Exercise 4

for the lecture

Computational Geometry

Dominik Bendle, Stefan Fritsch, Marcel Rogge and Matthias Tschöpe

December 18, 2018

Exercise 1 (Image Compression using Quadtrees) (1 + 3 + 1 + 1 points)

a) Best Case: We only have zero-entries.
Worst Case: Let $p_1, \cdots, p_k$ be the parents of the leaf nodes. The worst case scenario for data compressibility would be, if each p_i stores only one non-zero-entry.

b)

Algorithm 1 Calculate $\frac{s_{new}}{s_{old}}$ ratio

f(e, x)
Input: edge length e and fraction of non-zero data x.
Ouput: ratio $\frac{s_{new}}{s_{old}}$.

$s_{old} \coloneqq e^2 \cdot 8\text{Byte}$
`size_not_null_data` $\coloneqq x \cdot s_{old}$
`num_not_null_data` $\coloneqq x \cdot e^2$
`tree_size` $\coloneqq$ ggf verbessern
$s_{new} \coloneqq$ `size_not_null_data` $+$ `tree_size`
return $\frac{s_{new}}{s_{old}}$

c)

d) We assume, each string needs 8 Byte. Instead of using the strings "NE", "SE", "SW" and "NW", we can use integer values. Since, we only need four values, the standard int32 which can store $2^{32} - 1$ values is enough. If we assume a boolean value need exact one Bit, then we can do better. We could use four boolean values which decode the values $1, 2, 3$ and 4 as binary value. Further, for the non-zero-entries, we could try to find duplicates and store them only once.

Exercise 2 (kD-tree Construction) **(2 points)**

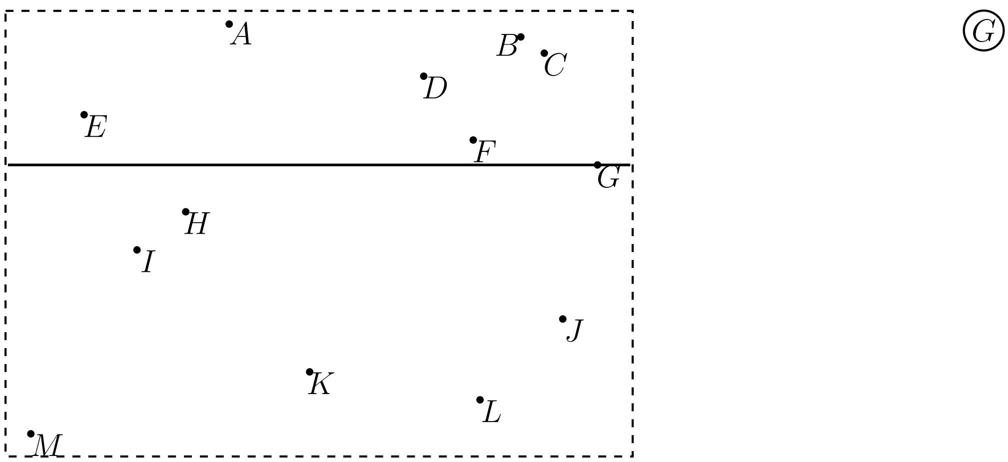
We use two lists x_{sorted} and y_{sorted} and sort them in the x respectively the y direction:

$$\begin{aligned} y_{sorted} &\coloneqq [A, B, C, D, E, F, G, H, I, J, K, L, M] \\ x_{sorted} &\coloneqq [M, E, I, H, A, K, D, F, L, B, C, J, G] \end{aligned} \tag{1}$$

Both list have a length of $len(y_{sorted}) = len(y_{sorted}) = 6$. We shorten the given rules by Ri, where $i \in \{1, 2, 3, 4\}$.

Step 1.

By $R2$ we first have to split in $y-$direction. This means we have to calculate the middle element of the y_{sorted} list, which is G. Hence, G is the first vertex in the $kD-$tree. This yields to the following results:



Step 2.

