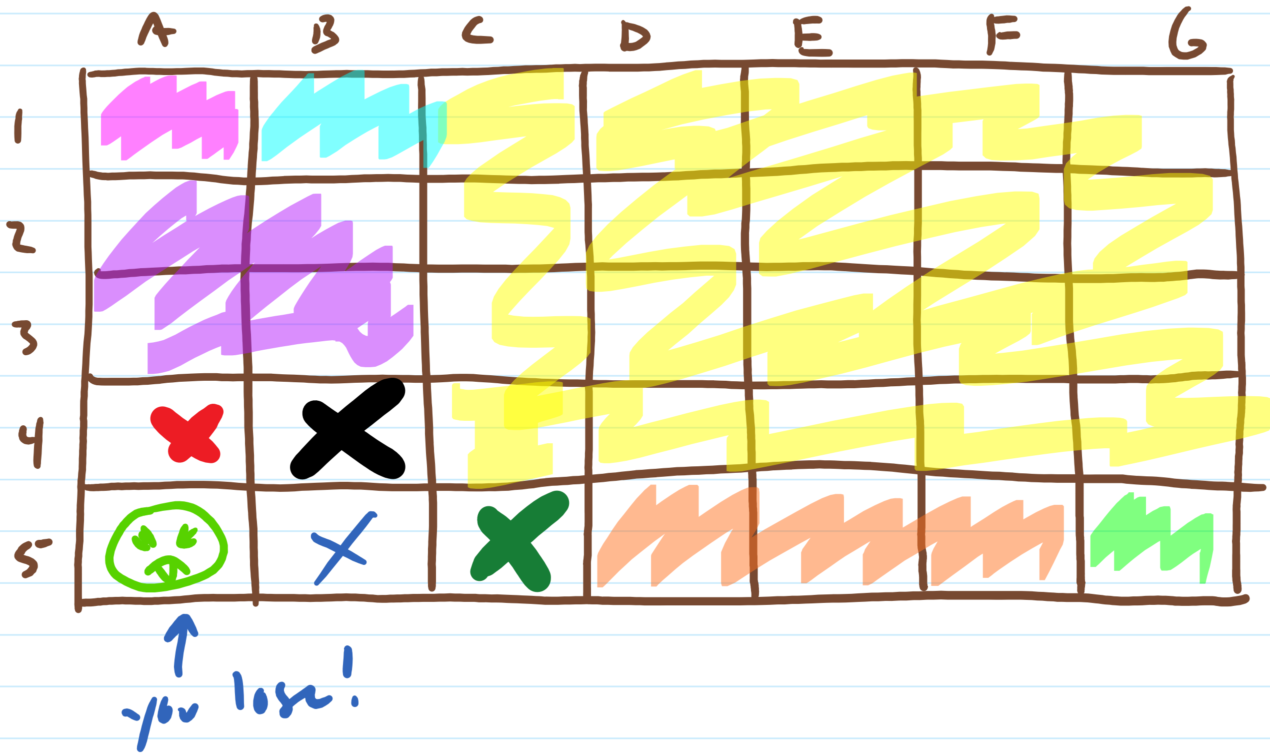
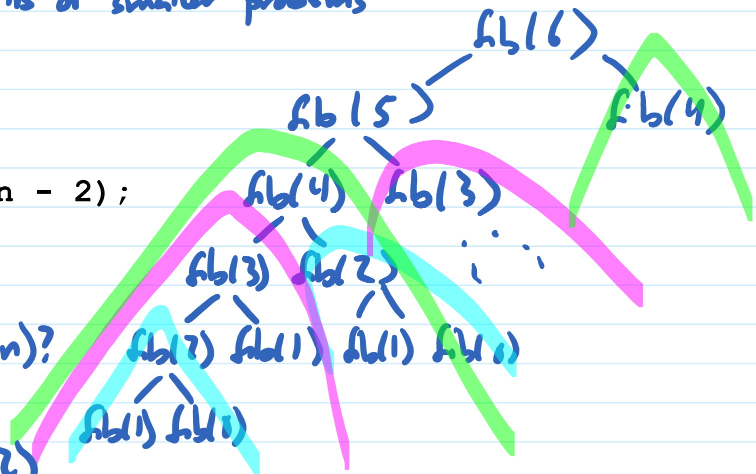

A
B
C
D
E
F
G
1
2
3
4
5
you lose!

0 1 1 2 3 5 8 13 21 34 55....

0+1 1+1 1+2 2+3

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

dynamic programming { work bottom-up; store results of smaller problems }



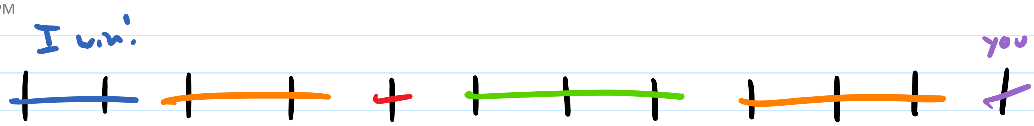
how many recursive calls to compute fib(n)?

count — f(n)

for n = 0, n = 1 : 1

for n ≥ 2 , $\underbrace{1}_{\text{call to fib(n)}} + \underbrace{\text{count(n-1)}}_{\text{call to fib(n-1) + recursive calls made below it}} + \text{count(n-2)}$

overlapping subproblems!

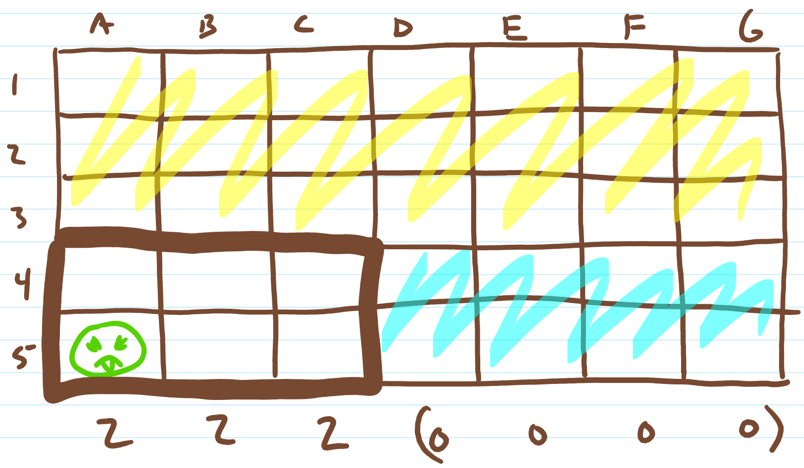


take 1, 2, or 3 sticks
want to take last stick

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	n= 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

chomp-states(2,3)

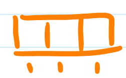
keys	value
000	W
100	L
110	W
111	W
200	W
210	L
211	
220	
221	
222	

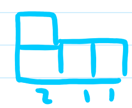
→ smap

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

chomp-successors

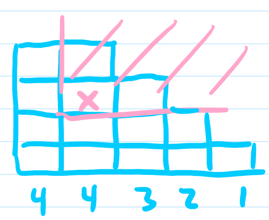




look up current state
if L ☹

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 →
for j = 0 to digit[c]-1
make has digit[i] in col i for i<c
j in col c
[min(j, digit[i]) for i>c]



c=1
j=2

result = 4 2 2 2 1

1¢, 23¢, 37¢ stamps (unlimited supply of each)

Make 50¢ using fewest possible stamps $50 = \underbrace{23 + 23 + 1 + 1 + 1 + 1}_{6 \text{ stamps}}$

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

$$num(n) = 1 + \min(num(n-37), num(n-23), num(n-1))$$