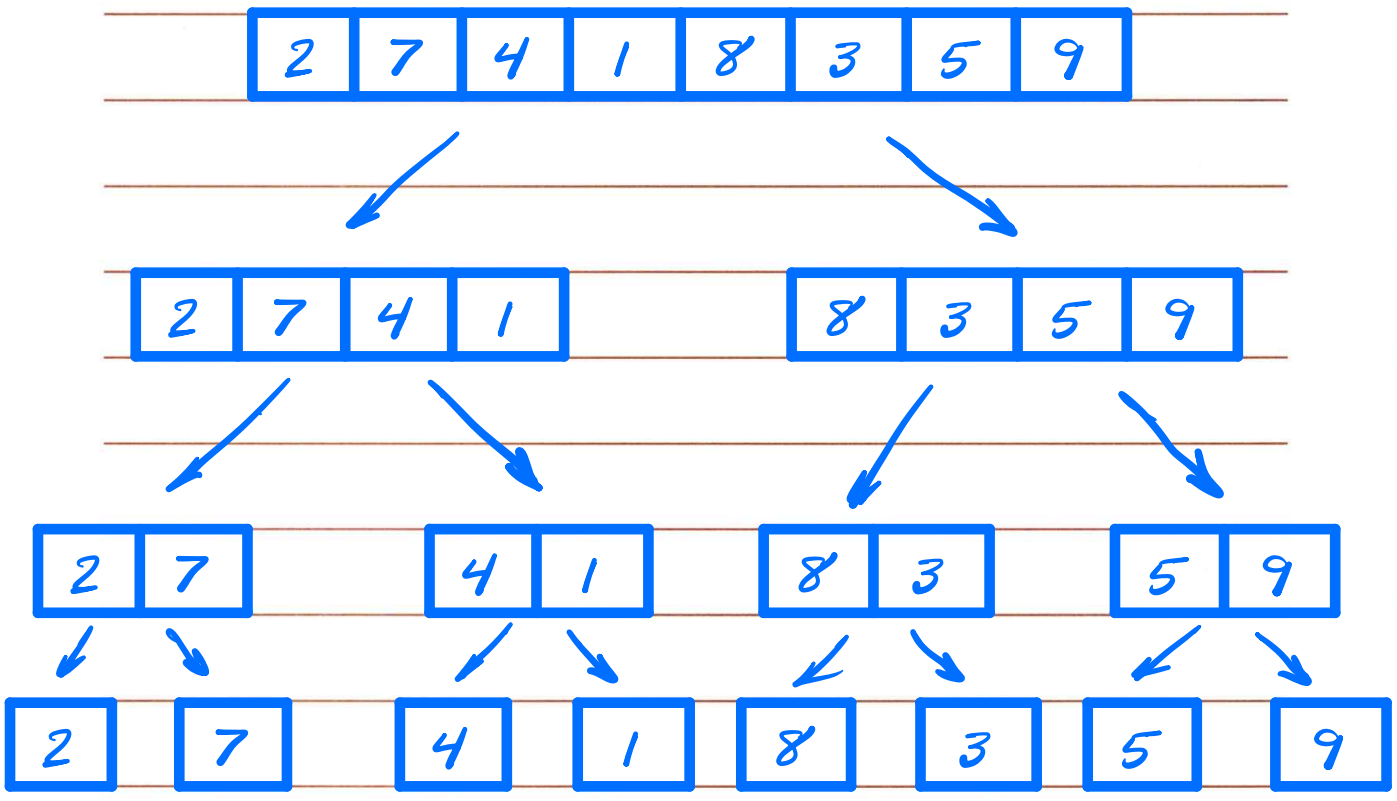
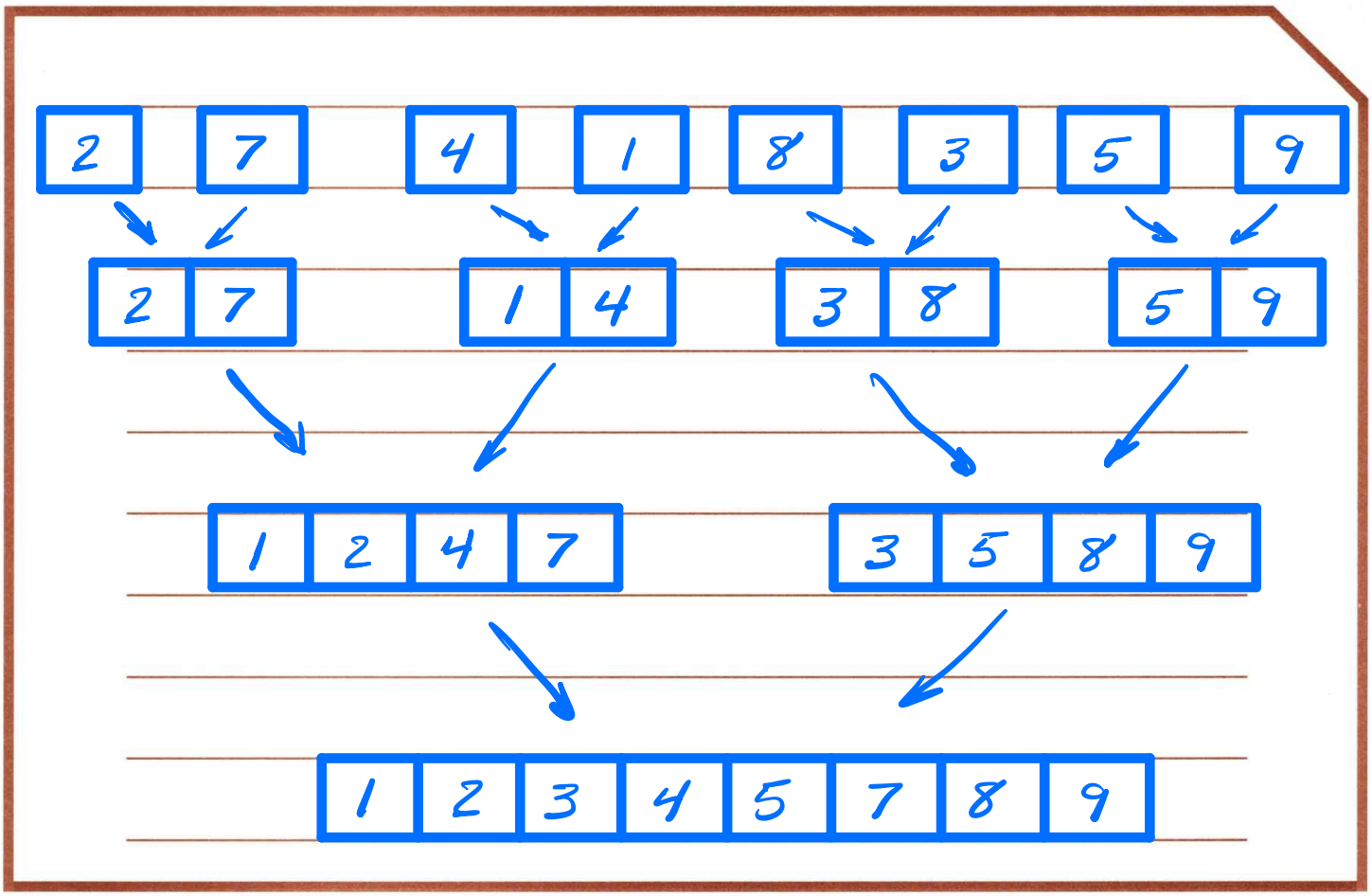
Divide & Conquer

