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.

Proof of Correctness

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

Base Case:

Inductive Step:

Runtime Analysis: (Method 2)

Divide – Takes ___

Conquer – If the original problem takes $T(n)$ time, the two subproblems take ___

Combine – Takes ___ on array of size n

Recurrence Eq. for merge sort:

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

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

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

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:

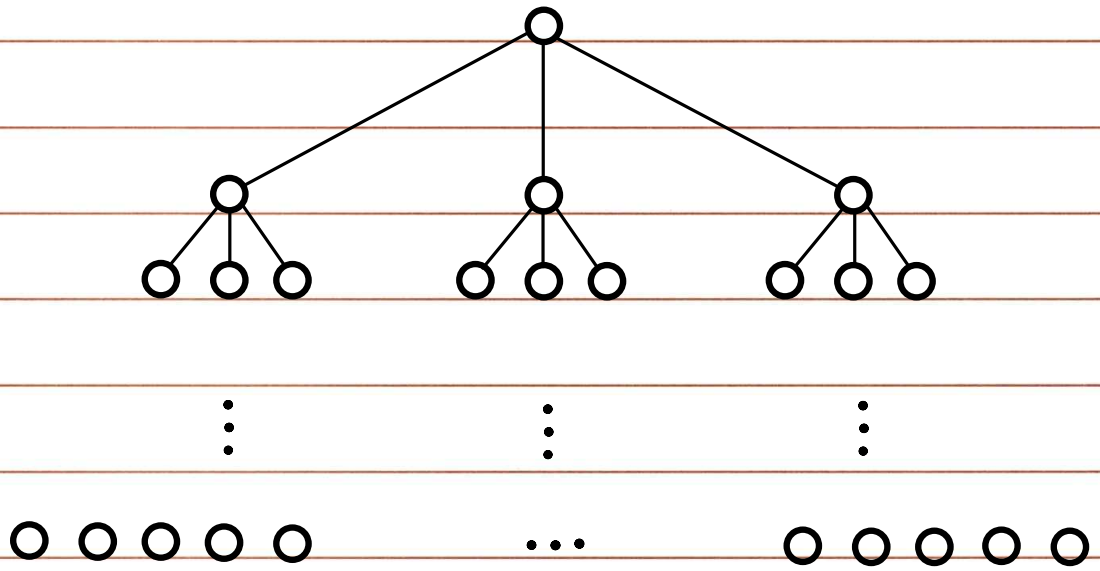
1- If $f(n) = O(n^{\log_b a - \varepsilon})$ for some $\varepsilon > 0$, then $T(n) = \Theta(n^{\log_b a})$

