Klee's Measure Problem Made Easy

Timothy Chan (UIUC)

[appeared in FOCS'13]



Beginning of Story…

- Victor Klee, "Can the measure of $U_1^n [a_i,b_i]$ be computed in less than O(n log n) steps?", Amer. Math. Monthly, 84:284-285, 1977

RESEARCH PROBLEMS

EDITED BY RICHARD GUY

In this Department the Monthly presents easily stated research problems dealing with notions ordinarily encountered in undergraduate mathematics. Each problem should be accompanied by relevant references (if any are known to the author) and by a brief description of known partial results. Manuscripts should be sent to Richard Guy, Department of Mathematics and Statistics, The University of Calgary, Calgary, Alberta, Canada, T2N 1N4. (From July 1976 to June 1977: Department of Pure Mathematics and Mathematical Statistics, University of Cambridge, 16 Mill Lane, Cambridge CB2 1SB, England.)

CAN THE MEASURE OF $\bigcup_1^n [a_i, b_i]$ BE COMPUTED IN LESS THAN $O(n \log n)$ STEPS?

VICTOR KLEE

Problem statement. Given n intervals $[a_1, b_1], \ldots, [a_n, b_n]$ in the real line, it is desired to find the measure of their union. How efficiently can that be done? The list of endpoints can be sorted with $O(n \log n)$ comparisons, and then $O(n)$ additions suffice to determine the measure of $\bigcup_1^n [a_i, b_i]$. However, the problem does not, *a priori*, require sorting, and it would be of interest to know whether there is a solution involving less than $O(n \log n)$ computational steps.

Reason for interest. In addition to its intrinsic appeal, the problem is of interest because it arises naturally in approximating the areas of certain subsets of the cartesian plane. Suppose that $[u_1, v_1], \ldots, [u_n, v_n]$ are intervals in the real line, f_i and g_i are continuous real functions on $[u_i, v_i]$, and

$$P_i = \{(x, y): u_i \leqq x \leqq v_i, f_i(x) \leqq y \leqq g_i(x)\}.$$

Even when the areas of the individual P_i's are easily computed, it may be difficult to compute the area of the union $P = \bigcup_1^n P_i$ if the P_i's have a complicated intersection-pattern. However, when the functions are Lipschitzian the area of P is closely approximated by sums of the form

$$(*) \qquad \sum_{k=1}^{m} (\text{measure of } L_k \cap P)(s_k - s_{k-1}),$$

where $s_0 < s_1 < \ldots < s_m$ is an appropriate (sufficiently dense) sequence in $\bigcup_1^n [u_i, v_i]$ and L_k is the line of abscissa $(s_{k-1} + s_k)/2$. Since

$$L_k \cap P = \bigcup_{i=1}^{n} L_k \cap P_i,$$

computing (*) involves mn problems of the sort mentioned above.

Solution by sorting. Suppose that each left endpoint a_i is tagged with 1 and each right endpoint b_i is tagged with -1. By means of various procedures requiring $O(n \log n)$ comparisons, the sequence of all $2n$ endpoints can be arranged in increasing order $e_1 \leqq e_2 \leqq \ldots \leqq e_{2n}$ such that if $t_1, \ldots, t_{2n}$ is the corresponding permutation of the tags, then $t_i \geqq t_{i+1}$ whenever $e_i = e_{i+1}$; that is, the sorting does not place a right endpoint before a left endpoint of the same numerical value. With EXCESS denoting the excess of the number of left endpoints over the number of right endpoints, the following program constructs the sequence $[c_1, d_1], \ldots, [c_m, d_m]$ of components of $\bigcup_1^n [a_i, b_i]$ and then computes the measure of $\bigcup_1^n [a_i, b_i]$.

```
begin
  MEASURE ← 0; m ← EXCESS ← 0;
  for i ← 1 until 2n do
  begin
    EXCESS ← EXCESS + t_i;
    if EXCESS = 1 then begin m ← m + 1; c_m ← e_i end;
    if EXCESS = 0 then d_m ← e_i
  end;
  for i ← 1 until m do MEASURE ← MEASURE + (d_i − c_i)
end
```

Problems in d-space. For d-dimensional intervals $I_1, \ldots, I_n$ in cartesian d-space, each given as the product of d real intervals, we state the following three problems and in each case ask how efficiently the problem can be solved.

PARTITION PROBLEM: *Find a sequence $H_1, \ldots, H_m$ of intervals such that $\bigcup_1^m H_k = \bigcup_1^n I_k$ and no two of the H_k's have a common interior point.*

MINIMUM PARTITION PROBLEM: *Find $H_1, \ldots, H_m$ as described such that m is minimum.*

MEASURE PROBLEM: *Find the d-measure of $\bigcup_1^n I_k$.*

When $d = 1$ the above algorithm solves all three problems in $O(n \log n)$ steps, but it is not obvious that the three problems are actually of the same computational complexity. It would be of special interest to know whether the measure problem can be solved in a number of steps that is bounded by a polynomial in d and n.

The books [1] and [2] are recommended as general references on sorting and computational complexity, while the methods of [3] can probably be applied to the partition problems.

References

1. A. V. Aho, J. E. Hopcroft and J. D. Ullman, The Design and Analysis of Computer Algorithms, Addison-Wesley, Reading, Mass., 1974.
2. D. E. Knuth, The Art of Computer Programming, Vol. 3: Sorting and Searching, Addison-Wesley, Reading, Mass., 1973.
3. H. T. Kung, F. Luccio and F. P. Preparata, On finding the maxima of a set of vectors, J. Assoc. Comp. Mach., 22 (1975) 469–476.

DEPARTMENT OF MATHEMATICS, UNIVERSITY OF WASHINGTON, SEATTLE, WA 98195.

Klee's Original Measure Problem

- Given n (possibly overlapping) intervals on the real line, compute the measure of the union



- O(n log n) time is easy by sorting

RESEARCH PROBLEMS

EDITED BY RICHARD GUY

In this Department the Monthly presents easily stated research problems dealing with notions ordinarily encountered in undergraduate mathematics. Each problem should be accompanied by relevant references (if any are known to the author) and by a brief description of known partial results. Manuscripts should be sent to Richard Guy, Department of Mathematics and Statistics, The University of Calgary, Calgary, Alberta, Canada, T2N 1N4. (From July 1976 to June 1977: Department of Pure Mathematics and Mathematical Statistics, University of Cambridge, 16 Mill Lane, Cambridge CB2 1SB, England.)

CAN THE MEASURE OF $\bigcup_1^n [a_i, b_i]$ BE COMPUTED IN LESS THAN $O(n \log n)$ STEPS?

VICTOR KLEE

Problem statement. Given n intervals $[a_1, b_1], \ldots, [a_n, b_n]$ in the real line, it is desired to find the measure of their union. How efficiently can that be done? The list of endpoints can be sorted with $O(n \log n)$ comparisons, and then $O(n)$ additions suffice to determine the measure of $\bigcup_1^n [a_i, b_i]$. However, the problem does not, *a priori*, require sorting, and it would be of interest to know whether there is a solution involving less than $O(n \log n)$ computational steps.

