Transitions: DS + Algorithms

→ Sorting

Recap

- Reading: on Friday

  Also: oops on last reading!

  ↳ Please finish by tomorrow (Thurs)

- Make sure to upload worksheets

  ↳ Today's?

$$\sum_{i=1}^{n}\left(\sum_{j=1}^{2i} 1\right) = \sum_{i=1}^{n} 2i \quad \left(2\cdot 1 + 2\cdot 2 \cdots + 2\cdot n\right)$$

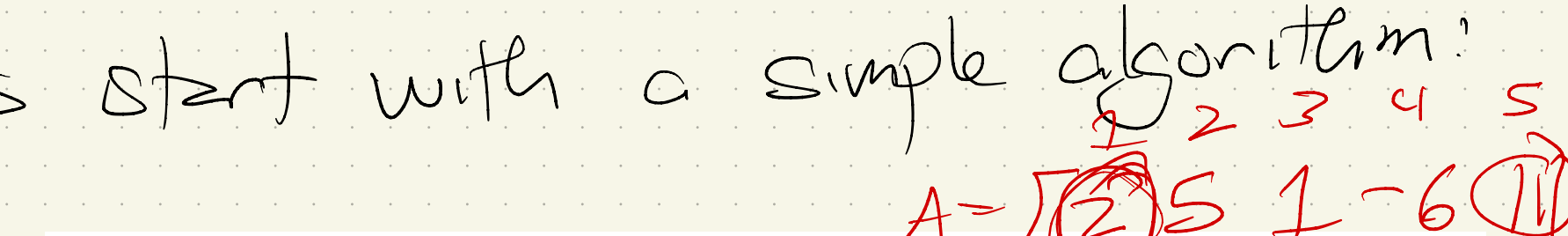
$$= 2\left(1 + 2 + \cdots + n\right)$$

$$= 2\cdot\left(\sum_{i=1}^{n} i\right) = 2\cdot\frac{n(n+1)}{2}$$

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

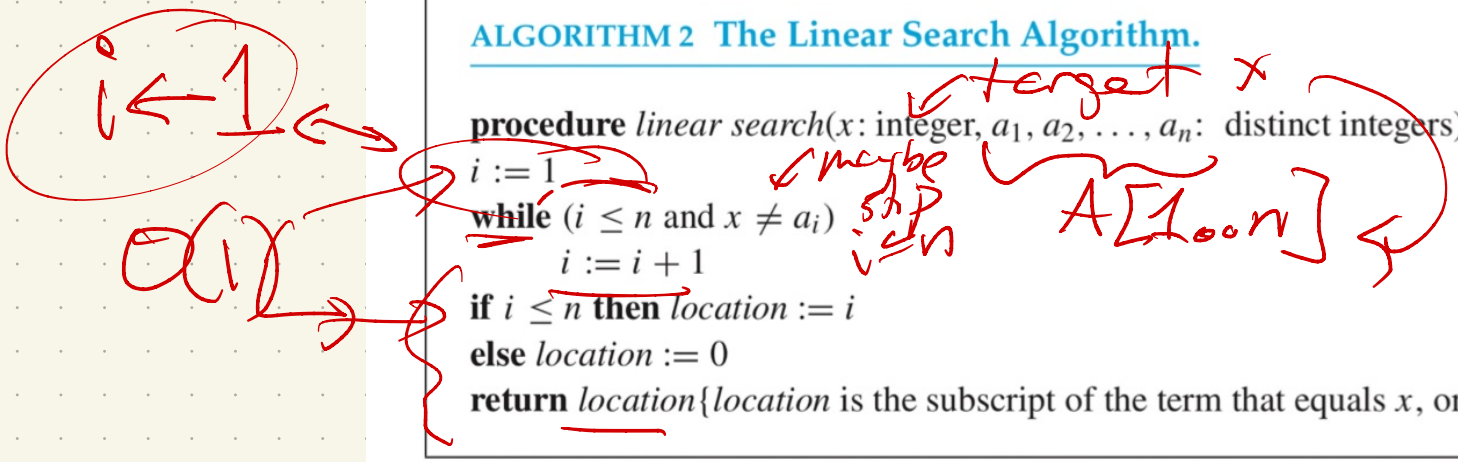
Today: An application of Big-O + "pseudocode"

