09-C

BST Application

Multi-Level Search Tree

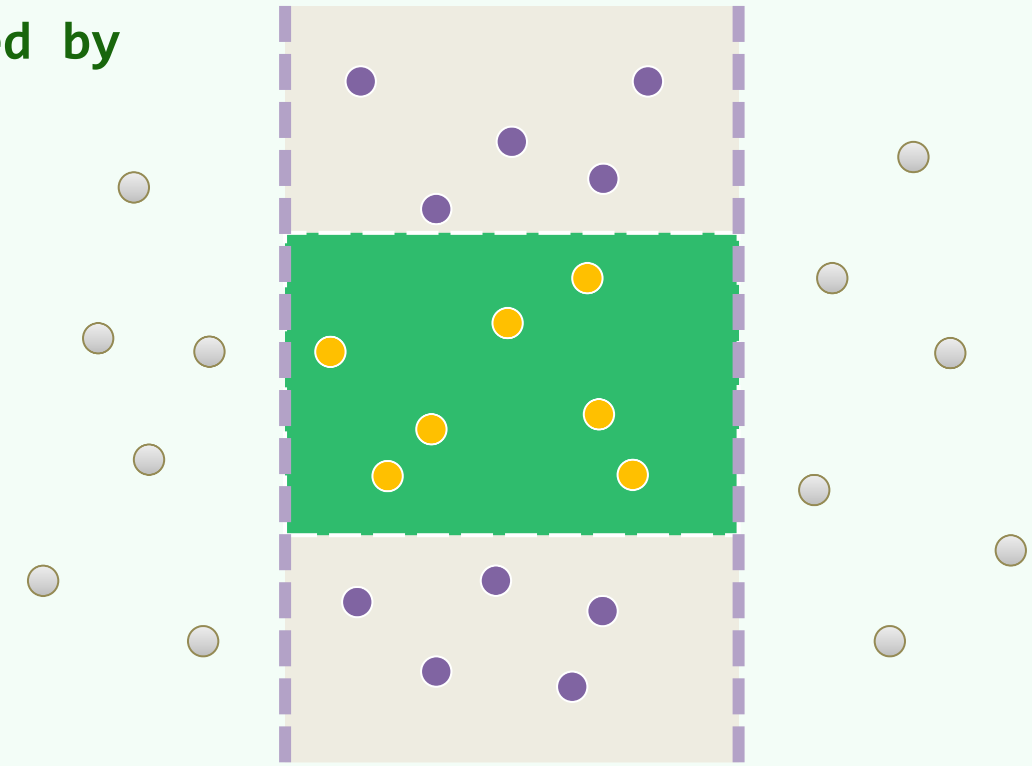
邓俊辉

deng@tsinghua.edu.cn

我们竟为这无用的找寻浪费了这么多天！我想找寻的心上人绝对不会在这里出现。

2D Range Query = x-Query + y-Query

- Is there any structure which answers range query FASTER than kd-trees?
- An m-D orthogonal range query can be answered by the INTERSECTION of m 1D queries
- For example, a 2D range query can be divided into two 1D range queries:
  - find all points in $[x_1, x_2]$; and then
  - find from these candidates those lying in $[y_1, y_2]$



Worst Cases

❖ Using kd-trees needs $\mathcal{O}(1+\sqrt{n})$ time. But here ...

