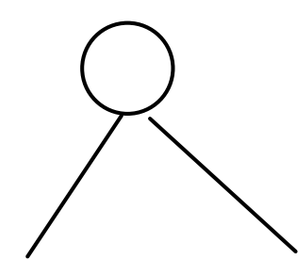
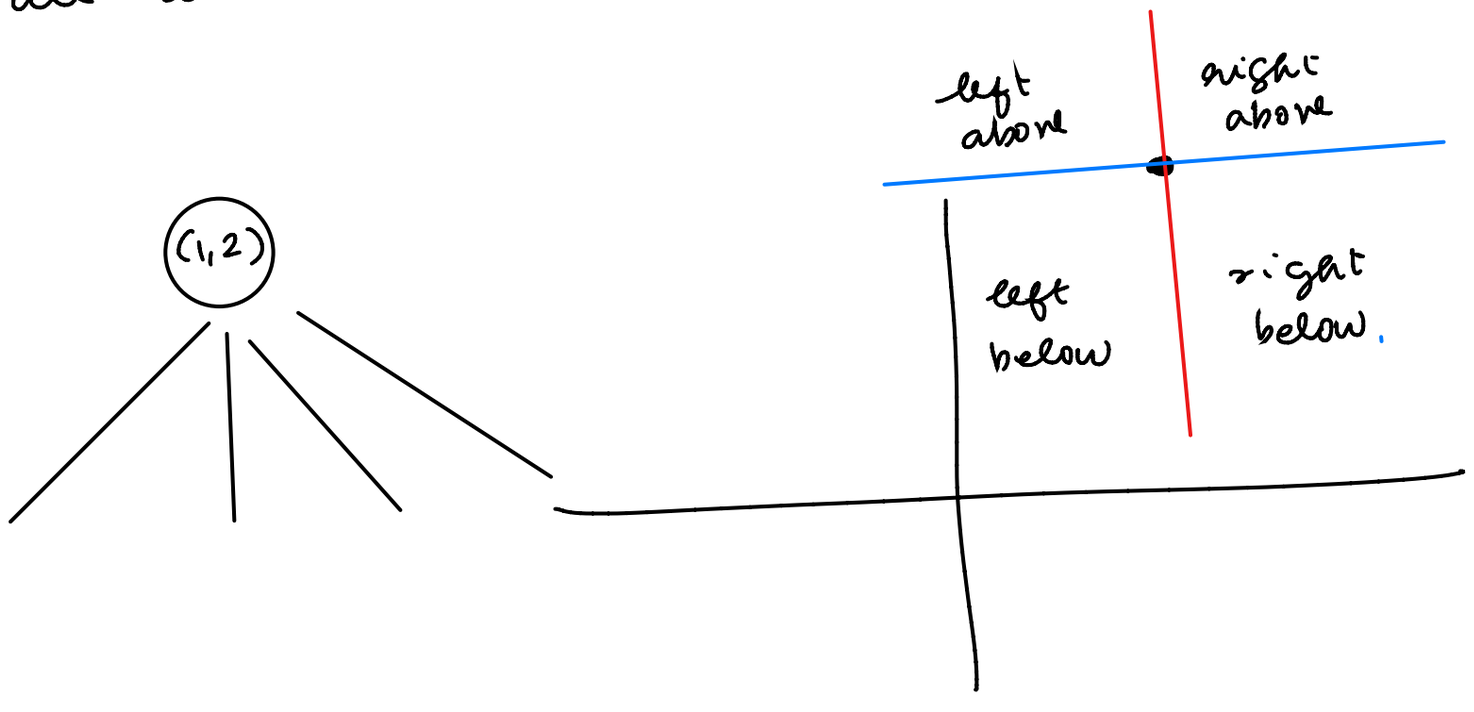
kD- Trees

- 2d or 3d tree 'kd tree
- At each level of the tree, we compare with a particular dim
  - Typically one circulates dimensions (round robin)
    - cut dimension of that level



