

BBD trees

Hubert Bernacki

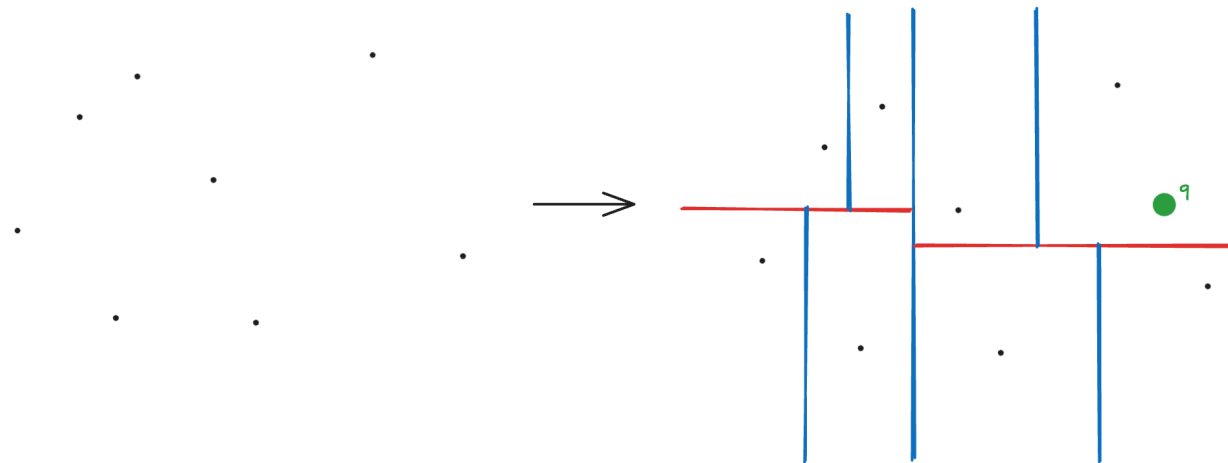
2026-05-28

Nearest neighbor search

Problem (Nearest Neighbor): Given a set S of n points in $\mathbb{R}^d$.

Preprocess them to efficiently answer queries:

Query: $q \in \mathbb{R}^d$, what is the closest point $p \in S$ to q ?

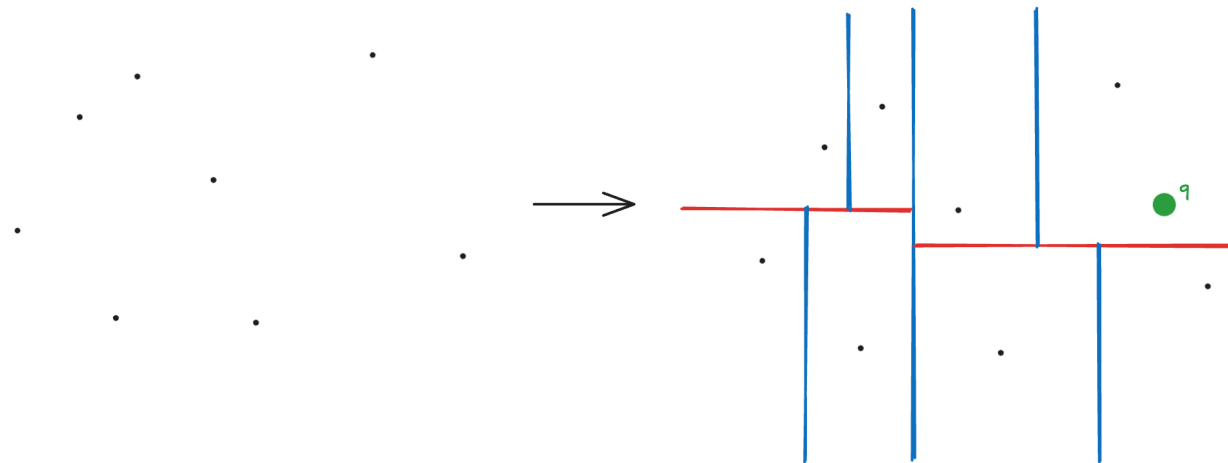


Nearest neighbor search

Problem (Nearest Neighbor): Given a set S of n points in $\mathbb{R}^d$.

Preprocess them to efficiently answer queries:

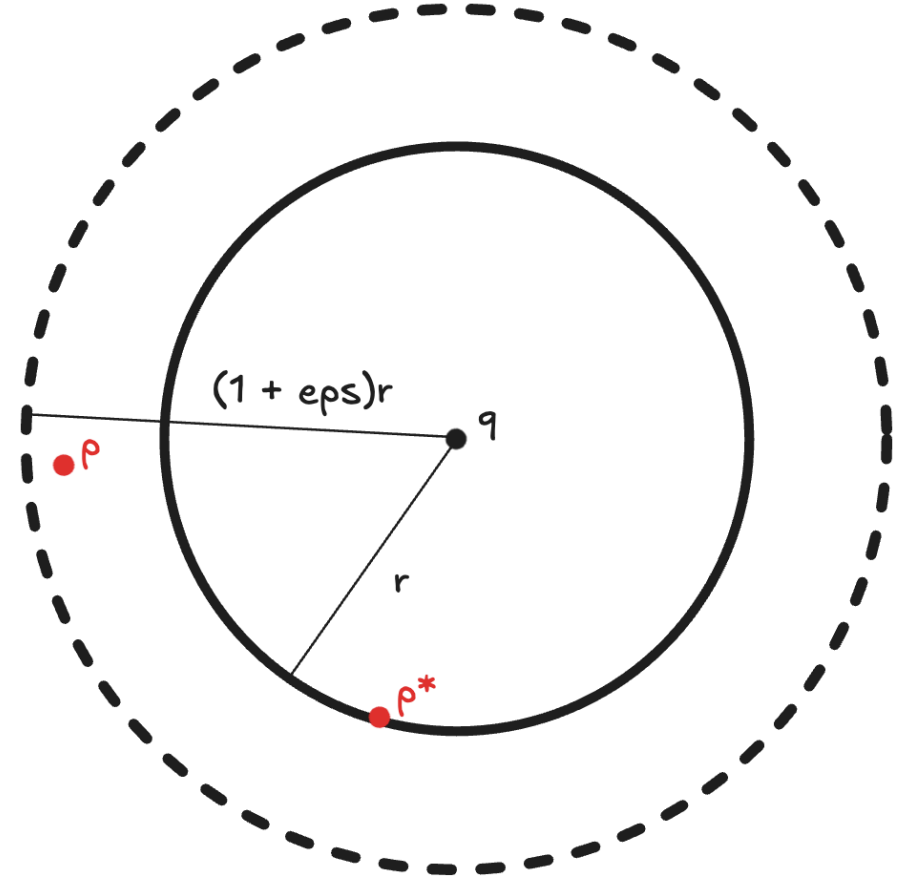
Query: $q \in \mathbb{R}^d$, what is the closest point $p \in S$ to q ?



Goal: A data structure which takes $O(dn)$ space, and $O(\log n)$ queries.

Approximate Nearest Neighbor

Problem (Approximate Nearest Neighbor): Given a set S of n points in $\mathbb{R}^d$. Preprocess them to efficiently answer queries:
Query: $q \in \mathbb{R}^d$, find a $p \in S$ such that

$$\text{dist}(p, q) \leq (1 + \varepsilon) \text{dist}(p^*, q)$$


NN, ANN structures comparison

Structure	Space	Query	Citation
Brute Force	$O(dn)$	$O(dn)$	—
kd-tree	$O(dn)$	$O(\log n)^* / O(n)$	Friedman (1977)
Voronoi	$O\left(n^{\lceil \frac{d}{2} \rceil}\right)$	$O(\log n)$	Clarkson (1988)
BBD Tree	$O(dn)$	$O(c_{d,\varepsilon} \log n)^{**}$	Arya et al. (1998)

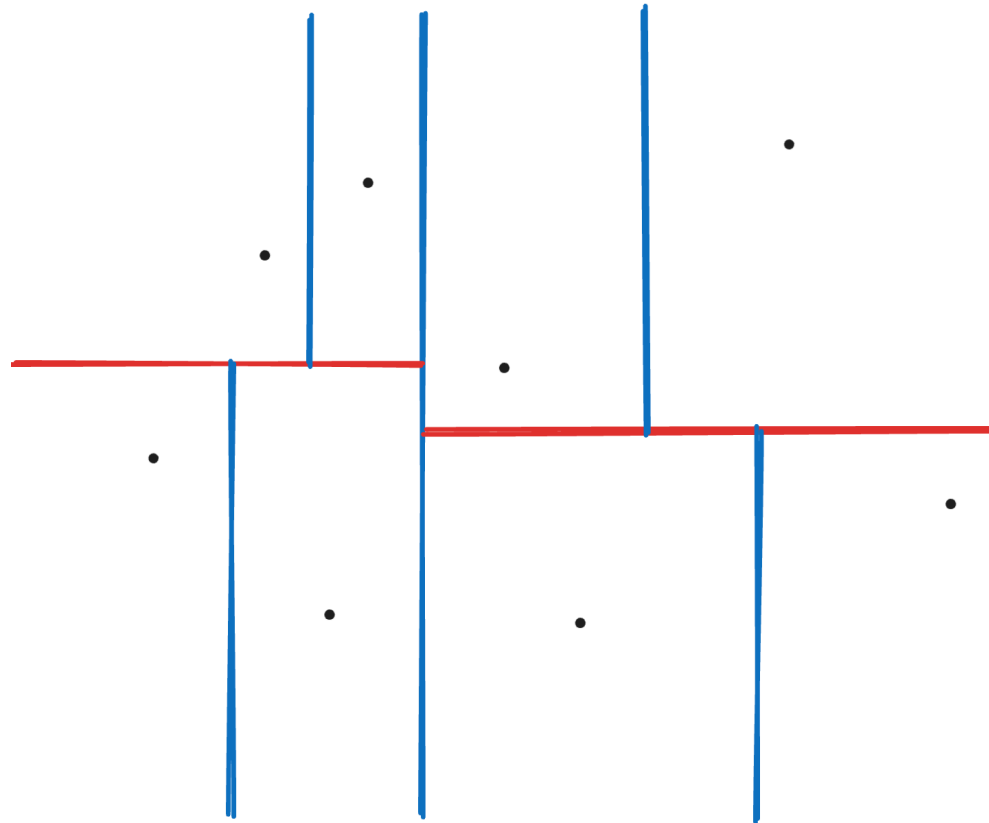
* $O(\log n)$ expected case / $O(n)$ worst case.

