k Nearest Neighbors algorithm

Outline

- KNN
  - Background
  - Definition
  - Distance measures
  - How to choose k
  - Digit recognition
  - K-nn properties
- K-D TREE CONSTRUCTION & QUERY
- HOMEWORK

Origins of K-NN

- Nearest Neighbors have been used in statistical estimation and pattern recognition in the beginning of 1970’s (non-parametric techniques).

- The method prevailed in several disciplines and still it is one of the top 10 Data Mining algorithm.

the average of the 5 people you spend the most time
網易公开课
– Jim
“你是你最常接触的五个人的平均值。”
"You are the average of the five people you spend the most time with."

You are the average of the five people you spend most time with.

—Jim Rohn

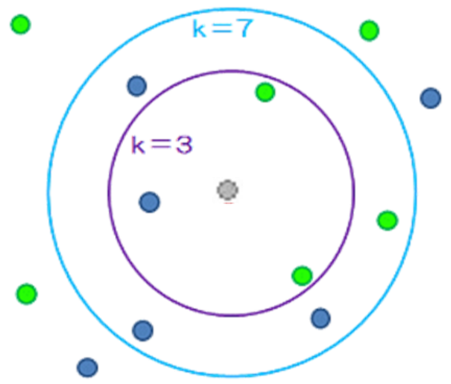
Scott Dinsmore:: How to find work you love| TED Talk

- It’s how people judge by observing our peers.

- We tend to move with people of similar attributes so does data

Definition

- K-Nearest Neighbor is considered a lazy learning algorithm that classifies data sets based on their similarity with neighbors.
- “K” stands for number of data set items that are considered for the classification.



Note: K -Nearest Neighbors is called a non-parametric method
Unlike other supervised learning algorithms, K -Nearest Neighbors doesn’t learn an explicit mapping f from the training data
It simply uses the training data at the test time to make predictions

Technically

- Given training data$\{(x_1, y_1), \dots, (x_N, y_N)\}$ and a test point x
  - N pairs ; x_i is a vector consisting of D features , y_i - label
- Goal: predict the output y for an unseen test example x
- Prediction rule: look at the K most similar training examples

Technically

- Forms of the output:

  for classification: a discrete variable $y_i \in \{1, \dots, C\}$
  assign the majority class label (majority voting)

  for regression： a continuous (real-valued) variable $y_i \in R$
  assign the average response

- The algorithm requires:

  Parameter K : number of nearest neighbors to look for

  Distance function: To compute the similarities between examples

K-NN: Some distance measures

- **Euclidean distance** is commonly used

In the Euclidean plane, if **p** = (p_1, p_2) and **q** = (q_1, q_2) then the distance is given by

$$\mathrm{d}(\mathbf{p},\mathbf{q}) = \sqrt{(q_1-p_1)^2 + (q_2-p_2)^2}.$$

In three-dimensional Euclidean space, the distance is

$$d(p,q) = \sqrt{(p_1-q_1)^2 + (p_2-q_2)^2 + (p_3-q_3)^2}.$$

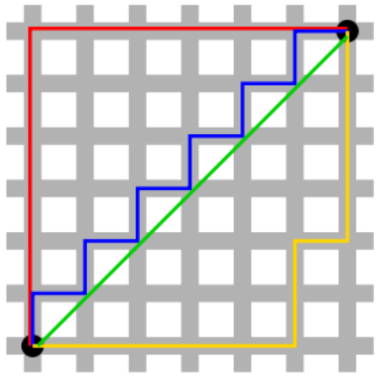
In general, for an *n*-dimensional space, the distance is

$$d(p,q) = \sqrt{(p_1-q_1)^2 + (p_2-q_2)^2 + \cdots + (p_i-q_i)^2 + \cdots + (p_n-q_n)^2}.$$

distance measures

- **Manhattan Distance**(or City Block distance)

$$d\ (\mathbf{p},\mathbf{q}) = |p_1 - q_1| + |p_2 - q_2|$$



In general, for an *n*-dimensional space, the Manhattan distance is

$$d\ (\mathbf{p},\mathbf{q}) = \|\mathbf{p} - \mathbf{q}\| \ = \sum_{i=1}^{n} |p_i - q_i|$$

distance measures

- **Minkowski Distance**(闵可夫斯基距离)

The Minkowski distance of order p between two points

$$p = (p_1, \ p_2, \ \ldots, \ p_n) \ \ \textbf{and} \ q = (q_1, \ q_2, \ \ldots, \ q_n) \in \mathbb{R}^n$$

is defined as:

$$\left(\sum_{i=1}^{n} |p_i - q_i|^m \right)^{1/m}$$

Minkowski distance is typically used with m being 1 or 2, which correspond to the Manhattan distance and the Euclidean distance, respectively.

In the limiting case of m reaching infinity, we obtain the Chebyshev distance.

distance measures

- **Cosine similarity** is a measure of similarity between two non-zero vectors of an inner product space that measures the cosine of the angle between them.

the cosine similarity, cos(θ) is represented using:

$$\text{similarity} = \cos(\theta) = \frac{A \cdot B}{\|A\|\|B\|} = \frac{\sum\limits_{i=1}^{n} A_i \times B_i}{\sqrt{\sum\limits_{i=1}^{n} (A_i)^2} \times \sqrt{\sum\limits_{i=1}^{n} (B_i)^2}}$$

distance measures

- **Hamming distance** (汉明距离) counts the number of features where the two examples disagree.

The Hamming distance between:

•**1011101** and **1001001** is 2.
•**2173896** and **2233796** is 3.

In other words, it measures the minimum number of *substitutions* required to change one string into the other .

K -NN: Feature Normalization

Note: Features should be on the same scale

Example: if one feature has its values in millimeters and another has in centimeters, we would need to normalize

One way is:

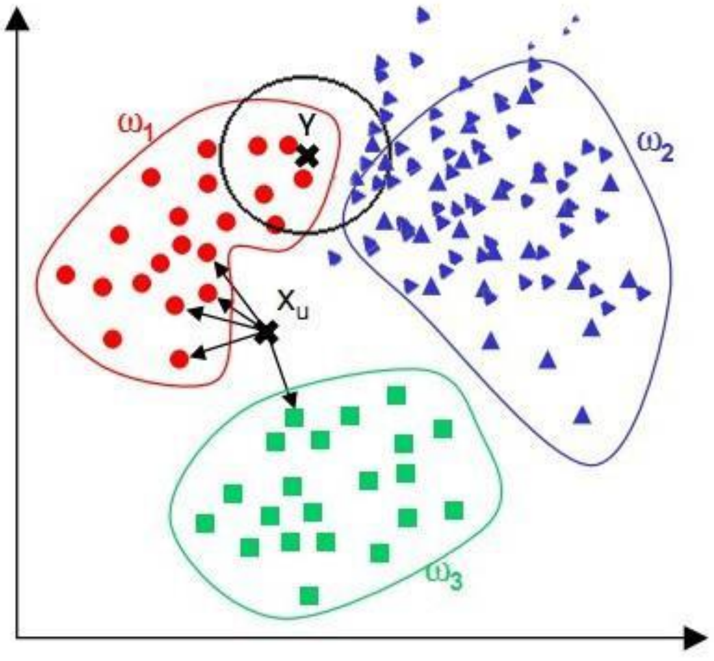
Replace p_i by $z_{im} = \frac{(x_{im} - \bar{x_m})}{\sigma_m}$ (make them zero mean, unit variance)

$\bar{x_m} = \frac{1}{N}\sum_{i=1}^{N} x_{im}$: mean of m^{th} feature

$\sigma_m^2 = \frac{1}{N}\sum_{i=1}^{N}(x_{im} - \bar{x_m})^2$: variance of m feature

K-NN: Feature Weighting

- Scale each :ation
- Can use ou res are more important
- Can learn t (not covered)



How about the weight of the sample？

K-NN: How to Choose k?

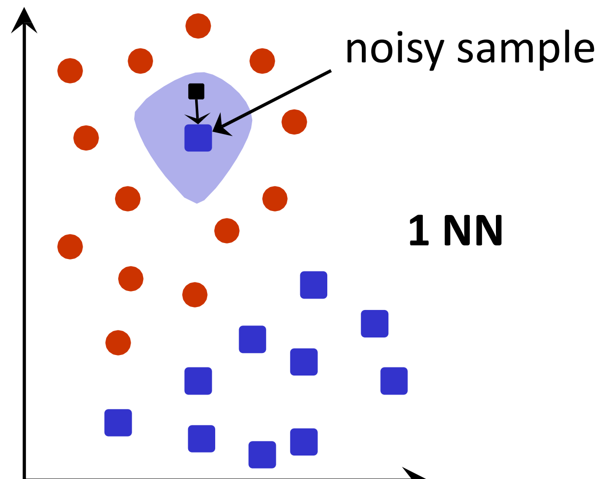
- In theory, if infinite number of samples available, the larger is k, the better is classification.

- Impossible in practice since # samples is finite

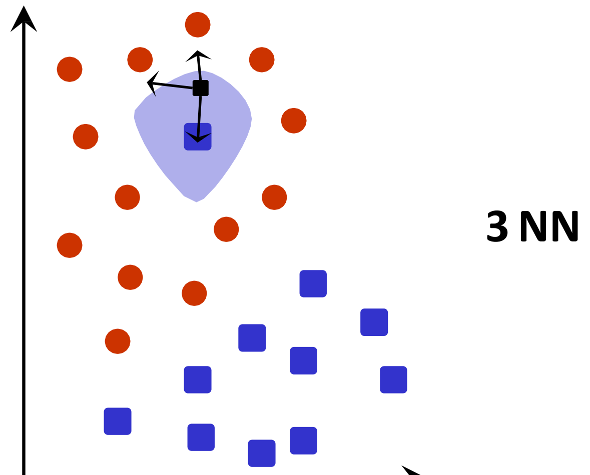
- ☐ k=1 the nearest one
- ☐ k=N the majority

K-NN: How to Choose k?

- **k** = 1 is often used for efficiency, but sensitive to "noise"



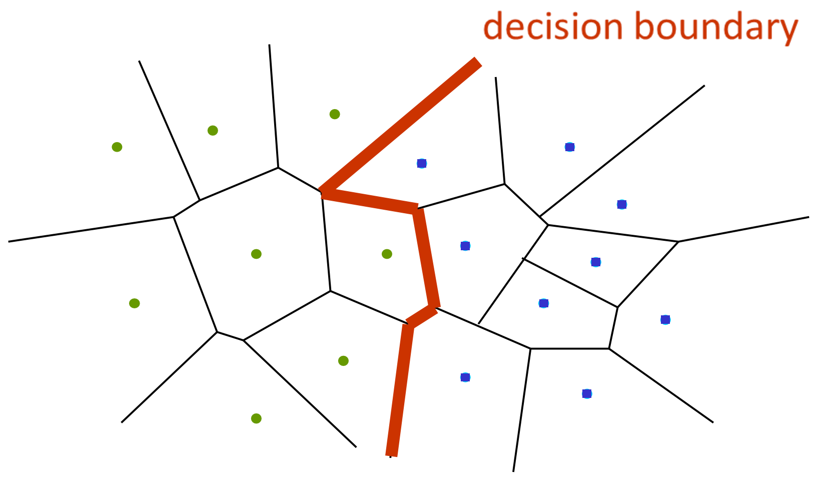
every example in the blue shaded area will be misclassified as the blue class



every example in the blue shaded area will be classified correctly as the red class

1NN Visualization

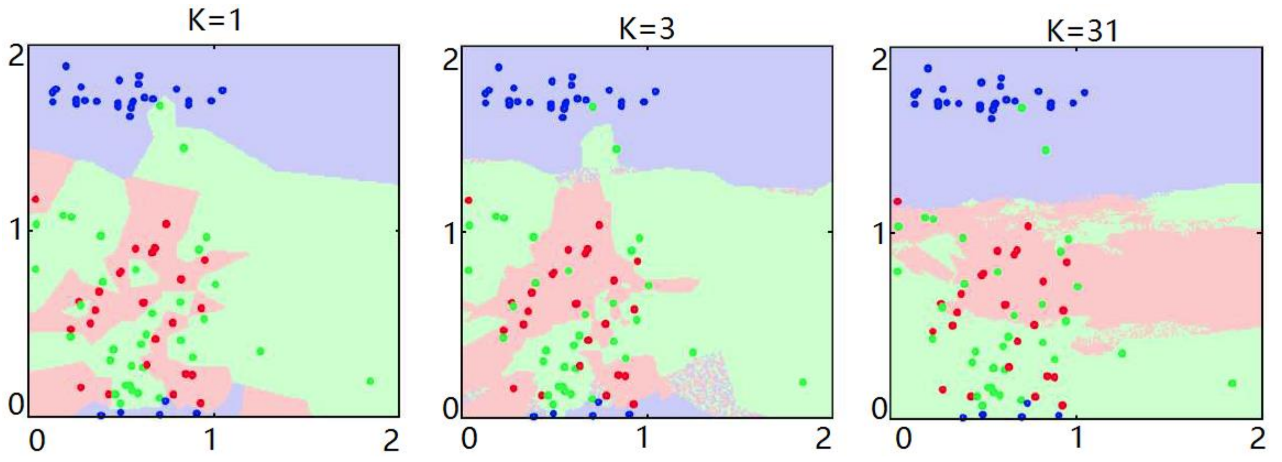
- **Voronoi Diagram** (维诺图) is useful for visualization



A Voronoi diagram divides the space into such cells.

Each cell contains one sample, and every location within the cell is closer to that sample than to any other sample.

Choice of K - Neighborhood Size



Small K

Creates many small regions for each class
sensitive to “noise”
(May lead to non-smooth) decision boundaries and overfiting

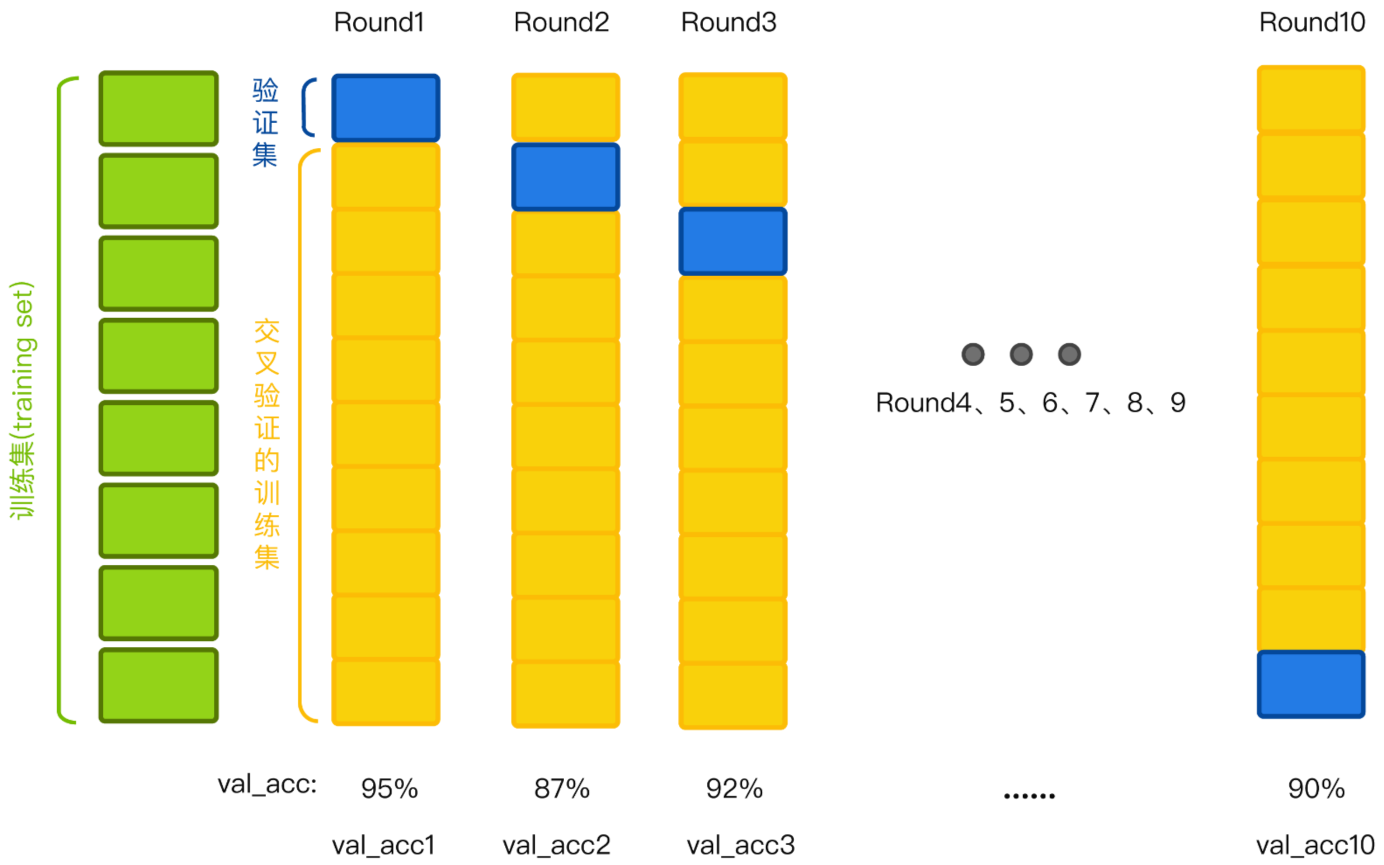
Large K

creates fewer larger regions ,
usually produces smoother decision boundaries ,
can reduce the impact of class label noise.
(caution: too smooth decision boundary can underfiting)

Holdout

- Divide labled data into two sets
    - A training set (80%) and a validation set (20%)
- Predict the class labels for validation set by using the examples in training set
- This is an estimate of the generalization error
- Choose the number of neighbors *k* that maximizes the classification accuracy
- Once we have chosen the model, and report performance on a test set.

K-fold cross-validation



Final Validation Accuracy = mean(val_acc1 + val_acc2 + ... + val_acc10)

Leave-One-Out Method

```
• For $k = 1, 2, \ldots, K$
      $err(k) = 0$
      For $i = 1, 2, \ldots, n$
        * Predict the class label $\widehat{y}_i$ for $\mathbf{x}_i$
          using the remaining data points
        * $err(k) = err(k) + 1$ if $\widehat{y}_i \neq y_i$
• Output $k^* = \arg\min_{1 \leq k \leq K} err(k)$
```

K-NN: How to Choose?

- What distance measure to use?

  Often Euclidean distance is used
  Locally adaptive metrics
  More complicated with non-numeric data, or when different dimensions have different scales

- Choice of *k*?

  odd number
  1-NN often performs well in practice
  Interesting theoretical properties if k < sqrt(n), n is # of examples
  Can choose k through cross-validation and so on

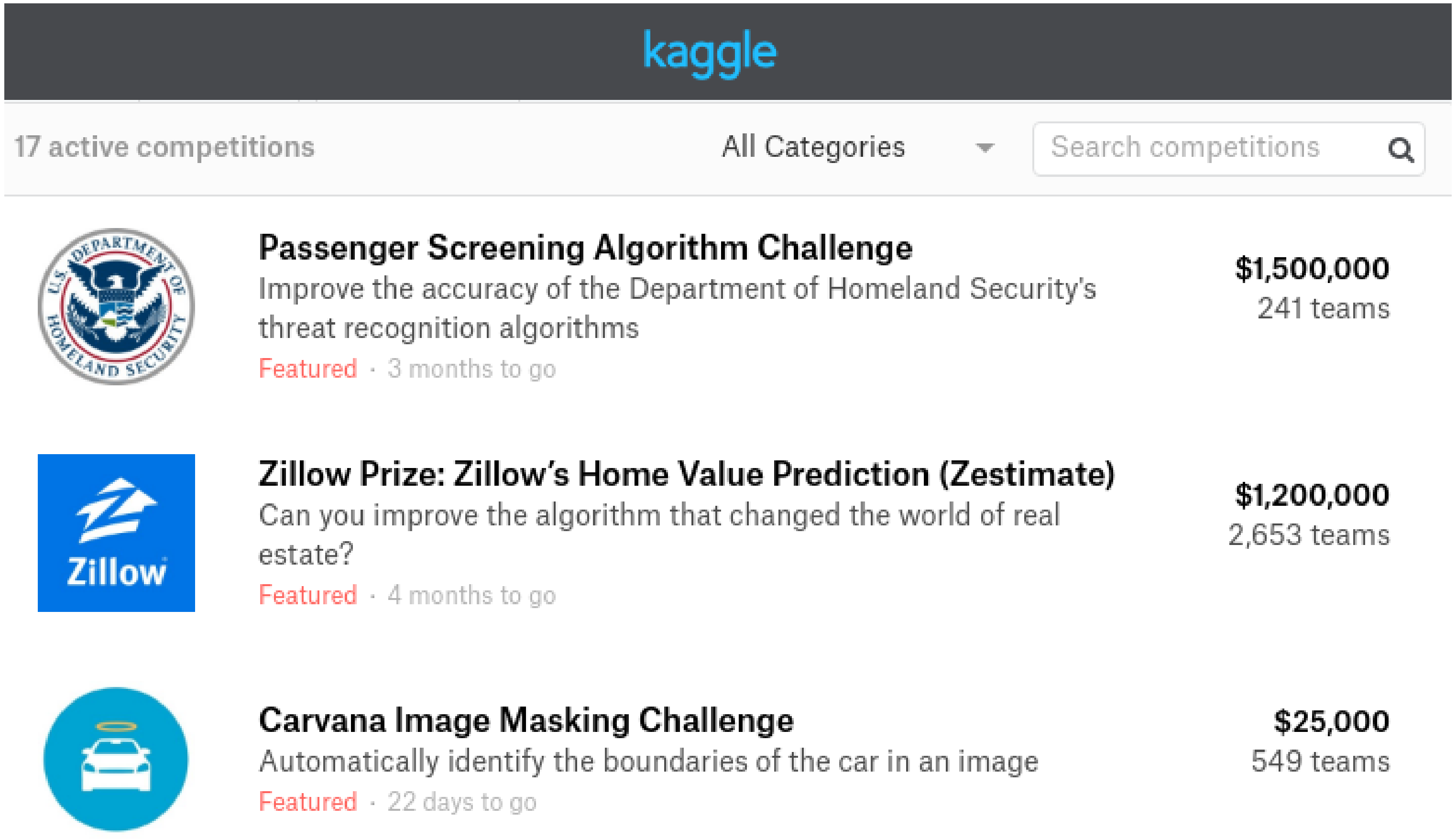
Example: Digit Recognition

kaggle™

Kaggle is a platform for predictive modelling and analytics competitions to produce the best models for predicting and describing the data.

https://www.kaggle.com/

Example: Digit Recognition


kaggle

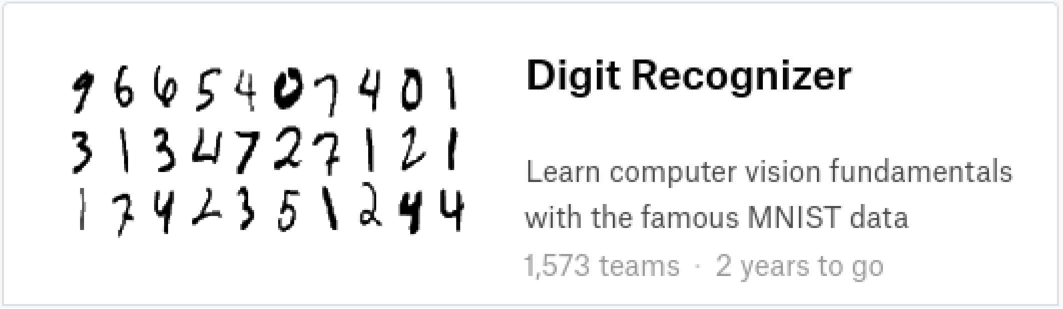
17 active competitions

Passenger Screening Algorithm Challenge
Improve the accuracy of the Department of Homeland Security's threat recognition algorithms
Featured · 3 months to go
$1,500,000
241 teams

Zillow Prize: Zillow's Home Value Prediction (Zestimate)
Can you improve the algorithm that changed the world of real estate?
Featured · 4 months to go
$1,200,000
2,653 teams

Carvana Image Masking Challenge
Automatically identify the boundaries of the car in an image
Featured · 22 days to go
$25,000
549 teams

Example: Digit Recognition


Digit Recognizer

Learn computer vision fundamentals with the famous MNIST data

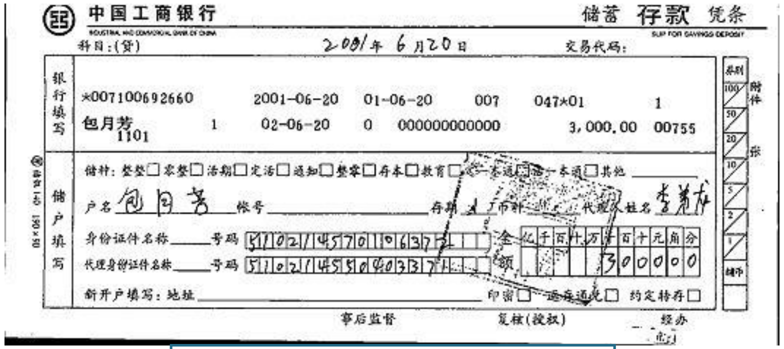
1,573 teams · 2 years to go

Type	Classifier	Preprocessing	Error rate (%)
Convolutional neural network	Committee of 5 CNNs	Expansion of the training data	0.21
Convolutional neural network	Committee of 35 CNNs	Width normalizations	0.23
Deep neural network	6-layer	None	0.35
K-Nearest Neighbors	K-NN with non-linear deformation	Shiftable edges	0.52
Support vector machine	Virtual SVM	Deskewing	0.56
Non-linear classifier	PCA + quadratic classifier	None	3.3
Linear classifier	Pairwise linear classifier	Deskewing	7.6

Nearest neighbour is still fairly competitive!