Reason for interest. In addition to its intrinsic appeal, the problem is of interest because it arises naturally in approximating the areas of certain subsets of the cartesian plane. Suppose that $[u_1, v_1], \ldots, [u_n, v_n]$ are intervals in the real line, f_i and g_i are continuous real functions on $[u_i, v_i]$, and

$$P_i = \{(x, y): u_i \leqq x \leqq v_i, f_i(x) \leqq y \leqq g_i(x)\}.$$

Even when the areas of the individual P_i's are easily computed, it may be difficult to compute the area of the union $P = \bigcup_1^n P_i$ if the P_i's have a complicated intersection-pattern. However, when the functions are Lipschitzian the area of P is closely approximated by sums of the form

$$(*) \qquad \sum_{k=1}^{m} (\text{measure of } L_k \cap P)(s_k - s_{k-1}),$$

where $s_0 < s_1 < \ldots < s_m$ is an appropriate (sufficiently dense) sequence in $\bigcup_1^n [u_i, v_i]$ and L_k is the line of abscissa $(s_{k-1} + s_k)/2$. Since

$$L_k \cap P = \bigcup_{i=1}^{n} L_k \cap P_i,$$

computing (*) involves m problems of the sort mentioned above.

Solution by sorting. Suppose that each left endpoint a_i is tagged with 1 and each right endpoint b_i is tagged with -1. By means of various procedures requiring $O(n \log n)$ comparisons, the sequence of all $2n$ endpoints can be arranged in increasing order $e_1 \leqq e_2 \leqq \ldots \leqq e_{2n}$ such that if $t_1, \ldots, t_{2n}$ is the corresponding permutation of the tags, then $t_i \geqq t_{i+1}$ whenever $e_i = e_{i+1}$; that is, the sorting does not place a right endpoint before a left endpoint of the same numerical value. With EXCESS denoting the excess of the number of left endpoints over the number of right endpoints, the following program constructs the sequence $[c_1, d_1], \ldots, [c_m, d_m]$ of components of $\bigcup_1^n [a_i, b_i]$ and then computes the measure of $\bigcup_1^n [a_i, b_i]$.

```
begin
  MEASURE ← 0; m ← EXCESS ← 0;
  for i ← 1 until 2n do
  begin
    EXCESS ← EXCESS + t_i;
    if EXCESS = 1 then begin m ← m + 1; c_m ← e_i end;
    if EXCESS = 0 then d_m ← e_i
  end;
  for i ← 1 until m do MEASURE ← MEASURE + (d_i − c_i)
end
```


Problems in d-space. For d-dimensional intervals $I_1, \ldots, I_n$ in cartesian d-space, each given as the product of d real intervals, we state the following three problems and in each case ask how efficiently the problem can be solved.

PARTITION PROBLEM: *Find a sequence $H_1, \ldots, H_m$ of intervals such that $\bigcup_1^n H_k = \bigcup_1^n I_k$ and no two of the H_k's have a common interior point.*

MINIMUM PARTITION PROBLEM: *Find $H_1, \ldots, H_m$ as described such that m is minimum.*

MEASURE PROBLEM: *Find the d-measure of $\bigcup_1^n I_k$.*

When $d = 1$ the above algorithm solves all three problems in $O(n \log n)$ steps, but it is not obvious that the three problems are actually of the same computational complexity. It would be of special interest to know whether the measure problem can be solved in a number of steps that is bounded by a polynomial in d and n.

The books [1] and [2] are recommended as general references on sorting and computational complexity, while the methods of [3] can probably be applied to the partition problems.

References

1. A. V. Aho, J. E. Hopcroft and J. D. Ullman, The Design and Analysis of Computer Algorithms, Addison-Wesley, Reading, Mass., 1974.

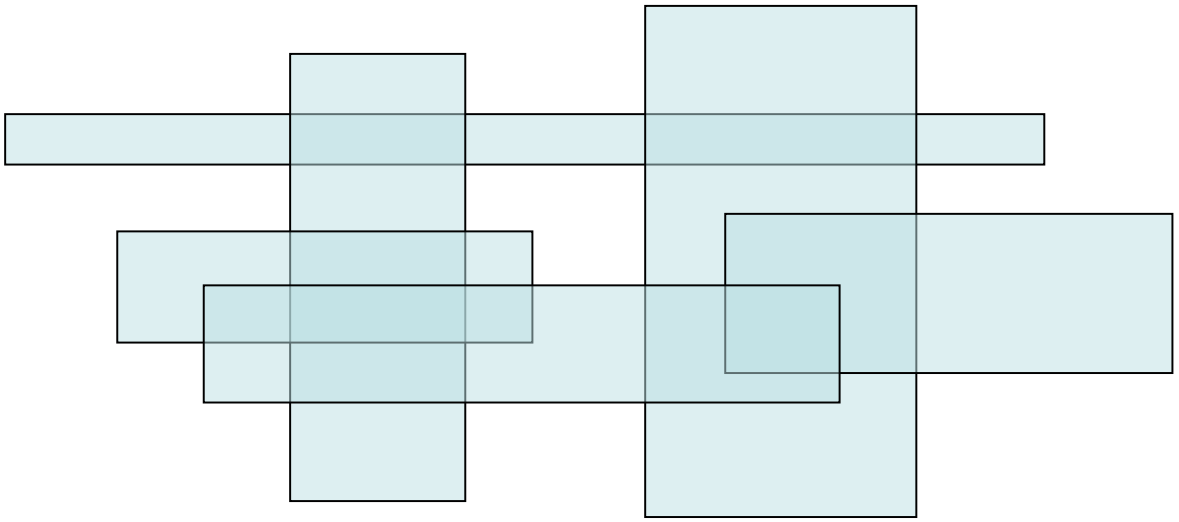
2. D. E. Knuth, The Art of Computer Programming, Vol. 3: Sorting and Searching, Addison-Wesley, Reading, Mass., 1973.

3. H. T. Kung, F. Luccio and F. P. Preparata, On finding the maxima of a set of vectors, J. Assoc. Comp. Mach., 22 (1975) 469–476.

DEPARTMENT OF MATHEMATICS, UNIVERSITY OF WASHINGTON, SEATTLE, WA 98195.

