Warming-up Module 2: Smallest Triangle Problem

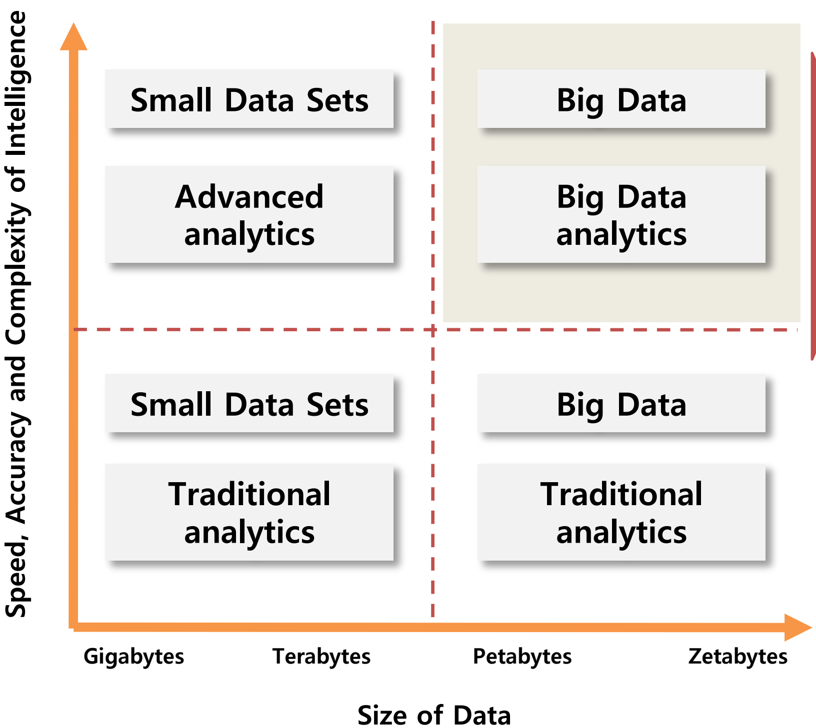
Younghoon Kim
(nongaussian@hanyang.ac.kr)

Purpose

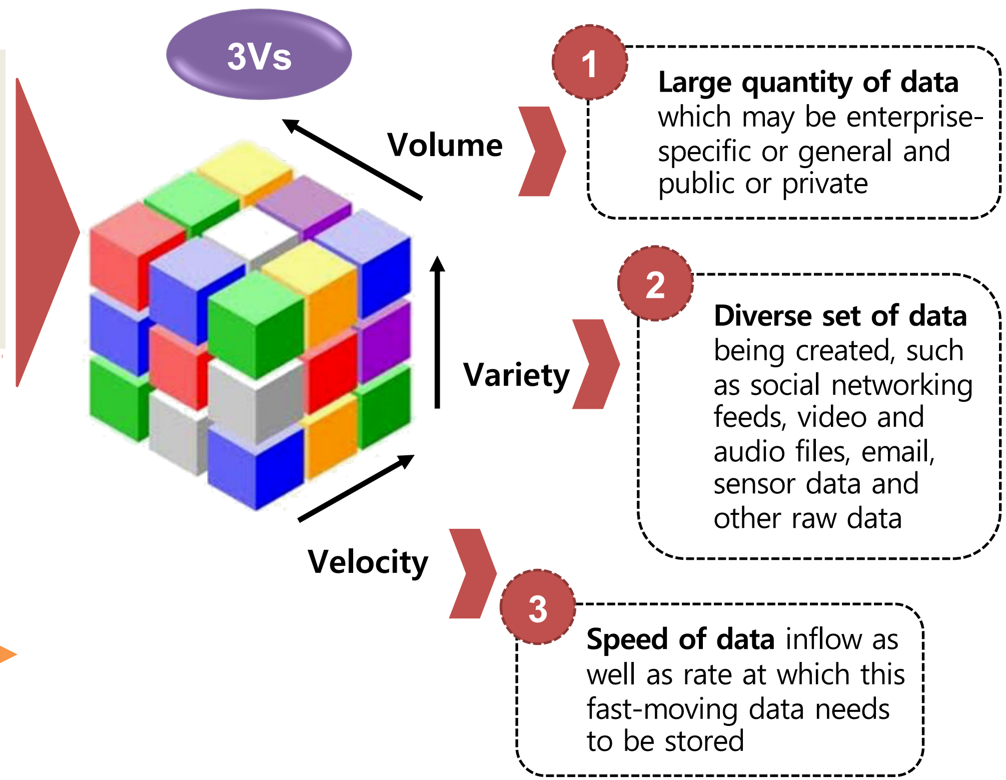
- Encourage your teamwork
  - Know the strong points of your team's members
  - E.g.,
    - "A writes Java code very well!"
    - "B is very good at math and algorithm!"
- Understand why processing big data is so hard

What is Big Data?

