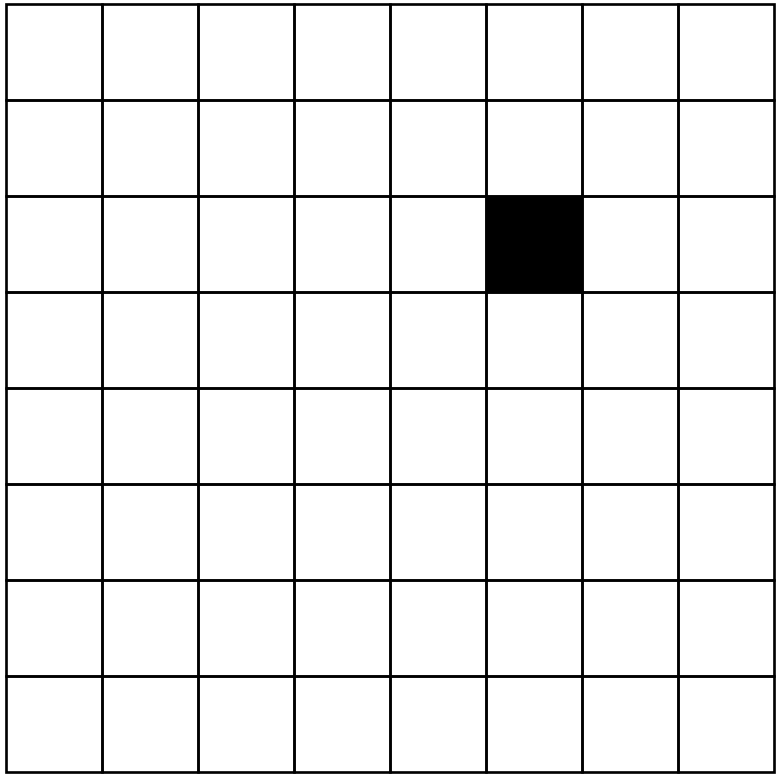
CS 4102: Algorithms

Lecture 2: Recurrences

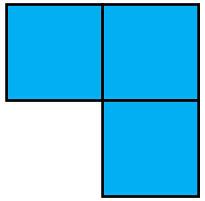
David Wu

Fall 2019

Warm Up



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

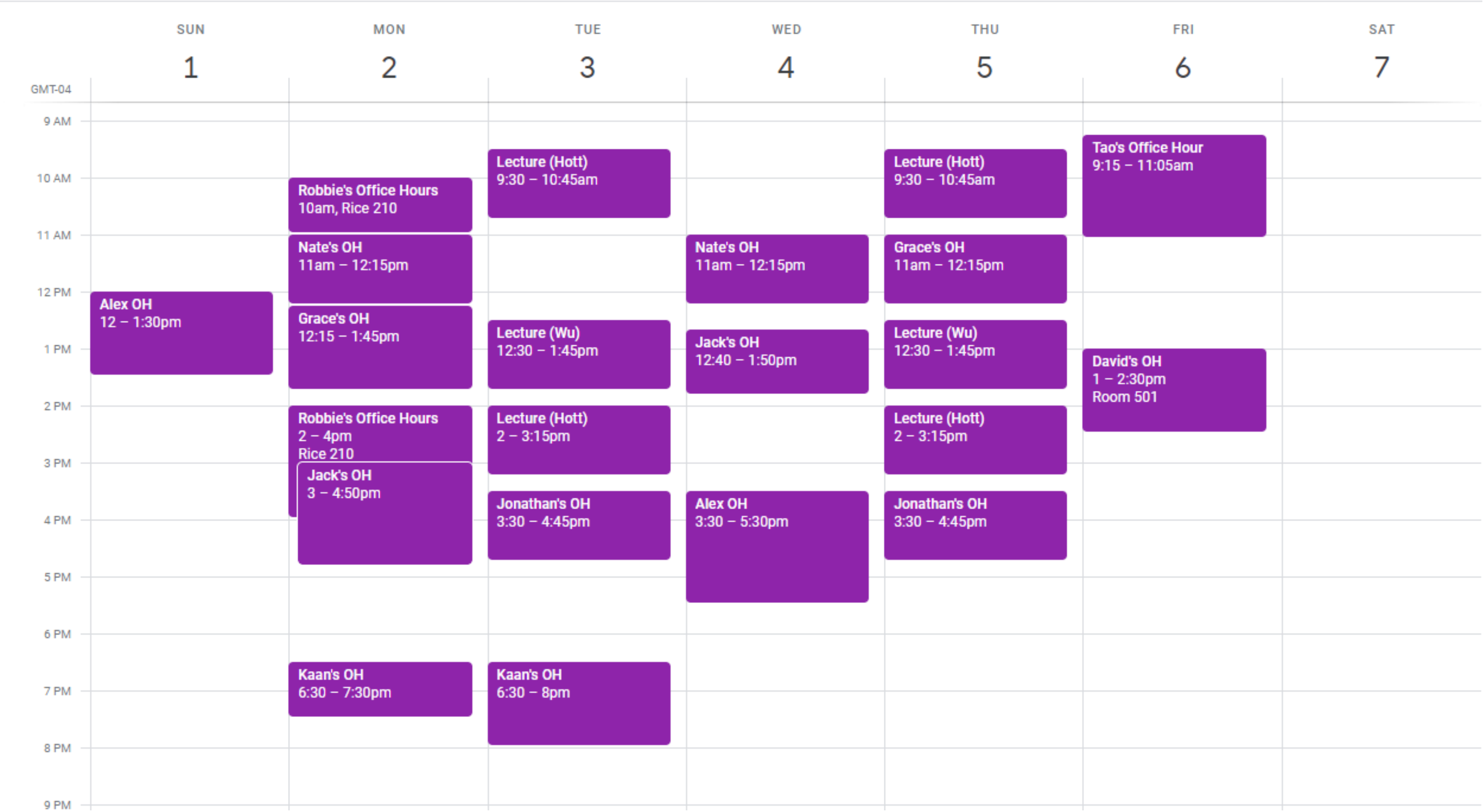


Tromino

https://nstarr.people.amherst.edu/trom/puzzle-8by8/

Office Hours

Friday, 1:00-2:30pm, Rice 501



Today’s Keywords

Recursion

Recurrences

Asymptotic notation and proof techniques

Divide and conquer

Trominoes

Merge sort

CLRS Readings: Chapters 3 & 4

