Transitions: DS + Algorithms

Recursive algorithms

First:

Questions on older reading?

Plus worksheet 9

c & d:

```
for i ← 1 to n
    total ← 0                    ← O(1)
    for j ← 1 to i
        total ← total + A[j]     ← i times
    B[i] ← total                 ← O(1)
Return B
```

inserting into list or sorting

$$\sum_{i=1}^{n} (1 + i + 1)$$

$$= \sum_{i=1}^{n} 1 + \sum_{i=1}^{n} i + \sum_{i=1}^{n} 1$$

$$\sum_{j=1}^{i} 1 = i$$

c) $\sum_{i=1}^{n} \sum_{j=1}^{i} 1 = \Theta(n^2)$

d) Runtime $\geq$

$\sum_{i=1}^{n} (1 + i + 1)$

negligible in $O(\cdot)$

$= n + \Theta(n^2) + n$

$= \Theta(n^2)$

e) n input $\hookrightarrow n^2$ time

$2n \hookrightarrow (2n)^2 = 4n^2$

Hypothetical: Spps sorting

$\sum_{i=1}^{n} (n \log n + i) = \sum_{i=1}^{n} n \log n + \sum_{i=1}^{n} i$

$= n^2 \log n + n^2$

Multiplication: Can we do better?

Claim:

$$x \cdot y = \begin{cases} 0 & \text{if } x = 0 \\ \lfloor x/2 \rfloor \cdot (y+y) & \text{if } x \text{ is even} \\ \lfloor x/2 \rfloor \cdot (y+y) + y & \text{if } x \text{ is odd} \end{cases}$$

$x' \quad y'$

$7 \cdot 10 = 20$

$= \lfloor \frac{7}{2} \rfloor \cdot (10+10) + 10$

$= 3 \cdot (20) + 10$

$= 70$

Why? Proof by cases:

if x is even: $\exists k$ s.t. $x = 2k$

Then $x \cdot y = 2k \cdot y$

$= k \cdot 2y = k(y+y) = \lfloor \frac{x}{2} \rfloor (y+y)$ ✓

$7 \cdot 11 \Rightarrow 3 \cdot (11+11) + 11$

$66 + 11 = 77$

if x is odd:

$\exists \ell$ s.t. $x = 2\ell + 1$

$x \cdot y = (2\ell+1) \cdot y = 2\ell \cdot y + y = \ell \cdot 2y + y$

$= \ell(y+y) + y = \lfloor \frac{x}{2} \rfloor (y+y) + y$ ∎

How to make an algorithm?

Note: historical name! Not mine---

```
PeasantMultiply(x, y):
  if x = 0
    return 0
  else
    x' ← ⌊x/2⌋
    y' ← y + y
    prod ← PeasantMultiply(x', y')      ⟨⟨Recurse!⟩⟩
    if x is odd
      prod ← prod + y
    return prod
```

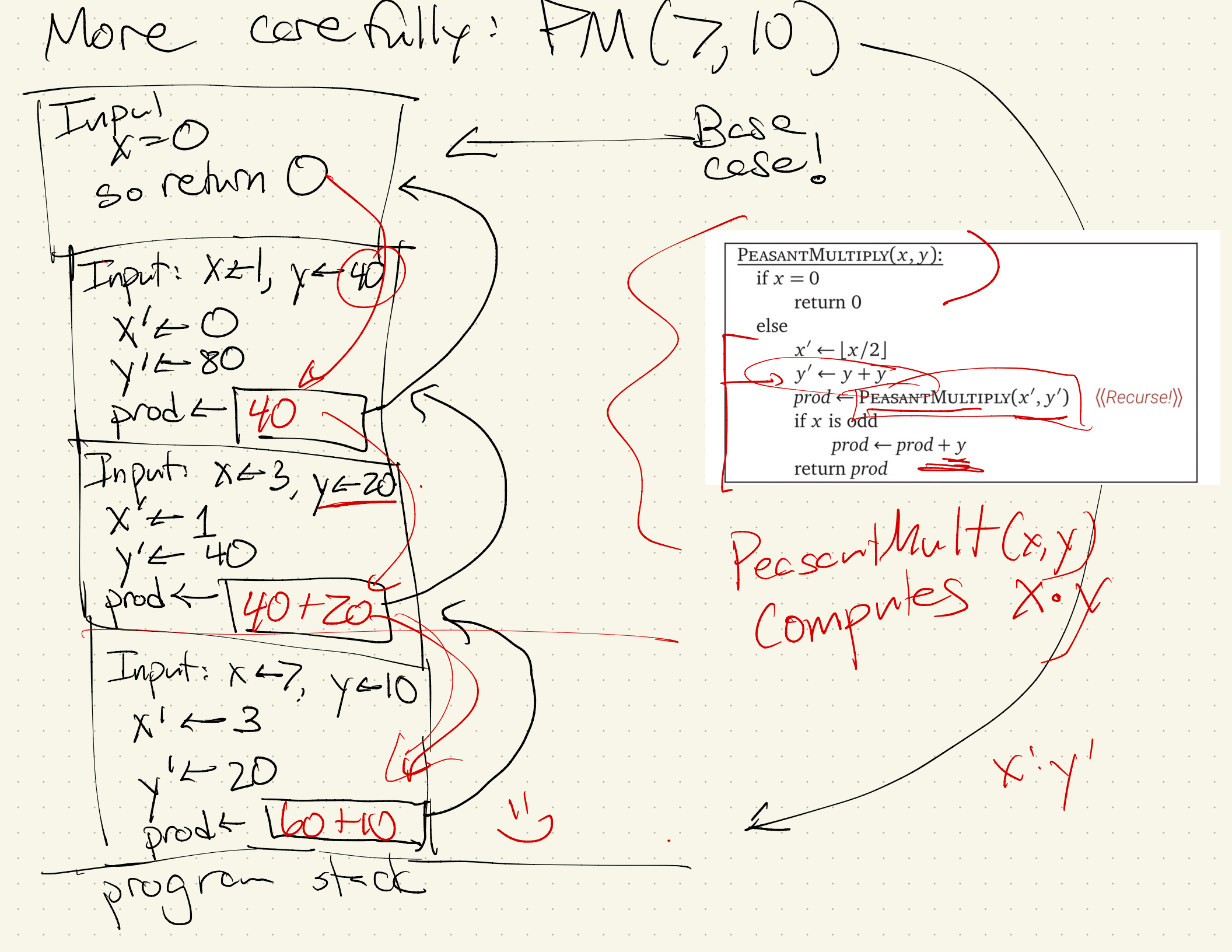
Suppose we call PM(7, 10):

