Klee’s measure problem

Hyunjoon Cheon

Oct. 17, 2024

Overview

Back to the basics

Find the colored area

$A = ?$

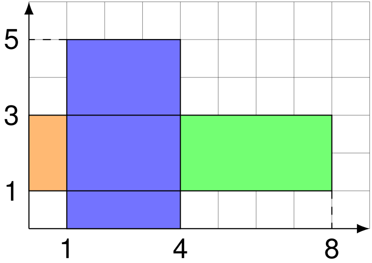
Back to the basics

Find the colored area

$A = 25$

Back to the basics

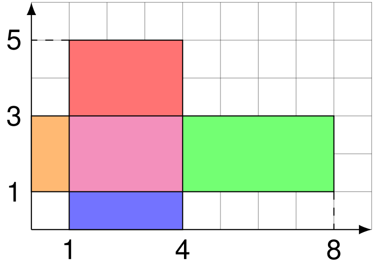
Find the colored area



$$A = 25 = 2 + 15 + 8$$

Back to the basics

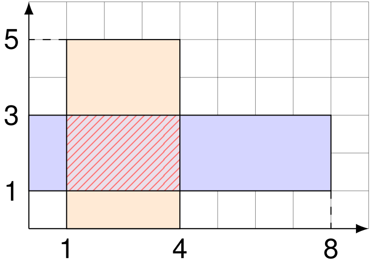
Find the colored area



$$A = 25 = 2 + 6 + 6 + 3 + 8$$

Back to the basics

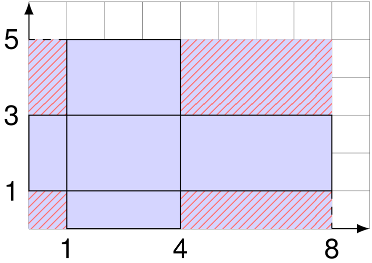
Find the colored area



$$A = 25 = 16 + 15 - 6$$

Back to the basics

Find the colored area



$A = 25 = 40 - 15$

Today's Topic

Problem (Klee's measure problem)

Given a set $B = \{b_1, b_2, \ldots, b_n\}$ *of* n d*-dimensional boxes, find the volume of the union of all boxes in* B.

Definition (Hyperrectangle a.k.a. *Box*)

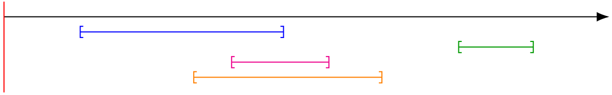
A hyperrectangle is a Cartesian product of finite intervals.

What we will discuss

- Two space partitioning approaches toward Klee's measure problem.

The first steps: 1-dim. case

Klee, 1977



The first steps: 1-dim. case

Klee, 1977

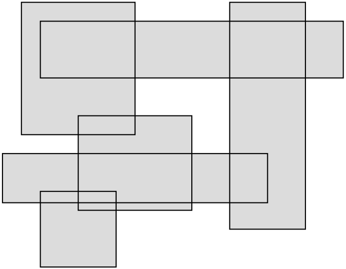
This trivial algorithm

1. utilizes sweeping
2. uses no special data structures
3. runs in
   - $O(n \log n)$-time[1] ($O(n)$ w/ sorted input)
   - $\Theta(n)$-space ($O(1)$ w/ sorted input)

[1] Actually $O(n \log p)$, where p is the minimum number of lines stabbing all intervals

