Homework 5

Due 10/16/19

October 9, 3019

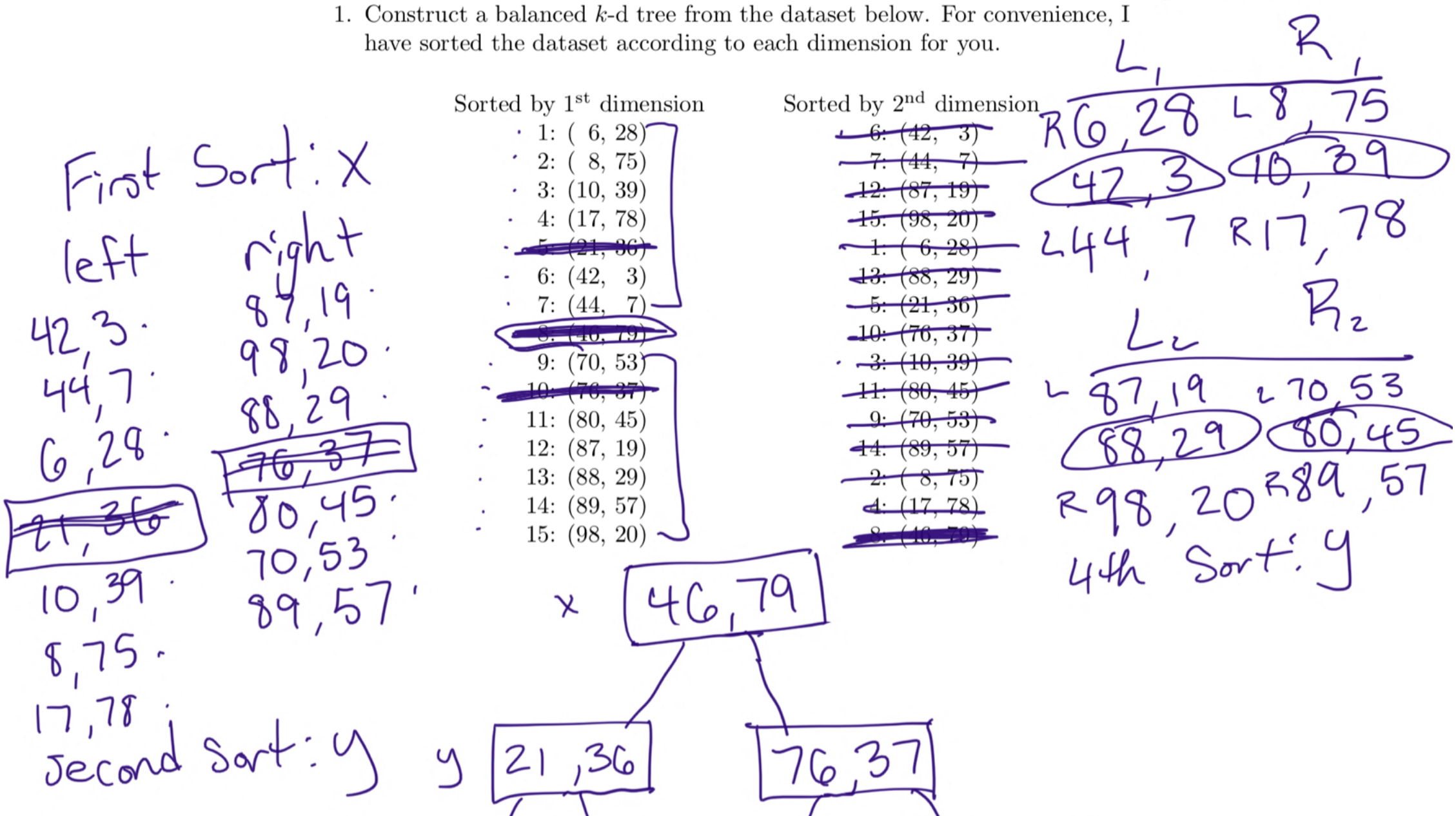
1. Construct a balanced k-d tree from the dataset below. For convenience, I have sorted the dataset according to each dimension for you.

Sorted by 1st dimension

- 1: (6, 28)
- 2: (8, 75)
- 3: (10, 39)
- 4: (17, 78)
- ~~5: (21, 36)~~
- 6: (42, 3)
- 7: (44, 7)
- ~~8: (46, 79)~~
- 9: (70, 53)
- ~~10: (76, 37)~~
- 11: (80, 45)
- 12: (87, 19)
- 13: (88, 29)
- 14: (89, 57)
- 15: (98, 20)

Sorted by 2nd dimension

- ~~6: (42, 3)~~
- ~~7: (44, 7)~~
- ~~12: (87, 19)~~
- ~~15: (98, 20)~~
- ~~1: (6, 28)~~
- ~~13: (88, 29)~~
- ~~5: (21, 36)~~
- ~~10: (76, 37)~~
- ~~3: (10, 39)~~
- ~~11: (80, 45)~~
- ~~9: (70, 53)~~
- ~~14: (89, 57)~~
- ~~2: (8, 75)~~
- ~~4: (17, 78)~~
- ~~8: (46, 79)~~



2. Suppose we wish to modify the basic BST data structure so that it can return the minimum in constant time. In order to return the min in constant time, we add a field *min* to the BST, which points to the node containing the minimum value. Give pseudocode for a modified version of the BST.Delete operation that updates the *min* field as necessary without increasing the asymptotic complexity of BST.Delete. A reference implementation of BST.Delete appears below.

```
1  Algorithm: BST.Delete(node)
2  if node has two children then
3  |  swapnode = node.right
4  |  while swapnode has a left child do
5  |  |  swapnode = swapnode.left
6  |  end
7  |  Swap node's parent and children links with swapnode
8  |  if node is the BST root then
9  |  |  Set root to be swapnode
10 |  end
11 end
12 if node has no children then
13 |  if node is the root then
14 |  |  Set root to be NIL
15 |  else
16 |  |  Set node.parent's child to be NIL
17 |  end
18 else
   |  /* node must have one child                                    */
19 |  if node is the root then
20 |  |  Set root to be node's child
21 |  else
22 |  |  Set node.parent's child to be node's child
23 |  end
24 |  Set node's child's parent to be node.parent
25 end
```

```
if min == node
| if min has no childern
| | then min = minparent
else
| min.parent.left = min.right
| min.right.parent = min.parent
| min = min.right
| while min.left != NIL
| |  min = min.left
```