Homework

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

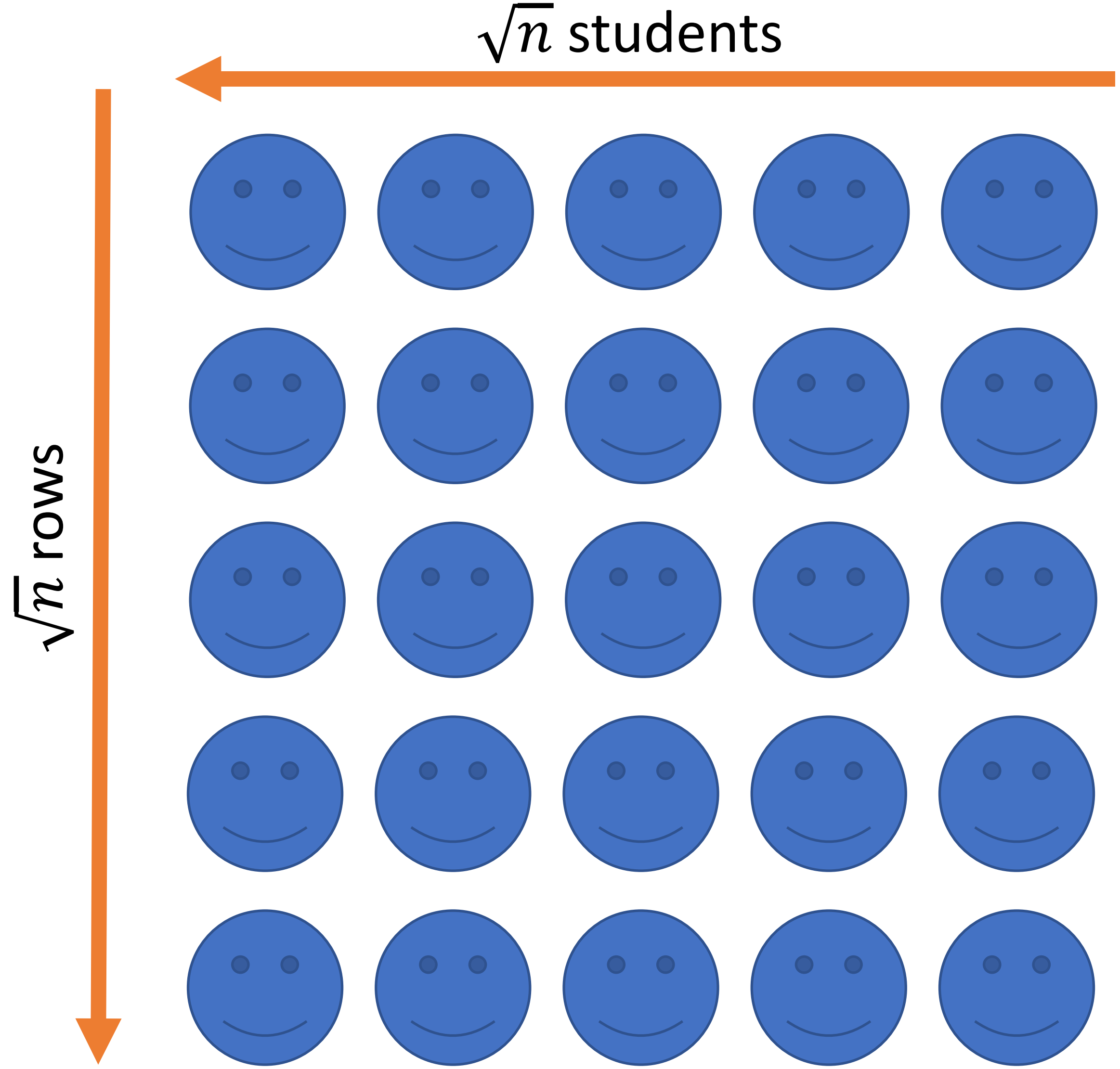
- Everyone stand
- Professor walks around counting people
- When counted, sit down

Complexity?

- Class of n students
- $O(n)$ “rounds”

Other suggestions?

Good Attendance



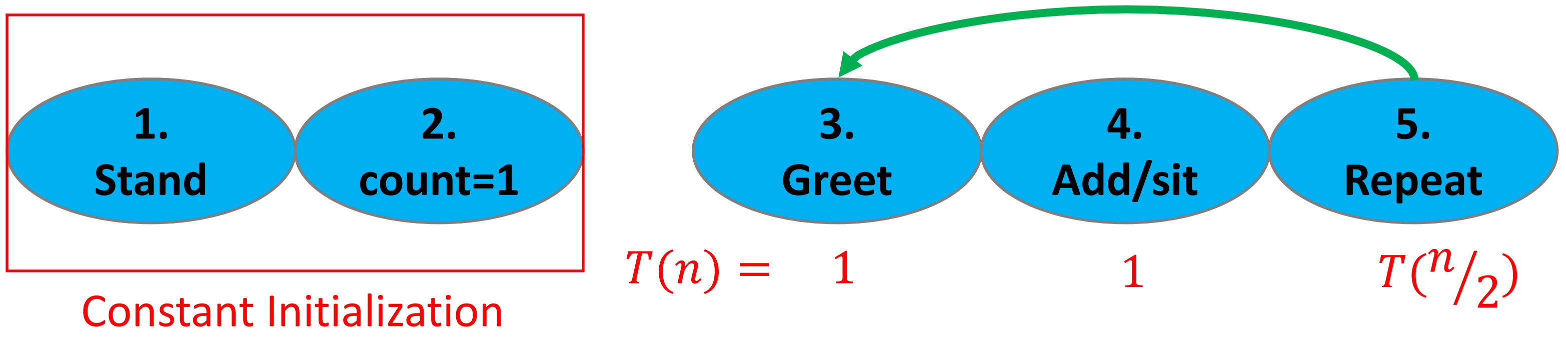
$O(\sqrt{n})$

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



Recurrence

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

$$T(1) = 3$$

How can we “solve” this?