Big Data relates to rapidly growing, *Structured and Unstructured datasets* with sizes beyond the ability of conventional database tools to store, manage, and analyze them. In addition to its size and complexity, it refers to its ability to help in *"Evidence-Based" Decision-making*, having a high impact on business operations



Source: CRISIL GR&A analysis



Source: CRISIL GR&A analysis

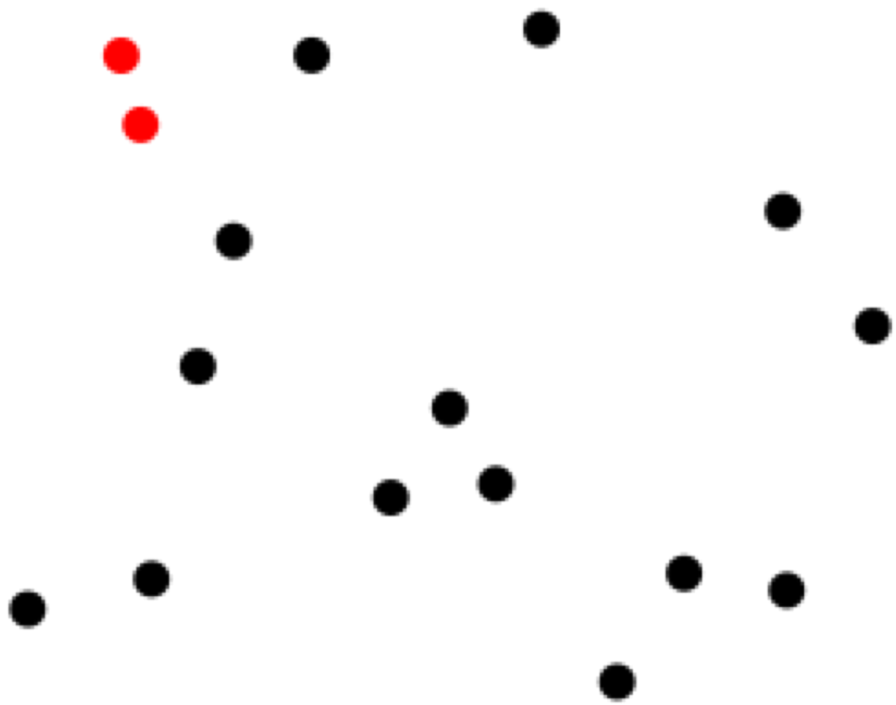
How to Deal with Big Data?

- Sample & analysis with small data
- Find more efficient algorithms
- Distribute a task & compute it in parallel

Finding the Closest Pair

- Given
  - A set of d-dimensional points
    - $D = \{p_1, p_2, \ldots, p_n\}$
    - p_i : a d-dimensional point $<p_{i1}, \ldots, p_{id}>$
- Find
  - A pair of points from D whose Euclidean distance is the smallest

Finding the Closest Pair



A Naïve Algorithm

- mind ← ∞
- minpair ← (-1, -1)
- For i = 0 to n-2
  - For j = i+1 to n-1
    - d ← Compute d(p_i, p_j)
    - if mind > d
      - mind ← d
      - minpair ← (i, j)
- return mind, minpair

