CS314 - Induction, Recurrences + Recursion

Note Title 8/28/2013

Announcements

- HW due Wed. at start of class
- next HW out Wed, due the following Friday

For any induction proof:

4 pieces: - know what you are inducting on

- base case
- Inductive Hypothesis
- Inductive Step

Ex: The Gossip Problem

- There are n people, & each knows a unique secret.
- Every time 2 people call each other, they tell all of the secrets that they know to each other.

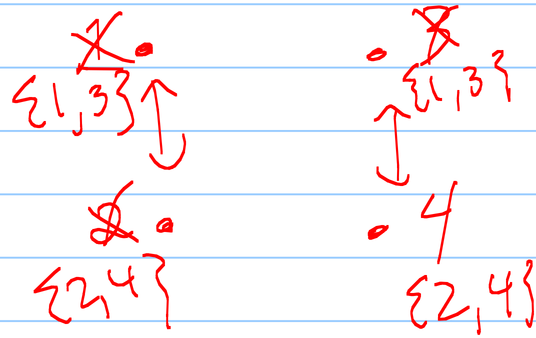
How many phone calls are necessary before everyone knows all the secrets?

Thm: If $n \geq 4$, then $2n-4$ calls are enough.

Proof: induction on # of people

Base case: 4 people

goal!: $2 \cdot 4 - 4 = 4$ calls



1 & 3 talk
2 & 4 talk
1 & 2 talk
3 & 4 talk

IH: For $k < n$ people, $2k-4$ calls sufice.

IS: n people

n calls 1

take person n away

n $\cdot$ $n-1$

$2(n-1)-4$

$= 2n-6$

n calls 1

$\blacksquare$

Now, recursion:

Induction starts at bottom & builds up.

Recursion is the natural dual idea:

- Start with n things
- Reduce to smaller subproblem(s)
- Eventually stop at some small <u>base case</u>

Solving recurrences

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

$$M(n) = 2M\left(\frac{n}{2}\right) + n$$

$$T(n) = T\left(\frac{3n}{4}\right) + n$$

How to solve?

- Unrolling
- Master theorem : $S(n) = aS\left(\frac{n}{b}\right) + f(n)$
- Guess & check
- characteristic eqn method (Annihilator method)

Recursion

Based on the idea of reduction:
reducing X to Y means giving an algorithm to solve X which may use Y as a subroutine.

Ex: (from last class)

the Congress partitioning
↳ used priority queue

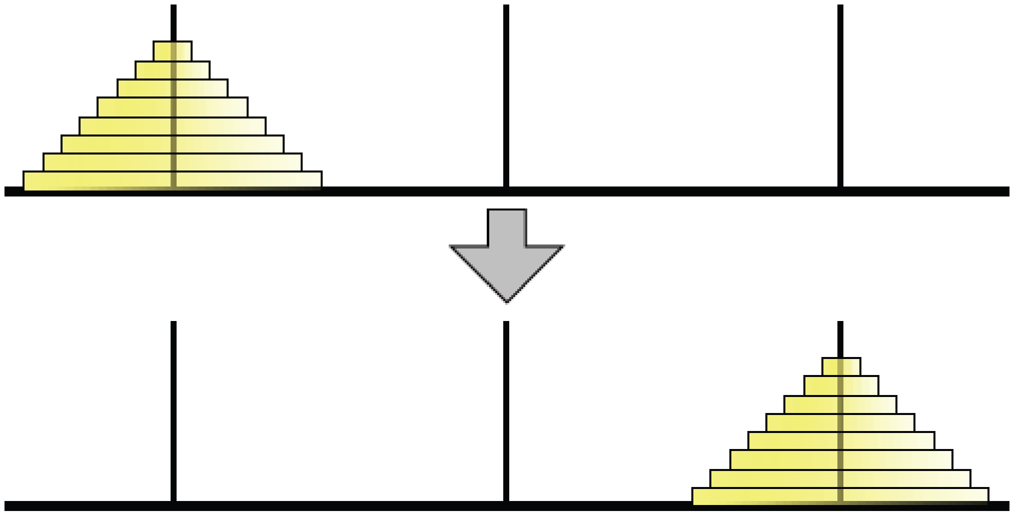
Recursion (cont)

Reduce a problem to a simpler instace of the same problem.

Necessary pieces (like induction):

- base case
- recursive call (to smaller input)
- (some extra work)

Towers of Hanoi

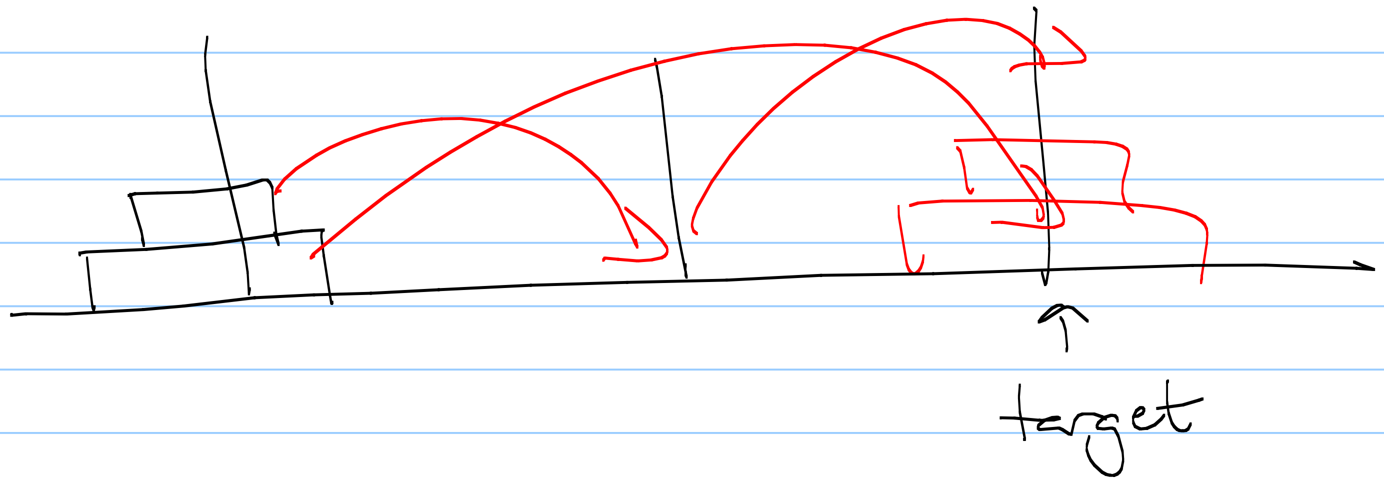


The Tower of Hanoi puzzle

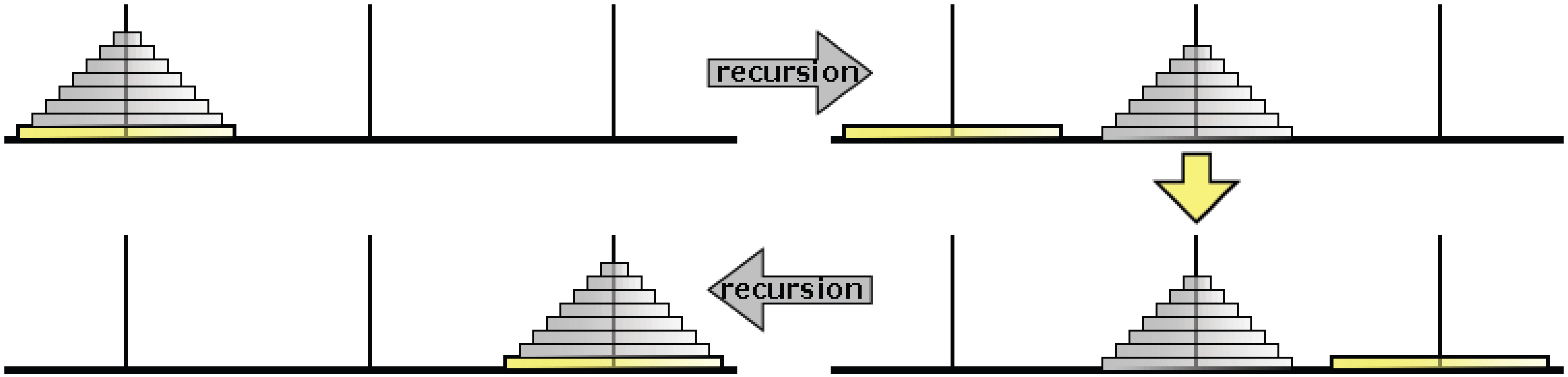
Rules:

- no
- move 1 at a time

How?



Think recursively!



Base case?

The trick: think recursively!

Most people try to unroll the recurrence — not necessary!

Think of this like a procedure:

<u>HANOI(n, src, dst, tmp):</u>
  if $n > 0$
    HANOI($n-1, src, tmp, dst$)
    move disk n **from** src **to** dst
    HANOI($n-1, tmp, dst, src$)

Proof of correctness

induction on # of disks, n:

Base case   n=1   recursive calls do nothing + 1 disk moves correctly

IH: For k<n disks, alg. is correct.

IS: n disks.
by IH, n-1 top move to temp
b/c one goes to dest,
by IH, n-1 top move correctly to dest,
(don't violate rules in top level) ☺

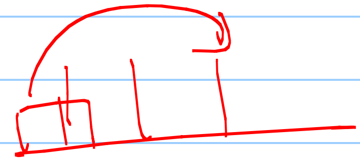
moves

Run time: Let $H(k)$ = time to solve towers of Hanoi w/ k pancakes

$H(0) = 0$

$H(1) = 1$

$H(2) = 3$



$H(n) = H(n-1) + 1 + H(n-1)$

$H(n) = 2H(n-1) + 1 \iff a_n = 2a_{n-1} + 1$

$x - 2 = 0$

root $= 2$

$f(n) = 1$ degree $= 0$

$H(n) = c_1 \cdot 2^n + c_2$

$= 2^n - 1$ (?)

Another (old) example: Merge Sort

According to Knuth, suggested by von Neumann around 1945.

Idea: 1) Subdivide array into 2 parts.
2) Recursively sort the 2 parts.
3) Merge them back together.

Input:	S	O	R	T	I	N	G	E	X	A	M	P	L
Divide:	S	O	R	T	I	N	G	E	X	A	M	P	L
Recurse:	I	N	O	S	R	T	A	E	G	L	M	P	X
Merge:	A	E	G	I	L	M	N	O	P	S	R	T	X

Key: If thinking recursively only ~~step~~ 3 is non-trivial!

```
MergeSort(A[1..n]):
  if (n > 1)
      m ← ⌊n/2⌋
      MergeSort(A[1..m])
      MergeSort(A[m+1..n])
      Merge(A[1..n], m)
```

(Again, avoid unrolling.)

What's my base case here?

size 1 (or less)

How to merge?

Input:	S	O	R	T	I	N	G	E	X	A	M	P	L
Divide:	S	O	R	T	I	N	G	E	X	A	M	P	L
Recurse:	I	N	O	S	R	T	A	E	G	L	M	P	X
Merge:	A	E	G	I	L	M	N	O	P	S	R	T	X

Write a subroutine:

```
MERGE(A[1..n], m):
  i ← 1;  j ← m + 1
  for k ← 1 to n
      if j > n
          B[k] ← A[i];  i ← i + 1
      else if i > m
          B[k] ← A[j];  j ← j + 1
      else if A[i] < A[j]
          B[k] ← A[i];  i ← i + 1
      else
          B[k] ← A[j];  j ← j + 1

  for k ← 1 to n
      A[k] ← B[k]
```

Proof of correctness: Actually, 2 of them.

Lemma: MERGE results in sorted order.

pf: induction on sizes of $A[i..m]$ and $A[j..n]$

Runtime :