Range Queries and Trees

1. Search Kd Trees
   - two dimensional range queries
   - cost of searching a Kd tree
2. Range Trees
   - motivation, definition, and example
   - constructing a range tree
   - searching a range tree
3. Range Trees in CGAL
   - package overview

MCS 481 Lecture 14
Computational Geometry
Jan Verschelde, 14 February 2025

Range Queries and Trees

1. Search Kd Trees
   - two dimensional range queries
   - cost of searching a Kd tree
2. Range Trees
   - motivation, definition, and example
   - constructing a range tree
   - searching a range tree
3. Range Trees in CGAL
   - package overview

database queries

An example of a database query: list all employees within a given salary range and a number of children between 2 and 5.

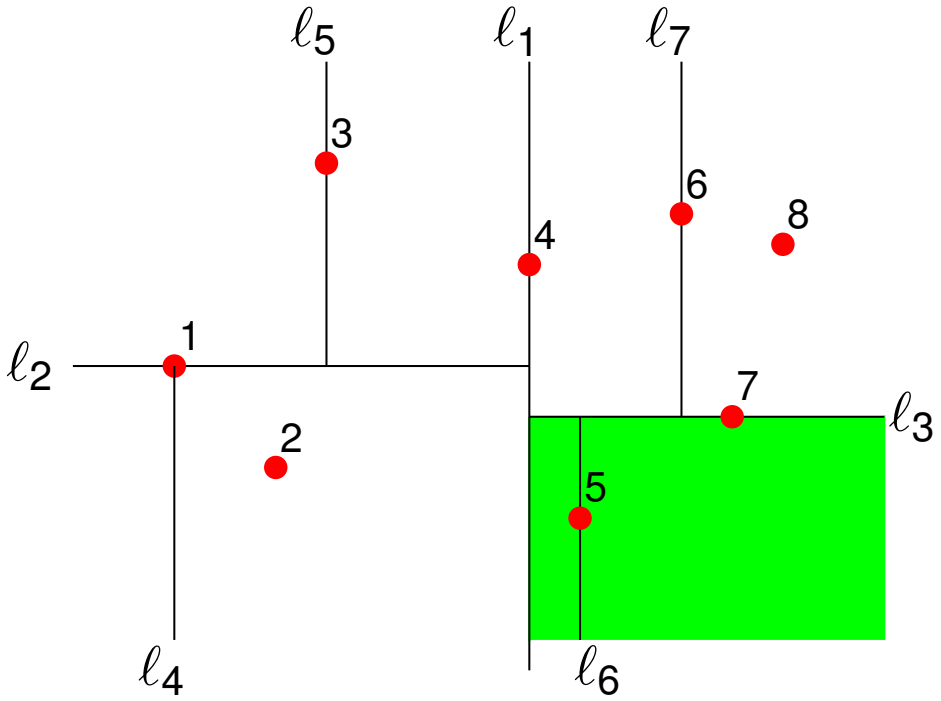
A Kd tree generalizes the binary search tree to 2 dimensions,

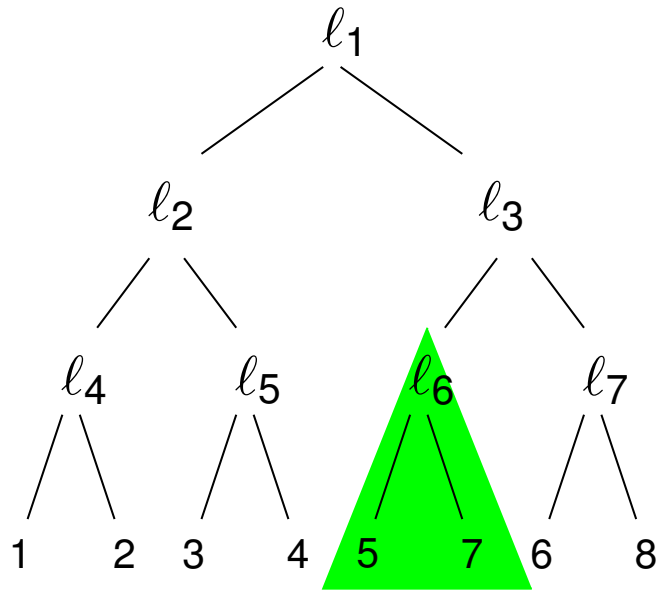
- obtained by recursively splitting the set of points,
- alternating between vertical and horizontal lines.

A Kd tree for n points uses $O(n)$ storage and
can be constructed in time $O(n \log(n))$.

In CGAL, the package dD Spatial Searching uses the Kd tree.

a region in a two dimensional Kd tree





Definition (region of a node in Kd tree)

A *region of a node in a Kd tree* is the intersection of

- the half plane defined by the splitting line to the node and
- the regions of the nodes in the path from the root to the node.