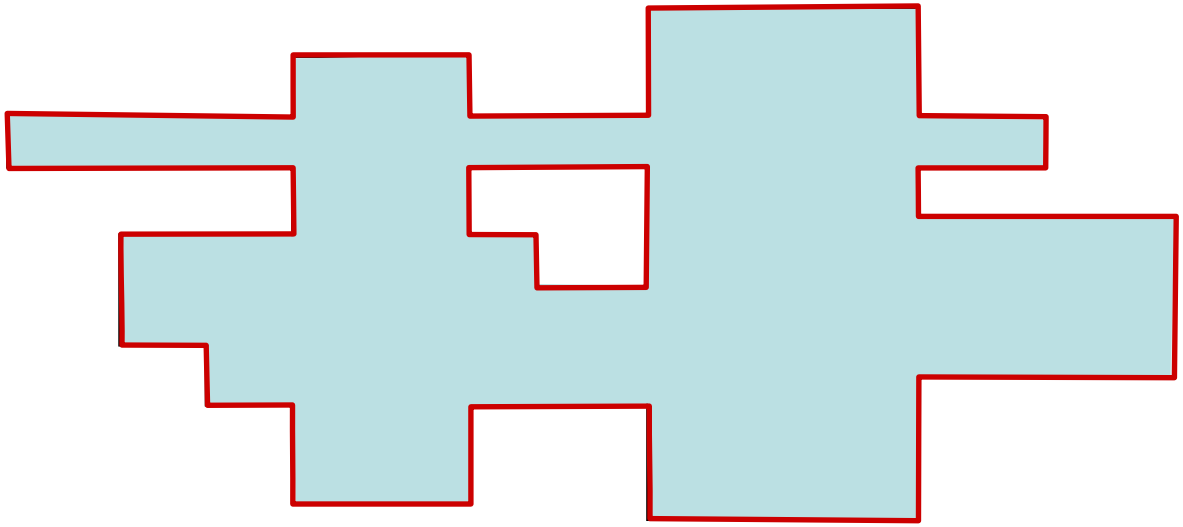
What About d Dimensions?

- Given n (possibly overlapping but axis-parallel) rectangular boxes in d dimensions, compute the measure of the union



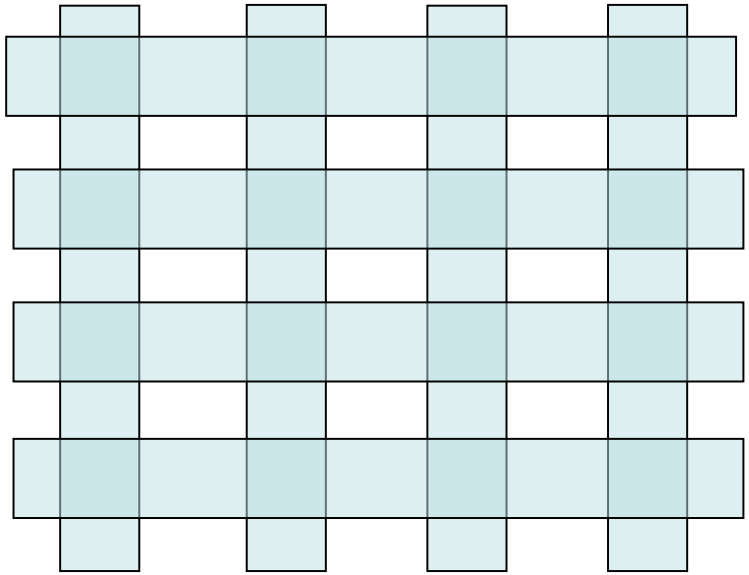
What About d Dimensions?

- Given n (possibly overlapping but axis-parallel) rectangular boxes in d dimensions, compute the measure of the union



Known Result in 2D

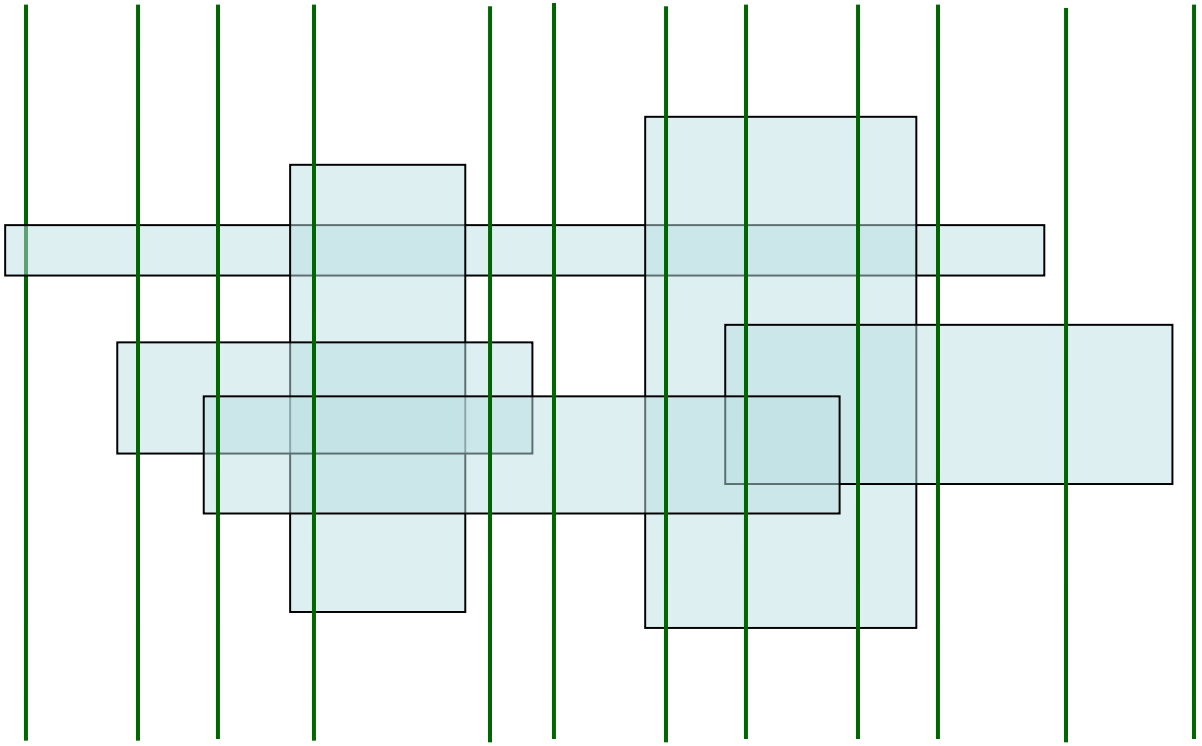
- Union of n rectangles may have $\Omega(n^2)$ boundary edges

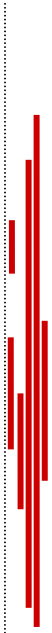


- Bentley'77: $O(n \log n)$ time (same as 1D!)