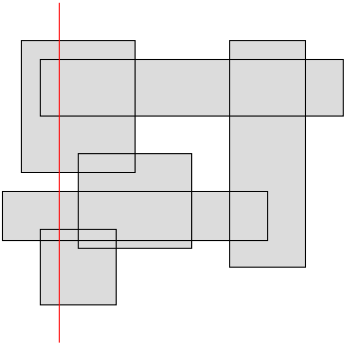
The first steps: 2-dim. case

Bentley, 1977



The first steps: 2-dim. case

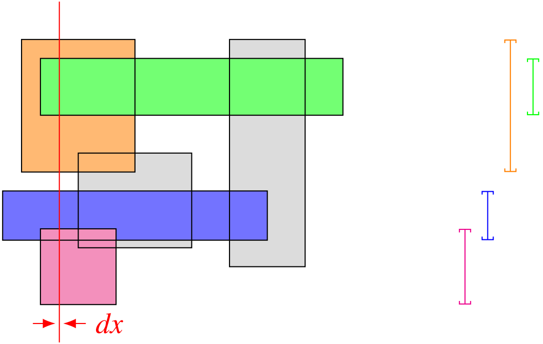
Bentley, 1977



Try sweeping!

The first steps: 2-dim. case

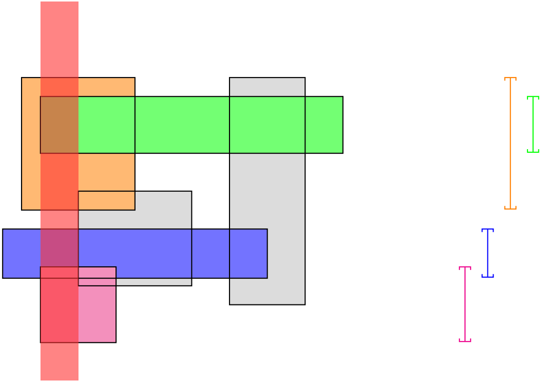
Bentley, 1977



The volume differential is $dx \cdot$ (measure of the cross section)

The first steps: 2-dim. case

Bentley, 1977



Measure intervals and compute the volume of the slab

The first steps: 2-dim. case

Bentley, 1977

This algorithm[2]

1. uses sweeping
2. uses segment tree maintaining partial measure of corresponding intervals
3. runs in
   - $O(n \log n)$-time (at most $2n$ updates on segment tree)
   - $\Theta(n)$-space

[2] The article is unpublished. Referred to Leeuwen and Wood, 1981 instead.

Beyond 3-dimension

- Bentley's algorithm easily extends to d-dimensional boxes, where $d \geq 2$ is arbitrary integer
- Shows $O(n^{d-1} \log n)$ running time.

- Unfortunately, the known (non-tight) lower bound is $\Omega(n \log n)$ regardless of the dimensionality.
- How can we design improved algorithms?

Space partitioning

- *Partition*ing a given (euclidean) space.

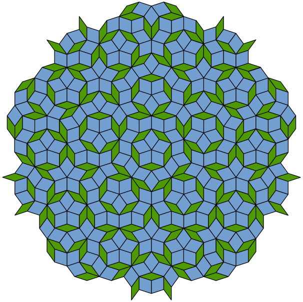


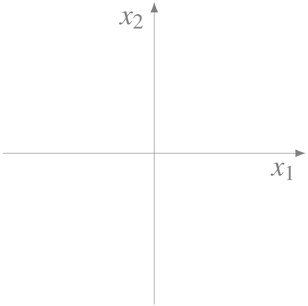
Figure: Penrose tiling

Data structures

- k-d tree
- Quadtree/Octree
- $\text{B}_{\text{inary}}\text{S}_{\text{pace}}\text{P}_{\text{artitioning}}$ tree
- and many others...

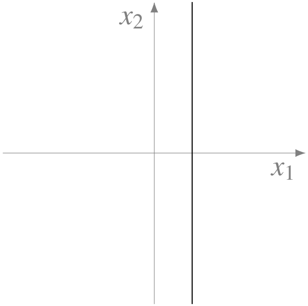
Data structures

- k-d tree
- Quadtree/Octree
- BSP tree
- and many others...



