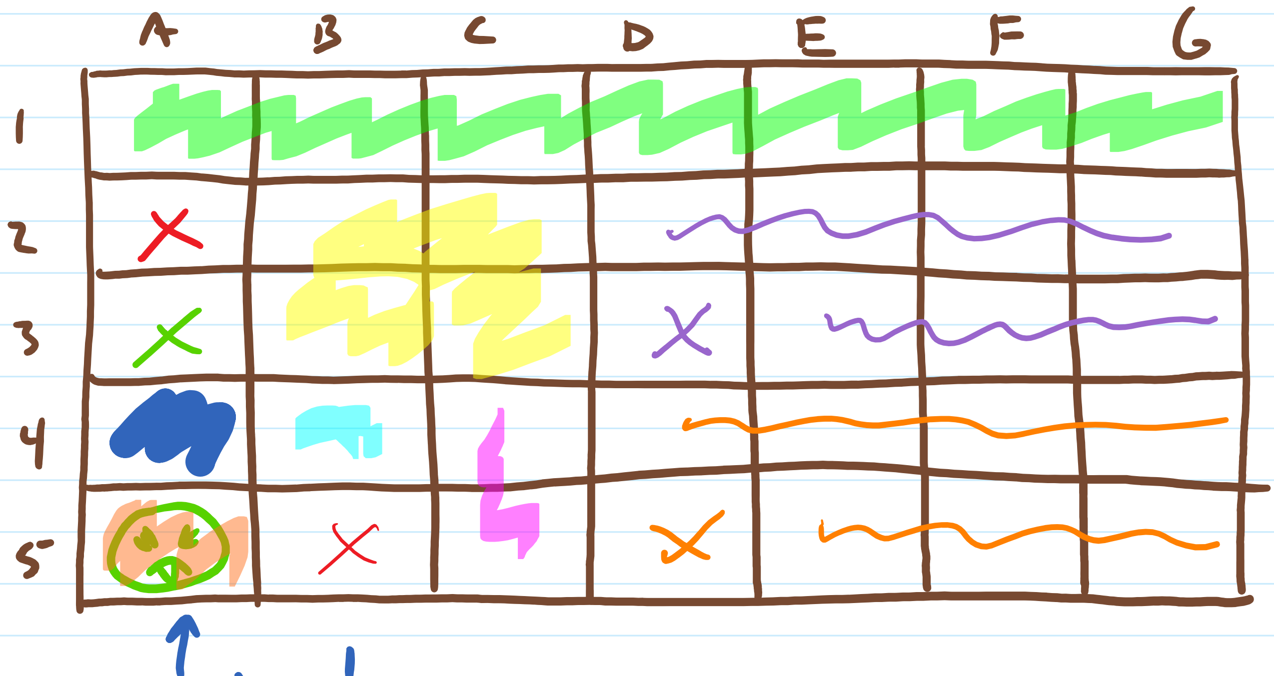
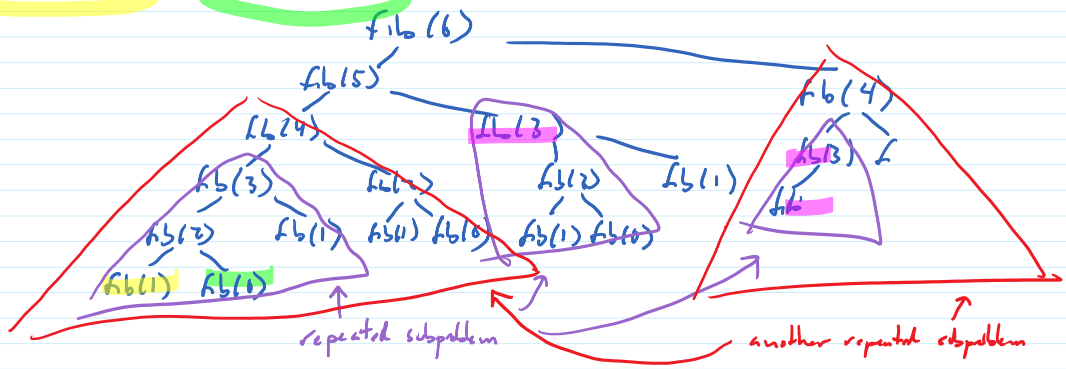


you lose!

