Recursion continued

Midterm Exam

- 2 parts

- Part 1 – done in recitation
  - Programming using server
  - Covers material done in Recitation
- Part 2 – Friday 8am to 4pm in CS110 lab
  - Question/Answer
  - Similar format to Inheritance quiz

Fibonacci's Rabbits

- Suppose a newly-born pair of rabbits, one male, one female, are put on an island.
  - A pair of rabbits doesn't breed until 2 month old.
  - Thereafter each pair produces another pair each month
  - Rabbits never die.
- How many pairs will there be after months?

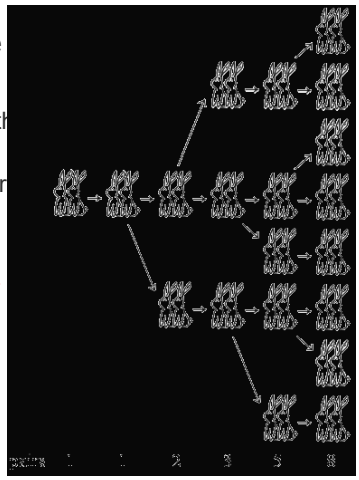


image from: http://www.jimloy.com/algebra/fibo.htm

3

Do some cases, see a pattern?

m0:	1 young		1
m1:	1 mature		1
m2:	1 mature	1 young	2
m3:	2 mature	1 young	3
m4:	3 mature	2 young	5
m5:	5 mature	3 young	8

m6?

The pattern...

m0: 1 young 1
m1: 1 mature 1
m2: 1 mature 1 young 2
m3: 2 mature 1 young 3
m4: 3 mature 2 young 5

m_n = m_{n-1} (rabbits never die) +
m_{n-2} (newborn pairs)

How fast does this rabbit population grow?

Fibonacci numbers

- The *Fibonacci numbers* are a sequence of numbers F_0, F_1, ... F_n defined by:

$$F_0 = F_1 = 1$$

$$F_i = F_{i-1} + F_{i-2} \text{ for any } i > 1$$

- Write a method that, when given an integer *i*, computes the *nth* Fibonacci number.

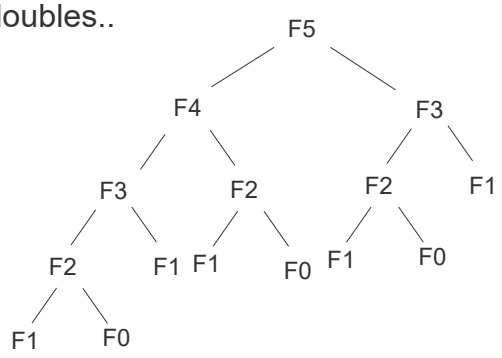
Fibonacci numbers

- recursive Fibonacci was expensive because it made many, many recursive calls
  - fibonacci(n) recomputed fibonacci(n-1, ... ,1) many times in finding its answer!
  - this is a common case of "overlapping subproblems", where the subtasks handled by the recursion are redundant with each other and get recomputed

7

Fibonacci code

- Let's run it for n = 1,2,3,... 10, ... , 20,...
- What happens if n = 5, 6, 7, 8, ...
- Every time n increments with 2, the call tree more than doubles..



Growth of rabbit population

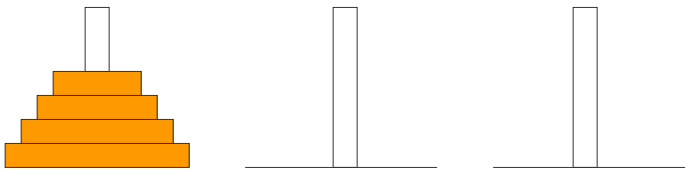
1 1 2 3 5 8 13 21 34 ...

every 2 months the population at least

DOUBLES

Recursive Algorithms

Example: Tower of Hanoi, move all disks to third peg without ever placing a larger disk on a smaller one.