** In d fixed dimensions, $c_{d,\varepsilon} \leq d \lceil 1 + 6\frac{d}{\varepsilon} \rceil^d$.

NN, ANN structures comparison

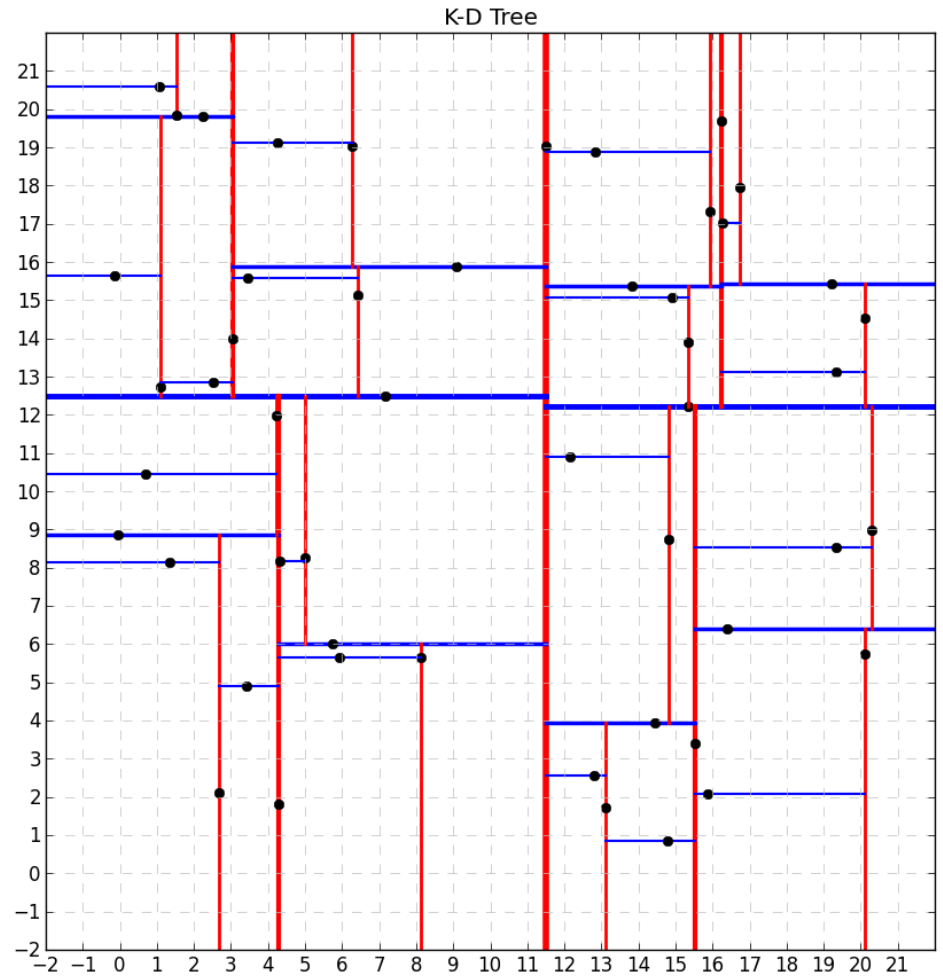
Kd-trees

A tree which partitions the space with hyperplanes.



NN, ANN structures comparison

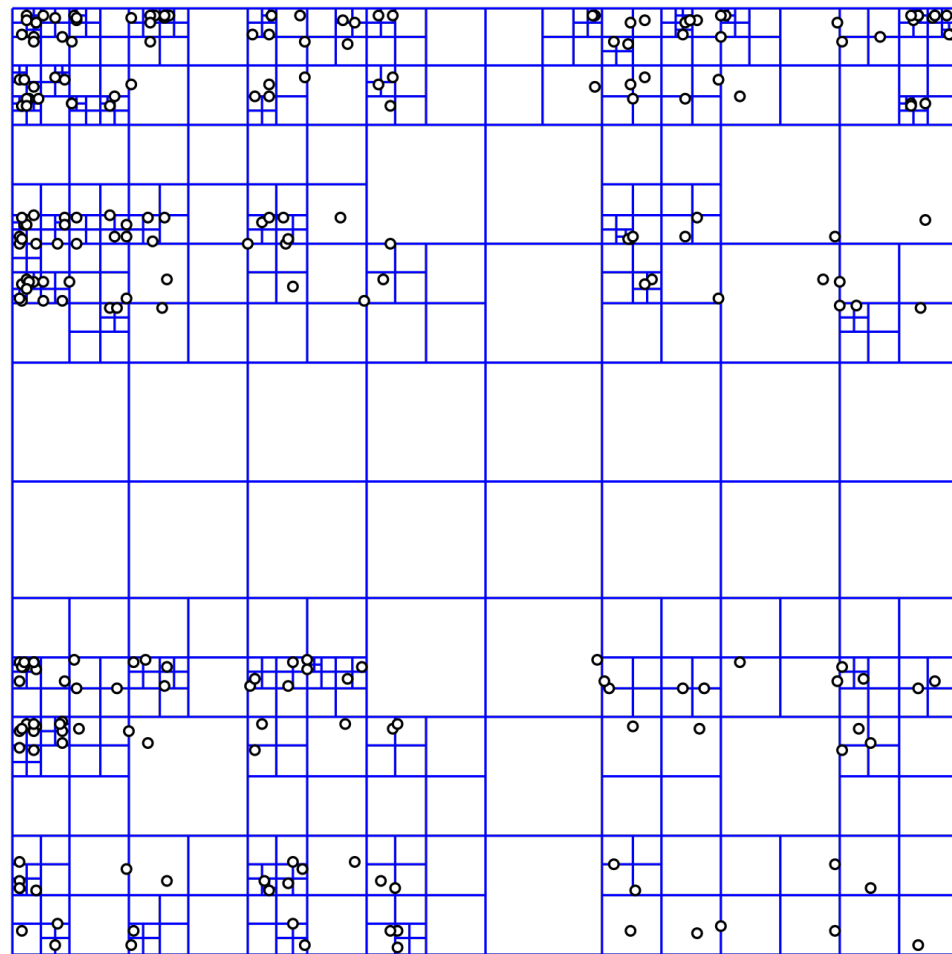
- + Exponential cardinality reduction (dividing points in half)
- + Easy location of points ($O(\log n)$ if balanced)
- bounding boxes can have large ratios