Base case?

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

Why?

Let’s Solve the Recurrence!

Special case: $n = 2^k$

$T(1) = 3$

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

$2 + T(n/4)$

$2 + T(n/8)$

$\dots$

3

k

$$T(n) = 3 + \sum_{i=1}^{\log_2 n} 2 = 2\log_2 n + 3$$

What if $n \neq 2^k$?

More people in the room ⇒ more time

- $\forall\, 0 < n < m, T(n) < T(m)$
- $T(n) \leq T(m) = T\left(2^{\lceil \log_2 n \rceil}\right) = 2\lceil \log_2 n \rceil + 3$

$= O(\log n)$

These are unimportant.
Why?

Asymptotic Notation

[CLRS Chapter 3]

$$O(g(n))$$

- At most within constant factor of g for sufficiently large n
- $\{\text{functions } f : \exists \text{ constants } c, n_0 > 0 \text{ such that } \forall n > n_0, f(n) \leq c \cdot g(n)\}$

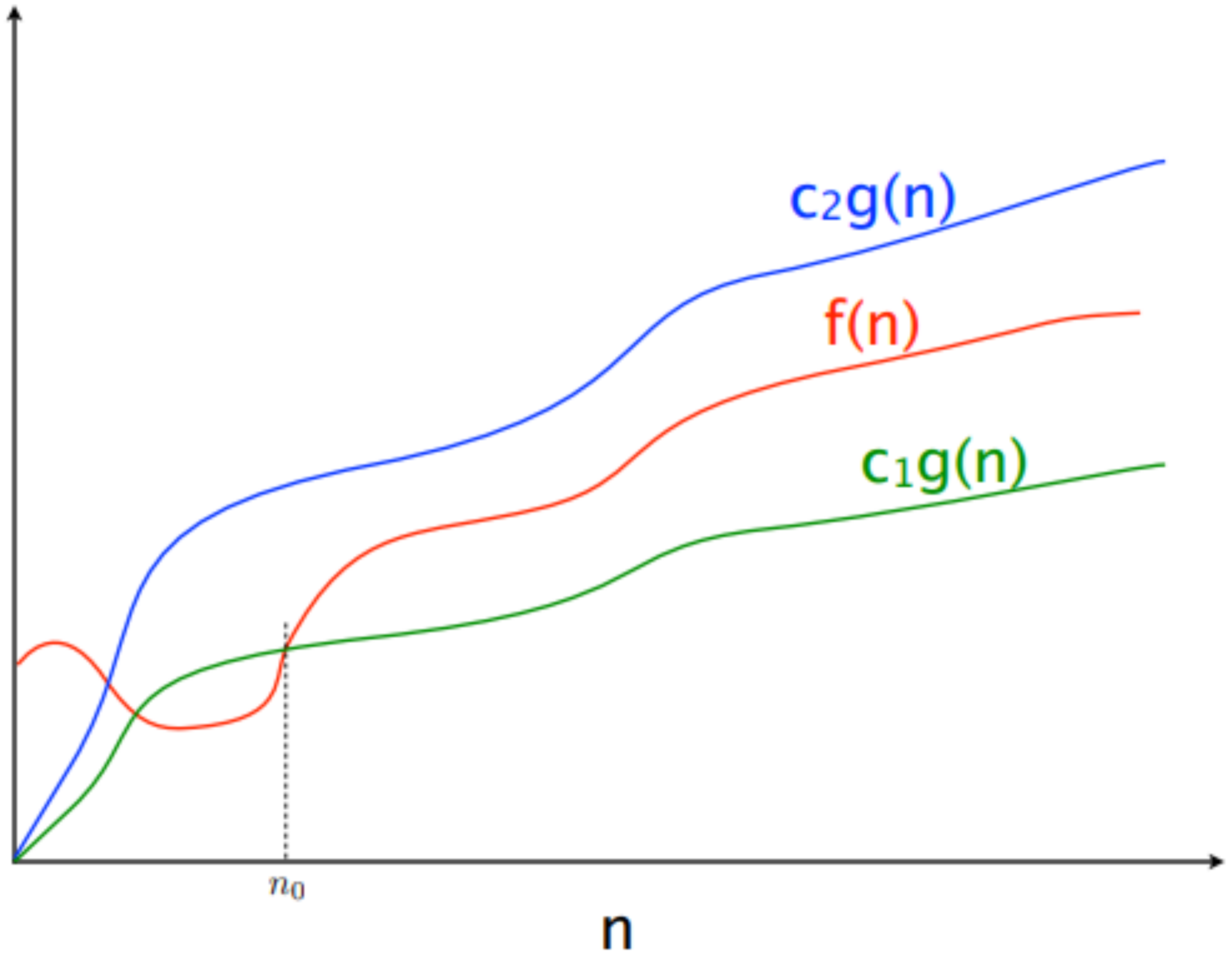
$$\Omega(g(n))$$

- At least within constant factor of g for sufficiently large n
- $\{\text{functions } f : \exists \text{ constants } c, n_0 > 0 \text{ such that } \forall n > n_0, f(n) \geq c \cdot g(n)\}$

$$\Theta(g(n))$$

- "Tightly" within constant factor of g for sufficiently large n
- $\Omega(g(n)) \cap O(g(n))$