Let's start with a simple algorithm:



ALGORITHM 2 The Linear Search Algorithm.

procedure *linear search*(x: integer, $a_1, a_2, \ldots, a_n$: distinct integers)
$i := 1$
while ($i \leq n$ and $x \neq a_i$)
  $i := i + 1$
if $i \leq n$ **then** *location* $:= i$
else *location* $:= 0$
return *location* {*location* is the subscript of the term that equals x, or is 0 if x is not found}



Runtime: Let $T(n)$ = time linear search takes on a list of size n

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

$T(n) = \Omega(n)$ via worst-case

$T(n) = \Theta(n)$

big-O: $f(n)$ is $O(g(n))$ if

$\exists c, n_0$ s.t. $\forall n > n_0$

$f(n) \leq c \cdot g(n)$

big-Omega Ω:

$f(n)$ is $\Omega(g(n))$ if

$\exists c', n_0'$ s.t. $\forall n > n_0'$

$f(n) \geq c' \cdot g(n)$

both: $\Theta(g(n))$ "Theta of $g(n)$"

Pseudocode format:

Varies by book! So we'll see a few.

In a pinch, pretend you're in Python or Ruby.

→ high level & readable.

I realize this is not a "definition" - that is the point!

It's about effective communication.

So: examples & practice!

Insertion sort:

```
for (i = 0; i < n; i++) {
    j = i;
    while ((j > 0) && (s[j] < s[j - 1])) {
        swap(&s[j], &s[j - 1]);
        j = j-1;
    }
}
```

i = i+1

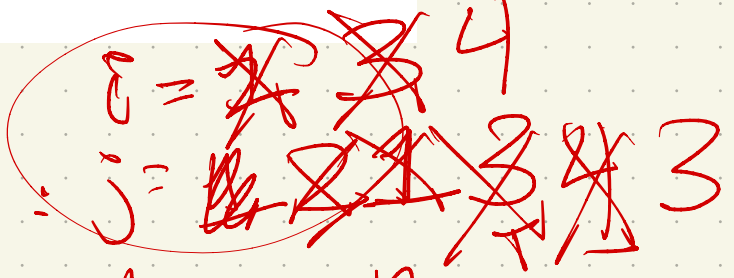
ick

Translate to pseudocode:

input S[1..n]

for i ← 2 to n-1
  j ← i
  while (j > 1) and S[j] < S[j-1]
    [swap S[j] ↓ S[j-1]
     j ← j-1]

Let's try

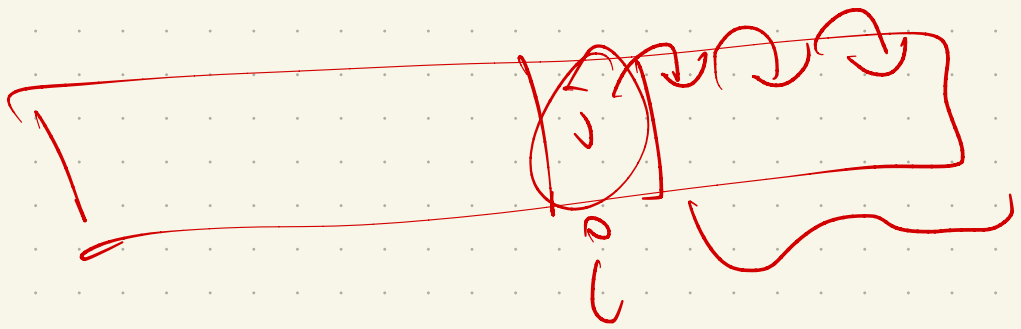


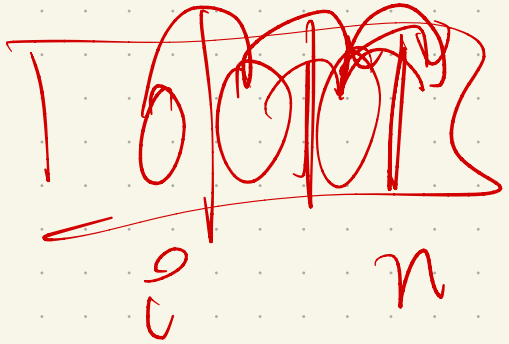
S[j] = value stored at spot j in list/array S