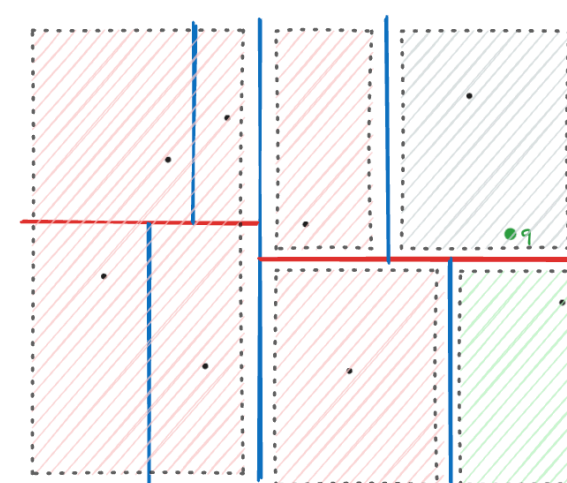
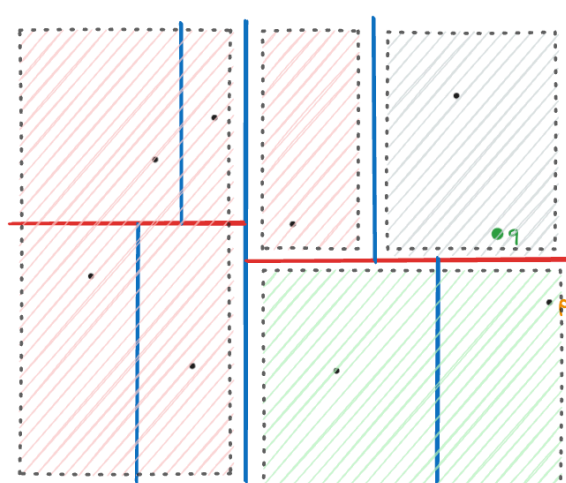
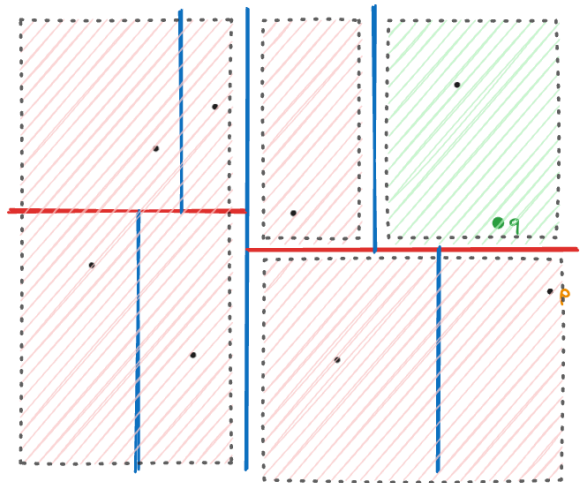
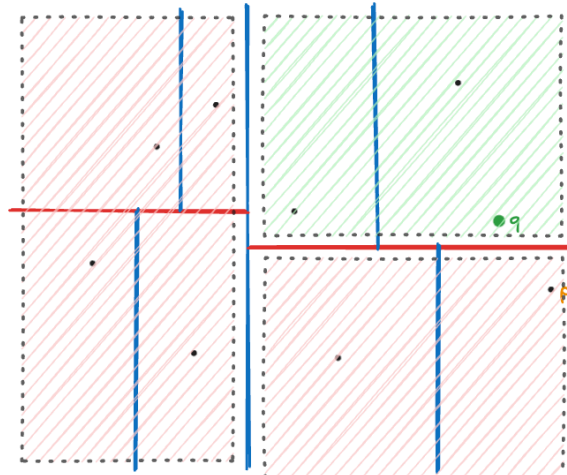
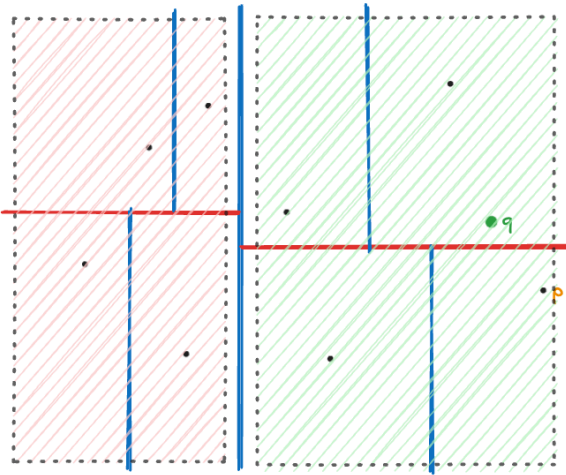
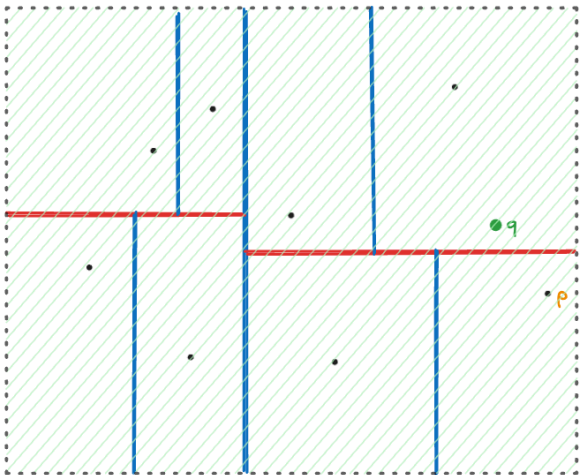
NN, ANN structures comparison

Quadrees

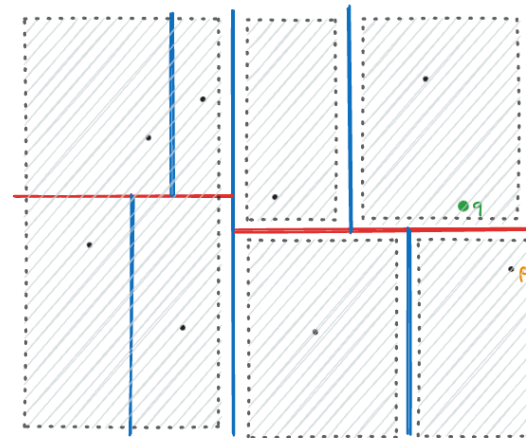
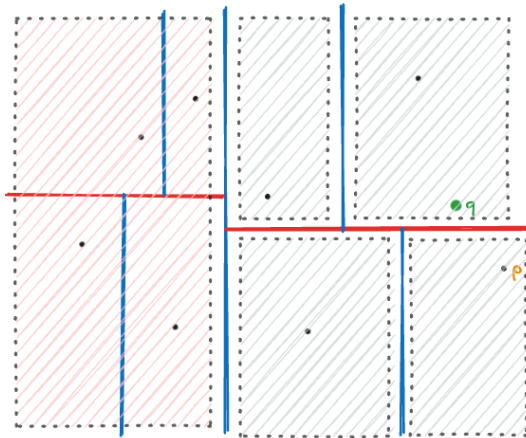
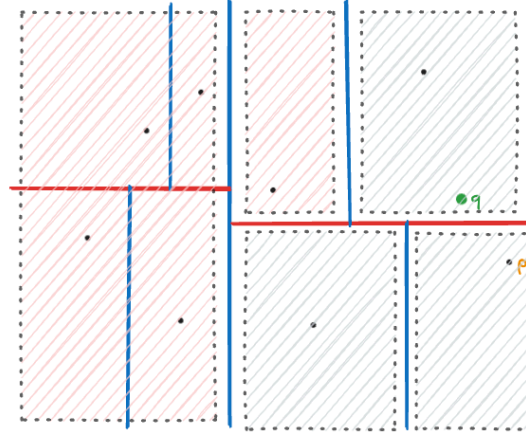
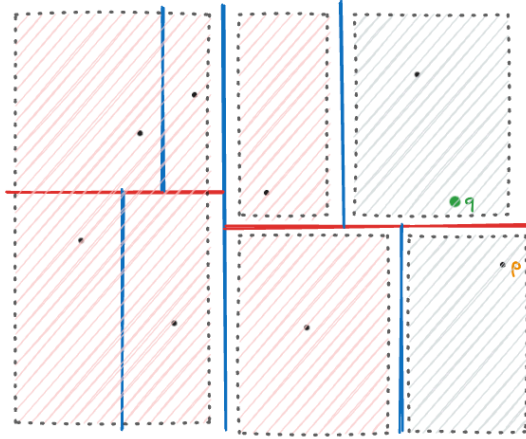
- + Exponential geometric size reduction
- unbalanced



NN query, priority search

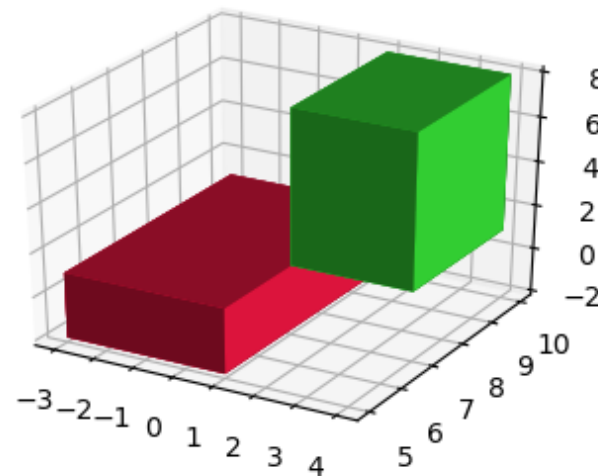


NN query, priority search



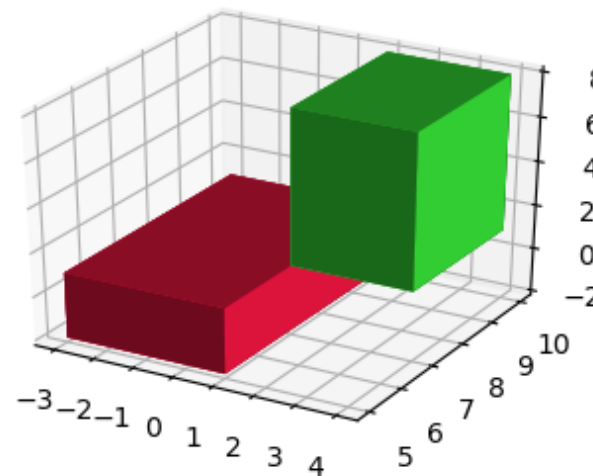
Balanced Box-Decomposition tree

Def. (Rectangle): Rectangle in $\mathbb{R}^d$ is d -fold product of closed intervals on the coordinate axes.



Balanced Box-Decomposition tree

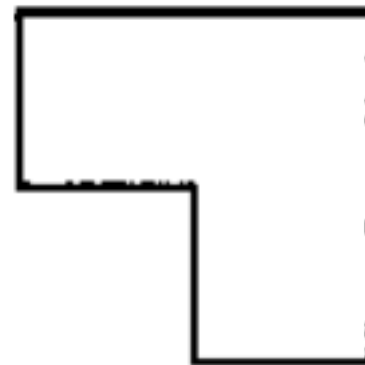
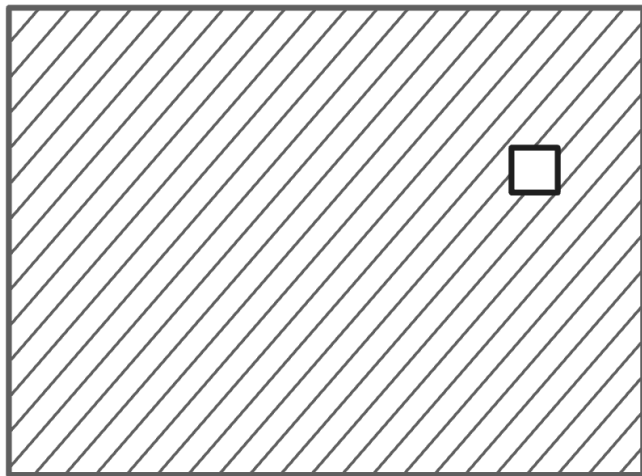
Def. (Rectangle): Rectangle in $\mathbb{R}^d$ is d -fold product of closed intervals on the coordinate axes.



Def. (Box): A Rectangle whose aspect ratio is bounded (≤ 3)

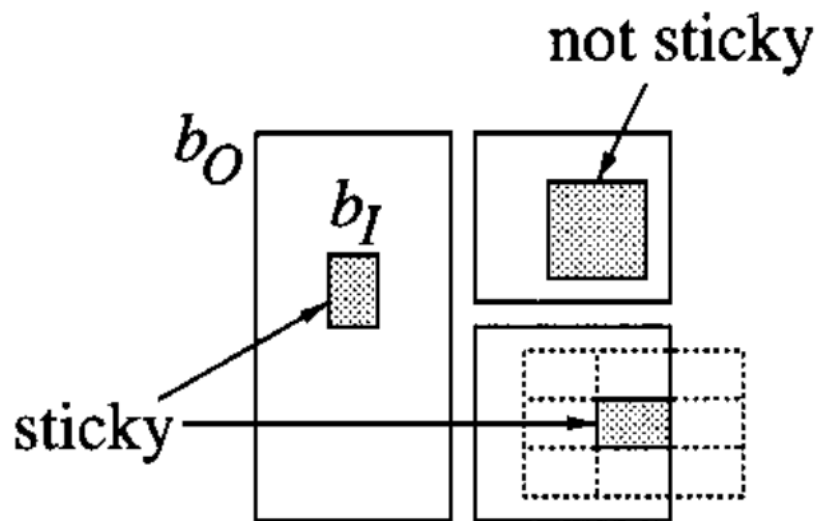
Balanced Box-Decomposition tree

Def. (Cell): A box or a set theoretic difference of 2 boxes, one enclosed within the other



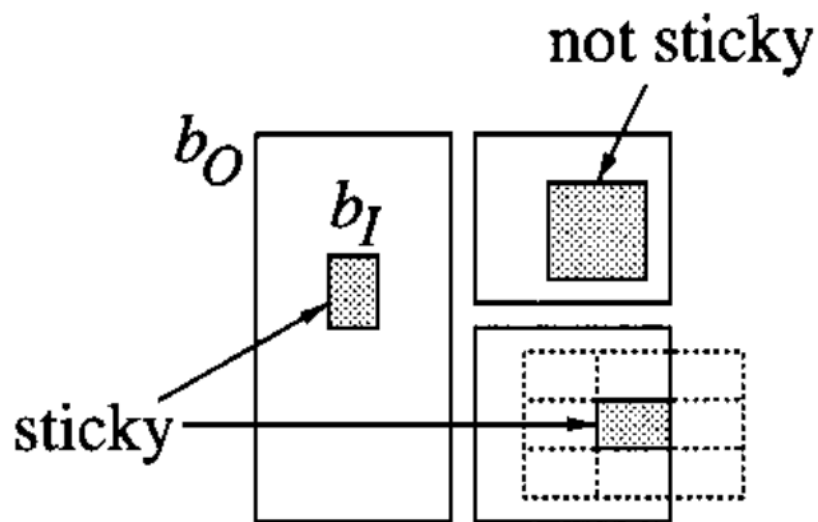
Balanced Box-Decomposition tree

Def. (Sticky): Inner box b_I is *sticky* to the outer box b_O if in each dimension it either touches the b_O or is one length apart from it.



Balanced Box-Decomposition tree

Def. (Sticky): Inner box b_I is *sticky* to the outer box b_O if in each dimension it either touches the b_O or is one length apart from it.

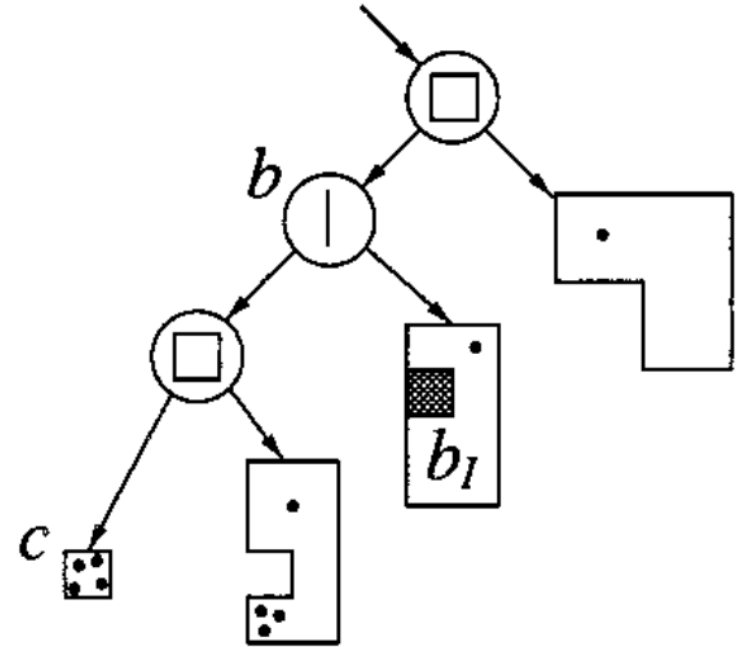
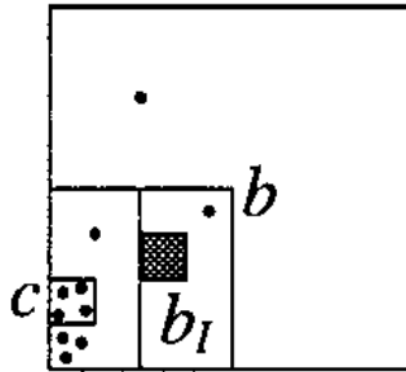
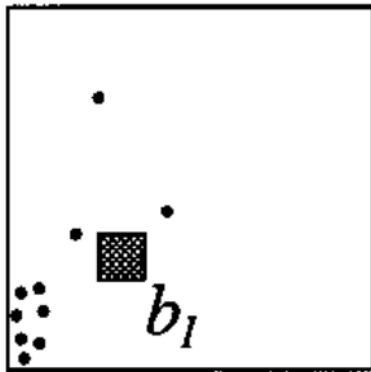


BBD tree will maintain the property that for each cell the inner box is sticky to the outer box.

BBD tree

Balanced Box-Decomposition tree. A spatial decomposition tree.

- Each node partitions space into two **cells**.
- Each leaf cell is associated with a single point lying within it.



BBD tree

It achieves:

- exponential cardinality reduction (balanced)
- exponential geometric size reduction

Main theorem

BBD tree has:

- $O(dn \log n)$ construction time
- $O(dn)$ size
- $O(c_{d,\varepsilon} \log n)$ ANN query time
- $O((c_{d,\varepsilon} + kd) \log n)$ k-ANN query time

Main theorem

BBD tree has:

- $O(dn \log n)$ construction time
- $O(dn)$ size
- $O(c_{d,\varepsilon} \log n)$ ANN query time
- $O((c_{d,\varepsilon} + kd) \log n)$ k-ANN query time

Note

Once the data structure has been built, queries can be answered for any error bound and for any Minkowski metric

Main theorem

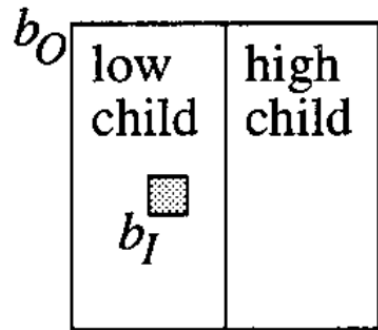
Theorem: Consider a set S of n data points in $\mathbb{R}^d$. There is a constant $c_{d,\varepsilon}$, such that in $O(dn \log n)$ time it is possible to construct a data structure of size $O(dn)$, such that for any Minkowski metric:

- given any $\varepsilon > 0$ and $q \in \mathbb{R}^d$, a $(1 + \varepsilon)$ -approximate nearest neighbor of q in S can be reported in $O(c_{d,\varepsilon} \log n)$ time
- More generally, given $\varepsilon > 0$, $q \in \mathbb{R}^d$, and any k , $1 \leq k \leq n$, a sequence of k $(1 + \varepsilon)$ -approximate nearest neighbors of $q \in S$ can be computed in $O((c_{d,\varepsilon} + kd) \log n)$ time.

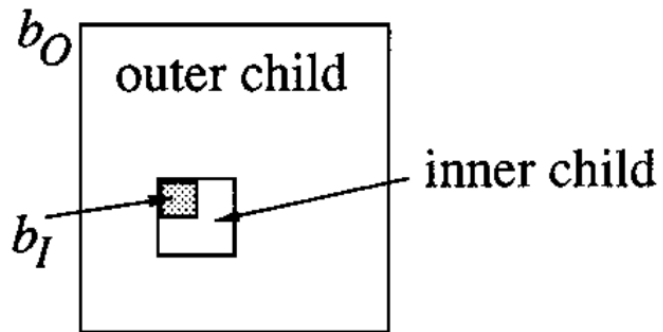
BBD construction

Operations / tree nodes

Fair split



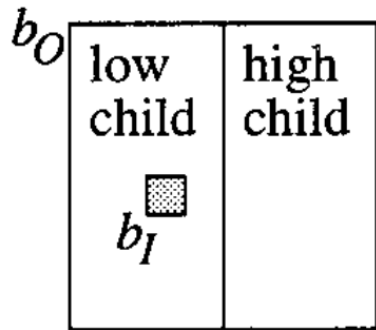
Shrink



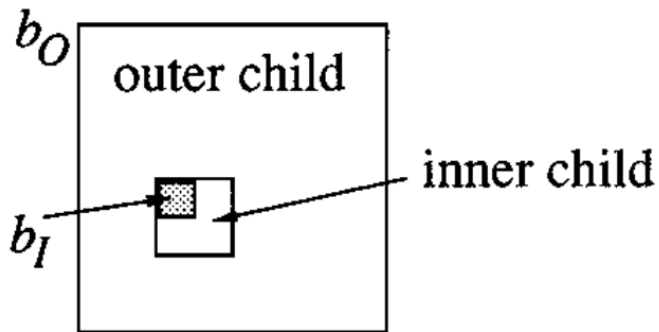
BBD construction

Operations / tree nodes

Fair split



Shrink



Construction invariants

- All boxes satisfy aspect ratio bound (3).
- Inner box of the parent lies in one of the children (inner for shrink).
- Inner boxes are sticky for their enclosing outer boxes

Split – lowers geom. complexity

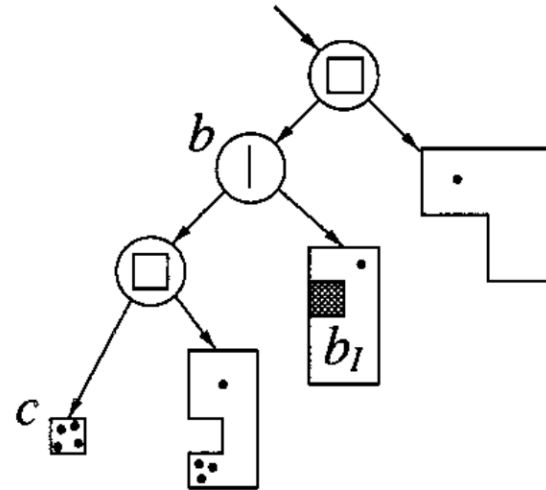
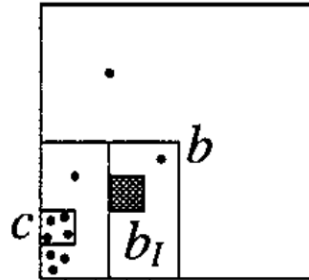
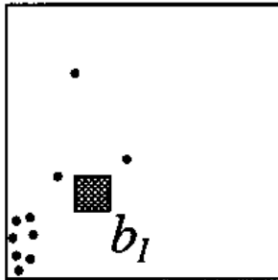
Shrink – maintains balance

BBD construction

construction process

Recursive construction, given a cell and a subset of points subdivide the cell. Do it until there is only 1 point left.

Easiest to analyze: Apply shrinks and splits interchangeably.



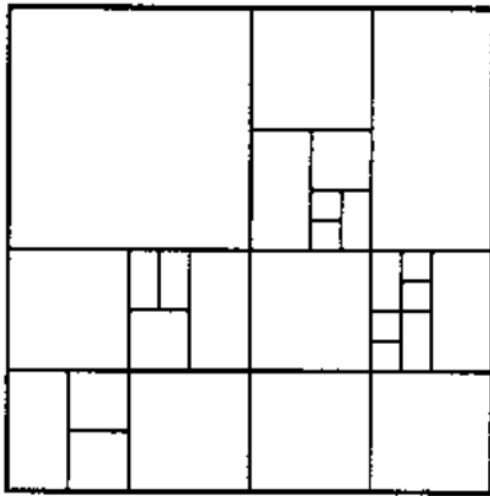
BBD construction

We want to split a cell C with n_c points in it.

BBD construction

We want to split a cell C with n_c points in it.

Midpoint algorithm



Split

Perform splits cyclically in each dimension. Always split in half.

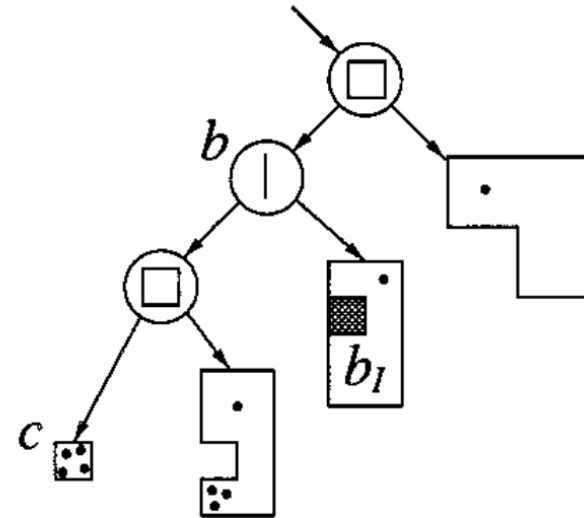
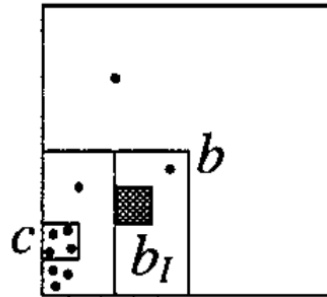
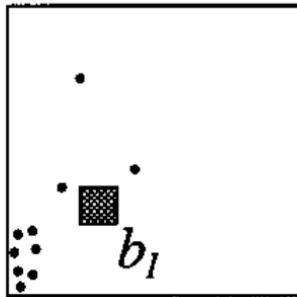
If the starting cell was $[0, 1]^d$, all sides are length 2^{-i} , $i \in \mathbb{Z}_{\geq 0}$.

Fair split = midpoint split

BBD construction

Centroid shrink

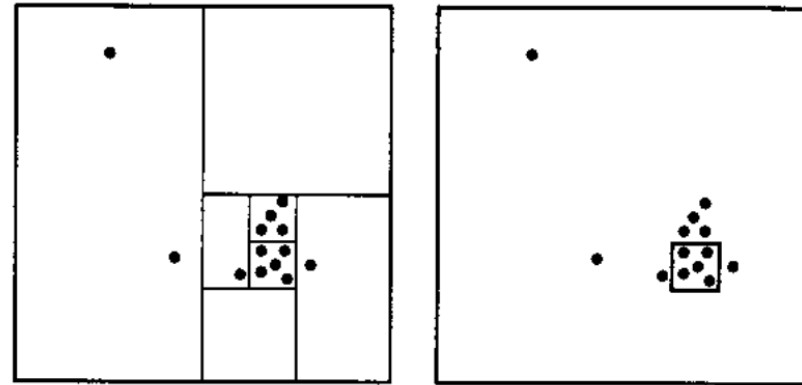
Produces up to 3 nodes, 2 shrinking, 1 splitting.



Goal: split into constant number of subcells, each with $\leq \frac{2}{3}n_c$ points

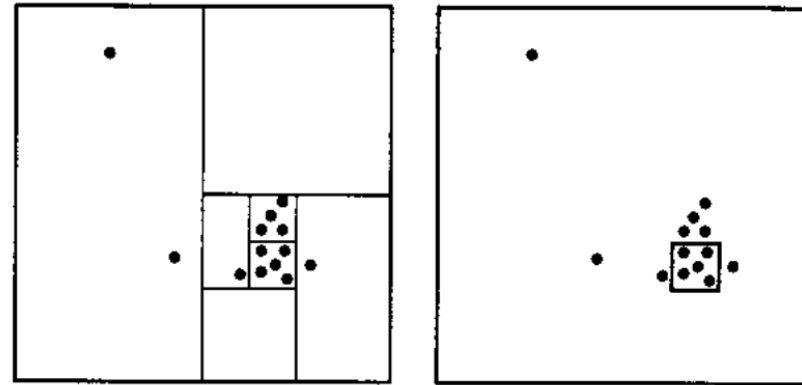
BBD construction – midpoint algorithm (simplified)

- apply midpoint split, if each subcell contains $\leq \frac{2}{3}n_c$ points, done. Else recursion in the cell containing larger number of points.
- the outer box of the larger of 2 last cells created becomes an inner box in the shrink B_I' . Discard all other splits done in the process.



BBD construction – midpoint algorithm (simplified)

- apply midpoint split, if each subcell contains $\leq \frac{2}{3}n_c$ points, done. Else recursion in the cell containing larger number of points.
- the outer box of the larger of 2 last cells created becomes an inner box in the shrink B_I' . Discard all other splits done in the process.

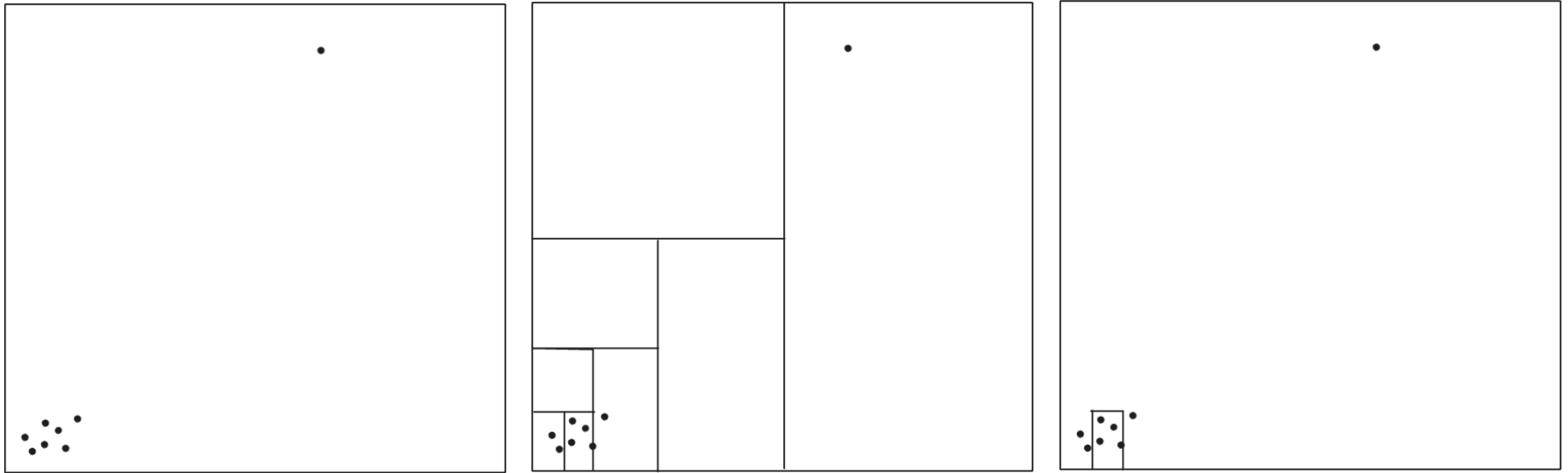


Outside the inner box: $\leq \frac{2}{3}n_c$ points, in the inner cell: $\leq \frac{2}{3}n_c$ points.

BBD construction – midpoint algorithm

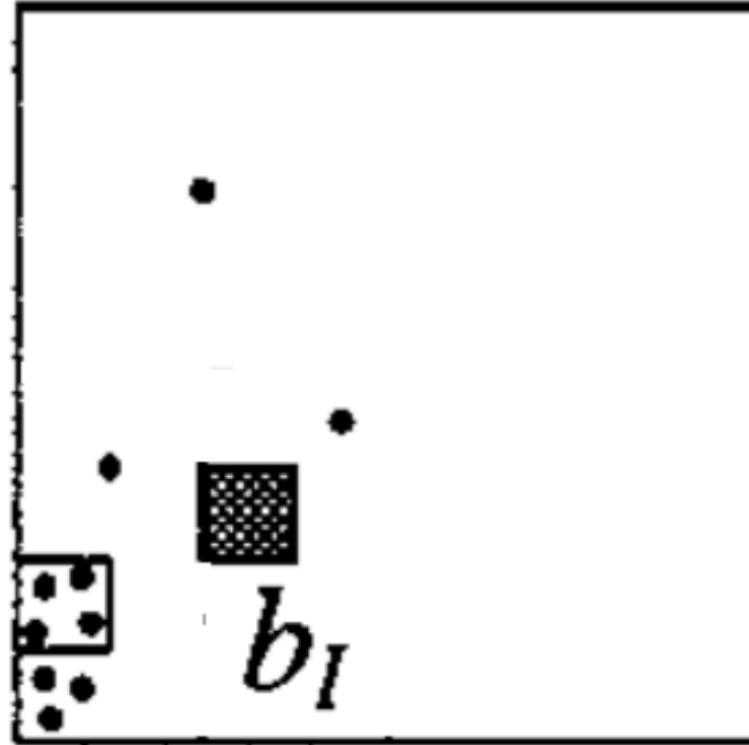
Problems:

1. The number of splits is not a function of n .



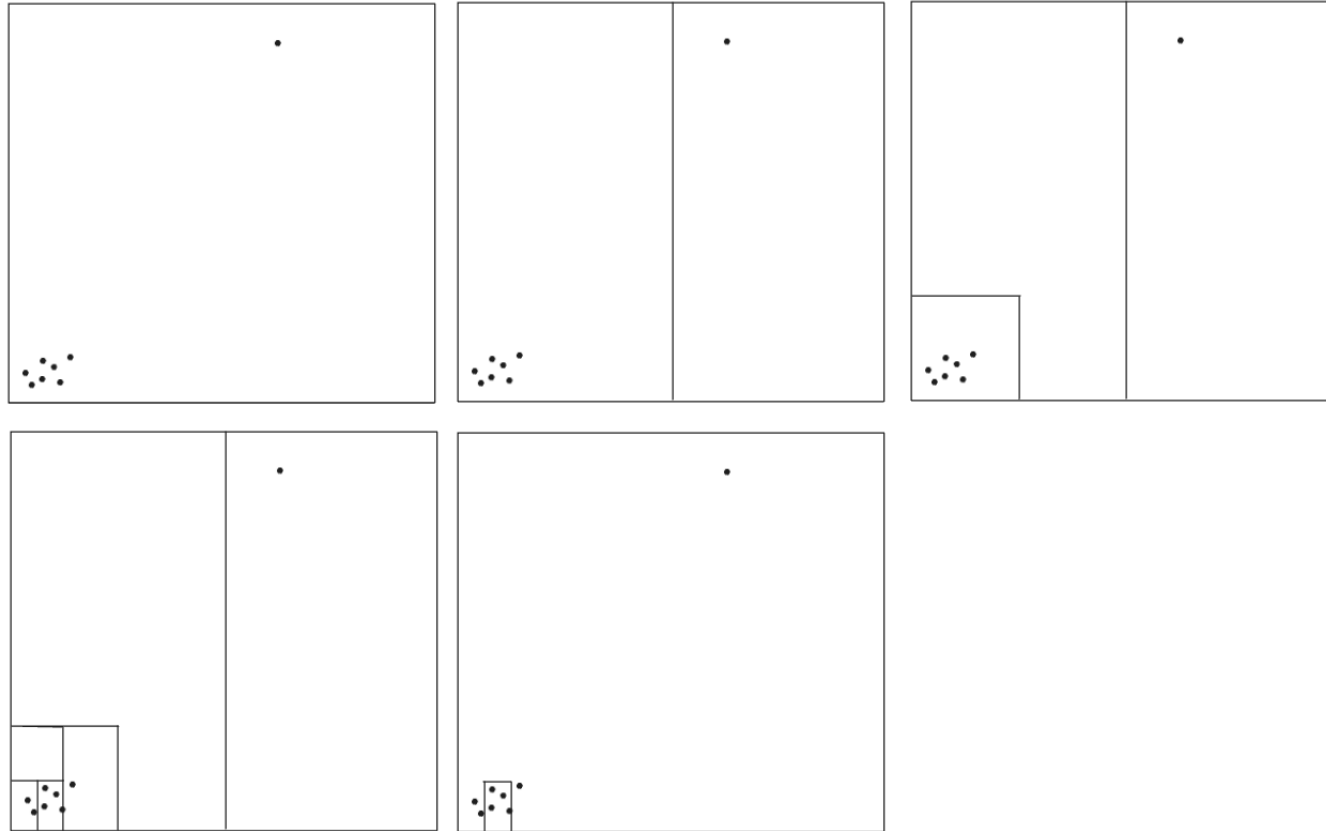
BBD construction – midpoint algorithm

2. The shrinking box may not contain the inner box of the original cell.



BBD construction – midpoint algorithm

Solution 1: Compute a minimum bounding box before each split.

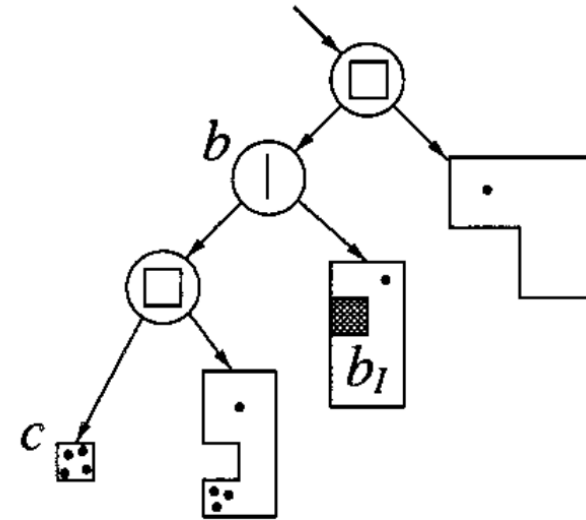
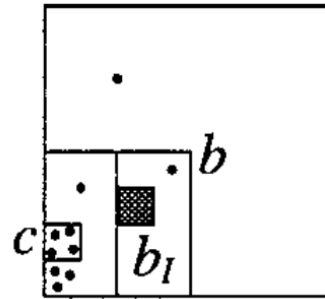
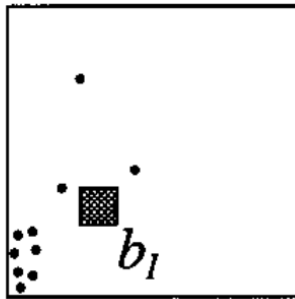


BBD construction – midpoint algorithm

Solution 2: Replace 1 shrink with a shrink, split, shrink sequence.