x=7, y=10 → x' = ⌊7/2⌋ = 3, y' = 10+10 = 20

prod = 70 = 70 x' y'

x=3, x'=1, y=20, y'=40, prod = 60 = 60 x' y'

x=1, x'=0, y=40, y'=80, prod = 40


More carefully: PM(7, 10)
Input
x=0
so return 0
Base case!
Input: x←1, y←40
x'←0
y'←80
prod←40
Input: x←3, y←20
x'←1
y'←40
prod←40+20
Input: x←7, y←10
x'←3
y'←20
prod←60+10
program stack
PeasantMultiply(x, y):
if x = 0
return 0
else
x' ← ⌊x/2⌋
y' ← y + y
prod ← PeasantMultiply(x', y')
⟨⟨Recurse!⟩⟩
if x is odd
prod ← prod + y
return prod
PeasantMult(x,y)
Computes x·y
x·y

$k<n \quad P(k) \rightarrow P(n)$

Correctness

Induction on x, for <u>any</u> y:

Base case: $x=0$

$x \cdot y = 0$ and alg correctly returns this value.

IH: For values $x' < x$, the algorithm returns $x' \cdot y$ "(for any y")

IS: Consider $x \cdot y$, $x > 0$: NTS alg returns $x \cdot y$ correctly

If x is even: $x = 2k$ algorithm computes correctly (by IH)

$x'y' = \lfloor \frac{x}{2} \rfloor \cdot (y+y)$

$= k(y+y)$

and $\lfloor \frac{x}{2} \rfloor \cdot (y+y) = k(y+y) = k(2y) = x \cdot y$

If odd: $2k+1 = x$ algorithm computes $x' \cdot y'$ correctly by IH

$= \lfloor \frac{x}{2} \rfloor \cdot (y+y)$

After rec call to compute $x'y'$

recall: $x = 2\ell+1$

$= \lfloor \frac{x}{2} \rfloor (y+y)$

↳ x is odd, so I add $+y$

Why is this correct?

$x' \cdot y' + y = \lfloor \frac{2\ell+1}{2} \rfloor \cdot y' + y$

$= \ell \cdot (y+y) + y$

$= \ell(2y) + y$

$= 2\ell \cdot y + y = y(2\ell+1)$

$= y \cdot x = x \cdot y$ ∎

How can we calculate runtime?