Runtime? T(n) = time it takes on list of size n

Reverse

sorted





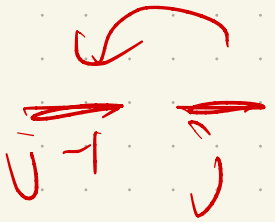
→ for $i \leftarrow 2$ to $n-1$

$j \leftarrow i$

while $(j > 1)$ and $(S[j] < S[j-1])$

swap $S[j] \leftrightarrow S[j-1]$

$j \leftarrow j-1$



$$T(n) = \sum_{i=2}^{n-1} \left(1 + \sum_{j=1}^{i} 5\right)$$

$5 + 5 + \cdots + 5$ (i times)

$(2, 3, \ldots, n-1)$ → $n-2$ terms

$$= \sum_{i=2}^{n-1} 1 + \sum_{i=2}^{n} \left(\sum_{j=1}^{i} 5\right)$$

$$= (n-2) + \sum_{i=2}^{n} 5i = n-2+5\sum_{i=2}^{n} i = \Theta(n) \rightarrow O(n^2)$$

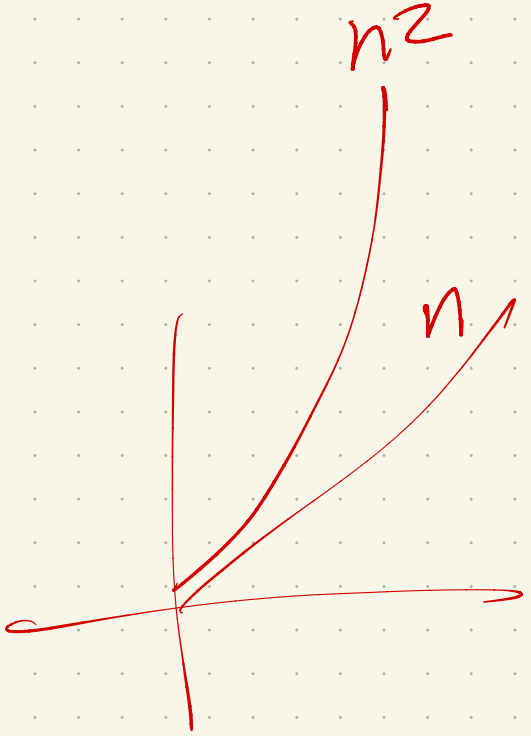
$$5\sum_{i=2}^{n} i = 5\left(\frac{n\cdot(n+1)}{2} - 1\right)$$

$$= \frac{5}{2}n^2 + \frac{5}{2}n - 5$$

$$= O(n^2)$$

$$T(n) \leq O(n) + O(n^2)$$

$$= O(n^2)$$



Bubblesort

B(n):

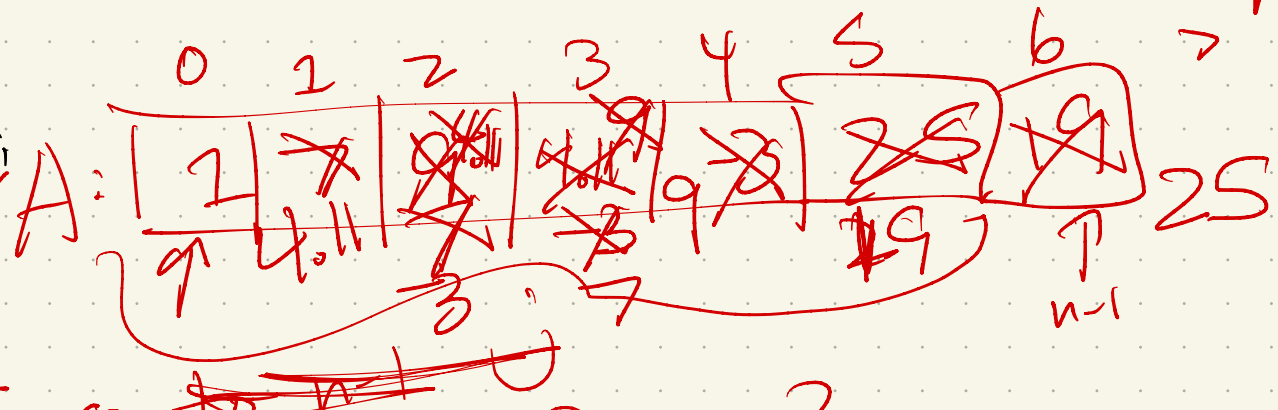
```
procedure bubbleSort(A : list of sortable items)
    n := length(A)
    for i := 0 to n - 1 do
        for j := 0 to n - i - 2 do
            if A[j] > A[j + 1] then
                // Swap the elements
                temp := A[j]
                A[j] := A[j + 1]
                A[j + 1] := temp
            end if
        end for
    end for
end procedure
```

