ELTE | FACULTY OF INFORMATICS

3D Point Cloud processing and analysis
Nearest Neighbor Search

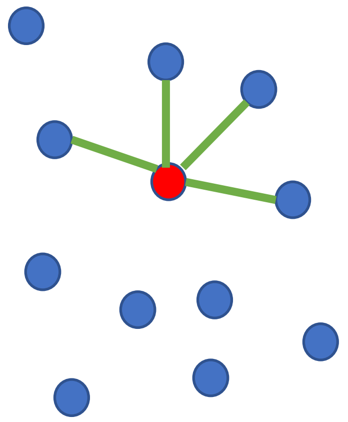
Massinissa Aouragh:
Faculty of Informatics, Department of Artificial Intelligence
Robert Bosch Kft
m2j7au@inf.elte.hu

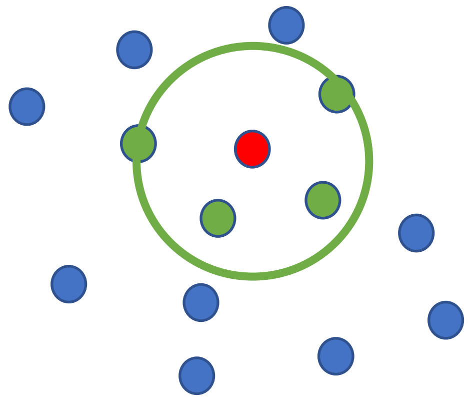
Nearest Neighbor search

- Irregularity of point cloud, neighborhood of pixel is easily identified by creating a grid around the pixel
- Nearest neighbor search serves as basic element for building local neighborhoods
- NN search is used in radius noise removal, statical noise removal, and surface normal estimations.

Nearest Neighbor search

- K-NN: Given N points in a space S, for a query point p Є S, K-NN finds the K closest points to p in space S
- Radius-NN: Finds all points q that satisfy the criteria ||q – p|| < r





Search Trees

- Radius-NN slightly faster than the K-NN approach, the time complexity of finding distances that are smaller than r is O(N)
- Finding the top K smallest distances takes from O(NlogK) to O(NlogN) depending on the sorting algorithm
- Complexity of the problem skyrockets when we deal with huge data in real-time (e.g., HDL-64 scans ~100k at 20Hz in O(N^2) = 10^9)
- Space partitioning can be one of the solutions for efficient realization of the algorithms

Space partitioning

First split the space into different areas, and only some areas are searched instead of all the data:

- Binary Search Tree (BST)
- K-dimensional (k-d) Tree
- Octree

Binary Search Tree

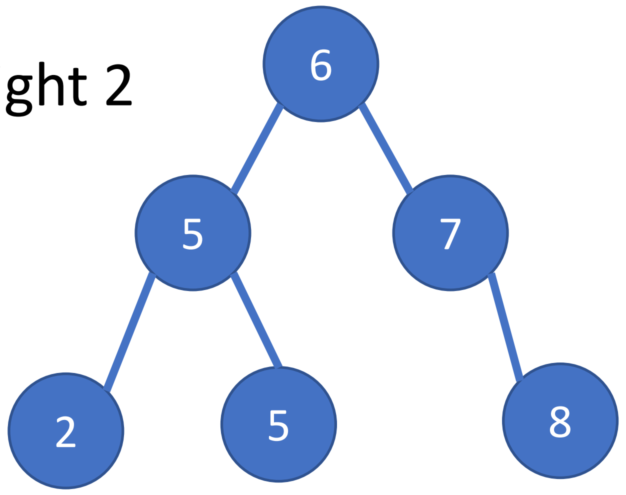
- Node based tree data structure
- Each node stores a key
- Each key is greater than the keys in the left subtree but lower than the keys in the right subtree
- Basic operations: tree construction, insertion, searching and deletion