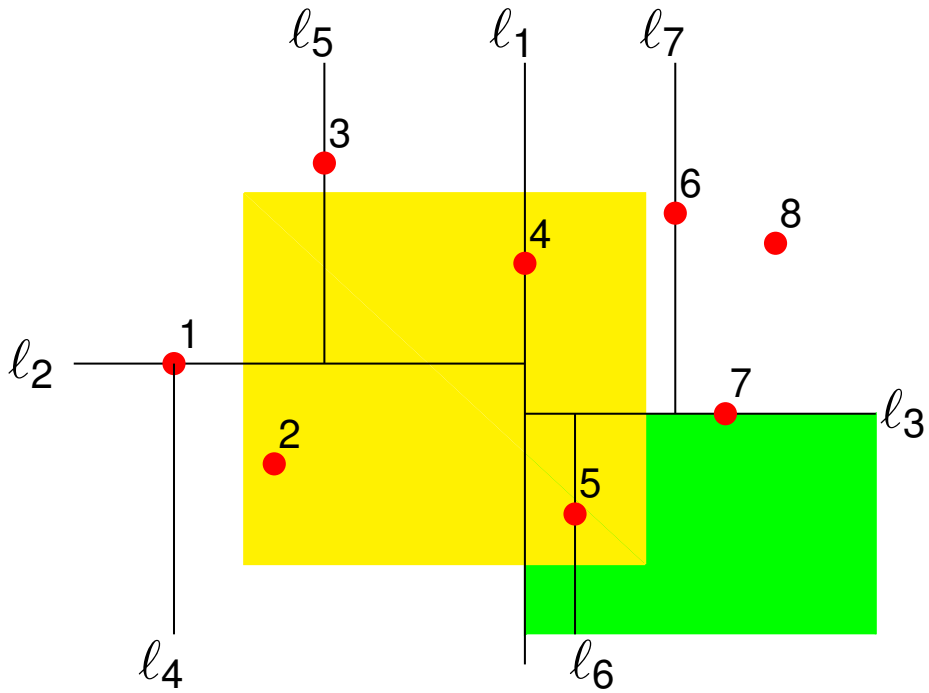
The region of the root node is the entire plane.

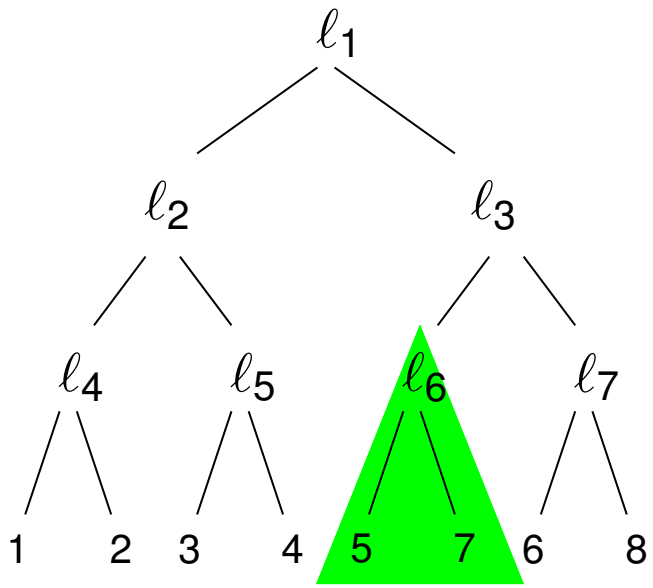
region and range

Definition (range)

A *range* is a rectangle $[x : x'] \times [y : y']$.

A range may be entirely contained in some region,
but it may also overlap with several regions:





In the picture above, the region is green, the range is yellow.

searching a Kd tree

Algorithm SEARCHKDTREE(v, R)

Input: v, a node in a Kd tree, $R = [x : x'] \times [y : y']$.
Output: all points below v in the range R.

1. if ISLEAF(v) then
2. report x_v if in R

else

3. if REGION(LEFTCHILD(v)) $\subseteq R$ then
4. REPORTSUBTREE(LEFTCHILD(v))
5. else if REGION(LEFTCHILD(v)) $\cap R \neq \emptyset$ then
6. SEARCHKDTREE(LEFTCHILD(v), R)

Searching RIGHTCHILD(v) is similar and continued on next slide.

searching RIGHTCHILD(v)

7. if REGION(RIGHTCHILD(v)) $\subseteq R$ then
8. REPORTSUBTREE(RIGHTCHILD(v))
9. else if REGION(RIGHTCHILD(v)) $\cap R \neq \emptyset$ then
10. SEARCHKDTREE(RIGHTCHILD(v), R)

Exercise 1: In the algorithm SEARCHKDTREE, the tests

- REGION(CHILD(v)) $\subseteq R$, and
- REGION(CHILD(v)) $\cap R \neq \emptyset$

are written in their mathematical notation.
Describe in sufficient detail the instructions for these tests.
In particular, do these tests run in $O(1)$ time?