Example: Digit Recognition

Number Plate Recognition



中国工商银行 INDUSTRIAL AND COMMERCIAL BANK OF CHINA

储蓄 存款 凭条 SLIP FOR SAVINGS DEPOSIT

科目:(贷) 2001年 6月20日 交易代码:

银行填写: *007100692660 2001-06-20 01-06-20 007 047*01 1

包月芳 1101 1 02-06-20 0 000000000000 3,000.00 00755

储户填写: 储种: 整整☐ 零整☐ 活期☐ 定活☐ 通知☐ 整零☐ 存本☐ 教育☐ 一本通☐ 一本通☐ 其他

户名 包月芳 帐号 存期 币种 代理人姓名 [illegible]

身份证件名称 号码 51102145701063[illegible]

代理身份证件名称 号码 51102145504033[illegible]

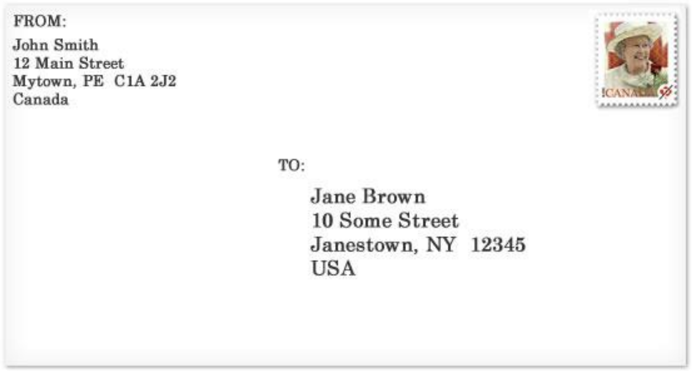
金额 亿 千 百 十 万 千 百 十 元 角 分 3 0 0 0 0 0

新开户填写: 地址 印密☐ 通存通兑☐ 约定转存☐

事后监督 复核(授权) 经办

Recognition of legal amounts on bank cheques

House no. Recognition



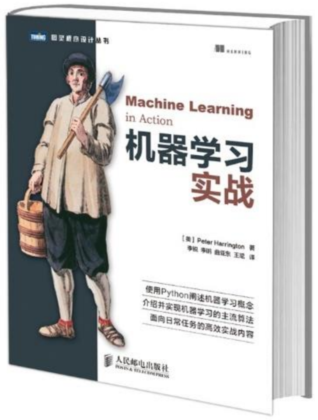
FROM:
John Smith
12 Main Street
Mytown, PE C1A 2J2
Canada

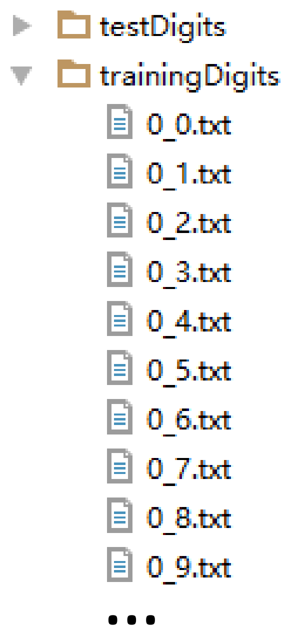
TO:

Jane Brown
10 Some Street
Janestown, NY 12345
USA

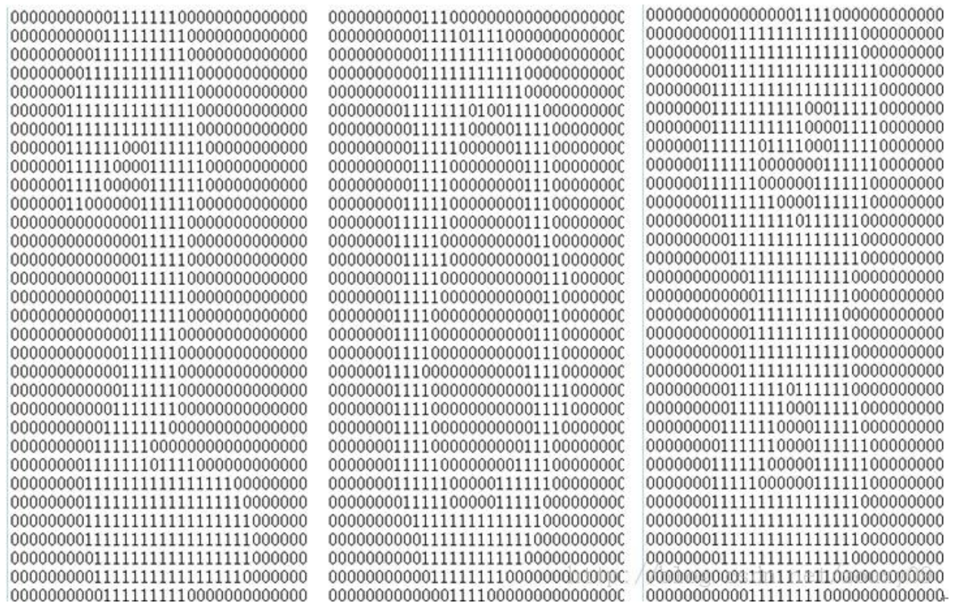
Postal mail sorting

Data Set


Machine Learning
in Action
机器学习实战
[美] Peter Harrington 著
使用Python阐述机器学习概念
介绍并实现机器学习的主流算法
面向日常任务的高效实战内容
人民邮电出版社



testDigits
trainingDigits
0_0.txt
0_1.txt
0_2.txt
0_3.txt
0_4.txt
0_5.txt
0_6.txt
0_7.txt
0_8.txt
0_9.txt
...



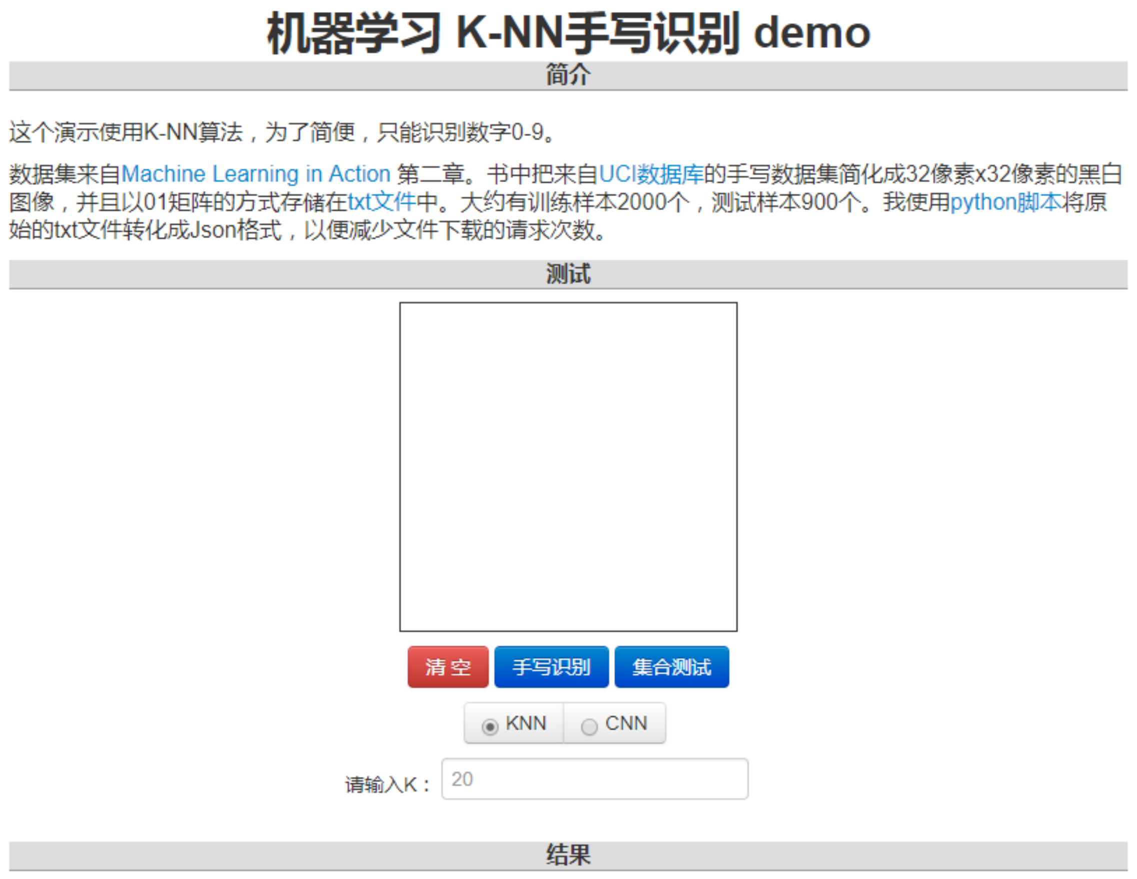
```
00000000000111111100000000000000
00000000001111111110000000000000
00000000011111111110000000000000
00000000111111111111000000000000
00000001111111111111000000000000
00000011111111111111000000000000
00000011111111111111000000000000
00000011111100011111100000000000
00000011111000011111100000000000
00000011110000011111100000000000
00000011000000111111000000000000
00000000000000111110000000000000
00000000000000111110000000000000
00000000000000111110000000000000
00000000000001111110000000000000
00000000000001111110000000000000
00000000000001111110000000000000
00000000000001111100000000000000
00000000000011111100000000000000
00000000000011111100000000000000
00000000000011111100000000000000
00000000000111111100000000000000
00000000001111111000000000000000
00000000011111100000000000000000
00000000111111101111000000000000
00000000111111111111111100000000
00000000111111111111111110000000
00000000111111111111111111000000
00000000111111111111111111000000
00000000011111111111111111000000
00000000011111111111111110000000
00000000001111111110000000000000
```

```
00000000001110000000000000000000
00000000001111011110000000000000
00000000001111111111000000000000
00000000001111111111100000000000
00000000011111111111100000000000
00000000111111101001111000000000
00000000011111100000111100000000
00000000011111000000111100000000
00000000011110000000011100000000
00000000011110000000011100000000
00000000111110000000011100000000
00000001111110000000011100000000
00000001111100000000001100000000
00000000111110000000000110000000
00000000111100000000000111000000
00000001111100000000001100000000
00000001111000000000001100000000
00000001111000000000001110000000
00000001111000000000001110000000
00000011110000000000011110000000
00000001111000000000001111000000
00000001111000000000001111000000
00000001111000000000001110000000
00000001111000000000011100000000
00000001111100000000111100000000
00000001111110000011111100000000
00000000111110000011111000000000
00000000011111111111111000000000
00000000001111111111110000000000
00000000001111111111100000000000
00000000000111111110000000000000
00000000000001111000000000000000
```

```
00000000000000001111000000000000
00000000011111111111111000000000
00000000111111111111111000000000
00000000111111111111111110000000
00000001111111111111111110000000
00000001111111111000111110000000
00000001111111111000011110000000
00000011111101111000111110000000
00000011111100000001111110000000
00000011111100000011111100000000
00000001111111000011111100000000
00000000111111110111111000000000
00000000011111111111111000000000
00000000011111111111111000000000
00000000000111111111110000000000
00000000000011111111110000000000
00000000000111111111100000000000
00000000000111111111110000000000
00000000001111111111110000000000
00000000001111111111110000000000
00000000011111101111110000000000
00000000011111100011111000000000
00000000111111000011111000000000
00000000111111000011111100000000
00000001111110000011111100000000
00000001111100000011111100000000
00000001111111111111111000000000
00000001111111111111111000000000
00000001111111111111111000000000
00000000111111111111110000000000
00000000011111111111000000000000
00000000000111111110000000000000
```

32x32 pixel images: *d* = 1024
1,934 training samples
946 test samples

Example: Digit Recognition



机器学习 K-NN手写识别 demo

简介

这个演示使用K-NN算法，为了简便，只能识别数字0-9。

数据集来自Machine Learning in Action 第二章。书中把来自UCI数据库的手写数据集简化成32像素x32像素的黑白图像，并且以01矩阵的方式存储在txt文件中。大约有训练样本2000个，测试样本900个。我使用python脚本将原始的txt文件转化成Json格式，以便减少文件下载的请求次数。

测试

[✓ KNN] CNN

请输入K：20

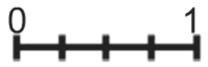
结果

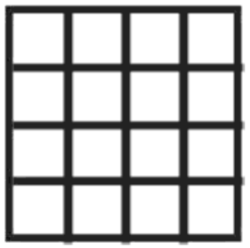
http://211.87.235.83/other/HandwritingRecognition/

Curse of Dimensionality

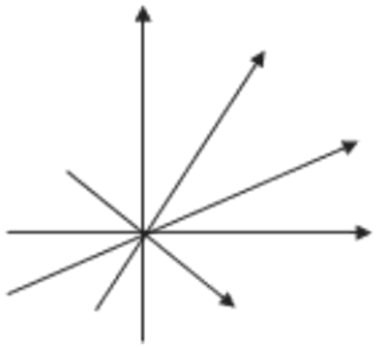
- K-NN breaks down in high-dimensional space, “Neighborhood” becomes very large and easily misleading.
- The curse of dimensionality refers to various phenomena that arise in high-dimensional spaces that do not occur in low-dimensional settings of everyday experience.
  - Storage complexity
  - Computational complexity
  - Sampling
  - Combinatorics
  - Nearest neighbor search
  - Distance functions
  - Nonparametric estimation
  - …

sampling





……



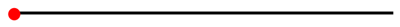
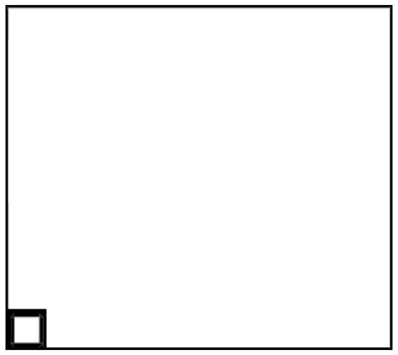
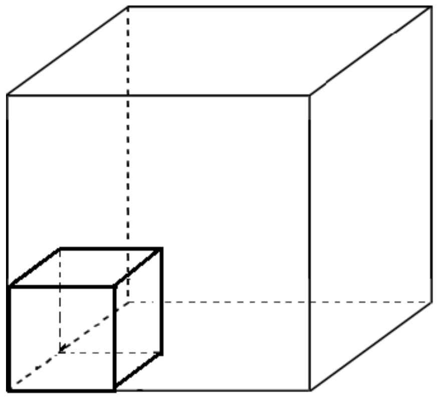
1维：5

2维：5*5=25

10维：5^10= 976 5625

Nearest neighbor search

- Assume 5000 points uniformly distributed in the unit hypercube and we want to apply 5-nn. (Suppose our query point is at the origin)
  - In 1-dimension, we must go a distance of 5/5000 = 0.001 on the average to capture 5 nearest neighbors
  - In 2 dimensions, we must go $\sqrt{0.001}$ to get a square that contains 0.001 of the volume.
  - In d dimensions, we must go $\mathbf{(0.001)^{1/d}}$

维数灾难的几个表现

- 噪声影响
  - 特征空间：101维
  - 正负样本在第一维的距离：1
  - 样本在其余维的噪声：10%
  - “噪声距离”：$\sqrt{100 \times 0.1^2} = 1$
  - □即使噪声只有10%，高维空间的“噪声距离”足以掩盖正负样本的本质区别

http://www.cnblogs.com/zhangchaoyang/articles/2801525.html
http://blog.csdn.net/zc02051126/article/details/49618633
https://zh.wikipedia.org/wiki/%E7%BB%B4%E6%95%B0%E7%81%BE%E9%9A%BE
http://blog.csdn.net/zc02051126/article/details/49618633

K-Nearest Neighbor: Properties

- Advantages:
  - Simple and intuitive, Training is very fast, easily implementable
  - Particularly suitable for multi-classification problems
  - With infinite training data and large enough K, K-NN approaches work well!
- Disadvantages:
  - Sensitive to noisy features
  - Store all the training data in memory even at test time
  - Slow at query time: O(nd) computations for each test point
  - In high dimensions, distance notions can be counter-intuitive!
  - May perform badly in high dimensions (curse of dimensionality)

Also called:

- Memory/Instance-based learning
- Lazy learning

Reducing Complexity

- Various exact and approximate methods for reducing complexity
- Computational complexity

  use smart data structures, like k-d trees

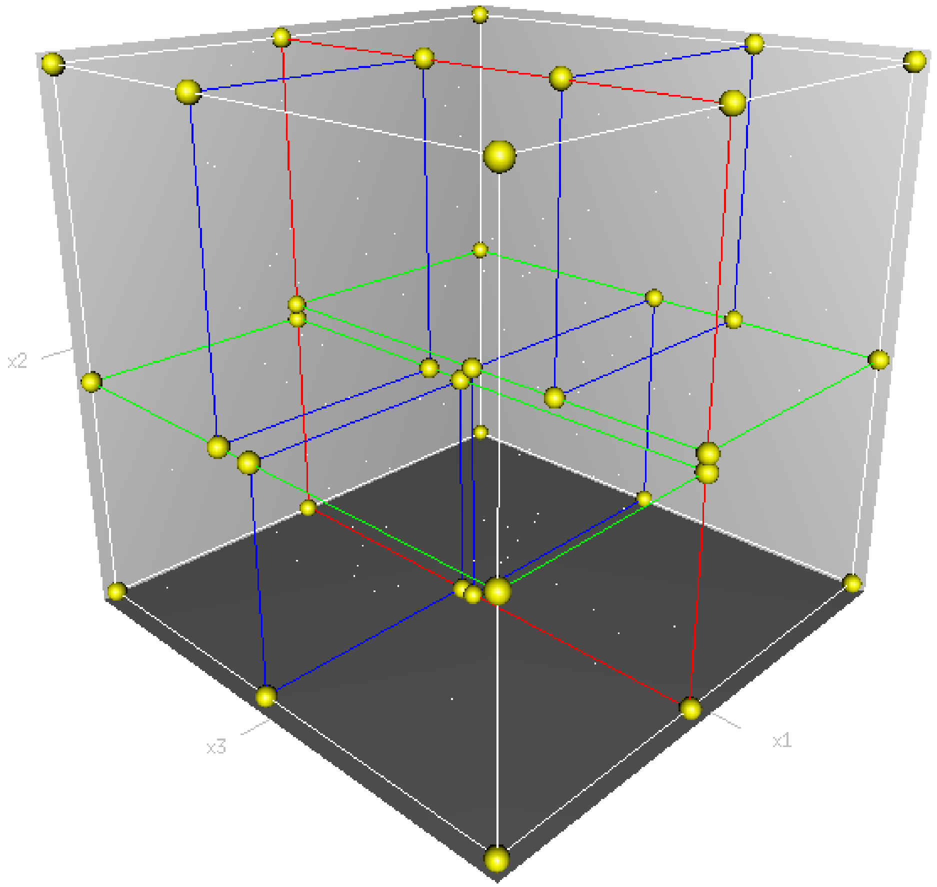
  ANN 、BBF算法、LSH(局部敏感哈希)、Randomized K-d trees