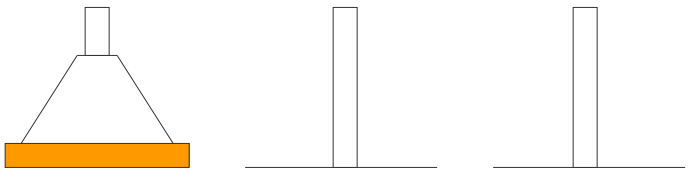


Try to find the pattern by cases

- One disk is easy
- Two disks...
- Three disks...
- Four disk...

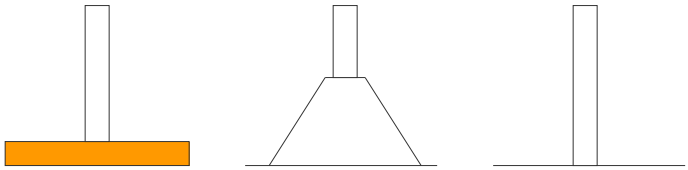
Recursive Algorithms

Example: Tower of Hanoi, move all disks to third peg without ever placing a larger disk on a smaller one.



Recursive Algorithms

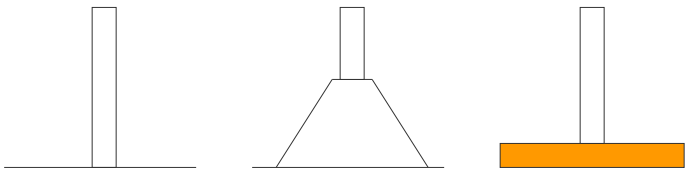
Example: Tower of Hanoi, move all disks to third peg without ever placing a larger disk on a smaller one.



13

Recursive Algorithms

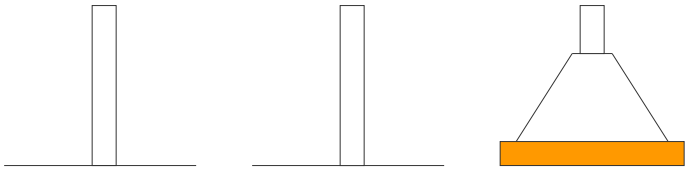
Example: Tower of Hanoi, move all disks to third peg without ever placing a larger disk on a smaller one.



14

Recursive Algorithms

Example: Tower of Hanoi, move all disks to third peg without ever placing a larger disk on a smaller one.



15

Parade

- A parade consists of a set of bands and floats in a single line.
- To keep from drowning each other out, bands cannot be placed next to another band
- Given the parade is of length n, how many ways can it be organized

Counting ways

- Let P(n) = the number of ways the parade can be organized.
- Parades can either end in a band or a float
- Let F(n) = the number of parades of length n ending in a float
- Let B(n) = the number of parades of length n ending in a band
- So:
  - P(n) = F(n) + B(n)

Recursive case

- Consider F(n)
  - Since a float can be placed at next to anything, the number of parades ending in a float is equal to
    - F(n) = P(n-1)
- Consider B(n)
  - The only way a band can end a parade is if the next to last unit is a float.
    - B(n) = F(n-1)
    - By substitution, B(n) = P(n-2)
- So:
  - P(n) = P(n-1) + P(n-2)

Base case

- How many parades configs can there be for:
  - n=1
  - 2 – float or band
- How many parade configs can there be for :
  - n=2
  - 3
    - Float/float
    - Band/float
    - Float/band

Dictionary lookup

- Suppose you're looking up a word in the dictionary (paper one, not online!)
- You probably won't scan linearly thru the pages – inefficient.
- What would be your strategy?

Binary search

```
binarySearch(dictionary,  word){

   if (dictionary has one page) {// base case
       scan the page for word
   }

   else {// recursive case

       open the dictionary to a point near the middle
       determine which half of the dictionary contains word

       if (word is in first half of the dictionary) {
            binarySearch(first half of dictionary, word)
       }
       else {
            binarySearch(second half of dictionary, word)
       }
   }
```

Binary search

- Write a method `binarySearch` that accepts a sorted array of integers and a target integer and returns the index of an occurrence of that value in the array.
  - If the target value is not found, return -1

index	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
value	-4	2	7	10	15	20	22	25	30	36	42	50	56	68	85	92	103

```
int index  = binarySearch(data, 42);  // 10
int index2 = binarySearch(data, 66);  // -1
```