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

(indices 0 1 2 3 4 5 6 7 8; 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);
    }
}
```



count(n)

How many recursive calls to compute fib(n)?

for n = 0 or n = 1   1 call   count(0) = count(1) = 1

n > 1   count(n) = 1 + count(n-1) + count(n-2)

n =	0	1	2	3	4	5	6	
	1	1	3	5	9	15	25	...

count(n) ≥ fib(n) → exponential

$$\frac{1}{\sqrt{5}}\left[\left(\frac{1+\sqrt{5}}{2}\right)^n - \left(\frac{1-\sqrt{5}}{2}\right)^n\right]$$

Zoo : $\approx 10^{-6}_{sec} \cdot \left(\frac{3}{2}\right)^n$ $n=60 \rightsquigarrow \sim 10$ hours

small constant

Human = $60_{sec} \cdot n$ $n=60 \rightsquigarrow \sim 1$ hour

large constant

low-order fxn w/ large constant
always eventually beats high-order fxn w/ low constant

Stamps

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

Make 50¢ using fewest possible stamps

$50 = 37 + \underbrace{1 + 1 + \cdots + 1}_{13 \text{ times}}$ 14 stamps

$= 23 + \underbrace{23 + 1 + 1 + 1 + 1}_{\text{best way to make 27¢ postage}}$ 6 stamps

In general: if $s_1, s_2, \ldots, s_k$ is shortest list with $s_i \in \underbrace{\{\overset{v_1}{1}, \overset{v_2}{23}, \overset{v_3}{37}\}}_{V}$ $k=3$

and $\sum s_i = n$

then $s_1, \ldots, s_{k-1}$ is shortest list with sum $n - s_k$

optimal substructure

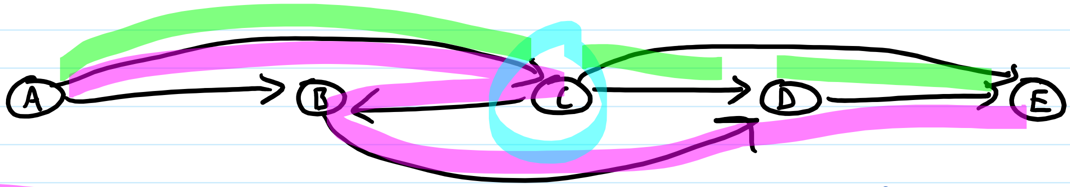
fewest(n) = fewest stamps to make n cents postage

$$= \begin{cases} 0 & \text{if } n = 0 \\ \min\limits_{\substack{v_i \in V \\ v_i \le n}} \left(1 + \text{fewest}(n - v_i)\right) \end{cases}$$

if opt soln for n uses v_i then
opt soln for n = 1 + opt soln for $n - v_i$ (opt substructure)

but don't know which v_i opt soln for n uses
↓
try each one and see which is best

Longest (simple) Path in a DAG



$v_1\ v_2 \dots v_k$

longest $A \rightsquigarrow E$: $ACBDE$

longest path $C \rightsquigarrow E$ (optimal substructure)

E D B C A

0 1 2 3 4

E: 1+0

D: 1+1

D 1+1
E 1+0
B 1+2

C: 1+3
B: 1+2

longest path $v_1 \rightsquigarrow v_k$ (from v_1, to v_k)

$= v_1 \to v_2 \; + $ longest path $v_2 \rightsquigarrow v_k$

don't know what v_2 is so try every possibility

$$\text{longest}(v_i) = \begin{cases} 0 & \text{if } v_i = \text{ending point} \\ \max\limits_{v_i \to v_j} & 1 + \text{longest}(v_j) \end{cases}$$

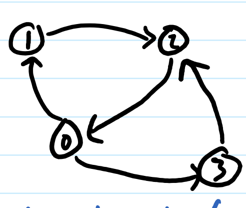
need to order vertices so that when working on v_i, already done with v_j for edges $v_i \to v_j$

topological sort!

```
longest[to] = 0
for each vertex v_i ≠ to in order of topo sort
    long = -∞
    for each edge v_i → v_j
        long ← max(long, 1 + longest[v_j])
    longest[v_i] ← long
```

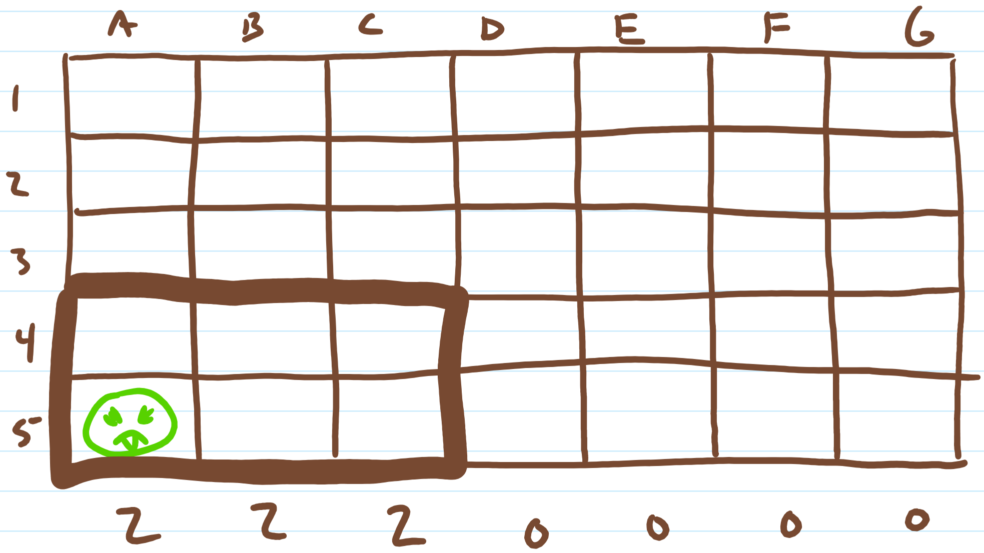
$O(n+m)$ with adj list

(for every vert v_i:
for every edge from v_i
⋮)



doesn't work if there are cycles

longest path $0 \rightsquigarrow 1$ $\neq$ longest of $[0 \to v_j + \text{longest } v_j \rightsquigarrow v_k]$ over all v_j



get list of all states

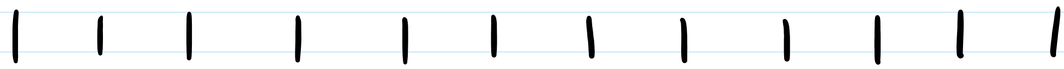
record 000 as W

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

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

← heights of columns

1-2-3 Nim



Take 1, 2, or 3 sticks
Last stick wins

win(n) = whether there is a winning strategy
for you if you start your turn with n sticks

=

n= 0 n=1 2 3 4 5 6 7 8 9 10 11 12
L