Exercise 5

for the lecture

Computational Geometry

Dominik Bendle, Stefan Fritsch, Marcel Rogge and Matthias Tschöpe

January 17, 2019

Exercise 1 (Nearest Neighbor Search in a Quadtree) (4 points)

Data structures:

```
class quadTree:
    def __init__(self, bound=None, ne=None, se=None, sw=None, nw=None):
        self.bound = bound # also include bounding box
        self.ne = ne
        self.se = se
        self.sw = sw
        self.nw = nw

class leaf:
    def __init__(self, val=None, bound=None):
        self.val = val
        self.bound = bound
```

Building the quadtree from a list:

```
def createQuadTree(X, B=None):
    # g_MaxPts is a global variable
    if len(X) <= g_MaxPts:
        return leaf(X,B)

    # we will usually not do this here
    if not B:
        B = ( (min([x[0] for x in X]), max(x[0] for x in X))
            , (min([x[1] for x in X]), max(x[1] for x in X))
            )

    xsplit = (B[0][0]+B[0][1])/2
    ysplit = (B[1][0]+B[1][1])/2

    ne = []
    se = []
    sw = []
    nw = []

    for x in X:
        ise = x[0] >= xsplit
        isn = x[1] >= ysplit
        if isn and ise:
            ne.append(x)
        if isn and not ise:
            nw.append(x)
        if not isn and ise:
            se.append(x)
        if not isn and not ise:
            sw.append(x)

    # descend into the four subregions
    tNE = createQuadTree(ne, ((xsplit, B[0][1]), (ysplit, B[1][1])))
    tSE = createQuadTree(se, ((xsplit, B[0][1]), (B[1][0], ysplit)))
    tSW = createQuadTree(sw, ((B[0][0], xsplit), (B[1][0], ysplit)))
    tNW = createQuadTree(nw, ((B[0][0], xsplit), (ysplit, B[1][1])))
    return quadTree(bound=B, ne=tNE, se=tSE, sw=tSW, nw=tNW)
```

Finally, finding the nearest neighbor of a given point x in the tree T:

```
def findNearest(x, T, dist=np.inf, best=None):
    # check if dist radius intersects current region
    # if any of the inequalities hold, then there is nothing to do
    B = T.bound
    if x[0]+dist < B[0][0] \
            or x[0]-dist > B[0][1] \
            or x[1]+dist < B[1][0] \
            or x[1]-dist > B[1][1]:
                return (best, dist)

    if isinstance(T, quadTree):
        # search recursively in the children
        for t in [T.ne, T.se, T.sw, T.nw]:
            (best, dist) = findNearest(x, t, dist, best)

    if isinstance(T, leaf):
        for v in T.val:
            d = dist2d(x, v)
            if d < dist:
                best = v
                dist = d

    return (best, dist)
```

Sample command line output: A visualization is given in Figure 1, the black rectangles indicate all regions that are actually entered.

```
Time per Method:
naive:               5.108457999085658 ms
kd-tree:             0.12103399967600126 ms
quadtree:            0.2925829994637752 ms
Speedup (to naive):  17.459859282487596
Speedup (to kd):     0.4136740682056837
Both methods found same nearest point!
```

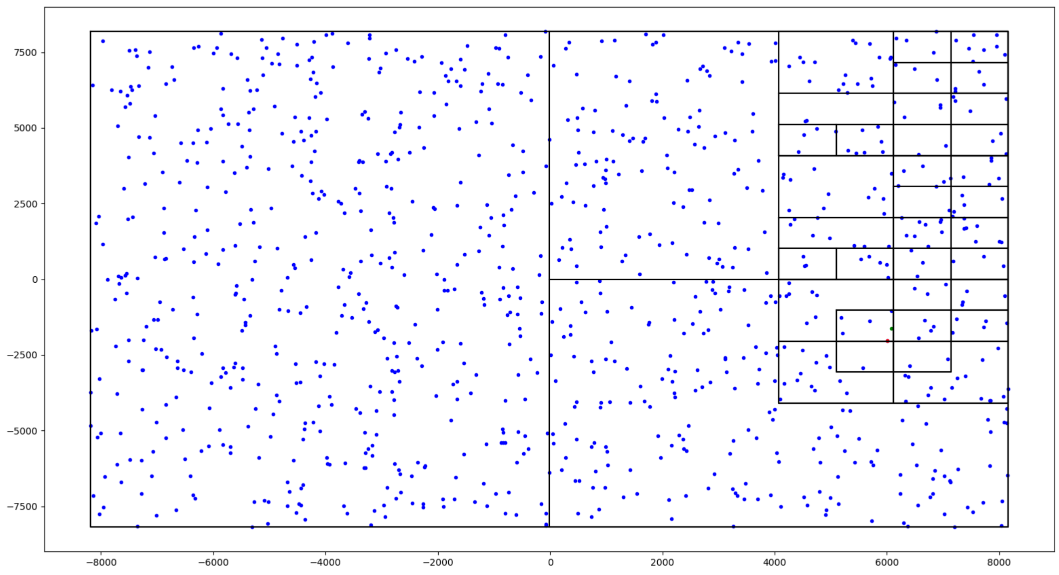
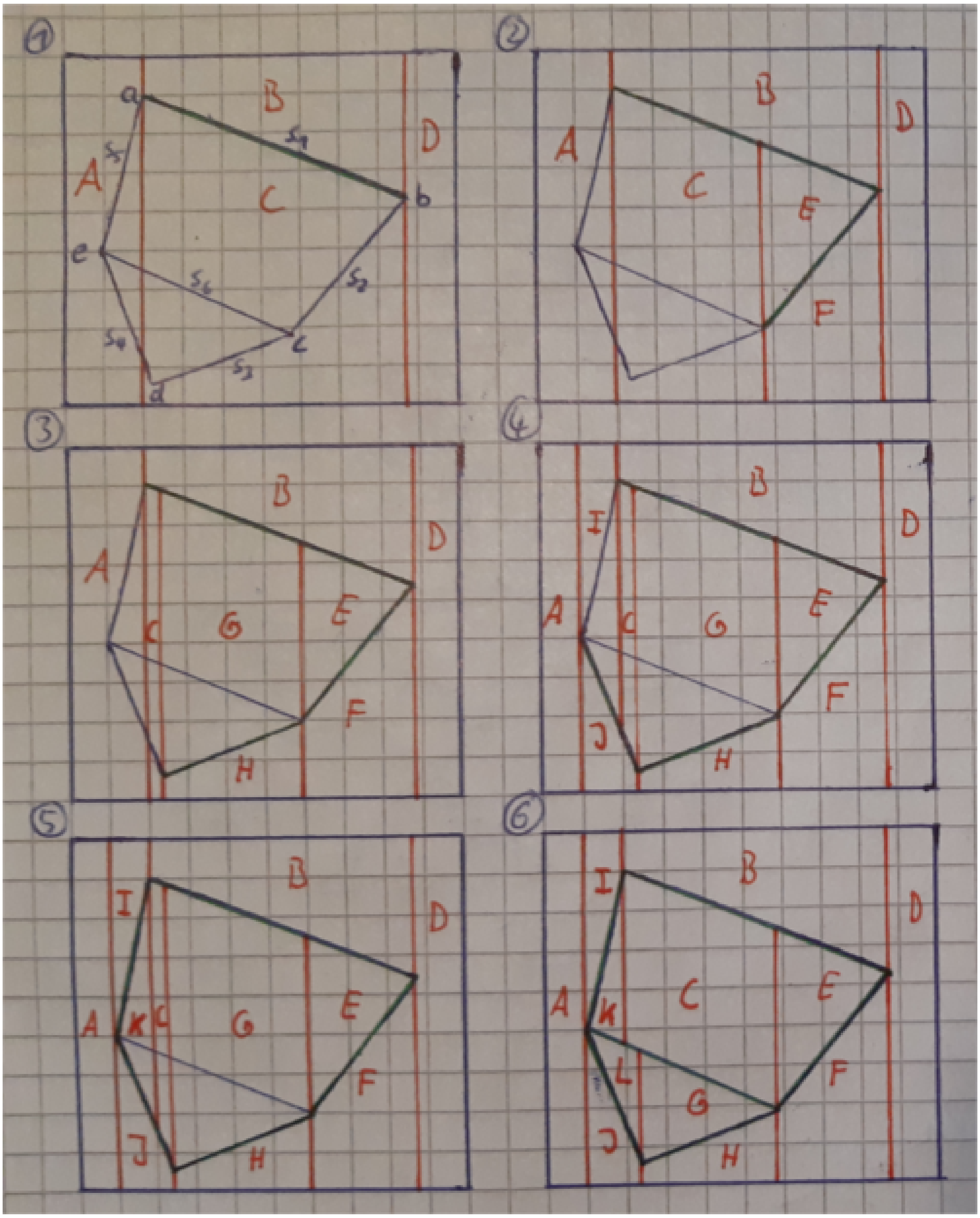
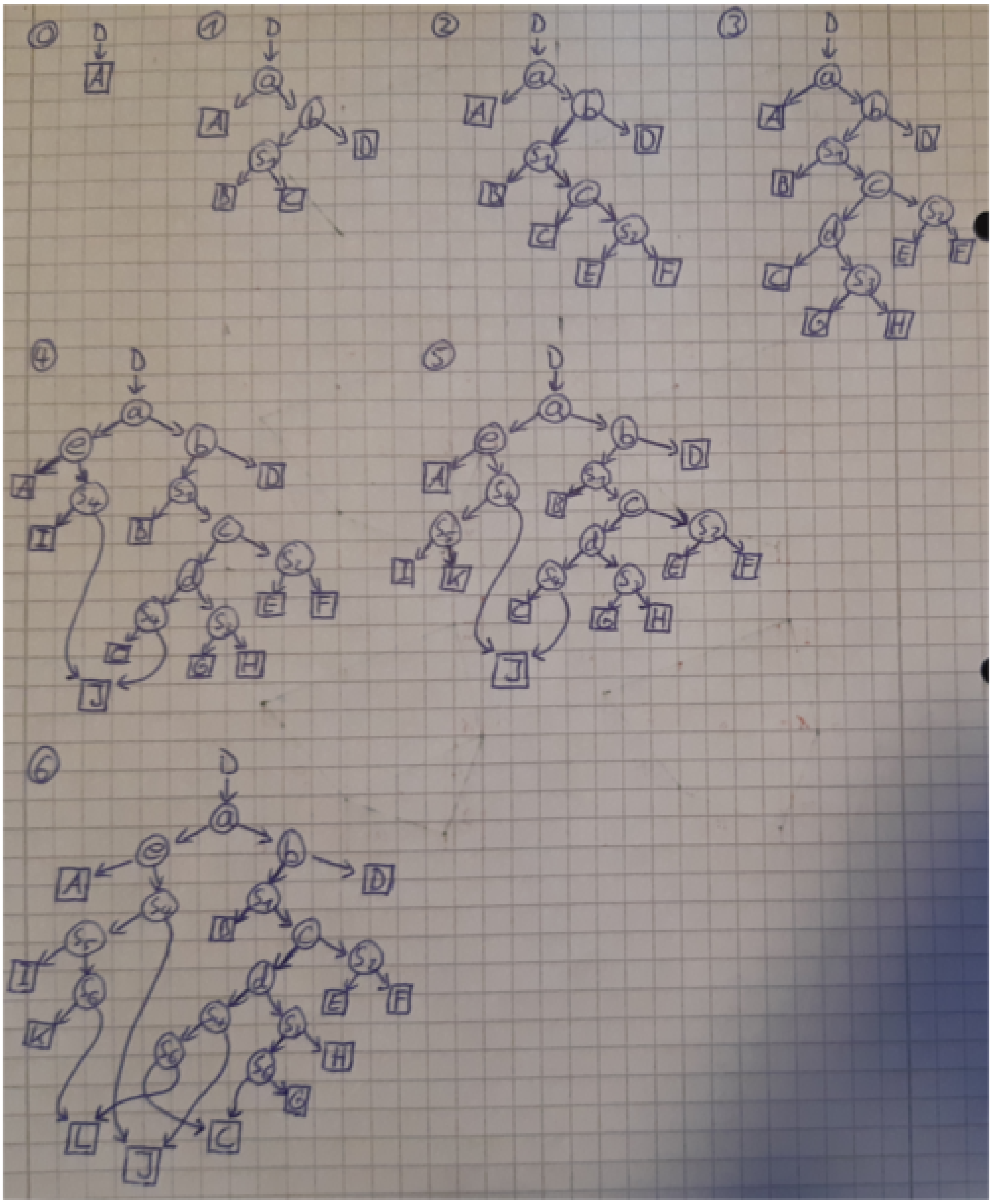


Figure 1: Nearest Neighbor Search

Exercise 3 (Trapezoidal Map and Search Structure from a Planar Subdivision) (5 points)




0
1
2
3
4
5
6
D
a
b
c
d
e
S_1
S_2
S_3
S_4
S_5
S_6
A
B
C
D
E
F
G
H
I
J
K
L