❖ The x-query returns

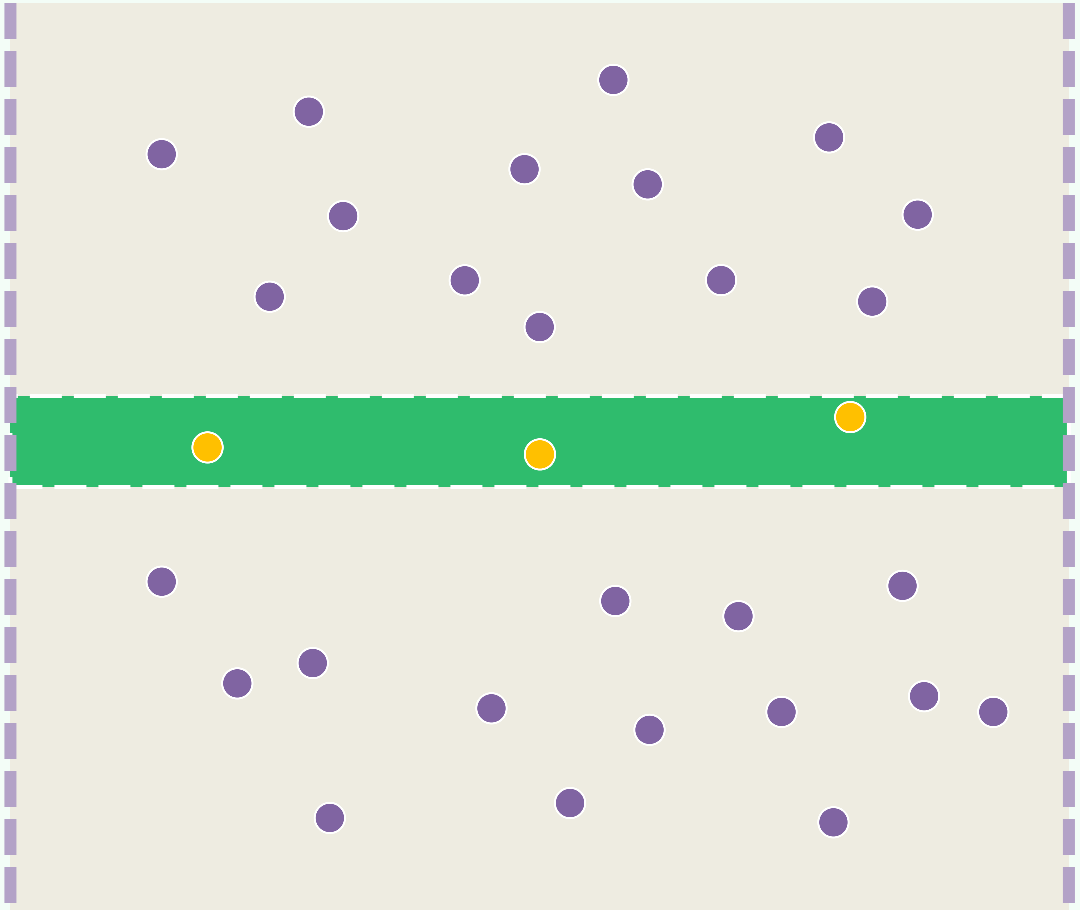
(almost) all points whereas

the y-query rejects

(almost) all

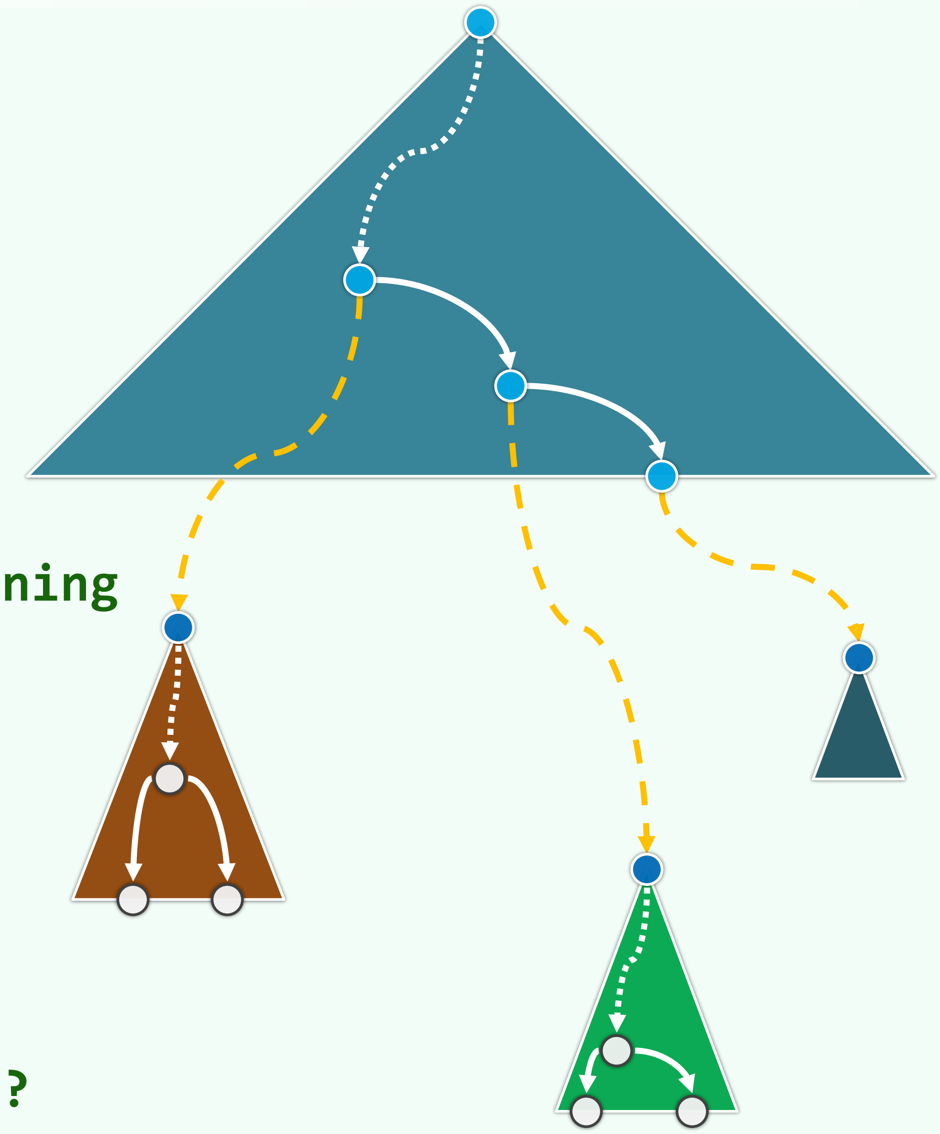
❖ We spent Ω(n) time

before getting r = 0 points