Let $T(n)$ = runtime of this alg on x of size n

```
PEASANTMULTIPLY(x, y):
  if x = 0
    return 0
  else
    x' ← ⌊x/2⌋        ← one division
    y' ← y + y        ← one addition
    prod ← PEASANTMULTIPLY(x', y')    ⟨⟨Recurse!⟩⟩
    if x is odd
      prod ← prod + y
    return prod
```

$T(n) = 1 + 1 + 1 + T(\text{whatever size } x' \text{ is}) + 1 + \cancel{1} \; n \log r$

$= 5 + T(x' \text{ size})$

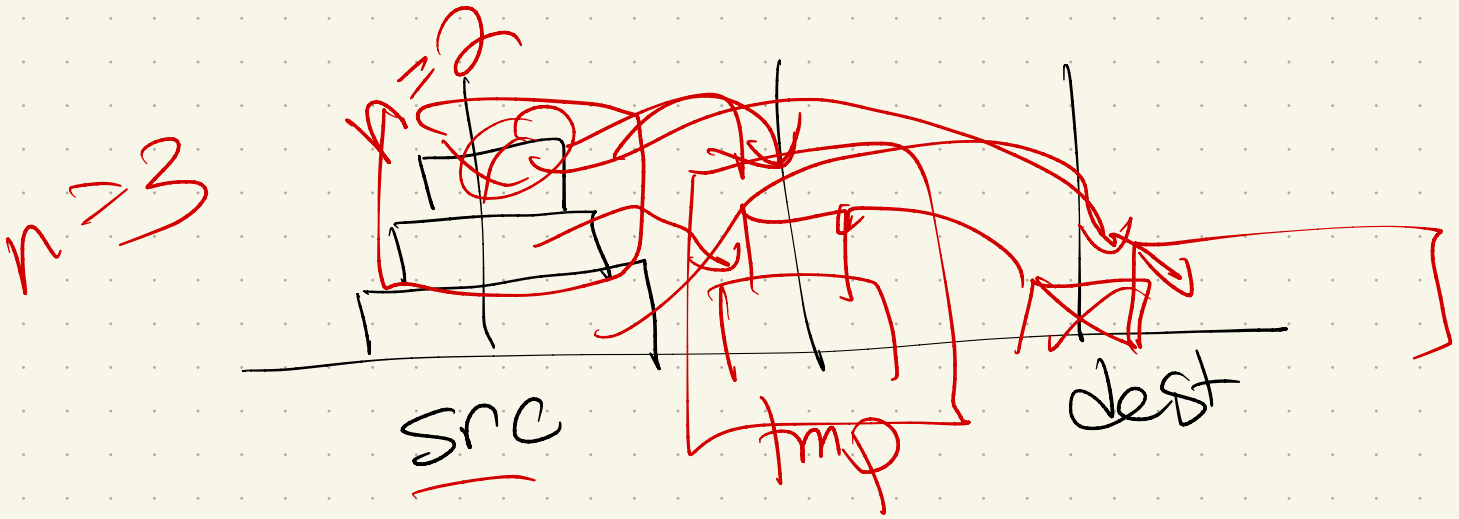
rewrite: $a_n = 5 + a_{n/2} < n$ not easy!

Towers of Hanoi:

```
HANOI(n, src, dst, tmp):
  if n > 0
      HANOI(n − 1, src, tmp, dst)     ⟨⟨Recurse!⟩⟩
      move disk n from src to dst
      HANOI(n − 1, tmp, dst, src)     ⟨⟨Recurse!⟩⟩
```

Figure 1.4. A recursive algorithm to solve the Tower of Hanoi

What is this:



Claim: Algorithm works

Proof of Correctness.
(induction on #disks)

Base case:

IH:

IS:

```
HANOI(n, src, dst, tmp):
  if n > 0
      HANOI(n − 1, src, tmp, dst)    ⟨⟨Recurse!⟩⟩
      move disk n from src to dst
      HANOI(n − 1, tmp, dst, src)    ⟨⟨Recurse!⟩⟩
```

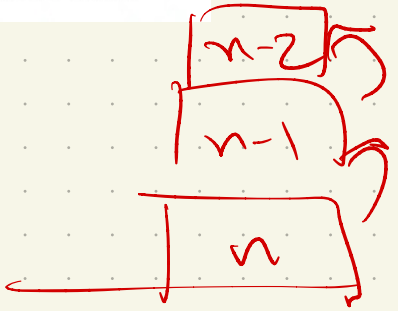
Figure 1.4. A recursive algorithm to solve the Tower of Hanoi

Runtime?

Count # times the disk moves!

```
HANOI(n, src, dst, tmp):
  if n > 0
    HANOI(n − 1, src, tmp, dst)    ⟨⟨Recurse!⟩⟩
    move disk n from src to dst
    HANOI(n − 1, tmp, dst, src)    ⟨⟨Recurse!⟩⟩
```

Figure 1.4. A recursive algorithm to solve the Tower of Hanoi



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

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

Let a_k = # of moves a call to Hanoi$(k, \cdot, \cdot)$ takes

$$a_n = 2a_{n-1} + 1$$

$$= 2(2a_{n-2} + 1) + 1$$

$$= 2(2(2a_{n-3} + 1) + 1) + 1$$

$$= 2(2(\cdots(a_0$$

$$\approx 2^n$$

Solving recurrences (& then MergeSort)