CSCI 3100: Algorithms

Today:

Now: recursion

- Induction started at the bottom & builds up.

Recursion: the natural dual idea:

Recurrence relations:

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

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

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

How to solve?

Master Thm:

Let $T(n) = \begin{cases} aT\left(\frac{n}{b}\right) + n^c & \text{if } n > 1 \\ d & \text{if } n = 1 \end{cases}$

for constants $a, b, c, \& d$

Then:
- if $\log_b a < c$, $T(n) = \Theta(f(n))$
- if $\log_b a = c$, $T(n) = \Theta(n^c \log n)$
- if $\log_b a > c$, $T(n) = \Theta(n^{\log_b a})$

There are other versions:

The Master Theorem. *The recurrence* $T(n) = aT(n/b) + f(n)$ *can be solved as follows.*

- *If* $a\,f(n/b) = \kappa\,f(n)$ *for some constant* $\kappa < 1$*, then* $T(n) = \Theta(f(n))$.
- *If* $a\,f(n/b) = K\,f(n)$ *for some constant* $K > 1$*, then* $T(n) = \Theta(n^{\log_b a})$.
- *If* $a\,f(n/b) = f(n)$*, then* $T(n) = \Theta(f(n)\log_b n)$.
- *If none of these three cases apply, you're on your own.*

Really, saying the same thing!

(We'll talk about the proof later...)

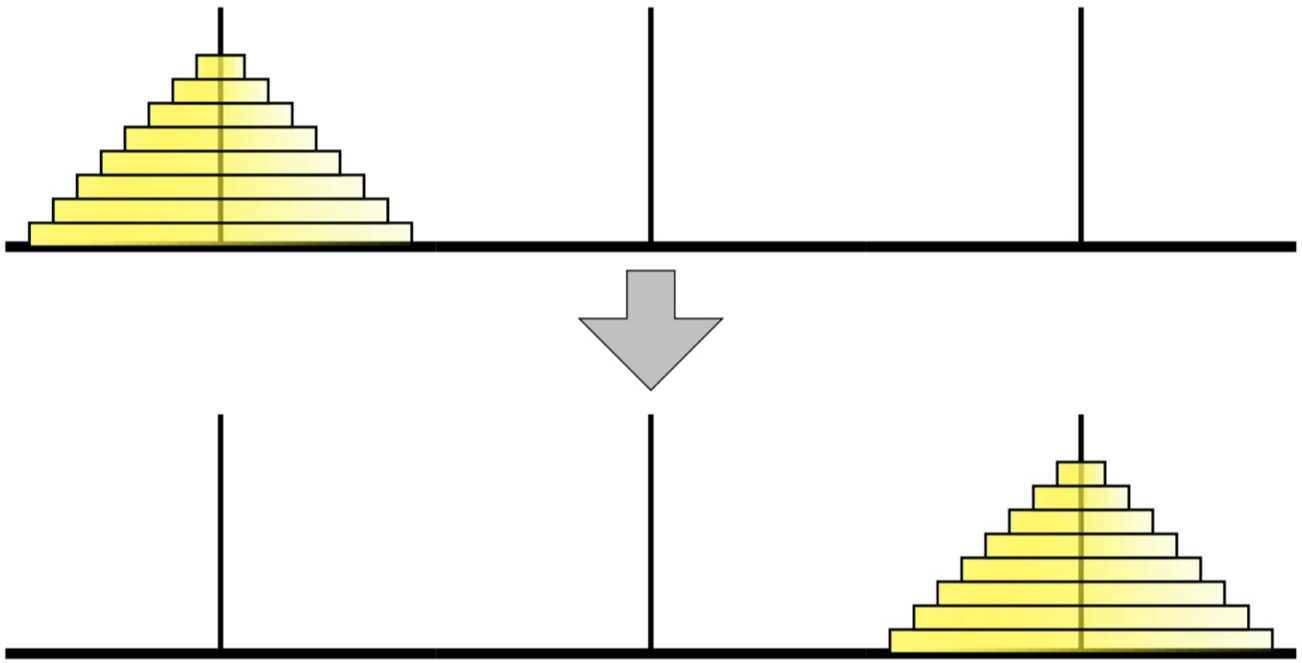
Recursive algorithms:

Based on reduction:

Reduce to a smaller instance of the same problem.