1. Divide problem into n subproblems.

2. Conquer: i.e., solve the subproblems recursively, or if trivial solve the problem itself.

3. Combine the solutions to the subproblems.





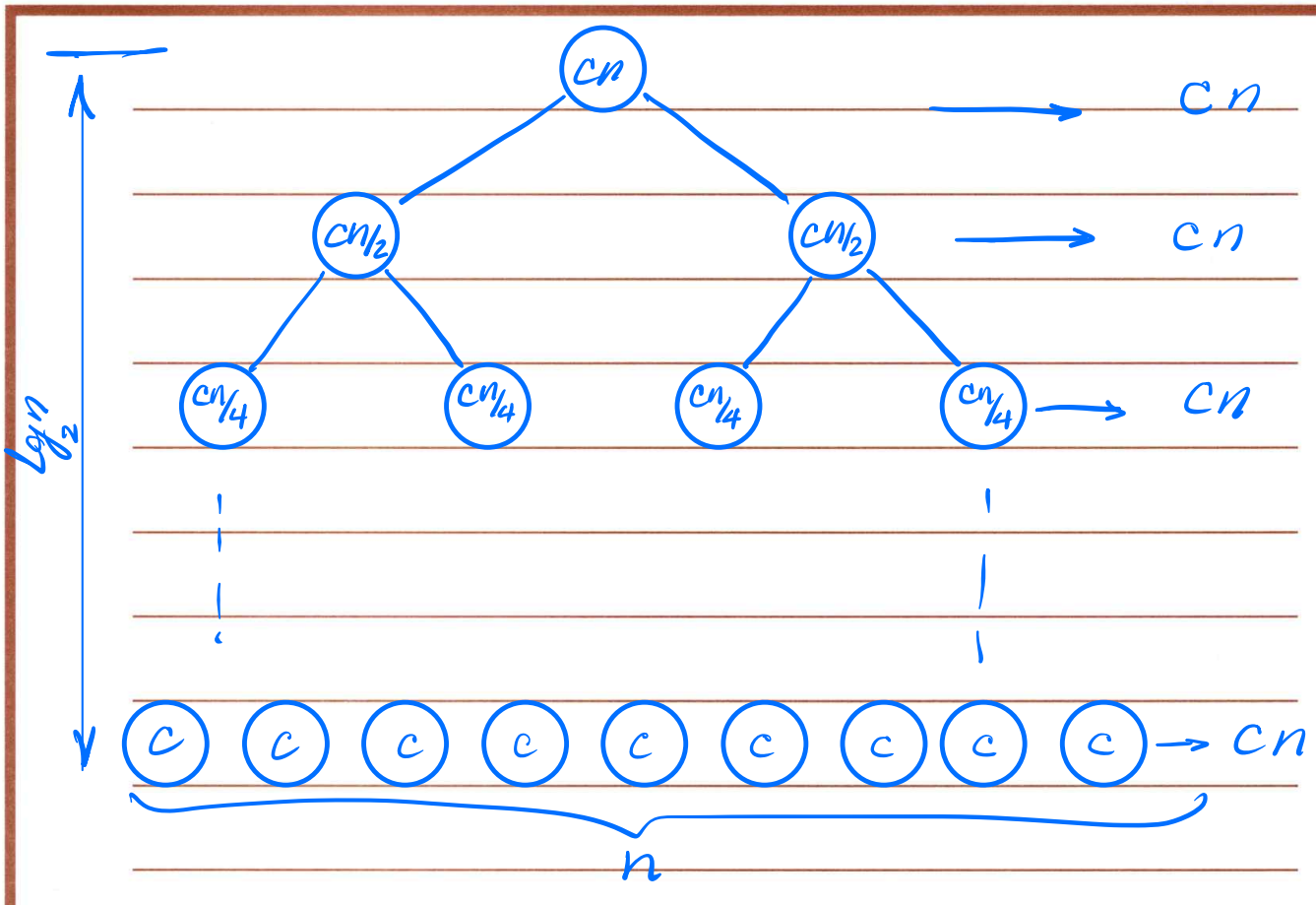

```
MergeSort(A, p, r)
    if p < r then
        q = ⌊(p+r)/2⌋
        MergeSort(A, p, q)
        MergeSort(A, q+1, r)
        Merge(A, p, q, r)
    endif
```

Runtime Analysis: (Method 1)

- Draw the recursion tree

- Determine the cost of operations at each node of the tree.

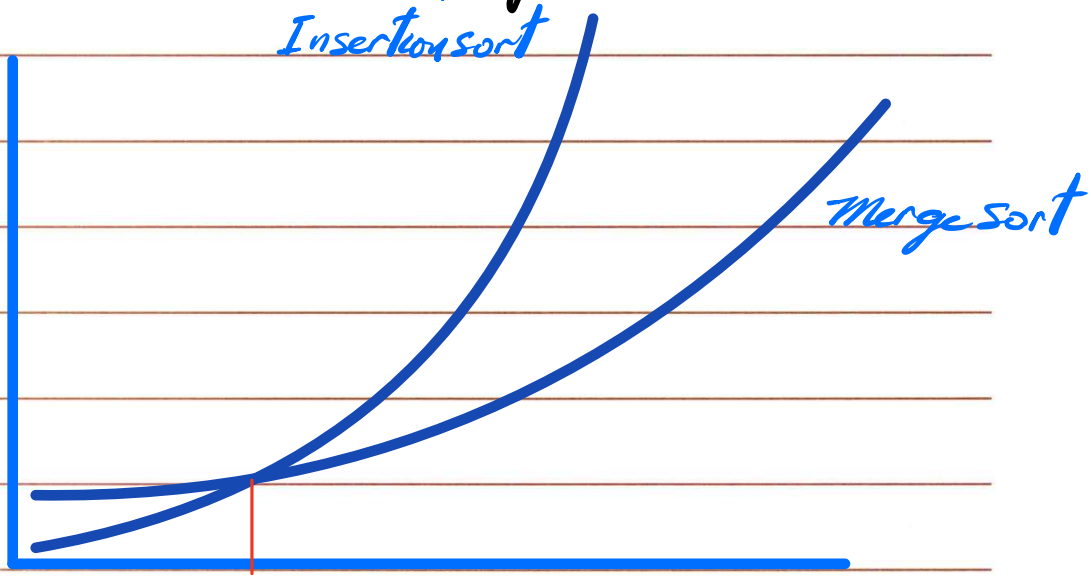
- Determine the total cost of the algorithm by adding up the costs of all the nodes of the tree.