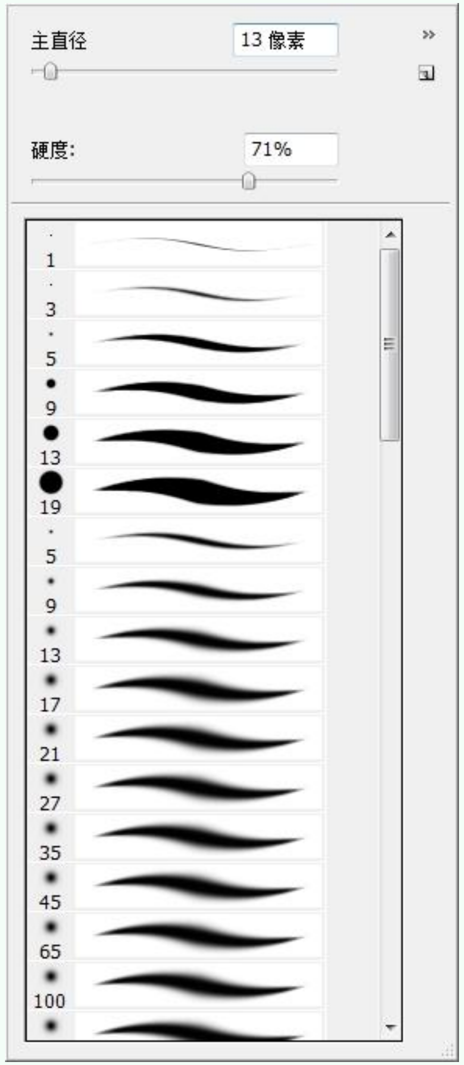


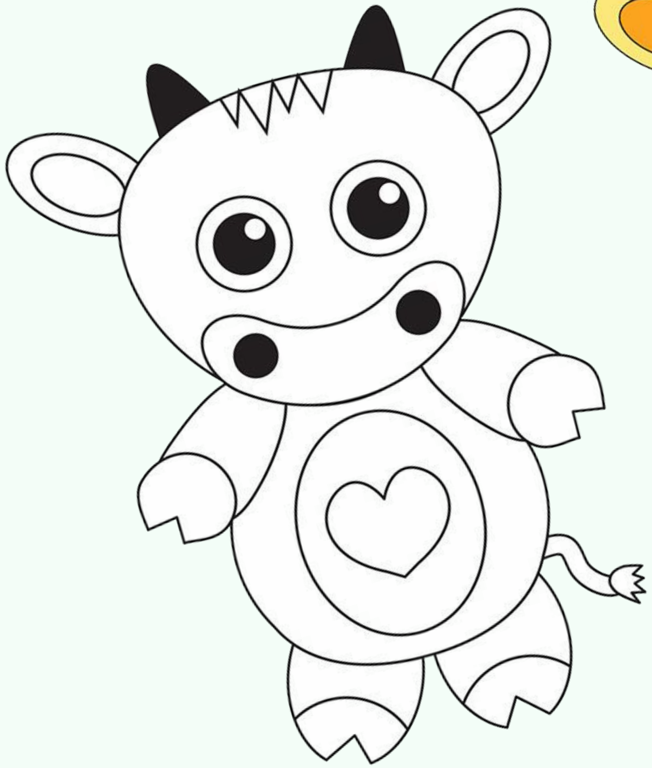
2D Range Query = x-Query * y-Query

- Tree of trees
  - build a 1D BBST (called x-tree) for the first range query (x-query);
  - for each node v in the x-range tree, build a y-coordinate BBST (y-tree), containing the canonical subset associate with v
- It's an x-tree of (a number of) y-trees, called a Multi-Level Search Tree
- How to answer range queries with such an MLST?



