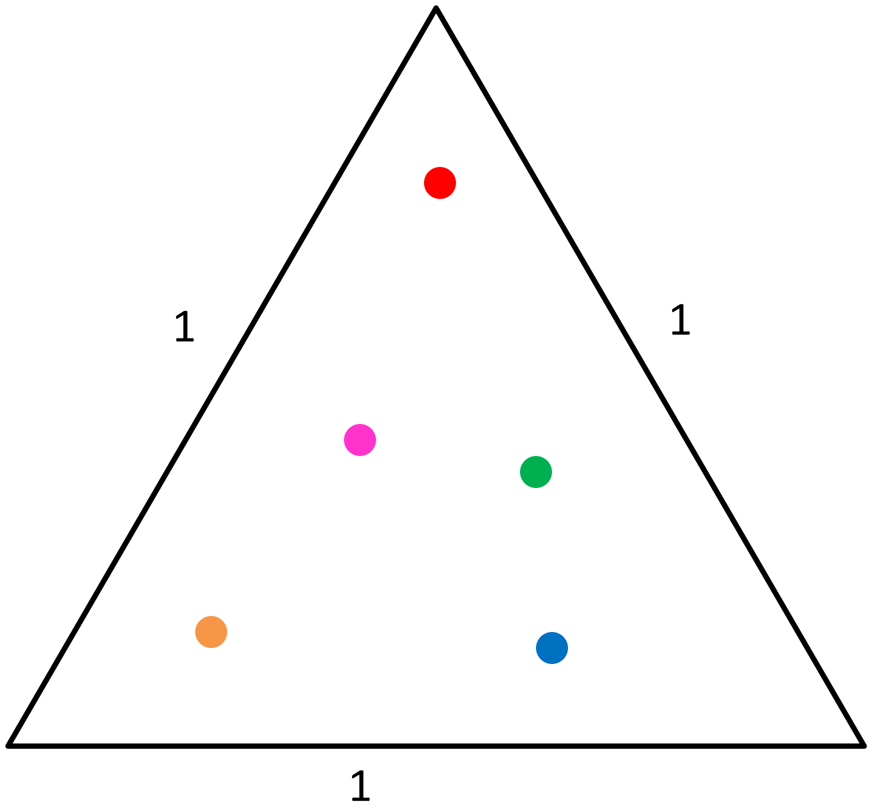
CS4102 Algorithms

Fall 2018

Warm up

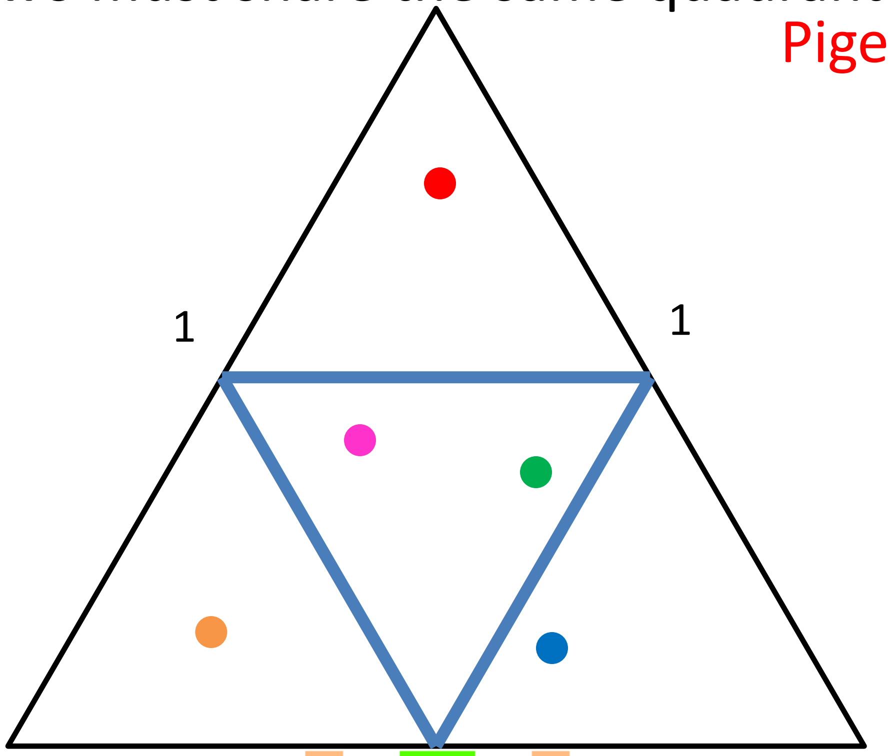
Given 5 points on the unit equilateral triangle, show there's always a pair of distance $\leq \frac{1}{2}$ apart

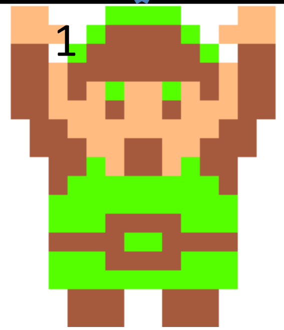


If points p_1, p_2 in same quadrant, then $\delta(p_1, p_2) \leq \frac{1}{2}$

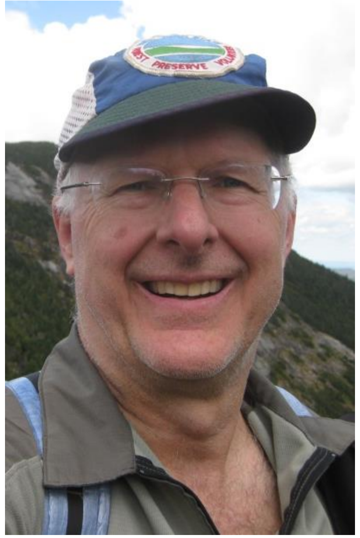
Given 5 points, two must share the same quadrant

Pigeonhole Principle!





Historical Aside: Master Theorem



Jon Bentley

Dorothea Haken

No Picture Found

James Saxe

Today's Keywords

- Substitution Method
- Divide and Conquer
- Closest Pair of Points

CLRS Readings

- Chapter 4

Homework

- Hw1 due 11pm Friday, Sept 14
  - Written (use Latex!)
  - Asymptotic notation
  - Recurrences
  - Divide and conquer
- Hw2 released TODAY
  - Programming assignment (Python or Java)
  - Divide and conquer

Master Theorem

$$T(n) = aT\left(\frac{n}{b}\right) + f(n)$$

- Case 1: if $f(n) = O(n^{\log_b a - \varepsilon})$ for some constant $\varepsilon > 0$, then $T(n) = \Theta(n^{\log_b a})$
- Case 2: if $f(n) = \Theta(n^{\log_b a})$, then $T(n) = \Theta(n^{\log_b a} \log n)$
- Case 3: if $f(n) = \Omega(n^{\log_b a + \varepsilon})$ for some constant $\varepsilon > 0$, and if $af\left(\frac{n}{b}\right) \leq cf(n)$ for some constant $c < 1$ and all sufficiently large n, then $T(n) = \Theta(f(n))$

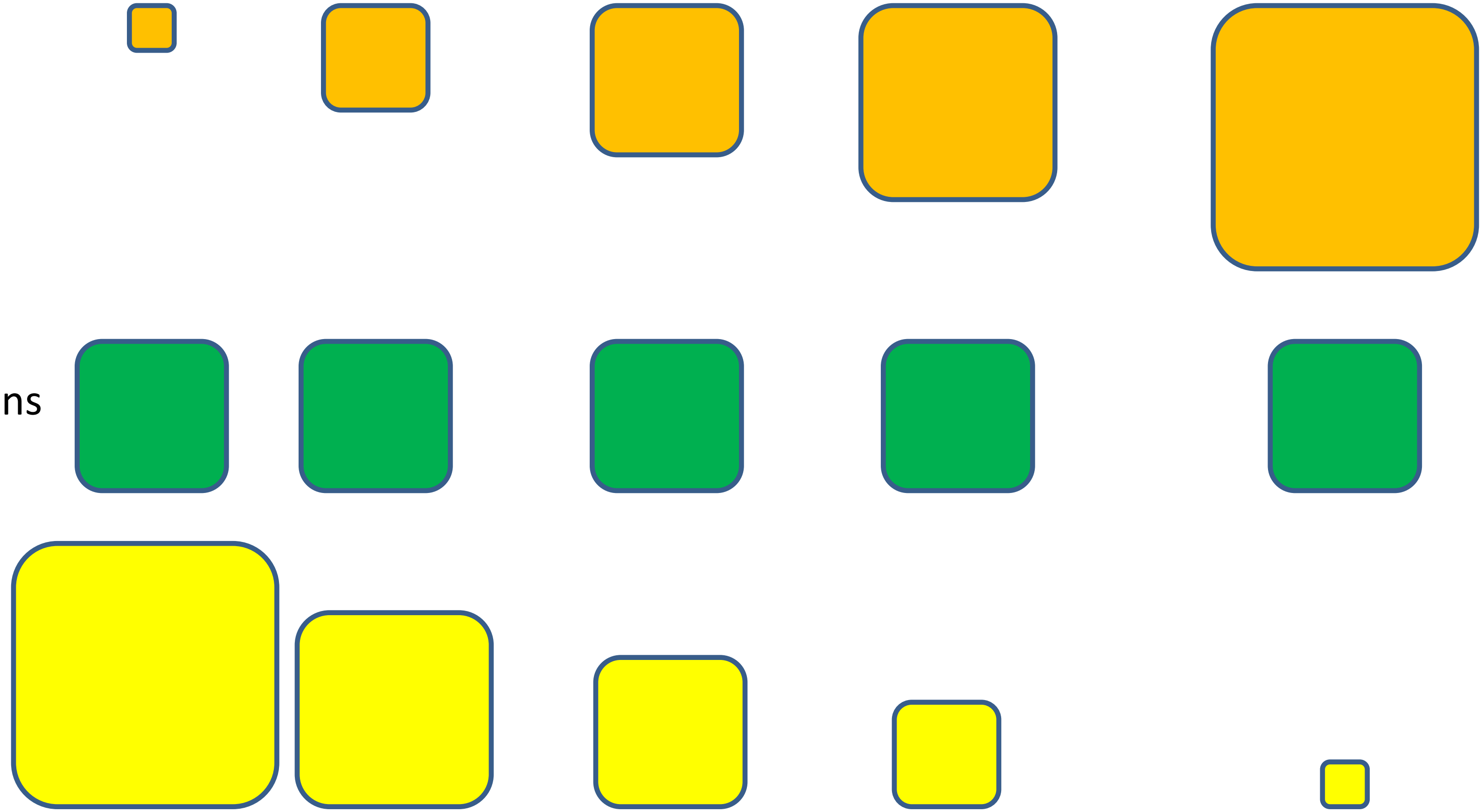
3 Cases

$$T(n) = f(n) + af\left(\frac{n}{b}\right) + a^2 f\left(\frac{n}{b^2}\right) + a^3 f\left(\frac{n}{b^3}\right) + \cdots + a^L f\left(\frac{n}{b^L}\right)$$

Case 1:
Most work
happens at
the leaves

Case 2:
Work happens
consistently
throughout

Case 3:
Most work
happens at
top of tree



Master Theorem Example 1

$T(n) = aT\left(\frac{n}{b}\right) + f(n)$

- Case 1: if $f(n) = O(n^{\log_b a - \varepsilon})$ for some constant $\varepsilon > 0$, then $T(n) = \Theta(n^{\log_b a})$
- Case 2: if $f(n) = \Theta(n^{\log_b a})$, then $T(n) = \Theta(n^{\log_b a} \log n)$
- Case 3: if $f(n) = \Omega(n^{\log_b a + \varepsilon})$ for some constant $\varepsilon > 0$, and if $af\left(\frac{n}{b}\right) \leq cf(n)$ for some constant $c < 1$ and all sufficiently large n, then $T(n) = \Theta(f(n))$