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

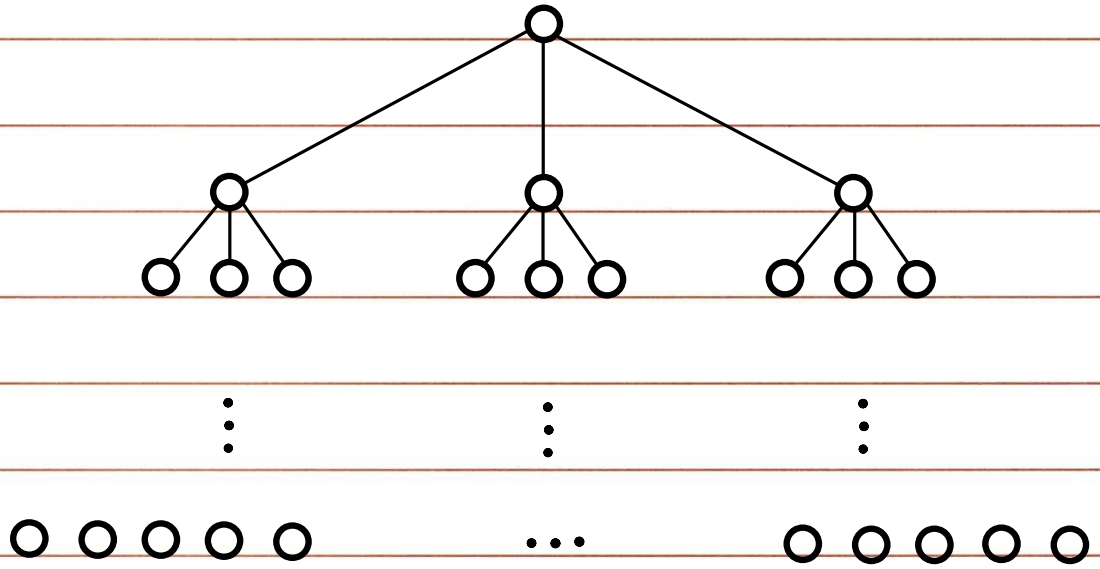
Intuition Behind the Master Method

Case #1



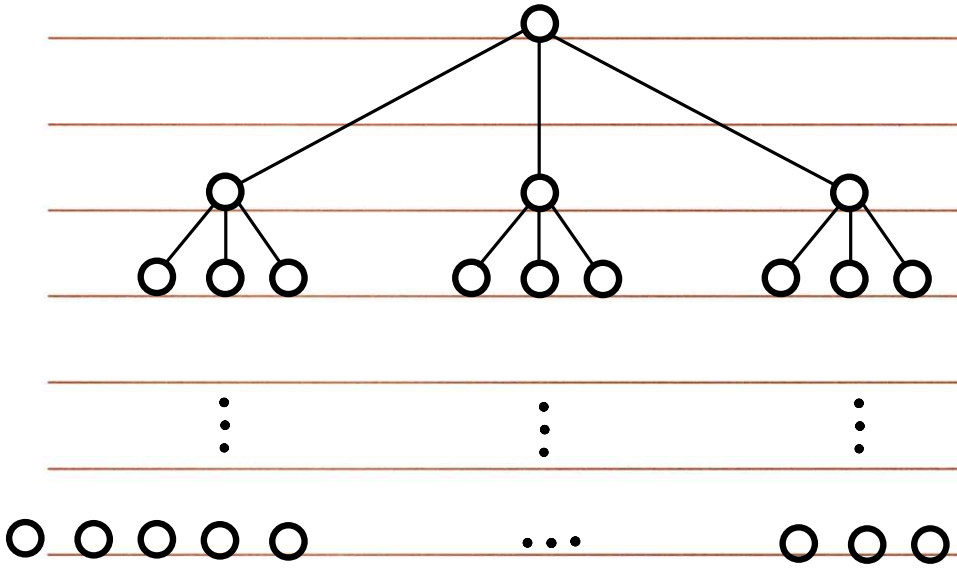
Intuition Behind the Master Method

Case #3



Intuition Behind the Master Method

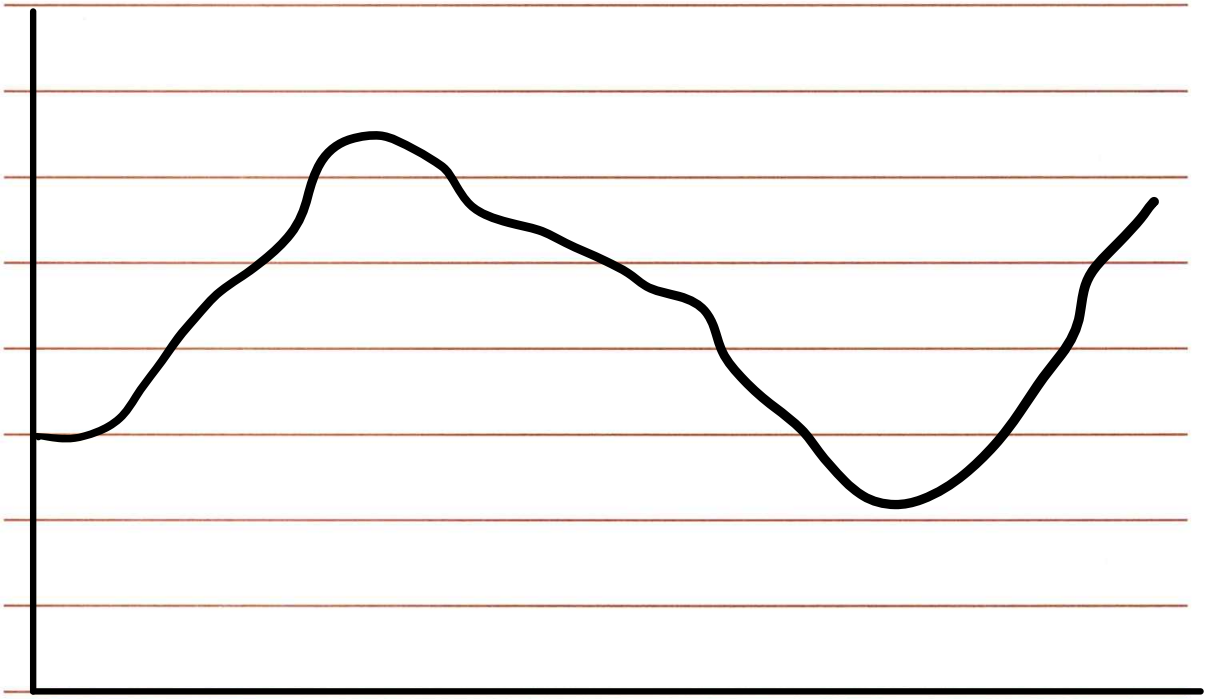
Case #2



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.



Dense Matrix Multiplication

Assuming square dense matrices,

$$[A][B] = [C]$$

Try divide & conquer

Finding the minimum and the maximum in an unsorted array.