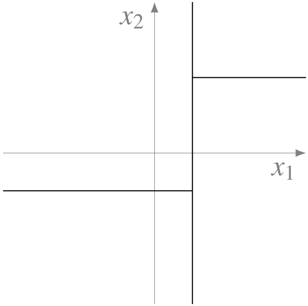
Data structures

- *k*-d tree
- Quadtree/Octree
- BSP tree
- and many others...



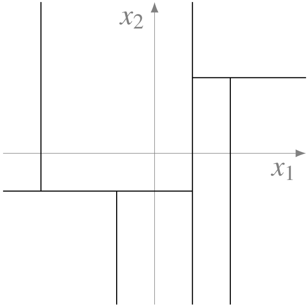
Data structures

- k-d tree
- Quadtree/Octree
- BSP tree
- and many others...



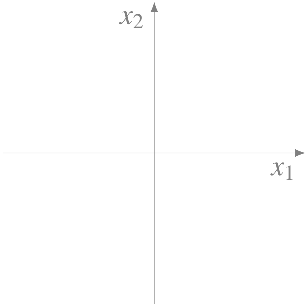
Data structures

- *k*-d tree
- Quadtree/Octree
- BSP tree
- and many others...



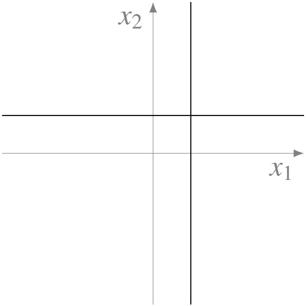
Data structures

- k-d tree
- Quadtree/Octree
- BSP tree
- and many others...



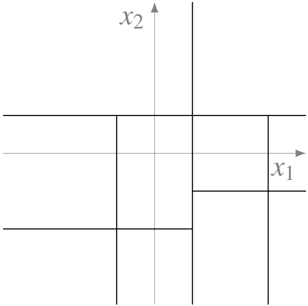
Data structures

- k-d tree
- Quadtree/Octree
- BSP tree
- and many others...



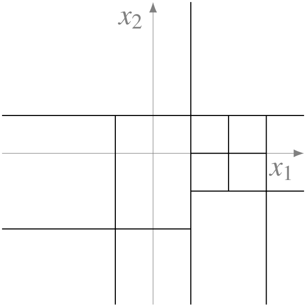
Data structures

- k-d tree
- Quadtree/Octree
- BSP tree
- and many others...



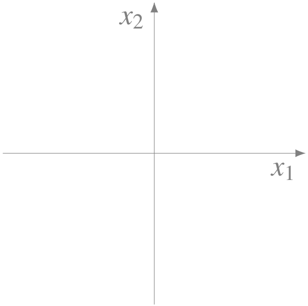
Data structures

- k-d tree
- Quadtree/Octree
- BSP tree
- and many others...



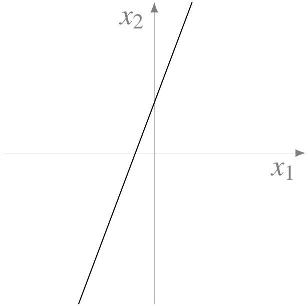
Data structures

- k-d tree
- Quadtree/Octree
- BSP tree
- and many others...



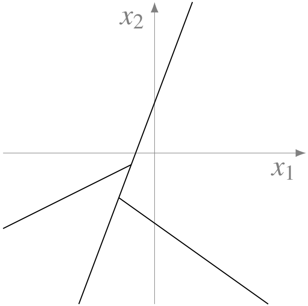
Data structures

- k-d tree
- Quadtree/Octree
- BSP tree
- and many others...



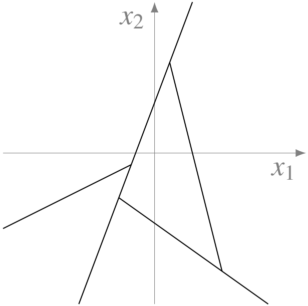
Data structures

- k-d tree
- Quadtree/Octree
- BSP tree
- and many others...