Exercise 2: Verify the correctness of SEARCHKDTREE.

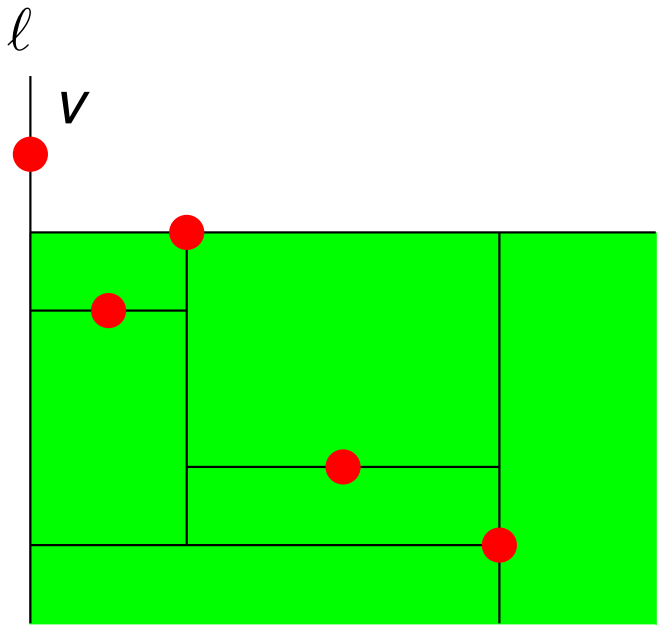
Range Queries and Trees

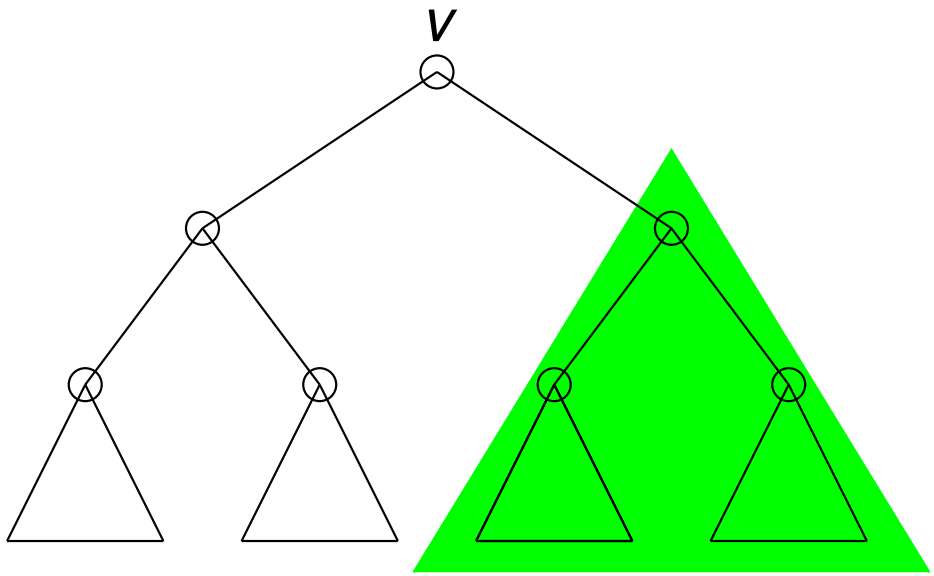
1. Search Kd Trees
   - two dimensional range queries
   - cost of searching a Kd tree
2. Range Trees
   - motivation, definition, and example
   - constructing a range tree
   - searching a range tree
3. Range Trees in CGAL
   - package overview

cost of searching a Kd tree

Lemma (cost of SEARCHKDTREE)

