464. Can I Win

Add to List

QuestionEditorial Solution

My Submissions

- Total Accepted: **3323**
- Total Submissions: **15284**
- Difficulty: **Medium**
- Contributors: **taylorty**

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.

Player1 can win if and only if they can make a move after which player2 will lose no matter how they move. So we can check all possible moves of player1 and call the same function to see if player2 will lose after this move. This is the main idea. I use caching to memorize states for each possible desiredTotal and set of chosable integers. Since maxChoosableInteger is not greater than 20, all currently possible moves can be represented as a bit flag.

```
class Solution {
public:
    bool canWin(int key, int desiredTotal,vector<unordered_map<int,bool>> &cache, int
mx) {
        if(cache[desiredTotal-1].find(key) != cache[desiredTotal-1].end())
           return cache[desiredTotal-1][key];
        for(int i = mx-1; i >= 0;--i)
           if(key & (1 << i))
           {
               key ^= (1 << i);
               if(i+1 >= desiredTotal || !canWin(key,desiredTotal-i-1,cache,mx))
               {
                  cache[desiredTotal-1][key] = true;
                  key |= (1 << i);
                  return true;
               }
               key |= (1 << i);
           }
        cache[desiredTotal-1][key] = false;
        return false;
```

```
    }
    bool canIWin(int maxChoosableInteger, int desiredTotal) {
        if(desiredTotal <= 1)
          return true;
        if(maxChoosableInteger*(maxChoosableInteger+1) < desiredTotal*2)
           return false;
        vector<unordered_map<int,bool>> cache(desiredTotal);
        vector<bool> v(maxChoosableInteger,true);
        int key = (1 << maxChoosableInteger)-1;
        return canWin(key,desiredTotal,cache,maxChoosableInteger);
    }
};
```