Data structures

- k-d tree
- Quadtree/Octree
- BSP tree
- and many others...



Which one is effective?

- Problem-by-problem.
- Key is *how* to partition the space.
  Note: the sweeping algorithms effectively partition the space into slabs in which some “characteristics” are preserved.
- We will see two approaches for the Klee’s measure problem.

Approach 1

Overmars and Yap, 1991

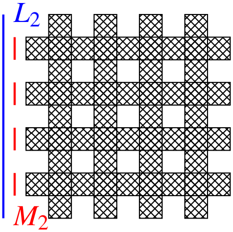
Key ideas

The linear increase ($O(n^{d-1} \log n)$) w.r.t. dimensionality is due to the recursive structure of the computation.
Suppose we sweep the whole space along the last (x_d) axis. Can we maintain the $(d-1)$-dim. cross section in a single data structure?

Idea

Figure: Trellis (취병)

Idea



The volume is $\prod L_i - \prod(L_i - M_i)$ and computing L_i's and M_i's are simple problems.

Divide the space into this trellis pattern!

Data structure

Orthogonal partition tree

- A balanced binary tree.
- A node α has an associated region C_α.; C_{root} is the whole space.
- For any two children α_1 and α_2 of a node α, $\{C_{\alpha_1}, C_{\alpha_2}\}$ is a partition of C_α.

Data structure

Orthogonal partition tree for Klee's measure problem

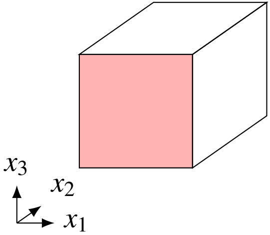
- M_α is the total measure under the subtree rooted at α.
- T_α is the number of boxes covering whole C_α but not $C_{parent(\alpha)}$.
  - If $T_\alpha > 0$ (C_α is fully covered), $M_\alpha = V(C_\alpha)$
  - Else $M_\alpha = M_{left(\alpha)} + M_{right(\alpha)}$
- A leaf λ maintains a set B_λ of boxes that intersect with the interior of C_λ but do not cover $C_{parent(\lambda)}$.

Terms

Definition (i-boundary)

For a d-box, its i-boundaries are its $(d-1)$-dim. faces perpendicular to x_i-axis. Note that a d-box has two i-boundaries for all $1 \leq i \leq d$.

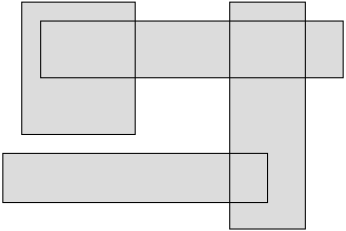
Example



The red face $[0,1] \times \{0\} \times [0,1]$ is a 2-boundary of the unit cube.

Partition strategy

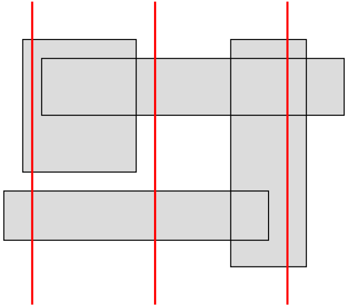
3-dim. case; 2-dim. projection



1. Split x_1-axis into $2\sqrt{n}$ intervals such that each contains at most $\sqrt{n}$ 1-boundaries.
2. For each 1-boundaries contained in a slab, split the slab along its 2-boundaries
3. For all other 2-boundaries, split along the $\sqrt{n}$-th 2-boundaries.

Partition strategy

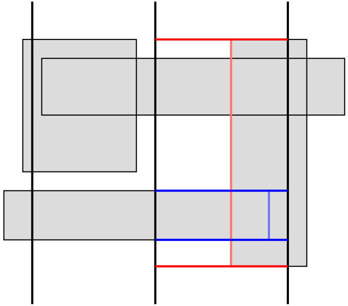
3-dim. case; 2-dim. projection



1. Split x_1-axis into $2\sqrt{n}$ intervals such that each contains at most $\sqrt{n}$ 1-boundaries.
2. For each 1-boundaries contained in a slab, split the slab along its 2-boundaries
3. For all other 2-boundaries, split along the $\sqrt{n}$-th 2-boundaries.