- Each level of the recursion tree takes cn time
- There are $\log_2 n$ levels in the tree.

$$\text{Total cost} = \sum_{i=1}^{\lg n} cn = cn \lg n = \theta(n \lg n)$$

Merge sort - Insertion sort hybrid



Proof of Correctness

Let $P(n)$ be the proposition that mergesort correctly sorts any array of length n.

Base Case:

Consider an array of 1 element. This array is already sorted.

$\therefore$ Base case proven

Inductive Step:

As inductive hypothesis we assume that $P(k)$ is true for all $1 \leq k < n$. We now need to show that merge sort also works correctly on any array of size n.

Suppose we call mergesort on an array of size n. It will then recursively call mergesort on two arrays of size $n/2$. By I.H. these calls will correctly sort these arrays.

Once the two halves are correctly sorted, we will use the merge function that correctly* combines the two sorted subarrays. Therefore after the completion of the merge operation, we will have a correct sorting of the original array of size n.

*Note: In this case the POC for the combine step is trivial. If the combine step is non-trivial, it has to be proven separately.

This concludes the inductive step
Having proven the base case and inductive step we have proven that P(n) is true for all $n \geq 1$

Runtime Analysis: (Method 2)

Divide – Takes $O(1)$

Conquer – If the original problem takes $T(n)$ time, the two subproblems take $2 \cdot T(n/2)$

Combine – Takes $O(n)$ on array of size n

Recurrence Eq. for merge sort:

$$T(n) = \begin{cases} \Theta(1) & \text{if } n = 1 \\ 2 \cdot T(n/2) + \Theta(n) + \Theta(1) & \end{cases}$$

Conquer → $2 \cdot T(n/2)$

Combine → $\Theta(n)$

Divide → $\Theta(1)$

Our recurrence equation for a D&C solution will (usually) look like:

$$T(n) = \begin{cases} \Theta(1) & \text{if } n \leq c \\ a \cdot T(n/b) + D(n) + C(n) & \end{cases}$$

- a: # of subproblems
- n/b: size of each subproblem
- $D(n)$: time to divide
- $C(n)$: time to combine

Master Method

It is a cookbook method for solving recurrences of the form

$$T(n) = a\,T(n/b) + f(n)$$

- where $a \geq 1$, $b \geq 1$ are constants
- $f(n)$ is an asymptotically positive function

Master Theorem

Given the above definition of the recurrence relation, $T(n)$ can be bounded asymptotically as follows:

1- If $f(n) = O(n^{\log_b a - \varepsilon})$ for some $\varepsilon > 0$, then $T(n) = \Theta(n^{\log_b a})$

2- If $f(n) = \Theta(n^{\log_b a})$ then $T(n) = \Theta(n^{\log_b a} \lg n)$

3- If $f(n) = \Omega(n^{\log_b a + \varepsilon})$ for some $\varepsilon > 0$, and if $a f(n/b) \leq c f(n)$ for some constant $c < 1$ and all sufficiently large n, then $T(n) = \Theta(f(n))$

Ex. $\begin{cases} n^{\log_b a} = n^{\prime} \\ f(n) = n^{\prime} \lg n \end{cases}$

Does this fall into case #3?

No! $f(n)$ does grow faster than $n^{\log_b a}$, but only logarithmically.

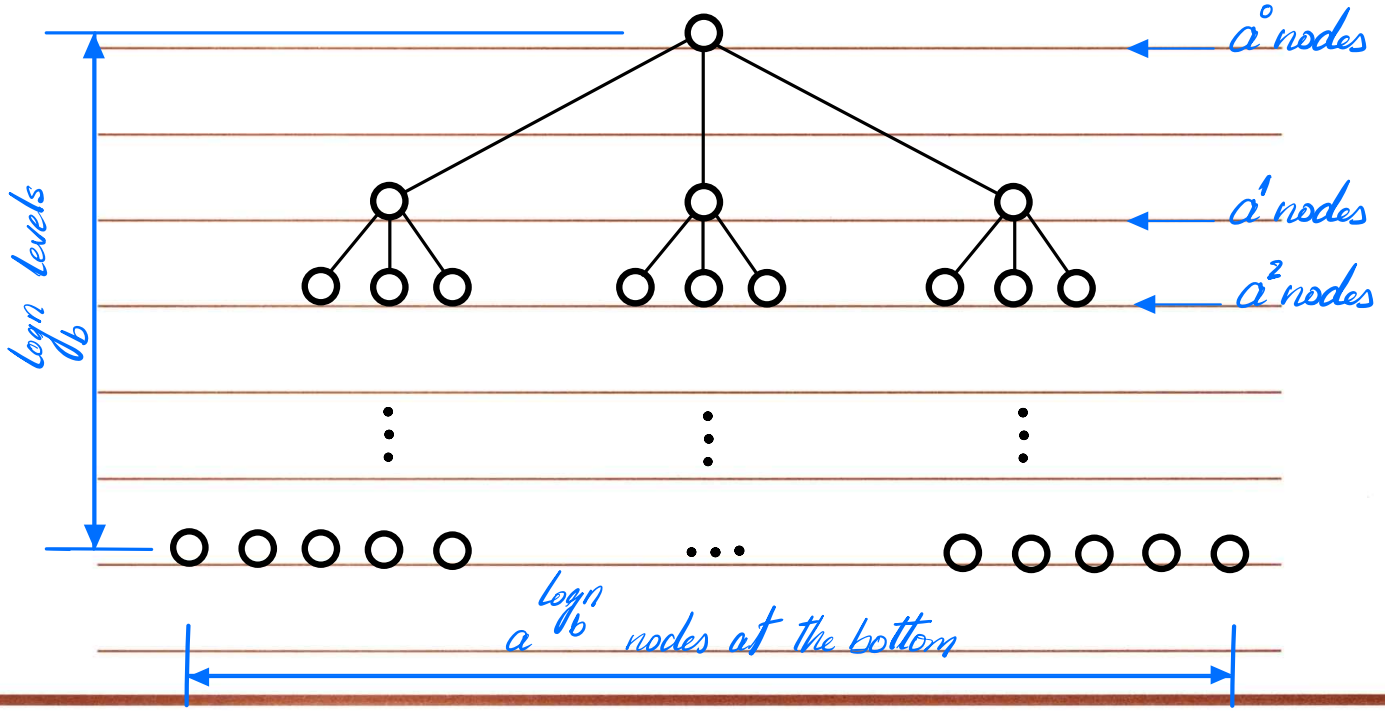
Generalizing case #2 of the Master Method