  球树、M树、VP树、MVP树

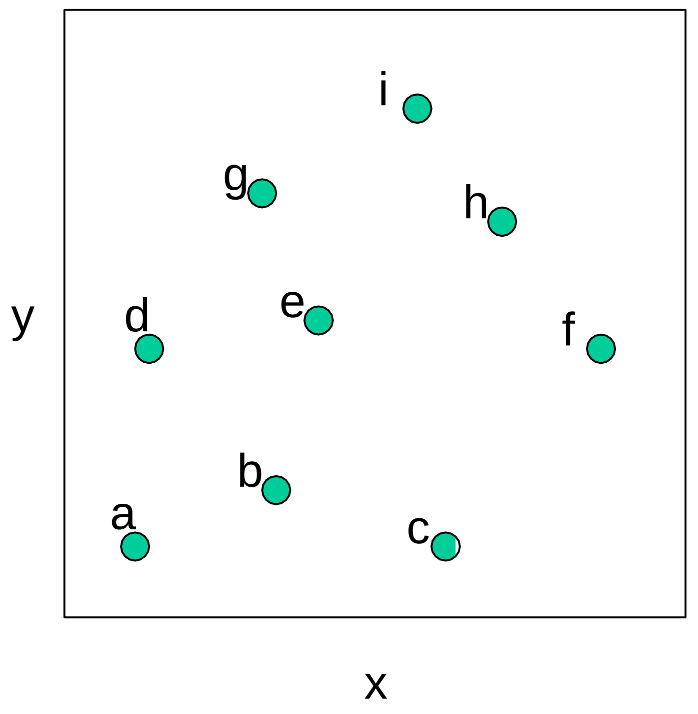
K-d Trees

- Invented in 1970s by Jon Bentley
- Name originally meant “3-d trees, 4-d trees, etc” where k was the # of dimensions
- Idea: Each level of the tree compares against 1 dimension.
- Tree used to store spatial data.
    - Nearest neighbor search.
    - Range queries.
    - Fast look-up!
- k-d trees are guaranteed $\log_2 n$ depth where n is the number of points in the set.
    - Traditionally, k-d trees store points in d-dimensional space

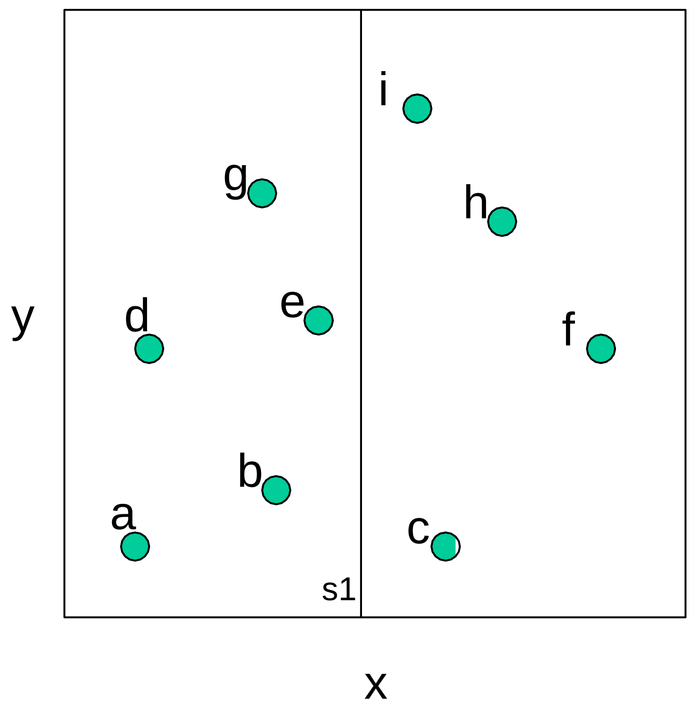
3D k-d tree

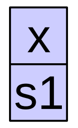


k-d tree construction(1)

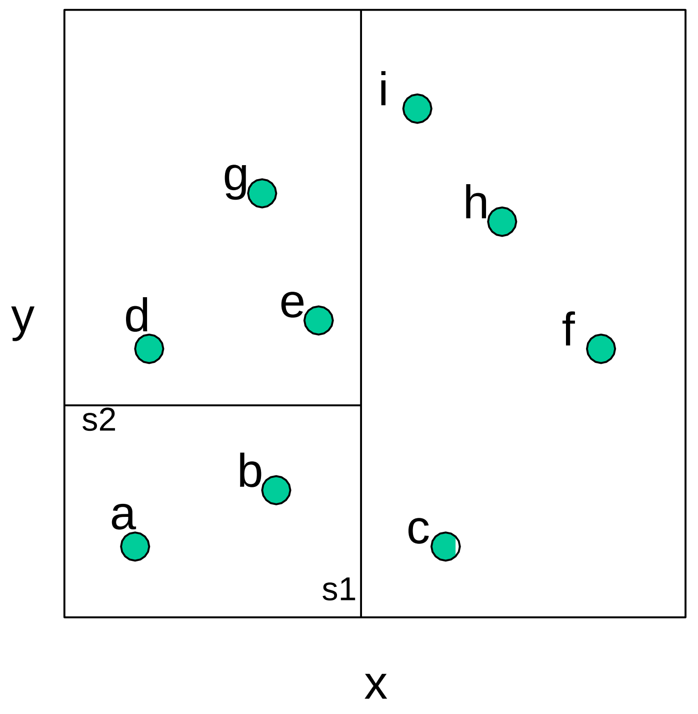


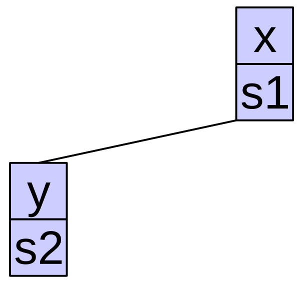
k-d tree construction(2)



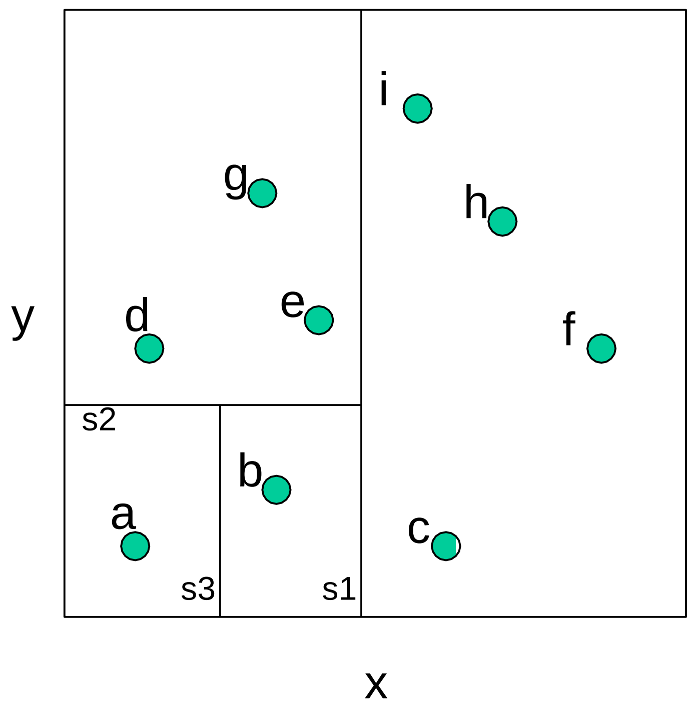


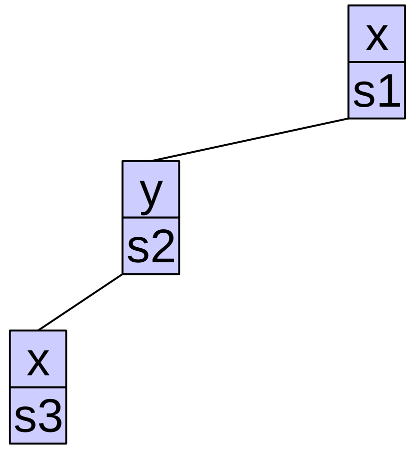
k-d tree construction(3)



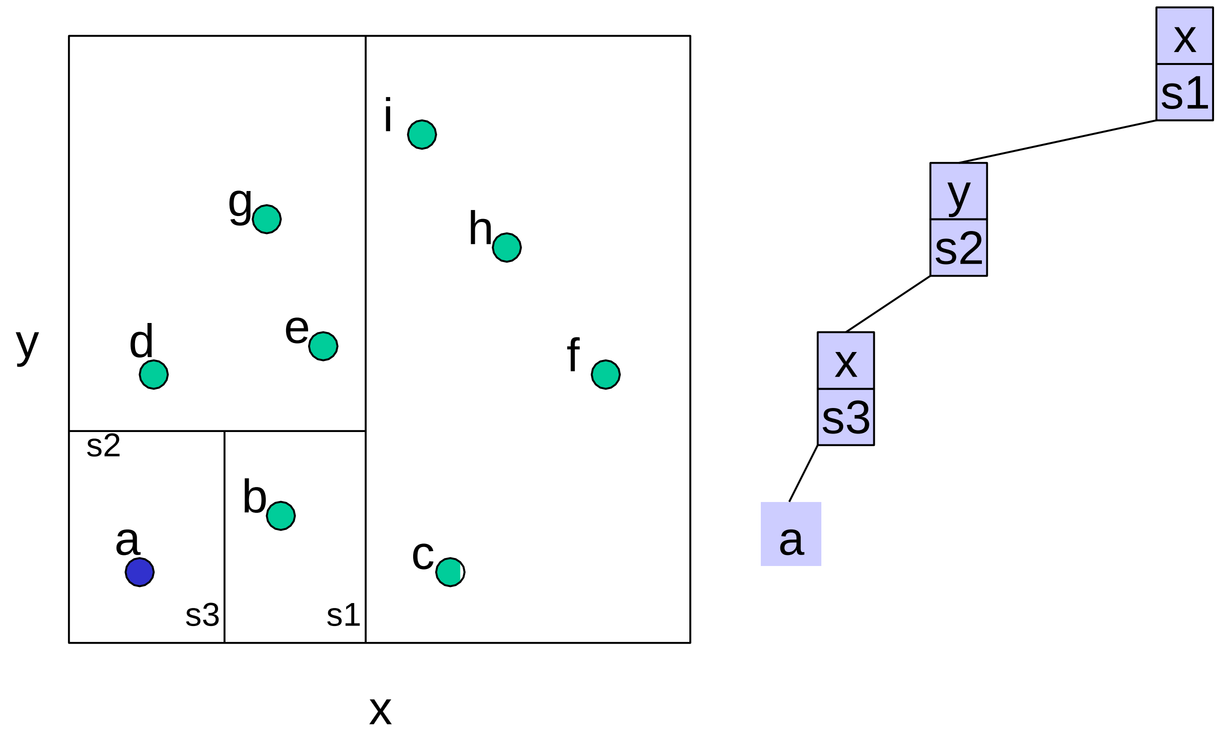


k-d tree construction(4)

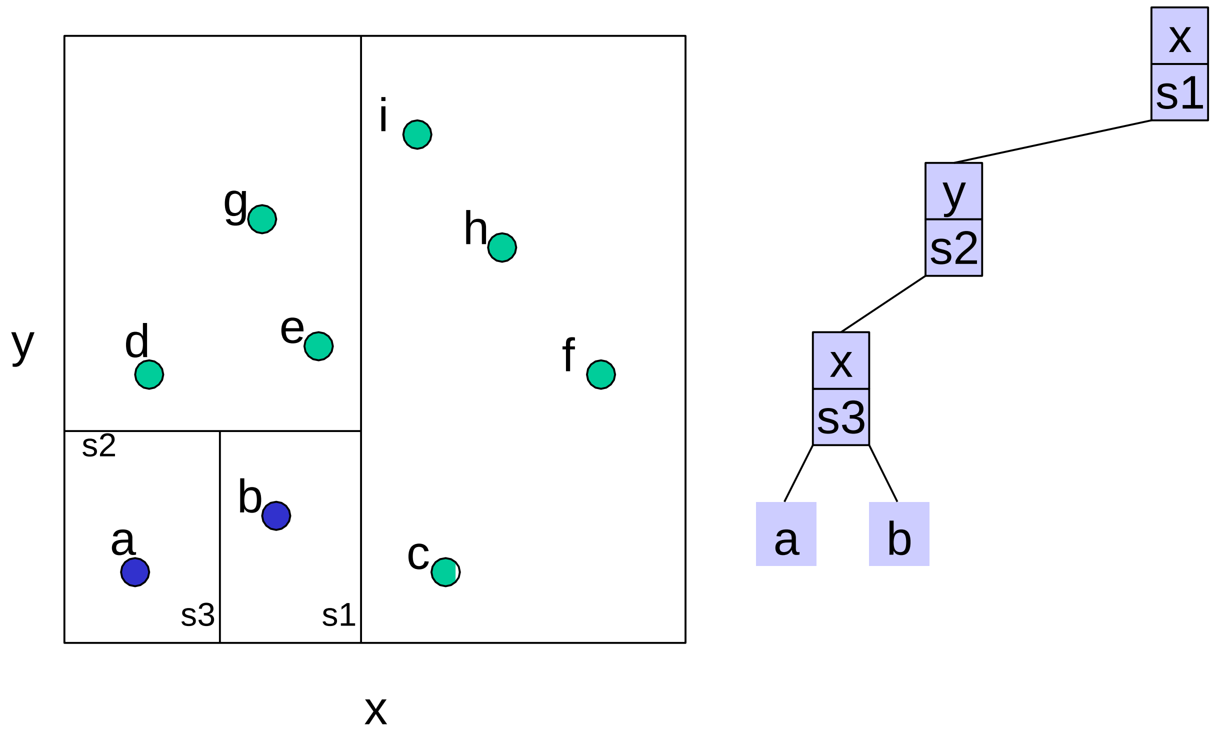




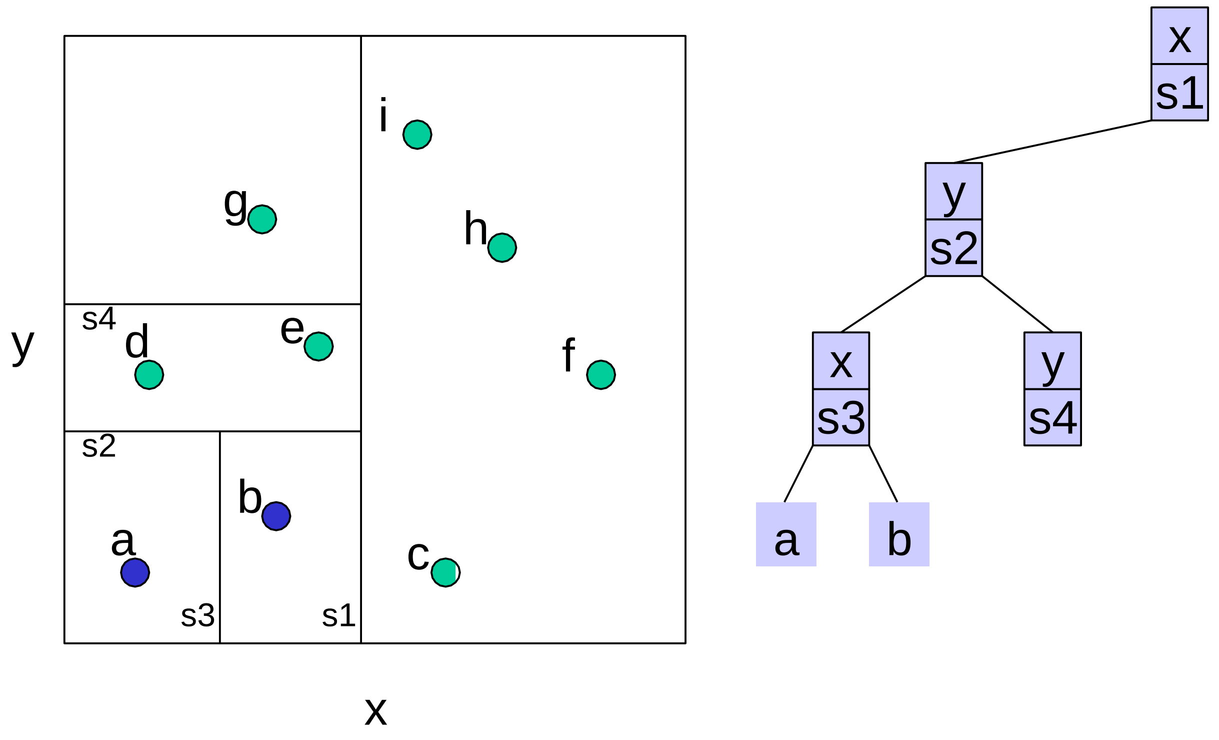
k-d tree construction(5)



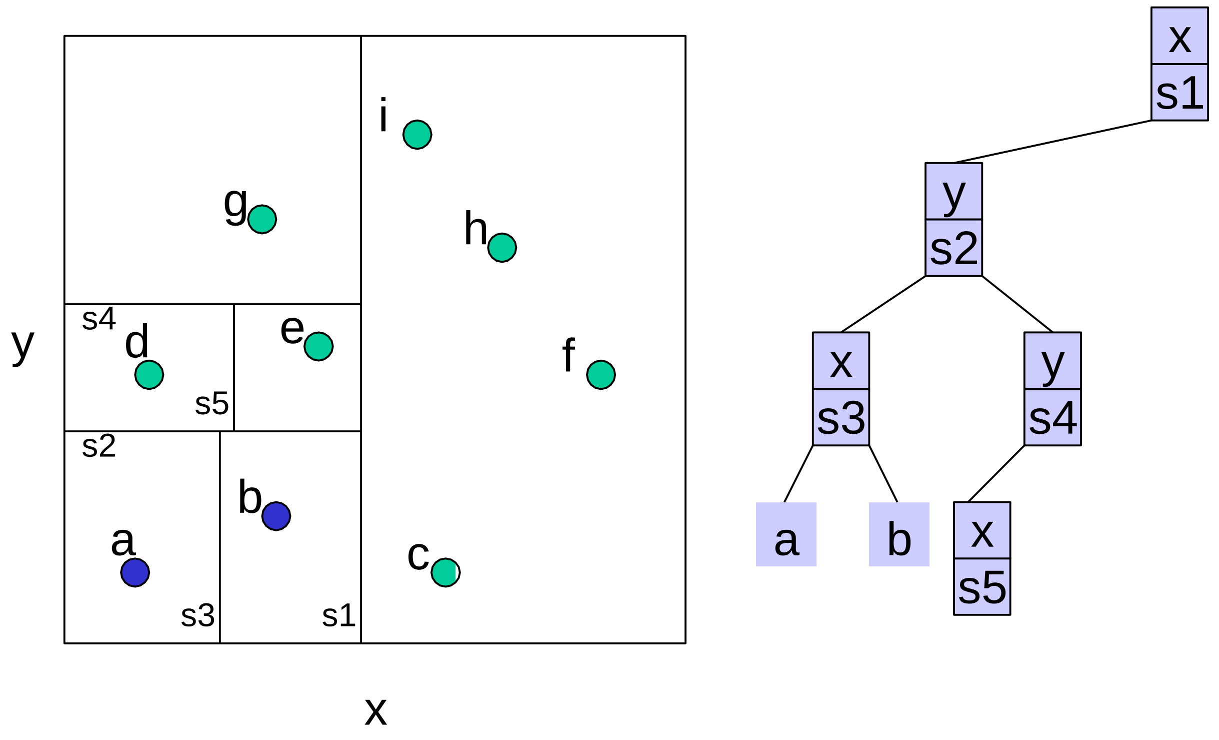
k-d tree construction(6)



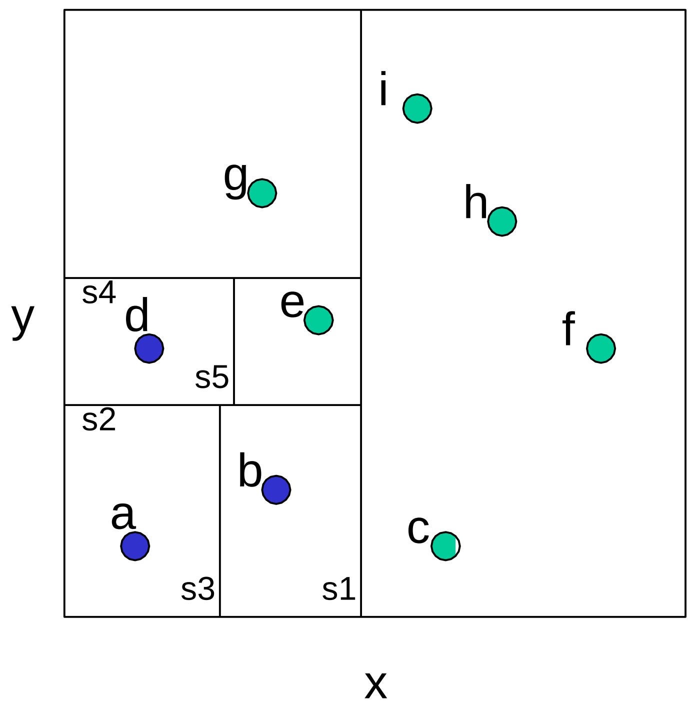
k-d tree construction(7)

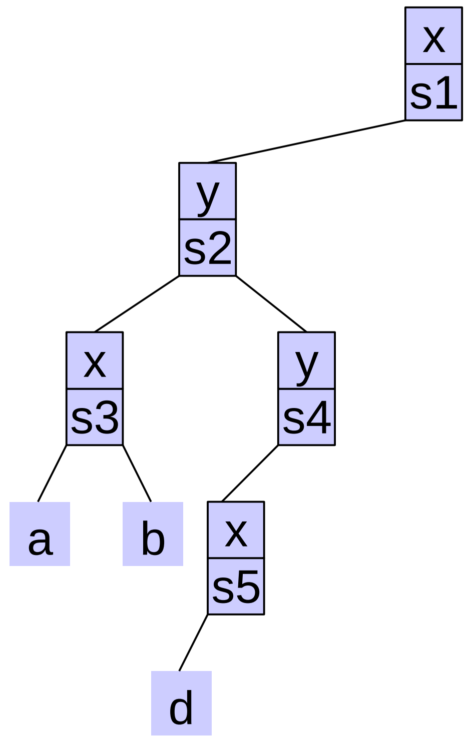


k-d tree construction(8)

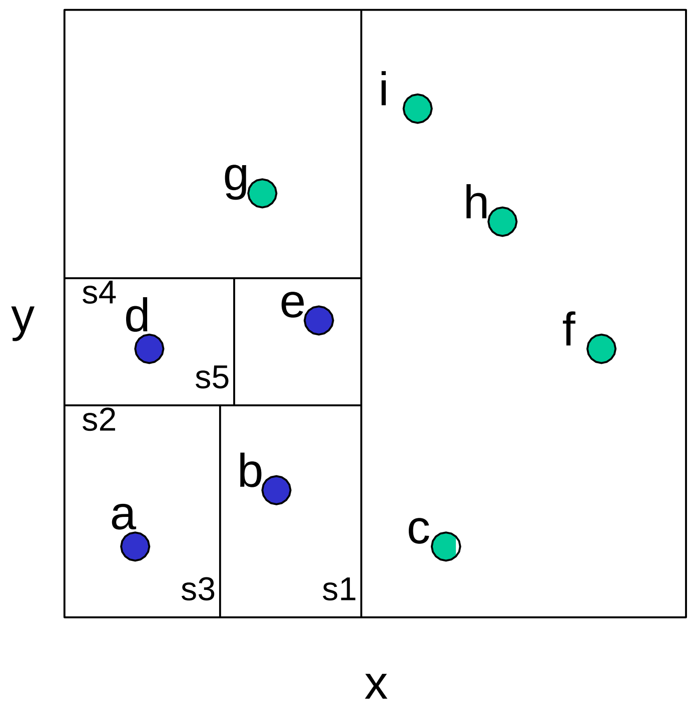


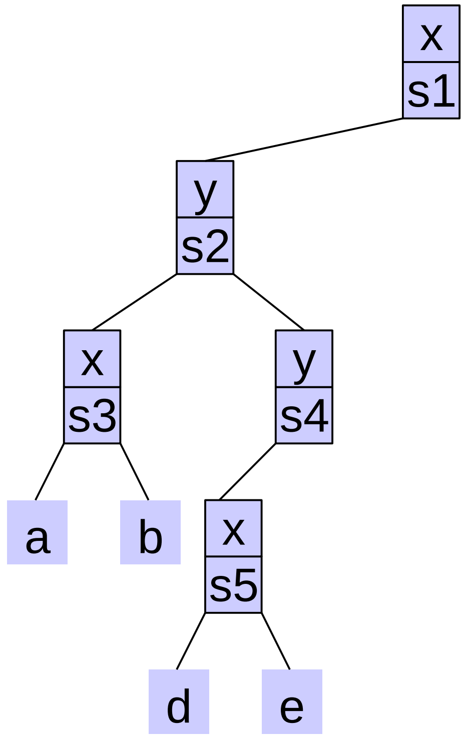
k-d tree construction(9)



