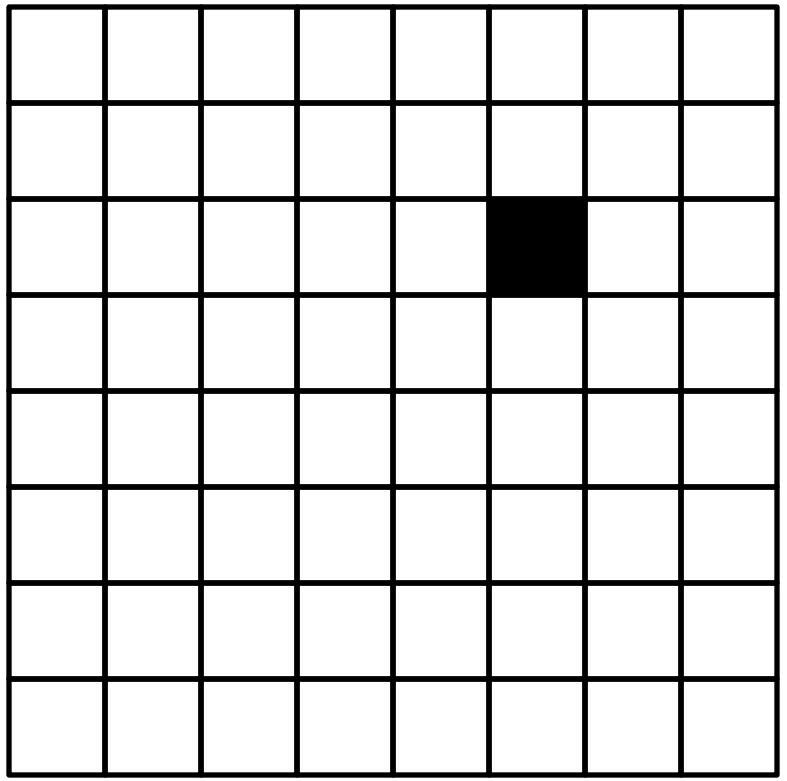
CS4102 Algorithms

Fall 2019

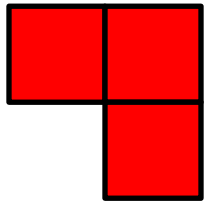
Warm up

Can you cover an 8×8 grid with 1 square missing using “trominoes?”

Can you cover this?



With these?



Office Hours

- Mondays, 10am-11am, 2-4pm

Today's Keywords

- Recursion
- Recurrences
- Asymptotic notation
- Divide and Conquer
- Trominoes
- Merge Sort