Asymptotic Notation



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

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

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

Asymptotic Notation Example

Direct Proof

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

Proof Technique: Give explicit constants $c, n_0 > 0$

- Let $c = 1, n_0 = 1$
- $f(1) = (1) \log\ (1) = 0, \quad g(1) = 1 \cdot 1^2 = 1$
- $\forall n \geq 1,\ \log(n) < n \Rightarrow \forall n \geq 1,\ n \log n \leq n^2$

$\exists$ constants $c, n_0 > 0$ such that $\forall n > n_0, f(n) \leq c \cdot g(n)$

Asymptotic Notation Example

Indirect Proof

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

Proof Technique: Proof by <u>contradiction</u>

- Assume the opposite: namely, that $n^2 \in O(n)$
- Then $\exists c, n_0 > 0$ such that $\forall n > n_0, n^2 \leq cn$
- Consider $n = \max(c, n_0) + 1$. In particular, $n > c$ and $n > n_0$
- Then $n^2 = n \cdot n > cn$, which is a contradiction

$\exists$ constants $c, n_0 > 0$ such that $\forall n > n_0, f(n) \leq c \cdot g(n)$

Proof Techniques

Direct Proof

- From the assumptions and definitions, directly derive the statement

Indirect Proof (Proof by Contradiction)

- Assume the statement is true, then find a contradiction

Proof by Cases

Induction

More Asymptotic Notation

$$o(g(n))$$

- Smaller than *any* constant factor of g for sufficiently large n
- {functions f : $\forall$ constants $c > 0$, $\exists n_0$ such that $\forall n > n_0, f(n) < c \cdot g(n)$}

Equivalently, ratio of $\frac{f(n)}{g(n)}$ is decreasing and tends towards 0:

$$f(n) \in o\big(g(n)\big) \Leftrightarrow \lim_{n \to \infty} \frac{f(n)}{g(n)} = 0$$

More Asymptotic Notation

$$o(g(n))$$

- Smaller than *any* constant factor of g for sufficiently large n
- {functions f : $\forall$ constants $c > 0$, $\exists n_0$ such that $\forall n > n_0, f(n) < c \cdot g(n)$}

$$\omega(g(n))$$

- Greater than *any* constant factor of g for large n
- {functions f : $\forall$ constants $c > 0$, $\exists n_0$ such that $\forall n > n_0, f(n) > c \cdot g(n)$}

Equivalently, $f(n) \in \omega\big(g(n)\big) \Leftrightarrow \lim_{n\to\infty} \frac{g(n)}{f(n)} = 0$

More Asymptotic Notation

$$o(g(n))$$

- Smaller than *any* constant factor of g for sufficiently large n
- $\{$functions f : $\forall$ constants $c > 0, \exists n_0$ such that $\forall n > n_0, f(n) < c \cdot g(n)\}$

$$\omega(g(n))$$

- Greater than *any* constant factor of g for large n
- $\{$functions f : $\forall$ constants $c > 0, \exists n_0$ such that $\forall n > n_0, f(n) > c \cdot g(n)\}$

$$\theta\big(g(n)\big)$$

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

Another Asymptotic Notation Example

Show: $n \log n \in o(n^2)$

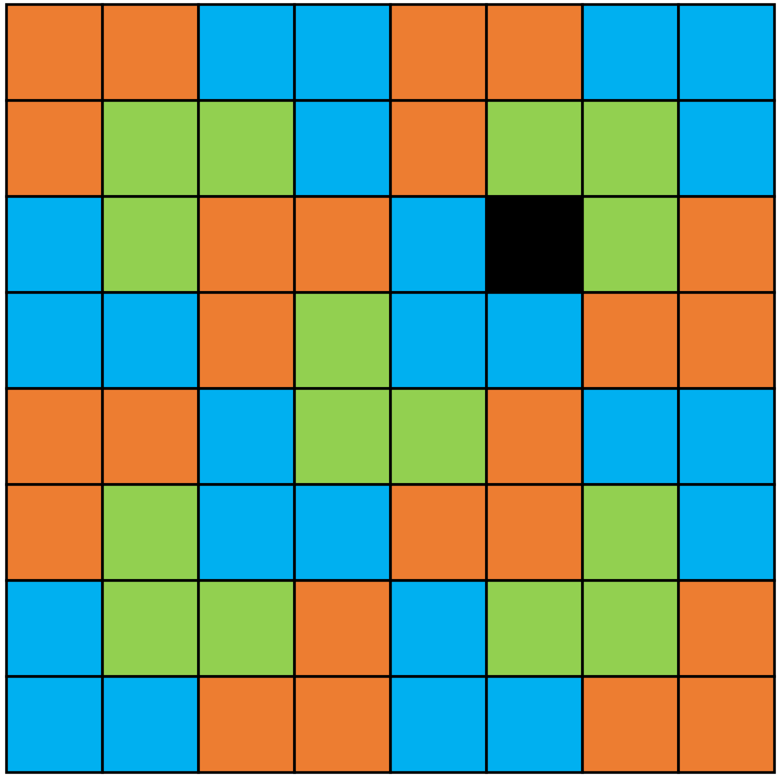
Direct Proof

Proof Technique: Show the statement directly

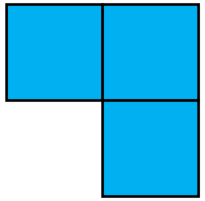
- $\lim_{n\to\infty} \frac{n \log n}{n^2} = \lim_{n\to\infty} \frac{\log n}{n} = 0$
- Equivalently, for every constant $c > 0$, we can find an n_0 such that $\frac{\log n_0}{n_0} = c$. Then for all $n > n_0$, $n \log n < c\, n^2$ since $\frac{\log n}{n}$ is a decreasing function

$\forall$ constants $c > 0, \exists n_0$ such that $\forall n > n_0, f(n) < c \cdot g(n)$

