

Algorithms - Spring '25

Induction

Pseudocode

Runtime



Recap

- Recding due by 8am, every day we have class
 - ↳ no excuses, but I'll drop lowest 3 to allow for illness / forgetting / travel
- HW O: due next Wed
- No class or office hour Monday
- Extra office hour on Tuesday from 3-5pm

Last time:

- Big - O
- Identities & Summations
- Induction

④

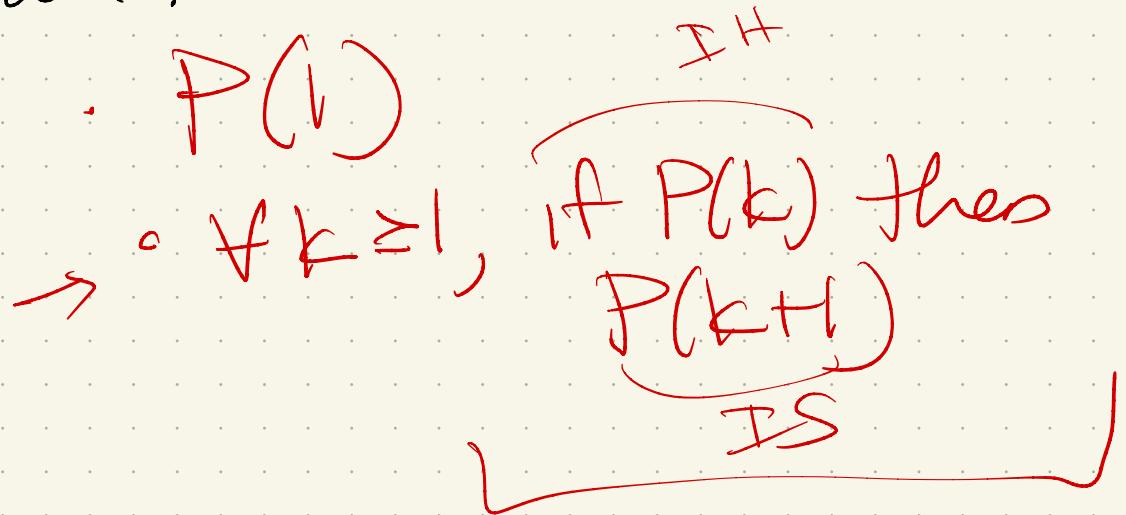
Induction

There is a template!
Base case: Prove statement for small value

Ind hypothesis: Assume true for values $\leq k$

Ind. step: Prove true for next value $k+1$

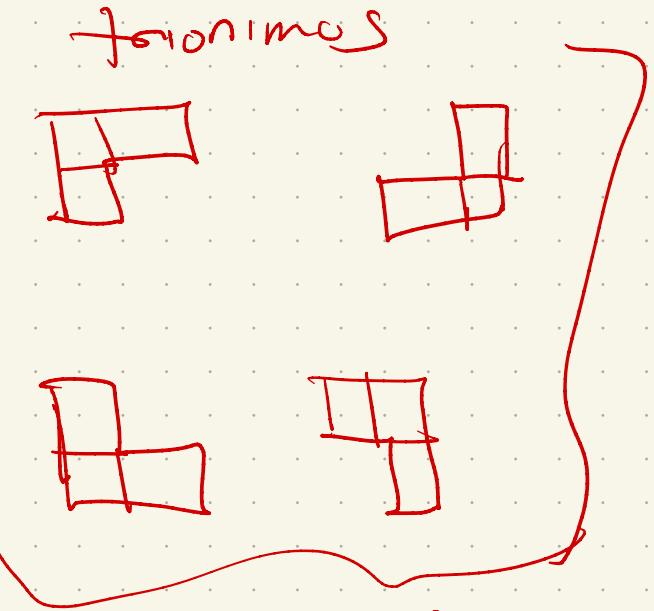
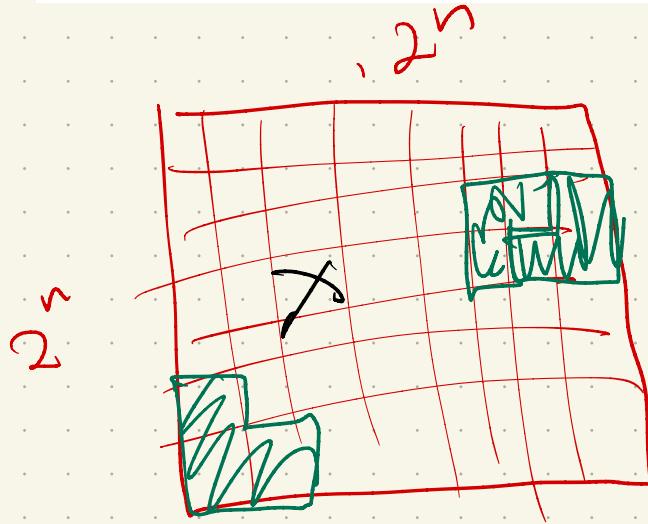
Think of this as "automating" a proof.