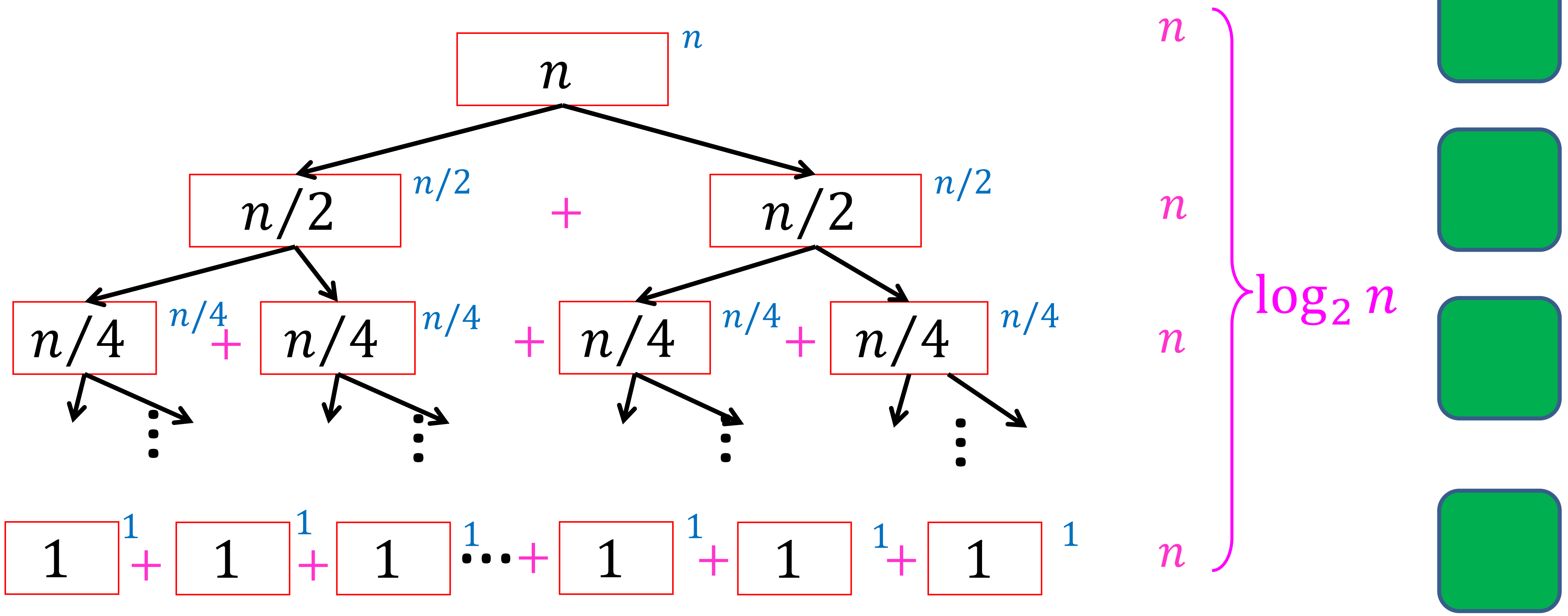
$$T(n) = 2T\left(\frac{n}{2}\right) + n$$

Case 2

$$\Theta\left(n^{\log_2 2} \log n\right) = \Theta(n \log n)$$

Tree method

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



Master Theorem Example 2

$T(n) = aT\left(\frac{n}{b}\right) + f(n)$

- Case 1: if $f(n) = O(n^{\log_b a - \varepsilon})$ for some constant $\varepsilon > 0$, then $T(n) = \Theta(n^{\log_b a})$
- Case 2: if $f(n) = \Theta(n^{\log_b a})$, then $T(n) = \Theta(n^{\log_b a} \log n)$
- Case 3: if $f(n) = \Omega(n^{\log_b a + \varepsilon})$ for some constant $\varepsilon > 0$, and if $af\left(\frac{n}{b}\right) \leq cf(n)$ for some constant $c < 1$ and all sufficiently large n, then $T(n) = \Theta(f(n))$

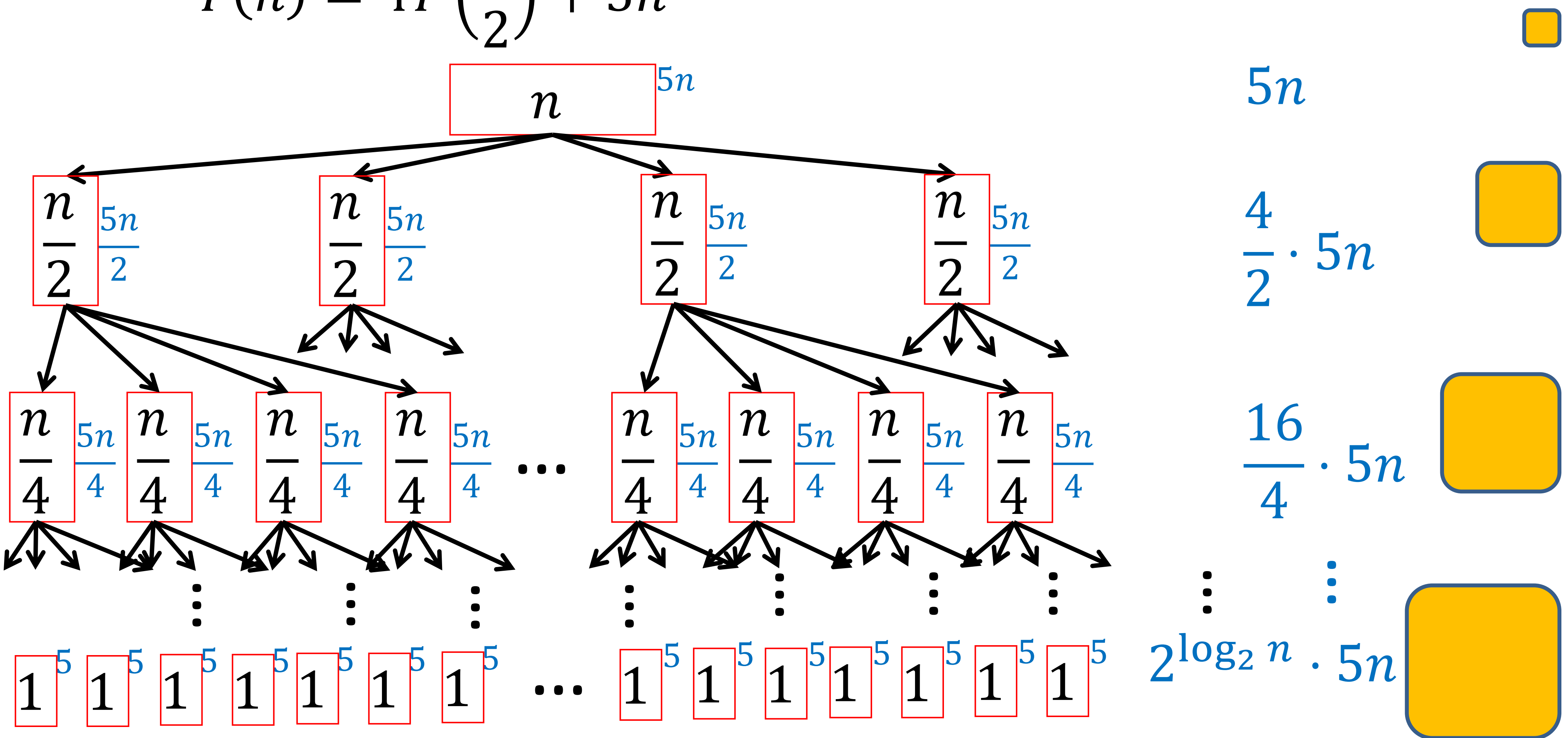
$$T(n) = 4T\left(\frac{n}{2}\right) + 5n$$

Case 1

$$\Theta\left(n^{\log_2 4}\right) = \Theta(n^2)$$

Tree method

$$T(n) = 4T\left(\frac{n}{2}\right) + 5n$$



Master Theorem Example 3

$T(n) = aT\left(\frac{n}{b}\right) + f(n)$

- Case 1: if $f(n) = O(n^{\log_b a - \varepsilon})$ for some constant $\varepsilon > 0$, then $T(n) = \Theta(n^{\log_b a})$
- Case 2: if $f(n) = \Theta(n^{\log_b a})$, then $T(n) = \Theta(n^{\log_b a} \log n)$
- Case 3: if $f(n) = \Omega(n^{\log_b a + \varepsilon})$ for some constant $\varepsilon > 0$, and if $af\left(\frac{n}{b}\right) \leq cf(n)$ for some constant $c < 1$ and all sufficiently large n, then $T(n) = \Theta(f(n))$

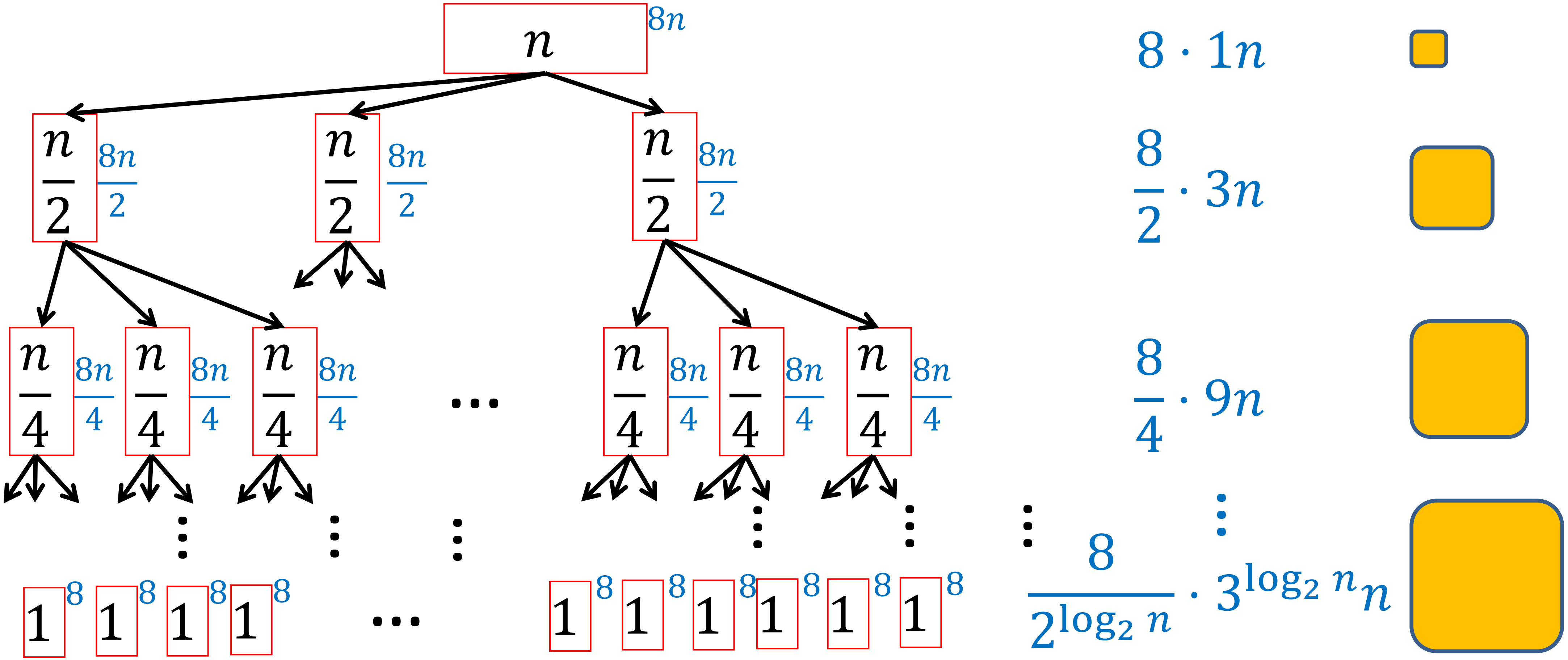
$$T(n) = 3T\left(\frac{n}{2}\right) + 8n$$

Case 1

$$\Theta\left(n^{\log_2 3}\right) \approx \Theta(n^{1.5})$$

Tree Method

$$T(n) = 3T\left(\frac{n}{2}\right) + 8n$$



Master Theorem Example 4

$T(n) = aT\left(\frac{n}{b}\right) + f(n)$

- Case 1: if $f(n) = O(n^{\log_b a - \varepsilon})$ for some constant $\varepsilon > 0$, then $T(n) = \Theta(n^{\log_b a})$
- Case 2: if $f(n) = \Theta(n^{\log_b a})$, then $T(n) = \Theta(n^{\log_b a} \log n)$
- Case 3: if $f(n) = \Omega(n^{\log_b a+\varepsilon})$ for some constant $\varepsilon > 0$, and if $af\left(\frac{n}{b}\right) \leq cf(n)$ for some constant $c < 1$ and all sufficiently large n, then $T(n) = \Theta(f(n))$