2- If $f(n) = \Theta(n^{\log_b a} \lg^k n)$ where $k \geq 0$,

then $T(n) = \Theta(n^{\log_b a} \lg^{k+1} n)$

Intuition Behind the Master Method

Case #1



$$a^{\log_b n} = n^{\log_b a}$$

total no. of subproblems =

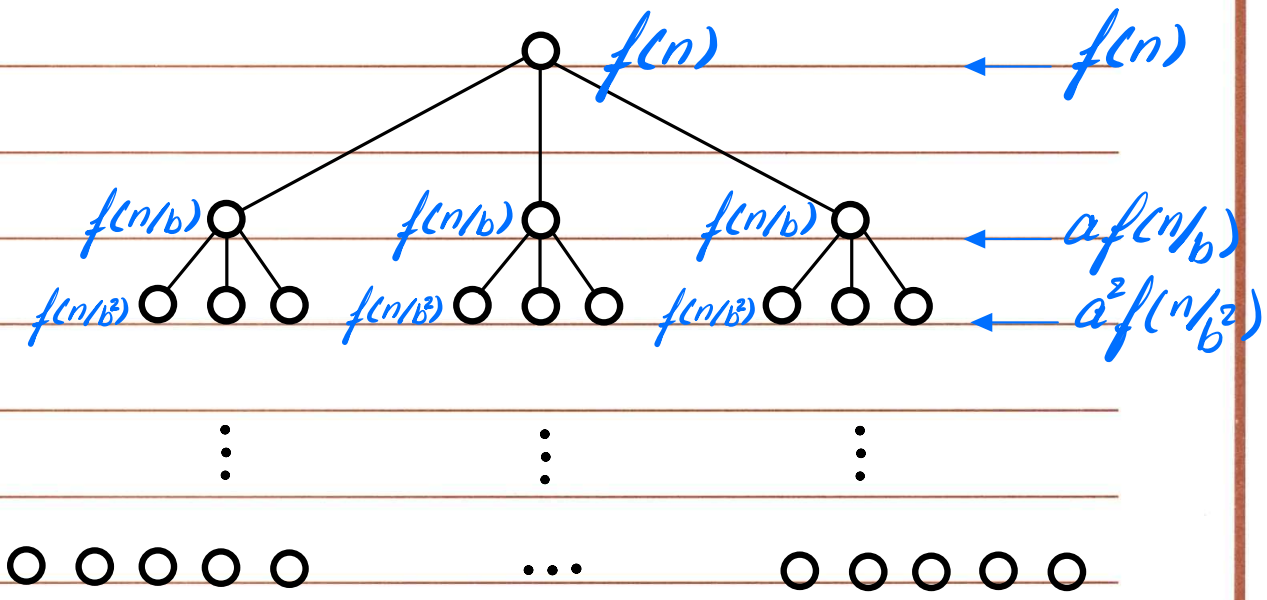
$$n^{\log_b a} + 1/a\, n^{\log_b a} + 1/a^2\, n^{\log_b a} + \cdots$$

$$= \frac{a}{(a-1)} n^{\log_b a} = \Theta(n^{\log_b a})$$

In case #1, the no. of subproblems dominates the complexity. Here, the tree is bottom heavy.

Intuition Behind the Master Method

Case #3



Regularity condition:

if $af(n/b) \leq cf(n)$ for some $c < 1$, then the cost of the combine and divide steps at each level must be a fraction of that in the previous level.

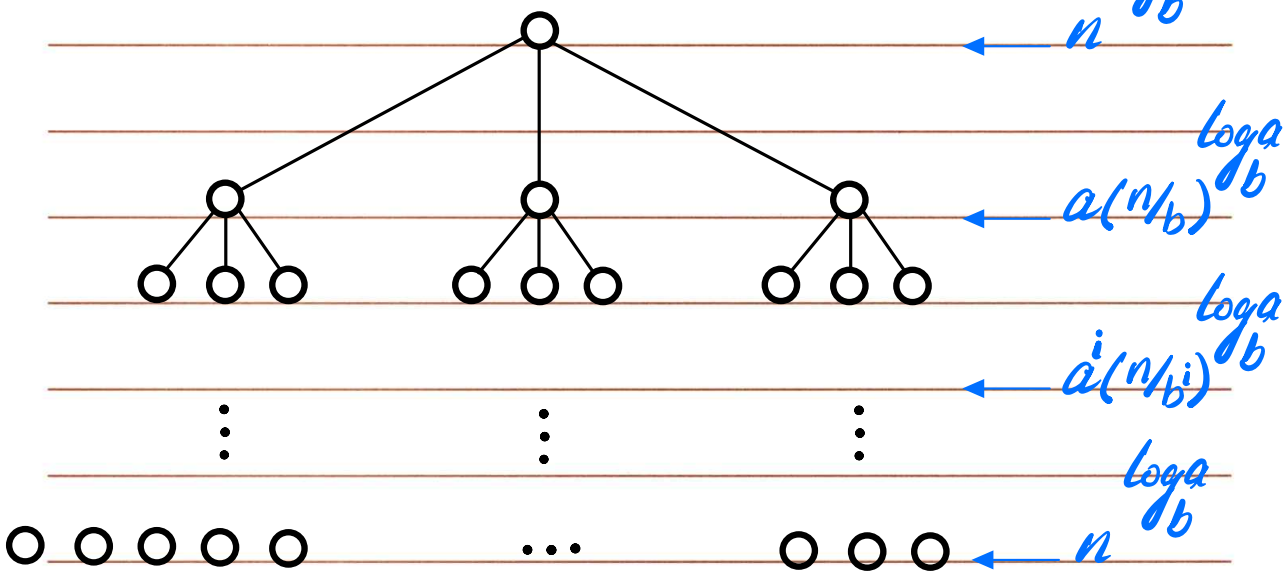
So, $f(n) + cf(n) + c^2 f(n) + \ldots =$

$f(n)/(1-c) \quad = \Theta(f(n))$

In case #3, the cost of combine and divide steps dominate the complexity. Here, the tree is top heavy.