For n points and output size k, SEARCHKDTREE *runs in $O(\sqrt{n}+k)$.*



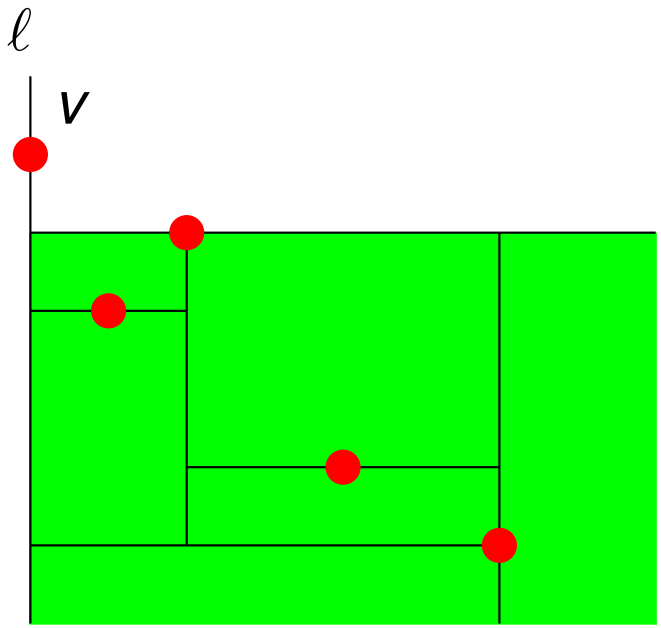


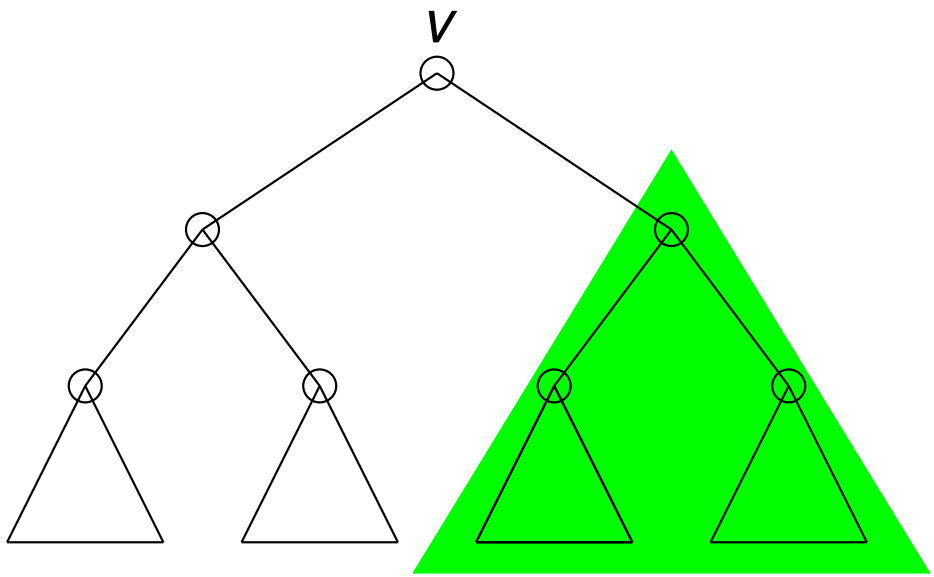
If the query range involves REGION(RIGHTCHILD(v)), then 2 of the 4 nodes are intersected in the Kd tree.

deriving a recurrence

Lemma (cost of SEARCHKDTREE)

For n points and output size k, SEARCHKDTREE *runs in $O(\sqrt{n}+k)$.*





If the n points are distributed evenly as in a balanced tree, then each of the 2 nodes intersected involve $n/4$ points.

a recurrence

Lemma (cost of SEARCHKDTREE)

For n points and output size k, SEARCHKDTREE *runs in $O(\sqrt{n}+k)$.*

The running time $T(n)$ depends on the number of visited nodes.

$T(n)$ satisfies the recurrence

$$T(n) = \begin{cases} O(1) & \text{if} \quad n = 1, \\ 2 + T(n/4) & \text{if} \quad n > 1. \end{cases}$$

The solution of the recurrence is $O(\sqrt{n})$.

Any vertical line intersects $O(\sqrt{n})$ regions in the Kd tree.

the performance of Kd trees

Theorem (performance of Kd trees)

A Kd tree for n points

- *uses $O(n)$ storage, and*
- *takes $O(n \log(n))$ time to build.*

A rectangular range query takes $O(\sqrt{n} + k)$ operations, where k is the output size.

Kd trees can be applied for points in dimensions 3 and higher.

The query time for a d-dimensional Kd tree is $O(n^{1-1/d} + k)$, where k is the output size.