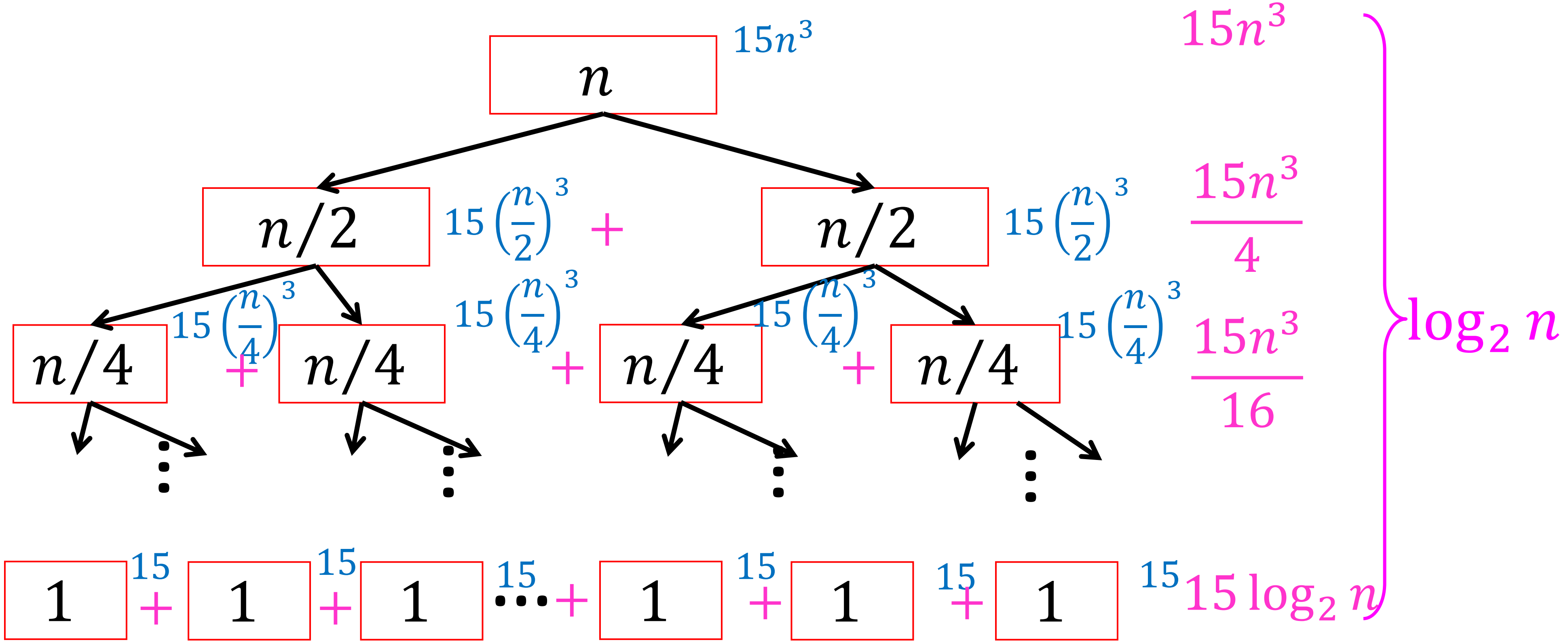
$$T(n) = 2T\left(\frac{n}{2}\right) + 15n^3$$

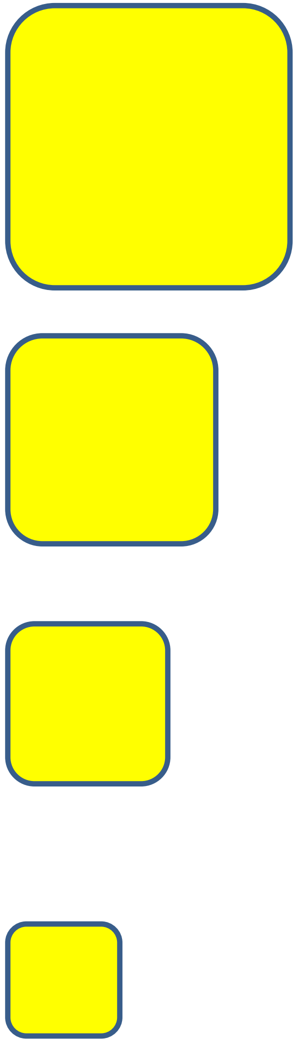
Case 3

$\Theta(n^3)$

Tree method

$$T(n) = 2T\left(\frac{n}{2}\right) + 15n^3$$





Recurrence Solving Techniques

Tree

Guess/Check

"Cookbook"

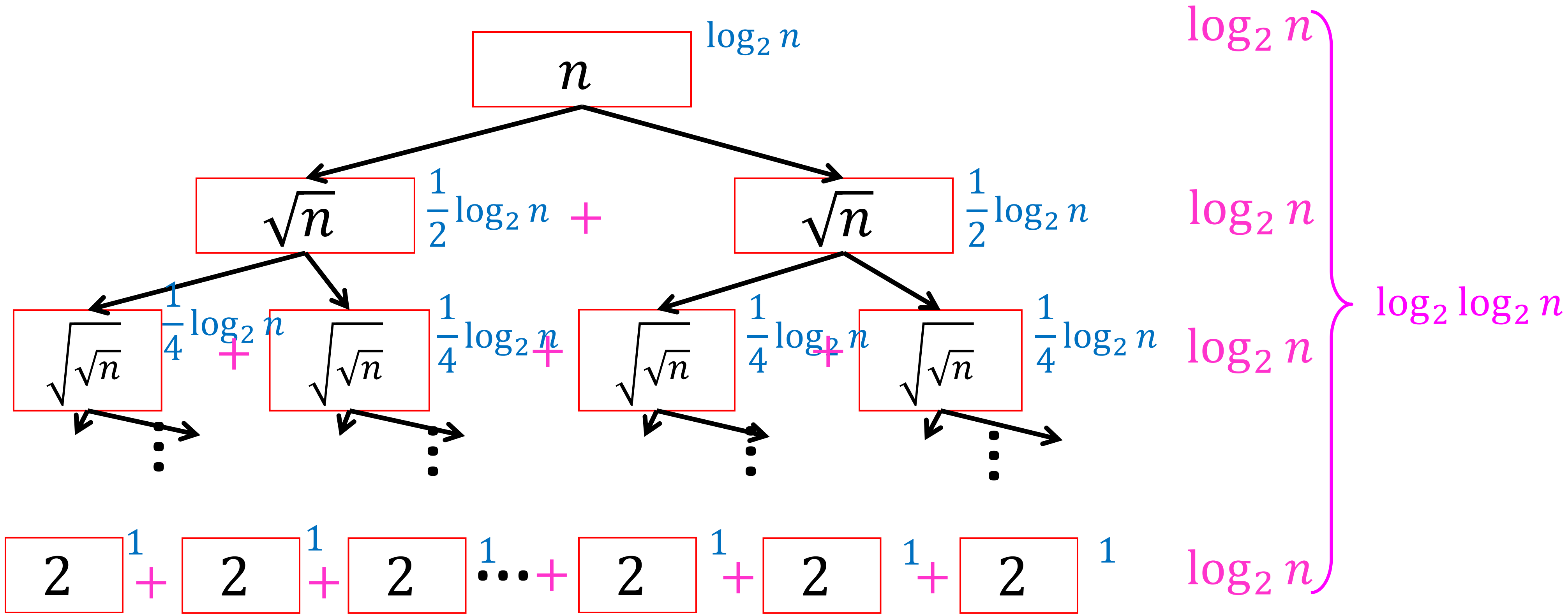
Substitution

Substitution Method

- Idea: take a "difficult" recurrence, re-express it such that one of our other methods applies.
- Example: $$T(n) = 2T(\sqrt{n}) + \log_2 n$$

Tree method

$$T(n) = 2T(\sqrt{n}) + \log_2 n$$



$$T(n) = O(\log_2 n \cdot \log_2 \log_2 n)$$

Substitution Method

- Idea: take a "difficult" recurrence, re-express it such that one of our other methods applies.
- Example:

$$T(n) = 2T(\sqrt{n}) + \log_2 n$$

Let $n = 2^m$, i.e. $m = \log_2 n$

$$T(2^m) = 2T\left(2^{\frac{m}{2}}\right) + m$$ Rewrite in terms of exponent!

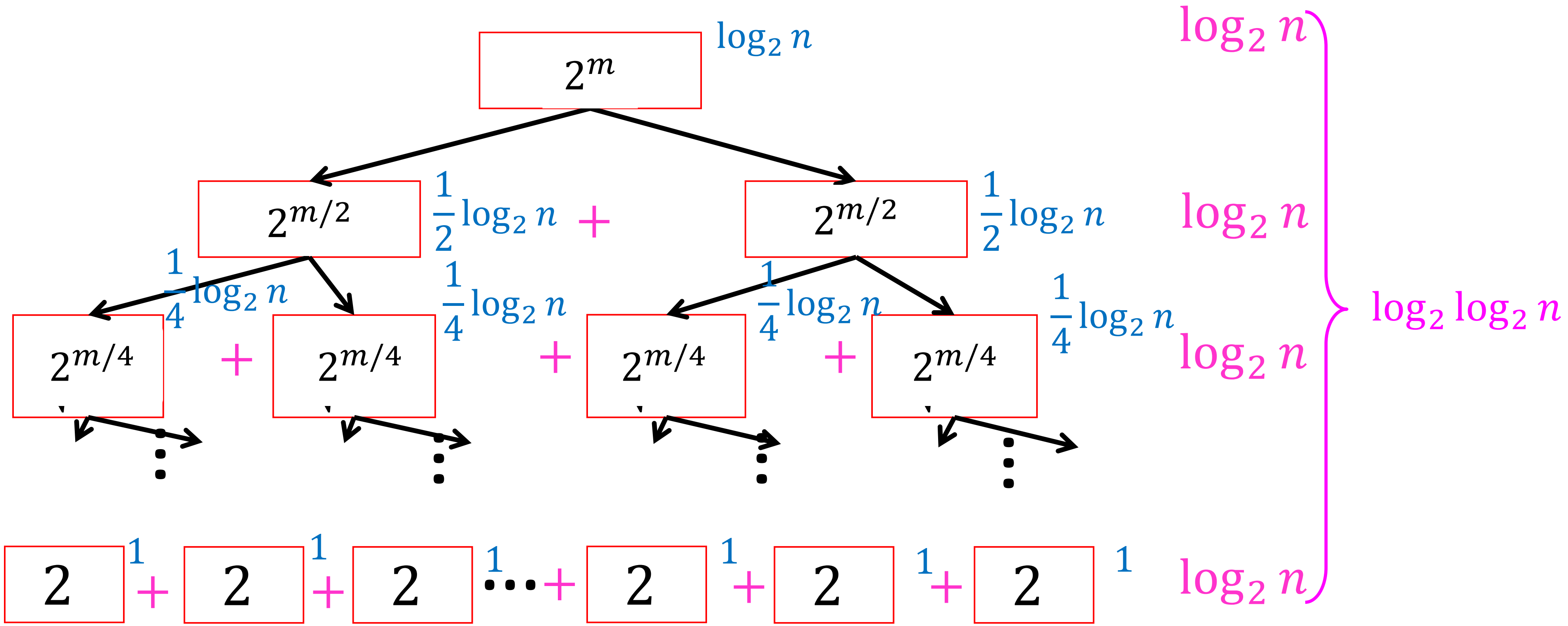
Let $S(m) = 2S\left(\frac{m}{2}\right) + m$ Case 2!