BST Construction

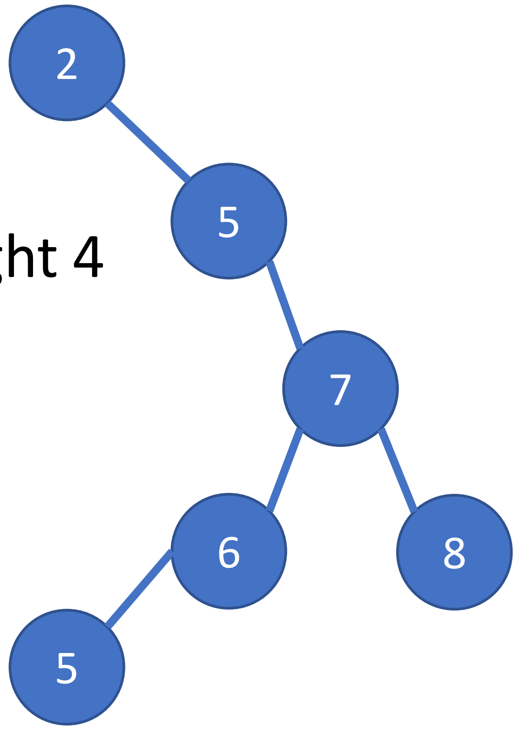
- Recursively inserting points from point clouds in the tree. The worst-case time complexity is O(h) where h is the height of the BST
- If tree is extremely imbalanced then h is the number of points in point cloud. If the tree is balanced, then h = $\log_2 N$

BST Construction

Height 2

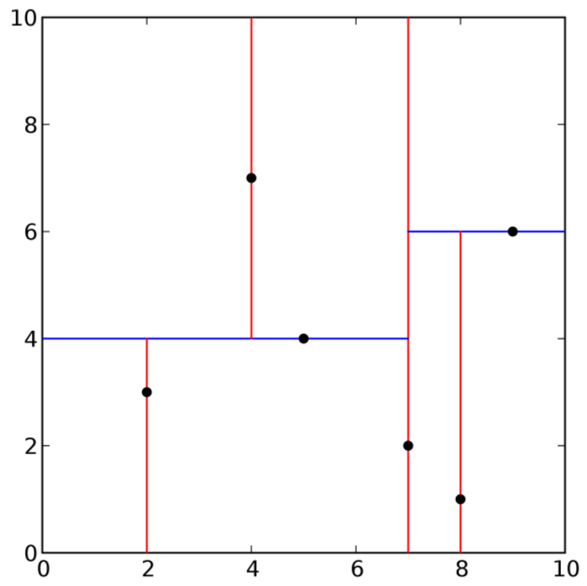


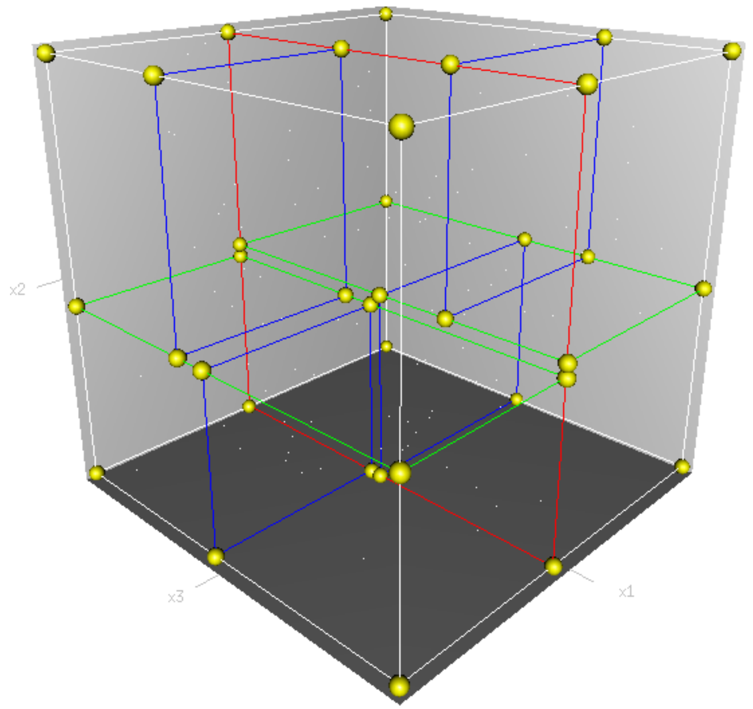
Height 4



K-Dimensional Tree

- Binary tree where every leaf node is a k-dimensional point
- Extension of BST into higher dimension