Back to Trominoes

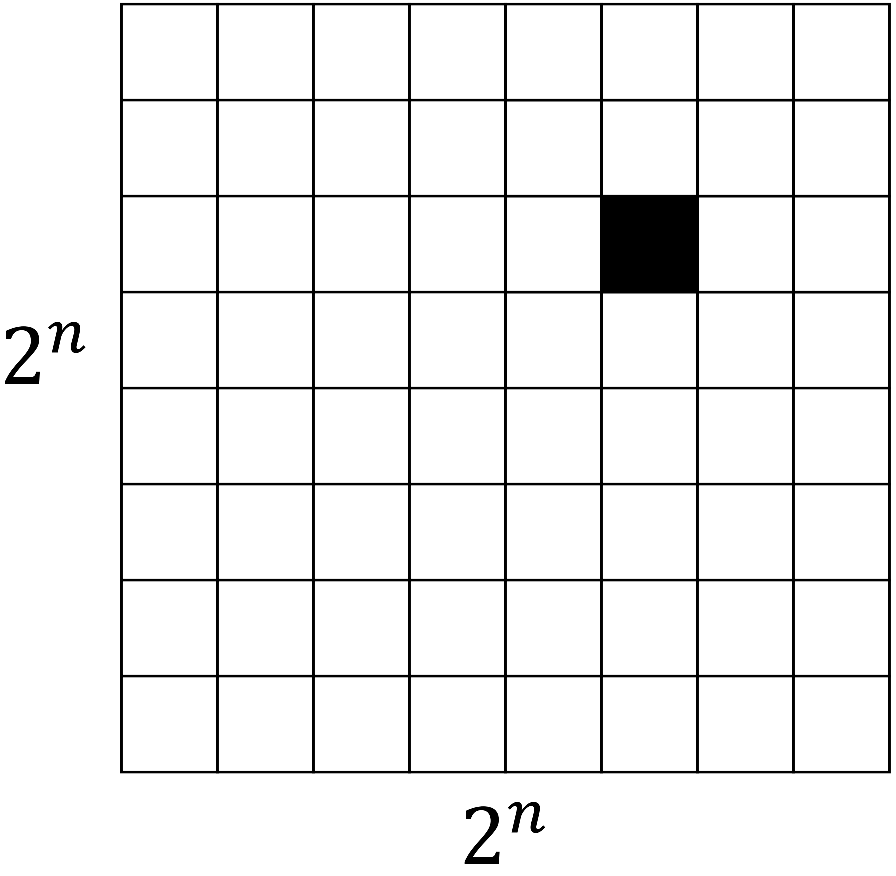


Can you cover an 8×8 grid with 1 square missing using “trominoes?”



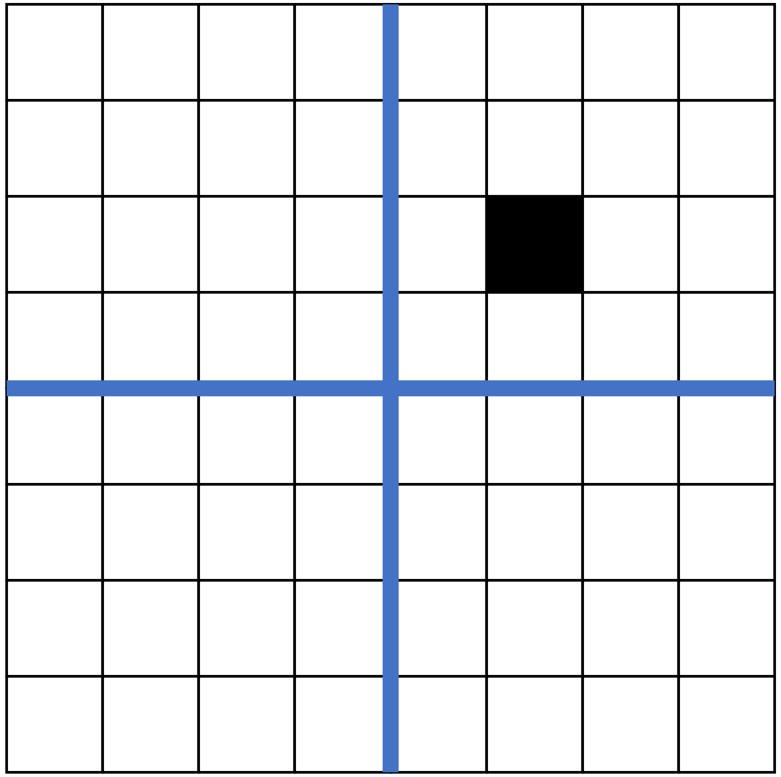
Tromino

Trominoes Puzzle Solution



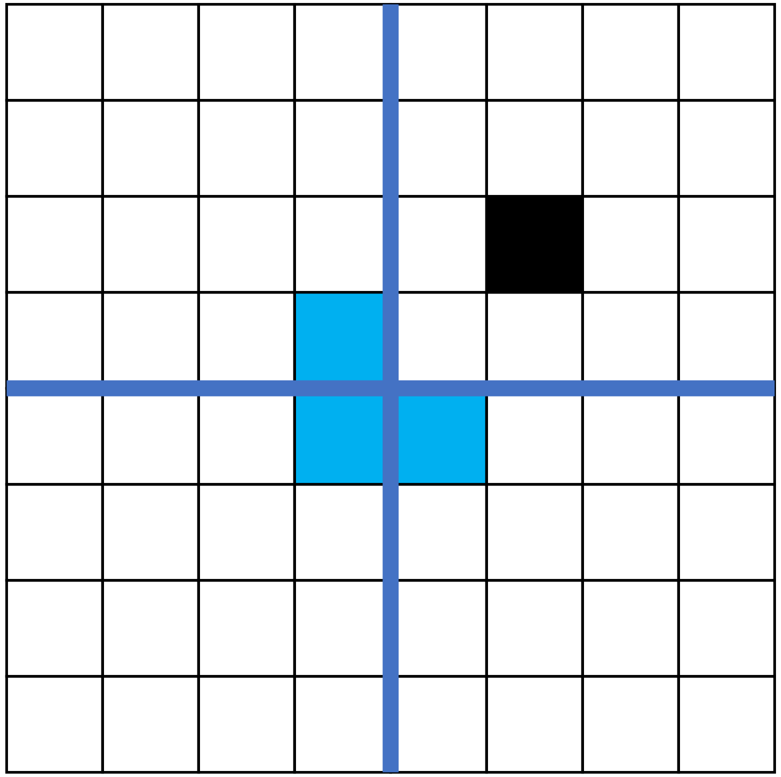
What about larger boards?