Bentley's Algorithm in 2D (Sketch)

- Sweep line approach





$\Rightarrow$ dynamic problem in 1D

Bentley's Algorithm in 2D (Sketch)

- Sweep line approach

$\Rightarrow$ dynamic problem in 1D

- Solve dynamic 1D problem by balanced search tree
  - $O(\log n)$ time per insert/delete

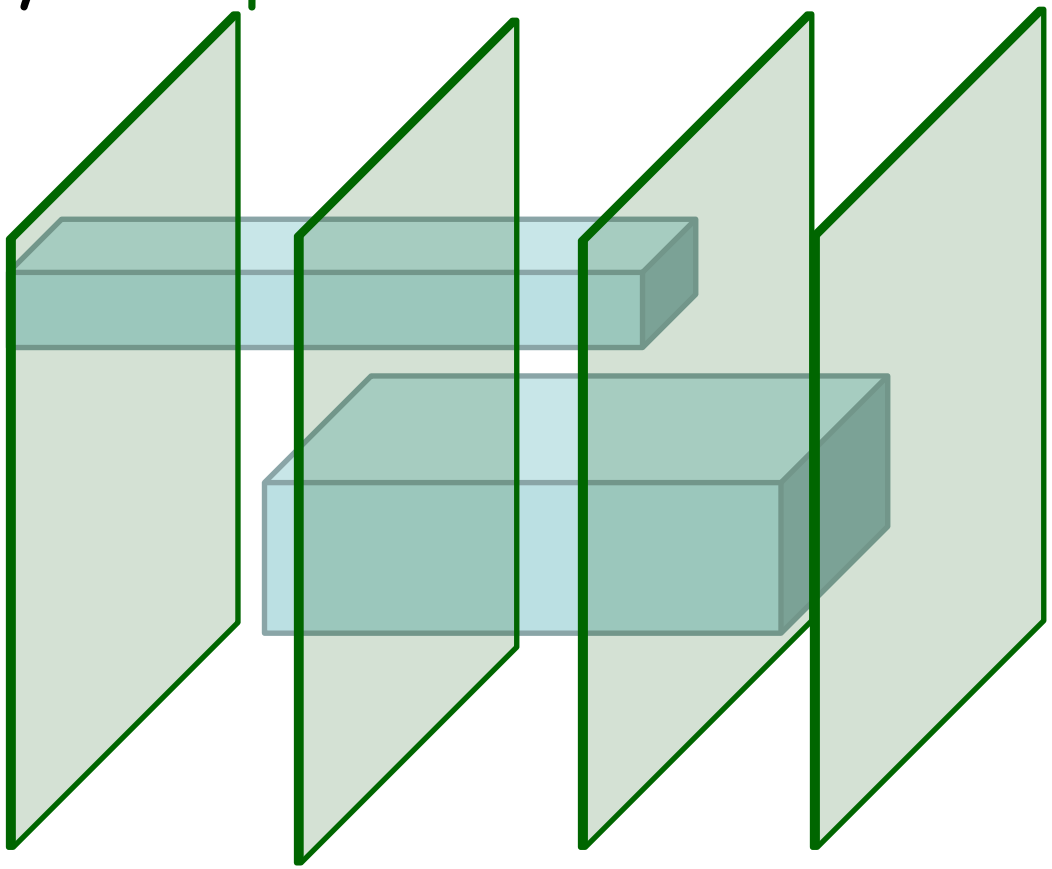
$\Rightarrow$ $O(n \log n)$ time

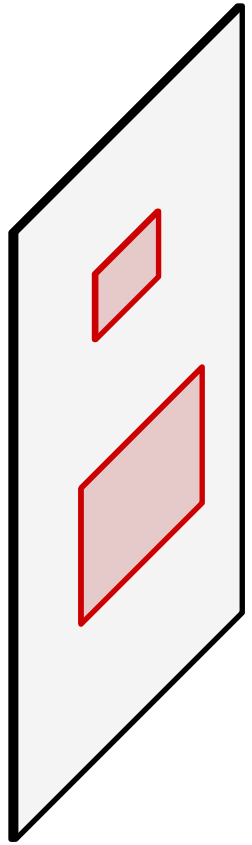
Known Results in d ≥ 3 Dimensions

- Union of n boxes may have $\Omega(n^d)$ boundary faces
- Bentley'77: $O(n^{d-1} \log n)$ time

Bentley's Algorithm in 3D

- Again by sweep





$\Rightarrow$ dynamic problem in 2D

Known Results in d ≥ 3 Dimensions

- Union of n boxes may have $\Omega(n^d)$ boundary faces
- Bentley'77: $O(n^{d-1} \log n)$ time
- Overmars,Yap'88: $O(n^{d/2} \log n)$ time

Overmars,Yap's Algorithm in 3D (Sketch)

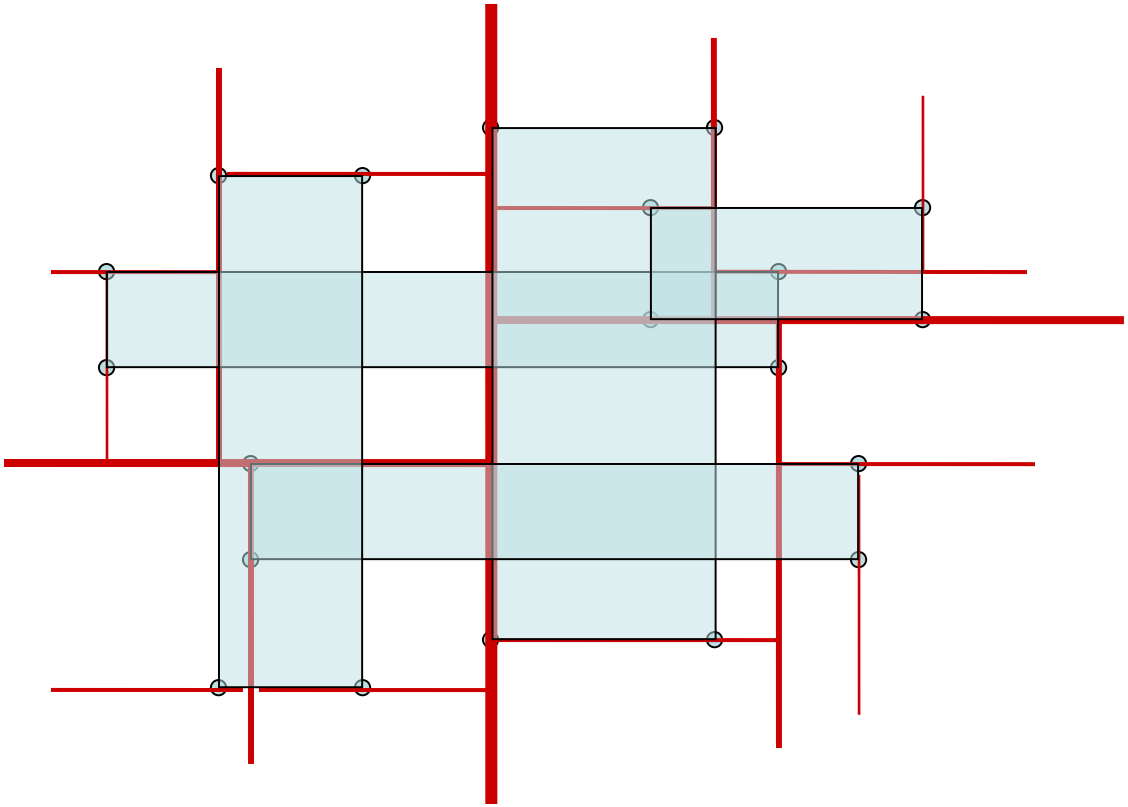
- Again by sweep

$\Rightarrow$ dynamic problem in 2D

- Solve dynamic 2D problem by…

Overmars,Yap's Algorithm in 3D (Sketch)