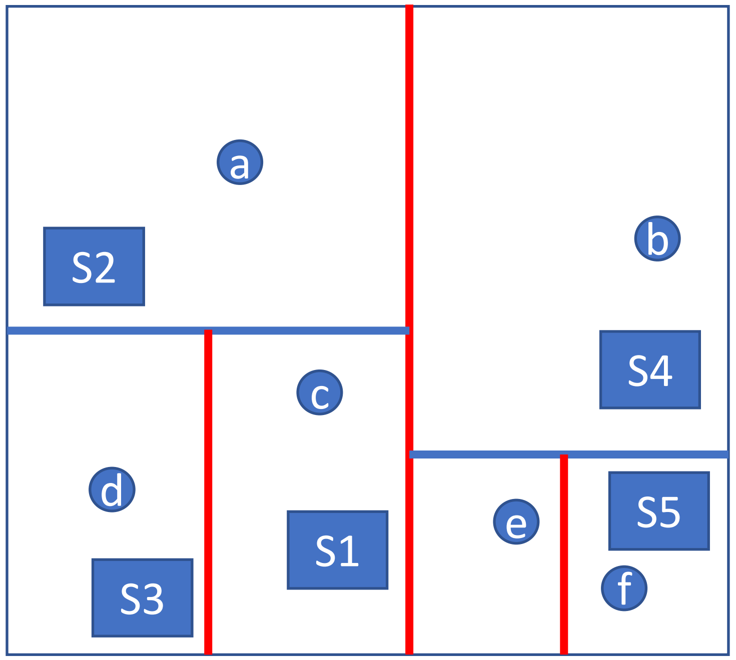


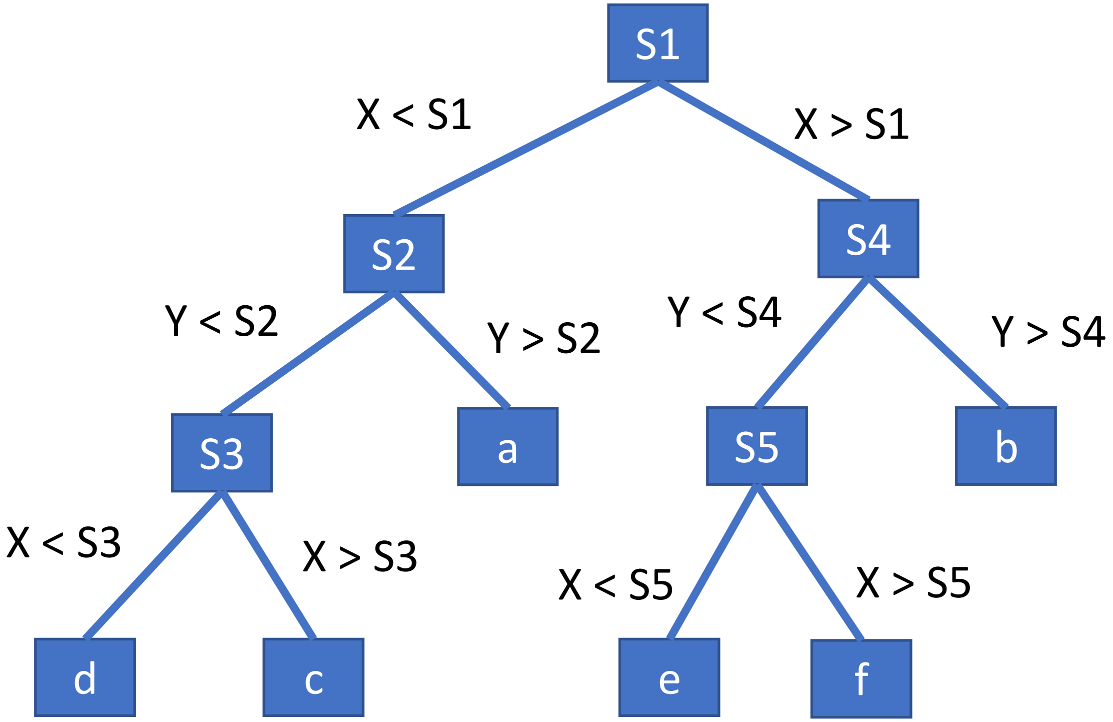


K-Dimensional Tree Construction

- If the node has only one point or the number of points is less than the leaf size, stop splitting and store the node as a leaf node
- Otherwise, divide the points of the node into two parts by a hyperplane perpendicular to the selected axis. Points to the left of the hyper plane go to the left subtree of that node, and points to the right of the hyper plane go to the right subtree.