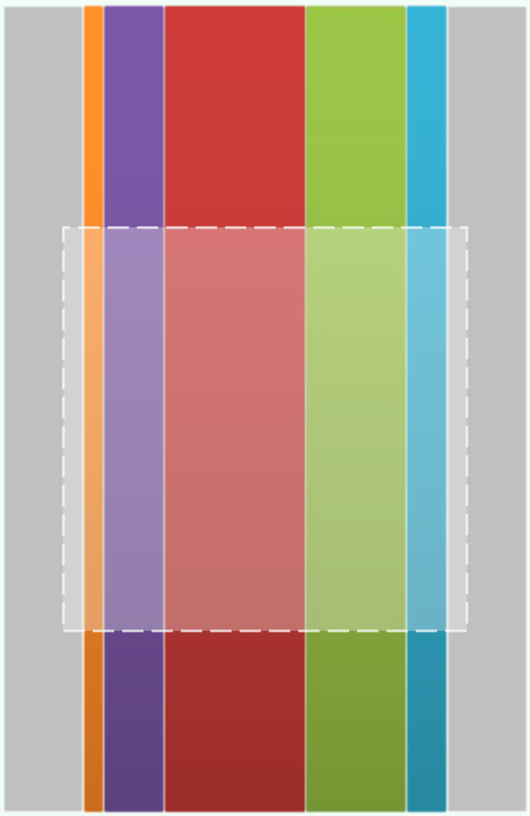
Painters' Strategy

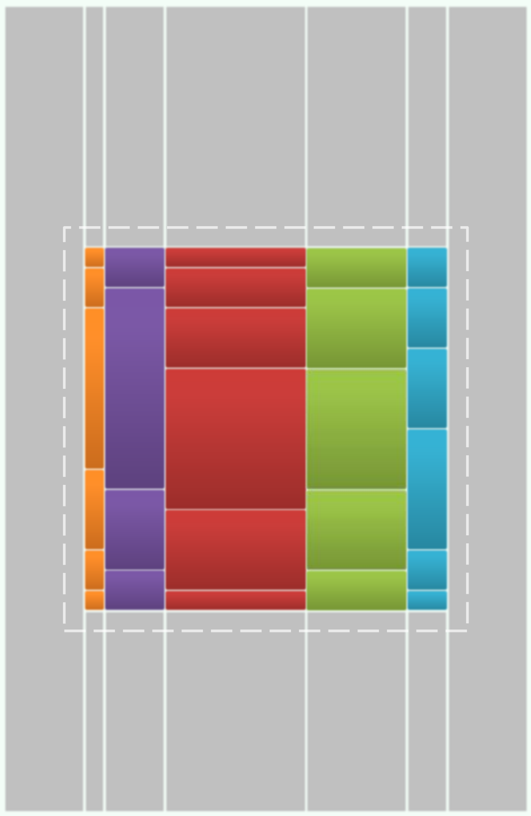


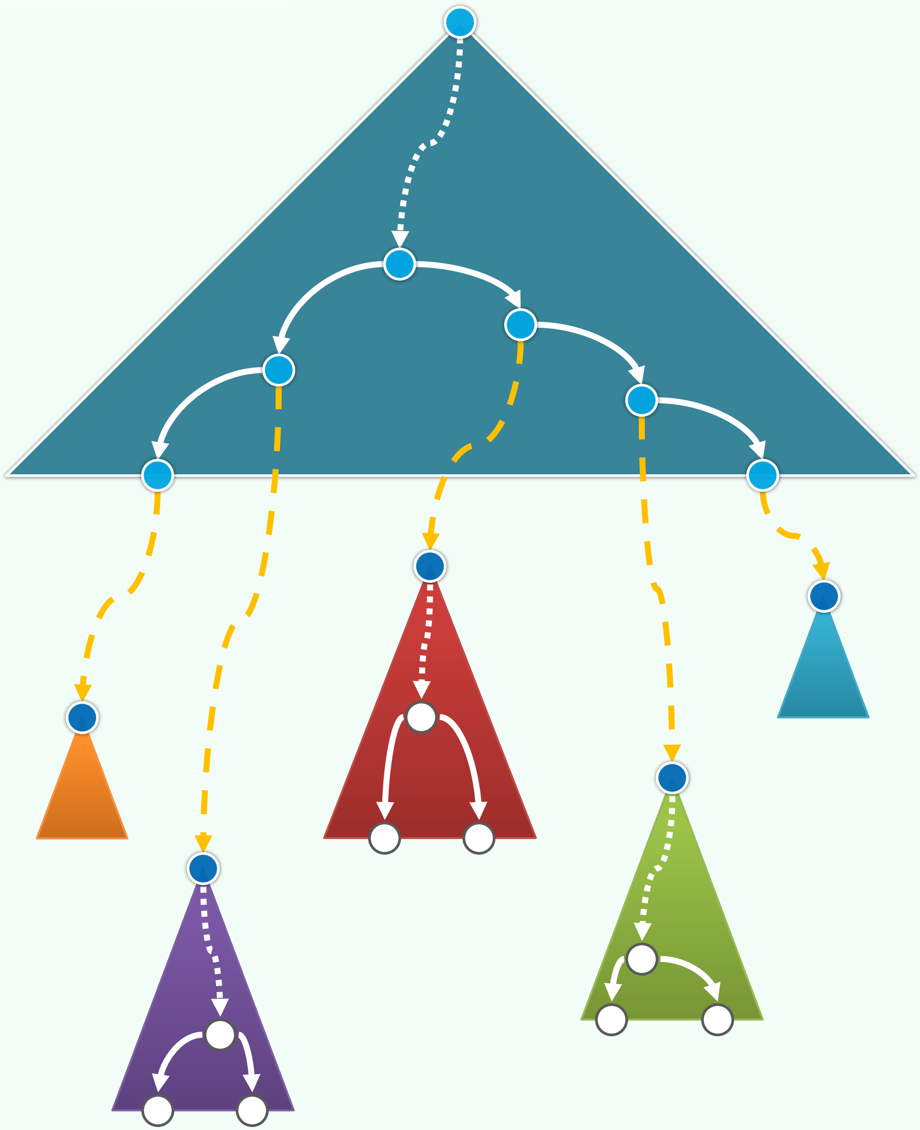


2D Range Query = x-Query * y-Queries

❖ Query Time = $\mathcal{O}(r + \log^2 n)$ ~ $\mathcal{O}(r + \log n)$







Query Algorithm

1. Determine the canonical subsets of points that satisfy the first query

   // there will be $\mathcal{O}(\log n)$ such canonical sets,