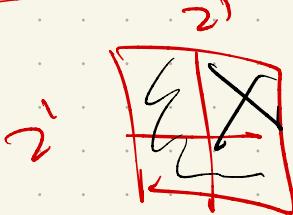
Learn it, use it, love it!

"Structural" induction:

Let n be a positive integer. Show that every $2^n \times 2^n$ checkerboard with one square removed can be tiled using right triominoes, where these pieces cover three squares at a time, as shown in Figure 4.

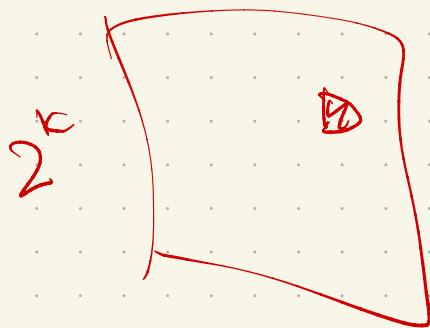


Base case: $2^1 \times 2^1$ board



No matter which square is removed
I can cover the other 3

IH: Assume I can tile
any $2^k \times 2^k$ board with
any 1 square removed



IS: Show I can do a $2^{k+1} \times 2^{k+1}$ board:

Induction on graphs/tree

3 Pseudo code + runtime: Discrete math examples

ALGORITHM 1 Finding the Maximum Element in a Finite Sequence.

```
procedure max( $a_1, a_2, \dots, a_n$ : integers)
max :=  $a_1$ 
for  $i := 2$  to  $n$ 
    if  $max < a_i$  then  $max := a_i$ 
return max {max is the largest element}
```

ALGORITHM 2 The Linear Search Algorithm.

```
procedure linear search( $x$ : integer,  $a_1, a_2, \dots, a_n$ : 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}
```

Pseudocode conventions here:

Variable assignment:

Boolean comparison:

$\text{if } (x = 5)$

Arrays: $A[0..n-1]$
- each $A[i]$

Loops: $\text{for } i \leftarrow 1 \text{ to } 100$
 $A[i] \leftarrow i$

Again, takes practice! Read
intro chapter + then give
HWO/HW1 a try.

Pseudocode format:

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

High level + readable.

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

It's about effective communication.

Initially:

- lots of examples
- lots of practice
- reach out if you have
questions
- in a pinch - peer
evaluation!

Example (& tie to runtimes):

Initializing arrays.

FIBONACCIMULTIPLY($X[0..m-1], Y[0..n-1]$):

$hold \leftarrow 0$

for $k \leftarrow 0$ to $n+m-1$

 for all i and j such that $i+j = k$

$hold \leftarrow hold + X[i] \cdot Y[j]$

$Z[k] \leftarrow hold \bmod 10$

$hold \leftarrow \lfloor hold/10 \rfloor$

return $Z[0..m+n-1]$

Space & runtime:

Q: What if all won't get filled?

More Complex: recursion

Algorithm 1 Quicksort

```
1: procedure QUICKSORT( $A, p, r$ )
2:   if  $p < r$  then
3:      $q = \text{PARTITION}(A, p, r)$ 
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

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