- Repeat the first two steps until the stopping criteria are met

K-Dimensional Tree Construction





Octree

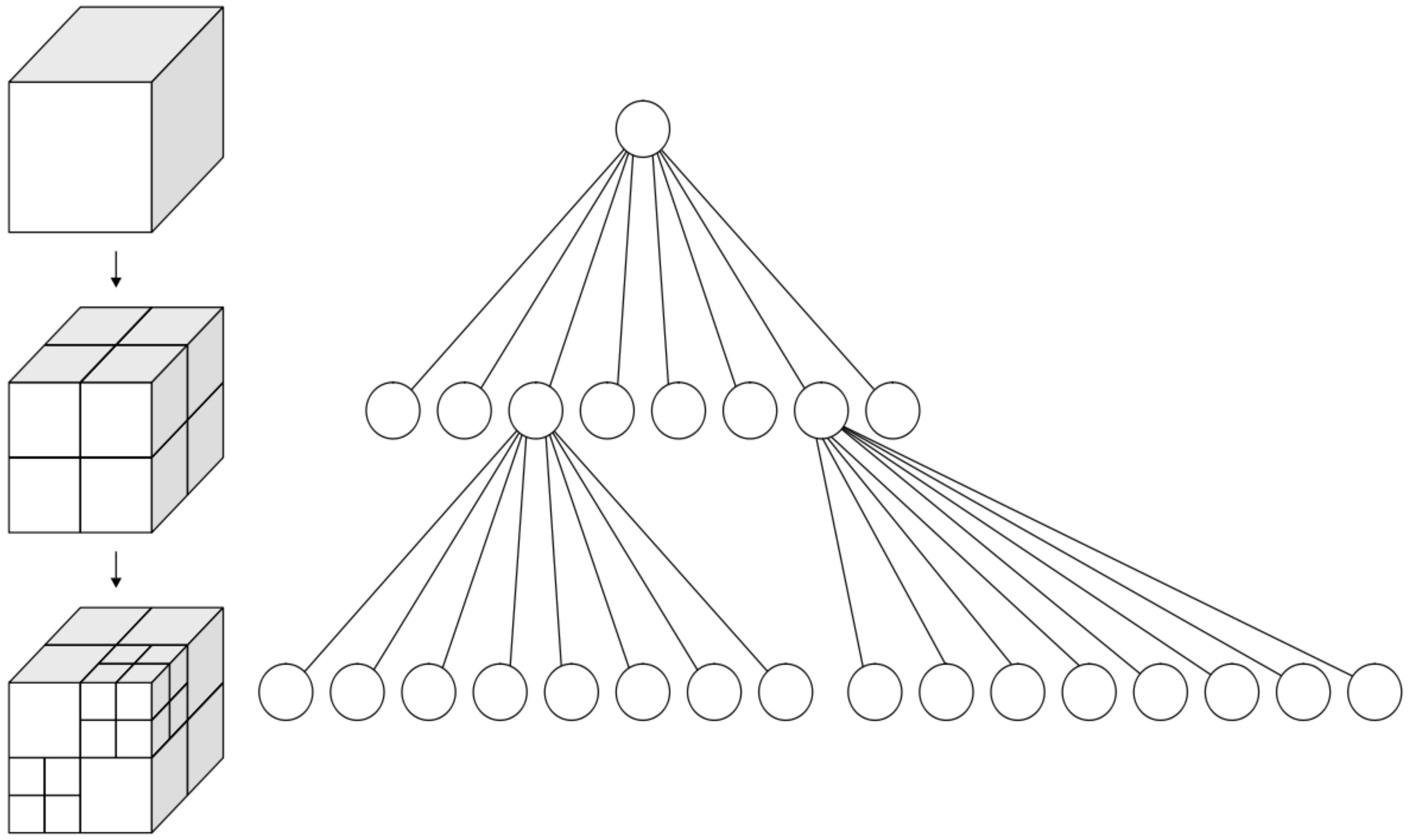
- Tree data structure wherein each node has eight children nodes
- Most commonly used to partition 3D space into eight octants
- Difference between k-d tree on that k-d trees split slong a dimension while octree split around a point