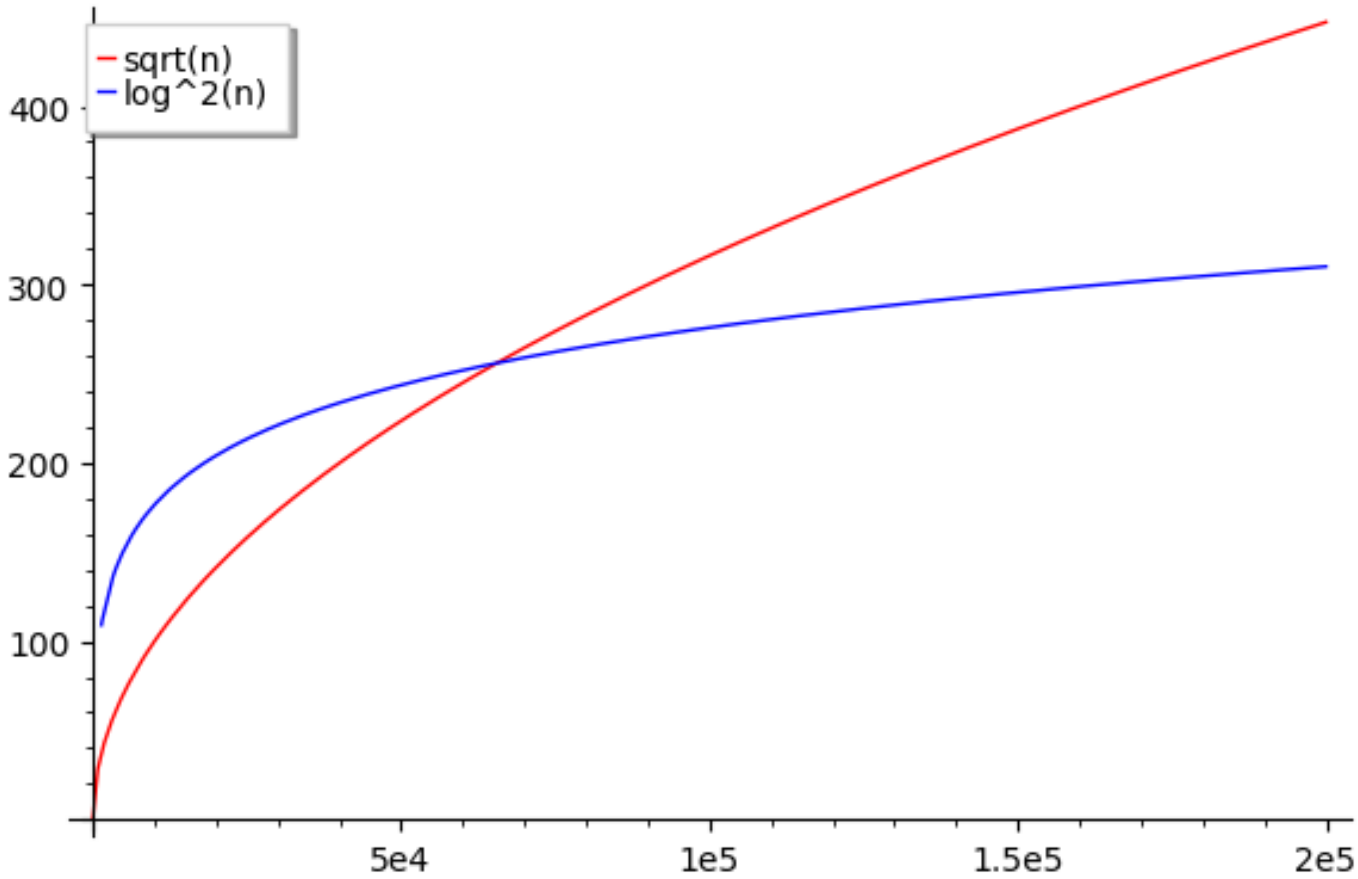
Range Queries and Trees

1. Search Kd Trees
   - two dimensional range queries
   - cost of searching a Kd tree
2. Range Trees
   - motivation, definition, and example
   - constructing a range tree
   - searching a range tree
3. Range Trees in CGAL
   - package overview

reduce the query time

The Kd tree is a geometric data structure, introduced as a generalization of balanced binary search trees, for two dimensions.

For n points and output size k, the query time is $O(\sqrt{n} + k)$.
Our motivation is to reduce the query time to $O(\log^2(n) + k)$.



definition of the range tree

Definition (range tree)

For a set of points P, a *range tree* is a tree:

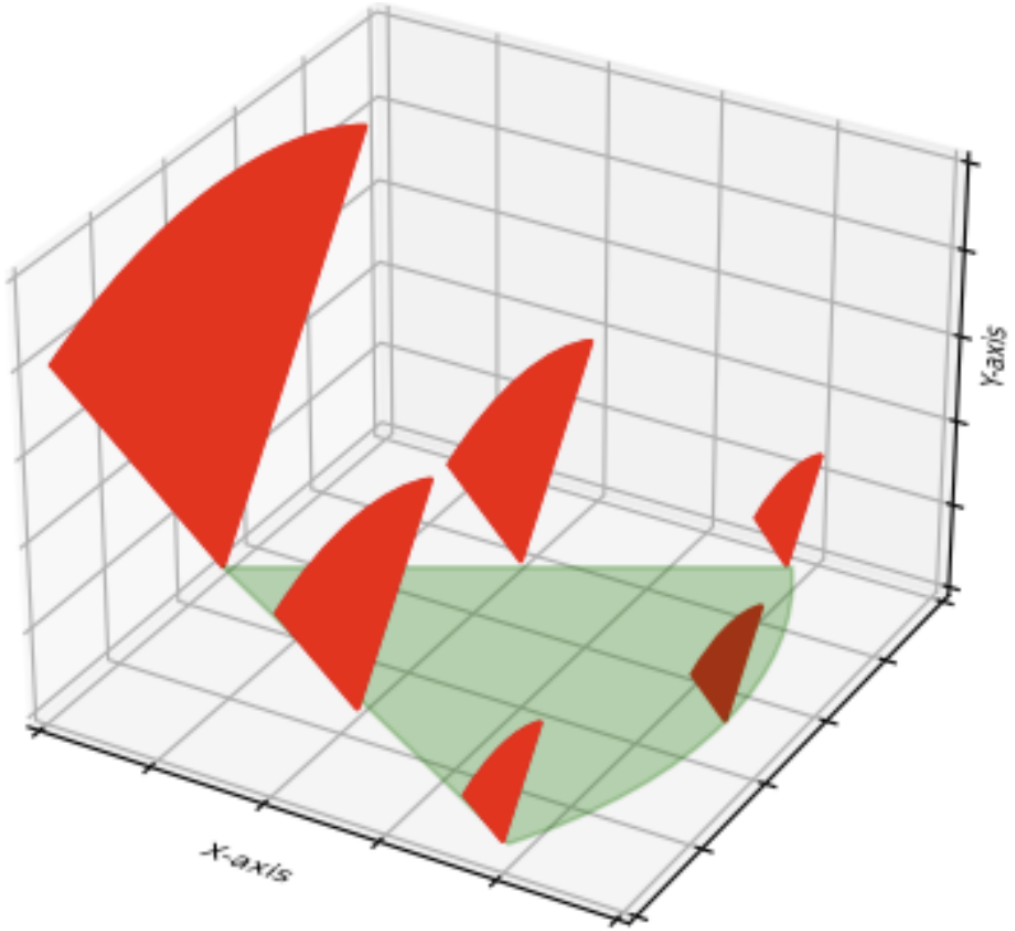
1. The main tree T is a balanced binary tree on the x-coordinate.
2. Every node v in T has an associated tree $T_{\text{assoc}}(v)$.

The associated tree $T_{\text{assoc}}(v)$ to a node v in T is

- a balanced binary tree on the y-coordinate,
- containing only those points of P with those x-coordinate stored in leaves under the node of the main tree.

Data structures where nodes have pointers to associated structures are often called *multi-level data structures*.

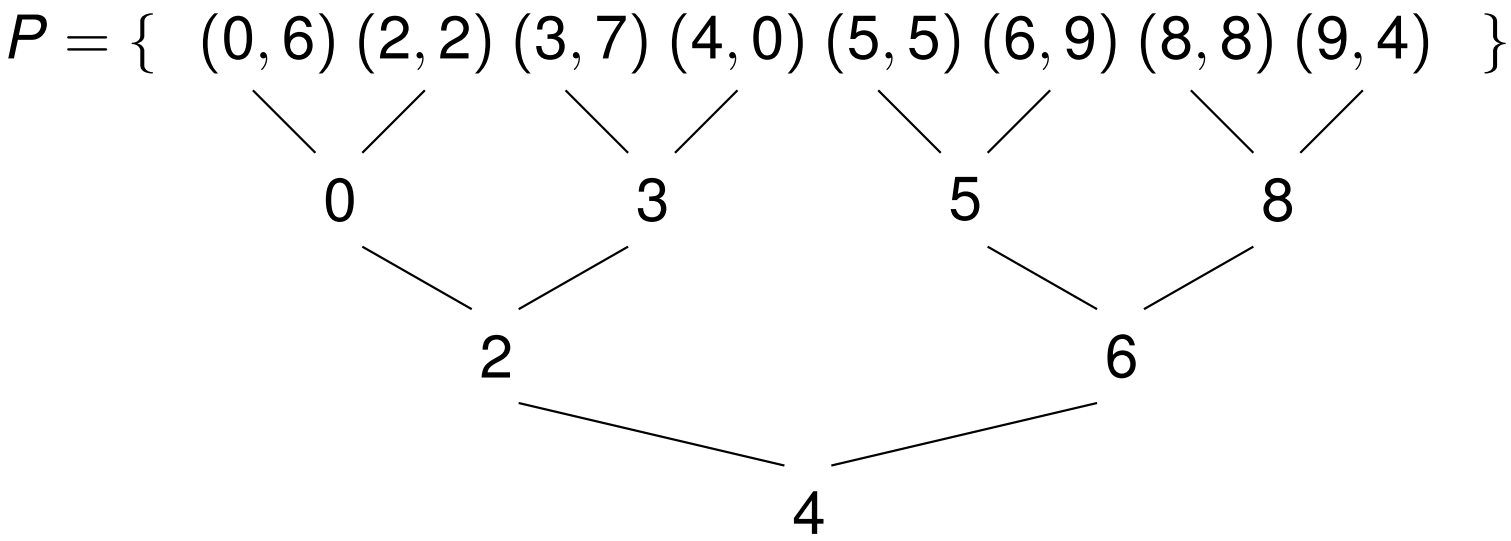
a range tree is a two-level data structure



