

Recursion continued

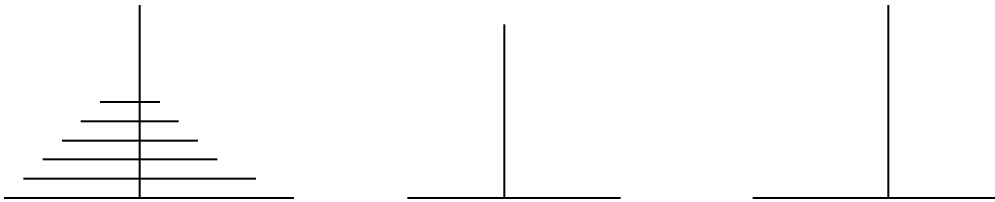
Feb 2

When to do Recursion

- When the recursive version does the same amount of work as the non-recursive, (same order of magnitude)
- When the recursive version is shorter or simpler than the non recursive one.

When there is a trade off between the iterative method vs a recursive one, usually the iterative one is slightly faster (lower constant, same order of magnitude), and the recursive one is typically simpler.

Example: Towers of Hanoi



Problem: Transfer all the rings from A to C, using B as a temporary storage. We can only move one ring at a time, and no ring can sit on a smaller ring.

Think recursively: How do we move 1 ring? Easy.

2 rings? Trivial

3 rings? Move two rings to the temporary ring, move the third to C, move the rest on C.

4 rings? Move 3 rings to the temp ring, move the fourth to C, move the three onto C.

This is by nature, a recursive problem.

Code that prints the motions:

```
hanoi(4, "A", "C", "B")
```

```
static void hanoi (int n, String from, String to, String Using) {  
    if (n == 1) {  
        System.out.println(from + "->" + to);  
    } else {  
        hanoi(n-1, from, using, to);  
        System.out.println(from + "->" + to);  
        hanoi(n-1, using, to, from);  
    }  
}
```

This is an exponential problem. Assuming each move takes 1 second:

64 rings → 584 billion years.

Monks thought that the world would end once the monks finished the problem.

Linear Data Structures, Stacks and Queues

Stacks

- A stack is a data structure in which elements are added and removed from one end only.
- A stack is last in first out (LIFO).

E.g. a stack of plates, stack of paper.

Stack Specifications

Stack()

- constructor, initializes the stack to the empty state.
- preconditions: none
- post conditions: the stack is empty

empty()

- boolean function that returns true if the stack is empty

push(Object x)

- push item x onto the stack
- precondition: none
- post condition, x is at the top of the stack

pop()

- removes the top item off the stack and returns it. If the stack is empty, it returns null, (or perhaps it crashes, as another interpretation)
- returns an Object. In order to get back the original type, a cast is required.

peek()

- returns the top item of the stack without removing it.
- Returns Object

Example:

```
Rectangle r = new Rectangle(3,4,3,4);
Stack stack = new Stack();
...
...
stack.push(r);
r = (Rectangle) stack.pop();
```

[See stack example ☺](#)