2584
25    cache[19] = 4181
26    cache[20] = 6765
27    cache[21] = 10946
28    cache[22] = 17711
29    cache[23] = 28657
30    cache[24] = 46368
31    cache[25] = 75025
32    cache[26] = 121393
33    cache[27] = 196418
34    cache[28] = 317811
35    cache[29] = 514229
36    cache[30] = 832040
37    cache[31] = 1346269
38    cache[32] = 2178309
39    cache[33] = 3524558
40    cache[34] = 5702867
41    cache[35] = 9227466
42    cache[36] = 14930355
43    cache[37] = 24157813
44    cache[38] = 39088168
45    cache[39] = 63246021
46    cache[40] = 102334155
47    cache[41] = 165580176
48    cache[42] = 267914281
49    cache[43] = 433494457
50    cache[44] = 701408738
51    cache[45] = 1134903195
52    cache[46] = 1836311933
53    cache[47] = 2970215128
54    cache[48] = 4806527061
55    cache[49] = 7776736189
56    cache[50] = 12586864067
57
58    print(cache[n])
59
60 if __name__ == '__main__':
61     main()
```

Python算法源码

非递归算法（带缓存优化）

```
1 import sys
2 import sys
3
4 def main():
5     n = int(input())
6     cache = [0] * 50
7     cache[1] = 1
8     cache[2] = 1
9     cache[3] = 2
10    cache[4] = 3
11    cache[5] = 5
12    cache[6] = 8
13    cache[7] = 13
14    cache[8] = 21
15    cache[9] = 34
16    cache[10] = 55
17    cache[11] = 89
18    cache[12] = 144
19    cache[13] = 233
20    cache[14] = 377
21    cache[15] = 610
22    cache[16] = 987
23    cache[17] = 1597
24    cache[18] = 2584
25    cache[19] = 4181
26    cache[20] = 6765
27    cache[21] = 10946
28    cache[22] = 17711
29    cache[23] = 28657
30    cache[24] = 46368
31    cache[25] = 75025
32    cache[26] = 121393
33    cache[27] = 196418
34    cache[28] = 317811
35    cache[29] = 514229
36    cache[30] = 832040
37    cache[31] = 1346269
38    cache[32] = 2178309
39    cache[33] = 3524558
40    cache[34] = 5702867
41    cache[35] = 9227466
42    cache[36] = 14930355
43    cache[37] = 24157813
44    cache[38] = 39088168
45    cache[39] = 63246021
46    cache[40] = 102334155
47    cache[41] = 165580176
48    cache[42] = 267914281
49    cache[43] = 433494457
50    cache[44] = 701408738
51    cache[45] = 1134903195
52    cache[46] = 1836311933
53    cache[47] = 2970215128
54    cache[48] = 4806527061
55    cache[49] = 7776736189
56    cache[50] = 12586864067
57
58    print(cache[n])
59
60 if __name__ == '__main__':
61     main()
```

动态规划算法源码

```
1 import sys
2 import sys
3
4 def main():
5     n = int(input())
6     cache = [0] * 50
7     cache[1] = 1
8     cache[2] = 1
9     cache[3] = 2
10    cache[4] = 3
11    cache[5] = 5
12    cache[6] = 8
13    cache[7] = 13
14    cache[8] = 21
15    cache[9] = 34
16    cache[10] = 55
17    cache[11] = 89
18    cache[12] = 144
19    cache[13] = 233
20    cache[14] = 377
21    cache[15] = 610
22    cache[16] = 987
23    cache[17] = 1597
24    cache[18] = 2584
25    cache[19] = 4181
26    cache[20] = 6765
27    cache[21] = 10946
28    cache[22] = 17711
29    cache[23] = 28657
30    cache[24] = 46368
31    cache[25] = 75025
32    cache[26] = 121393
33    cache[27] = 196418
34    cache[28] = 317811
35    cache[29] = 514229
36    cache[30] = 832040
37    cache[31] = 1346269
38    cache[32] = 2178309
39    cache[33] = 3524558
40    cache[34] = 5702867
41    cache[35] = 9227466
42    cache[36] = 14930355
43    cache[37] = 24157813
44    cache[38] = 39088168
45    cache[39] = 63246021
46    cache[40] = 102334155
47    cache[41] = 165580176
48    cache[42] = 267914281
49    cache[43] = 433494457
50    cache[44] = 701408738
51    cache[45] = 1134903195
52    cache[46] = 1836311933
53    cache[47] = 2970215128
54    cache[48] = 4806527061
55    cache[49] = 7776736189
56    cache[50] = 12586864067
57
58    print(cache[n])
59
60 if __name__ == '__main__':
61     main()
```