The green wedge represents the points sorted on the x-coordinate.
The red wedges represent the points sorted on the y-coordinate.

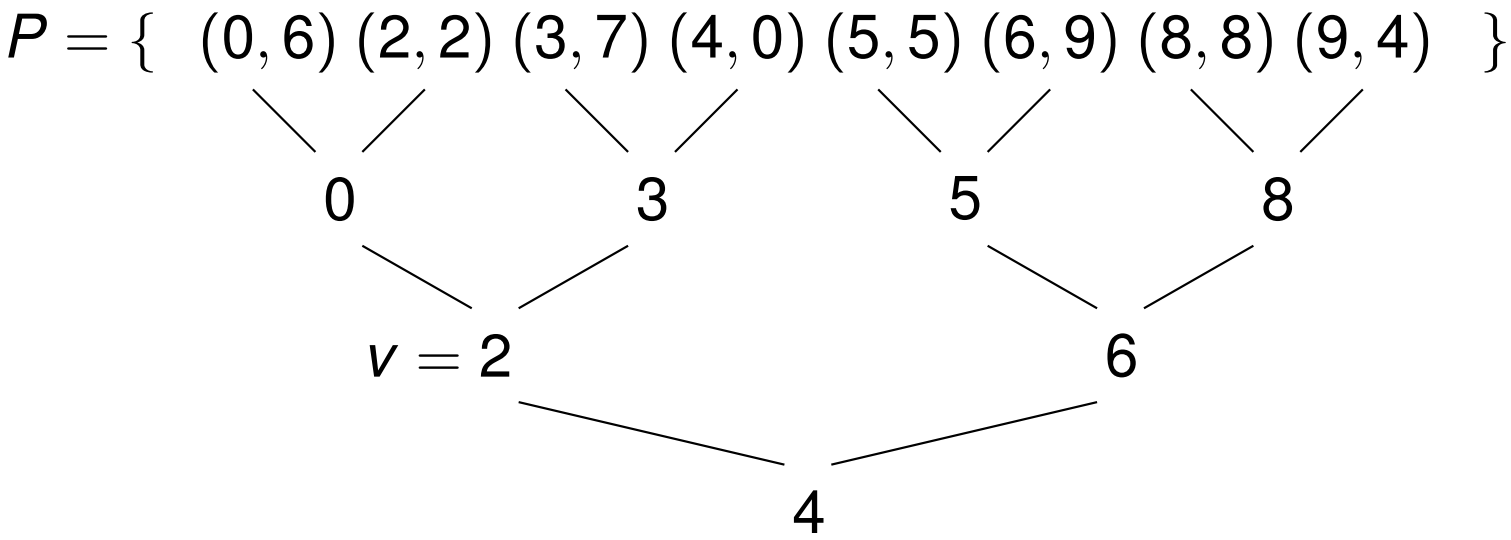
an example

For $P = \{(0,6),(2,2),(3,7),(4,0),(5,5),(6,9),(8,8),(9,4)\}$,
the balanced binary tree on the x-coordinate is

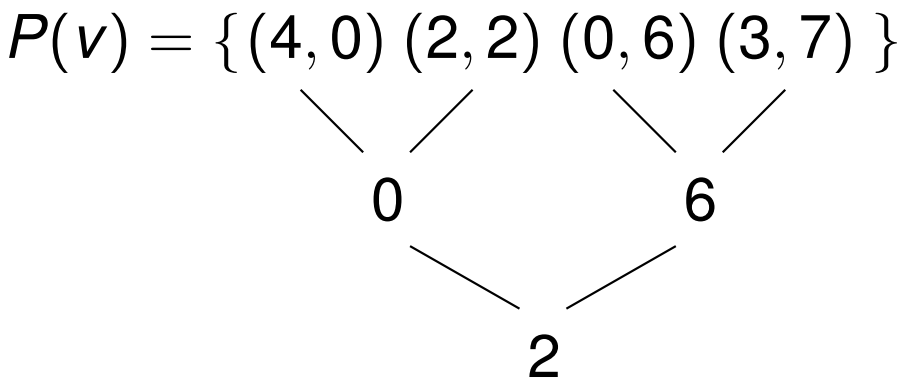


Consider $v = 2$, the set $\{(0,6),(2,2),(3,7),(4,0)\}$,
ordered on the y-coordinate is $P(v) = \{(4,0),(2,2),(0,6),(3,7)\}$.