CLRS Readings

- Chapters 3 & 4

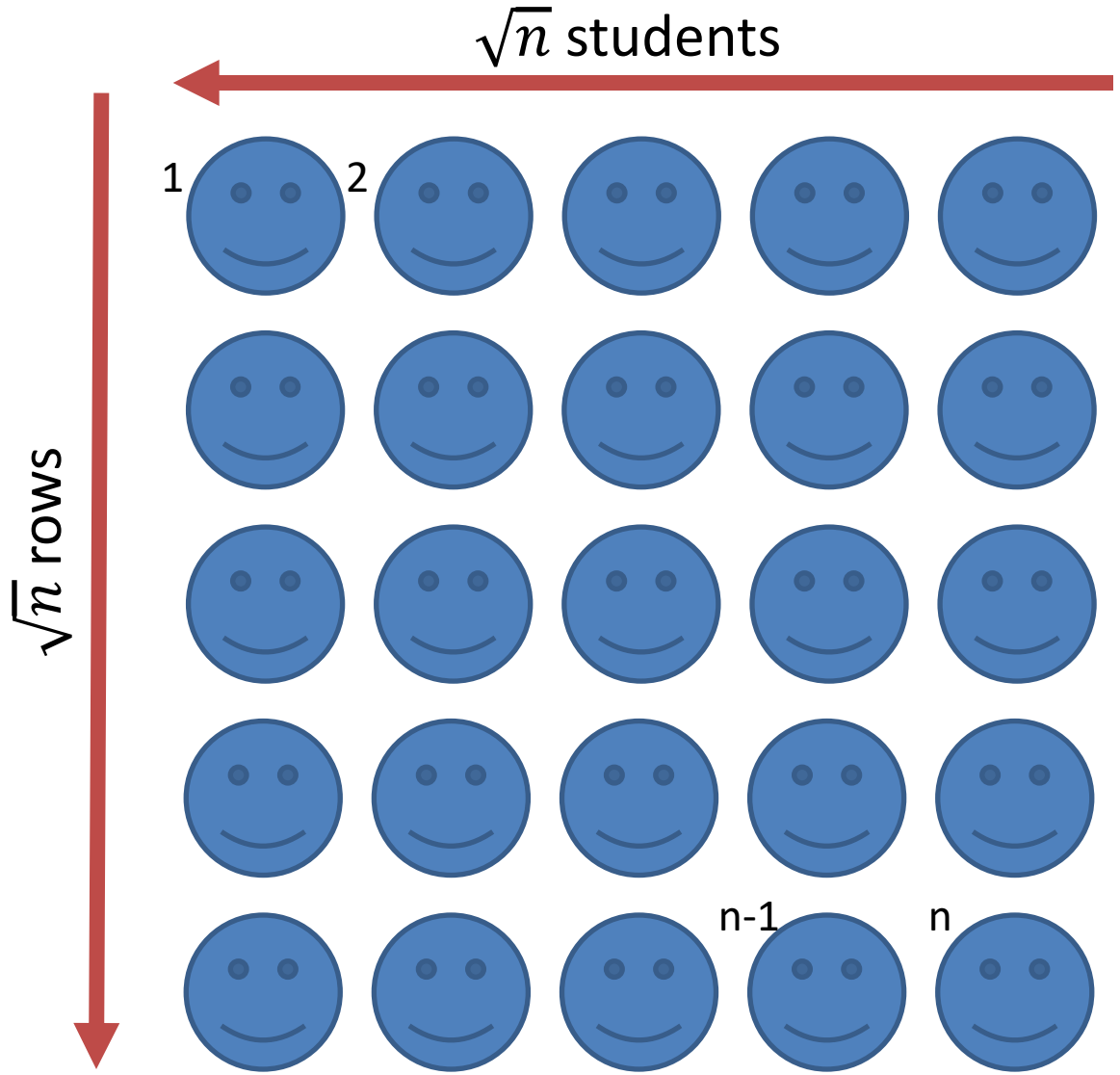
Homeworks

- Hw0 due 11pm Tuesday, Sept 2
  - Submit 2 attachments (zip and pdf)
- Hw1 released Tuesday, Sept 2
  - Due 11pm Thursday, Sept 12
  - Written (use Latex!)
  - Asymptotic notation
  - Recurrences
  - Divide and conquer

Attendance

- How many people are here today?
- Naïve algorithm
  1. Everyone stand
  2. Professor walks around counting people
  3. When counted, sit down
- Run time?
  - Class of n students
  - O(n)
- Other suggestions?

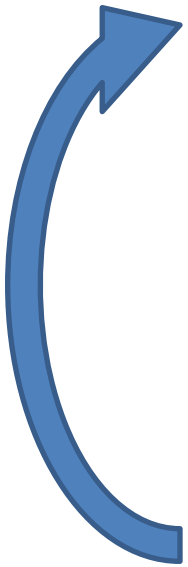
Good Attendance



$O(\sqrt{n})$

Better Attendance

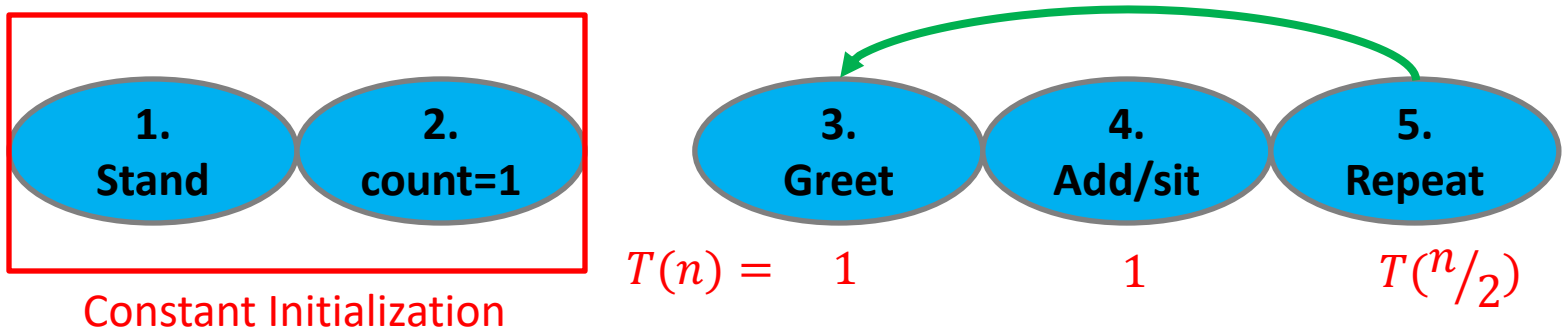
1. Everyone Stand
2. Initialize your “count” to 1
3. Greet a neighbor who is standing: share your name, full date of birth(pause if odd one out)
4. If you are older: give “count” to younger and sit.
   Else if you are younger: add your “count” with older’s
5. If you are standing and have a standing neighbor, go to 3



What was the run time of this algorithm?

What are we going to count?

Attendance Algorithm Analysis



Constant Initialization

$$T(n) = 1 + 1 + T(n/2)$$

How can we "solve" this?

$$T(1) = 3$$

Base case?

Do not need to be exact, asymptotic bound is fine.
Why?

Let's solve the recurrence!

$\log_2(n) = k \log_2 2 = 1$

Special case: $n = 2^k$

$T(1) = 3$

$T(n) = 2 + T(n/2)$

$2 + T(n/4)$

$2 + T(n/8)$

$\cdots$

3

k

$$T(n) = 3 + \sum_{i=0}^{\log_2 n} 2 = 2\log_2 n + 3$$

What if $n \neq 2^k$?

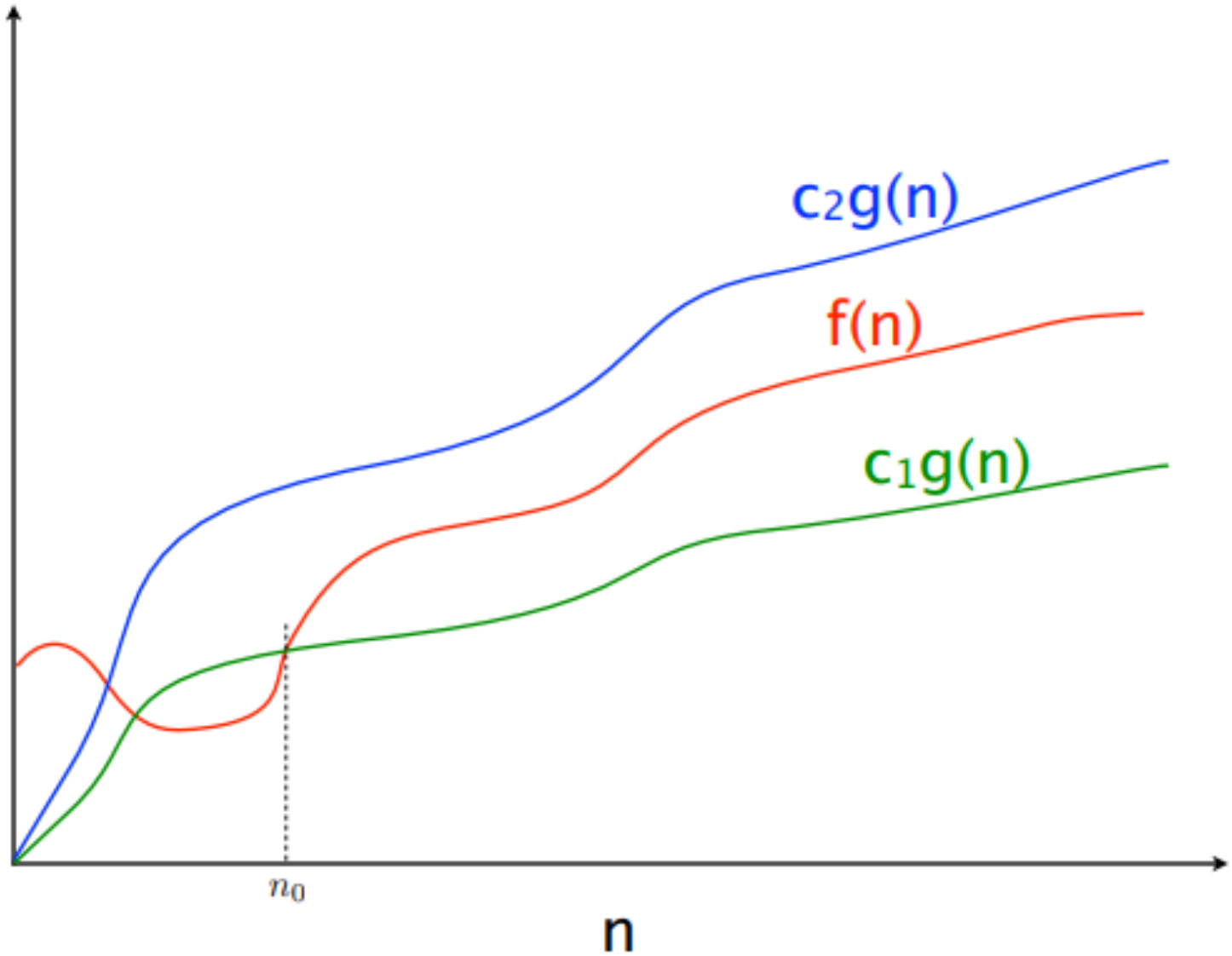
- More people in the room → more time
  - $\forall\ 0 < n < m, T(n) < T(m)$, where $2^k < n < 2^{k+1} = m$,
    i.e., $k < \log n < k + 1$
  - $T(n) \leq T(m) = T\left(2^{k+1}\right) = T\left(2^{\lceil \log_2 n \rceil}\right) = 2\lceil \log_2 n \rceil + 3 = O(\log n)$

These are unimportant.
Why?

Asymptotic Notation*

- $O(g(n))$
  - At most within constant of g for large n
  - {functions f |∃ constants $c, n_0 > 0$ s.t. $\forall n > n_0, f(n) \leq c \cdot g(n)$}
- $\Omega(g(n))$
  - At least within constant of g for large n
  - {functions f |∃ constants $c, n_0 > 0$s.t. $\forall n > n_0, f(n) \geq c \cdot g(n)$}
- $\Theta\big(g(n)\big)$
  - "Tightly" within constant of g for large n
  - $\Omega\big(g(n)\big) \cap O(g(n))$

*CLRS Chapter 3



$f(n) = O(g(n))$

$f(n) = \Theta(g(n))$

$f(n) = \Omega(g(n))$

Asymptotic Notation Example

- Show: $n \log n \in O(n^2)$

$O(g(n)) = \{f : \exists c, n_0 > 0, \forall n > n_0, f(n) \leq g(n) \cdot c\}$

$\exists c, n_0 > 0 : \forall n > n_0 \quad n \log n \leq c \cdot n^2$

let $c = 1, n_0 = 1$

$f(1) = 1 \log 1 = 0 \qquad g(1) = 1^2 = 1$

$f(1) < g(1) \cdot c = g(1) \cdot 1$

$0 \leq 1 \cdot 1^2 = 1.$

$\forall n > n_0 = 1$

$n \log n \leq n^2$

$\log n \leq n \quad \forall n > n_0$

Therefore $n \log n \in O(n^2)$

Asymptotic Notation Example

Direct Proof!

- To Show: $n \log n \in O(n^2)$
  - **Technique:** Find $c, n_0 > 0$ s.t. $\forall n > n_0, n \log n \leq c \cdot n^2$
  - **Proof:** Let $c = 1, n_0 = 1$. Then,

$$n_0 \log n_0 = (1) \log (1) = 0,$$
$$c\, n_0^2 = 1 \cdot 1^2 = 1,$$
$$0 \leq 1.$$

$$\forall n \geq 1, \log(n) < n \Rightarrow n \log n \leq n^2 \quad \square$$

Asymptotic Notation Example

$O(g(n)) = \{f: \exists c, n_0 > 0 : \forall n > n_0 \quad f(n) \leq g(n) \cdot c\}$

- Show: $n^2 \notin O(n)$

Proof: Assuming $n^2 \in O(n)$. then $\exists c, n_0 > 0 : \forall n > n_0 \quad n^2 \leq c \cdot n$.

$n^2 \leq cn$

$n \leq c$

$c \geq n$

c is not a constant. Contradiction!

therefore $n^2 \notin O(n)$

let $x = \max(c, n_0) + 1 \Rightarrow x > c$

$x^2 \leq c \cdot x$

$x \leq c$, contradiction

Therefore $n^2 \notin O(n)$

Asymptotic Notation Example

- To Show: $n^2 \notin O(n)$
  - **Technique: Contradiction**
  - **Proof:** Assume $n^2 \in O(n)$. Then $\exists c, n_0 > 0$ s.t. $\forall n > n_0, n^2 \leq cn$
    Let us derive constant c. For all $n > n_0 > 0$, we know:
    $cn \geq n^2$,
    $c \geq n$.

    Since c is dependent on n, it is not a constant.
    Contradiction. Therefore $n^2 \notin O(n)$. □

Proof by Contradiction!

Proof Techniques

- Direct Proof 
  - From the assumptions and definitions, directly derive the statement
- Proof by Contradiction 
  - Assume the statement is true, then find a contradiction
- Proof by Cases
- Induction

Asymptotic Notation

- $o(g(n))$
  - Below *any* constant of g for large n
  - {functions f |∀ constants c, ∃n_0 s.t. $\forall n > n_0, f(n) < c \cdot g(n)$}

- $\omega(g(n))$
  - Above *any* constant of g for large n
  - {functions f |∀ constants c, ∃n_0 s.t. $\forall n > n_0, f(n) > c \cdot g(n)$}

- $\theta\big(g(n)\big)$?
  - $o(g(n)) \cap \omega(g(n)) = \emptyset$

Asymptotic Notation Example

- $o(g(n)) = \{\text{functions } f \mid \forall \text{ constants } c, \exists n_0 \text{ s.t. } \forall n > n_0, f(n) < c \cdot g(n)\}$
- Show: $n \log n \in o(n^2)$

$\forall c > 0, \exists n_0 > 0, \forall n > n_0 \quad n \log n < c \cdot n^2$

$n \log n < c \cdot n^2$

$\log n < cn$

$\frac{\log n}{n} < c$

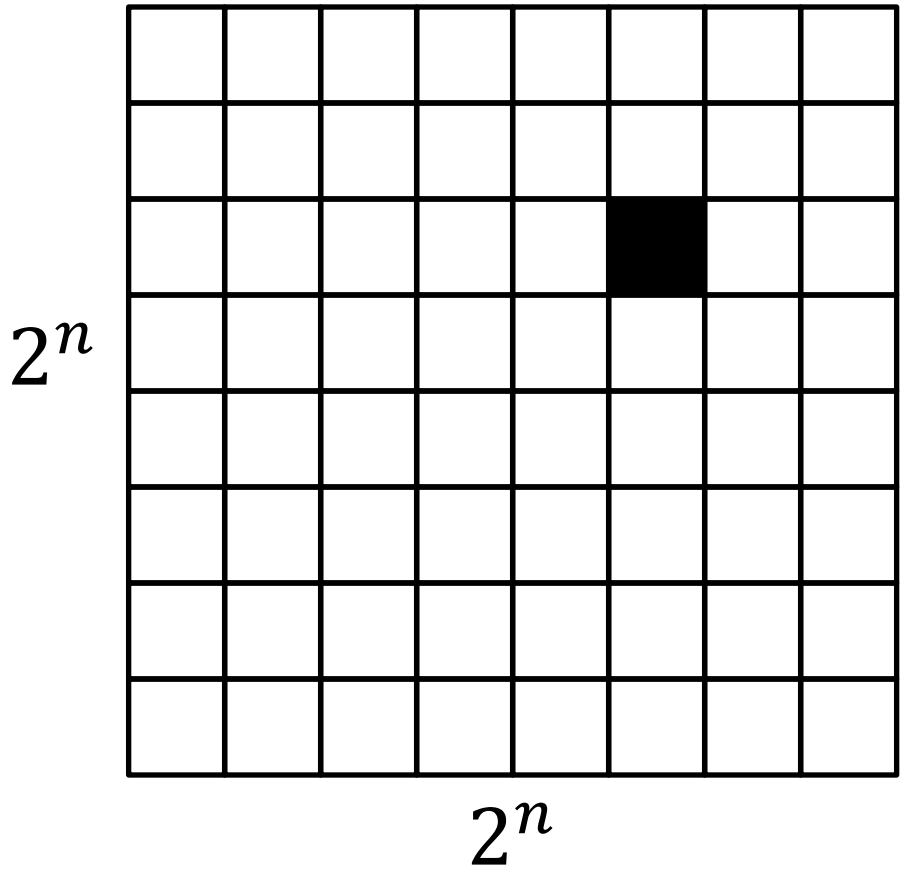
$\lim_{n \to \infty} \frac{\log n}{n} = 0$

therefore $n \log n \in o(n^2)$

Asymptotic Notation Example

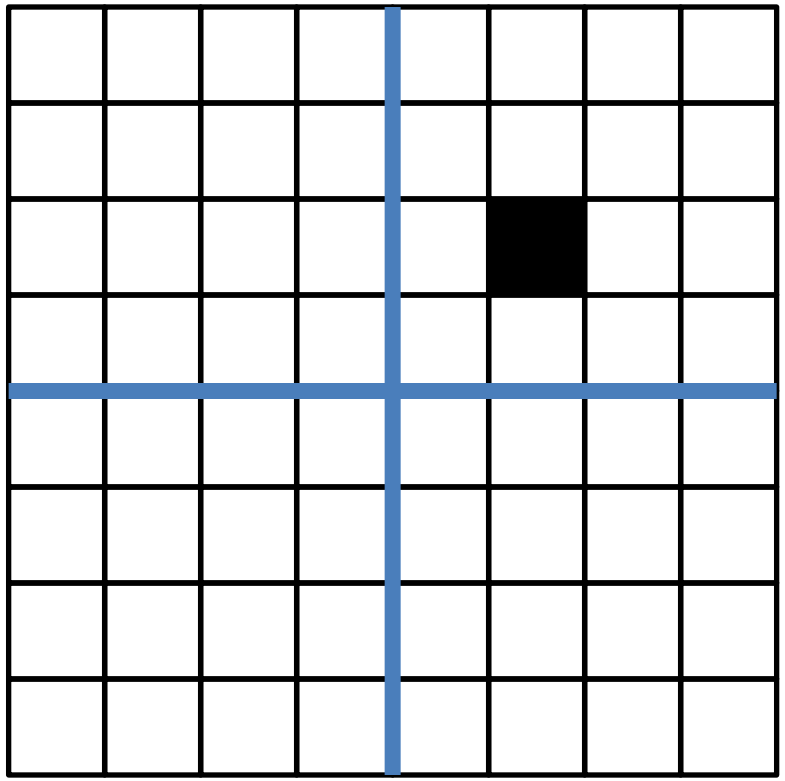
- $o\big(g(n)\big) = \{\text{functions } f \mid \forall \text{ constants } c, \exists n_0 \text{ s.t. } \forall n > n_0, f(n) < c \cdot g(n)\}$
- To Show: $n \log n \in o(n^2)$
  - given any c find a $n_0 > 0$ s.t. $\forall n > n_0, n \log n < c \cdot n^2$
  - Find a value of n in terms of c:
    - $n \log n < c \cdot n^2$
    - $\log n < c \cdot n$
    - $\frac{\log n}{n} < c$
  - For a given c, select any value of n_0 such that $\frac{\log n}{n} < c$

Trominoes Puzzle Solution



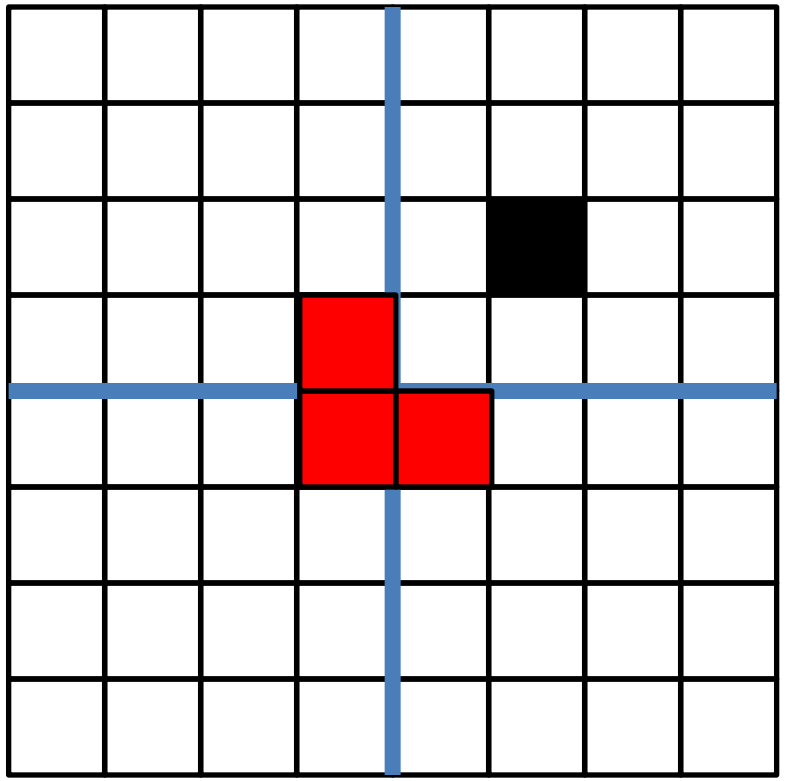
What about larger boards?