Time complexity = $O(n^2)$

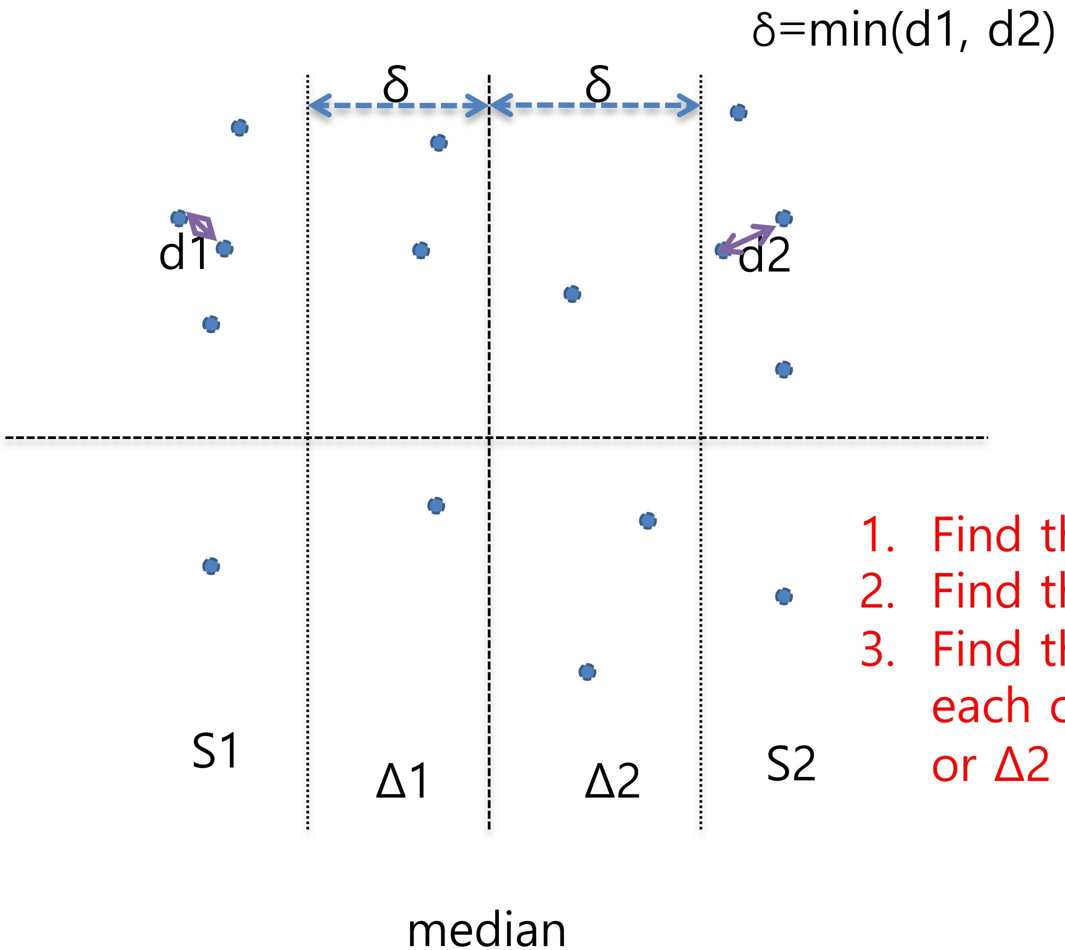
Sampling

- We cannot find the exact answer using sampling!
- ➔ Sampling is not a good solution

Too Slow!

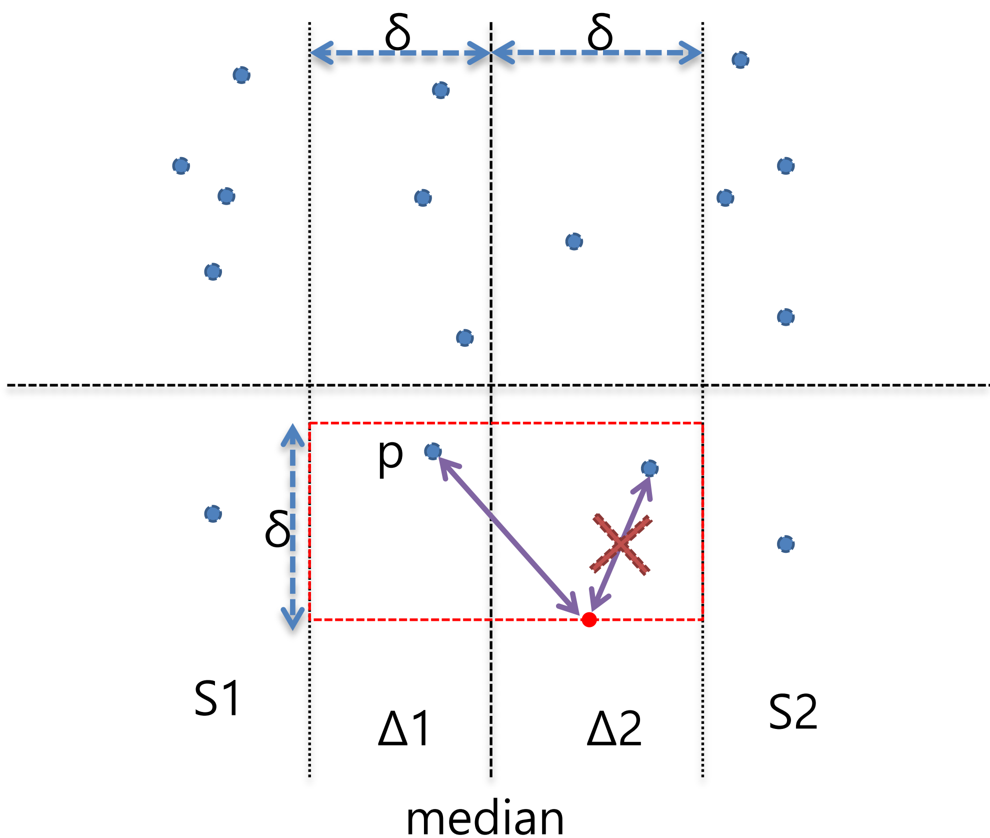
- Time complexity $\geq O(n^2)$
- How to improve the performance
  - 1. Parallelization
  - 2. Develop a more efficient algorithm