Intuition Behind the Master Method

Case #2 $f(n) = \Theta(n^{\log_b a})$



At level i, $a^i(n/b^i)^{\log_b a} = n^{\log_b a}(a^i \cdot b^{-i\log_b a})$

$= n^{\log_b a}(a^i \cdot (b^{\log_b a})^{-i})$

$= n^{\log_b a}(a^i \cdot a^{-i}) = n^{\log_b a}$

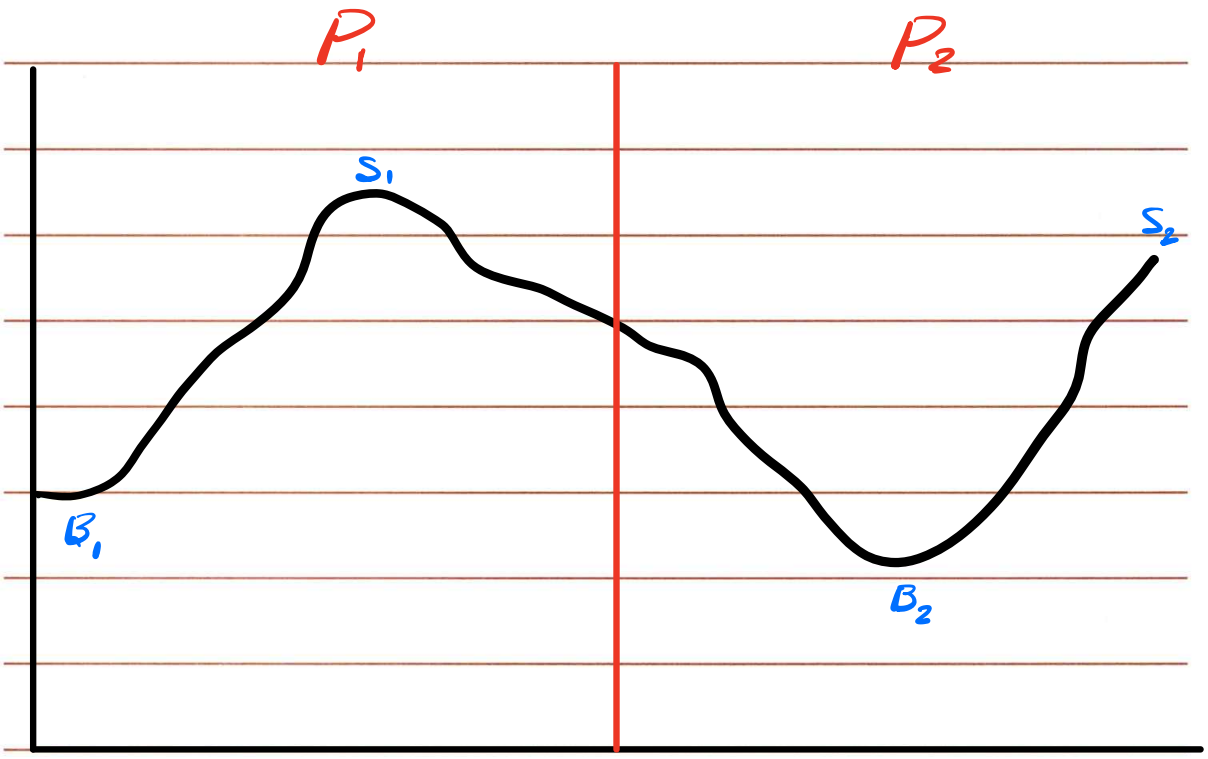
Since we have $\lg n$ levels, $T(n) = \Theta(n^{\log_b a} \lg n)$

In case #2, the cost of operations at each level is the same. The tree levels are equally weighted

Stock Market Problem

Given the price chart for a stock over a period of n day, which buy and sell days give us the maximum profit? (must buy before selling)

Brute force approach takes $\Theta(n^2)$ time.



Solve P_1 & P_2 recursively to find B_1, S_1, B_2 and S_2

Merge Step : Three cases to consider

Case 1 – Buy and Sell in P_1

$B = B_1$

$S = S_1$

Case 2 – Buy and Sell in P_2

$B = B_2$

$S = S_2$

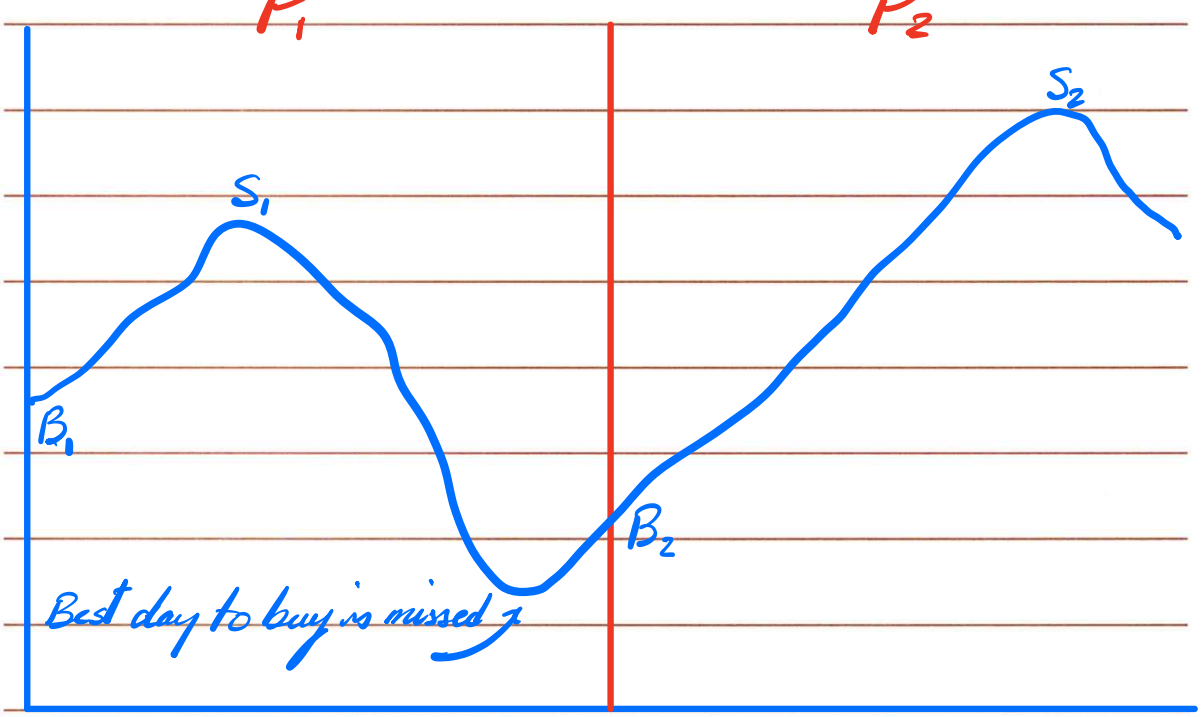
Case 3 – Buy in P_1 and Sell in P_2

$B = B_1$

$S = S_2$

This may not always work!