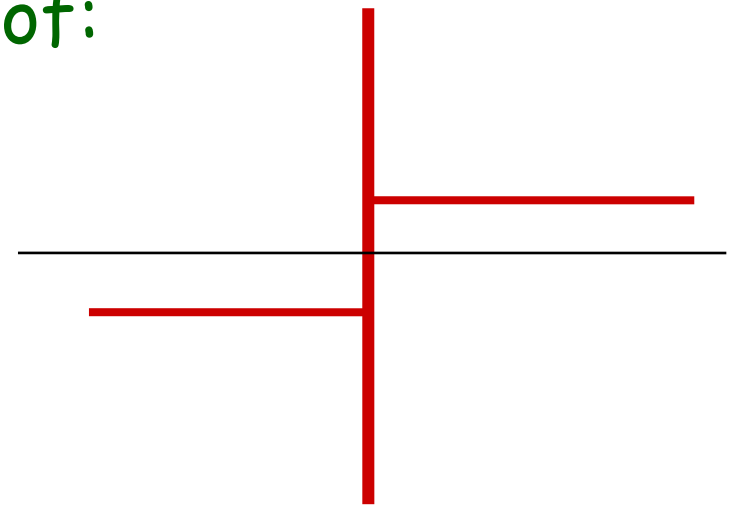
- Solve dynamic 2D problem by… k-d tree



Overmars,Yap's Algorithm in 3D (Sketch)

- Property 1: a horizontal/vertical line can hit $O(n^{1/2})$ cells of the k-d tree

- Quick Proof:



$T(n) = 2\,T(n/4) + O(1)$ Q.E.D.

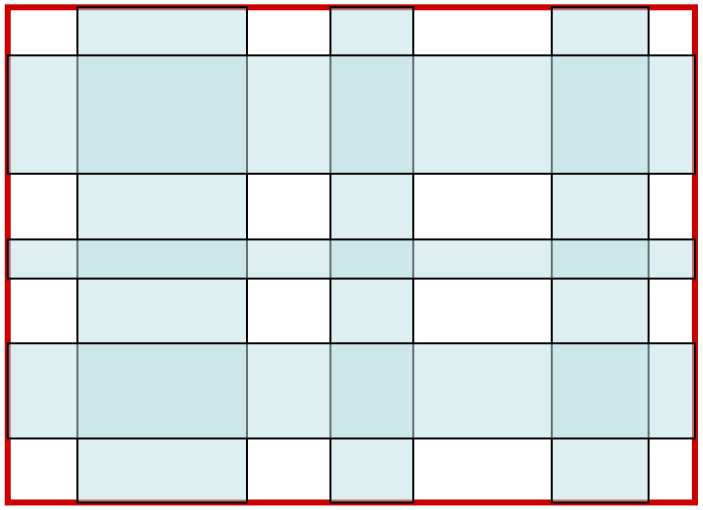
Overmars,Yap's Algorithm in 3D (Sketch)

- Property 1: a horizontal/vertical line can hit $O(n^{1/2})$ cells of the k-d tree

$\Rightarrow$ in each insert/delete, rectangle's boundary hits $O(n^{1/2})$ cells

Overmars,Yap's Algorithm in 3D (Sketch)

- Property 2: inside each leaf cell, no vertices
  $\Rightarrow$ union must look like this…



product of
two 1D problems

$\Rightarrow$ $O(\log n)$ time
per insert/delete

$\Rightarrow$ $O(n^{1/2} \log n)$ total time per insert/delete

$\Rightarrow$ $O(n^{3/2} \log n)$ time

Known Results in d ≥ 3 Dimensions

- Bentley'77: $O(n^{d-1} \log n)$ time

- Overmars,Yap'88: $O(n^{d/2} \log n)$ time
- Chan'08: $O(n^{d/2} \log\log n)$ time

Known Results in d ≥ 3 Dimensions

- Bentley'77: $O(n^{d-1} \log n)$ time
- Overmars,Yap'88: $O(n^{d/2} \log n)$ time
- Chan'08: $O(n^{d/2} \log\log\log\log\log\log n)$ time

Known Results in d ≥ 3 Dimensions

- Bentley'77: $O(n^{d-1} \log n)$ time
- Overmars,Yap'88: $O(n^{d/2} \log n)$ time
- Chan'08: $O(n^{d/2} 2^{O(\log^* n)})$ time
- Chan'13: $O(n^{d/2})$ time (today's result)

Warm-Up: New Algorithm in 2D

- Instead of sweep...
- just simple divide&conquer!

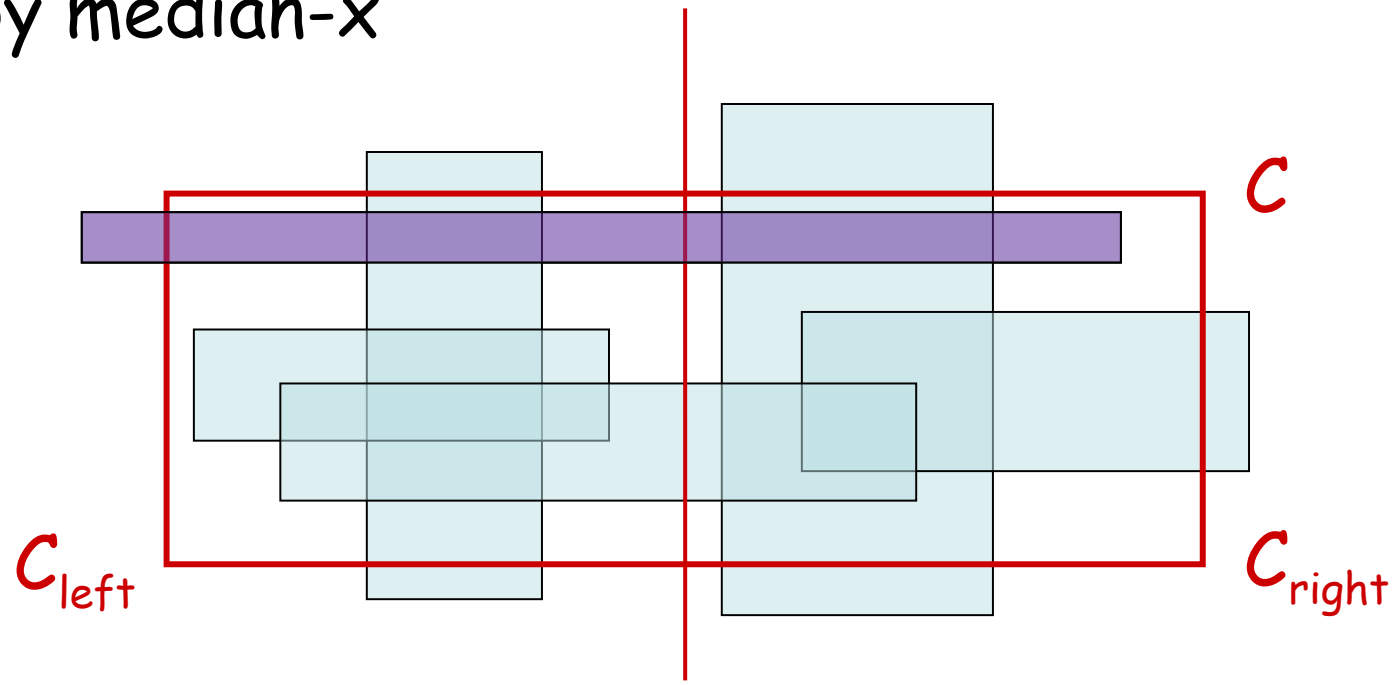
New Algorithm in 2D: First Attempt

- Given set B of n rectangles inside cell C:

1. divide C into C_{left} & C_{right}
2. recurse inside C_{left} & inside C_{right}

New Algorithm in 2D: How to Divide?

- … by median-x



- Problem: "long" rectangles!

  (# vertices is halved, not # rectangles;
  some rectangles may not have vertices inside cell)

New Algorithm in 2D

- Given set B of n rectangles inside cell C:

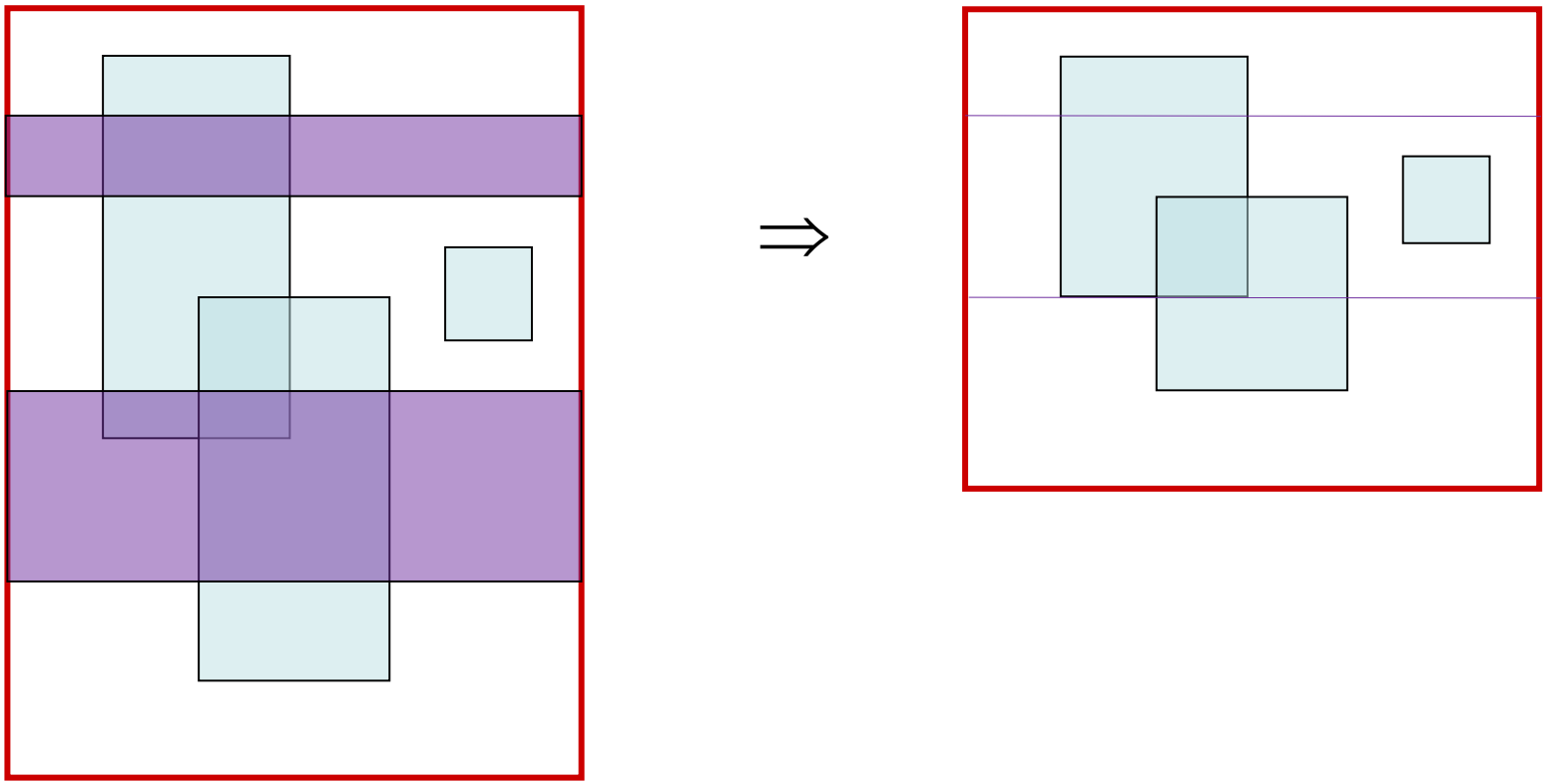
  0. simplify B ← new twist!

  1. divide C into C_{left} & C_{right}

  2. recurse inside C_{left} & inside C_{right}

New Algorithm in 2D: How to Simplify?

- … by removing all rectangles that don't have vertices inside C



New Algorithm in 2D: Summary & Analysis

- Given set B of n rectangles inside cell C:
  0. simplify B
     … by removing all rectangles that don't have vertices inside C
  1. divide C into C_{left} & C_{right}
     … by median-x
  2. recurse inside C_{left} & inside C_{right}

- Let N = # vertices inside C

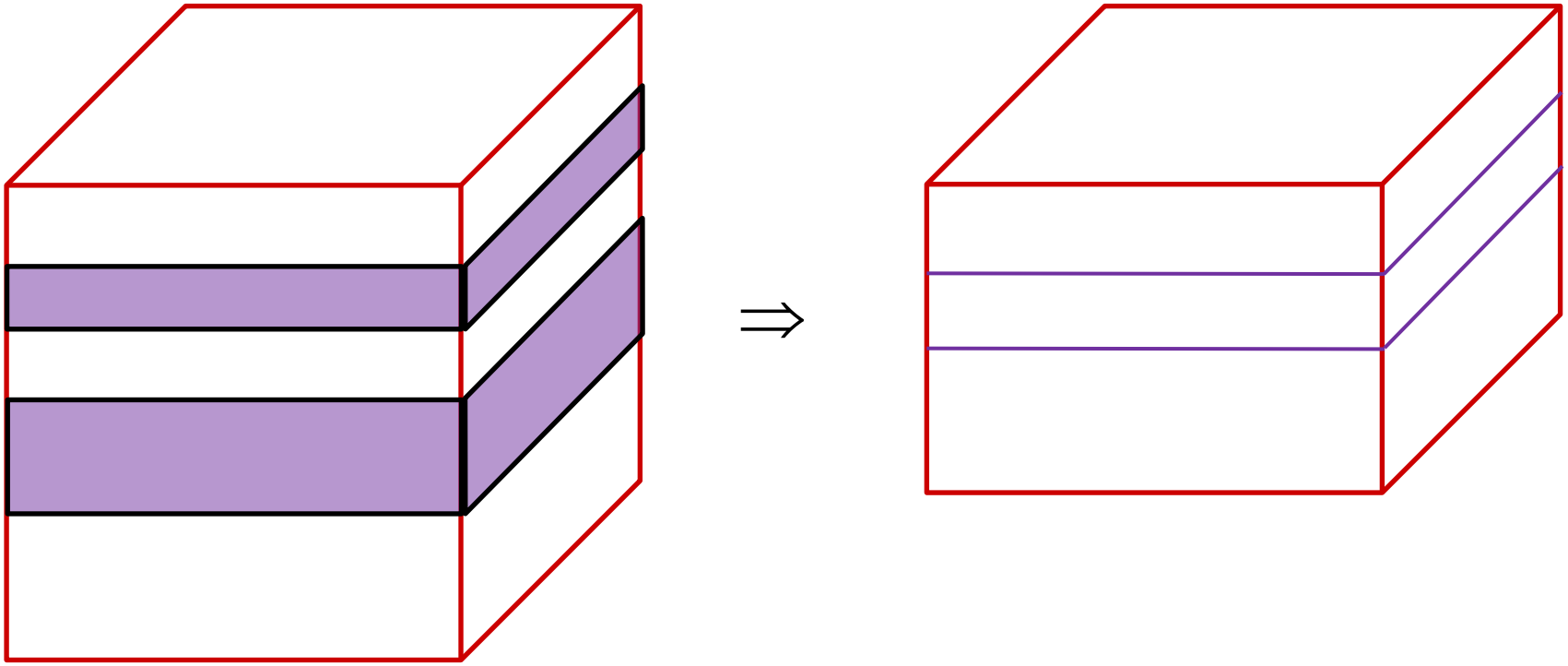
  $T(N) = 2\ T(N/2) + O(N) \Rightarrow \boxed{O(N \log N)}$

Extending New Algorithm to 3D

- Given set B of n boxes inside cell C:
  0. simplify B
     … by removing all boxes that don't have edges intersecting C
  1. divide C into C_{left} & C_{right}
     … by weighted median-x
  2. recurse inside C_{left} & inside C_{right}

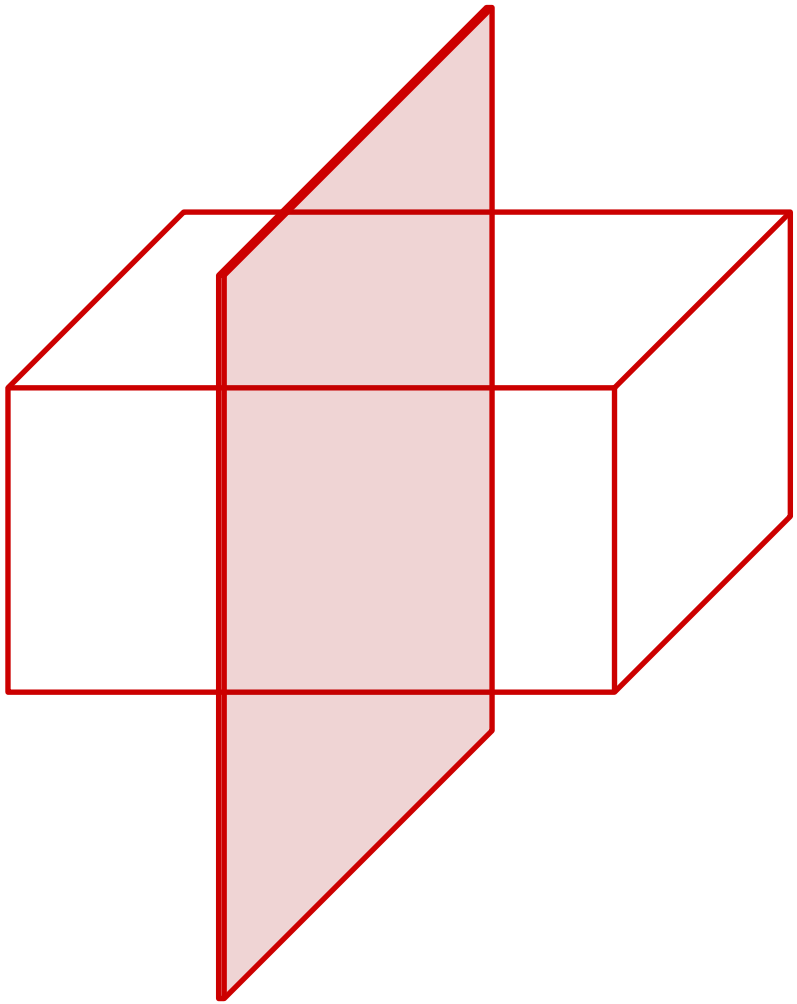
New Algorithm in 3D: How to Simplify?

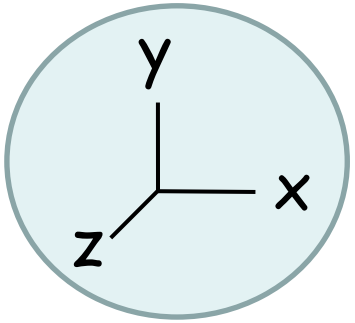
- … by removing all boxes that don't have edges intersecting C



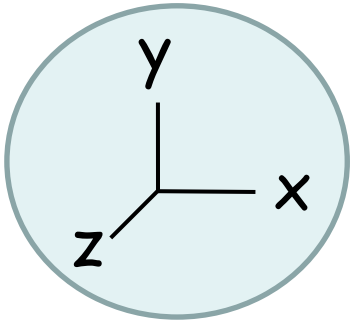
New Algorithm in 3D: How to Divide?