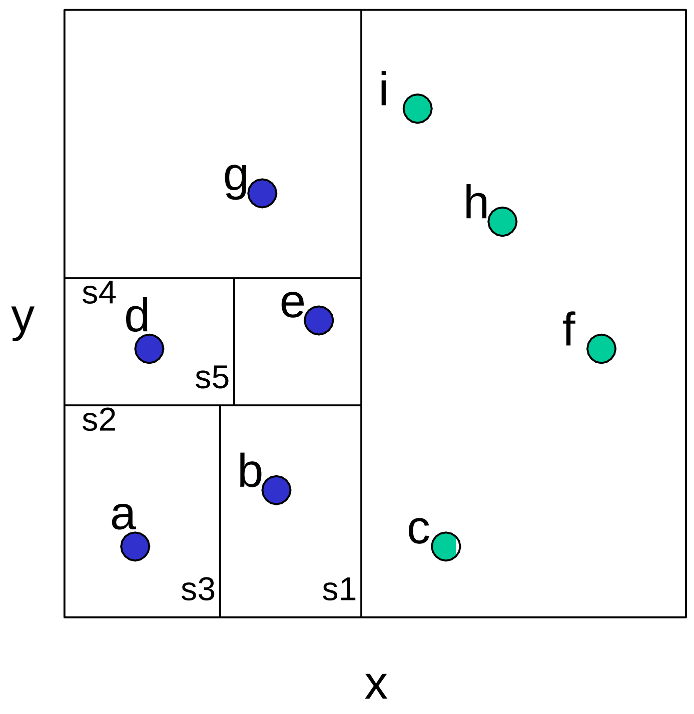


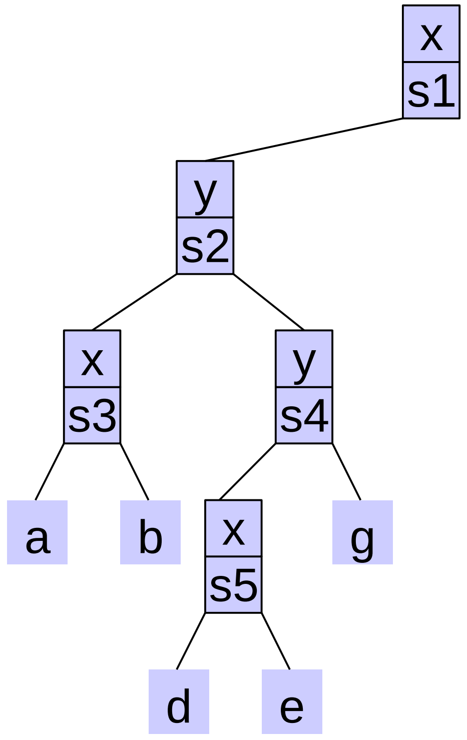
k-d tree construction(10)



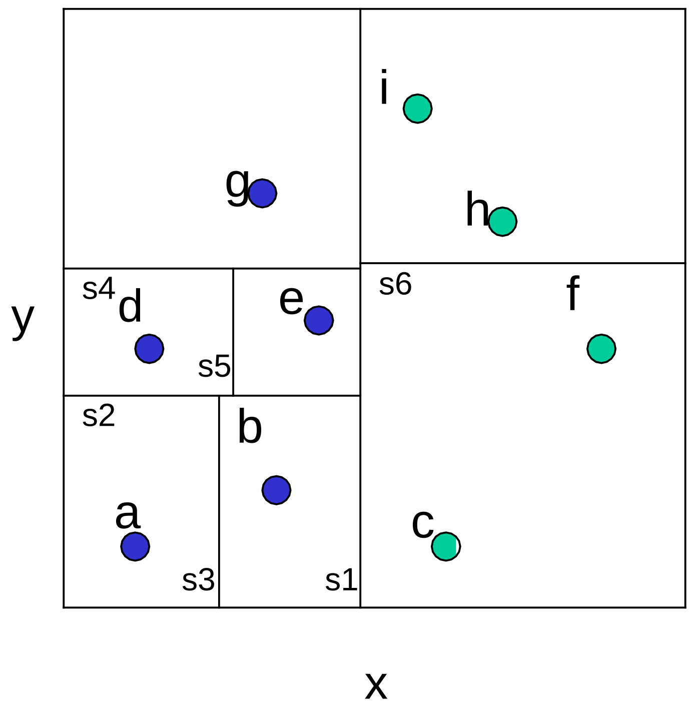


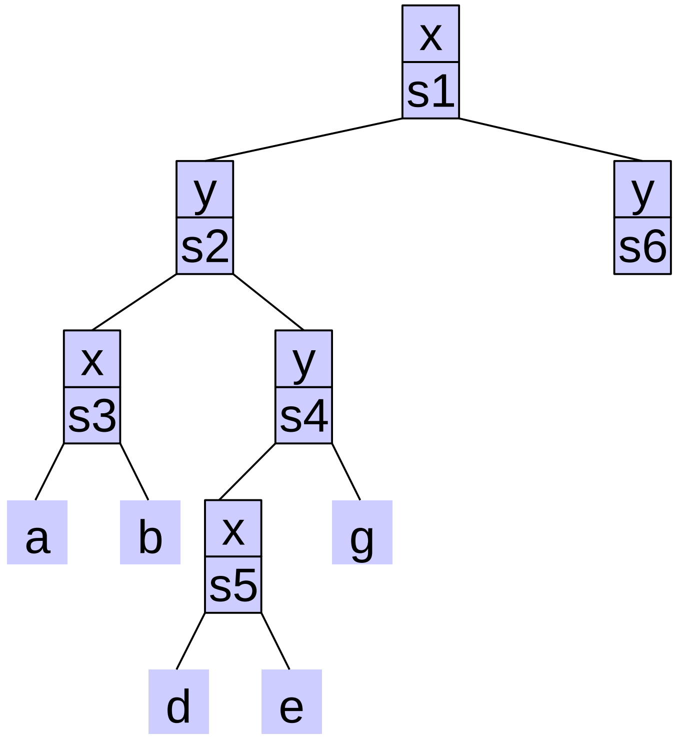
k-d tree construction(11)



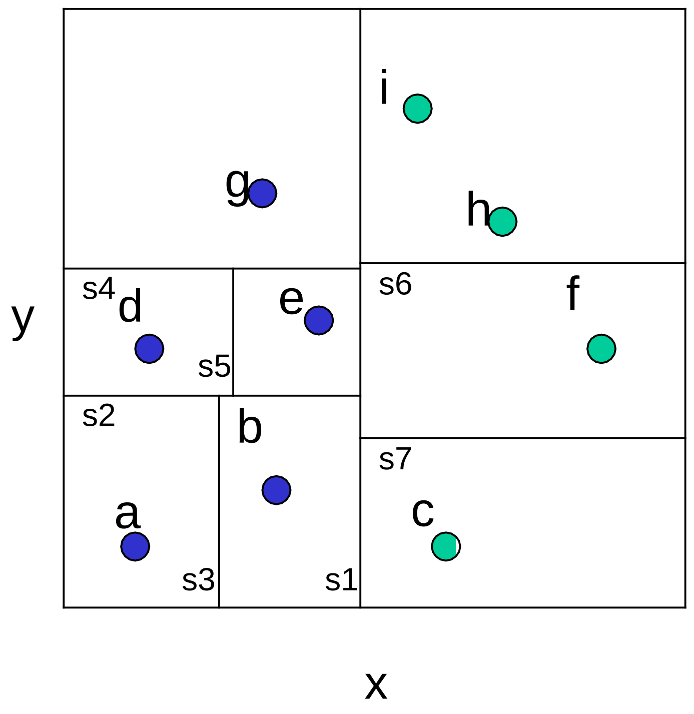


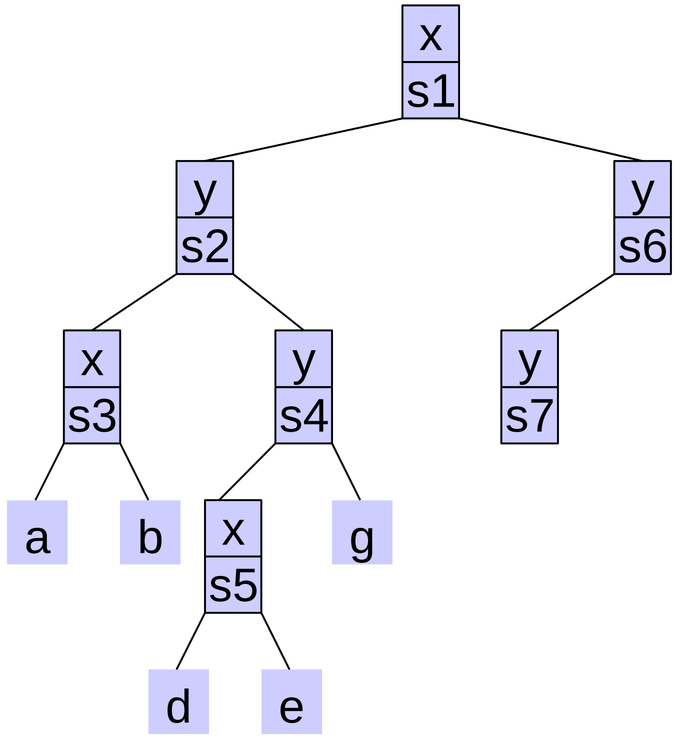
k-d tree construction(12)



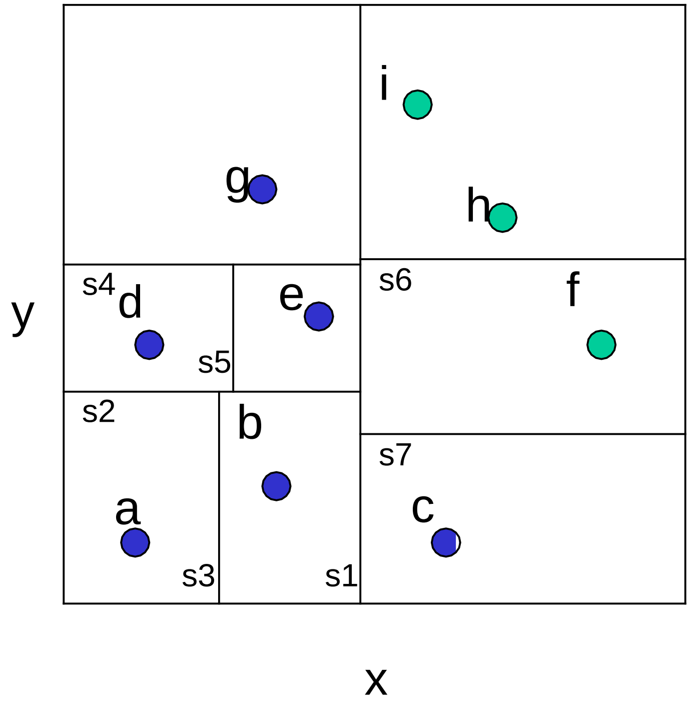


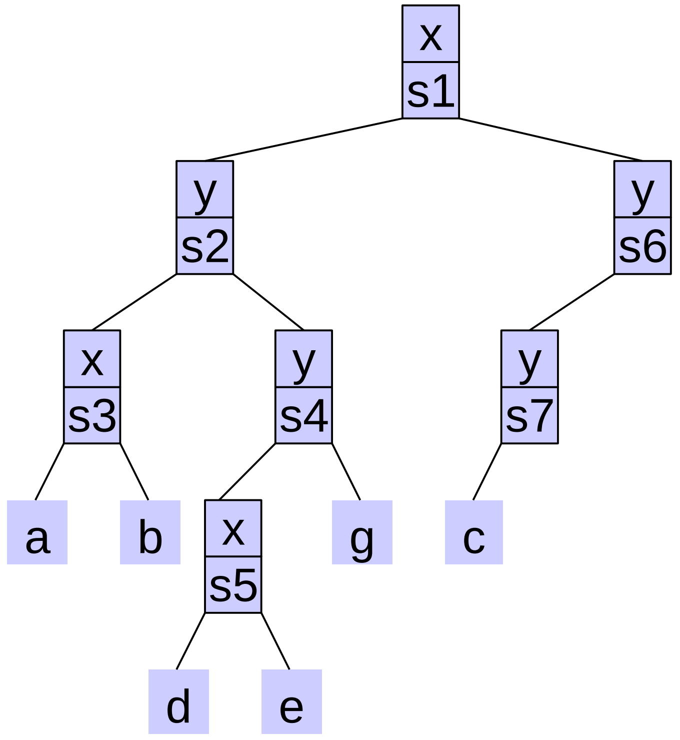
k-d tree construction(13)



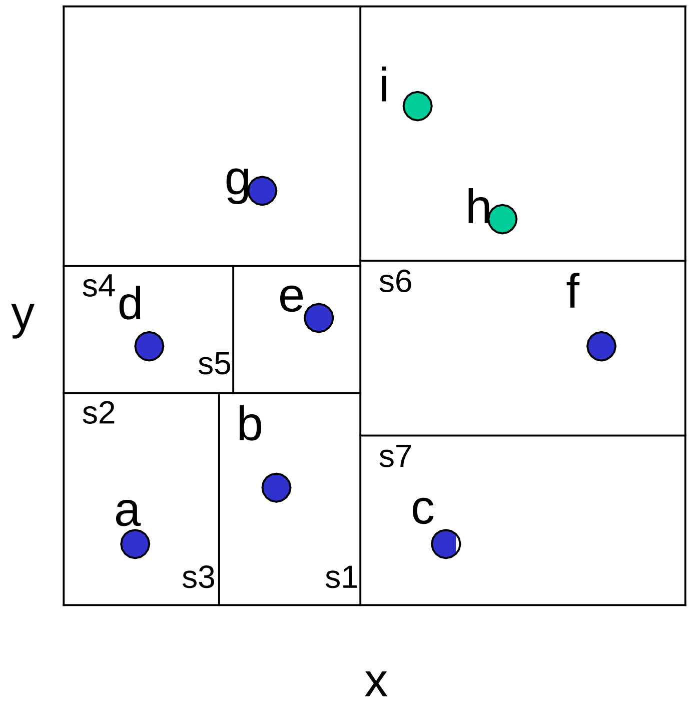


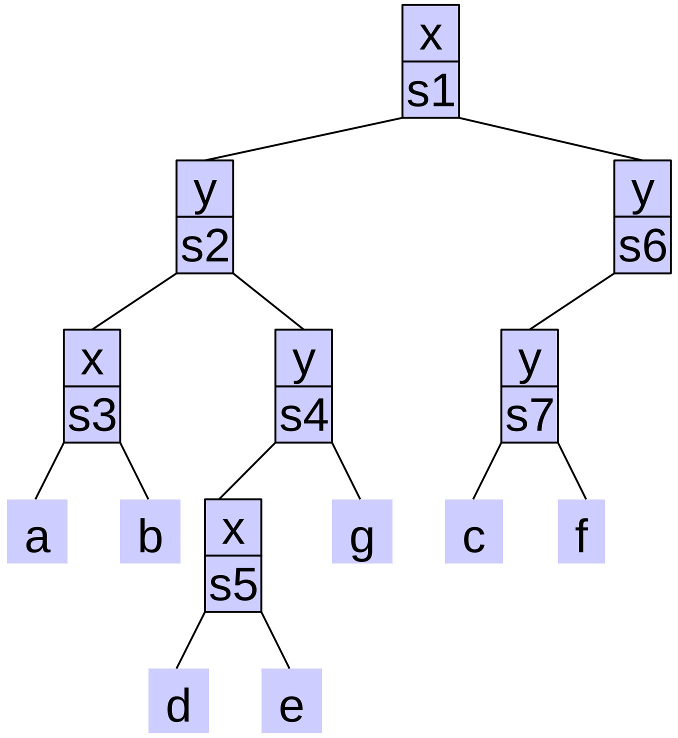
k-d tree construction(14)



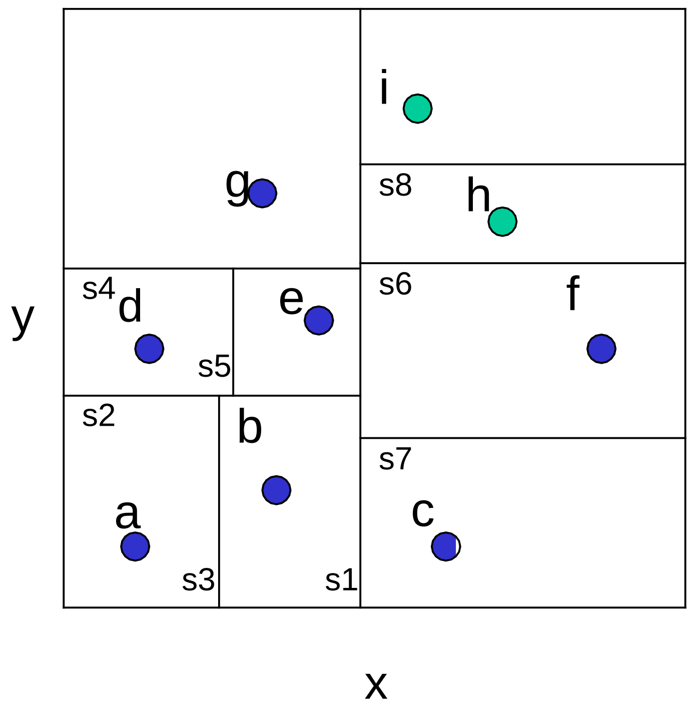


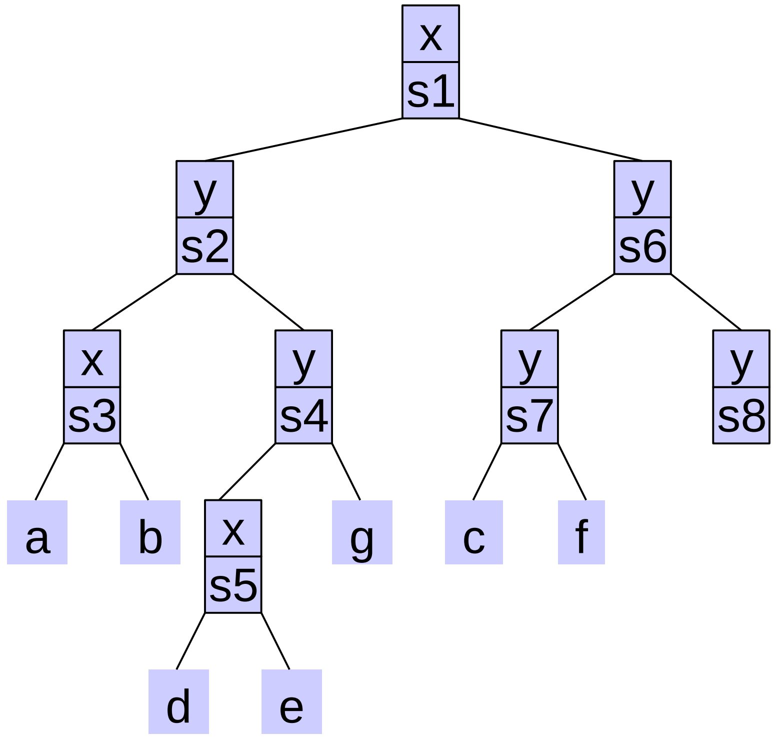
k-d tree construction(15)



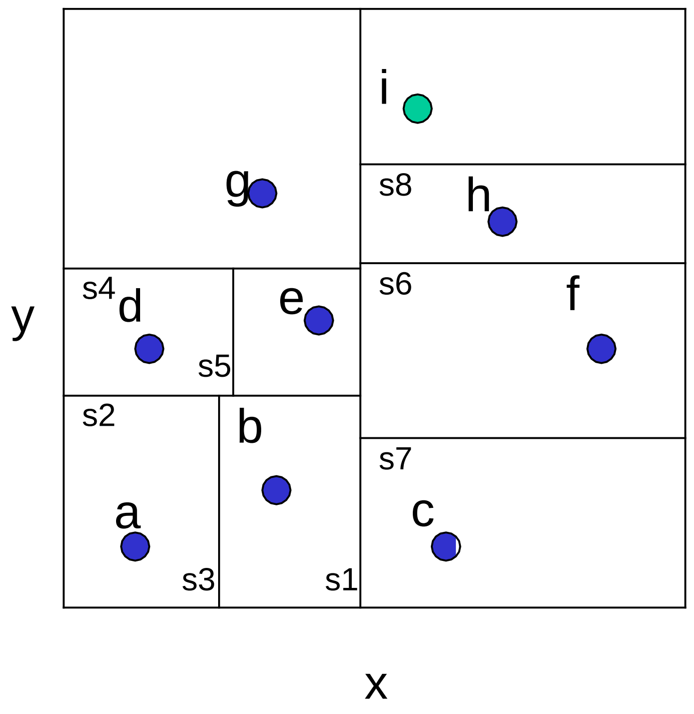


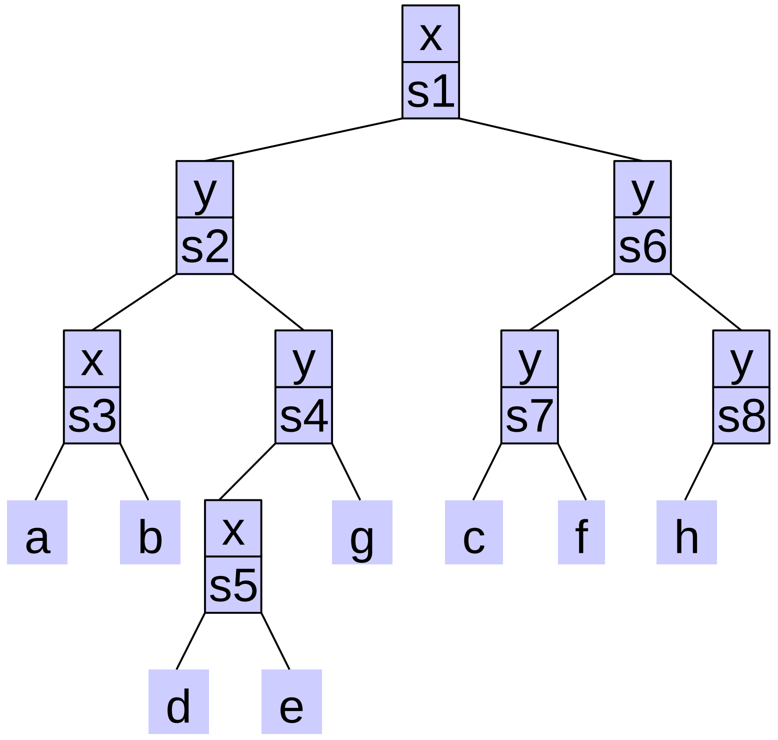
k-d tree construction(16)



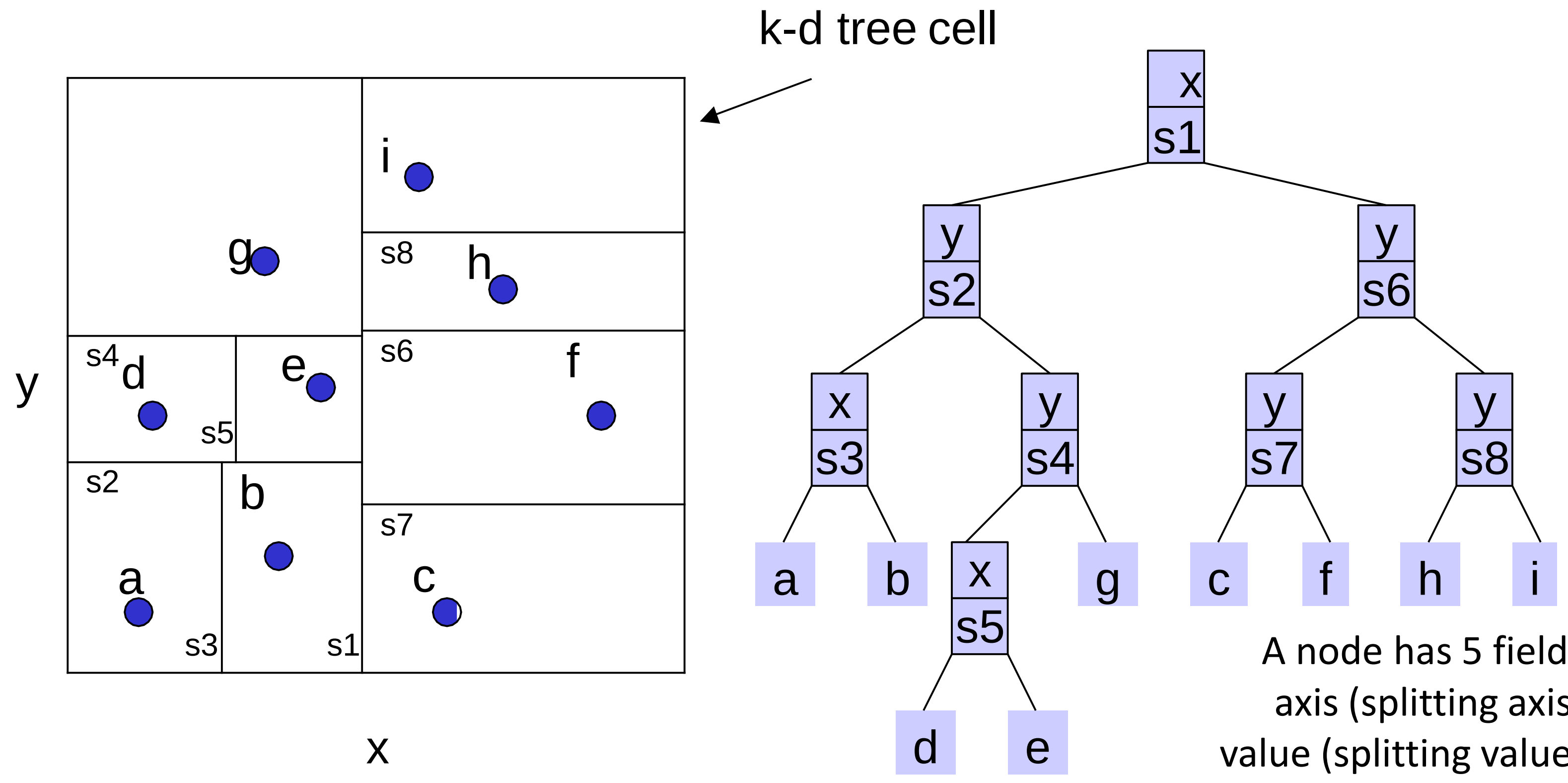


k-d tree construction(17)





k-d tree construction(18)



A node has 5 fields
axis (splitting axis)
value (splitting value)
left (left subtree)
right (right subtree)
point (holds a point if left and right children are null)

Construction strategy

- The construction algorithm is similar as in 2-d
- At the root we split the set of points into two subsets of same size by a hyperplane vertical to x_1-axis
- At the children of the root, the partition is based on the second coordinate: x_2-coordinate
- At depth d, we start all over again by partitioning on the first coordinate
- The recursion stops until there is only one point left, which is stored as a leaf

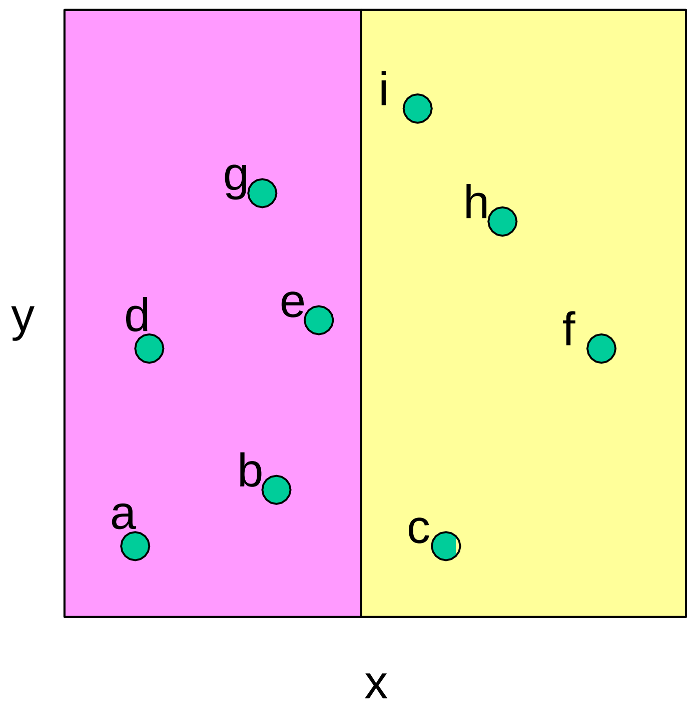
Q1： Which dimension is used to split the set ?

axis with widest spread, biggest variance, alternating each one

Q2： The split point of the dimension

median, middle point of interval

k-d Tree Splitting



sorted points in each dimension

	1	2	3	4	5	6	7	8	9
x	a	d	g	b	e	i	c	h	f
y	a	c	b	d	f	e	h	g	i

indicator for each set

a	b	c	d	e	f	g	h	i
0	0	1	0	0	1	0	1	1

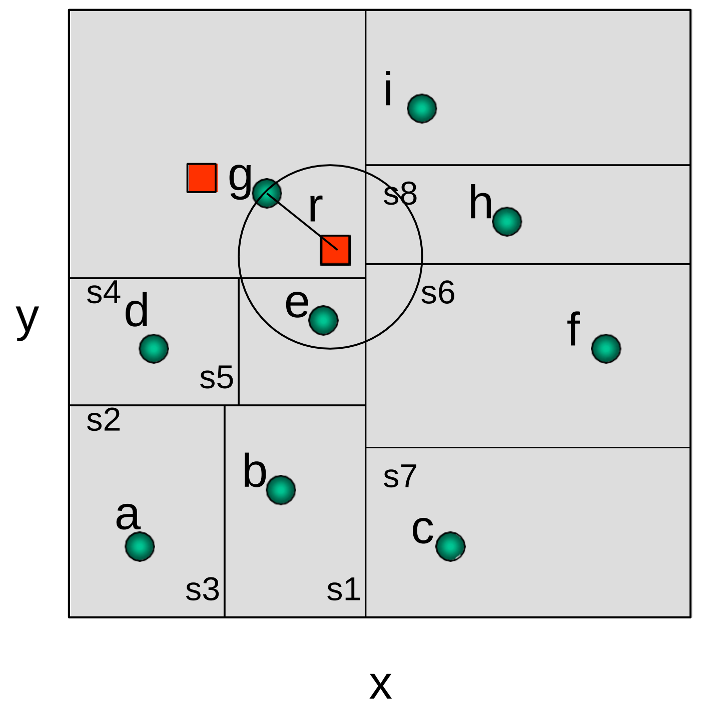
scan sorted points in y dimension and add to correct set

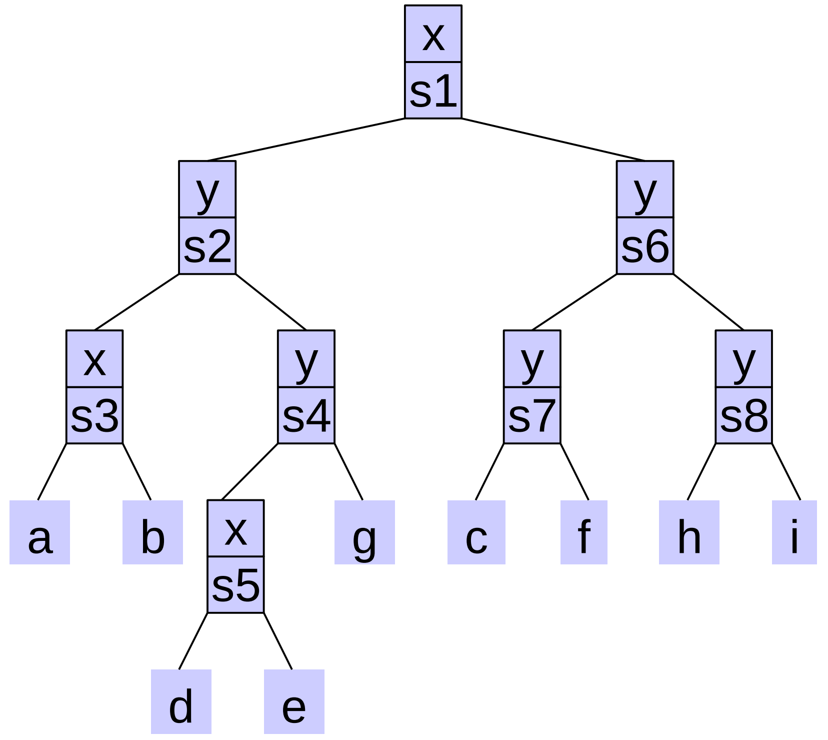
y	a	b	d	e	g	c	f	h	i

Constructing the k-d tree can be done in O(d nlogn) and O(dn) storage

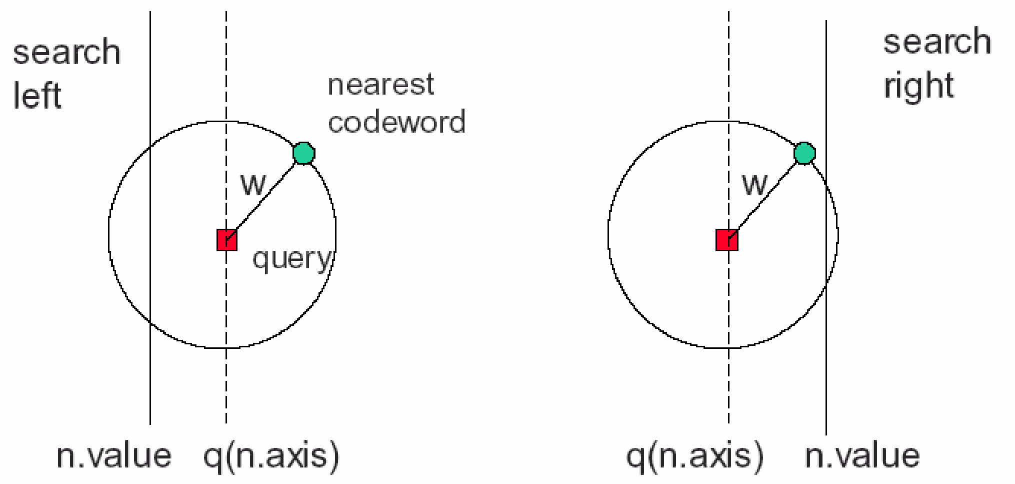
k-d Tree Nearest Neighbor Search

🟥 query point





Why does k-d tree work?



$q(n.axis) - w \leq n.value$
means the circle overlaps
the left subtree.

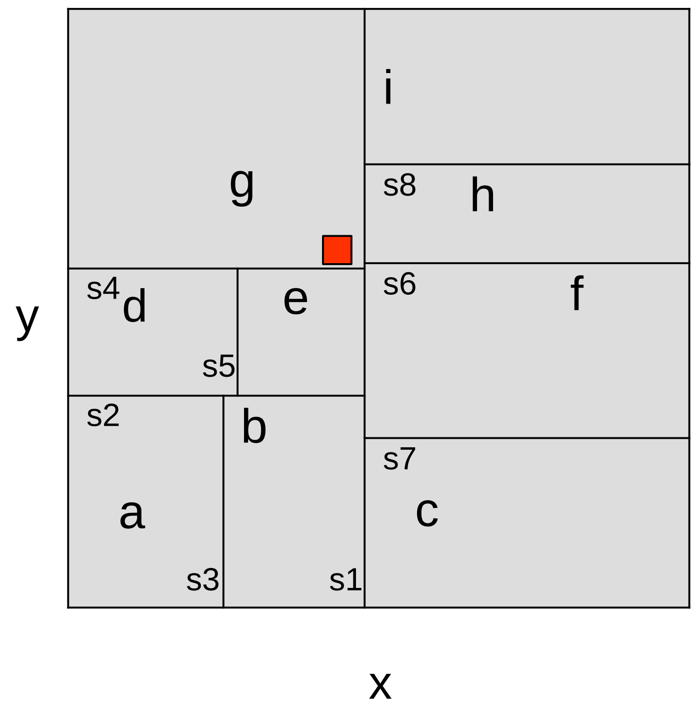
$q(n.axis) + w > n.value$
means the circle overlaps
the right subtree.

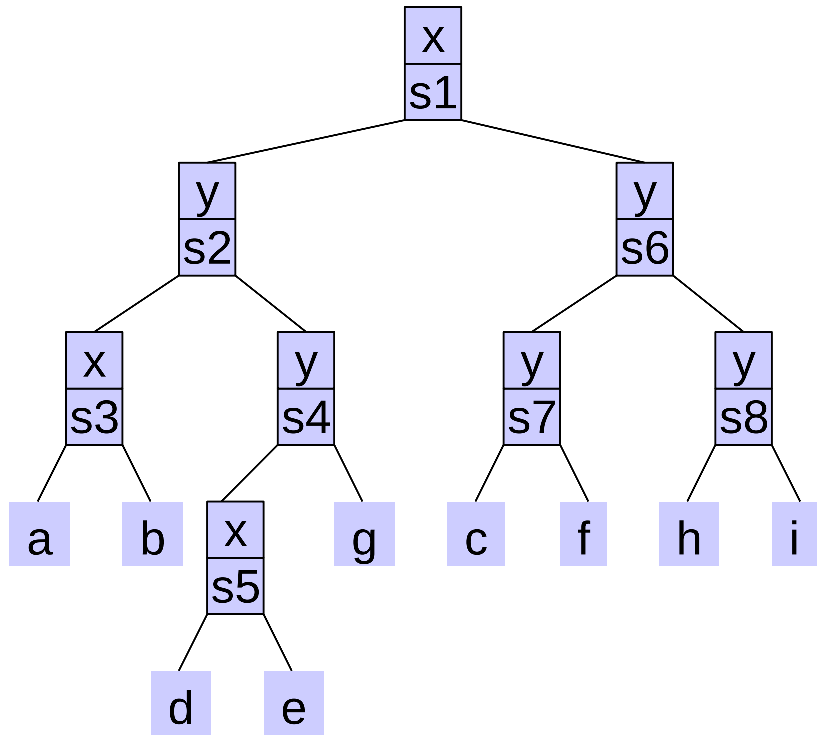
k-d Tree Nearest Neighbor Search

- Search recursively to find the point in the same cell as the query.
- On the return search each subtree where a closer point than the one you already know about might be found.

k-d Tree NNS (1)

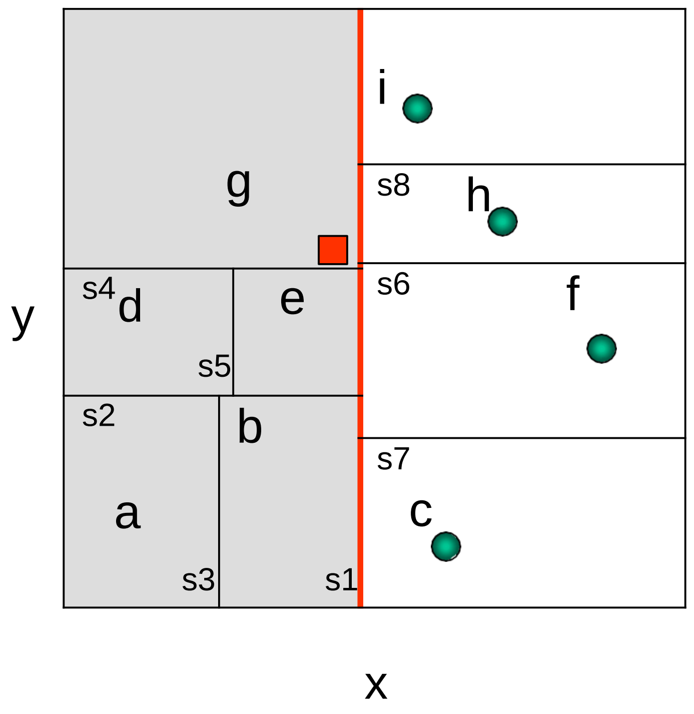
🟥 query point

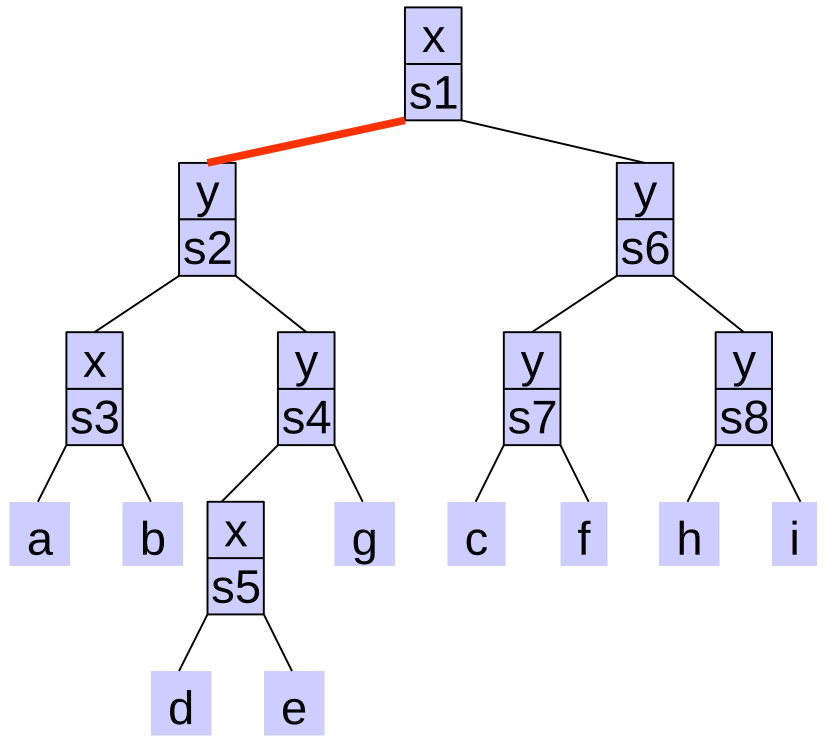




k-d Tree NNS (2)

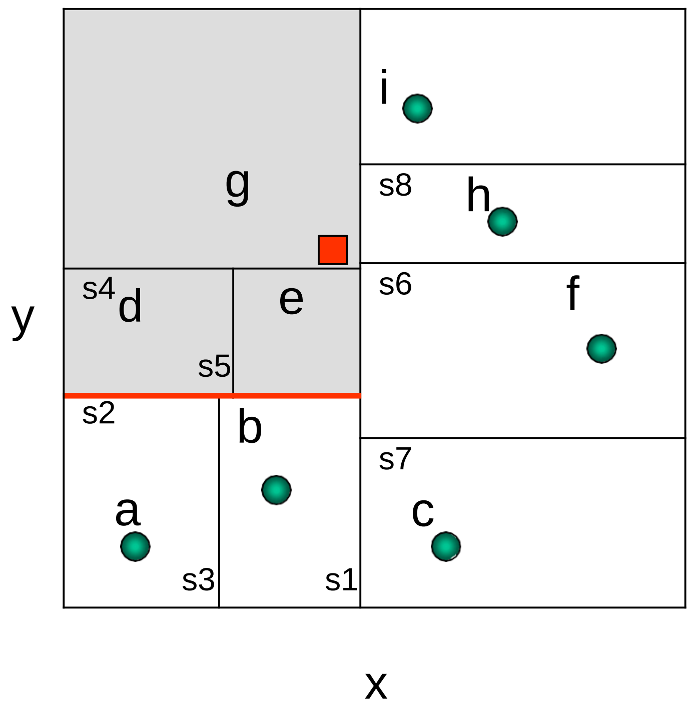
🟥 query point

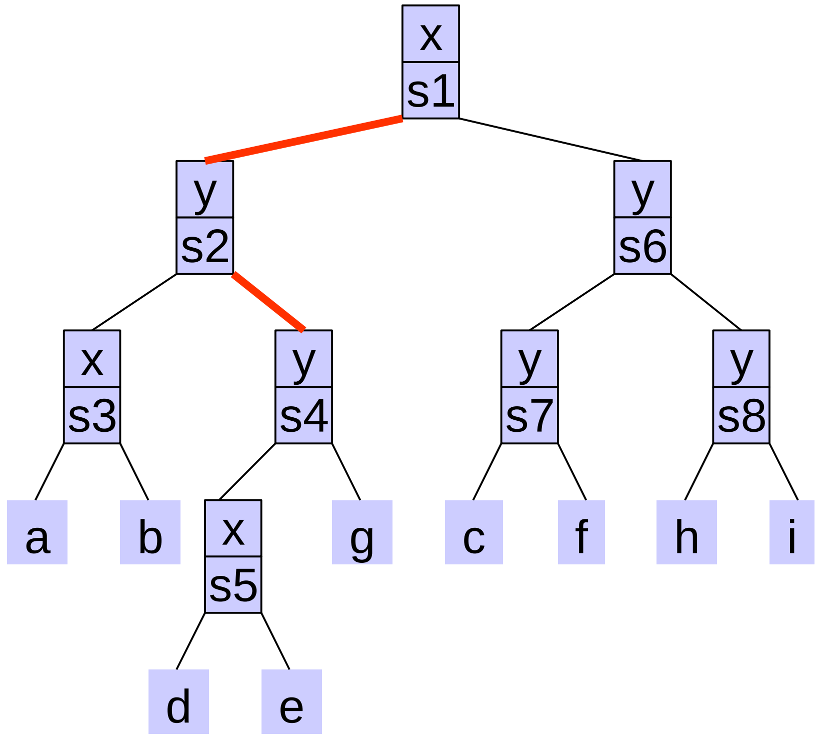




k-d Tree NNS (3)

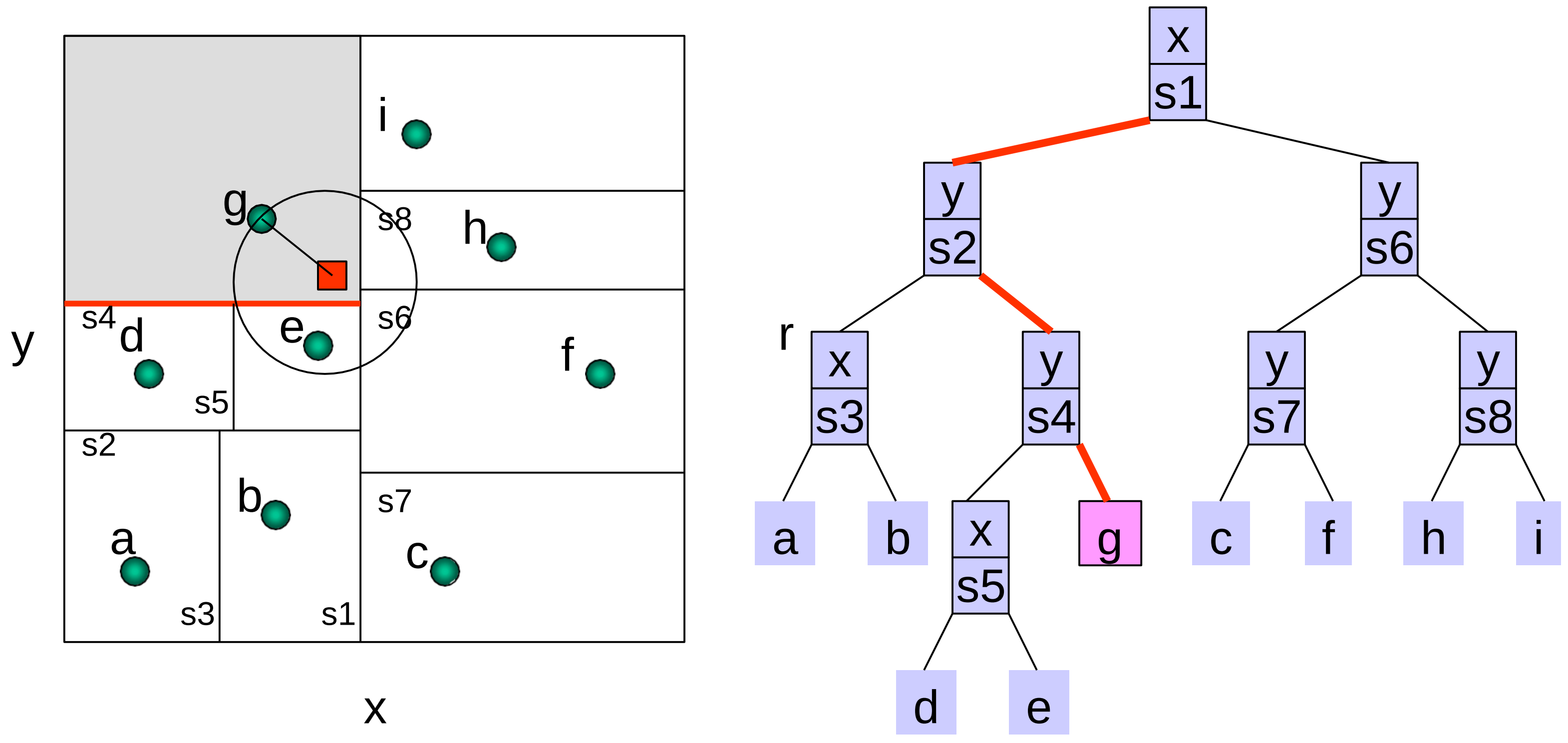
🟥 query point





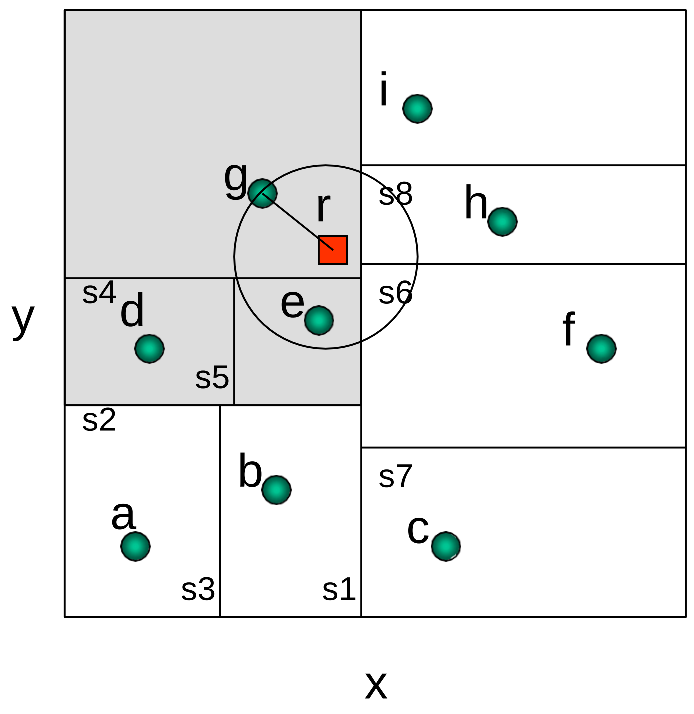
k-d Tree NNS (4)

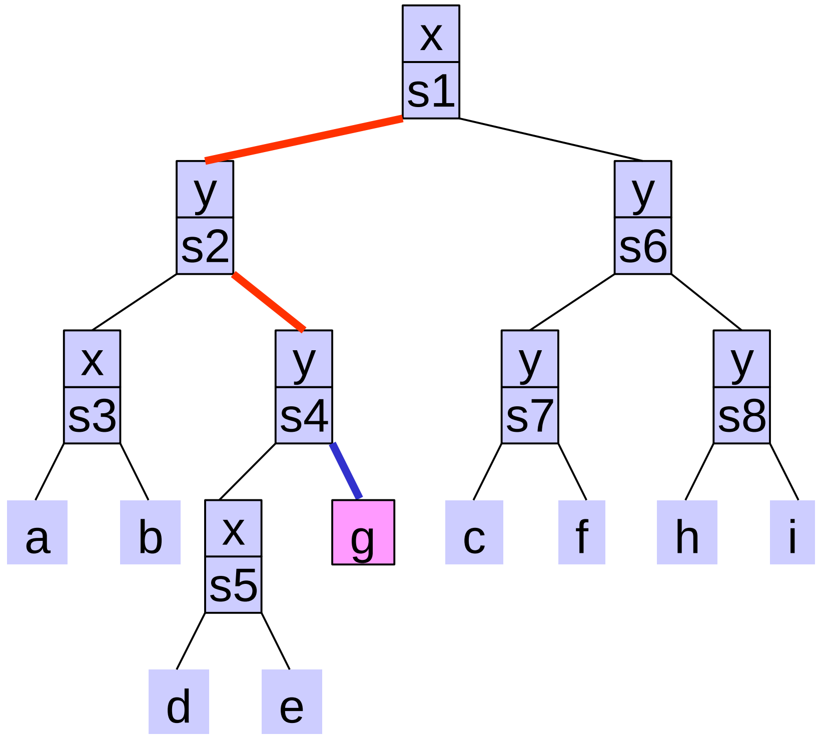
🟥 query point



k-d Tree NNS (5)

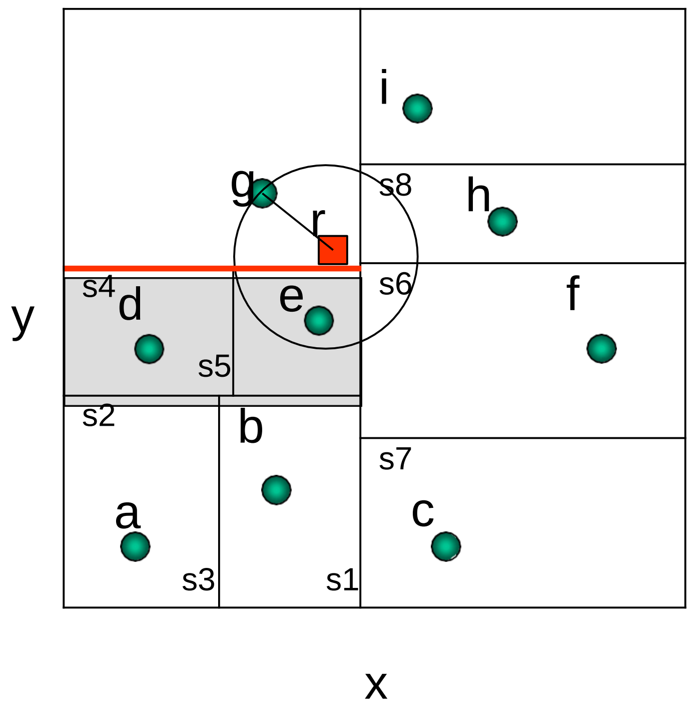
🟥 query point

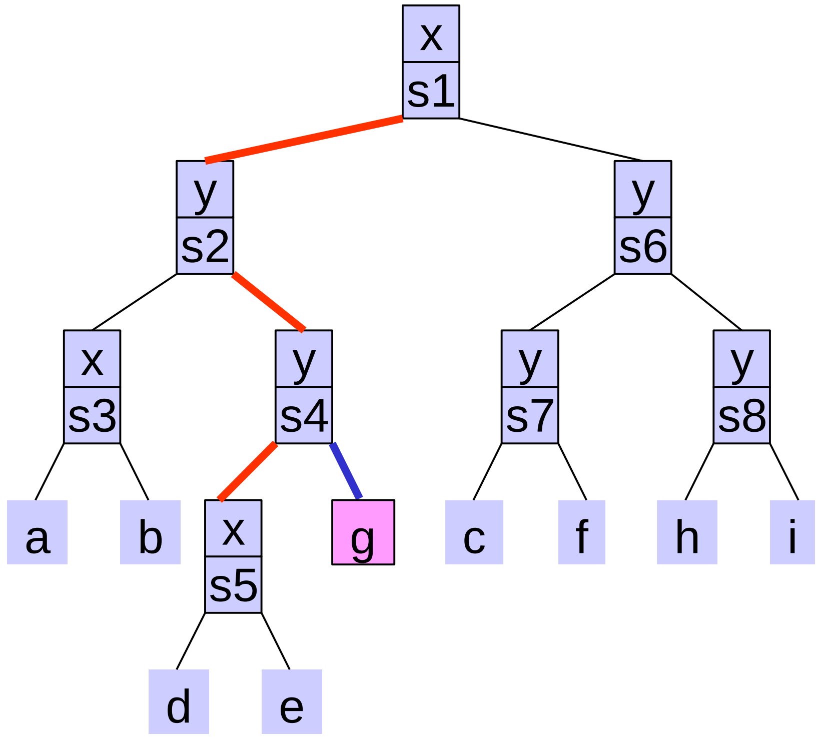




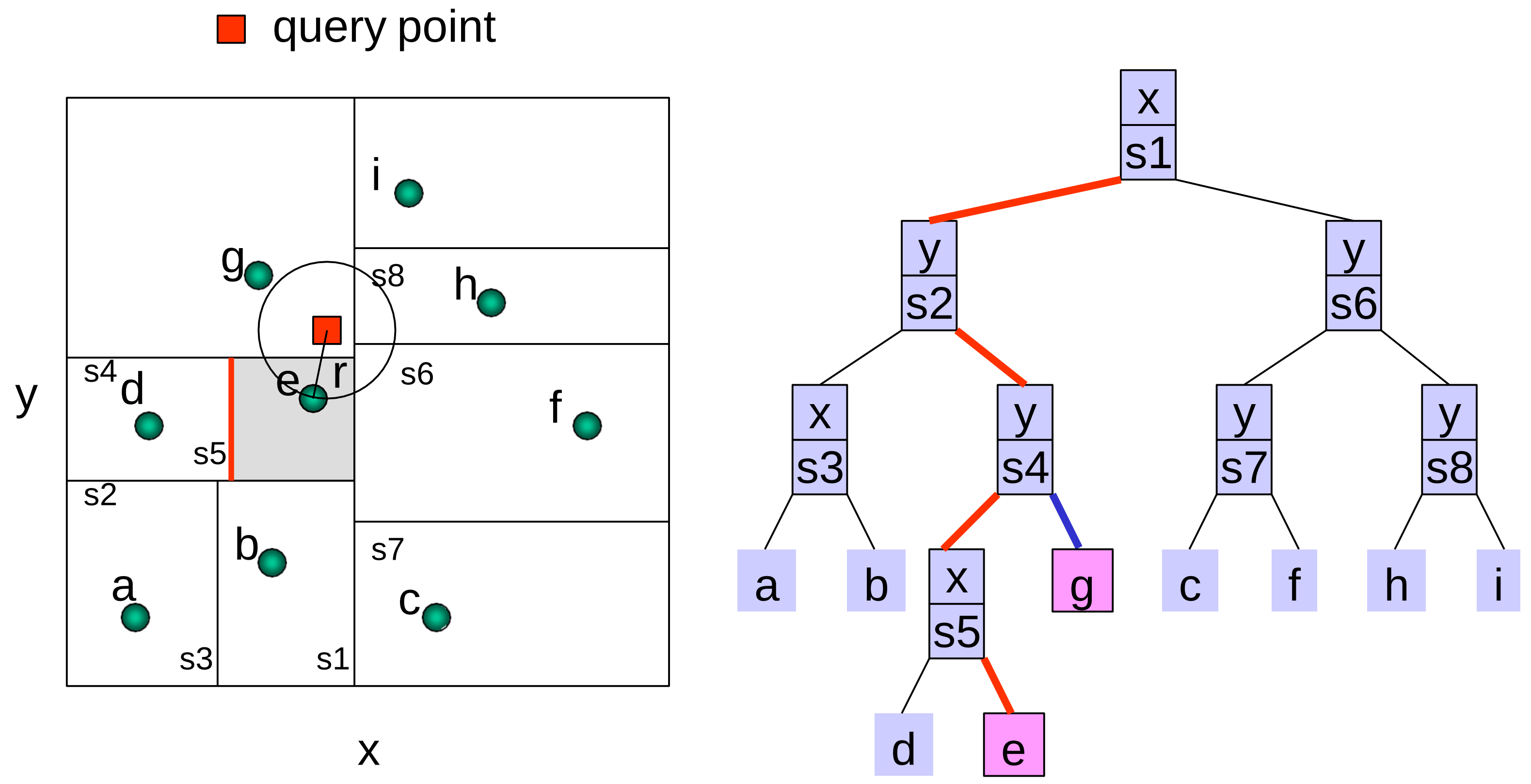
k-d Tree NNS (6)

🟥 query point





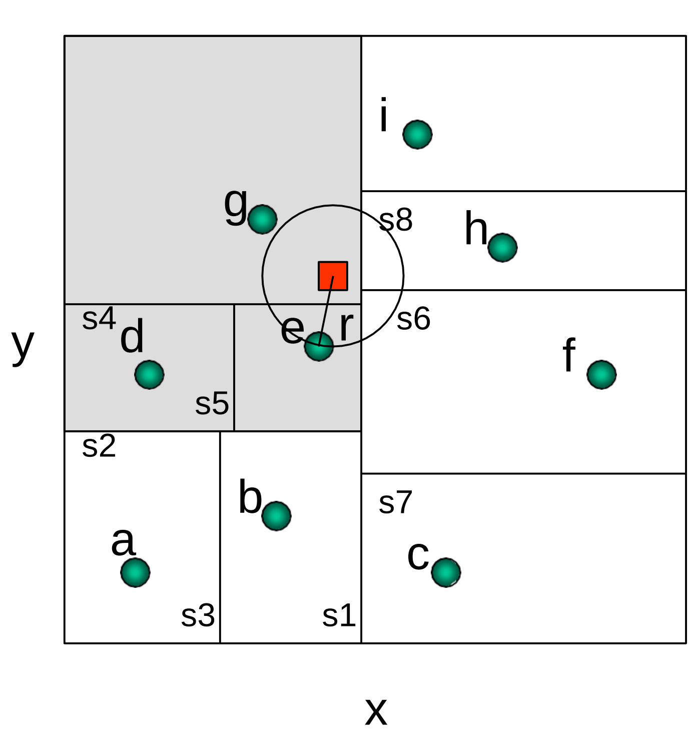
k-d Tree NNS (7)

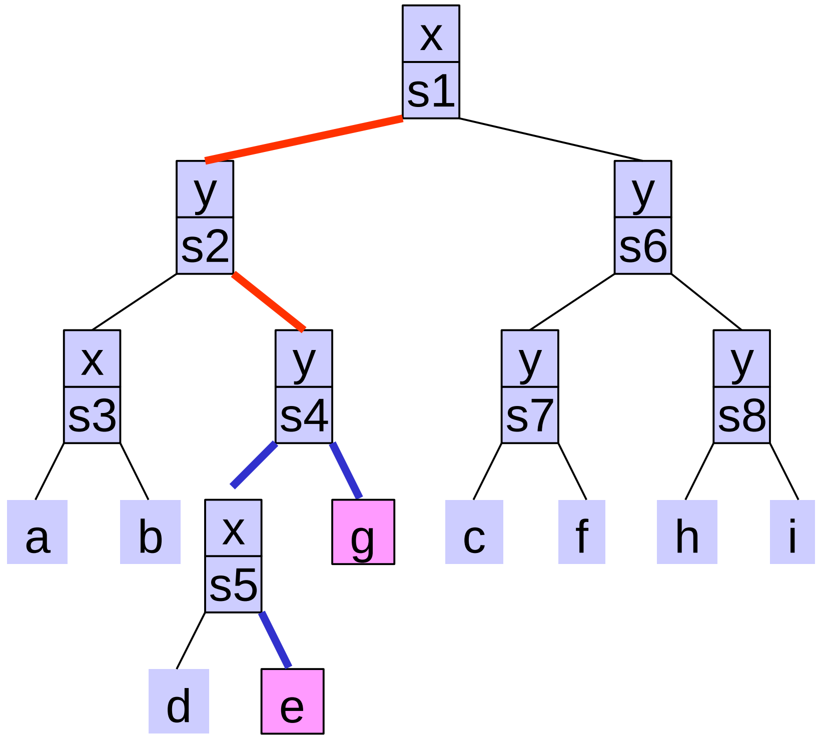


🟥 query point
i
g
s8
h
s4
d
e
r
s6
f
y
s5
s2
b
s7
a
c
s3
s1
x
x s1
y s2
y s6
x s3
y s4
y s7
y s8
a
b
x s5
g
c
f
h
i
d
e

k-d Tree NNS (8)

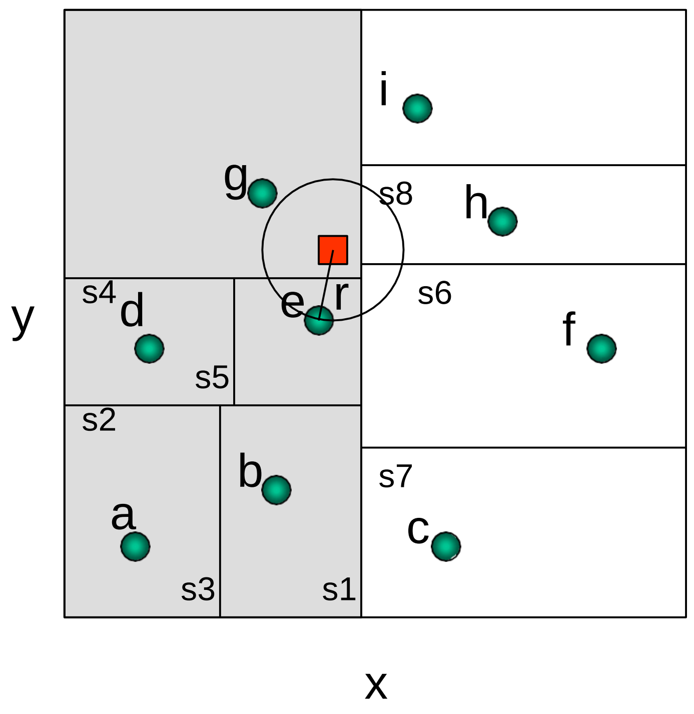
🟥 query point

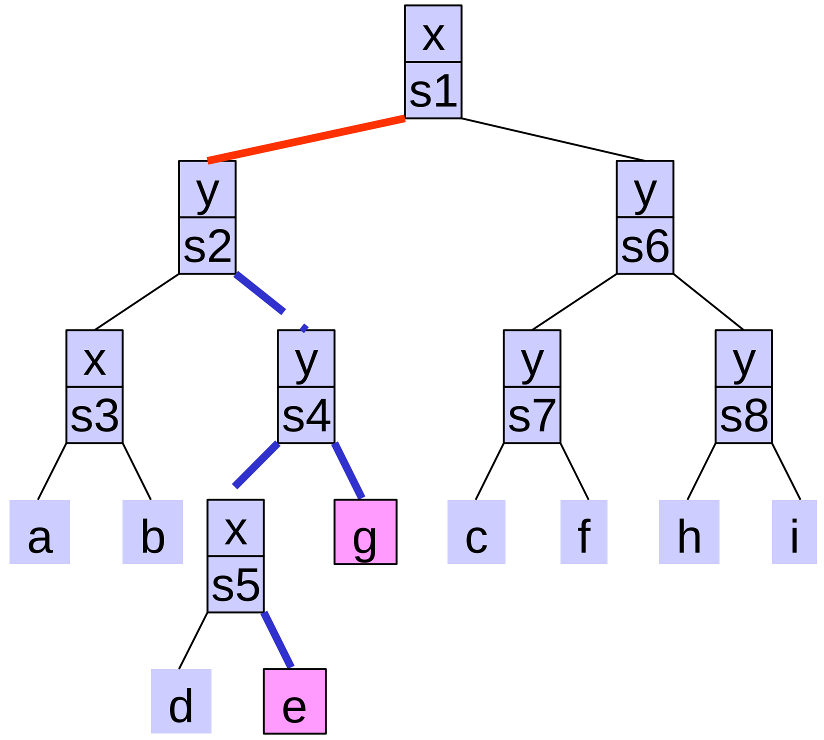




k-d Tree NNS (9)

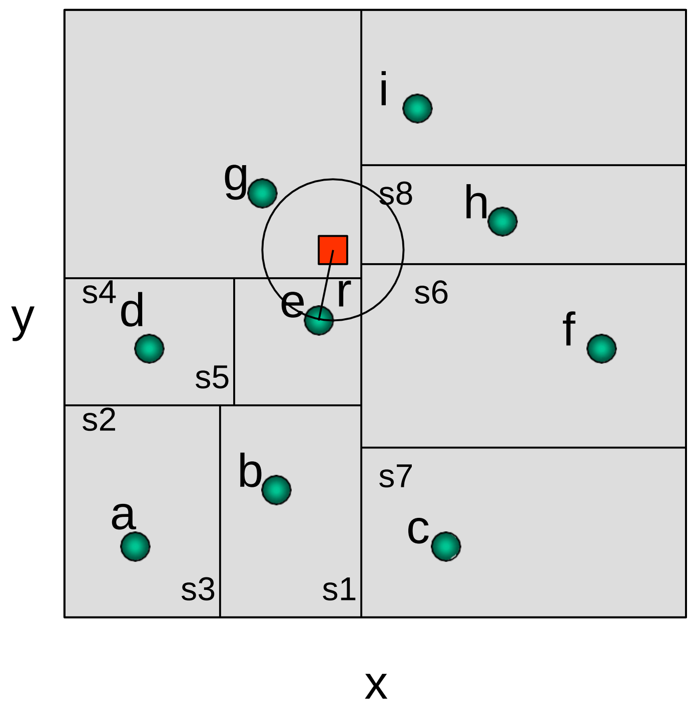
🟥 query point

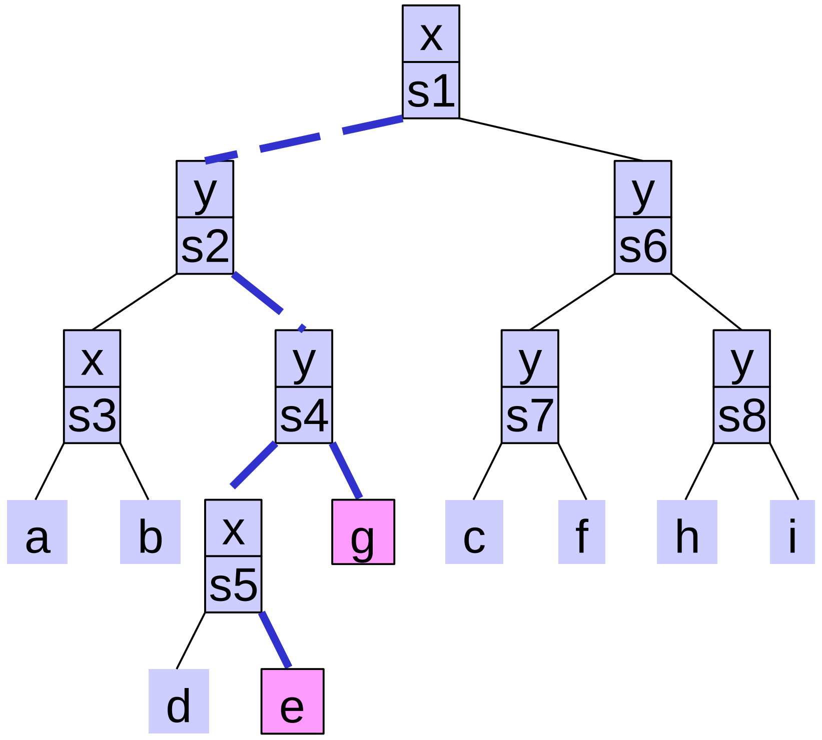




k-d Tree NNS (10)

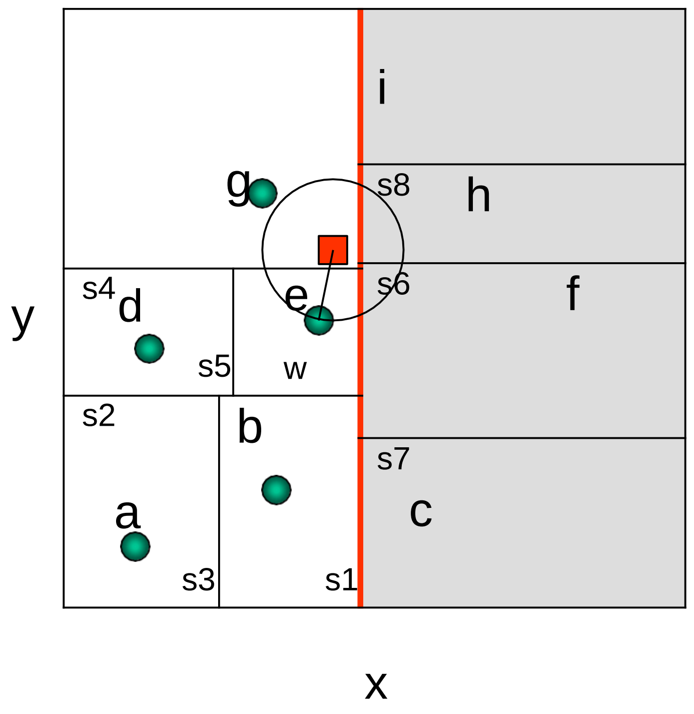
🟥 query point

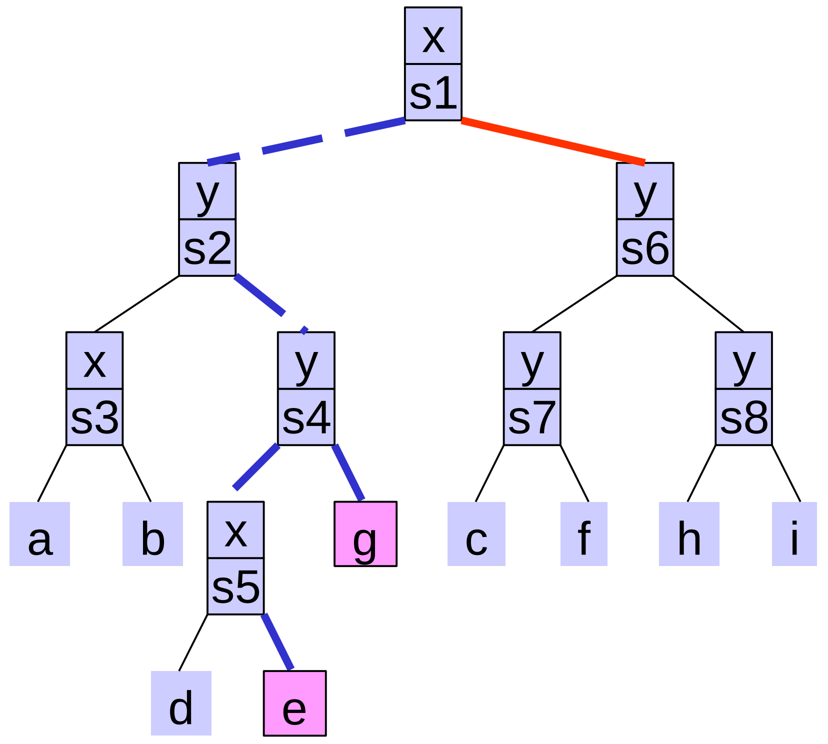




k-d Tree NNS (11)

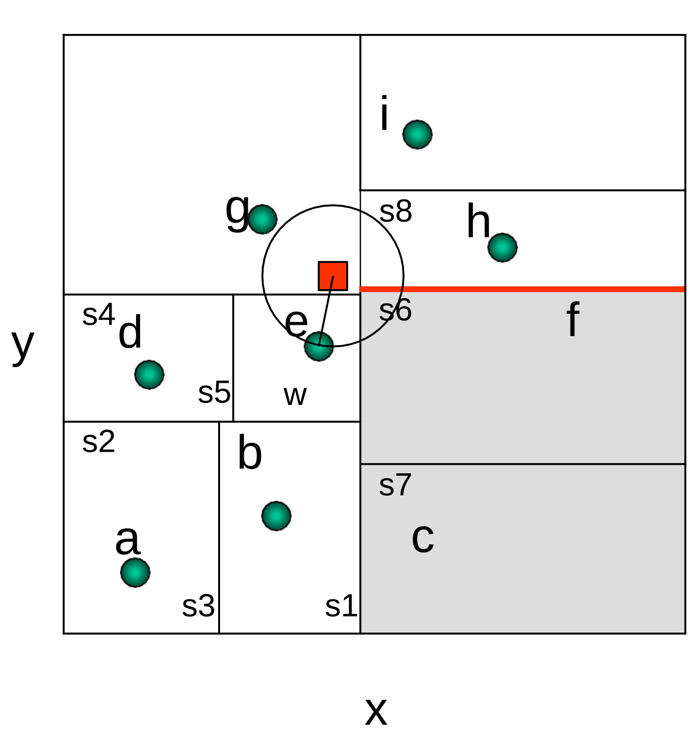
🟥 query point

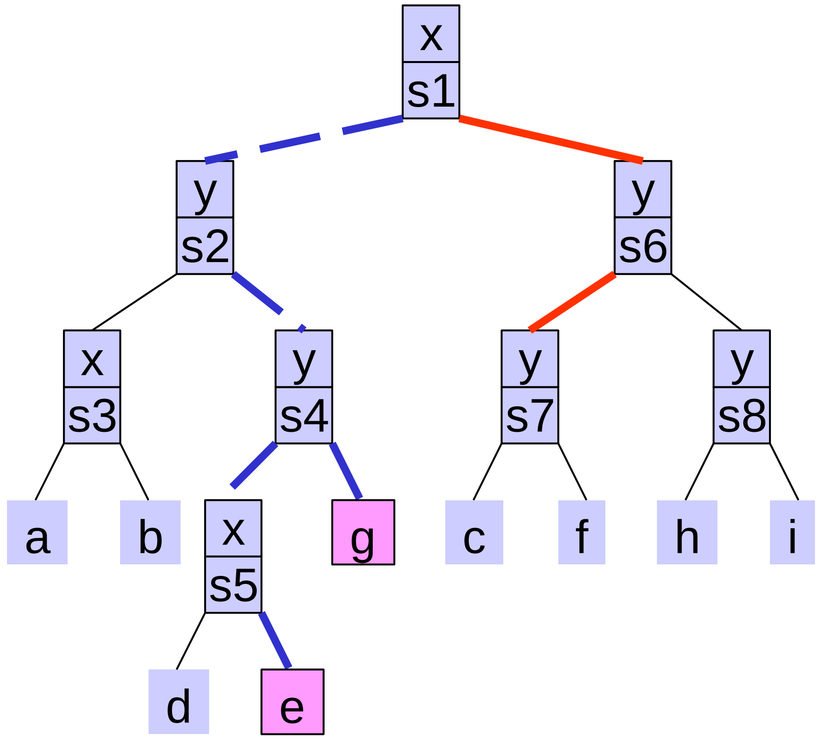




k-d Tree NNS (12)

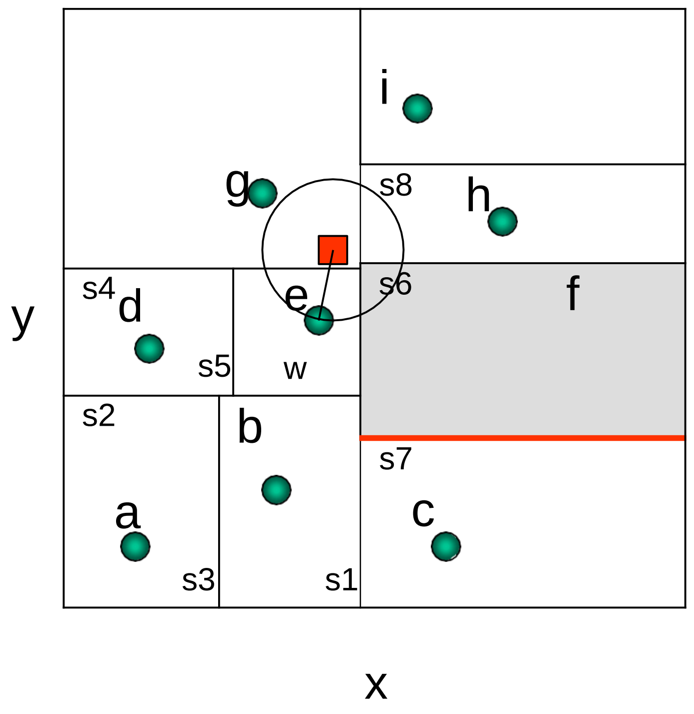
🟥 query point

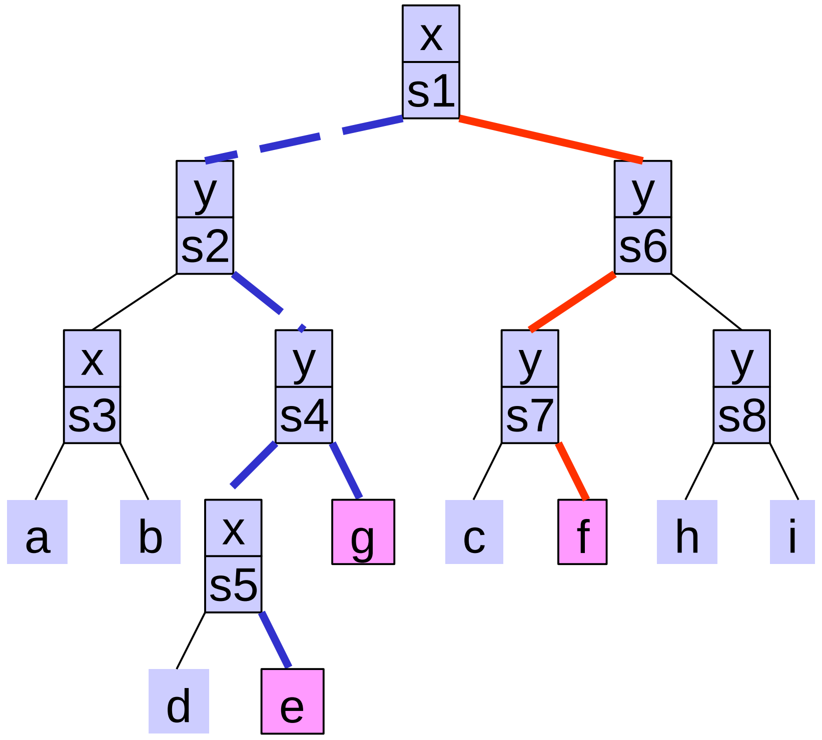




k-d Tree NNS (13)

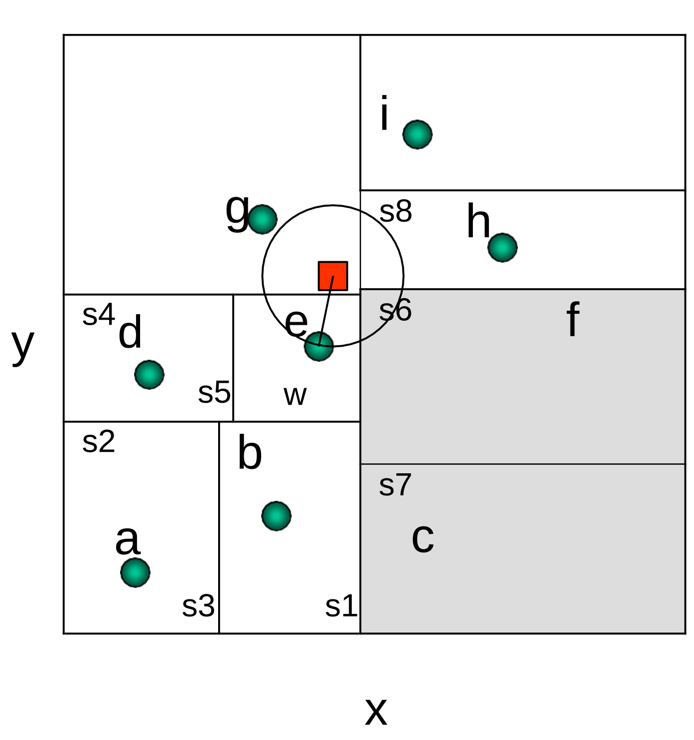
🟥 query point

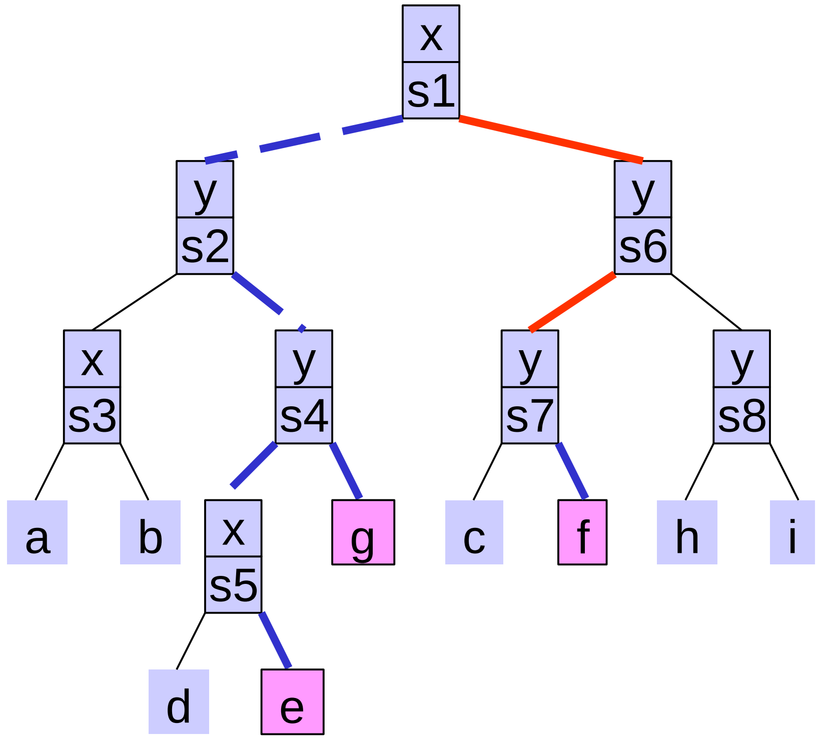




k-d Tree NNS (14)

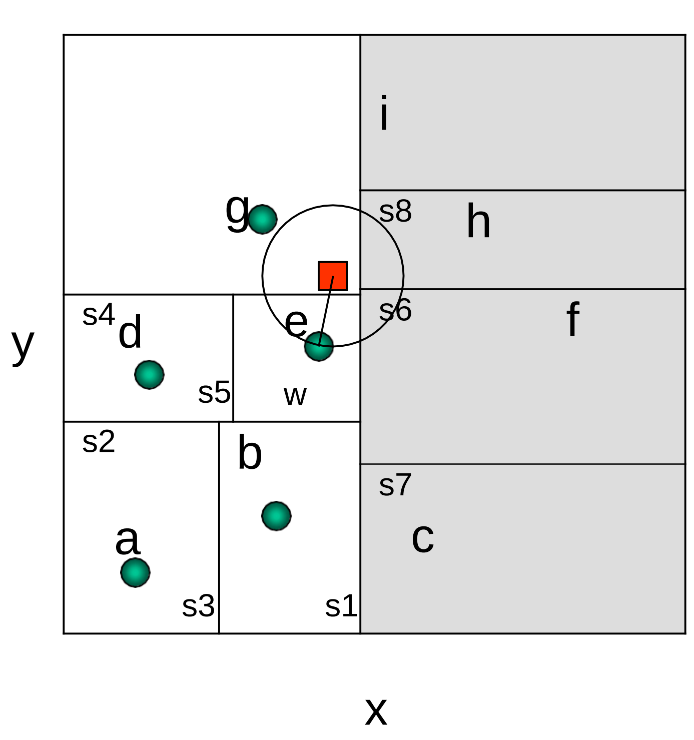
🟥 query point

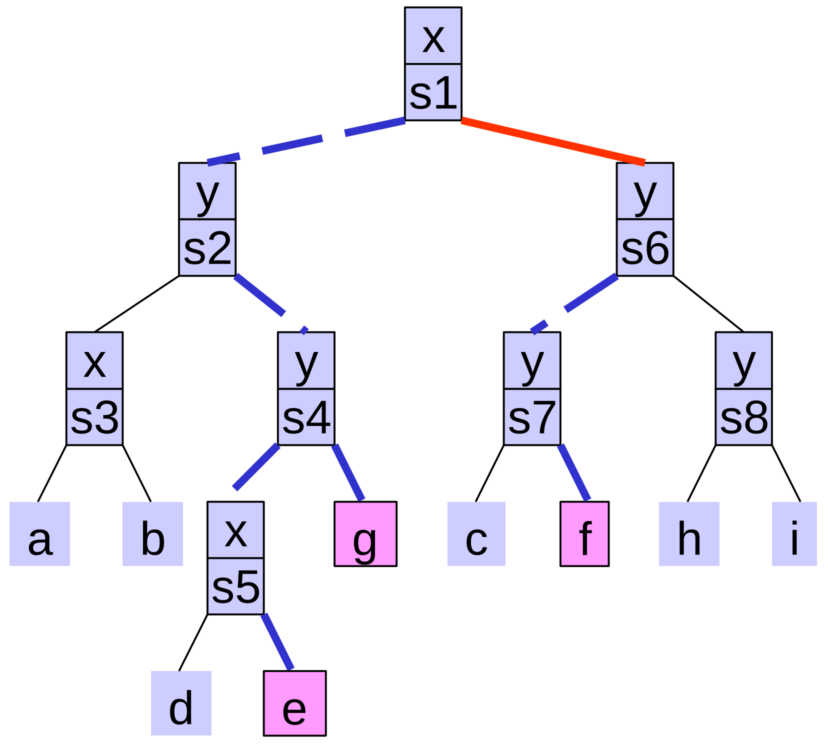




k-d Tree NNS (15)

