

## Konstruktion des kd-(Teil-)Baums, der einer Teilmenge von Punkten der Punktwolke entspricht.

```
def kd_subtree(cloud, subset, axis=0):
    """Erzeuge den (Teil-)Baum, der sich aus der sukzessiven
    Zerlegung der Teilmenge subset der Punktwolke cloud ergibt;
    gib die Wurzel dieses Teilbaums zurück."""
    dimension, nof_points = cloud.shape

    # Create a new node:
    node = BinaryTreeNode()

    # Determine cardinality of subset:
    columns = np.nonzero(subset)[0]

    if len(columns) == 0:
        return None

    if len(columns) == 1:
        # This is a leaf! Store (a copy of) the (single) point
        # in subset as data (i.e.: Remember column with Index
        # columns[0]) ...
        j = columns[0]
        node.contents = [j,np.copy(cloud[:,j])]
        # ... and the axis-coordinate as key ...
        node.key = (node.contents[1][axis],axis)
        # ... for this node and return it:
        return node

    # Implicit else:

    # Split subset of cloud along axis:
    median, left_block, right_block = split_cloud_of_points(
        cloud,
        subset,
        axis
    )
    # Store median and axis as key for this node ...
    node.key = (median,axis)

    # ... and continue recursively:
    axis = (axis + 1) % dimension
    # node.append_left(kd_subtree(cloud, subset=left_block, axis=axis))
    node.left = kd_subtree(cloud, subset=left_block, axis=axis)
    if node.left is not None:
        node.left.node_type='left'
        node.left.predecessor = node
    # node.append_right(kd_subtree(cloud, subset=right_block, axis=axis))
    node.right = kd_subtree(cloud, subset=right_block, axis=axis)
    if node.right is not None:
        node.right.node_type='right'
        node.right.predecessor=node

    # Finally, return the node (=root of constructed subtree):
    return node
```