Let $S(m) = \Theta(m \log m)$ Substitute Back

Let $\mathrm{T}(n) = \Theta(\log n \log \log n)$

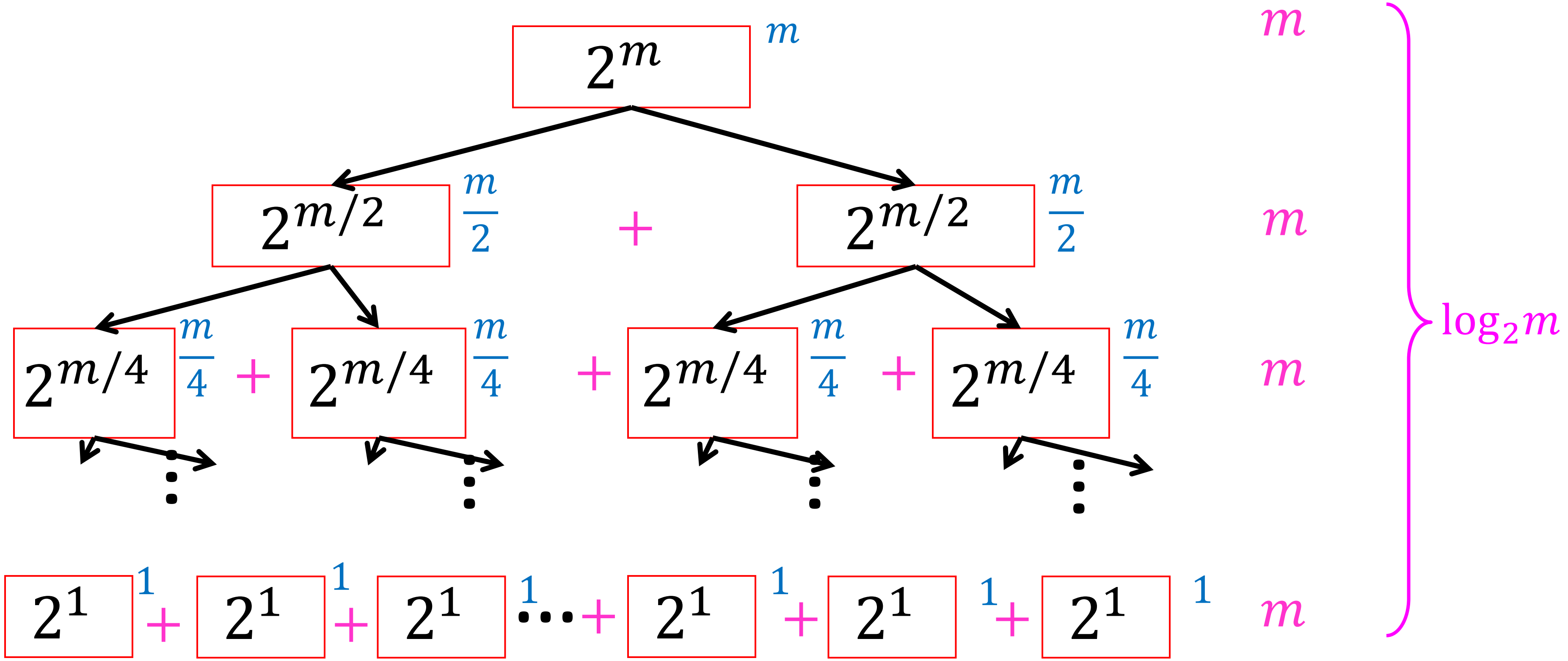
Tree method

$n = 2^m$ $\qquad$ $T(2^m) = 2T\left(2^{\frac{m}{2}}\right) + m$



Tree method

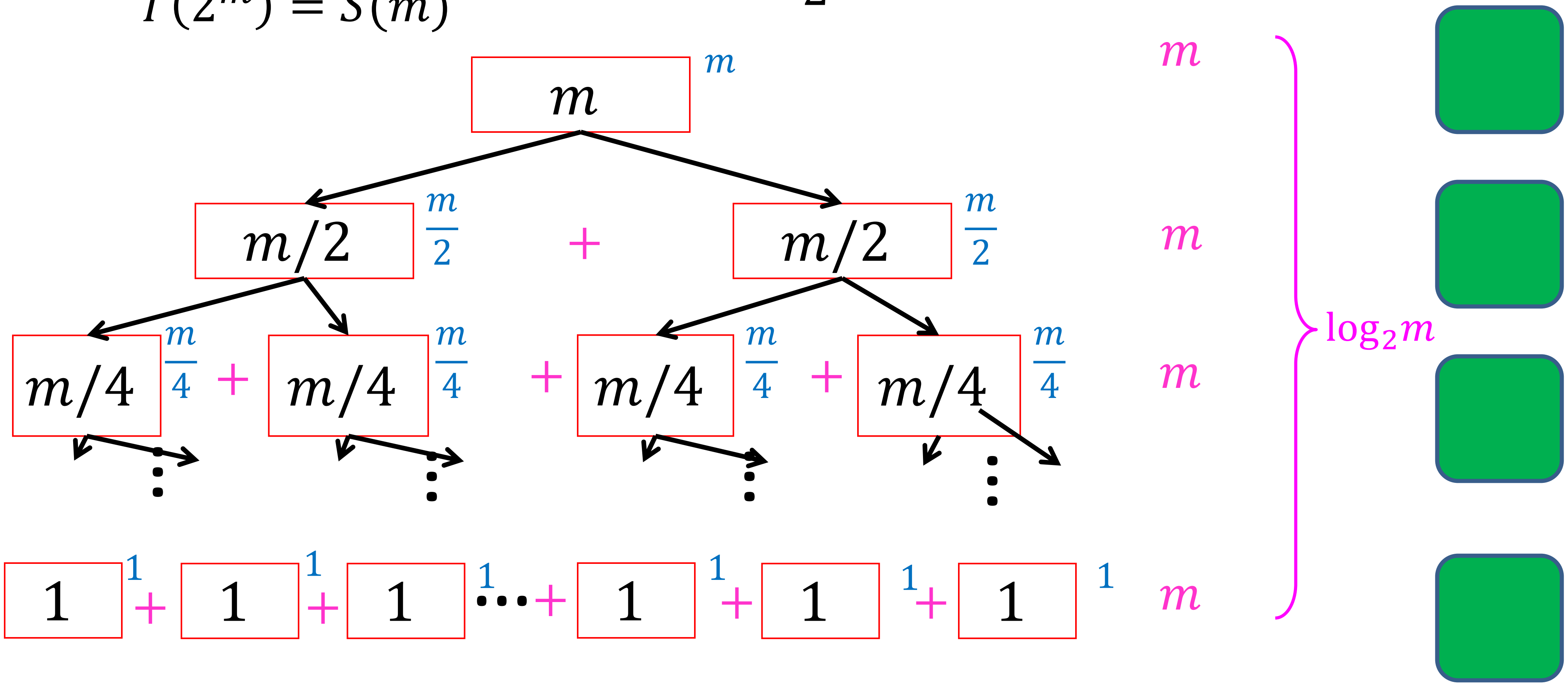
$n = 2^m$ $\qquad$ $T(2^m) = 2T(2^{m/2}) + m$



