Algorithms, Spring '25

Recursion

Recap

Recursive Algorithms : Chapter 1

1st half:

Setup, plus (hopefully) familiar examples:

- Towers of Hanoi
- Merge sort

2nd half:

- Recap of recurrences & "Master theorem"
- Linear time selection
- Multiplication (again)
- Exponentiation

More complex: recursion!

Algorithm 1 Quicksort

```
1: procedure QUICKSORT(A, p, r)
2:     if p < r then
3:         q = PARTITION(A, p, r)
4:         QUICKSORT(A, p, q − 1)
5:         QUICKSORT(A, q + 1, r)
6:     end if
7: end procedure
8: procedure PARTITION(A, p, r)
9:     x = A[r]
10:    i = p − 1
11:    for j = p to r − 1 do
12:        if A[j] < x then
13:            i = i + 1
14:            exchange A[i] with A[j]
15:        end if
16:        exchange A[i] with A[r]
17:    end for
18: end procedure
```

QuickSort Pseudocode Example

A high level note on recursion:

Recursion really can be simpler & useful!

Often depends upon the language and setup.

Counter-intuitive, but that's mostly because you haven't had a lot of practice.

Functional languages;

Recursion

- If you can solve directly (usually because input is small), do it!
- Otherwise, reduce to simple (usually smaller) instances of the same problem.

Result:

Recursion Fairy

- Helps to solidify that "black box" mentality, so you don't keep unpacking the next level.

(She's also called the "induction hypothesis".)

Multiplication: How?

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

Note: historical name! Not a commentary...

```
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
```

Runtime :

Towers of Hanoi: runtime

```
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

How ?? (no loop, + calls itself!)

Proof of correctness:

Runtime (for 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