- … by weighted median-x

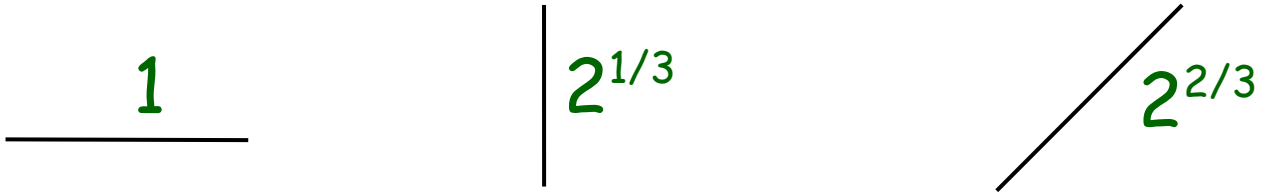




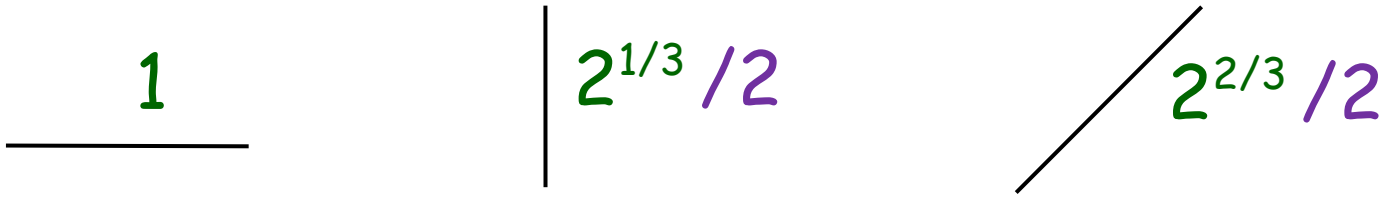
New Algorithm in 3D: How to Divide?



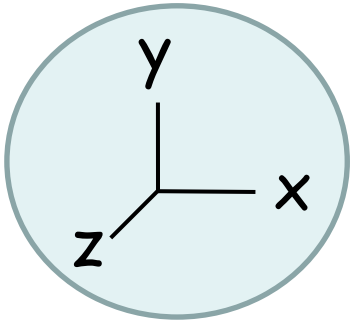
- … by weighted median-x
- Define weight of an edge:



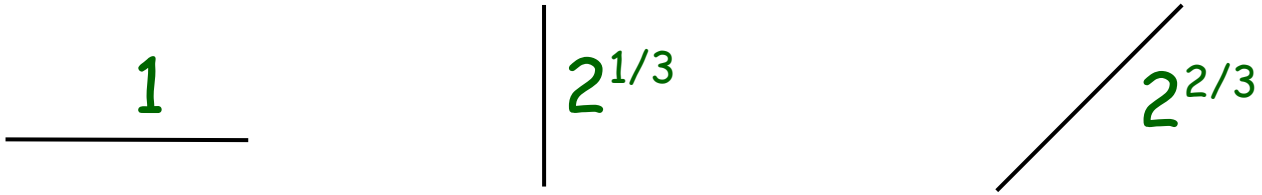
- Total weight of edges intersecting C_{left} (or C_{right}) =



New Algorithm in 3D: How to Divide?



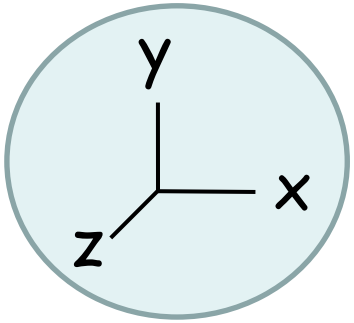
- … by weighted median-x
- Define weight of an edge:



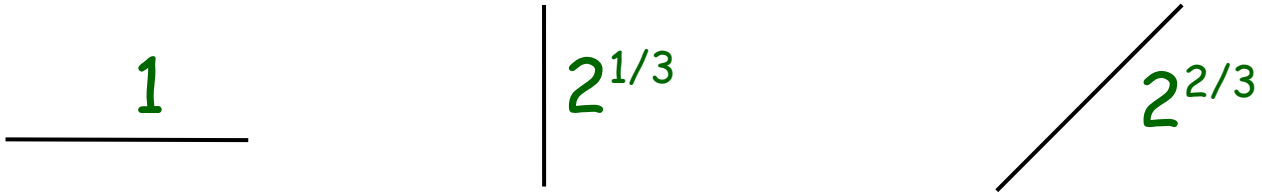
- Total weight of edges intersecting C_{left} (or C_{right}) =



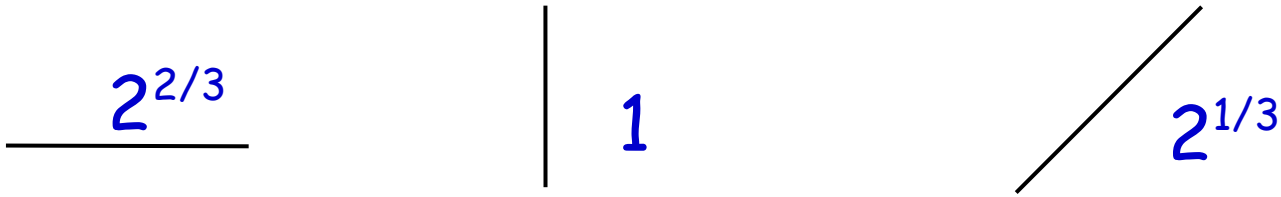
New Algorithm in 3D: How to Divide?



- … by weighted median-x
- Define weight of an edge:



- Total weight of edges intersecting C_{left} (or C_{right}) =



… after factoring out $2^{-2/3}$

New Algorithm in 3D: Analysis

- Let N = total weight of edges intersecting C

$\Rightarrow$ $N_{left}, N_{right} \leq 2^{-2/3} N$ after renaming axes

$$T(N) = 2\,T(N / 2^{2/3}) + O(N)$$

$\Rightarrow$ $\boxed{O(N^{3/2})}$

[Recall "Master Theorem": $T(N) = a\,T(N/b) + O(N^c)$
$\Rightarrow T(N) = O(N^{\log a/\log b})$ if $\log a/\log b > c$]

CLOSING REMARKS

Other Applications

- Cover Problem: given n boxes in d dimensions, does their union completely cover C?

- Depth Problem: given n boxes in d dimensions, find a point q minimizing/maximizing

  depth(q) = # of boxes containing q

(Conditional) Lower Bounds?

- Chan'08: The problem of finding cliques of size d in a graph with $V=n^{1/2}$ vertices reduces to the cover/measure problem

$\Rightarrow$ hard to beat $O(V^d) = O(n^{d/2})$ time (ignoring log factors) by purely "combinatorial" algorithms

- Open question: can matrix multiplication help? (Kunnemann [FOCS'22]: probably no for d=3...)

Special Cases: Unit Hypercubes

- Union may have $\Omega(n^{\lfloor d/2 \rfloor})$ boundary faces

- Bringmann'10: $O(n^{(d+2)/3})$ time
- Chan'13: $O(n^{d/3} \text{ polylog } n)$ time

Special Cases: Arbitrary Hypercubes

- Union may have $\Omega(n^{\lceil d/2 \rceil})$ boundary faces

- Agarwal,Kaplan,Sharir'07: $O(n^{4/3} \log^2 n)$ time (d=3)
- Agarwal'10: $O(n \log^4 n)$ time (d=3)

- Bringmann'10: $O(n^{(d+2)/3})$
- Chan'13: $O(n^{(d+1)/3} \text{ polylog } n)$

- Open question: lower bound ~d/3? THE END