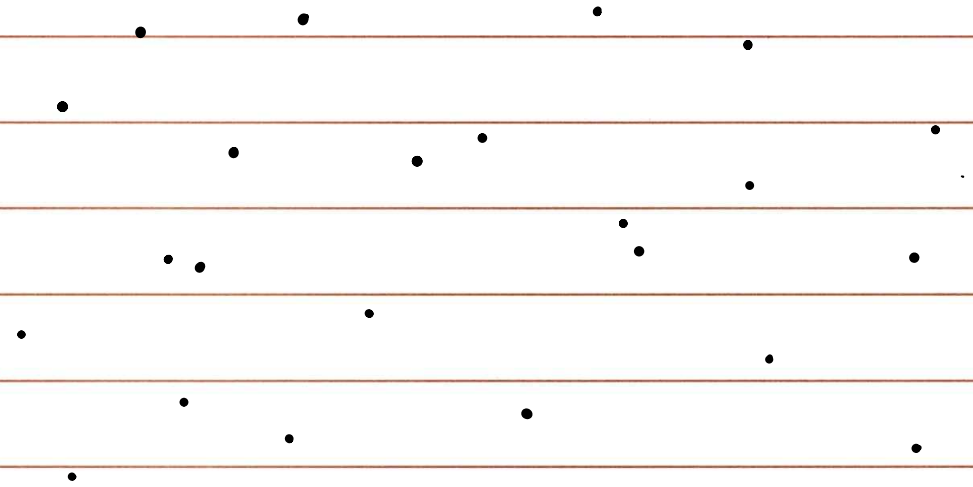
Number of comparisons required:

- Brute force

- Divide & Conquer

Problem Statement

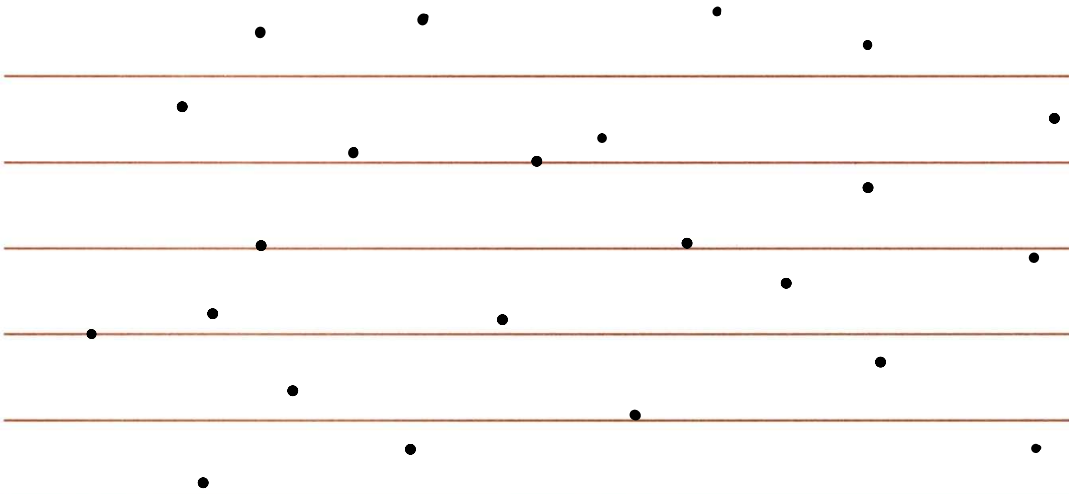
Find the closest pair of points on a Euclidean plane



Number of distance calculations and comparisons required:

- Brute force

- Divide & Conquer



Implementation

```
closest_pair (P)
    Construct Px : List of pts. sorted
                   by x-coord.
    Construct Py : List of pts. sorted
                   by y-coord.
    (p0, p1) = closest_pair_Rec(Px, Py)
```

```
closest_pair_Rec(Px, Py)
  if |P| ≤ 3 then
      solve it directly
  else
      construct Qx ... left half of Px
      construct Qy ... list of points in
                       Qx sorted by y-coord.
      construct Rx ... right half of Px
      construct Ry ... list of points in
                       Rx sorted by y-coord.
```

(q_0, q_1) = closest_pair_Rec(Q_x, Q_y)

(r_0, r_1) = closest_pair_Rec(R_x, R_y)

δ = Min$(d(q_0, q_1), d(r_0, r_1))$

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.

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

if $d(s, s') < \delta$ then

    Return (s, s')

elseif $d(q_0, q_1) < d(r_0, r_1)$ then

    Return(q_0, q_1)

else

    Return(r_0, r_1)

endif

Complexity analysis - Using The Master Theorem

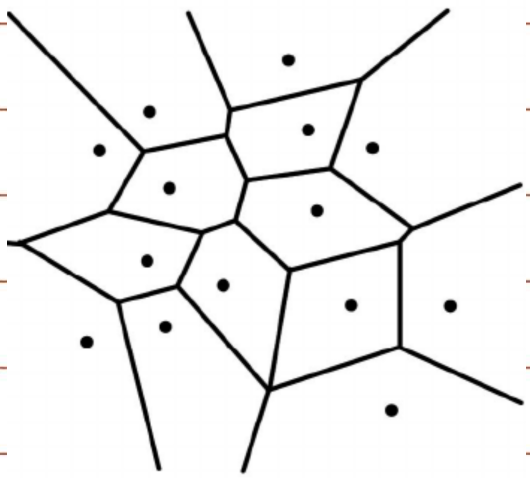
Cost of the preparation step (driver):

Cost of the divide & conquer solution:

Overall worst-case time complexity:

All Nearest Neighbors Problem

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



Voronoi Diagram