Modify the centroid shrinking such that:

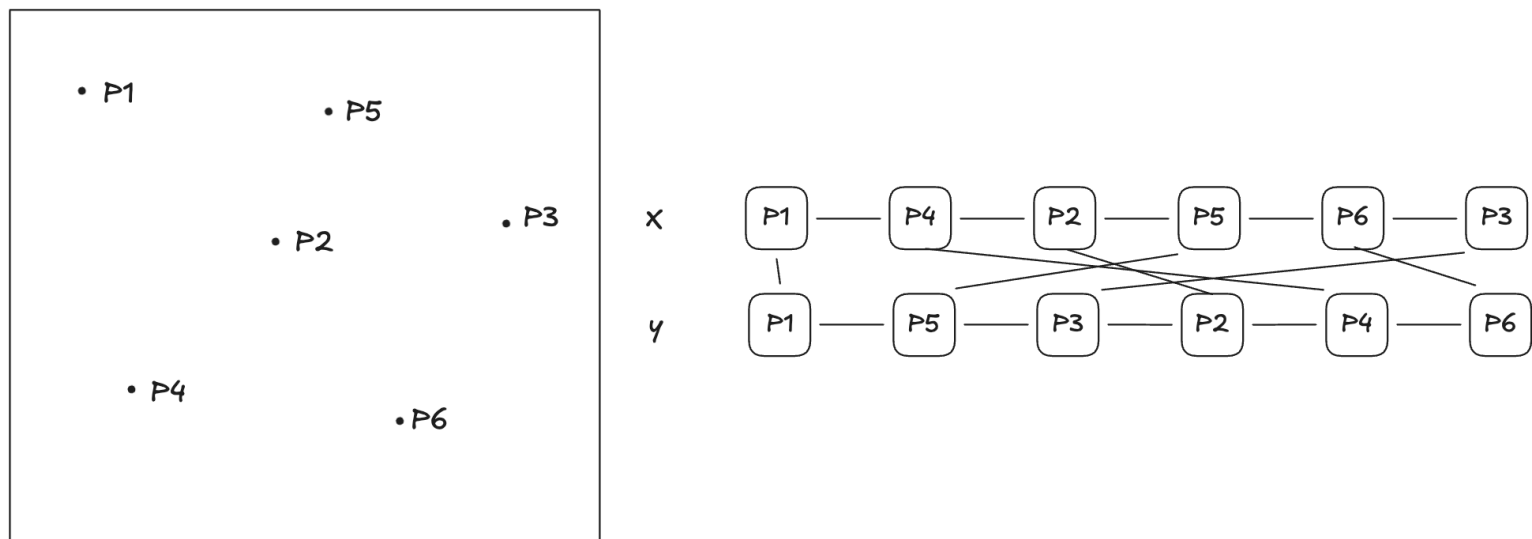
- when computing minimal enclosing box, include the B_I in it
- if B_I lies on the side with less points, then shrink and this split.



Construction complexity

Partitioning points

Store d separate linked lists of all of the points (just id's), one for each dimension.



$O(dn)$ space, $O(dn \log n)$ construction time

Construction complexity

Complexity

Alternating splits and shrinks $\Rightarrow$ tree has $O(\log n)$ height.

Binary tree with $O(\log n)$ height, takes $O(nd)$ space.

Construction complexity

Complexity

Alternating splits and shrinks $\Rightarrow$ tree has $O(\log n)$ height.

Binary tree with $O(\log n)$ height, takes $O(nd)$ space.

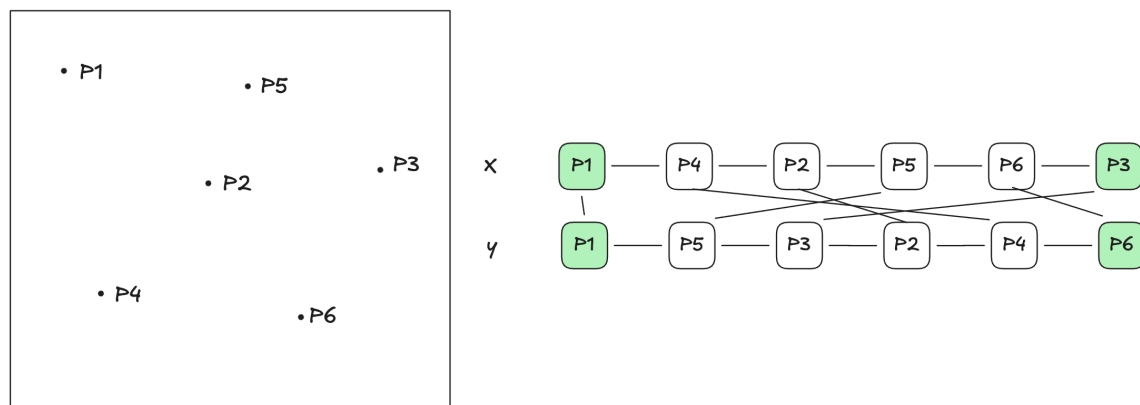
Construction time $O(dn \log n)$

- $O(dn \log n)$: to sort the lists
- $O(dn \log n)$: recursive construction ($\log n$ levels on each linear work)

Construction complexity

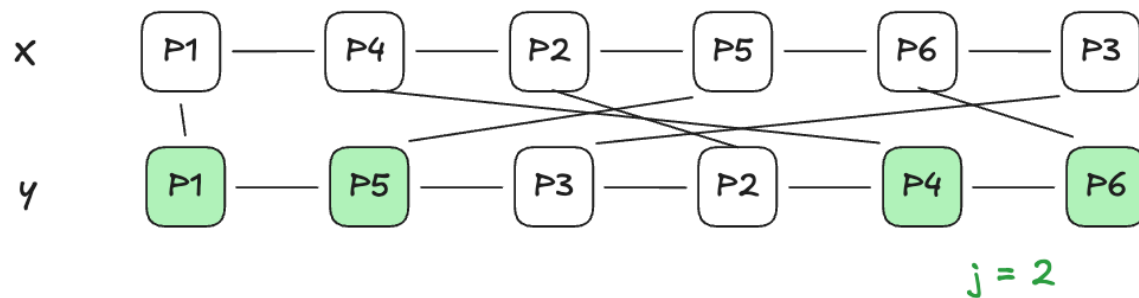
Lemma: Given a parent node associated with n_c points, each split can be performed in $O(dn_c)$ time.

Use the sorted lists to compute minimal enclosing boxes in $O(d)$ time.



Construction complexity

During shrink eliminate j points from the smaller side of the split in $O(dj)$ time. Walk the i -th list from both sides to find the splitting point in $O(j)$ time. split points into new lists in $O(dn_c)$ time.



BBD tree modifications

We want each leaf cell to have a point inside.

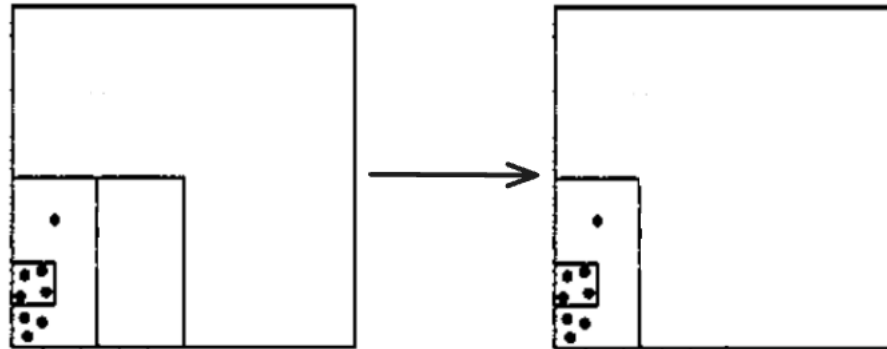
Trivial splits/shrinks – one cell in the partition is empty.

BBD tree modifications

We want each leaf cell to have a point inside.

Trivial splits/shrinks – one cell in the partition is empty.

1. Change consecutive trivial splits/shrinks to single trivial shrinks.
 $O(n)$ time.

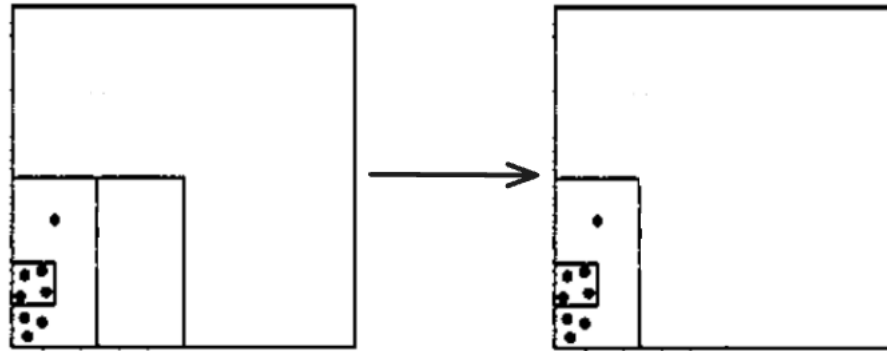


BBD tree modifications

We want each leaf cell to have a point inside.

Trivial splits/shrinks – one cell in the partition is empty.

1. Change consecutive trivial splits/shrinks to single trivial shrinks.
 $O(n)$ time.

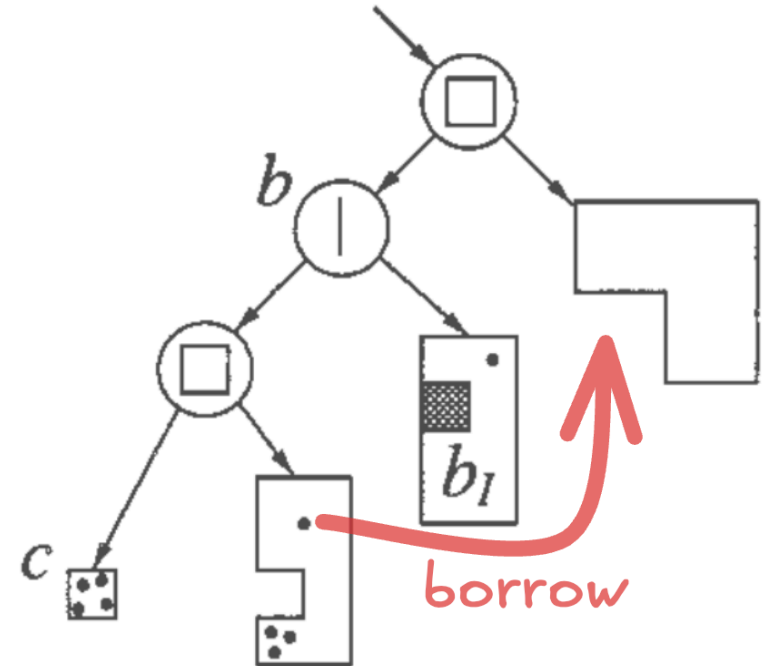
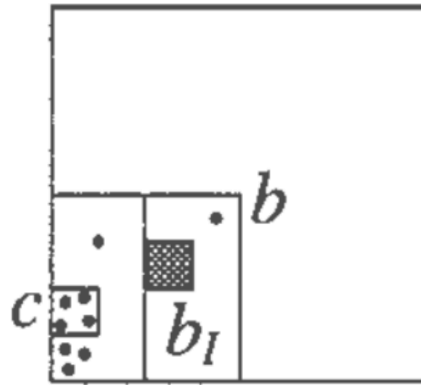
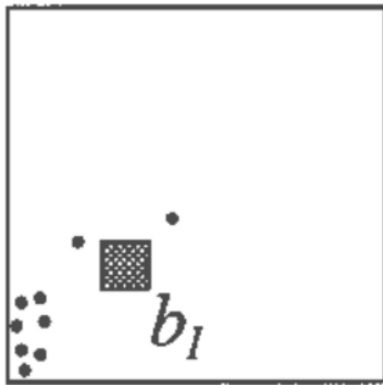


2. **Borrow** points such that each leaf cell has at least one point.