Divide-and-Conquer Algorithm



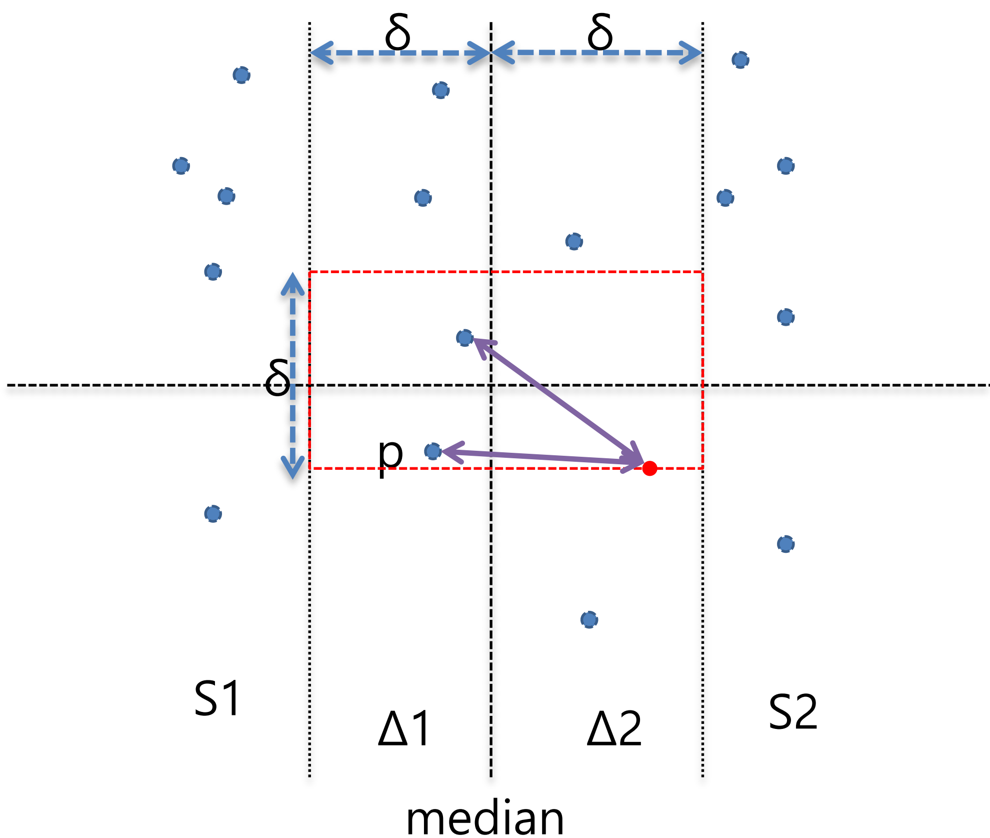
1. Find the closest pair in S1
2. Find the closest pair in S2
3. Find the closest pair of points each of which belongs to Δ1 or Δ2 respectively

Divide-and-Conquer Algorithm



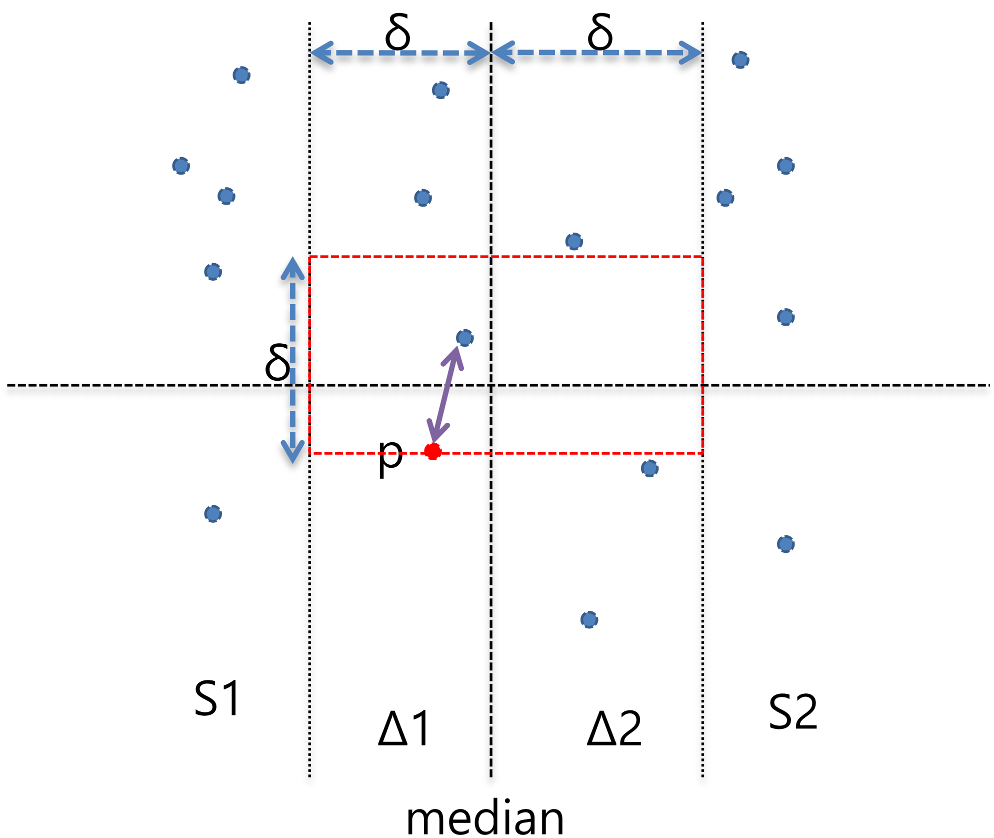
δ=min(d1, d2)

