NPTEL MOOC,JAN-FEB 2015
Week 1, Module 8

DESIGN AND ANALYSIS OF ALGORITHMS

MADHAVAN MUKUND, CHENNAI MATHEMATICAL INSTITUTE
http://www.cmi.ac.in/~madhavan

Calculating complexity

- Iterative programs
- Recursive programs

Example 1

- Maximum value in an array

```
function maxElement(A):
   maxval = A[0]
for i = 1 to n-1:
   if A[i] > maxval:
      maxval = A[i]
return(maxval)
```

Example 2

- Check if all elements in an array are distinct

```
function noDuplicates(A):
   for i = 0 to n-1:
      for j = i+1 to n-1:
         if A[i] == A[j]:
            return(False)
return(True)
```

Example 3

- Matrix multiplication

```
function matrixMultiply(A,B):
   for i = 0 to n-1:
     for j = 0 to n-1:
       C[i][j] = 0
       for k = 0 to n-1:
         C[i][j] = C[i][j] + A[i][k]*B[k][j]
return(C)
```

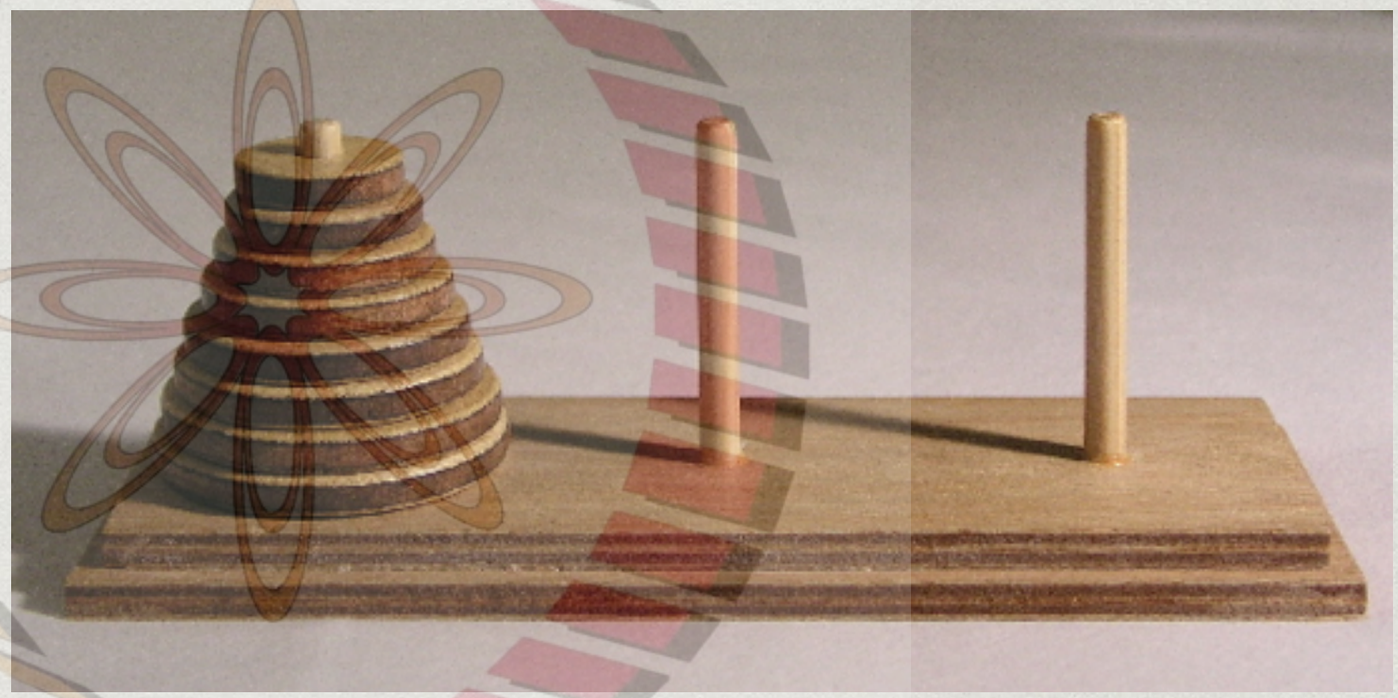
Example 4

- Number of bits in binary representation of n

```
function numberOfBits(n):
   count = 1
   while n > 1:
      count = count + 1
      n = n div 2
   return(count)
```

Example 5

Towers of Hanoi



- Three pegs, A, B, C
- Move n disks from A to B
- Never put a larger disk above a smaller one
- C is transit peg

Example 5

Recursive solution

- Move n-1 disks from A to C, using B as transit peg
- Move largest disk from A to B
- Move n-1 disks from C to B, using A as transit peg

Example 5

Solve recurrence by repeated substitution

- M(n) = number of moves to transfer n disks
  - M(n) = M(n-1) + 1 + M(n-1)
  - M(1) = 1
- Recurrence
  - Recursive expression for M(n)

Example 5

Complexity

- $M(n) = 2M(n-1) + 1$

  $= 2(2M(n-2)+1) + 1 = 2^2M(n-2) + (2+1)$

  $= 2^2(2M(n-3)+1) + 2 + 1 = 2^3M(n-3) + (4+2+1)$

  $= \ldots$

  $= 2^kM(n-k) + (2^k - 1)$

  $= \ldots$

  $= 2^{n-1}M(1) + (2^{n-1} - 1)$

  $= 2^{n-1} + 2^{n-1} - 1 =$

  $= 2^n - 1$

Summary

- Iterative programs
  - Focus on loops
- Recursive programs
  - Write and solve a recurrence
- Will see more complicated examples
  - Need to be clear about “accounting” for basic operations