Consider the following case



Corrected Merge Step

$M = Min(M_1, M_2)$

$X = Max(X_1, X_2)$

Case 1 - Buy and Sell in P_1

$B = B_1$, $S = S_1$

Case 2 - Buy and Sell in P_2

$B = B_2$, $S = S_2$

Case 3 - Buy in P_1 and Sell in P_2

$B = M_1$ $S = X_2$

Complexity analysis - Using The Master Theorem

$a = 2$, $b = 2$, and

The cost of the divide and combine steps $= \Theta(1)$

$n^{\log_b a} = n^{\log_2 2} = n^1$

$f(n) = \Theta(1)$

$\Rightarrow$ Case 1:

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

Dense Matrix Multiplication

Assuming square dense matrices,

$$\underbrace{\begin{bmatrix} A \end{bmatrix}}_{n} \underbrace{\begin{bmatrix} B \end{bmatrix}}_{n} = \underbrace{\begin{bmatrix} C \end{bmatrix}}_{n} \Big\}n$$

Brute force approach leads to $T(n) = \Theta(n^3)$

Try divide & conquer

$$\begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} \begin{bmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{bmatrix} \qquad \begin{bmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{bmatrix}$$

$$C_{11} = A_{11} \cdot B_{11} + A_{12} \cdot B_{21}$$
$$C_{12} = A_{11} \cdot B_{12} + A_{12} \cdot B_{22}$$
$$C_{21} = A_{21} \cdot B_{11} + A_{22} \cdot B_{21}$$
$$C_{22} = A_{21} \cdot B_{12} + A_{22} \cdot B_{22}$$

Complexity analysis - Using The Master Theorem

$a = 8$, $b = 2$, and

matrix additions

the cost of the divide and combine steps $= \Theta(n^2)$

$\Theta(1)$ $\quad$ $\Theta(n^2)$

$$n^{\log_b a} = n^{\log_2 8} = n^3$$

$$f(n) = \Theta(n^2)$$

$\Rightarrow$ Case 1:

$$T(n) = \Theta(n^3)$$

Compute 7 $n/2 \times n/2$ intermediate matrices

(Strassen's Algorithm)