swap A[j] & A[j+1]

$$B(n) = \sum_{i=0}^{n-1} \sum_{j=0}^{n-i-2} 5$$

$$= \sum_{i=0}^{n-1} 5(n-i-1)$$

Demo:



$i=0$ to $n-0-2 = n-2$

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

Runtime:

$i=0$ to $n-3$

$i=2$, j 0 to $n-4$

$$B(n) = \sum_{i=0}^{n-1} \sum_{j=0}^{n-i-2} 5$$

```
procedure bubbleSort(A : list of sortable items)
    n := length(A)
    for i := 0 to n - 1 do
        for j := 0 to n - i - 2 do
            if A[j] > A[j + 1] then
                // Swap the elements
                temp := A[j]
                A[j] := A[j + 1]
                A[j + 1] := temp
            end if
        end for
    end for
end procedure
```

$$B(n) = \sum_{i=0}^{n-1} \left[\underbrace{5 + 5 + \cdots + 5}_{j=0 \quad j=1 \quad \cdots \quad j \geq n-i-2} \right]$$

$$= \sum_{i=0}^{n-1} 5(n-i-1) = 5 \sum_{i=0}^{n-1} (n-i-1)$$

$$= 5\big((n-1) + (n-2) + \cdots + 0\big)$$

($i=0$, $i=1$, $\ldots$, $i=n-1$)

$$B(n) = 5\left((n-1) + (n-2) + \cdots + 0\right)$$

$$= 5\left(0 + 1 + \cdots + (n-2) + (n-1)\right)$$

$$= 5\sum_{i=1}^{n-1} i = 5\,\frac{(n-1)(n)}{2}$$

identity: $\sum_{i=1}^{k} i = \frac{k(k+1)}{2}$

$$= \frac{5}{2}n^2 - \frac{5}{2}n$$

$$= O(n^2)$$

■

- Bubble
- Insertion
- Selection

} $O(n^2)$

- Merge Sort $O(n \log n)$
- Quick sort $\leftarrow O(n^2)$ but in practice $O(n \log n)$

} recursive

Bucket $\leftarrow O(n k)$ → # of digits

Heap: $n \log n$

Next reading: Recursion

Consider multiplication: is it really O(1)?

Input: 2 numbers
↳ in decimal

$X[0..m-1]$, $Y[0..n-1]$

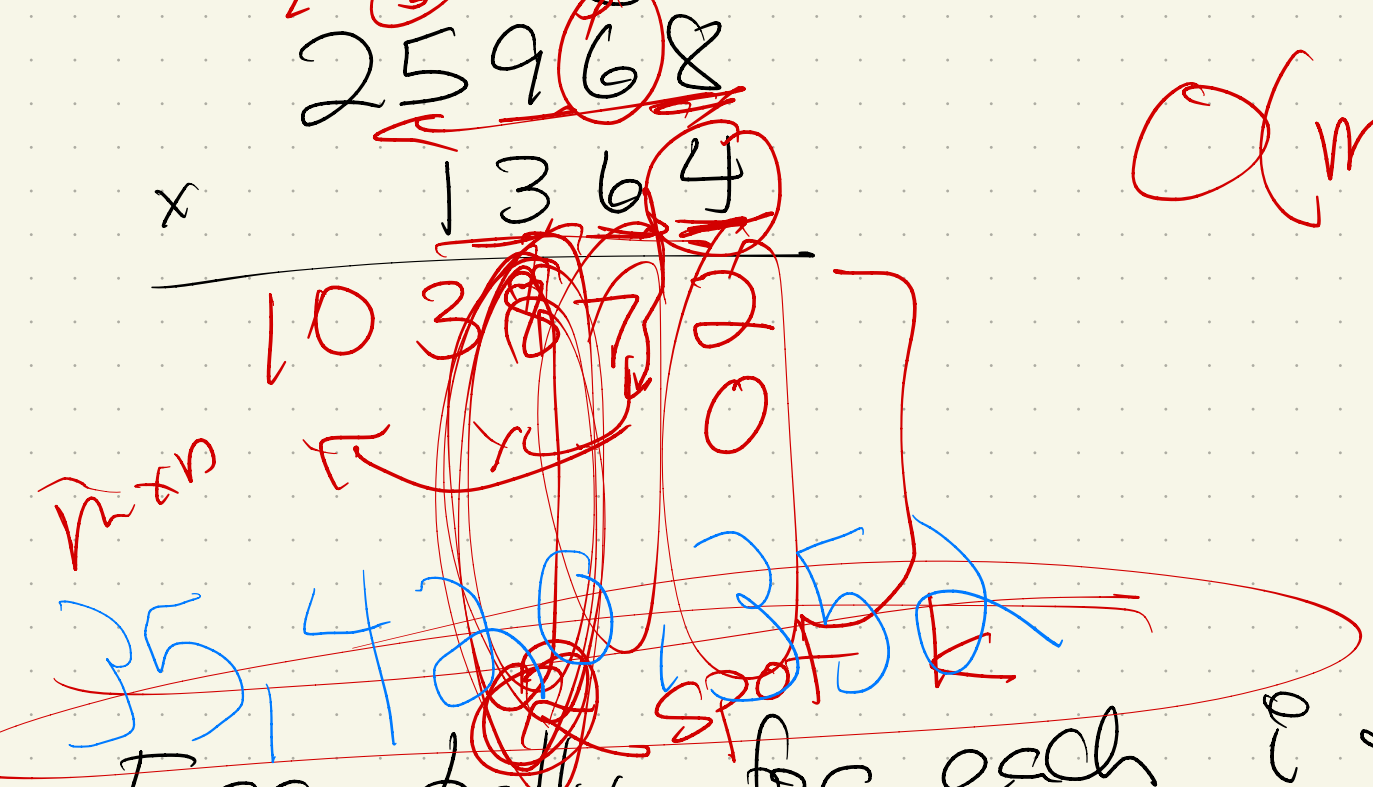
$X = \sum_{i=0}^{m-1} X[i] \cdot 10^i$, $Y = \sum_{j=0}^{n-1} Y[j] \cdot 10^j$

Ex: $x = 25968 = 8 \cdot 10^0 + 6 \cdot 10^1 + 9 \cdot 10^2 + 5 \cdot 10^3 + 2 \cdot 10^4$

and ~~X~~ $= [8, 6, 9, 5, 2]$
$0 \quad 1 \quad 2 \quad 3 \quad 4$

1364

Back to grade school:

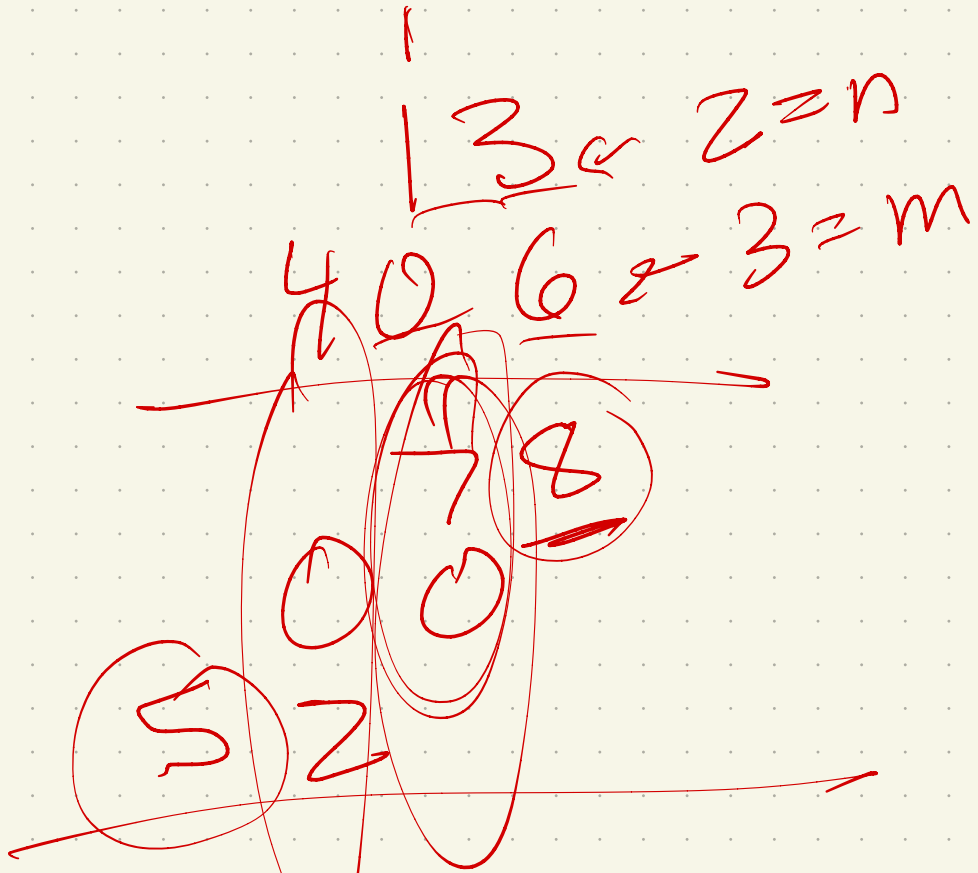


$O(m \cdot n)$

Essentially, for each i & j,

need $X[i] \cdot Y[j] \cdot 10^i \cdot 10^j$

$$\Rightarrow X \cdot Y = \sum_{i=1}^{n} \sum_{j=1}^{m} X[i] \cdot Y[j] \cdot 10^{i+j}$$



$$13 < 2 = n$$

$$40 \; 6 < 3 = m$$

$$X = [3, 1]$$

$$Y = [6, 0, 4]$$

$$0,\underset{10^3}{5}\;\underset{10^2}{2}\;\underset{10^1}{7}\;\underset{10^0}{8}$$

$$m+n-1 = 4$$

$$Z = [8, 7, 2, 5, 0]$$

$$10^{m+n-1}$$

Another view: Instead of just adding all, search for all that land in one spot k:

```
FIBONACCIMULTIPLY(X[0..m−1], Y[0..n−1]):
  hold ← 0
  for k ← 0 to n + m − 1
      for all i and j such that i + j = k
          hold ← hold + X[i] · Y[j]
      Z[k] ← hold mod 10
      hold ← ⌊hold/10⌋
  return Z[0..m + n − 1]
```

X: m digits Y: n digits

k all digits in product

product of X & Y

What is k?

$10^{i+j} = 10^k$

$i + j = k$

$0 + k$

$1 + k-1$

$2 + k-2$

$\vdots$

$k + 0$

$\sum_{k=0}^{n+m-1} (k)$

$= O((n+m-1)^2)$??

Runtime?

Different look:

```
FIBONACCIMULTIPLY(X[0..m-1], Y[0..n-1]):
  hold ← 0
  for k ← 0 to n + m - 1
      for all i and j such that i + j = k
          hold ← hold + X[i] · Y[j]
      Z[k] ← hold mod 10
      hold ← ⌊hold/10⌋
  return Z[0..m + n - 1]
```

Every pair of digits $X[i] \cdot Y[j]$ will be multiplied once & adding to hold

↳ $\Theta(mn)$ multiplications

Can we do better?

Claim:

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

Why? Proof by cases:

if x is even:

if x is odd:

How to make an algorithm?

```
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
```