Divide-and-Conquer Algorithm



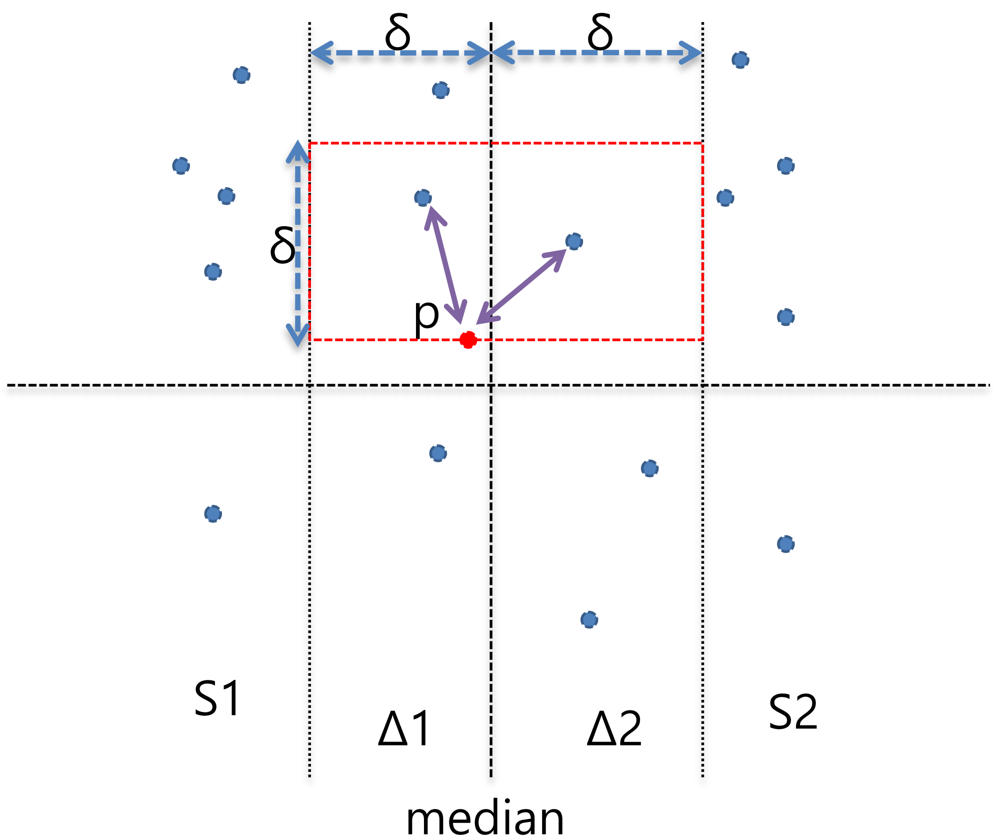
δ=min(d1, d2)

Divide-and-Conquer Algorithm



δ=min(d1, d2)