Trominoes Puzzle Solution



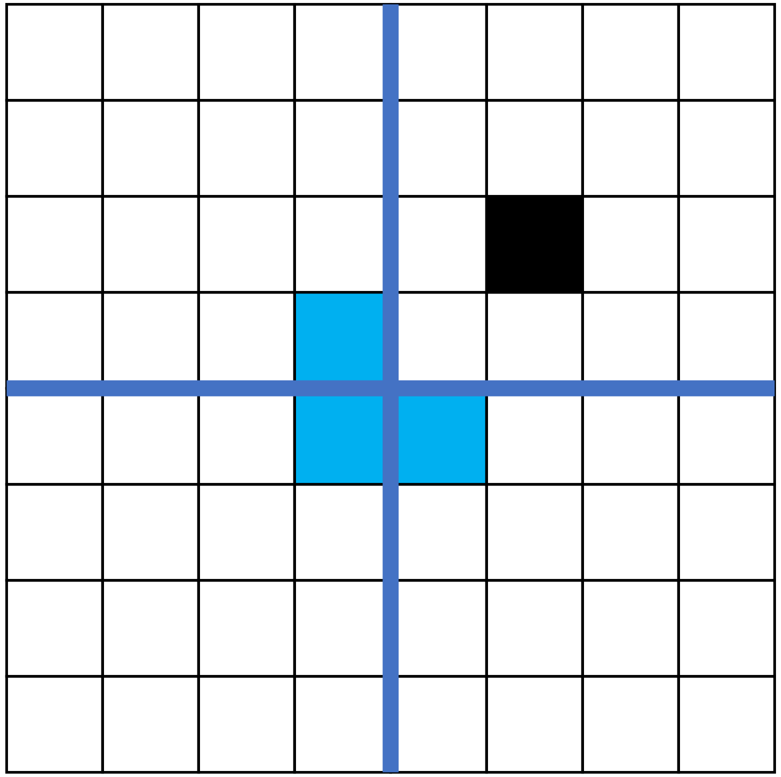
Divide the board into quadrants

Trominoes Puzzle Solution



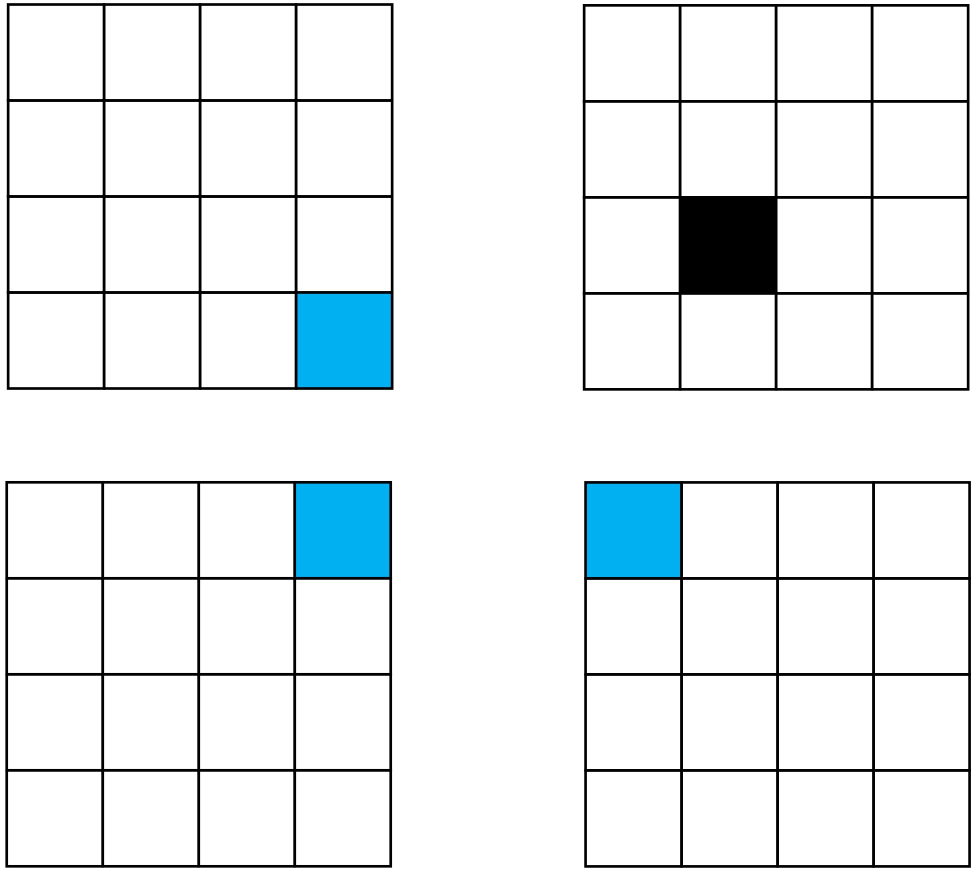
Place a tromino to occupy the three quadrants without the missing piece