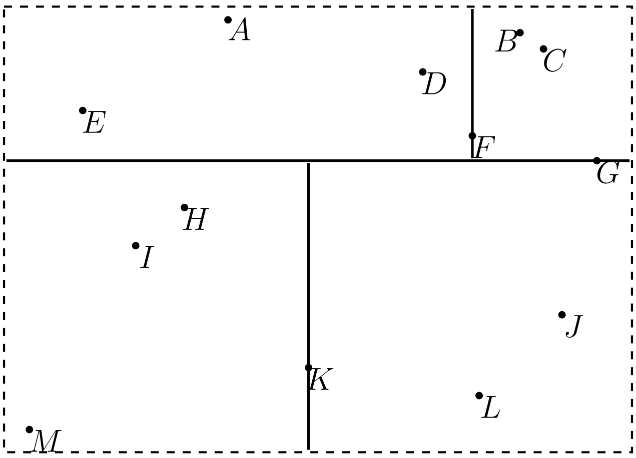
Now we have to split in $x-$direction. Therefore, we take the x_{sorted} list and split them in two new lists x_{above} and x_{below}, where both list are sorted in $x-$direction:

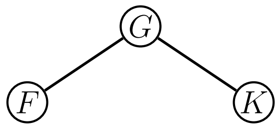
$$\begin{aligned} x_{above} &\coloneqq [E, A, D, F, B, C] \\ x_{below} &\coloneqq [M, I, H, K, J, G] \end{aligned} \tag{2}$$

Since, both lists have even length, we have to use $R4$ to calculate the middle elements:

$$\begin{aligned} n \coloneqq len(x_{above}) = 6 &\implies \text{middle element is } F \\ n \coloneqq len(x_{below}) = 6 &\implies \text{middle element is } K \end{aligned} \tag{3}$$

By $R3$, we have to order the points above, as the left child and the points below as the right child. This gives us:





Step 3.

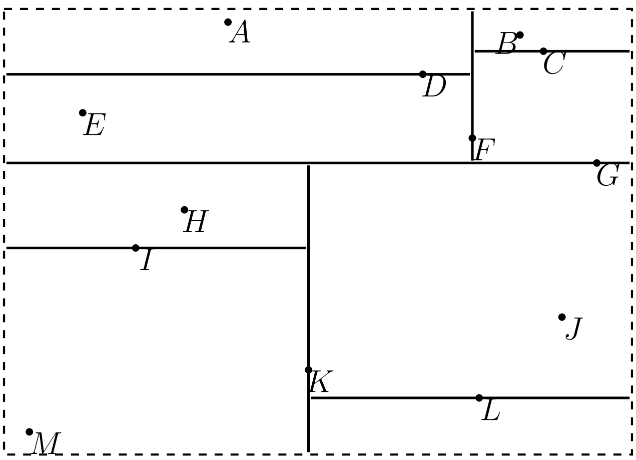
Now, we split in $y-$direction. Hence, we sort x_{above} and x_{below} in $y - direction$ and split each of them into two new lists y_{above_1}, y_{above_2} for the left and right upper part analogously for the part below. Then we get:

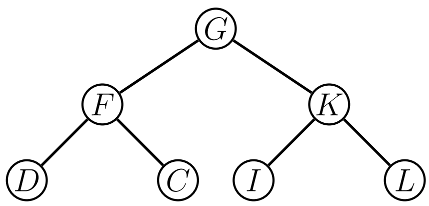
$$\begin{aligned} y_{above_1} &:= [A, D, E] \qquad & y_{above_2} &:= [B, C] \\ y_{below_1} &:= [H, I, M] \qquad & y_{below_2} &:= [J, L] \end{aligned} \tag{4}$$

For y_{above_2} and y_{below_2} we have to use $R4$ to calculate the middle elements. Let M be a function which calculates the middle elements for a list, by the given rules, then we get:

$$\begin{aligned} M(y_{above_1}) &= D \qquad & M(y_{above_2}) &= C \\ M(y_{below_1}) &= I \qquad & M(y_{below_2}) &= L \end{aligned} \tag{5}$$

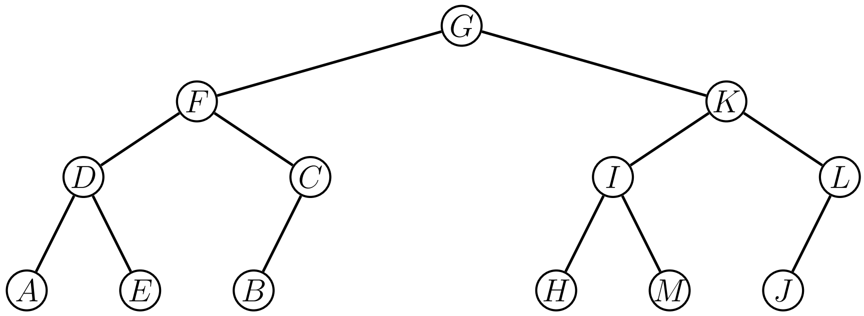
Updating the tree and the lines in the plane yields to:

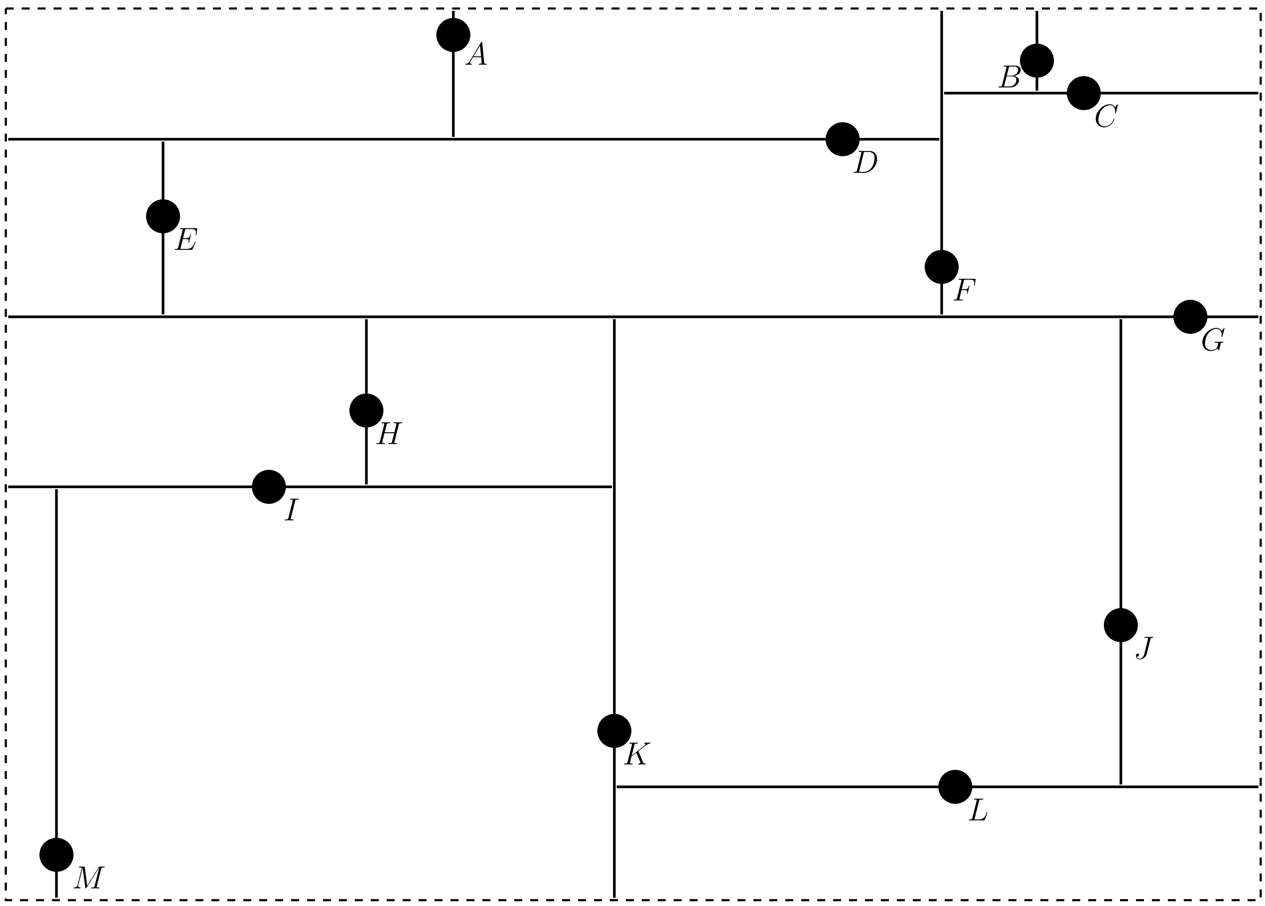




Step 4.

Now, we are splitting in $x-$direction. Since each remaining list only contains at least one element, we are done after insert those elements in the $kD-$tree. Again, by using $R3$ for inserting the new child nodes, we get:





Exercise 3 (Range Search in a kD-tree) (5 points)

Exercise 4 (Nearest Neighbor Search in a kD Tree) (4 points)