Trominoes Puzzle Solution



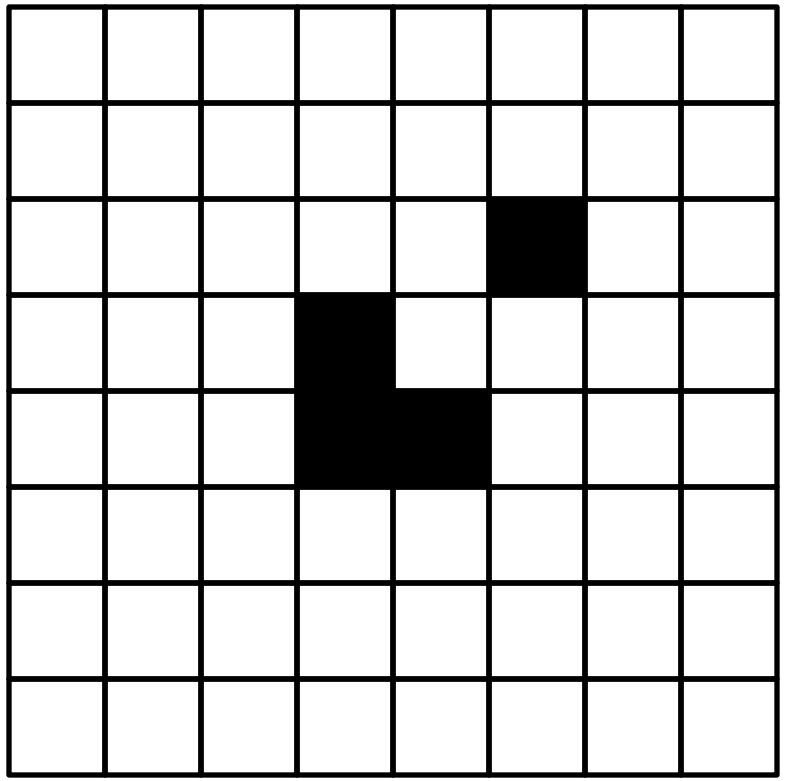
Divide the board into quadrants

Trominoes Puzzle Solution



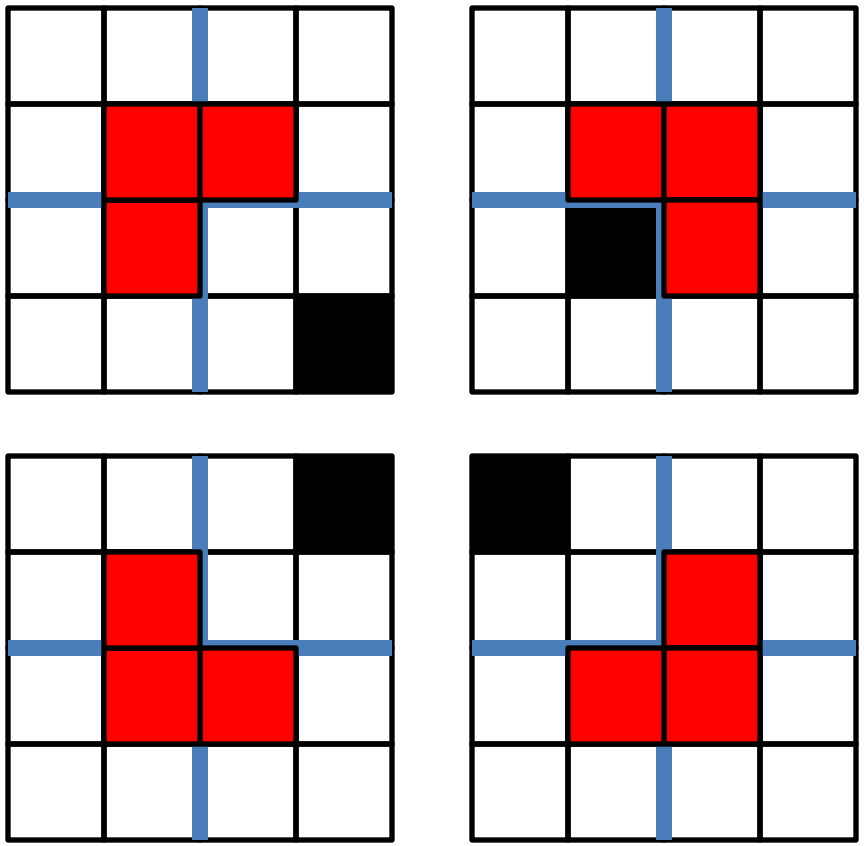
Place a tromino to occupy the three quadrants without the missing piece