Necessary pieces (like induction):

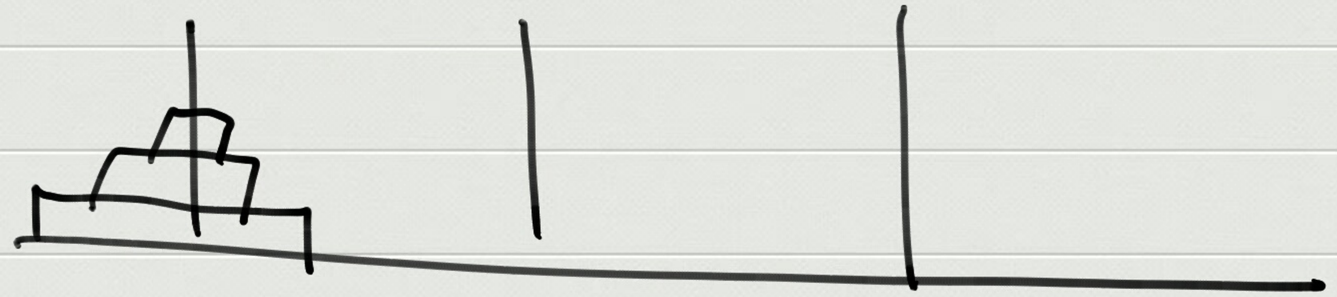
Classical example: Towers of Hanoi



The Tower of Hanoi puzzle

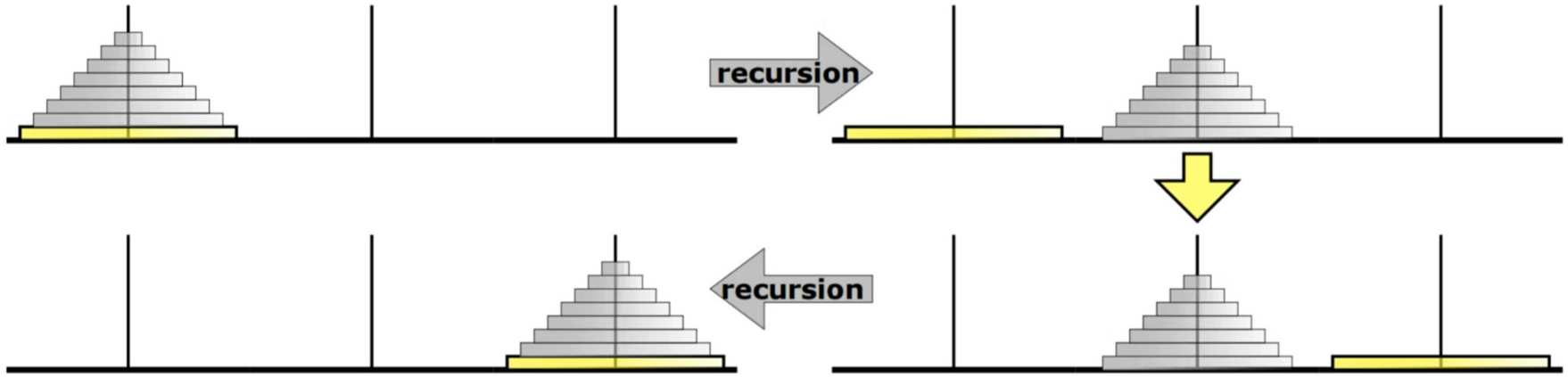
Strategy: think recursively!

Start small:



Bigger picture:

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



The Tower of Hanoi algorithm; ignore everything but the bottom disk

Proof of correctness:

Runtime:

Next: Sorting & searching

Interview question: Name the best sorting algorithm & justify your answer.

Merge Sort:
(supposedly, suggested by von Neumann in 1945)

Idea: ①

②

③

In action:

trace: recursive call

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	R	S	T	X

A mergesort example.

Algorithm: merging is the only hard part!

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

```
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]
```

Correctness:

Runtime: