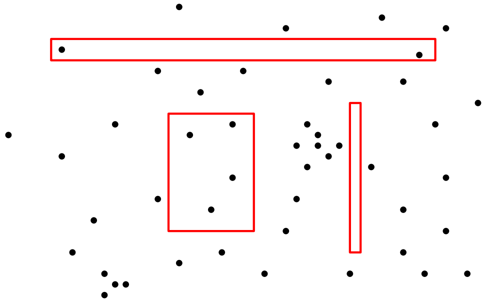
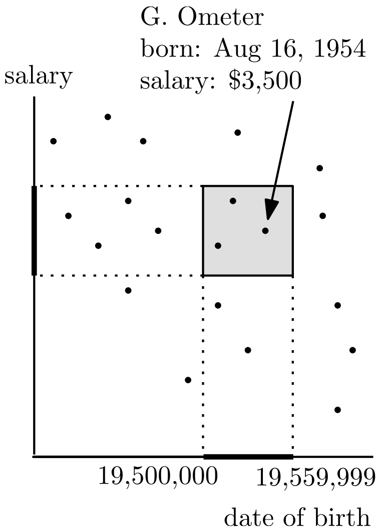
Range queries in 2D



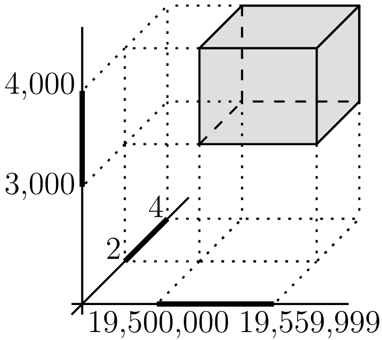
Database queries

A database query may ask for all employees with age between a_1 and a_2, and salary between s_1 and s_2



Database queries

Example of a 3-dimensional (orthogonal) range query: children in $[2, 4]$, salary in $[3000, 4000]$, date of birth in $[19{,}500{,}000\ ,\ 19{,}559{,}999]$



1D range query problem

1D range query problem: Preprocess a set of n points on the real line such that the ones inside a 1D query range (interval) can be reported fast

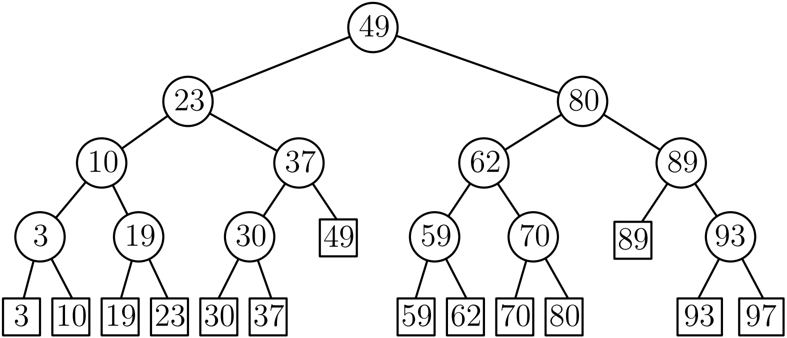
The points $p_1, \ldots, p_n$ are known beforehand, the query $[x, x']$ only later

A solution to a query problem is a data structure description, a query algorithm, and a construction algorithm

Question: What are the most important factors for the *efficiency* of a solution?

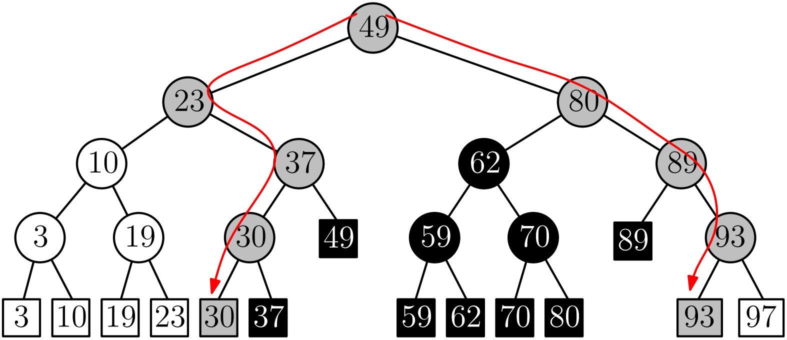
Balanced binary search trees

A balanced binary search tree with the points in the leaves



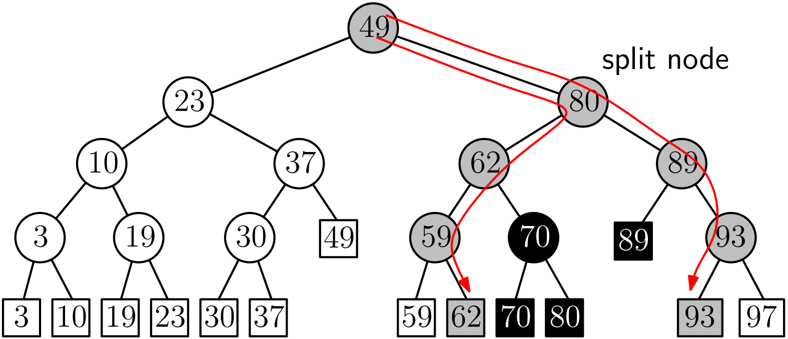
Example 1D range query

A 1-dimensional range query with $[25, 90]$



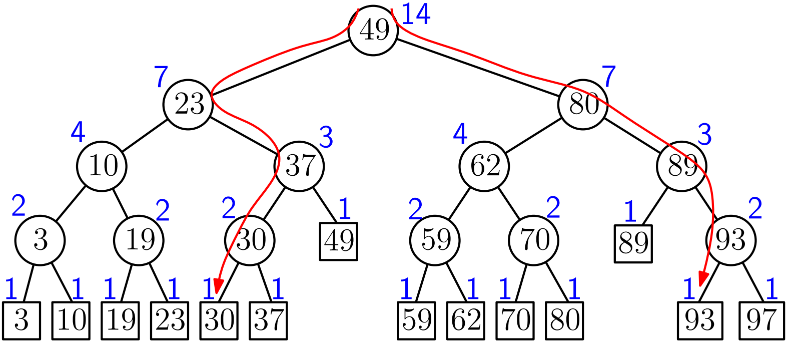
Example 1D range query

A 1-dimensional range query with $[61, 90]$



Example 1D range counting query

A 1-dimensional range counting query with $[25, 90]$



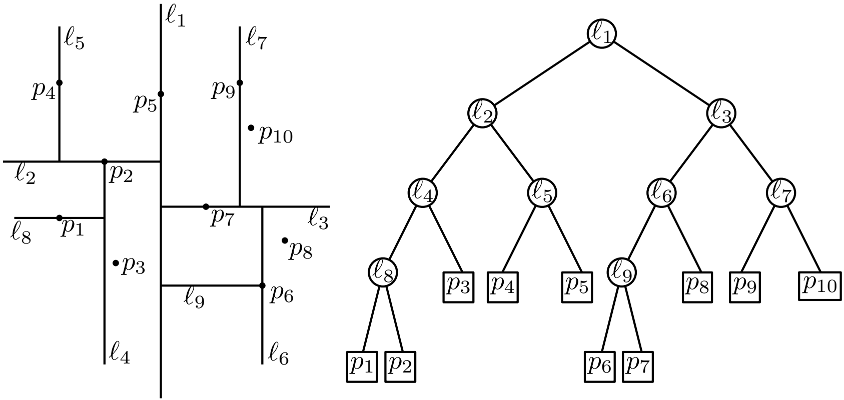
Kd-trees

Kd-trees, the idea: Split the point set alternatingly by x-coordinate and by y-coordinate

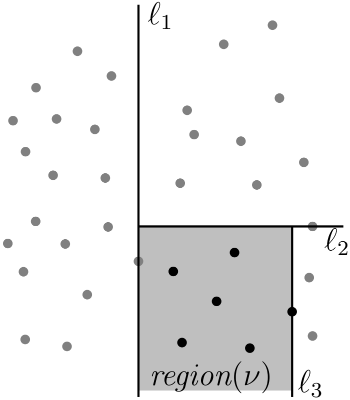
split by x-coordinate: split by a vertical line that has half the points left and half right

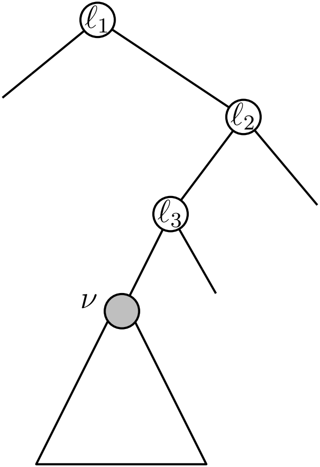
split by y-coordinate: split by a horizontal line that has half the points below and half above

Kd-trees

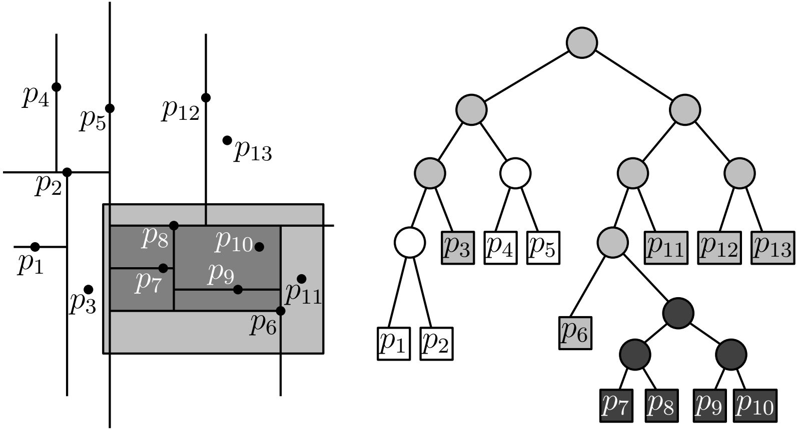


Kd-tree regions of nodes

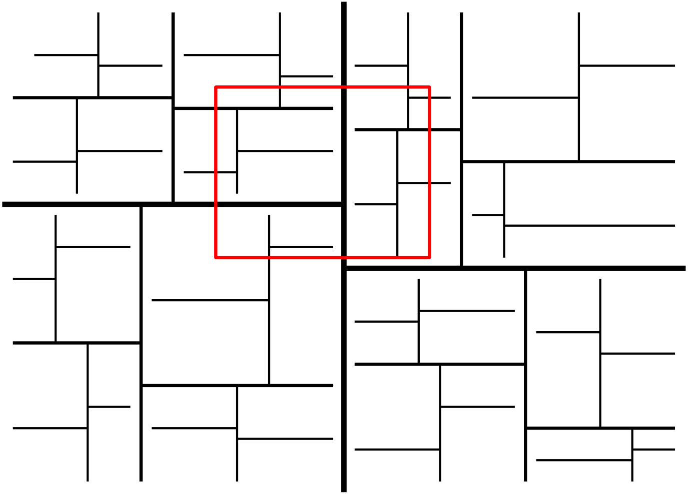




Kd-tree querying

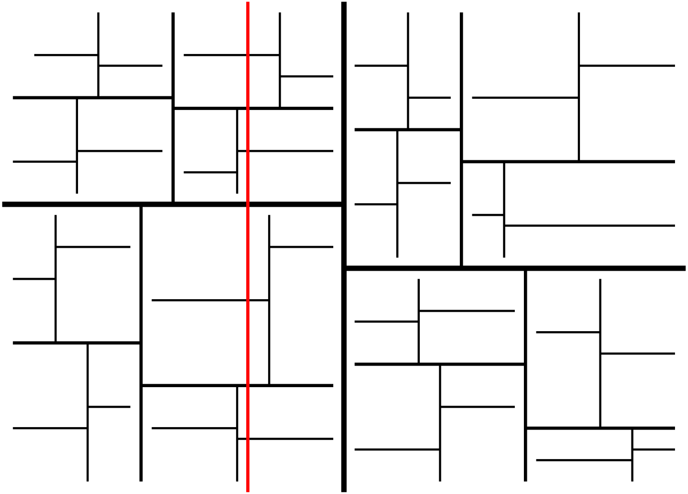


Kd-tree query time analysis



Question: How many grey and how many black *nodes*?

Kd-tree query time analysis



Question: How many grey and how many black *leaves*?

Kd-tree query time analysis

We observe: At every vertical split, ℓ is only to one side, while at every horizontal split ℓ is to both sides

Let $G(n)$ be the number of grey nodes in a kd-tree with n points (leaves). Then $G(1) = 1$ and:

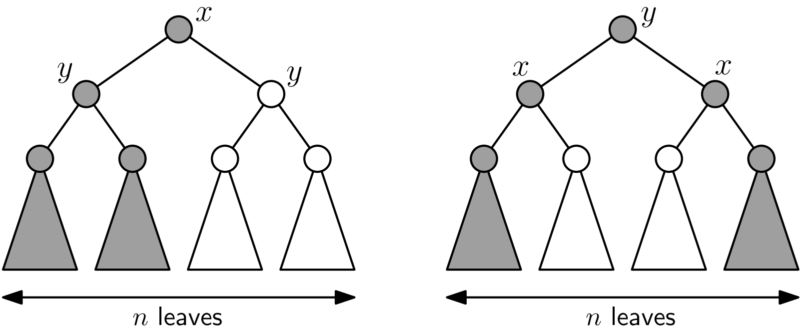
If a subtree has n leaves: $G(n) = 1 + G(n/2)$ at even depth

If a subtree has n leaves: $G(n) = 1 + 2 \cdot G(n/2)$ at odd depth

If we use *two levels at once*, we get:

$$G(n) = 2 + 2 \cdot G(n/4) \quad \text{or} \quad G(n) = 3 + 2 \cdot G(n/4)$$

Kd-tree query time analysis



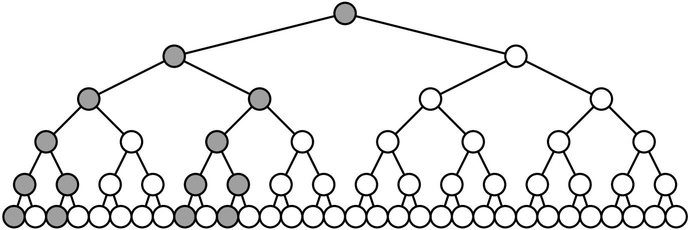
Kd-tree query time analysis

$G(1) = 1$

$G(n) = 2 \cdot G(n/4) + O(1)$

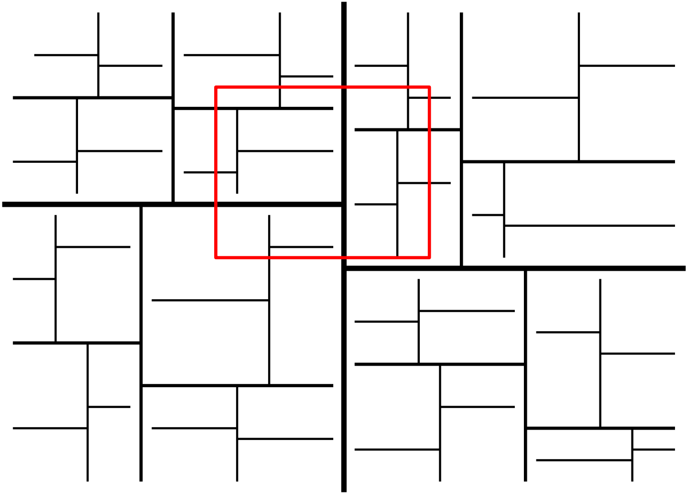
Question: What does this recurrence solve to?

Kd-tree query time analysis

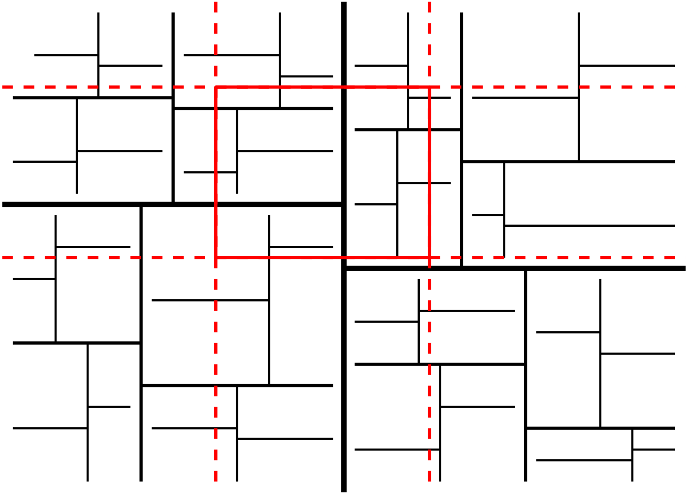


The grey subtree has unary and binary nodes

Kd-tree query time analysis



Kd-tree query time analysis



Result

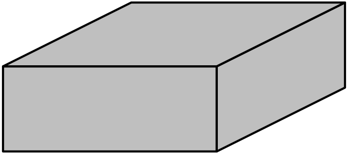
Theorem: A set of n points in the plane can be preprocessed in $O(n \log n)$ time into a data structure of $O(n)$ size so that any 2D range query can be answered in $O(\sqrt{n}+k)$ time, where k is the number of answers reported

For range counting queries, we need $O(\sqrt{n})$ time

Higher dimensions

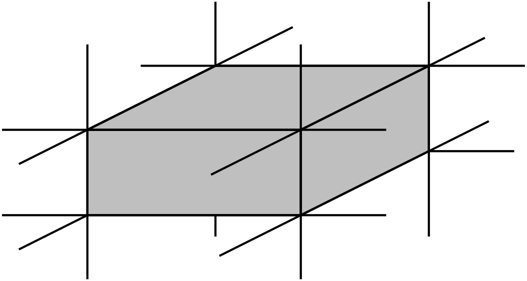
A 3-dimensional kd-tree alternates splits on x-, y-, and z-coordinate

A 3D range query is performed with a box



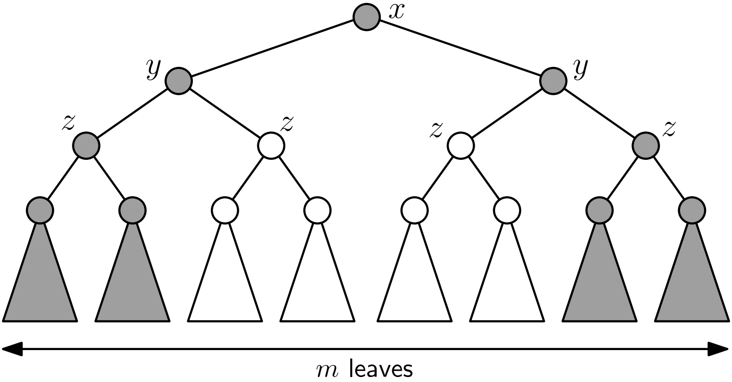
Higher dimensions

How does the query time analysis change?



Intersection of B and $region(v)$ depends on intersection of facets of B $\Rightarrow$ analyze by axes-parallel planes (B has no more grey nodes than six planes)

Higher dimensions



Kd-tree query time analysis

Let $G_3(n)$ be the number of grey nodes for a query with an axes-parallel plane in a 3D kd-tree

$G_3(1) = 1$

$G_3(n) = 4 \cdot G_3(n/8) + O(1)$

Question: What does this recurrence solve to?

Question: How many leaves does a perfectly balanced binary search tree with depth $\frac{2}{3}\log n$ have?

Result

Theorem: A set of n points in d-space can be preprocessed in $O(n \log n)$ time into a data structure of $O(n)$ size so that any d-dimensional range query can be answered in $O(n^{1-1/d} + k)$ time, where k is the number of answers reported