Trominoes Puzzle Solution



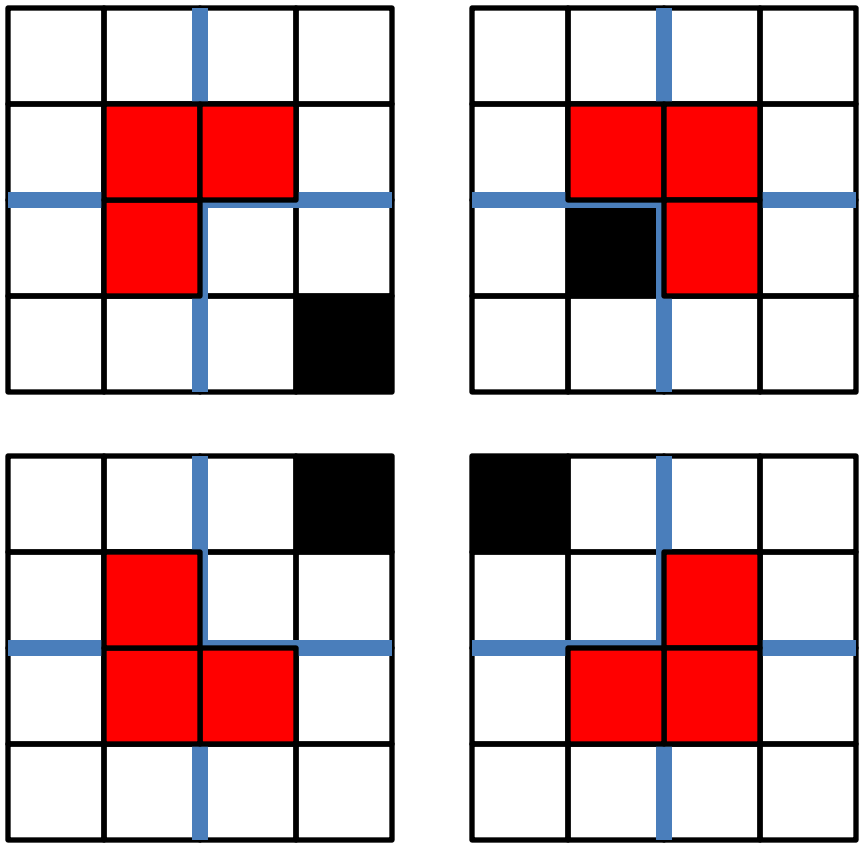
Each quadrant is now a smaller subproblem

Trominoes Puzzle Solution



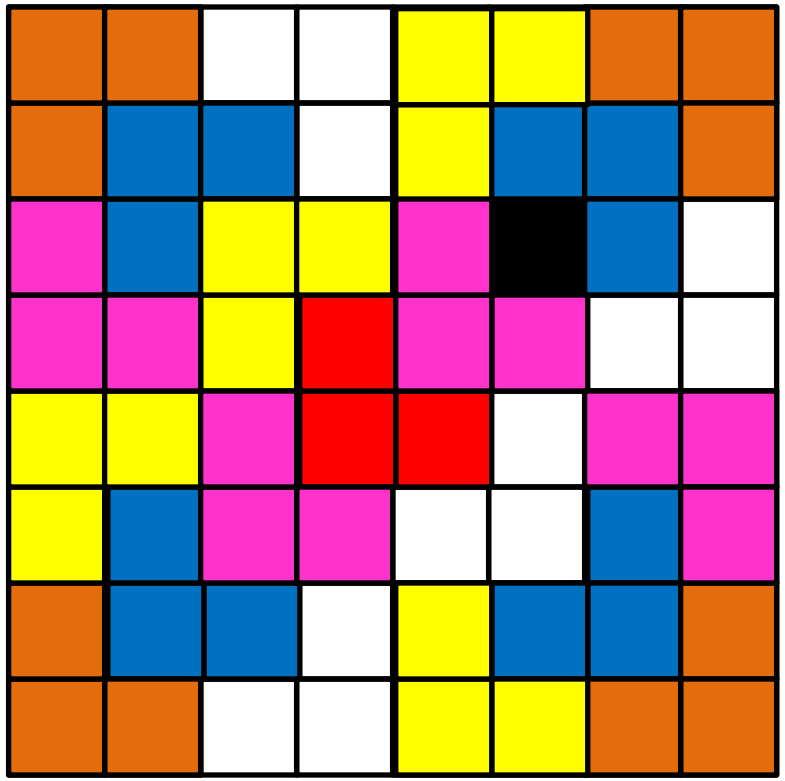
Solve **Recursively**

Divide and Conquer

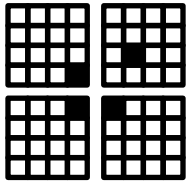


Our first algorithmic technique!

Trominoes Puzzle Solution

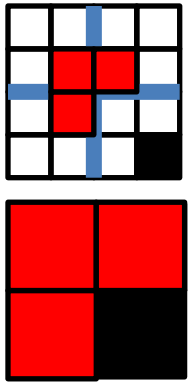


Divide and Conquer*

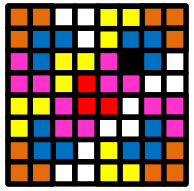


When is this a good strategy?

- **Divide**:
  - Break the problem into multiple subproblems, each smaller instances of the original
- **Conquer**:
  - If the suproblems are “large”:
    - Solve each subproblem recursively
  - If the subproblems are “small”:
    - Solve them directly (base case)



- **Combine**:
  - Merge together solutions to subproblems



*CLRS Chapter 4