   //   each of which is just represented as

   //   a node in the x-tree

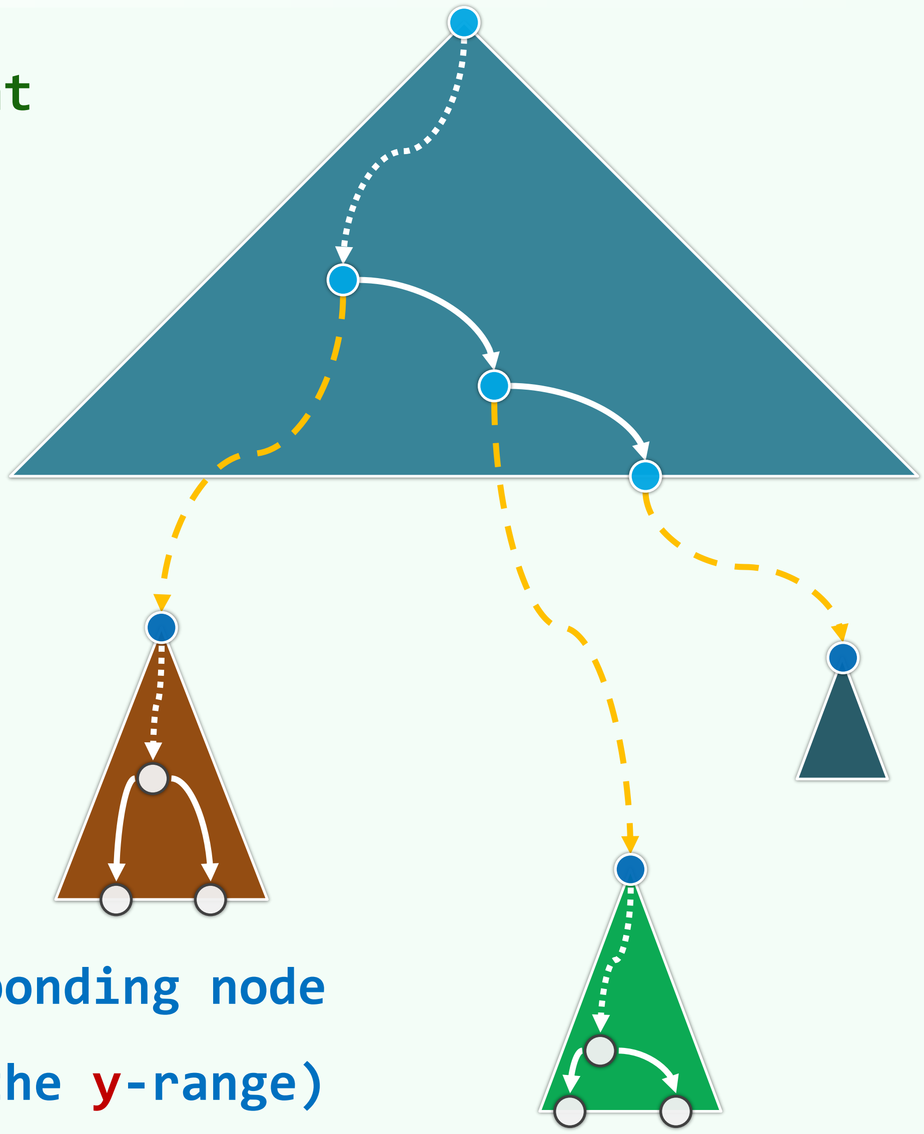
2. Find out from each canonical subset which points lie within the y-range

   // To do this,

   //   for each canonical subset,

   //   we access the y-tree for the corresponding node

   // this will be again a 1D range search (on the y-range)