Only 2 children for each node

- If all dimensions are cut we get 2^d children



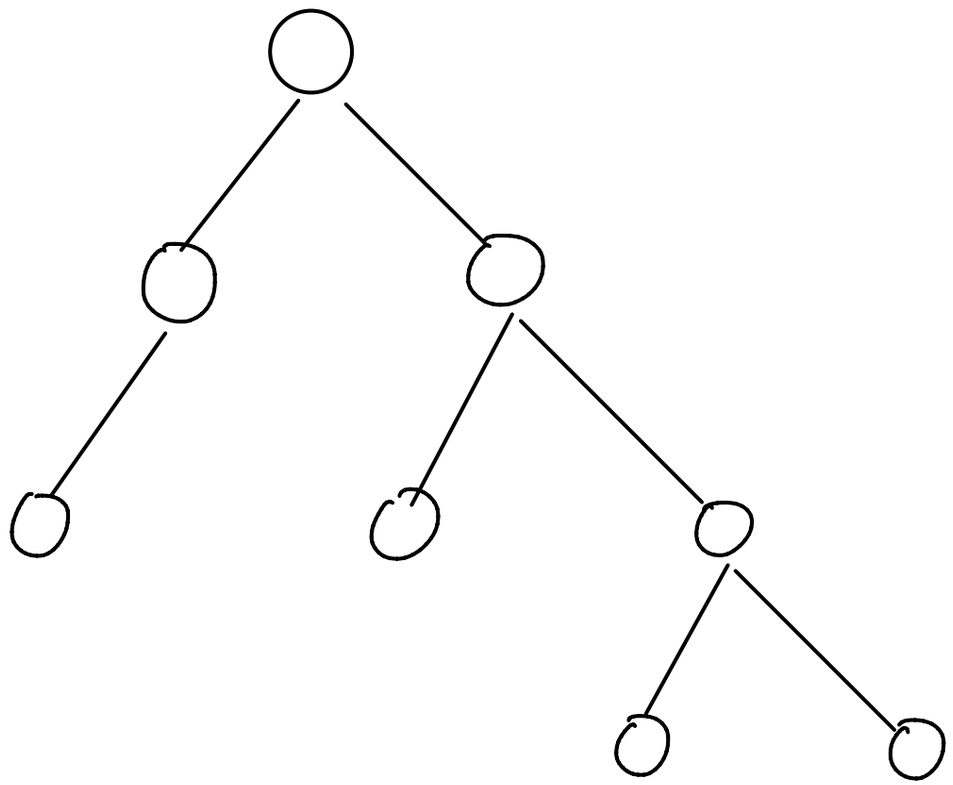
cut dimension

x

y

x

y



Property to be maintained

If coordinate of cut dimension is < coordinate of cut dimension of data at node

⇓

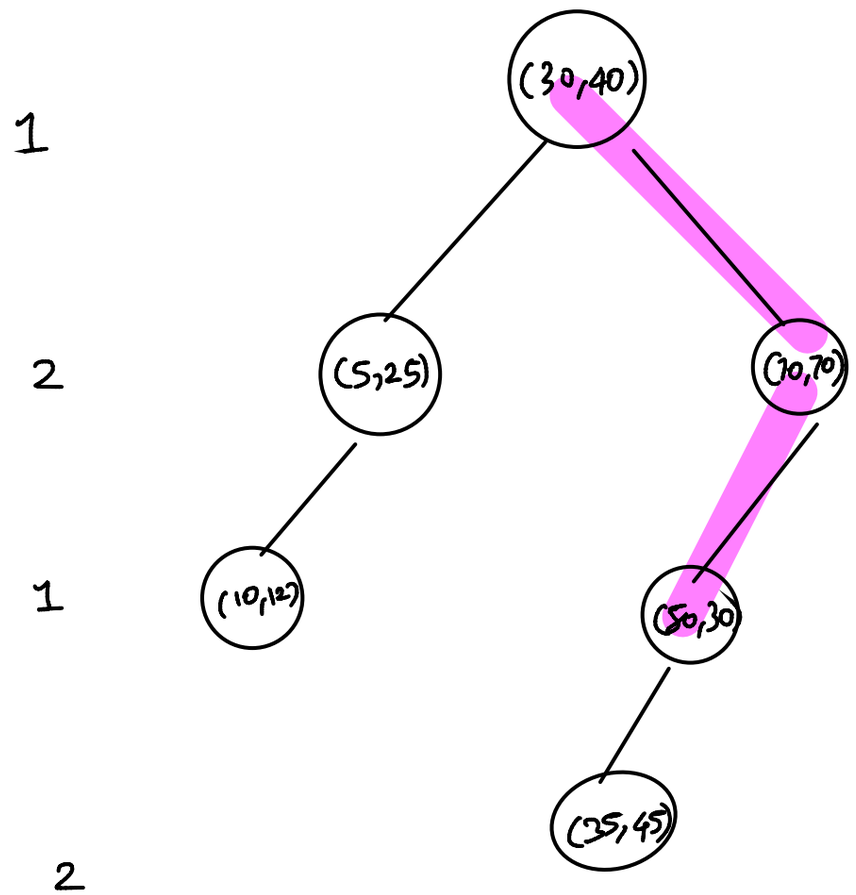
directed to left sub-tree

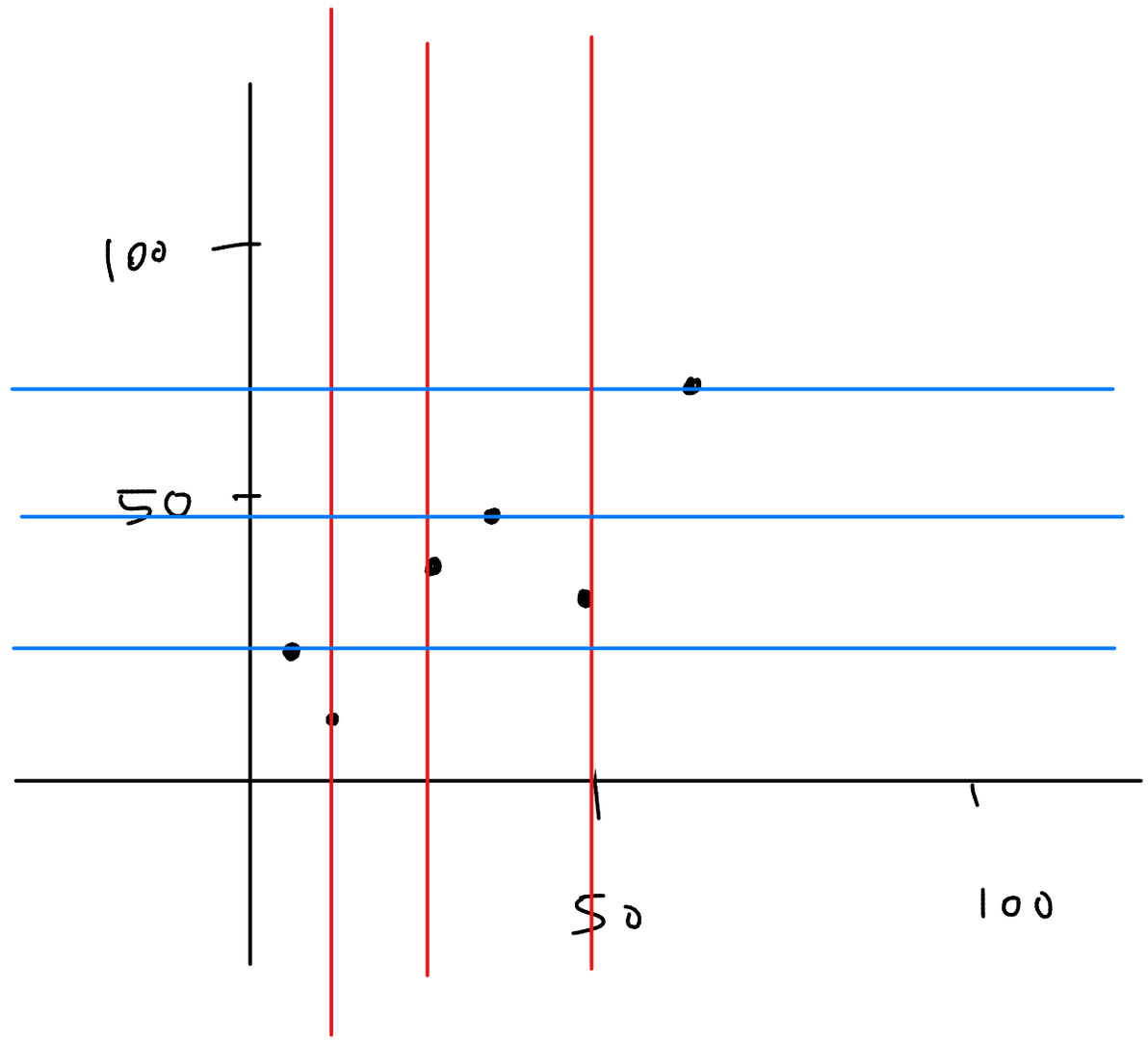
Data: (30,40), (5,25), (10,12), (70,70), (50,30), (35,45)