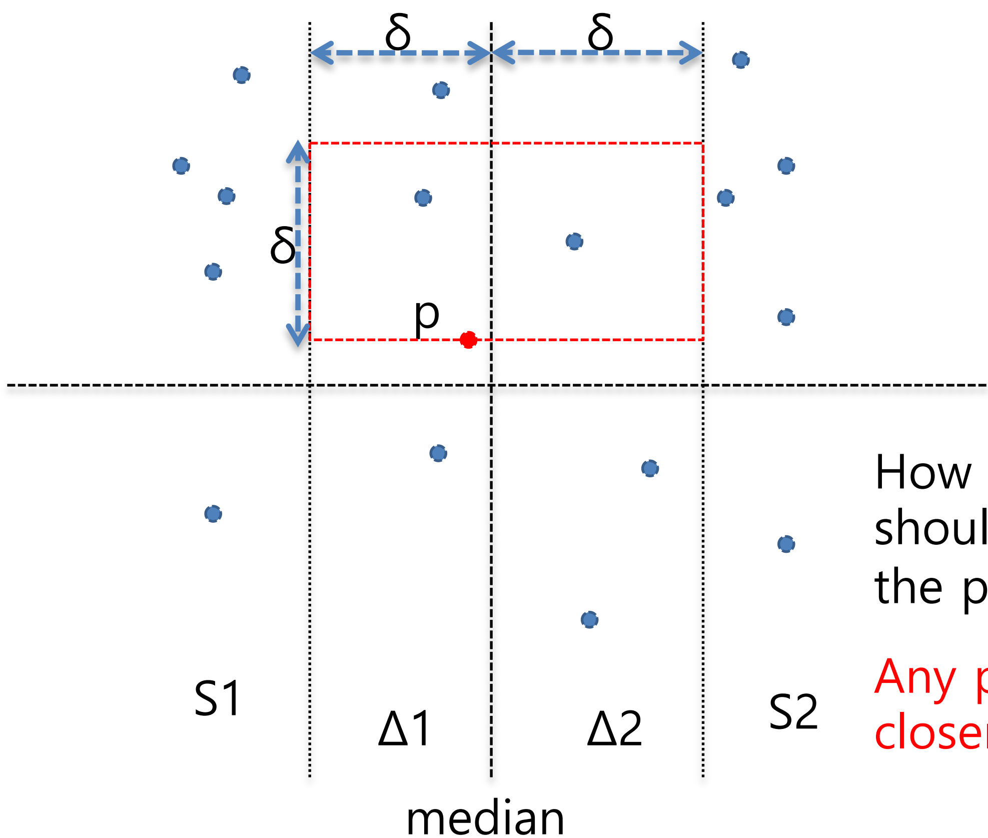
Divide-and-Conquer Algorithm



δ=min(d1, d2)

```
divide-and-conquer (xP, yP)
                  where xP is P(1) .. P(N) sorted by x coordinate, and
                          yP is P(1) .. P(N) sorted by y coordinate (ascending order)
if N ≤ 3 then
  return <closest, closestPair> of xP using brute-force algorithm
else
  xL ← points of xP from 1 to ⌈N/2⌉
  xR ← points of xP from ⌈N/2⌉+1 to N
  xm ← xP(⌈N/2⌉)x // x value of the median
  yL ← { p ∈ yP : px ≤ xm } // list of points sorted by y coordinate
  yR ← { p ∈ yP : px > xm }
  (dL, pairL) ← divide-and-conquer(xL, yL)
  (dR, pairR) ← divide-and-conquer(xR, yR)
  (dmin, pairMin) ← (dR, pairR)
  if dL < dR then
     (dmin, pairMin) ← (dL, pairL)
  endif
  yS ← { p ∈ yP : |xm - px| < dmin } // list of points sorted by y coordinate
  nS ← number of points in yS
  (closest, closestPair) ← (dmin, pairMin)
  for i from 1 to nS - 1
     k ← i + 1
     while k ≤ nS and yS(k)y - yS(i)y < dmin
        if |yS(k) - yS(i)| < closest then
           (closest, closestPair) ← (|yS(k) - yS(i)|, {yS(k), yS(i)})
        endif
        k ← k + 1
     endwhile
  endfor
  return <closest, closestPair>
endif
```

Divide-and-Conquer Algorithm



δ=min(d1, d2)

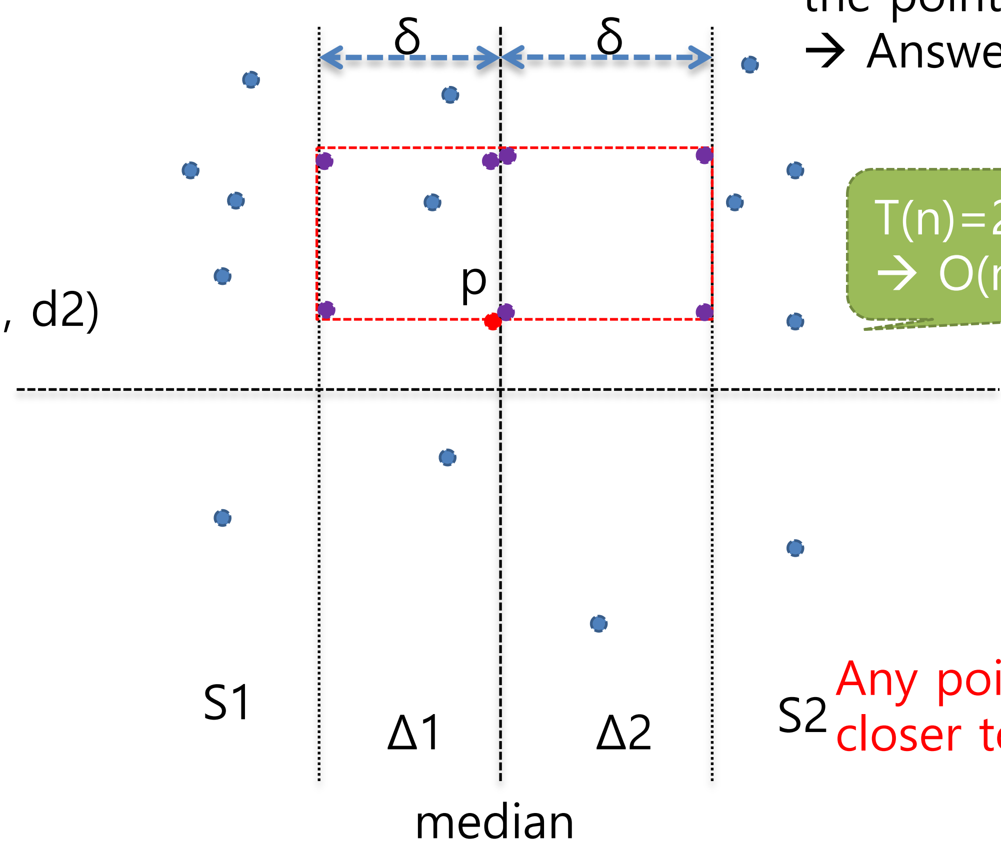
How many points in the red box should be considered with the point p?