Octree Construction

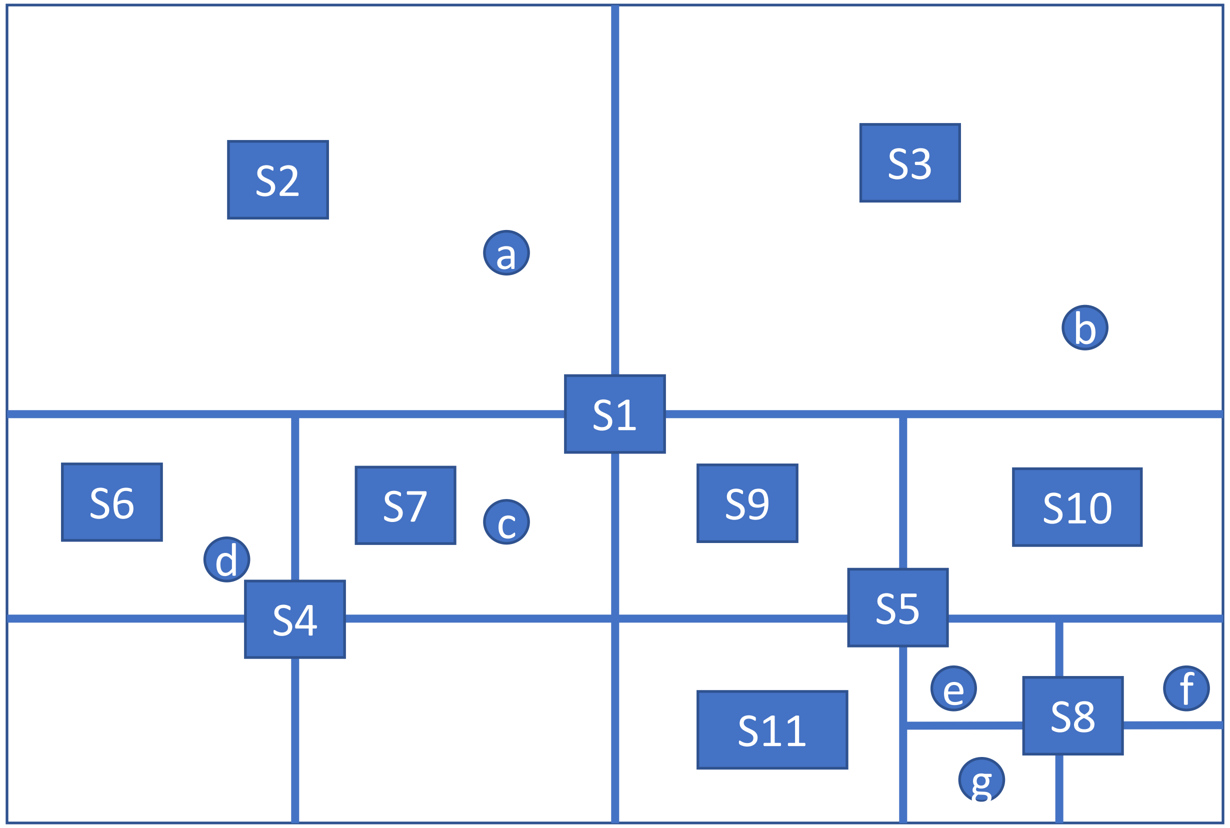
There is two kinds of octree:

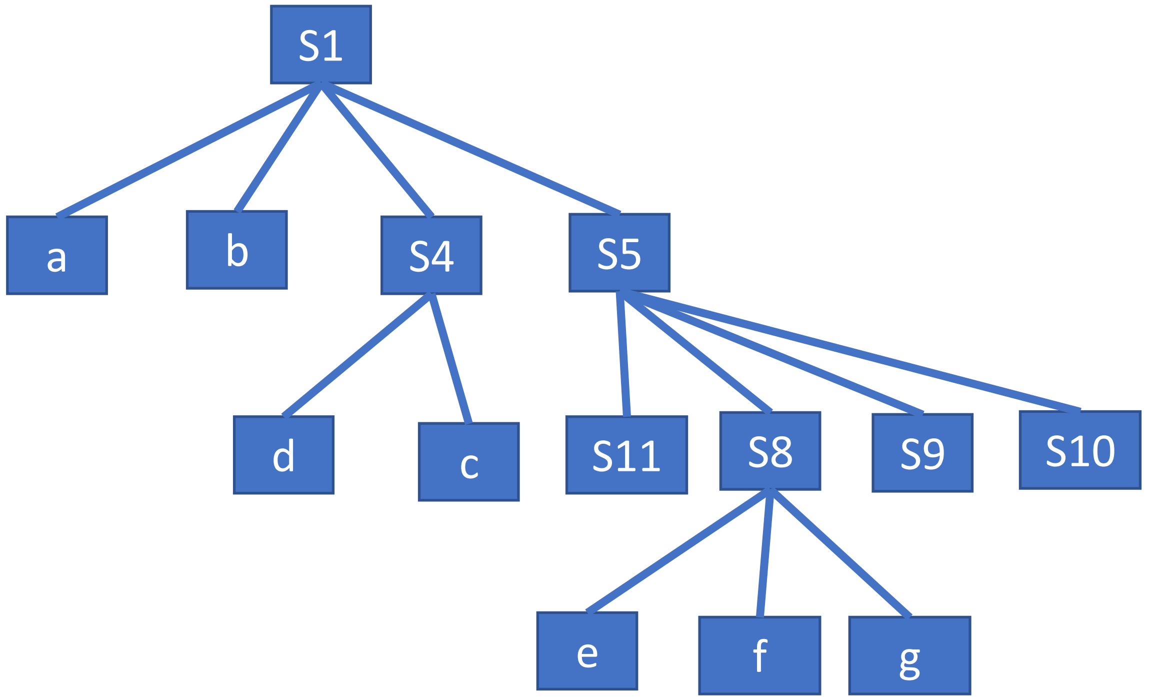
- Point region: store the center of a region as a pseudo 3D point that further defines one of the corners of each of the eight children
- Matrix based: The nodes are implicitly the center of the space they represent

Octree construction



Octree construction


S2
a
S3
b
S1
S6
S7
c
S9
S10
d
S4
S5
e
S8
f
S11
g


S1
a
b
S4
S5
d
c
S11
S8
S9
S10
e
f
g