Partition strategy

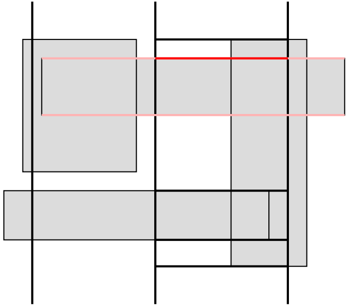
3-dim. case; 2-dim. projection



1. Split x_1-axis into $2\sqrt{n}$ intervals such that each contains at most $\sqrt{n}$ 1-boundaries.
2. For each 1-boundaries contained in a slab, split the slab along its 2-boundaries
3. For all other 2-boundaries, split along the $\sqrt{n}$-th 2-boundaries.

Partition strategy

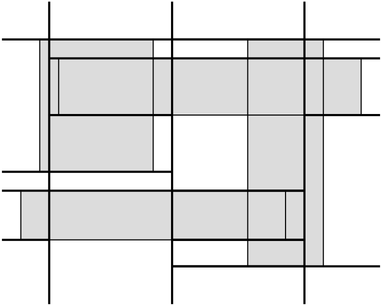
3-dim. case; 2-dim. projection



1. Split x_1-axis into $2\sqrt{n}$ intervals such that each contains at most $\sqrt{n}$ 1-boundaries.
2. For each 1-boundaries contained in a slab, split the slab along its 2-boundaries
3. For all other 2-boundaries, split along the $\sqrt{n}$-th 2-boundaries.

Partition strategy

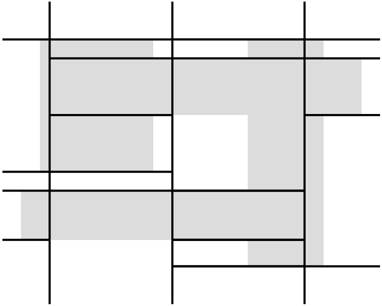
3-dim. case; 2-dim. projection



1. Split x_1-axis into $2\sqrt{n}$ intervals such that each contains at most $\sqrt{n}$ 1-boundaries.
2. For each 1-boundaries contained in a slab, split the slab along its 2-boundaries
3. For all other 2-boundaries, split along the $\sqrt{n}$-th 2-boundaries.

Partition strategy

3-dim. case; 2-dim. projection



1. Split x_1-axis into $2\sqrt{n}$ intervals such that each contains at most $\sqrt{n}$ 1-boundaries.
2. For each 1-boundaries contained in a slab, split the slab along its 2-boundaries
3. For all other 2-boundaries, split along the $\sqrt{n}$-th 2-boundaries.

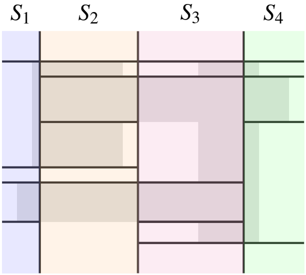
Characteristics

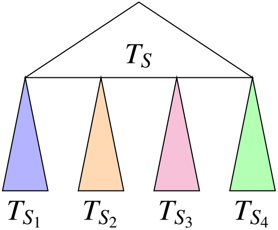
For 2-dimension case,

1. The space is divided into $O(n)$ cells.
   : $2\sqrt{n}$ slabs split into $4\sqrt{n}$ cells.
2. Each box of B partially covers at most $O(\sqrt{n})$ cells.
   : Each vertical line(1-boundary) cut through $4\sqrt{n}$ cells and horizontal line cut through $2\sqrt{n}$ slabs.
3. No cell contains vertices in its interior.
   : The vertices are in some horizontal boundaries.
   : The boxes are in trellis pattern
4. Each cell has at most $O(\sqrt{n})$ boxes partially covering it.
   : A cell contains at most $\sqrt{n}$ 1-boundaries and a slab contains at most $\sqrt{n}$ 2-boundaries.