Trominoes Puzzle Solution



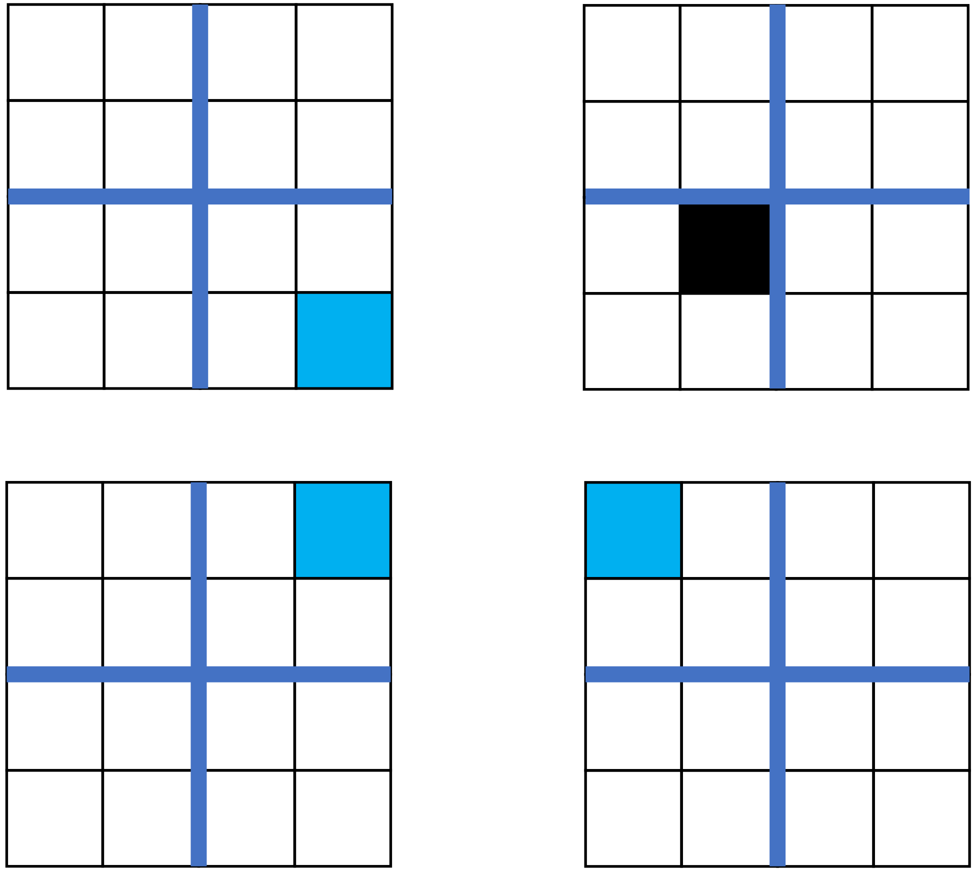
Place a tromino to occupy the three quadrants without the missing piece

Trominoes Puzzle Solution



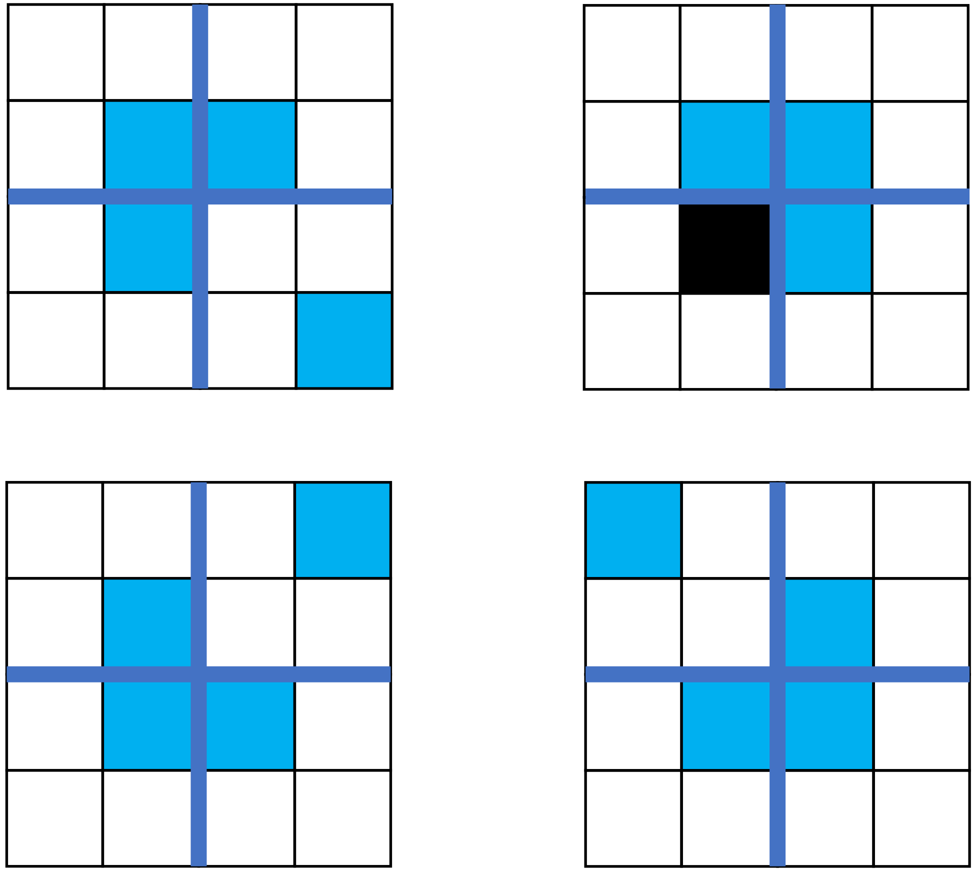
Observe: Each quadrant is now a smaller subproblem!