$$P = (A_{11} + A_{22})(B_{11} + B_{22})$$

$$Q = (A_{21} + A_{22}) B_{11}$$

$$R = A_{11} (B_{12} - B_{22})$$

$$S = A_{22} (B_{21} - B_{11})$$

$$T = (A_{11} + A_{12}) B_{22}$$

$$U = (A_{21} - A_{11})(B_{11} + B_{12})$$

$$V = (A_{12} - A_{22})(B_{21} + B_{22})$$

$$C_{11} = P + S - T + V$$

$$C_{12} = R + T$$

$$C_{21} = Q + S$$

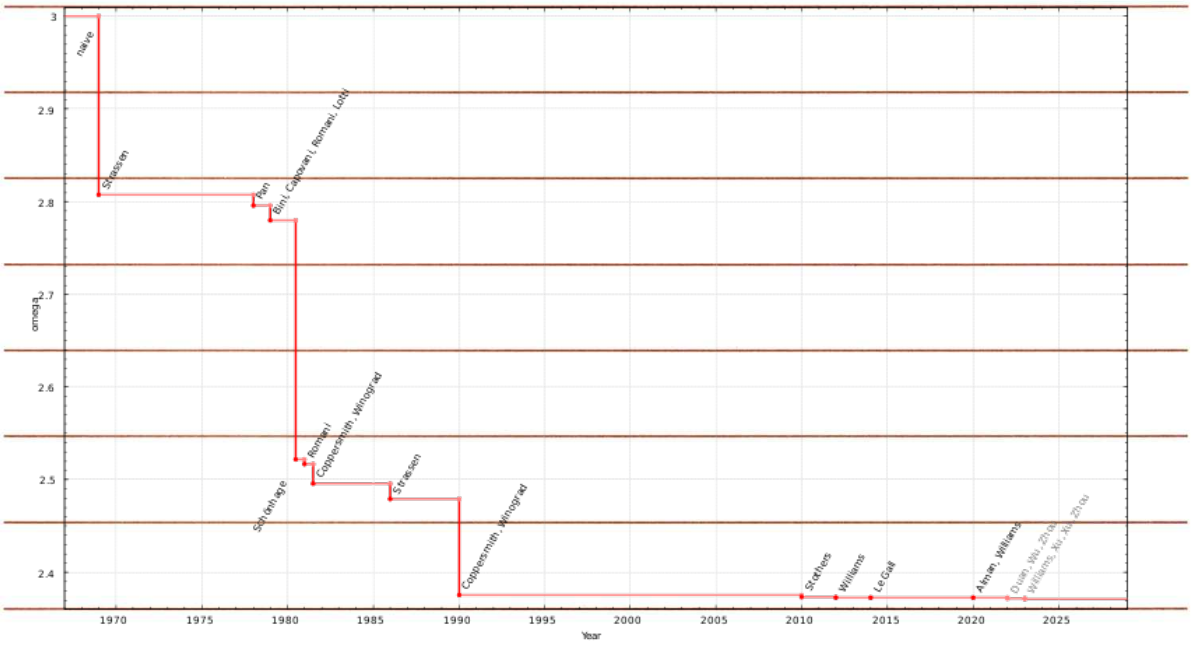
$$C_{22} = P + R - Q + U$$

Complexity Analyses

$a = 7$, $b = 2$, $f(n) = \Theta(n^2)$

$$n^{\log_b a} = n^{\log_2 7} \simeq n^{2.81}$$

$\Rightarrow$ Case#1: $T(n) = \Theta(n^{2.81})$

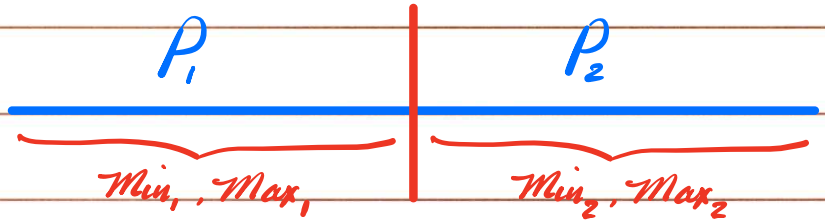


From Wikipedia

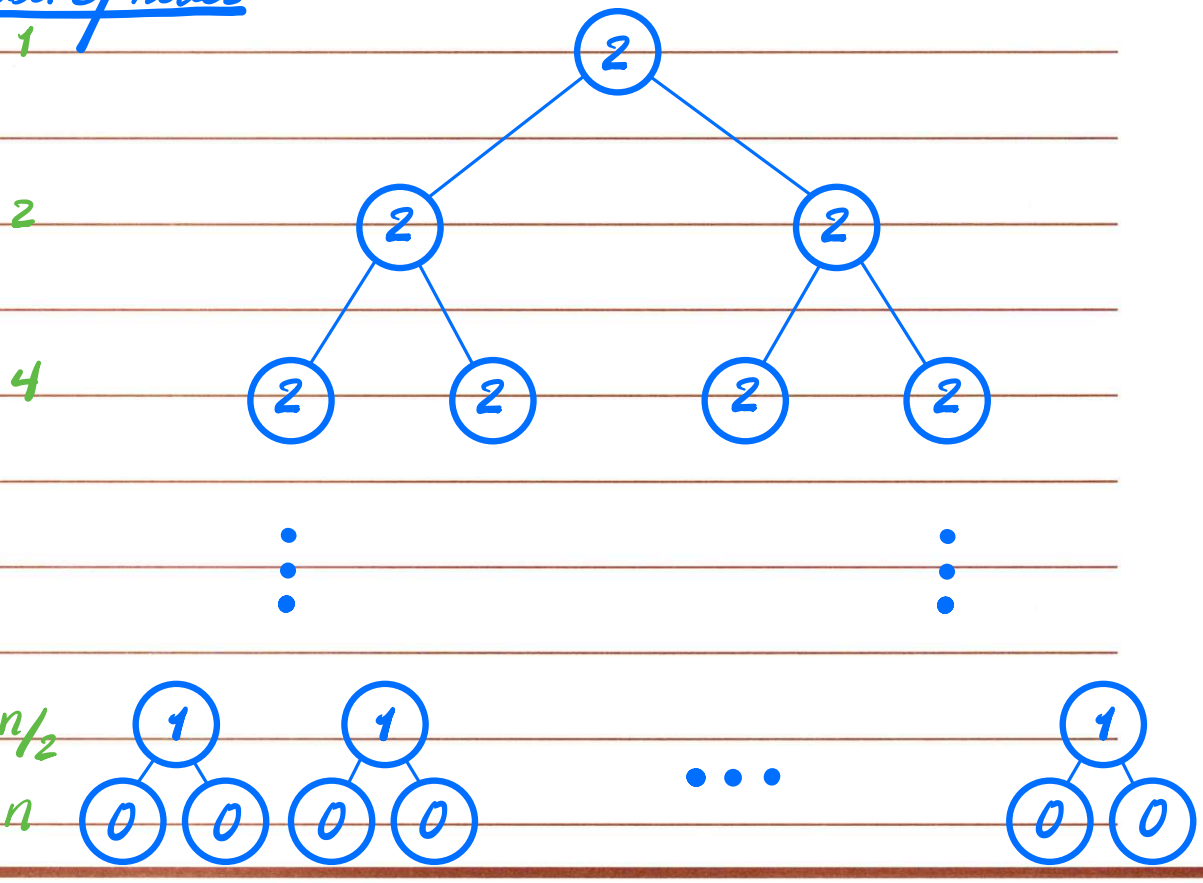
Finding the minimum and the maximum in an unsorted array.

Number of comparisons required:

- Brute force $(n-1)+(n-1) = 2n-2 = O(n)$
- Divide & Conquer ?