Tree method

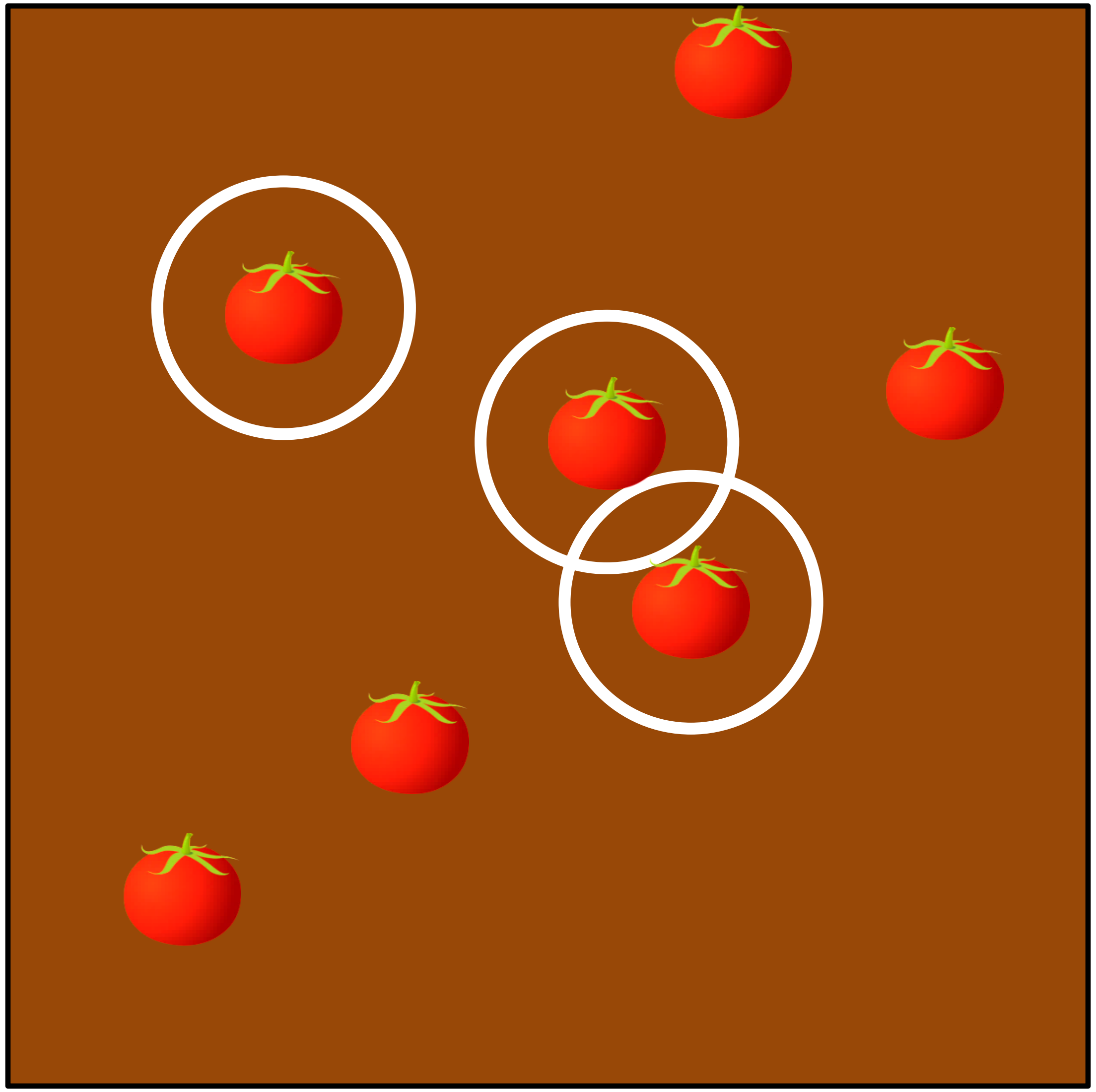
$$n = 2^m \qquad T(2^m) = S(m)$$

$$S(m) = 2S\left(\frac{m}{2}\right) + m$$



$$T(n) = O(m \cdot \log_2 m) = O(\log_2 n \cdot \log_2 \log_2 n)$$

Nate’s Garden

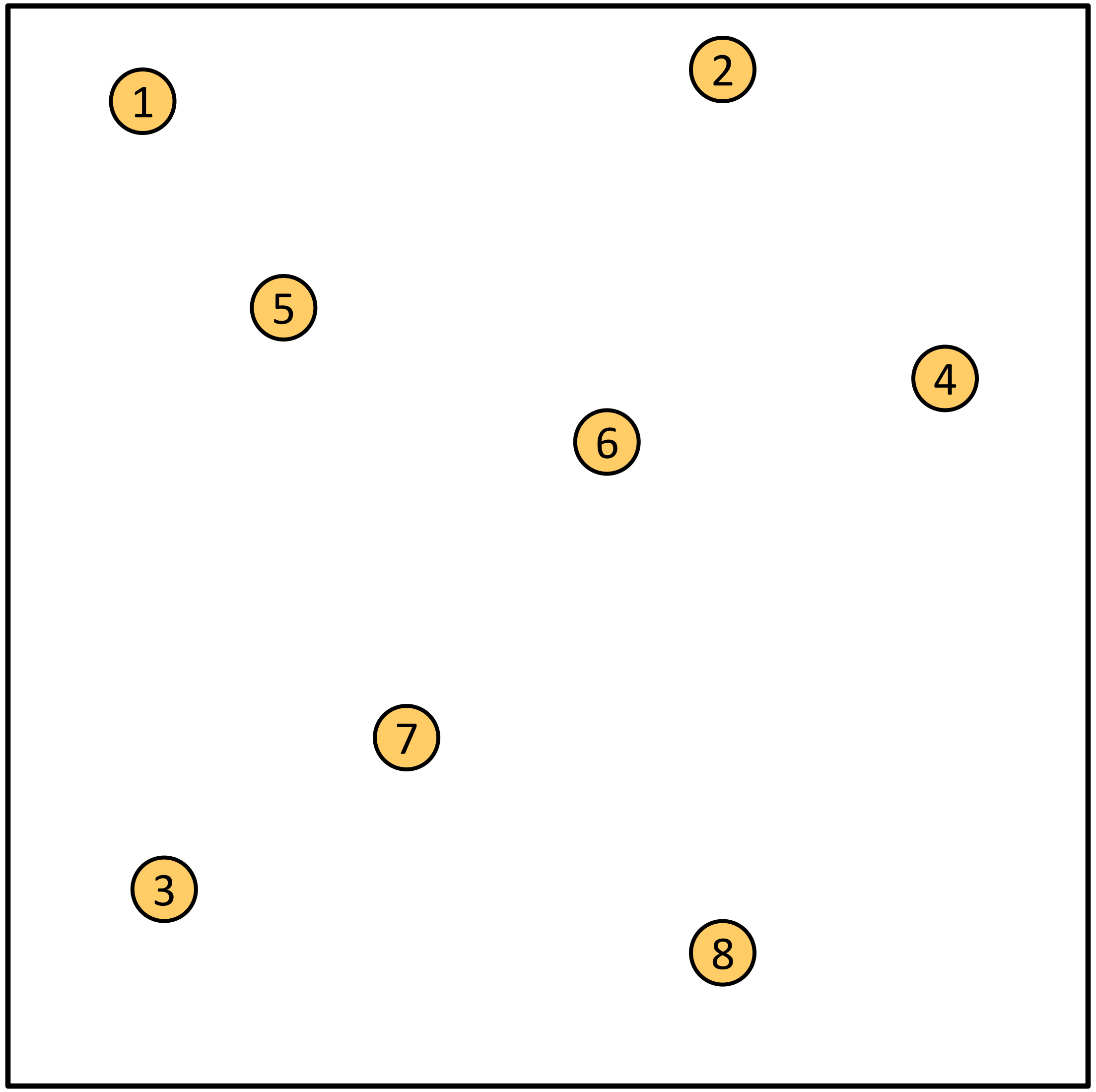


Need to find:
Closest Pair of Tomatoes

Closest Pair of Points

Given:
A list of points

Return:
Pair of points with smallest distance apart

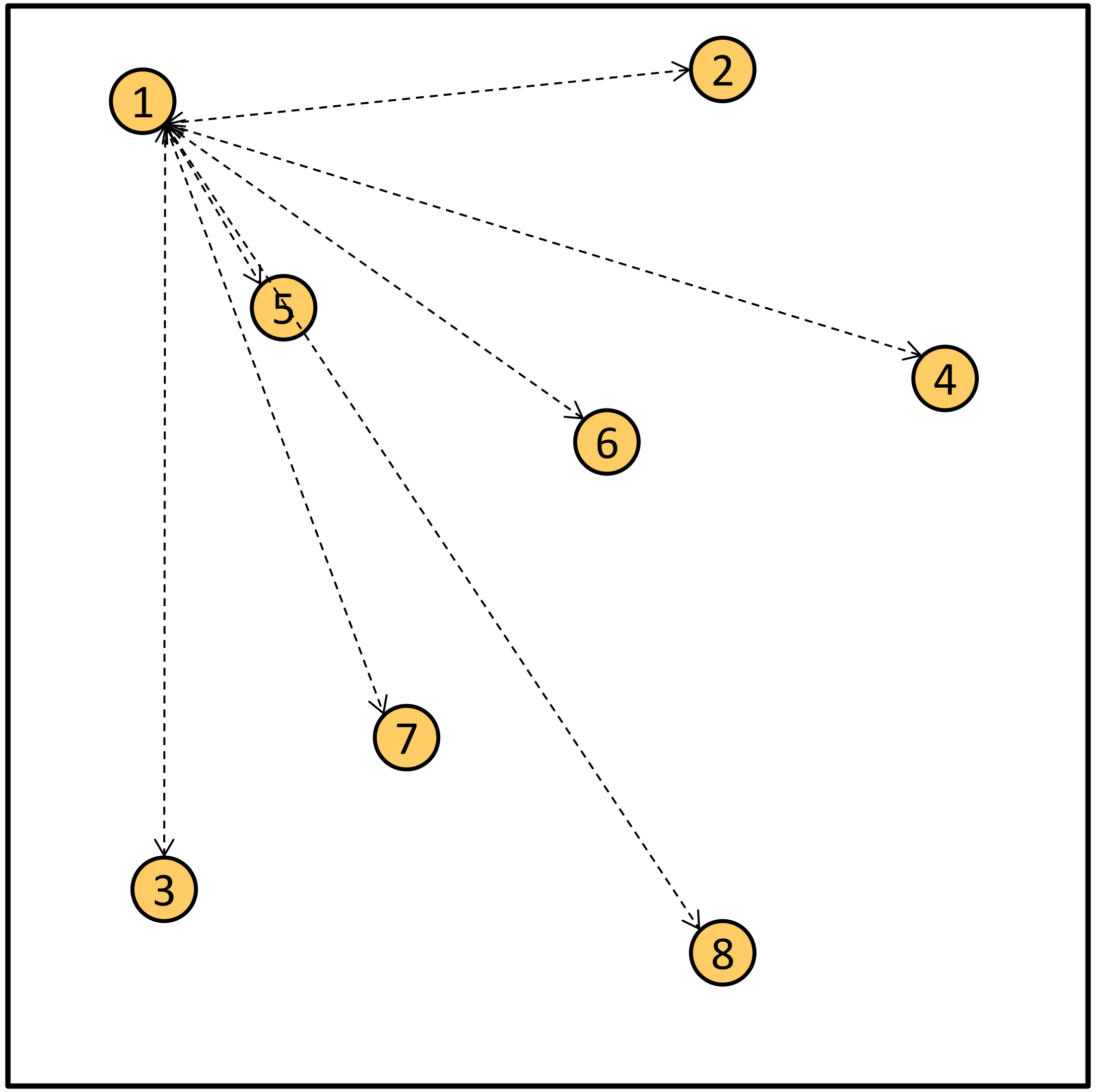


Closest Pair of Points: Naïve

Given:
A list of points

Return:
Pair of points with smallest distance apart

Algorithm: $O(n^2)$
Test every pair of points, return the closest.

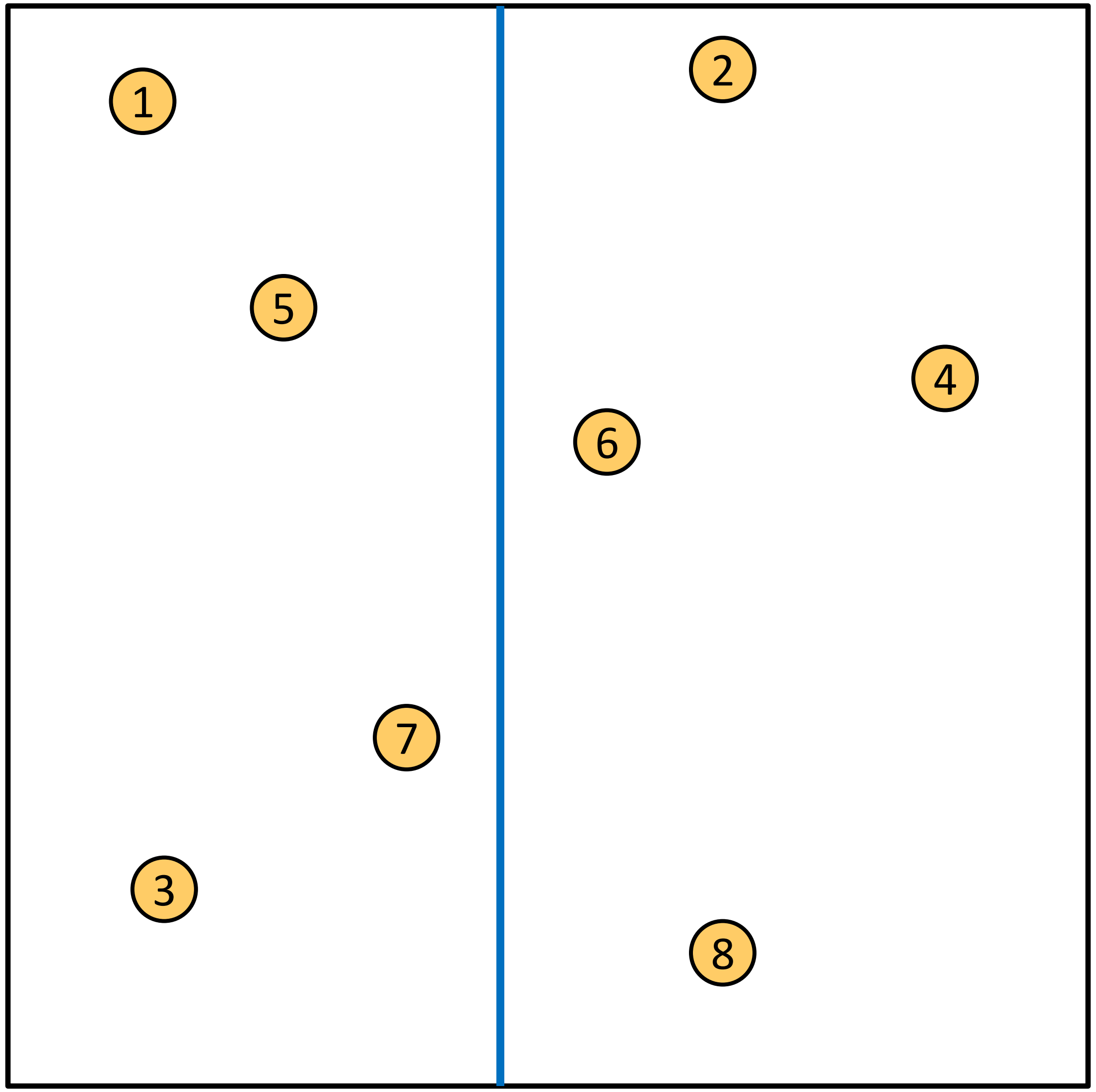


Closest Pair of Points: D&C

Divide: How?

At median x coordinate

Conquer:



Closest Pair of Points: D&C

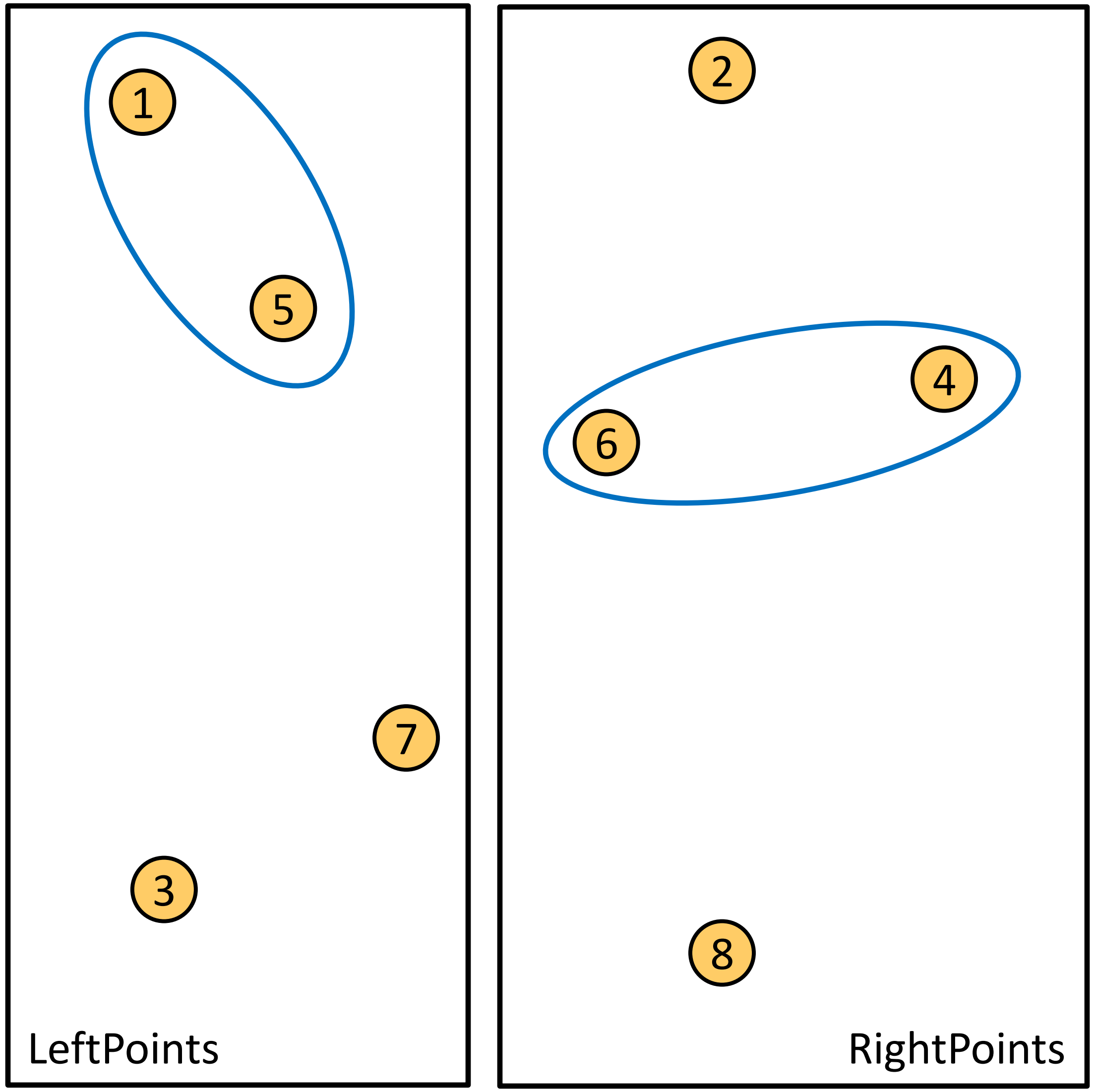
Divide:

At median x coordinate

Conquer:

Recursively find closest pairs from Left and Right

Combine:



Closest Pair of Points: D&C

Divide:

At median x coordinate

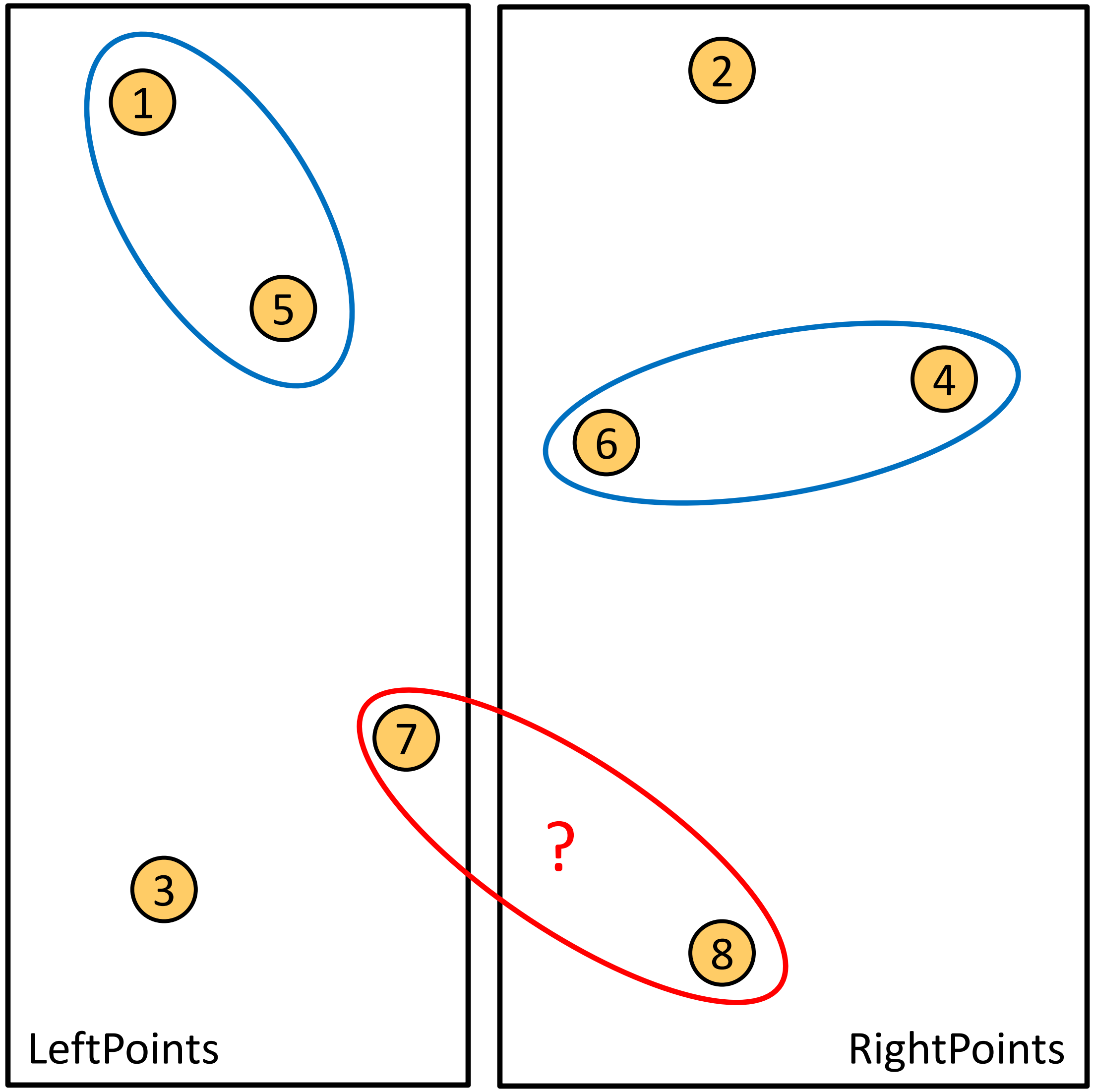
Conquer:

Recursively find closest pairs from Left and Right

Combine:

Return min of Left and Right pairs

Problem?



Closest Pair of Points: D&C

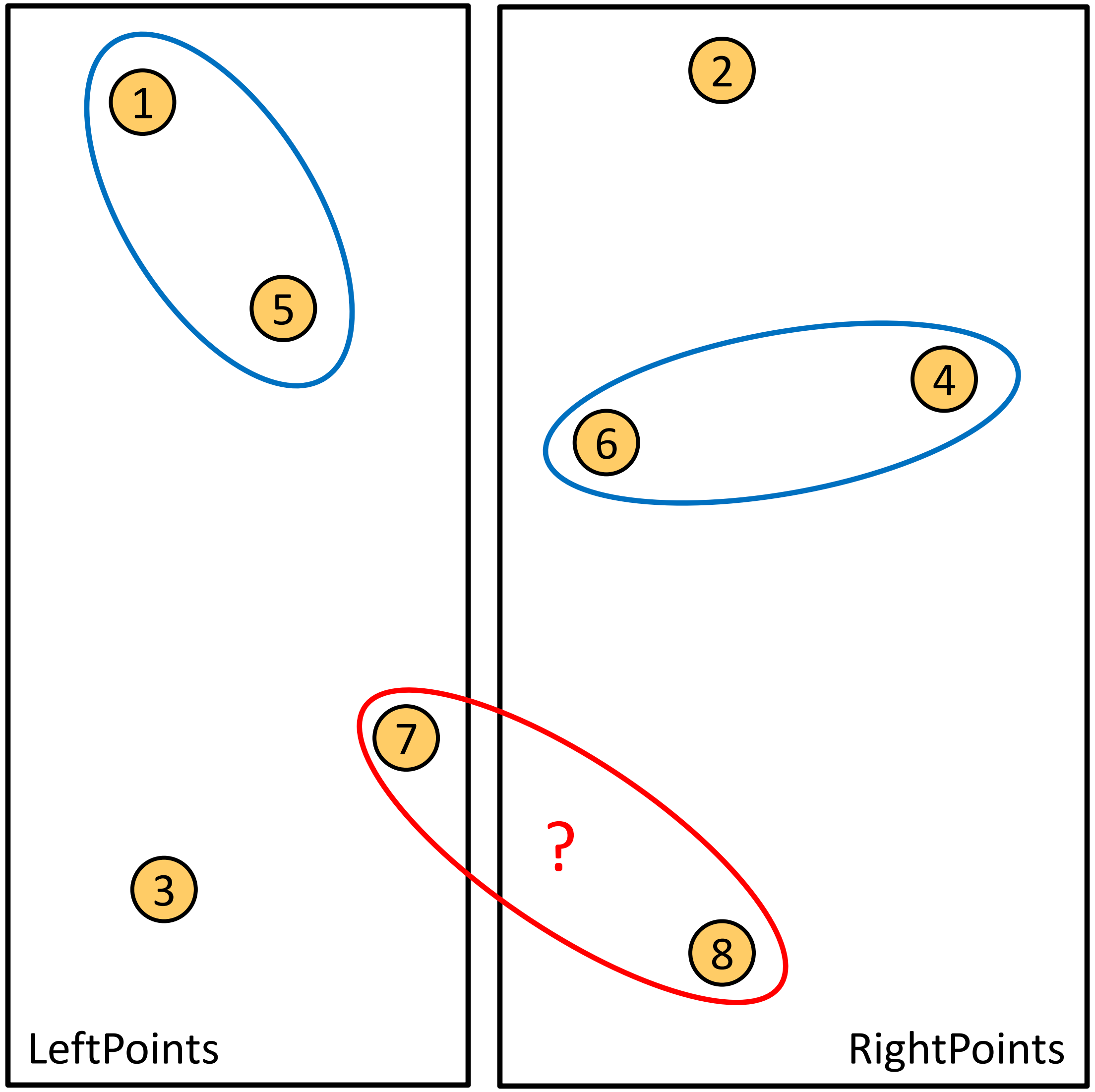
Combine:

2 Cases:

1. Closest Pair is completely in Left or Right

2. Closest Pair Spans our "Cut"

Need to test points across the cut



Spanning the Cut

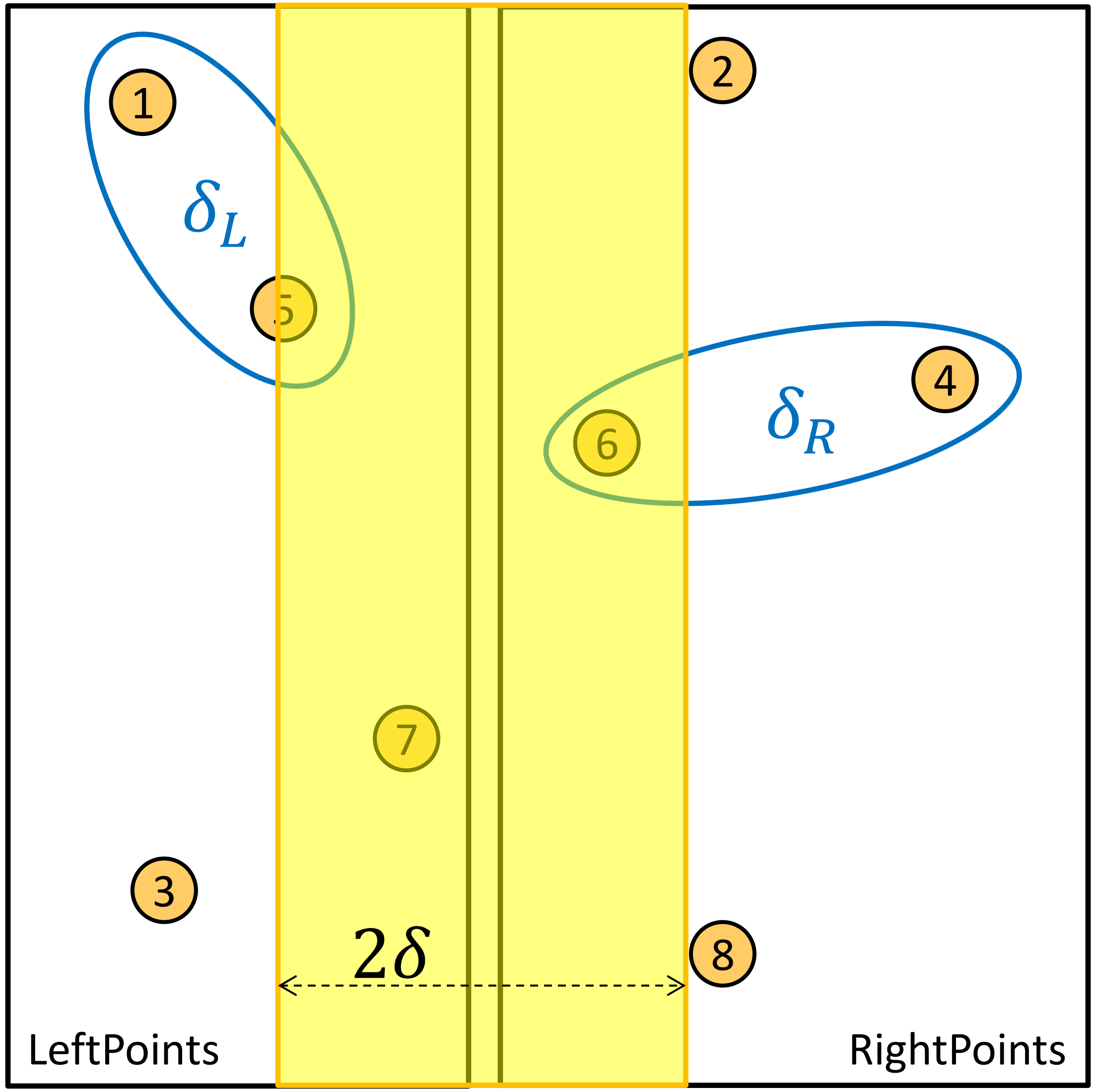
Combine:

2. Closest Pair Spanned our "Cut"

Need to test points across the cut

Compare all points within $\delta = \min\{\delta_L, \delta_R\}$ of the cut.