Data structure

Orthogonal partition tree





Characteristics

Orthogonal partition tree

For 2-dimension case,

1. The tree has $O(n)$ leaves.
2. Each box stored in at most $O(\sqrt{n})$ leaves.
3. No C_λ's contain vertices in its interior.
4. Each leaf stores at most $O(\sqrt{n})$ boxes.
5. Each box influences at most $O(\sqrt{n}\log n)$ T_α's.
   : from 1 and 2.

Analysis

3-dim. case

Box insertion

- A box is stored in at most $O(\sqrt{n})$ leaves ($O(\sqrt{n}\log n)$)
- M_λ is computed from two segment trees for each axis. ($O(\sqrt{n}\log n)$)
- M_α and T_α is updated for all nodes α between the leaves λ and the root ($O(\sqrt{n}\log n)$ updates)

Box deletion Similar to inserting analysis

Measure query M_{root} is the answer. $O(1)$.

Theorem

The algorithm runs in $O(n\sqrt{n}\log n)$-time.

Extend to higher dimension

Partition strategy

1. Split x_1-axis into $2\sqrt{n}$ intervals such that each contains at most $\sqrt{n}$ 1-boundaries.
2. For each boxes whose 1-boundaries contained in a 1-slab, split the slab along its 2-boundaries
3. For all others, split the 1-slab along the $\sqrt{n}$-th 2-boundaries.
4. For each boxes whose 1- and 2- boundaries in a 2-slab, split the slab along the 3-boundaries.
5. For all others, split the 2-slab along the $\sqrt{n}$-th 3-boundaries.
6. (repeat)

Extend to higher dimension

Partition strategy

1. Split x_1-axis into $2\sqrt{n}$ intervals such that each contains at most $\sqrt{n}$ 1-boundaries.
2. For each boxes whose 1-boundaries contained in a 1-slab, split the slab along its 2-boundaries
3. For all others, split the 1-slab along the $\sqrt{n}$-th 2-boundaries.
4. For each boxes whose 1- and 2- boundaries in a 2-slab, split the slab along the 3-boundaries.
5. For all others, split the 2-slab along the $\sqrt{n}$-th 3-boundaries.
6. (repeat)

Extend to higher dimension

Partition strategy

1. Split x_1-axis into $2\sqrt{n}$ intervals such that each contains at most $\sqrt{n}$ 1-boundaries.
2. For each boxes whose 1-boundaries contained in a 1-slab, split the slab along its 2-boundaries
3. For all others, split the 1-slab along the $\sqrt{n}$-th 2-boundaries.
4. For each boxes whose 1- and 2- boundaries in a 2-slab, split the slab along the 3-boundaries.
5. For all others, split the 2-slab along the $\sqrt{n}$-th 3-boundaries.
6. (repeat)

Summary

Overmars and Yap, 1991

For d-dim. Klee's measure problem, this algorithm

1. runs in $O(n^{d/2} \log n)$-time with the partition tree.
2. uses $O(n^{d/2})$ space
   : This can be reduced to $O(n)$ (only segment trees) by interleaving the measuring step with the partitioning.

Approach 2

Chen, 2013

Key ideas

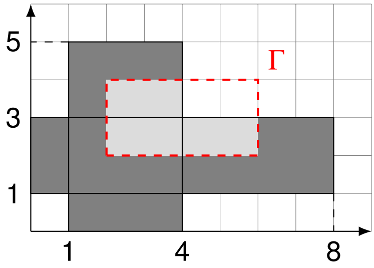
The logarithmic factor ($O(n^{d/2} \log n)$) comes from maintaining the tree while sweeping.
Can we design a different partitioning method that is free from maintaining a tree?