Any point in the box cannot be closer to p than δ

Divide-and-Conquer Algorithm

How many points in the red box should be considered with the point p?
→ Answer: at most 8

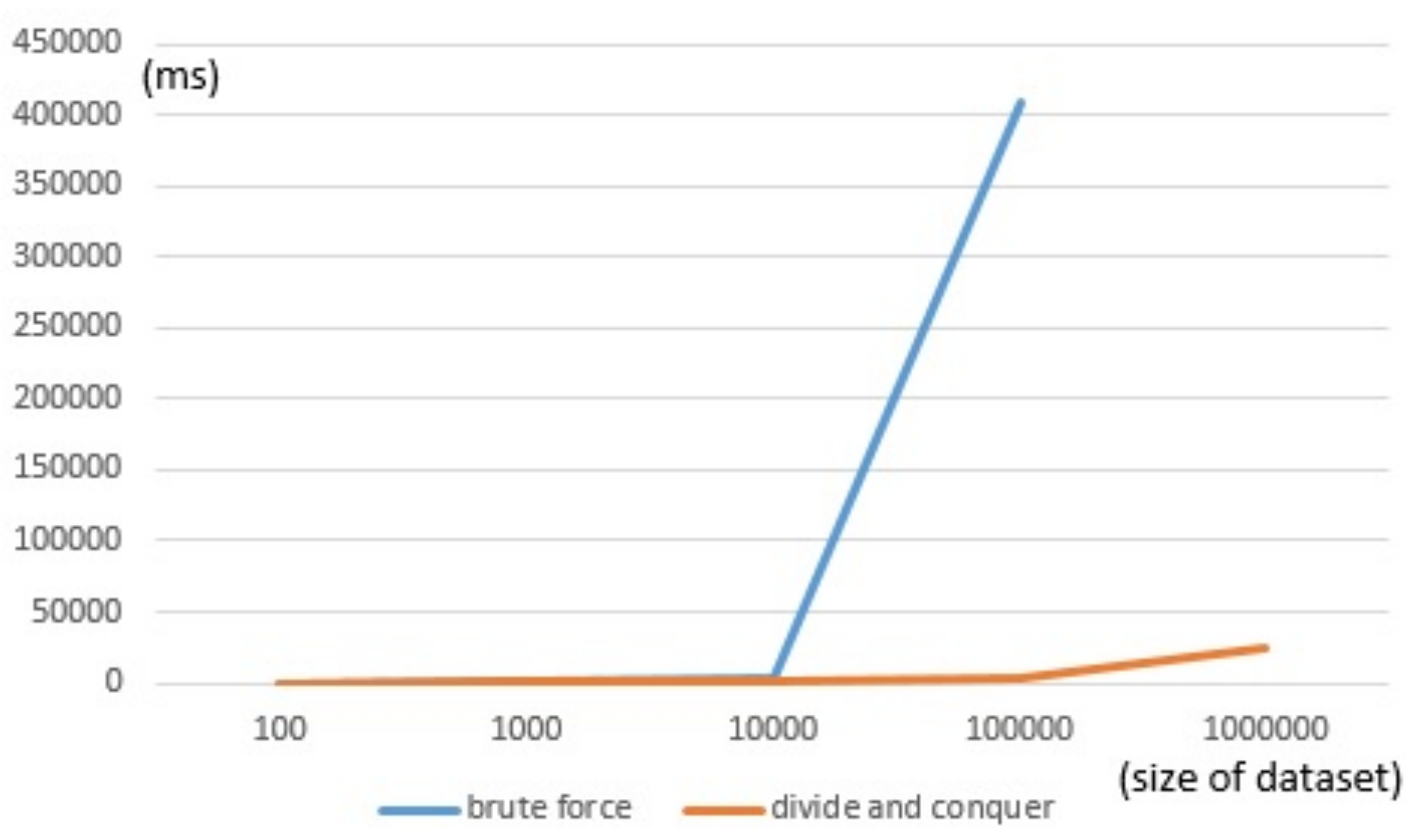
δ=min(d1, d2)



T(n)=2T(n/2) + O(7n)
→ O(nlogn)

