Recursion continued

Recursion - examples

- Problem: given a string as input, write it backward
- Base case?
- Recursion

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

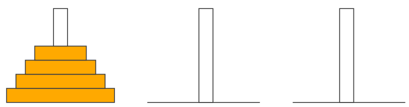
```
int index  = binarySearch(data, 42);  // 10
int index2 = binarySearch(data, 66);  // -1
```

Binary search

```
// Returns the index of an occurrence of the given
// value in the given array, or -1 if not found.
// Precondition: a is sorted
public int binarySearch(int[] a, int target) {
    return binarySearch(a, target, 0, a.length - 1);
}
// Recursive helper to implement search.
private int binarySearch(int[] a, int target,
                         int first, int last) {
    if (first > last) {
        return -1;   // not found
    } else {
        int mid = (first + last) / 2;
        if (a[mid] == target) {
            return mid;   // found it!
        } else if (a[mid] < target) {
            // middle element too small; search right half
            return binarySearch(a, target, mid+1, last);
        } else {  // a[mid] < target
            // middle element too large; search left half
            return binarySearch(a, target, first, mid-1);
        }
    }
}
```

Recursive Algorithms

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

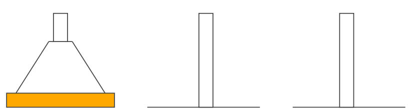


Try to find the pattern by cases

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

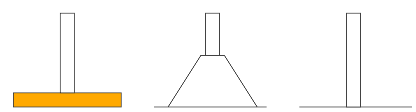
Recursive Algorithms

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



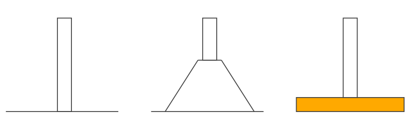
Recursive Algorithms

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



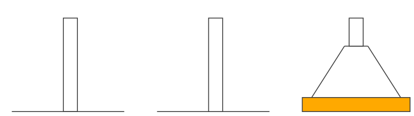
Recursive Algorithms

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.



Let's go play with it at: http://www.mazeworks.com/hanoi/index.htm
Or http://www.mathsisfun.com/games/towerofhanoi.html

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 months old.
  - Thereafter each pair produces another pair each month
  - Rabbits never die.
- How many pairs will there be after n months?

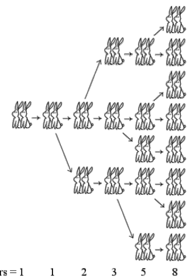


image from: http://www.jimloy.com/algebra/fibo.htm

13

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.

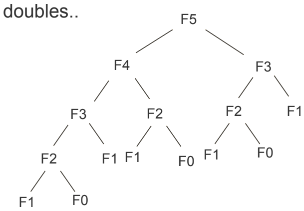
Fibonacci numbers

- 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 case, where the sub-tasks handled by the recursion are redundant with each other and get recomputed

15

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**

Fractals – the Koch curve