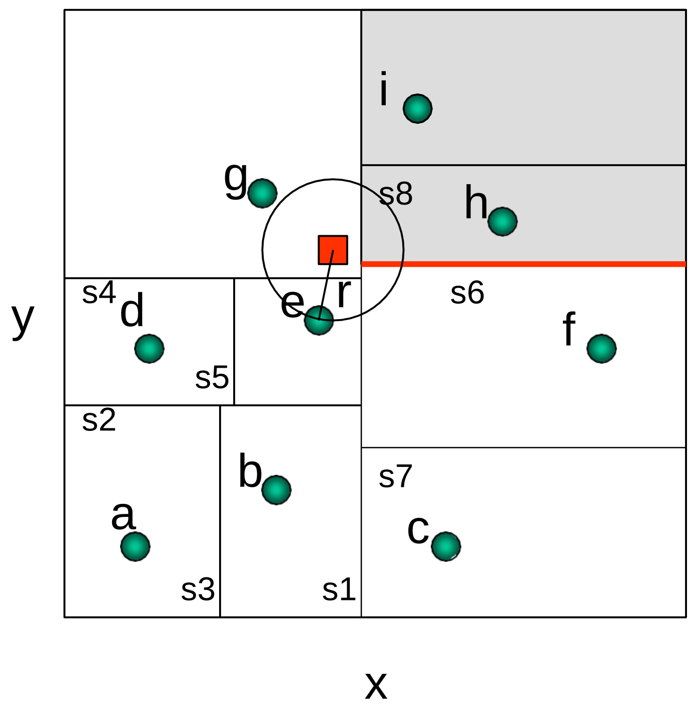
🟥 query point

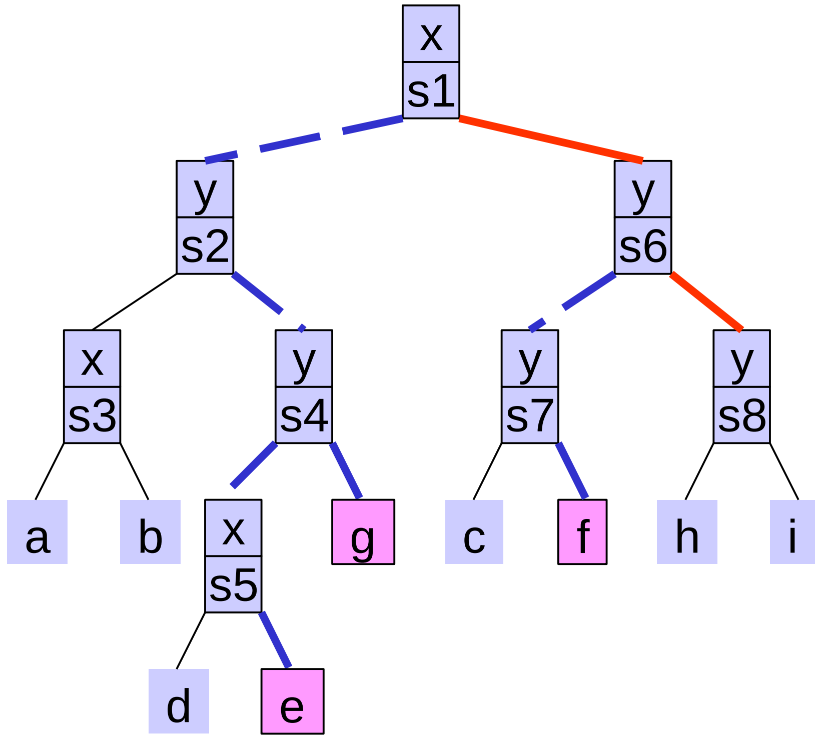




k-d Tree NNS (16)

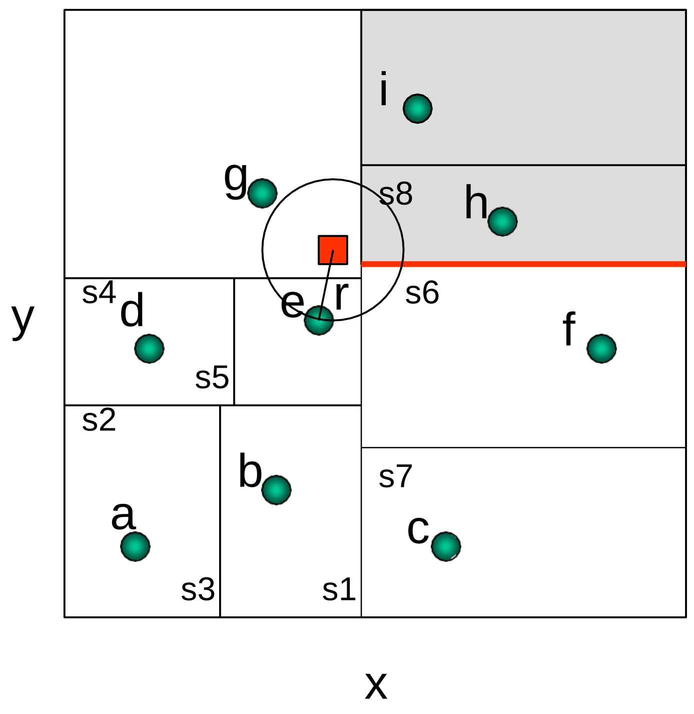
🟥 query point

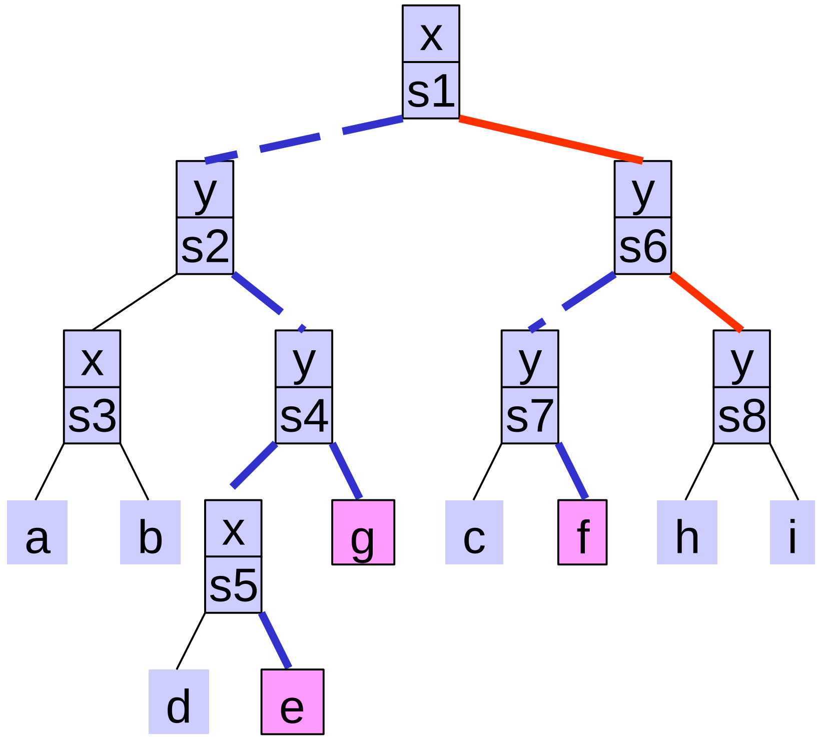




k-d Tree NNS (17)

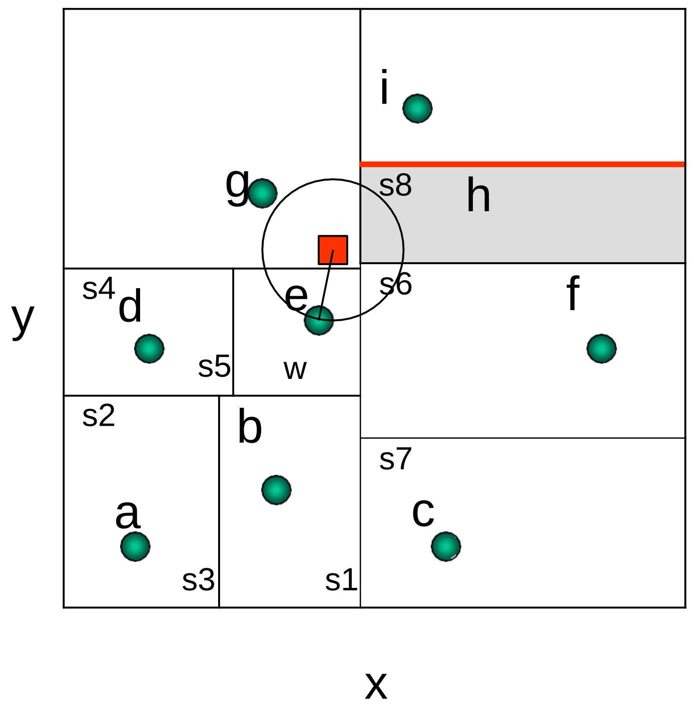
🟥 query point

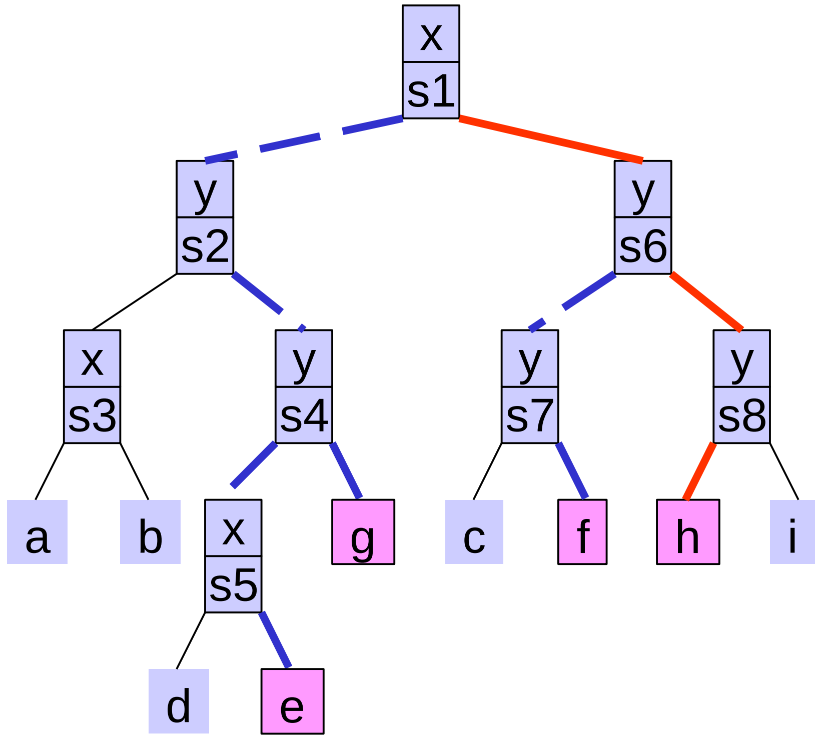




k-d Tree NNS (18)

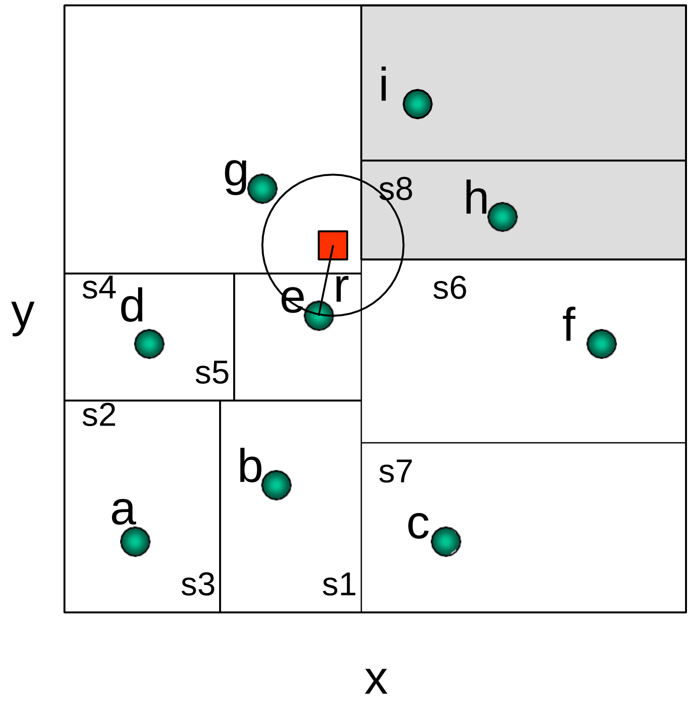
🟥 query point

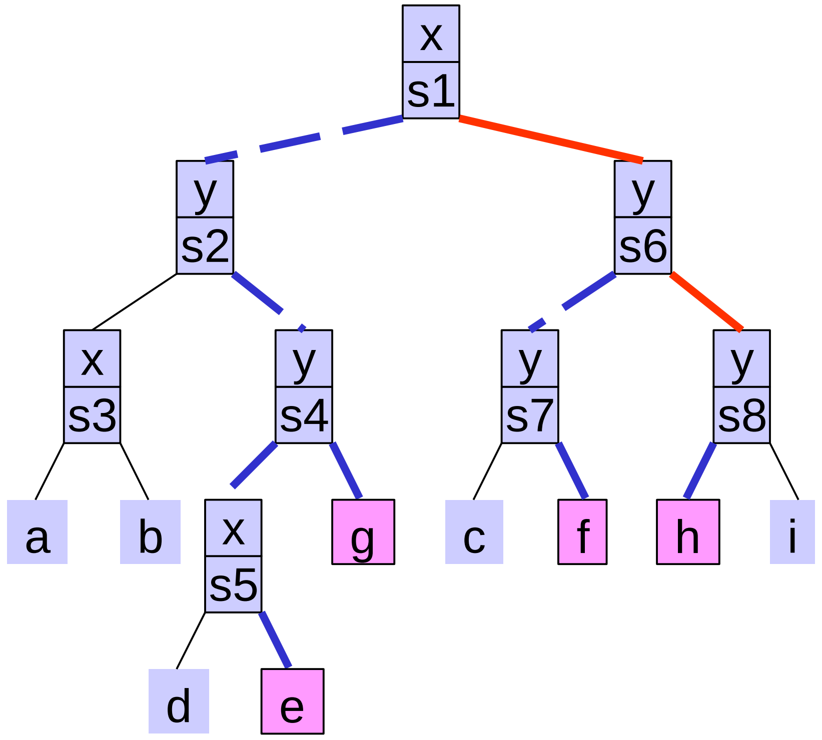




k-d Tree NNS (19)

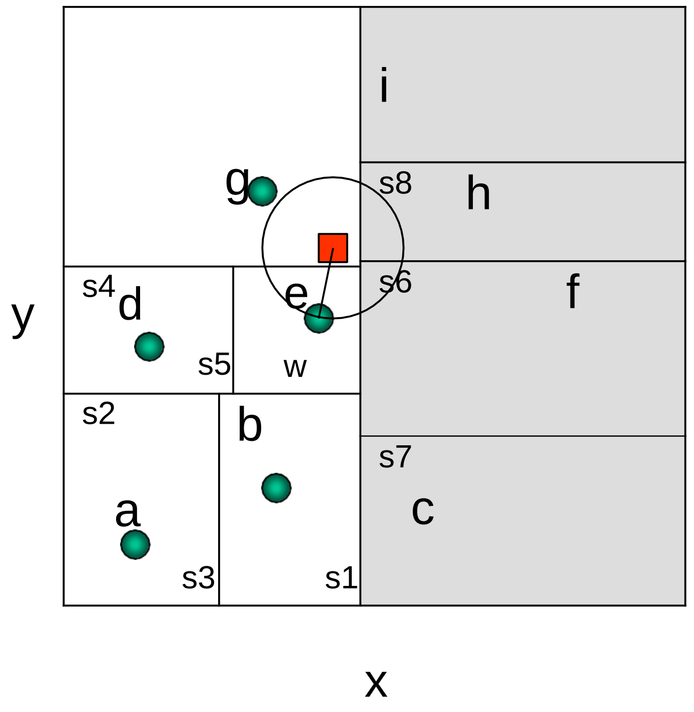
🟥 query point

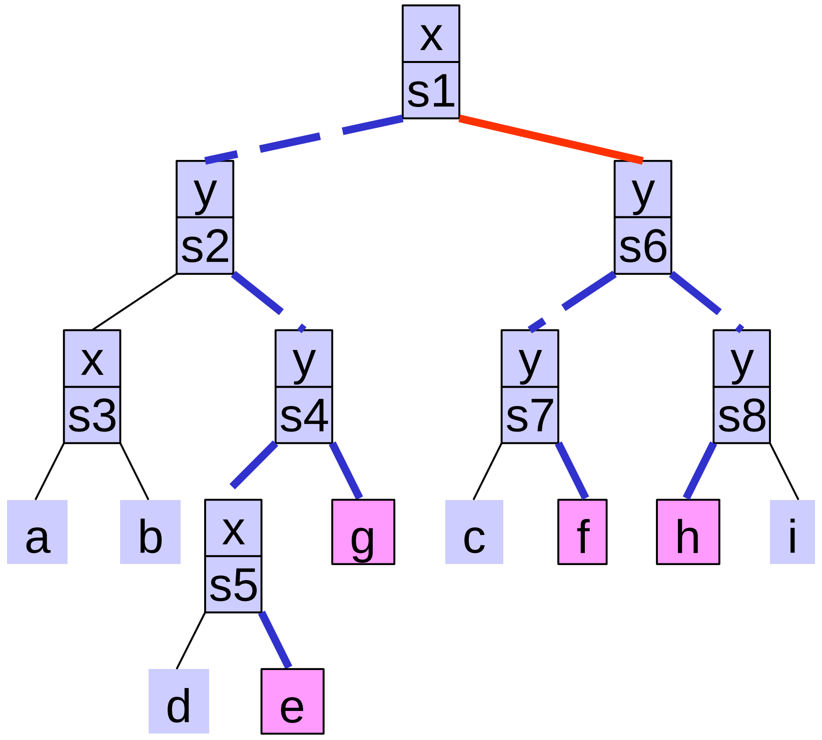




k-d Tree NNS (20)

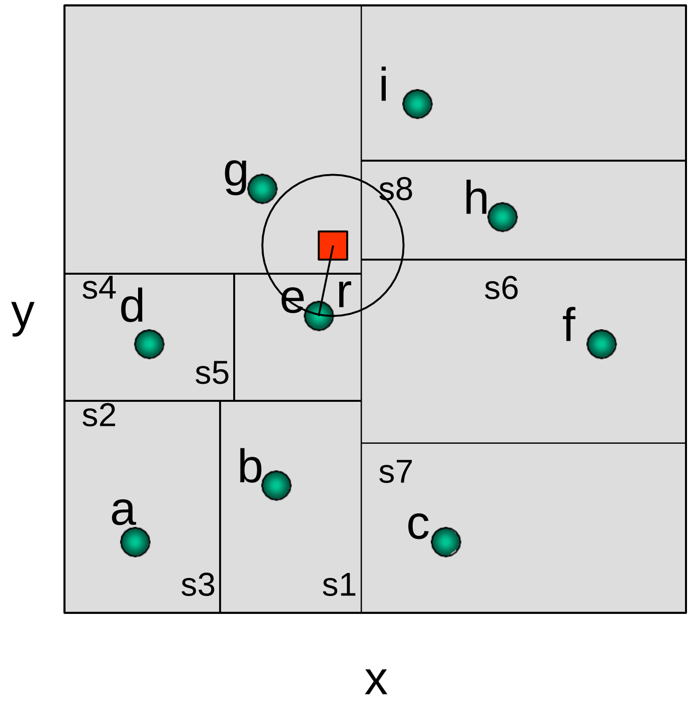
🟥 query point

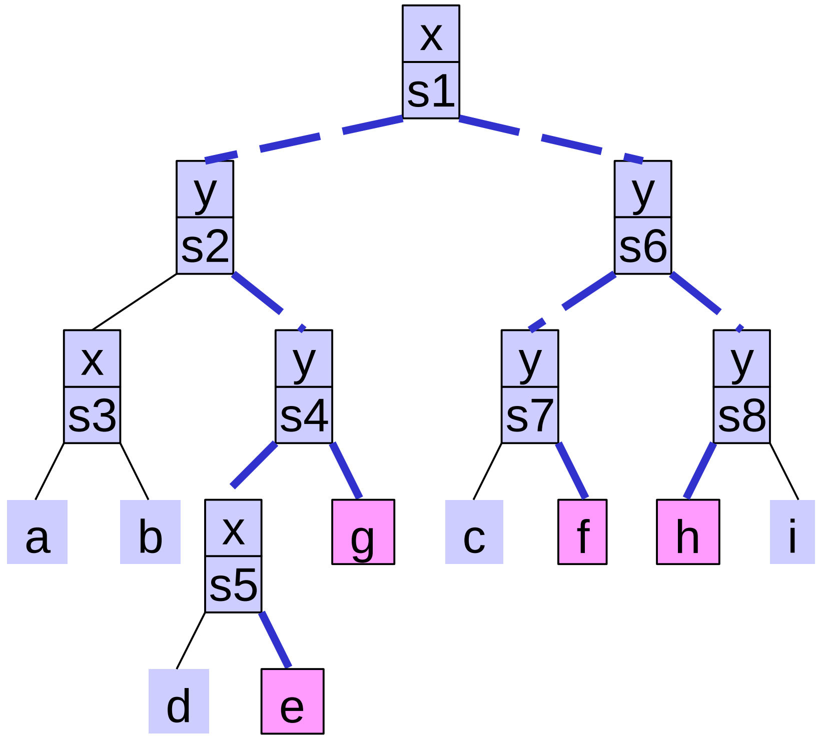




k-d Tree NNS (21)

🟥 query point





Nearest Neighbor Search

Main is NNS(q,root,null,infinity)

```
NNS(q: point, n: node, p: point, w: distance) : point {
if n.left = null then {leaf case}
   if distance(q,n.point) < w then return n.point else return p;
else
   if w = infinity then
     if q(n.axis) ≤ n.value then
        p := NNS(q,n.left,p,w);
        w := distance(p,q);
        if q(n.axis) + w > n.value then p := NNS(q, n.right, p, w);
     else
        p := NNS(q,n.right,p,w);
        w := distance(p,q);
        if q(n.axis) - w ≤ n.value then p := NNS(q, n.left, p, w);
   else //w is finite//
      if q(n.axis) - w ≤ n.value then
      p := NNS(q, n.left, p, w);
      w := distance(p,q);
      if q(n.axis) + w > n.value then p := NNS(q, n.right, p, w);
   return p
}
```

complexity

- A data structure to support range queries in **R^d**
- Building a static *k*-d tree from *n* points takes **O(dnlogn)** time if sorting is used to compute the median at each level.
- Space complexity: **O(dn)**
- Querying a balanced *k*-d tree takes **$O(n^{1-1/d}+k)$**time where k is the number of the reported points, and d is the dimension of the k-d tree.

K-D trees are not suitable for efficiently finding the nearest neighbor in very high dimensional spaces.

Why?

Too many branches need to be backtracked! Close to linear time

Reducing Complexity

Find projection to a lower dimensional space so that the distances between samples are approximately the same

- PCA(Principal Component Analysis)
- Projection to a Random subspace
- …

Home work

- Hello World of Machine Learning!

1.实现K-近邻算法识别手写数字数据集。
2.改变K的值、修改为随机选取样本、改变训练样本数目，观察对算法错误率的影响。
3.体会“机器学习：数据驱动的科学”。

Thanks !