

362

60



464. Can I Win

Notes

Description (/problems/can-i-win/description/)

Hints (/problems/can-i-win/hints/)

Submissions (/problems/can-i-win/submissions/)

Pick One (/problems/random-one-question/)

In the "100 game," two players take turns adding, to a running total, any integer from 1..10. The player who first causes the running total to reach or exceed 100 wins.

What if we change the game so that players cannot re-use integers?

For example, two players might take turns drawing from a common pool of numbers of 1..15 without replacement until they reach a total ≥ 100 .

Given an integer `maxChoosableInteger` and another integer `desiredTotal`, determine if the first player to move can force a win, assuming both players play optimally.

You can always assume that `maxChoosableInteger` will not be larger than 20 and `desiredTotal` will not be larger than 300.

Example

Input:

```
maxChoosableInteger = 10
desiredTotal = 11
```

Output:

```
false
```

Explanation:

No matter which integer the first player choose, the first player will lose.

The first player can choose an integer from 1 up to 10.

If the first player choose 1, the second player can only choose integers from 2 up to 10.

The second player will win by choosing 10 and get a total = 11, which is \geq desiredTotal.

Same with other integers chosen by the first player, the second player will always win.

Seen this question in a real interview before?



Companies ▲



LinkedIn 2 (/company/linkedin)

Related Topics ▲

[Dynamic Programming \(/tag/dynamic-programming/\)](/tag/dynamic-programming/)[Minimax \(/tag/minimax/\)](/tag/minimax/)

Similar Questions ▲

[Flip Game II \(/problems/flip-game-ii/\)](/problems/flip-game-ii/)[Guess Number Higher or Lower II \(/problems/guess-number-higher-or-lower-ii/\)](/problems/guess-number-higher-or-lower-ii/)[Predict the Winner \(/problems/predict-the-winner/\)](/problems/predict-the-winner/)

Java ▼



Notes

```
1 class Solution {
2     private Map<Integer, Boolean> cache;
3     private int maxChoosableInteger;
4
5     public boolean canIWin(int maxChoosableInteger, int desiredTotal) {
6         if ((1 + maxChoosableInteger) * maxChoosableInteger / 2 <
7         desiredTotal)
8             return false;
9
10        cache = new HashMap<>();
11        this.maxChoosableInteger = maxChoosableInteger;
12        return find(0, desiredTotal);
13    }
14
15    private boolean find(int taken, int desiredTotal) {
16        Boolean result = cache.get(taken);
17        if (result == null) {
18            result = Boolean.FALSE;
19            for (int i=0; i<maxChoosableInteger; i++) {
20                if ((taken & (1 << i)) != 0) {
21                    continue;
22                }
23                if (i + 1 >= desiredTotal || !find(taken | (1 << i),
24                desiredTotal - i - 1)) {
25                    result = true;
26                    break;
27                }
28                cache.put(taken, result);
29            }
30            return result;
31        }
```

☐ Custom Testcase ([Contribute](#))

Run

Submit

Copyright © 2018 LeetCode

[Contact Us \(/support/\)](/support/) | [Frequently Asked Questions \(/faq/\)](/faq/) | [Terms of Service \(/terms/\)](/terms/) | [Privacy Policy \(/privacy/\)](/privacy/) [United States \(/region/\)](/region/)