Trominoes Puzzle Solution



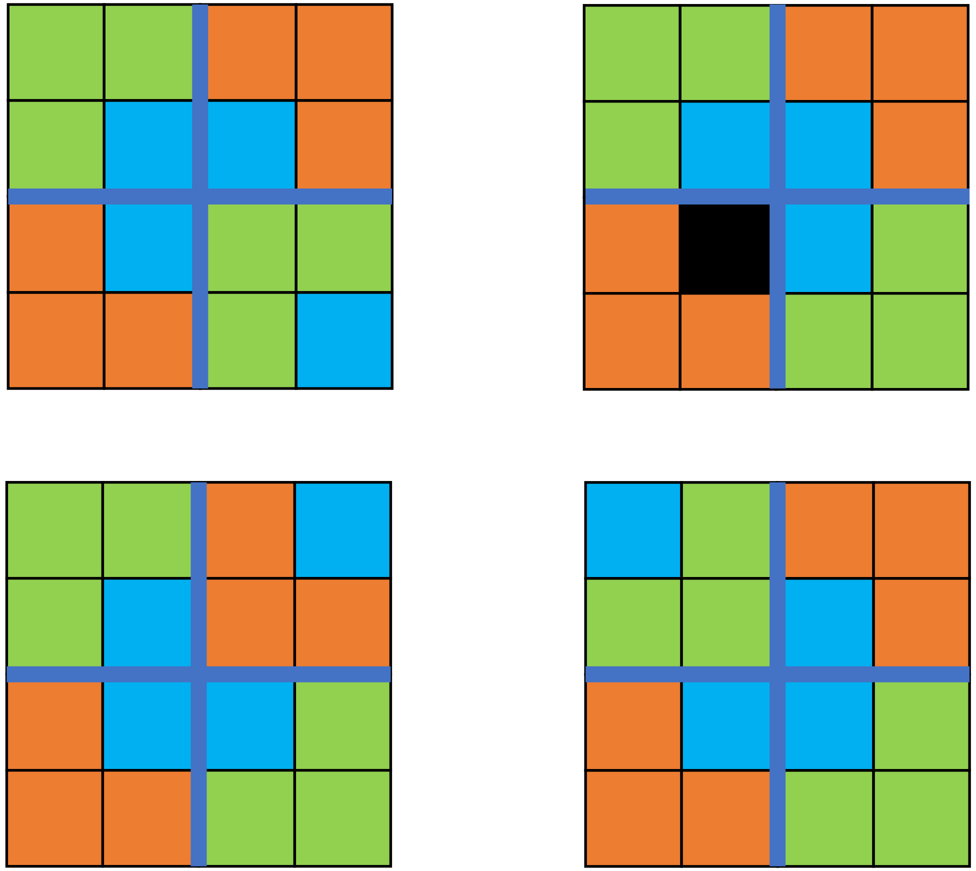
Solve **Recursively**

Trominoes Puzzle Solution



Solve **Recursively**

Trominoes Puzzle Solution



Our first algorithmic technique!

Divide and Conquer

[CLRS Chapter 4]

Divide:

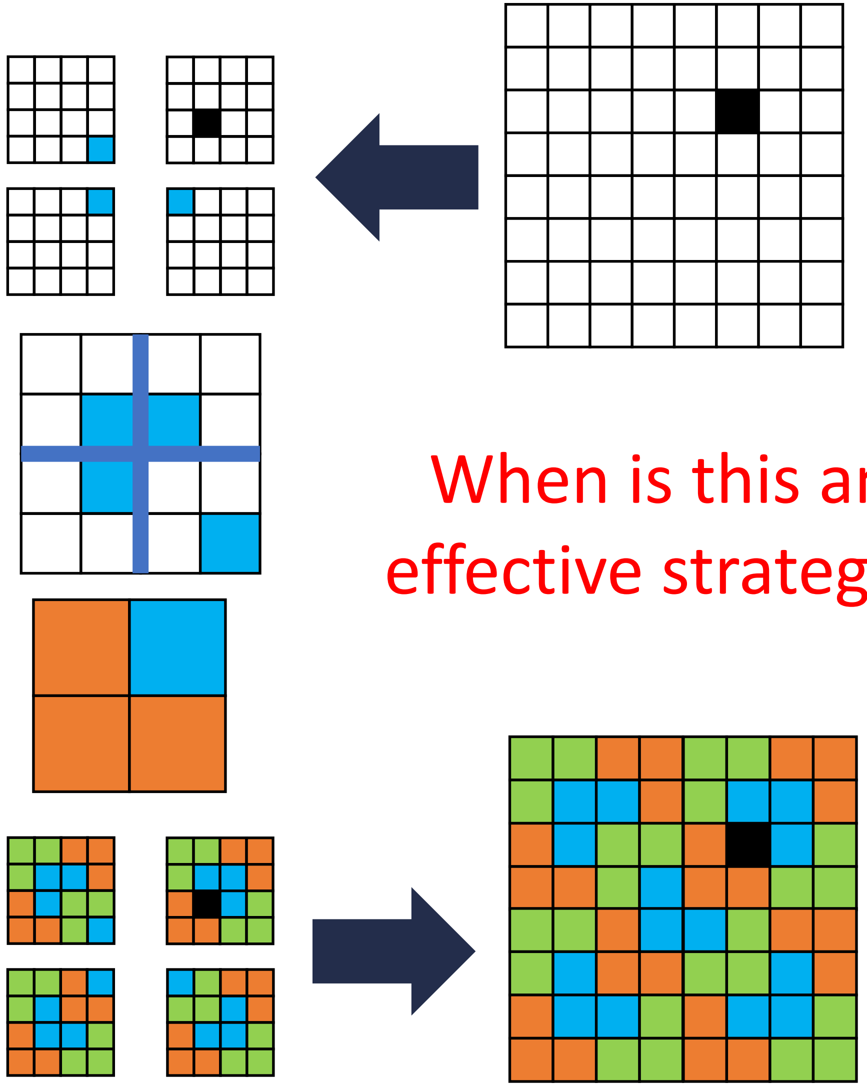
- Break the problem into multiple subproblems, each smaller instances of the original

Conquer:

- If the suproblems are "large":
  - Solve each subproblem recursively
- If the subproblems are "small":
  - Solve them directly (base case)

Combine:

- Merge solutions to subproblems to obtain solution for original problem



When is this an effective strategy?