the tree associated to a node



The tree $T_{\text{assoc}}(v)$ associated to $v = 2$:



Range Queries and Trees

1. Search Kd Trees
   - two dimensional range queries
   - cost of searching a Kd tree
2. Range Trees
   - motivation, definition, and example
   - constructing a range tree
   - searching a range tree
3. Range Trees in CGAL
   - package overview

constructing a range tree

Algorithm BUILD2DRANGETREE(P)

Input: P, a set of points in the plane.
Output: root of a 2D range tree.

1. Construct T_{assoc} for P on y-coordinate.
2. if $\#P = 1$ then
3. create leaf with point in P and T_{assoc}

   else
4. $(P_{\text{left}}, P_{\text{right}}) = \text{SPLITX}(P)$
5. $v_{\text{left}} = \text{BUILD2DRANGETREE}(P_{\text{left}})$
6. $v_{\text{right}} = \text{BUILD2DRANGETREE}(P_{\text{right}})$
7. create node v with v_{left}, T_{assoc}, v_{right}
8. return v

storage of a range tree

The construction cost for the range tree is $O(n \log(n))$,
as we presort the points on their x- and y-coordinate.

Lemma (storage of a range tree)

A range tree for n points requires $O(n \log(n))$ storage.

- The $\log(n)$ factor is the depth of the main tree.
- The n factor is from the storage cost of a one dimensional balanced binary search tree, associated with every node.

Range Queries and Trees

1. Search Kd Trees
   - two dimensional range queries
   - cost of searching a Kd tree
2. Range Trees
   - motivation, definition, and example
   - constructing a range tree
   - searching a range tree
3. Range Trees in CGAL
   - package overview

searching a range tree

Algorithm 2DRANGEQUERY($T, [x : x'] \times [y : y']$)
Input: 2D range tree T and a query range.
Output: all points in T in $[x : x'] \times [y : y']$.

1. $v_{\text{split}} = \text{FINDSPLITNODE}(T, x, x')$
2. if ISLEAF(v_{split}) then
3. check if v_{split} should be reported
   else
4. $v = \text{LEFTCHILD}(v_{\text{split}})$
5. while not ISLEAF(v) do
6. if $x \leq x_v$ then
7. 1DRANGEQUERY($T_{\text{assoc}}(\text{RIGHTCHILD}(v)), [y : y']$)
8. $v = \text{LEFTCHILD}(v)$
   else
9. $v = \text{RIGHTCHILD}(v)$

algorithm 2DRANGEQUERY continued

10. check if point stored in v should be reported
11. $v = \text{RIGHTCHILD}(v_{\text{split}})$
12. while not $\text{ISLEAF}(v)$ do
13. if $x' > x_v$ then
14. $\text{1DRANGEQUERY}(T_{\text{assoc}}(\text{LEFTCHILD}(v)), [y : y'])$
15. $v = \text{RIGHTCHILD}(v)$
    else
16. $v = \text{LEFTCHILD}(v)$

Exercise 3: Verify the correctness of 2DRANGEQUERY.

cost of 2DRangeQuery

Lemma (cost of searching a range tree)

Searching a range tree for n points has running time $O(\log^2(n) + k)$, where k is the output size.

The number of nodes visited is

$$\sum_v O(\log(n) + k_v) = O(\log^2(n)) + \underbrace{\sum_v k_v}_{=k}.$$

Range Queries and Trees

1. Search Kd Trees
   - two dimensional range queries
   - cost of searching a Kd tree
2. Range Trees
   - motivation, definition, and example
   - constructing a range tree
   - searching a range tree
3. Range Trees in CGAL
   - package overview

Range Trees in CGAL

At `https://doc.cgal.org`, starting at the Package Overview, navigate to the dD Range and Segment Trees section.

Looking at the User Manual of dD Range Trees: consider the figure of a two dimensional and a d-dimensional range tree.

The wiki of `cgal-swig-bindings` on the available CGAL packages does not (yet) list the range trees.

summary and exercises

We covered the second half of section 5.2
and section 5.3 in the textbook.

Consider the following activities, listed below.

1. Write the solutions to exercises 1, 2, and 3.
2. Consult the CGAL documentation and example code for working with Kd and range trees.
3. Consider the exercises 6, 8, 9 in the textbook.