Any point in the box cannot be closer to p than δ

Execution Time



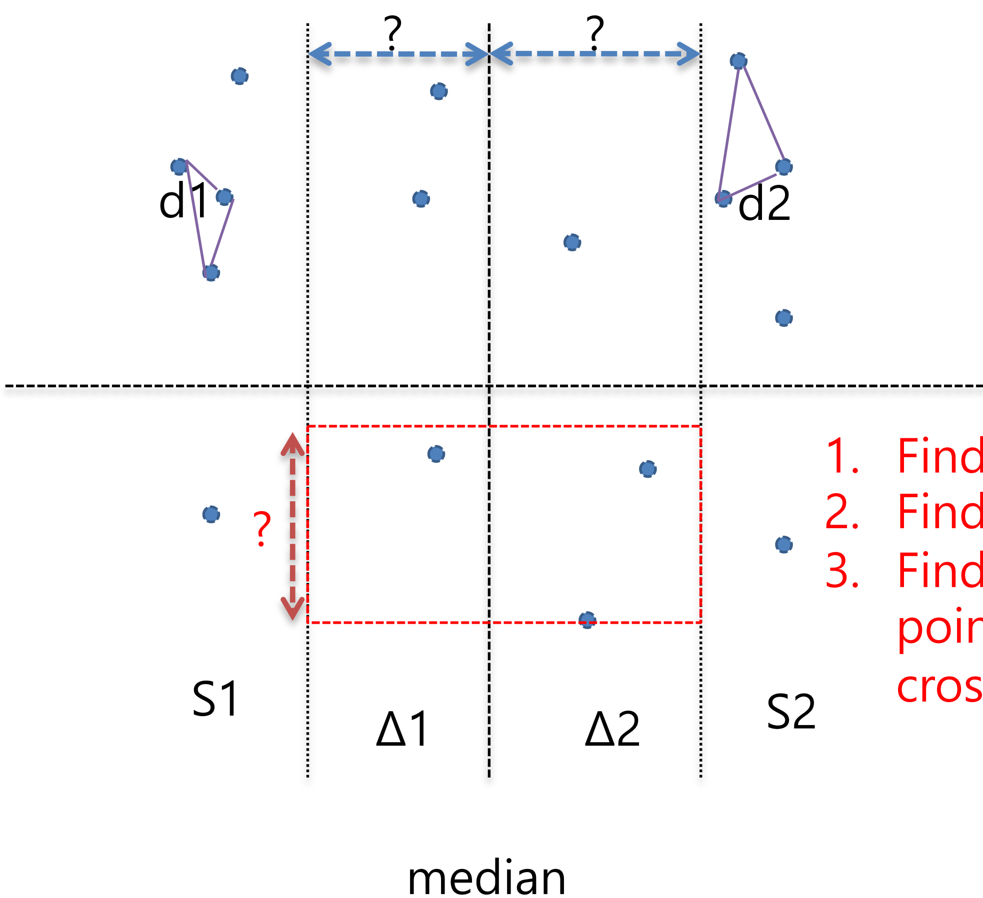
Smallest Triangle Problem

- Given
  - **n** points **A[1..n]** in the 2-D plane,
    - where **i**-th point has two attributes **A[i].x** and **A[i].y** representing x-coordinate and y-coordinate
- Goal
  - Find 3 points **A[i]**, **A[j]**, **A[k]** (i, j, k are distinct)
    - such that **d(A[i], A[j]) + d(A[j], A[k]) + d(A[k], A[i])** is minimized
    - d(A[i],A[j]) = $\sqrt{(A[i].x - A[j].x)^2 + (A[i].y - A[j].y)^2}$

Input and Output

- **Input**
  - A file starts with a positive integer **n** indicating the number of points
  - The following **n** lines contain point ID **A[i].id**, two real numbers **A[i].x** and **A[i].y** with a delimiter ‘,’
- **Output**
  - Output **3** lines in total
  - Each line show a point ID in the smallest triangle
  - Print on the screen (e.g., use ‘System.out.println()’)

Divide-and-Conquer Algorithm



1. Find the smallest tri. in S1
2. Find the smallest tri. in S2
3. Find the smallest tri. ir of points one of whose line crosses the median line

Discuss the time complexity

A Naïve Algorithm

- min ← ∞
- mintrip ← (-1, -1)
- For i = 0 to n-3
  - For j = i+1 to n-2
    - For k = j+1 to n-1
      - d_1 ← Compute $d(p_i, p_j)$
      - d_2 ← Compute $d(p_i, p_k)$
      - d_3 ← Compute $d(p_k, p_j)$
      - perim ← $d_1 + d_2 + d_3$
      - if min > perim
        - » min ← perim
        - » mintrip ← (i, j, k)
- return min, mintrip

Time complexity = $O(n^3)$

Data set

- Data sets
  - Github
  - Files:
    - Varying size: 100.dat ~ 1000000.dat
    - Dimensionality = 2
  - File format:
    - The first line contains an integer indicating the number of points
    - Each line has a point with delimiter = ','
    - The first column is its point ID
    - E.g., 17,0.187096,0.822353
- Command line input arguments
  - $ java SmallestTriangle <filename>
- Output → standard out (= print on the screen)
  - A point ID in the smallest triangle in each line
  - E.g.,
    - 17
    - 18
    - 20

Example

- **Input:**
  - 4
    0,0.0,0.0
    1,2.0,2.0
    2,2.0,1.0
    3,1.0,1.0
- **Output:**
  - 1
    2
    3

PBL Class

Blue: Instructor's lecture

Red: PBL team activity

	Week 1	Week 2	Week 3
Morning	Lecture (Theory)	Early-result presentation	Presentation
Afternoon	Lecture, Opening PBL problem, Q&A	Team consulting	Review & discussion

Submission Due: 11:59pm Friday

PBL Class

: Instructor's lecture

: PBL team activity

ASAP

2:00pm

Intro

No early presentation

Presentation with slides (5min per team)

Opening PBL problem, Q&A

Team coding

Submission Due: 11:59pm Today