Modified problem

Problem (Modified version of the problem)

For a set of d-boxes B and an open box Γ, find the complement volume of union of B within the domain Γ.

Example



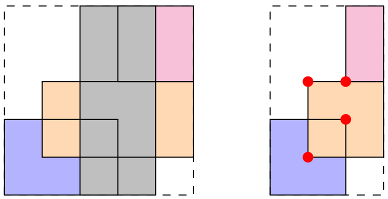
$A^C(\Gamma) = 2$

Algorithm

```
function MEASURE(B, Γ)
Given: C is a fixed, small constant
    if |B| < C then return the answer directly.
    Simplify B
    Cut Γ into two disjoint boxes Γ_L and Γ_R
    return MEASURE({b ∩ Γ_L | b ∈ B}, Γ_L)
        + MEASURE({b ∩ Γ_R | b ∈ B}, Γ_R)
end function
```

Simplification

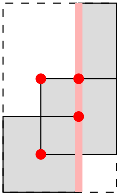
- Remove all slabs (a box of $\{x \mid a \leq x_i \leq b\}$ form in Γ) and adjust B and Γ.
- This costs linear time per axis.
- Note that the complement volume is preserved and all remaining boxes have a $(d-2)$-face intersecting with Γ.



Partitioning

2-dim. case

- Split Γ into two open boxes at the median of x_1-coord of all $(d-2)$-faces
- Swap axis number



Partitioning

3+-dim. case

- Assign a weight $2^{(i+j)/d}$ on all $(d-2)$-faces perpendicular to x_i and x_j-axes.
- The weight is bounded in $[1, 4]$.

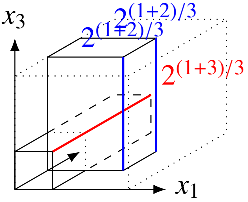


Figure: 1-faces of 3-d boxes intersecting the open domain

Partitioning

3+-dim. case

- Find a weighted median m among the intersection of the $(d-2)$ faces and the x_1-axis, and cut Γ through the hyperplane $x_1 = m$.

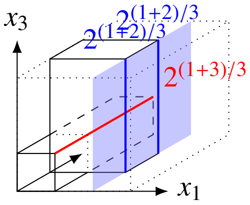


Figure: 1-faces of 3-d boxes intersecting the open domain

Partitioning

3+-dim. case

- Shift axis indices $(1 \to d \to (d-1) \to \cdots \to 3 \to 2 \to 1)$.
- Effectively a k-d tree.

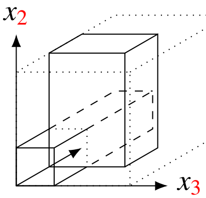


Figure: 1-faces of 3-d boxes intersecting the open domain

Partitioning

Weight decrease

- After cutting, $(d-2)$-faces not perpendicular to x_1-axis $(i, j \neq 1)$ will have weight $2^{(i-1+j-1)/d}$, decreased by $2^{2/d}$.
- $(d-2)$-faces perpendicular to x_1 axis $(j \neq 1)$ will have weight $2^{(d+j-1)/d}$, increased by $2^{(d-2)/d}$. Note that these faces are split into smaller domains by half, reducing the total weight by half.

Analysis

Simplifying
- $O(n)$ for each axis to identify a slab
- $O(n)$ for each axis to adjust box boundaries

Cut
- Finding (weighted) median is $O(n)$ after sorting
- The total weight is decreased by $2^{2/d}$ for each cutting step.

Theorem

The algorithm runs in $O(n^{d/2})$-time.

Algorithm Summary

Dim.	Time complexity
1	$\Theta(n \log n)$[3]
2	$\Theta(n \log n)$
3+	$\Omega(n \log n)$–$O(n^{d/2})$

[3] $O(n \log p)$ where p is min. #lines stabbing all intervals.