<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
<th>G</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

You lose!
long fib_rec(int n) {
    if (n < 2) {
        return n;
    } else {
        return fib_rec(n - 1) + fib_rec(n - 2);
    }
}

dynamic programming: Store results of smaller problems

how many recursive calls to compute fib(n)?
for n = 0, 1 = 1
for n > 2, 1 + count(n-1) + count(n-2)
call to fib(n)
call to fib(n-1)
call to fib(n-2)
overlapping subproblems!
I win:

- 
- 
- 
- 
- 
- 
- 

you


take 1, 2, or 3 sticks
want to take last stick

to determine if state $s$ is a win for current player
  if $s$ is end of game then $s$ is loss
  else
    go over all moves available for $s$
    if any one is a loss for current player
      then state $s$ is a win for current player
    else $s$ is a losing position

n: 0 1 2 3 4 5 6 7 8 9 10 11 12
L w w w L w w w L w w w L ...
get list of all states

record 000 as W

for each state s in list
  get list of successor states
  if all successors W
    record s as L
  else
    record s as W

look up current state
if L

if W
  get list of successors
  find successor s that is L
  output WIN + s

make array of size sum of digits
for each column c going down
  for j = 0 to digit(p(c)) - 1
    max has digit(i) in col i & j <
    min has digit(i) in col i & j <
    min(j, digit(i)) for i < c

4 4 3 2 1

i = 1, result = 1
j = 2
$1$, $23$, $37$ & stamps (unlimited supply of each)

Make 50 & using fewest possible stamps $\frac{50 = 23 + 23 + 1 + 1 + 1}{6}$ stamps

In general: if $v_1, v_2, \ldots, v_k$ is shortest list with $v_i \in \{1, 23, 37\}$

and $\sum v_i = n$

then $v_1, \ldots, v_{k-1}$ is shortest list with sum $n - v_k$

optimal substructure

\[\text{num}(n) = 1 + \min (\text{num}(n-37), \text{num}(n-23), \text{num}(n-1))\]