Number of nodes



Total no. of nodes above the bottom level = n-1

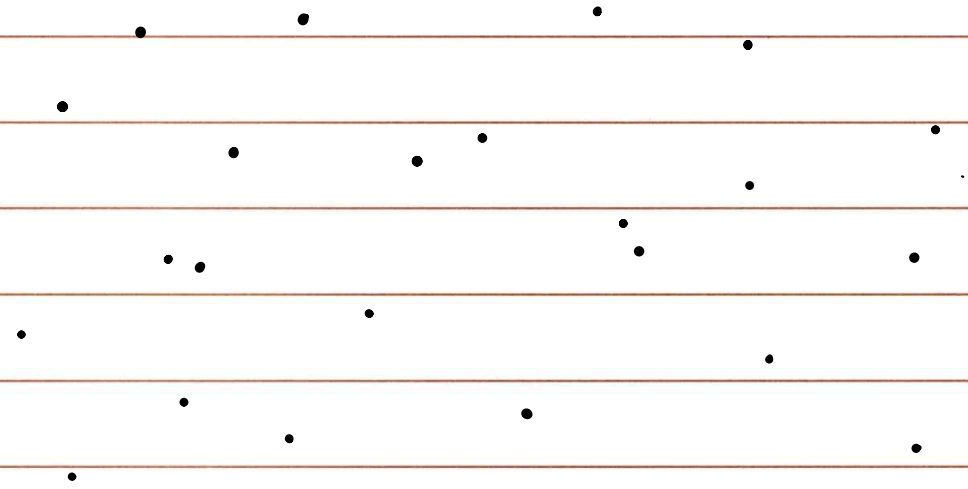
total no. of comparisons required =

$$(n-1)*2 - n/2$$

$$= 3n/2 - 2$$

Problem Statement

Find the closest pair of points on a Euclidean plane

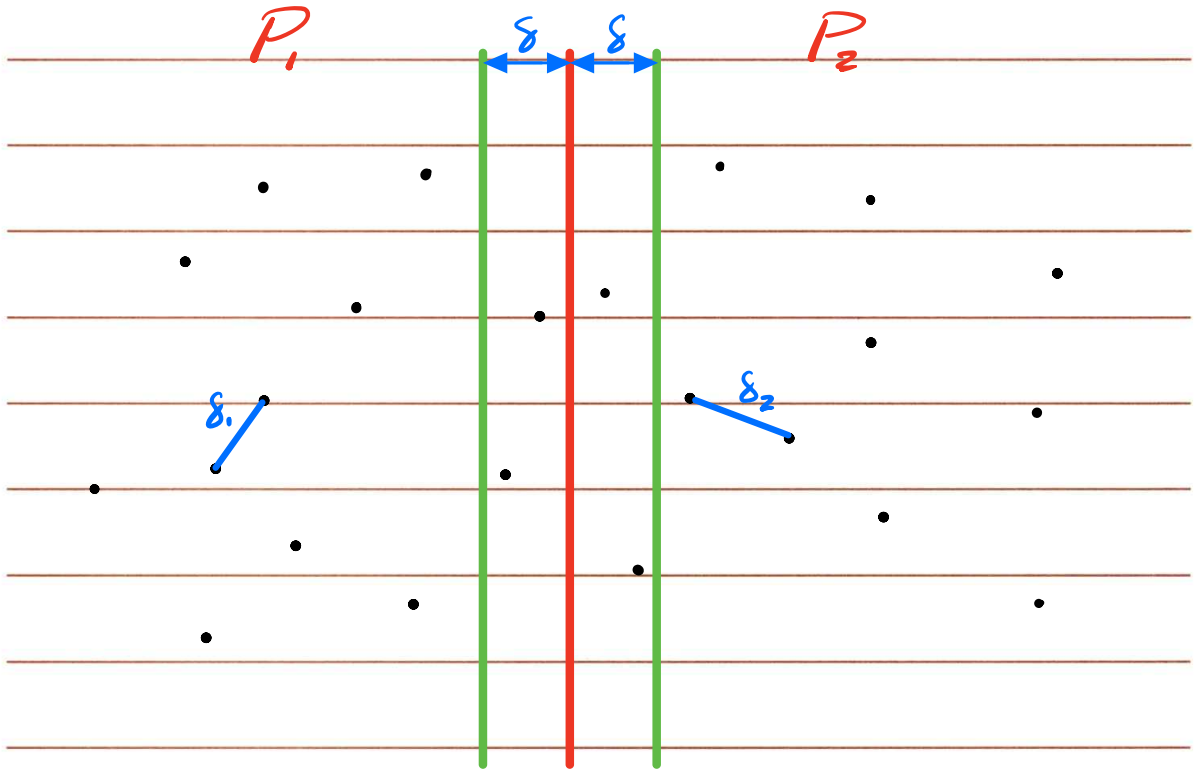


Number of distance calculations and comparisons required:

- Brute force

  $(n-1)+(n-2)+\cdots+1 \quad = \Theta(n^2)$

- Divide & Conquer ?



$\delta = \min(\delta_1, \delta_2)$

Merge Step : Three cases to consider

Case 1 - Both points are in P_1

(found recursively)

Case 2 - Both points are in P_2

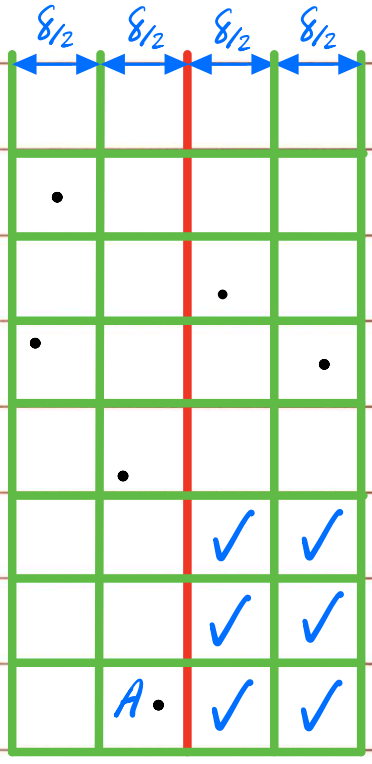
(found recursively)

Case 3 - One point is in P_1 and the other in P_2

How do we find these two pts.?

Consider a $\delta/2 \times \delta/2$ square 

Farthest two points within this square are $\frac{\delta\sqrt{2}}{2}$ away from each other. ($\frac{\delta\sqrt{2}}{2} < \delta$)



✓ candidate cells to look for a pt. that is closer than δ to A.

Implementation

closest_pair (P)

$\Theta(n \log n)$ Construct P_x : List of pts. sorted by x-coord.

$\Theta(n \log n)$ Construct P_y : List of pts. sorted by y-coord.

$(p_0, p_1) =$ closest_pair_Rec(P_x, P_y)

---

closest_pair_Rec(P_x, P_y)

if $|P| \leq 3$ then

solve it directly

else

$\Theta(1)$ construct Q_x ... left half of P_x

$\Theta(n)$ construct Q_y ... list of points in Q_x sorted by y-coord.

$\Theta(1)$ construct R_x ... right half of P_x

$\Theta(n)$ construct R_y ... list of points in R_x sorted by y-coord.

$(q_0, q_1) = \text{closest\_pair\_Rec}(Q_x, Q_y)$

$(r_0, r_1) = \text{closest\_pair\_Rec}(R_x, R_y)$

$\Theta(1)$ $\delta = \text{Min}(d(q_0, q_1), d(r_0, r_1))$

$\Theta(n)$ S = set of points in P within distance of δ from L (the center line)

construct S_y ... set of points in S sorted by y-coord.

$\Theta(n)$ for each point $s \in S_y$, compute distance from s to each of the next 11 points in S_y

Let (s, s') be the pair with min. distance

$\Theta(1)$

```
if d(s,s') < δ then
    Return (s,s')
elseif d(q0,q1) < d(r0,r1) then
    Return(q0,q1)
else
    Return(r0,r1)
endif
```

Complexity analysis - Using The Master Theorem

$a = 2$, $b = 2$, and

The cost of the divide and combine steps $= \Theta(n)$

$$n^{\log_b a} = n^{\log_2 2} = n^1$$

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

$\Rightarrow$ Case 2:

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

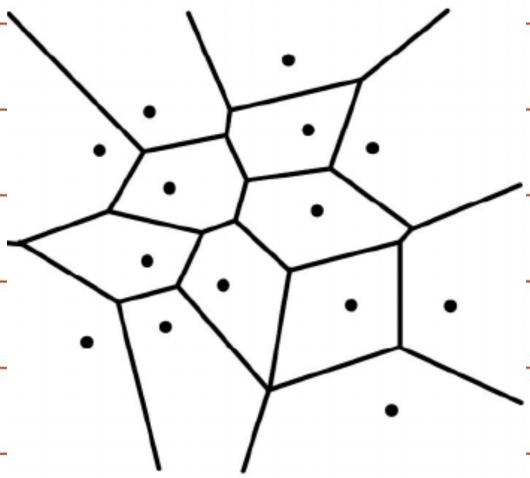
Cost of the preparation step (driver): $\Theta(n \lg n)$

Cost of the divide & conquer solution: $\Theta(n \lg n)$

Overall worst-case time complexity: $\Theta(n \lg n)$

All Nearest Neighbors Problem

Given n points in the plane, find a nearest neighbor of each



Voronoi Diagram