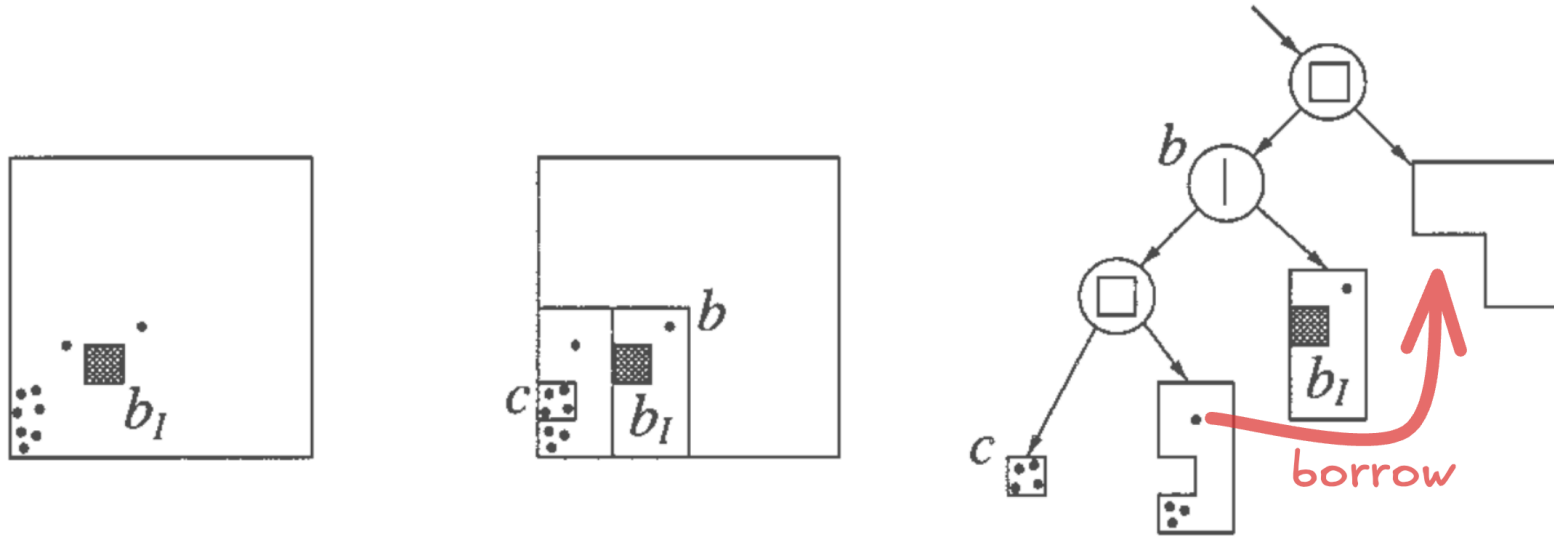
BBD tree modifications – borrow

Each leaf gets a point, each point is in 1 or 2 leaves. Procedure:

- nontrivial nodes: take 1 point from each child node. Pass 2 up.
- trivial nodes (!shrink): Pass 1 point up, use 1 to fill the empty leaf cell.



BBD tree modifications – summary

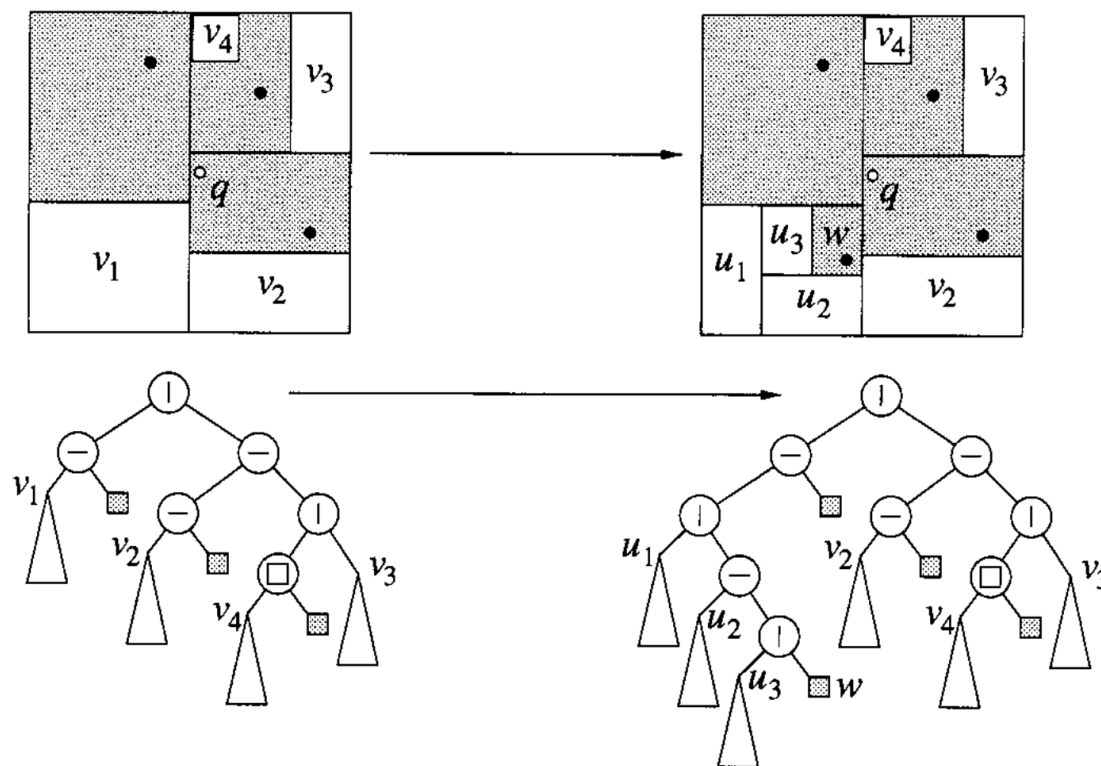


- The tree height is $O(\log n)$, every 4 levels n_c drops by a factor of $\frac{2}{3}$
- The cells have 1:2 or 1:1 aspect ratio. Every $4d$ levels sizes drop by $\frac{1}{2}$.
- Each leaf gets a point which is either in the cell or in the inner box, each point is in 1 or 2 leaves.

BBD ANN query – priority search

The query is the priority search as for the kd-trees.

Distance to cell := distance to the outer box.



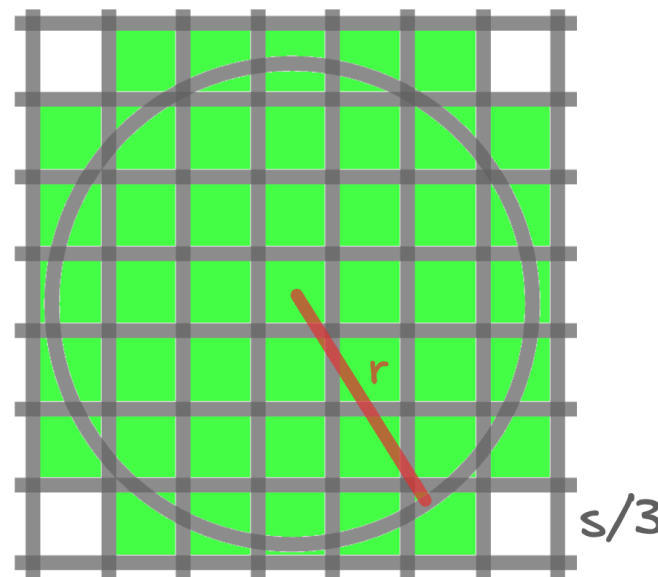
Query complexity

Lemma (Packing constraint): Given a BBD-tree for a set of points in R^d , the number of leaf cells of size at least $s > 0$ that intersect a Minkowski L_m ball of radius r is at most $\lceil 1 + 6\frac{r}{s} \rceil^d$.

Proof:

Smallest side length of a box of size s is $\frac{s}{3}$.

Enclose the ball in a hypercube of side length $2r$.*



Query complexity

The time to enumerate m nearest cells to the query point:
 $O(md \log n)$.

The search terminates if the distance to current cell is $> \frac{d(p,q)}{1+\epsilon}$.

q – query point,

p – currently closest point

Observation

If the last non-terminating cell visited was distance r from q .

Then $r \leq \frac{d(p,q)}{1+\epsilon}$, and

$r(1 + \epsilon) \leq d(p, q)$, thus all leaf cells visited had size $\geq \frac{r\epsilon}{d}$. Else they have a closer point than p .