Complexity: Preprocessing Time + Storage

- A 2-level search tree

  for n points in the plane

  can be built

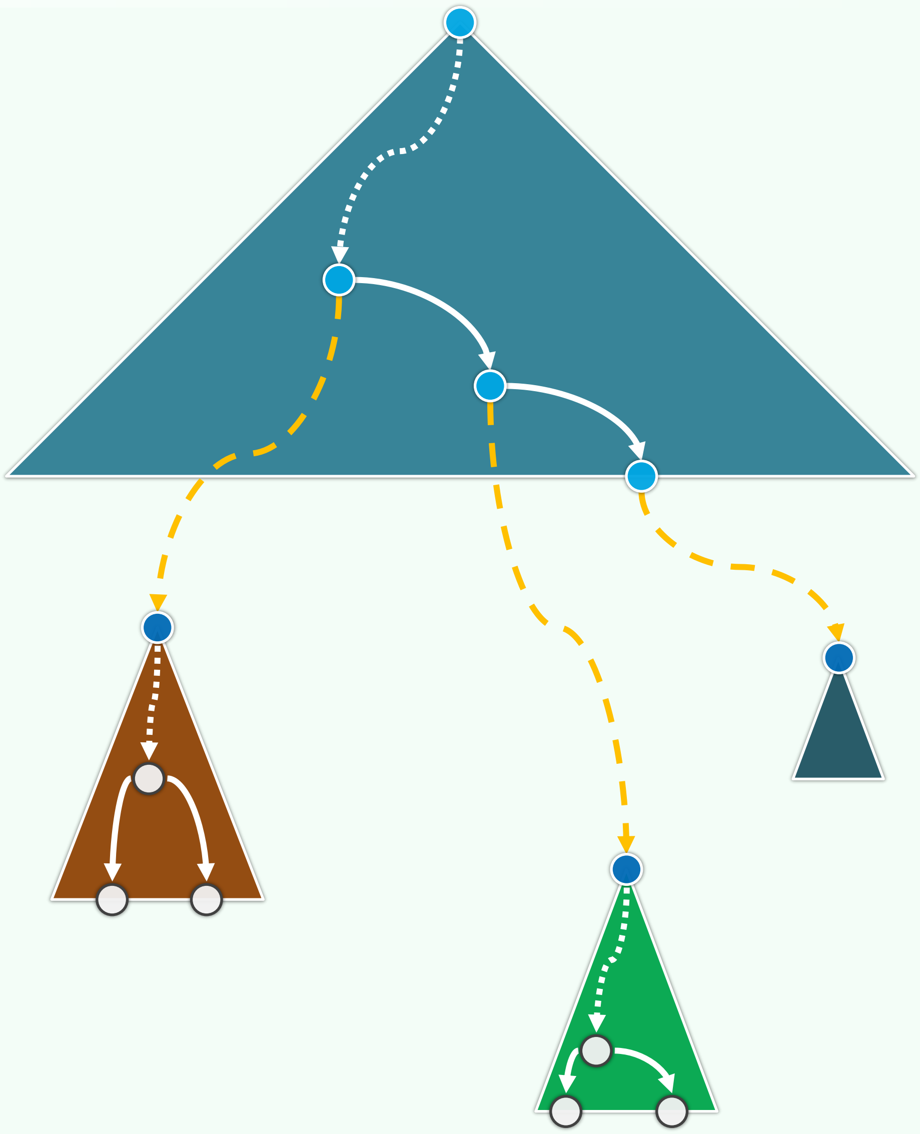
  in $\mathcal{O}(n \log n)$ time

- Each input point is stored in $\mathcal{O}(\log n)$ y-trees

- A 2-level search tree

  for n points in the plane

  needs $\mathcal{O}(n \log n)$ space



Complexity: Query Time

❖ Claim: A 2-level search tree for n points in the plane

answers each planar range query

in $\mathcal{O}(r + \log^2 n)$ time

❖ The x-range query needs $\mathcal{O}(\log n)$ time

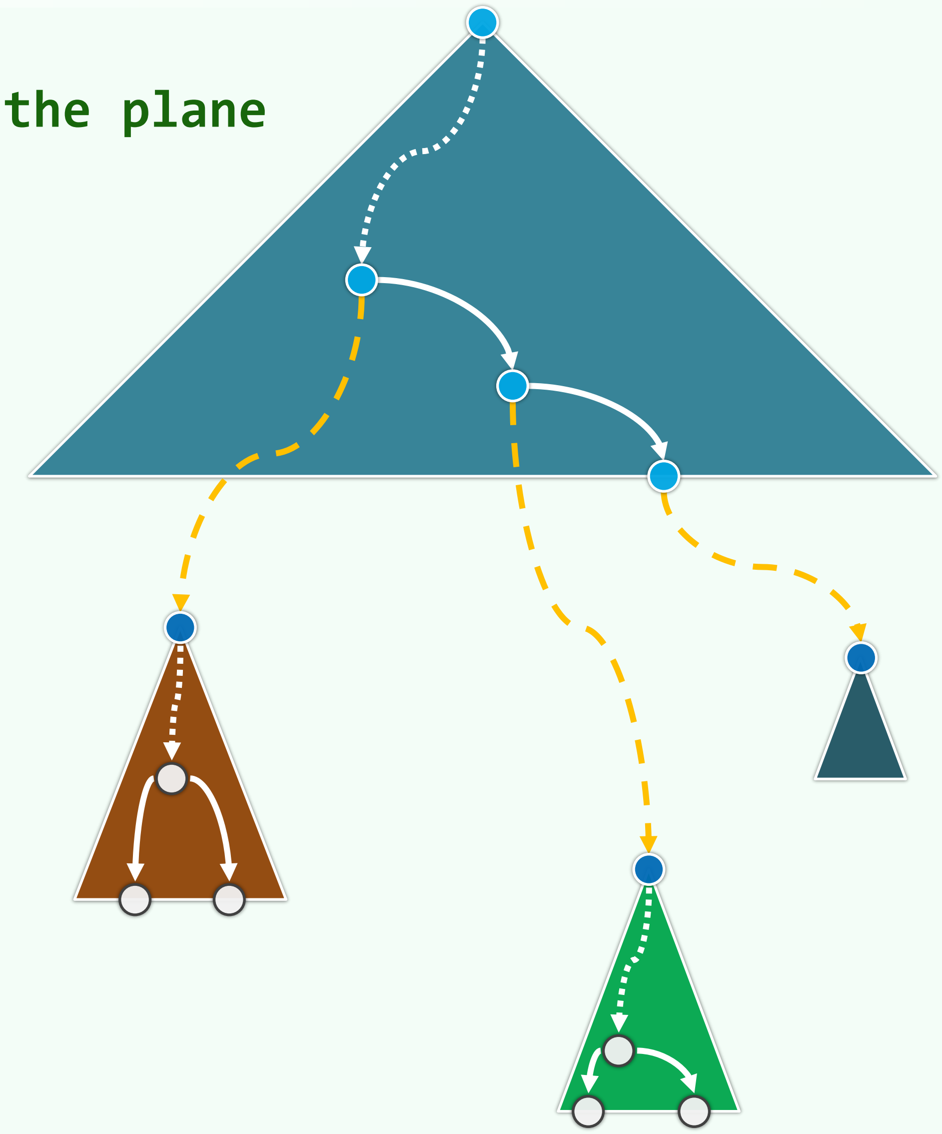
to locate the $\mathcal{O}(\log n)$ nodes

representing the canonical subsets

❖ Then for each of these nodes,

a y-range search is invoked,

which needs $\mathcal{O}(\log n)$ time



Beyond 2D

- Let S be a set of n points in $\mathcal{E}^d$, d ≥ 2
  - A d-level tree for S uses $\mathcal{O}(n \cdot \log^{d-1} n)$ storage
  - Such a tree can be constructed in $\mathcal{O}(n \cdot \log^{d-1} n)$ time
  - Each orthogonal range query of S can be answered in $\mathcal{O}(r + \log^{d} n)$ time
- For planar case, can the query time be improved to, say, $\mathcal{O}(\log n)$?