How many are there?



Spanning the Cut

Combine:

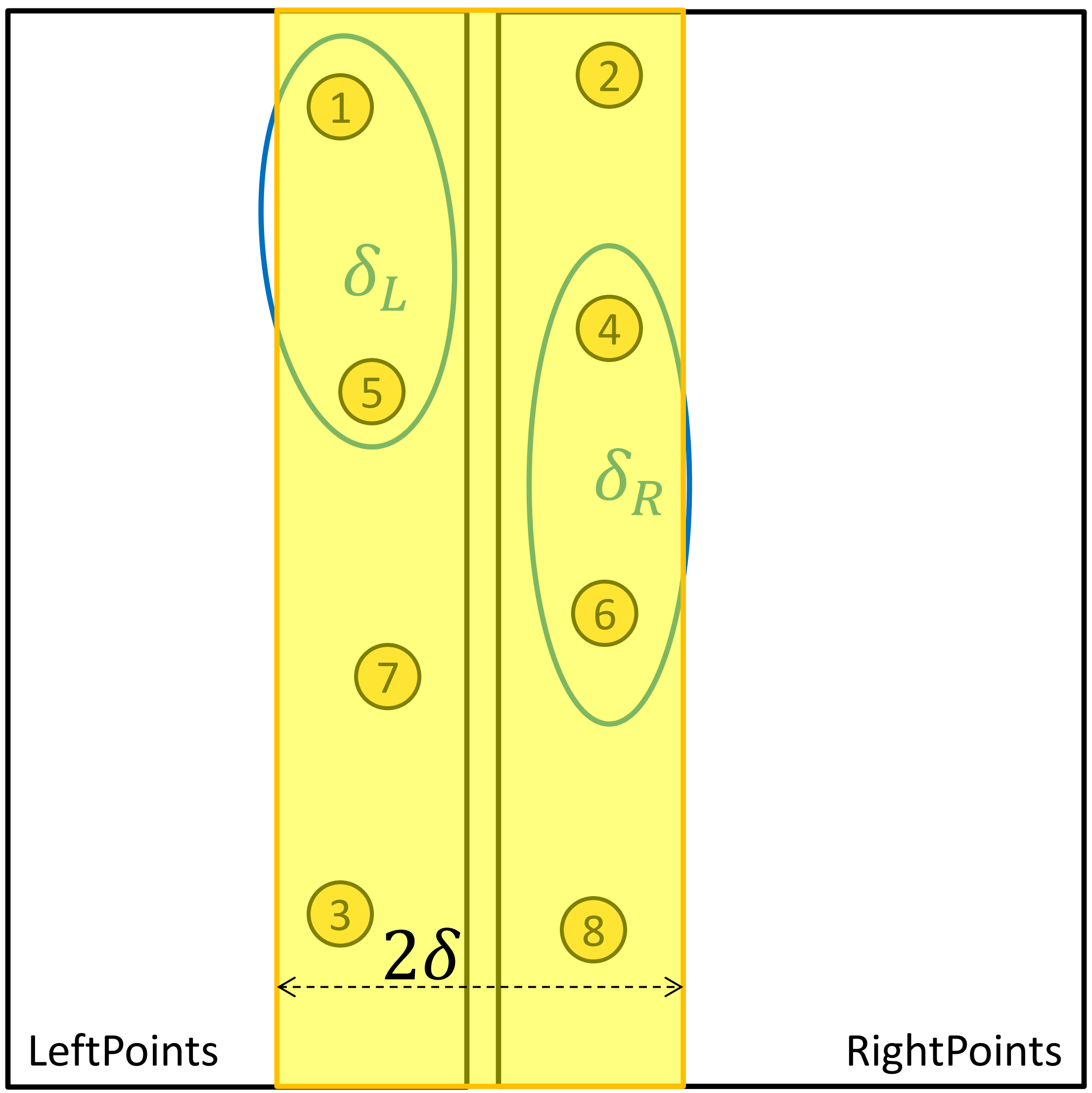
2. Closest Pair Spanned our "Cut"

Need to test points across the cut

Compare all points within $\delta = \min\{\delta_L, \delta_R\}$ of the cut.

How many are there?

$$T(n) = 2T\left(\frac{n}{2}\right) + \left(\frac{n}{2}\right)^2 = \Theta(n^2)$$



Spanning the Cut

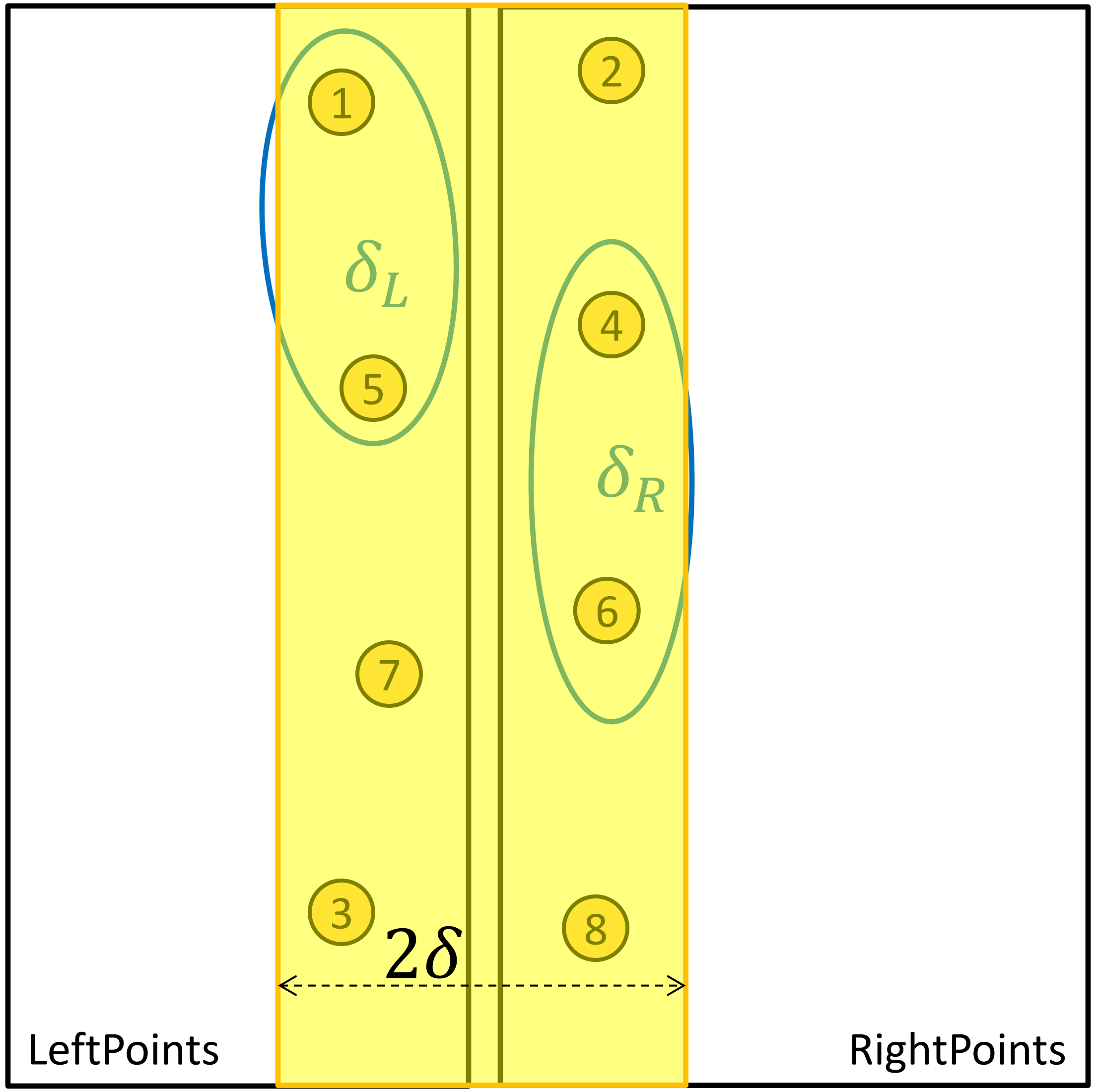
Combine:

2. Closest Pair Spanned our “Cut”

Need to test points across the cut

We don’t need to test all pairs!

Only need to test points within δ of one another



Reducing Search Space

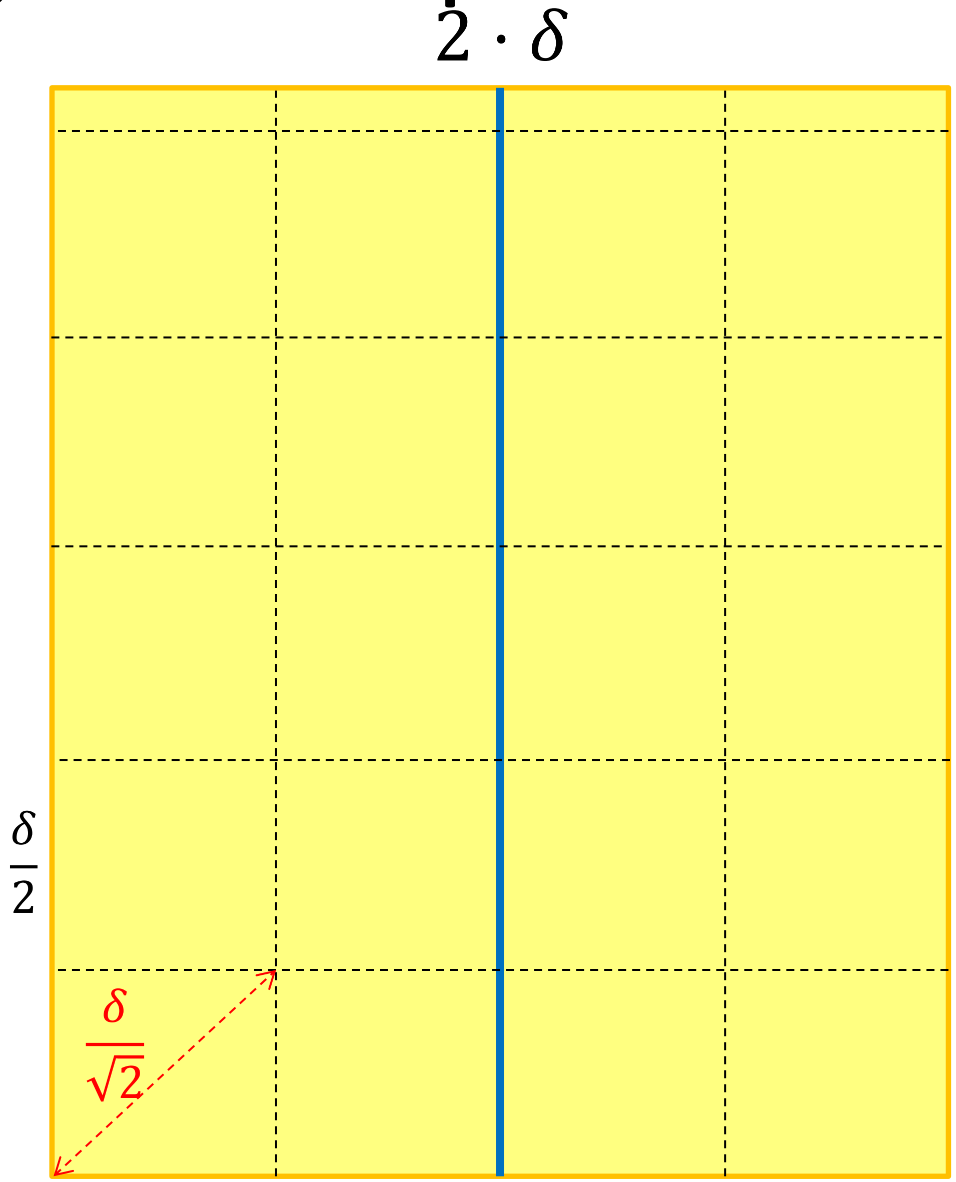
Combine:

2. Closest Pair Spanned our "Cut"

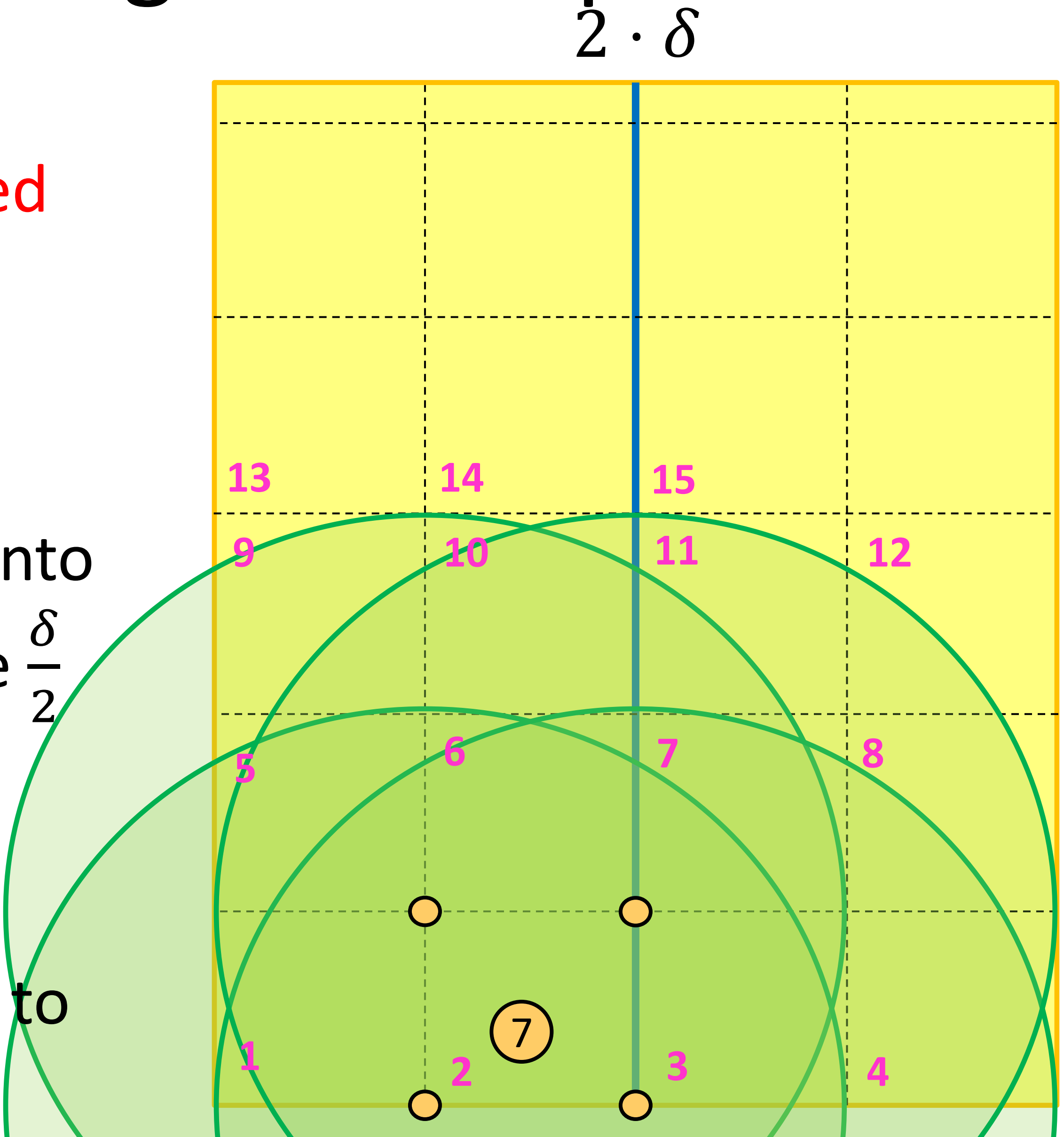
Need to test points across the cut

Divide the "runway" into square cubbies of size $\frac{\delta}{2}$

Each cubby will have at most 1 point!



Reducing Search Space



Combine:

2. Closest Pair Spanned our "Cut"

Need to test points across the cut

Divide the "runway" into square cubbies of size $\frac{\delta}{2}$

How many cubbies could contain a point $< \delta$ away?

Each point compared to ≤ 15 other points

Closest Pair of Points: D&C

0. Sort points by x

1. Divide: At median x

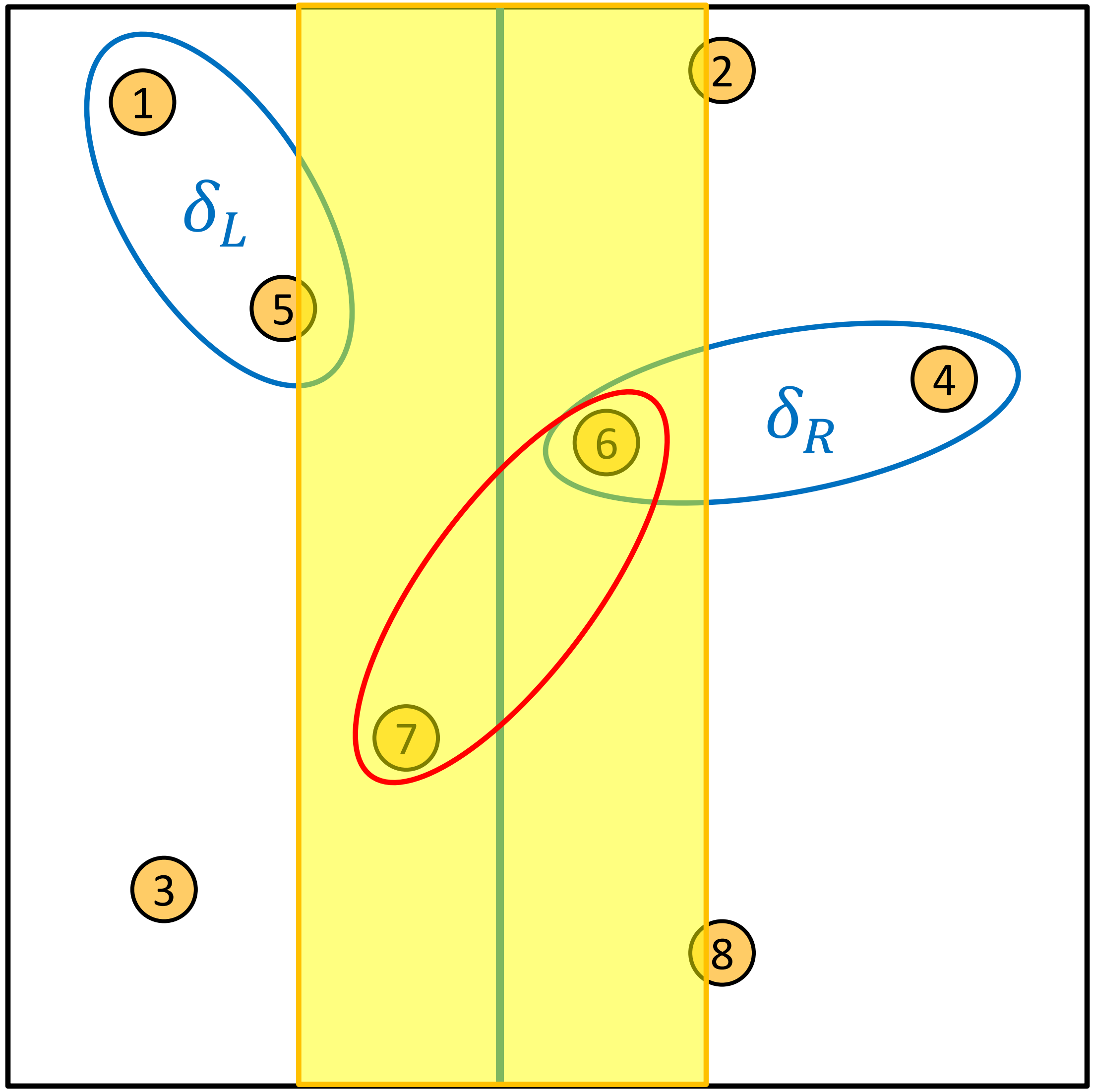
2. Conquer: If >2 points Recursively find closest pair on left and right

3. Combine:

   a. List points in "runway" in order according to y value

   b. Compare each point to the next 15 above it, save best found

   c. Return min from left, right, and 3b



Listing points in "Runway"

- Given: y-sorted lists from left and right
- Return: y-sorted points in "runway"
- Target run time? $O(n)$

Left, sorted by y

3	7	5	1

Right, sorted by y

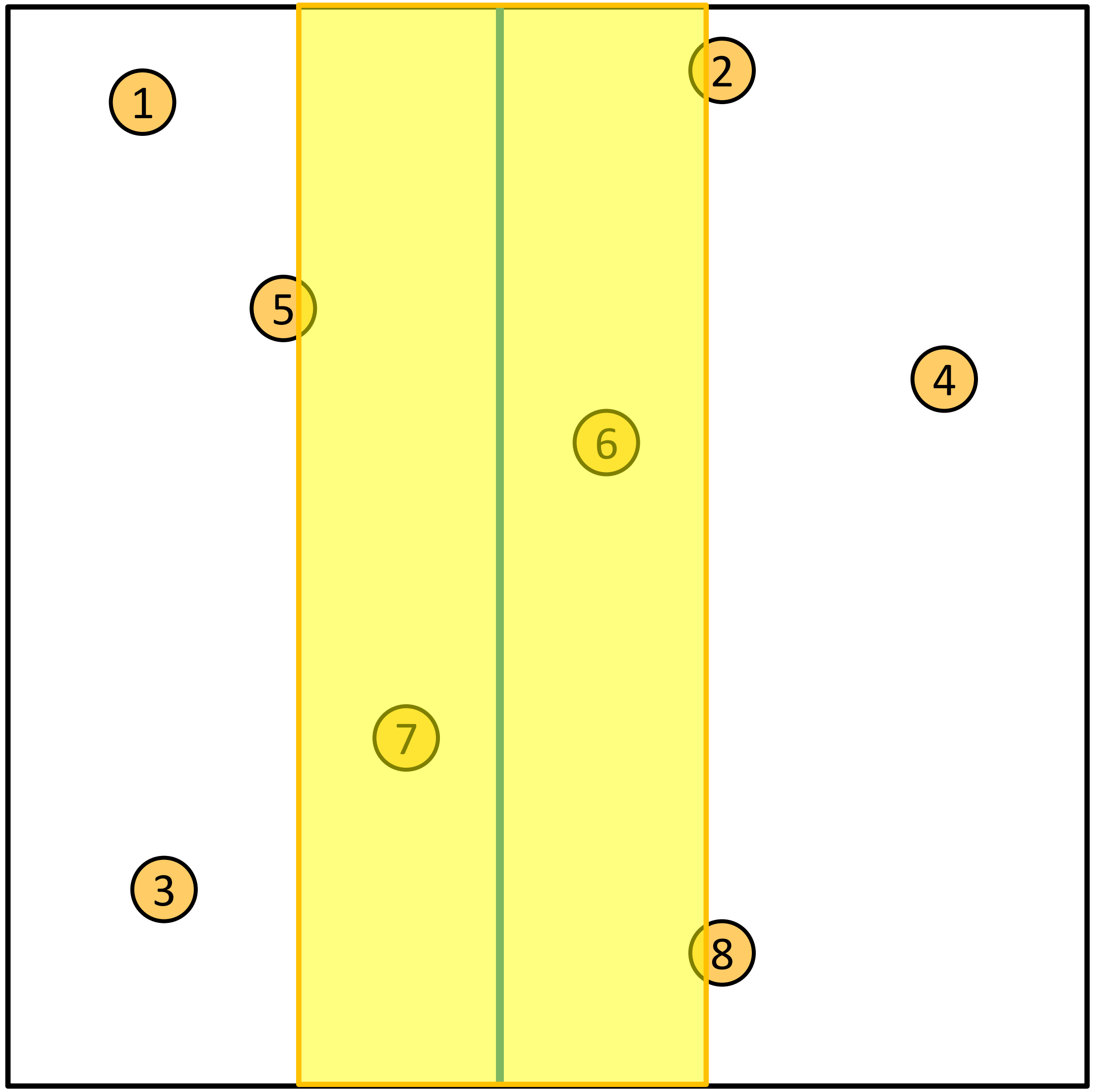
8	6	4	2

Merged, sorted by y

8	3	7	6	4	5	1	2

Runway, still sorted by y!

8	7	6	5	2



Run Time

0. Sort points by x $\Theta(n \log n)$

1. Divide: At median x $\Theta(1)$

2. Conquer: If >2 points, Recursively find closest pair on left and right $T\left(\frac{n}{2}\right)$

3. Combine:
   a. Merge points to sort by y $\Theta(n)$

   b. Compare each runway point to the next 15 runway points, save closest pair $\Theta(n)$

   c. Return y-sorted points and min from left, right, and 3b $\Theta(1)$

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

Case 2!

$$T(n) = \Theta(n \log n)$$