5 < 30

10 < 30 12 < 25

cut Dimensions





```
insert ( x, node t, cut-dim )

    if    t = NULL

          t = new node(x)

    else  if  (x == t.data)

          say it is duplicate

    else  if  (x[cut.dim] < t.data[cut-dim])

          insert (x, t.left, (cut-dim + 1) % total-dim)

    else

          insert (x, t.right, (cut-dim + 1) % total-dim)
```

Since there no compelte ordering

Find Min (dim)

* Reeursively traverse the tree

* at the current level .

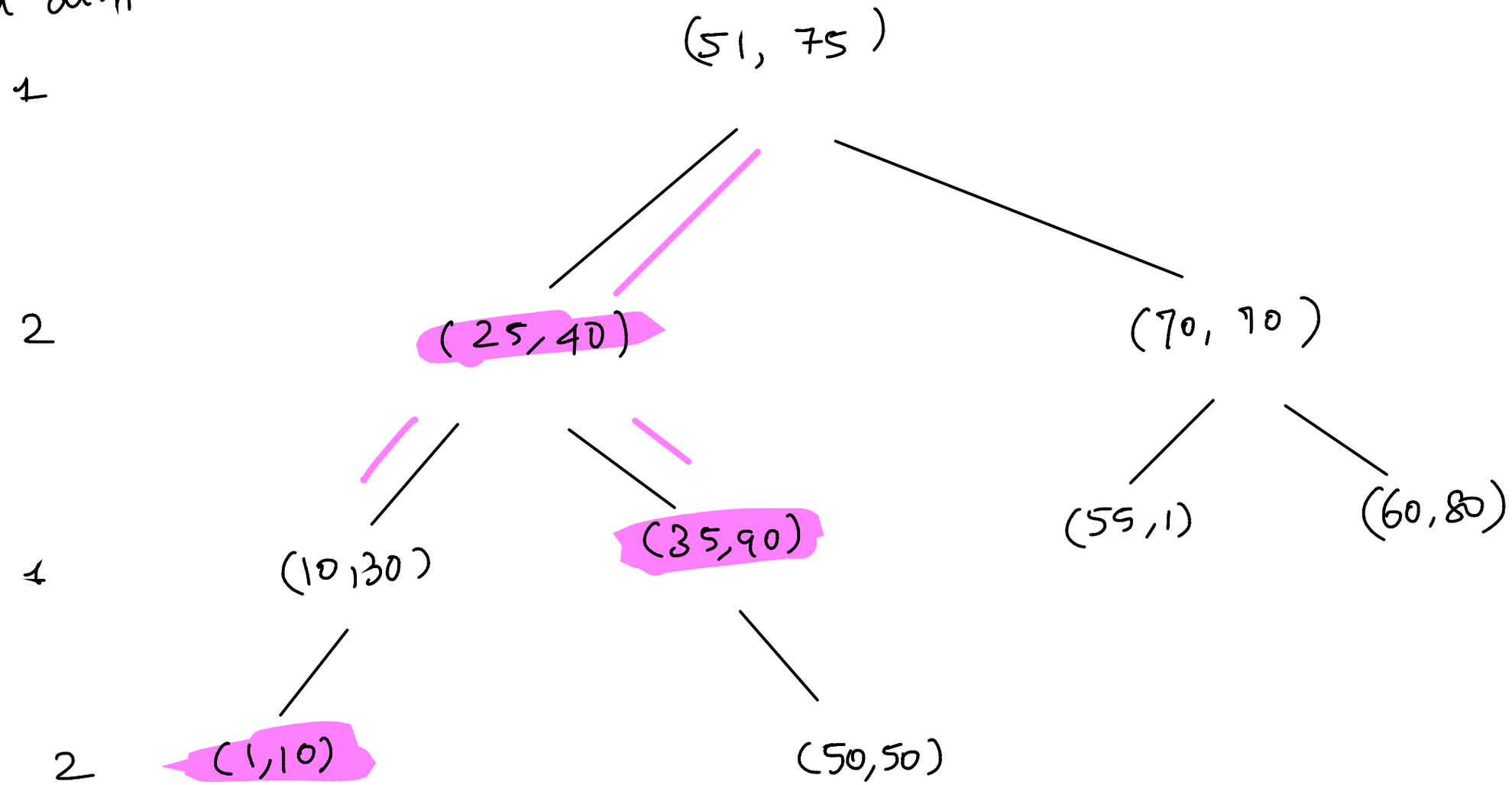
if cur. dim = dim

we can choose left

else

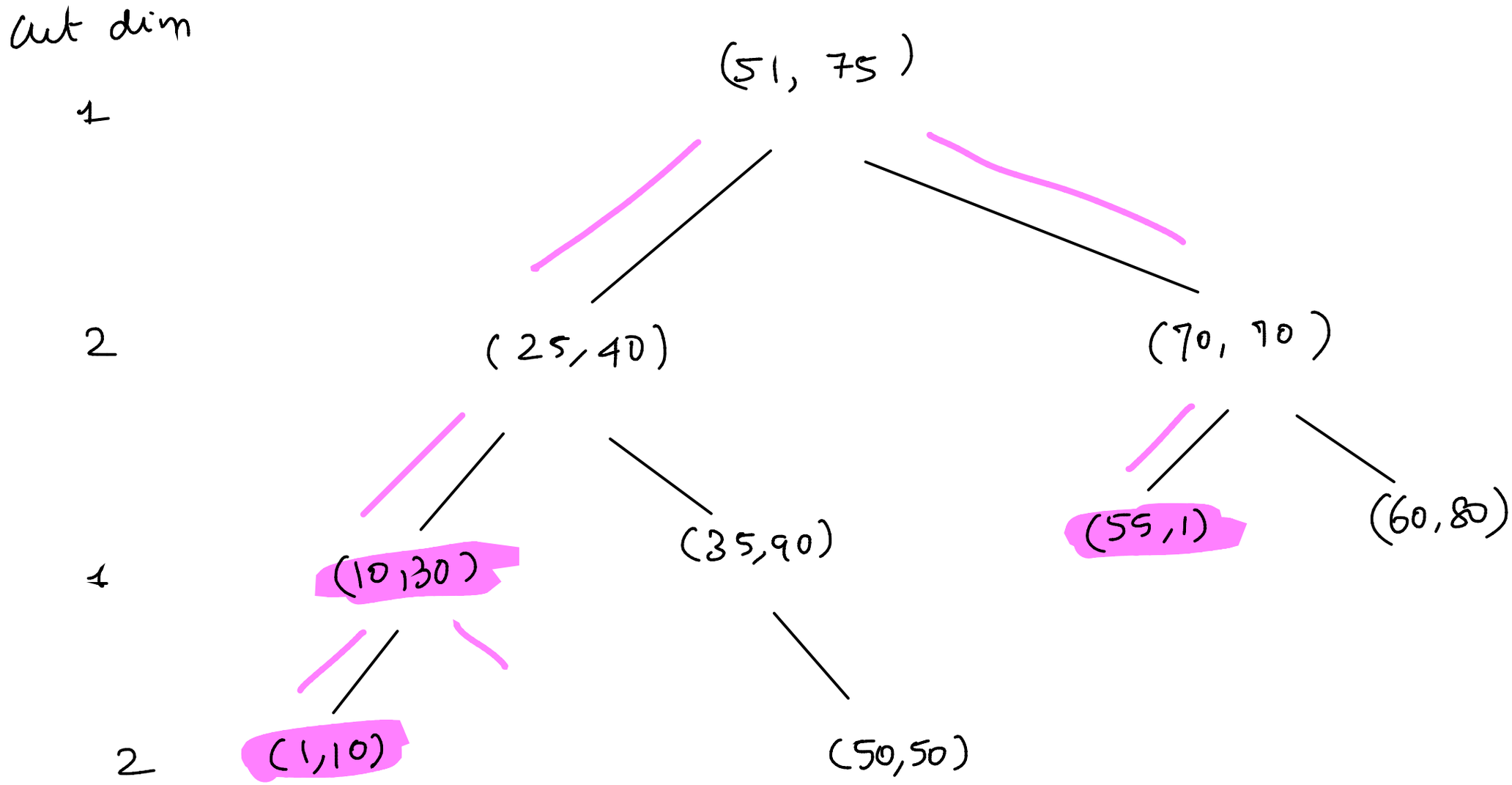
we have to search both sides

cut dim



Find min (dim: 1)

min {1, 25, 35}



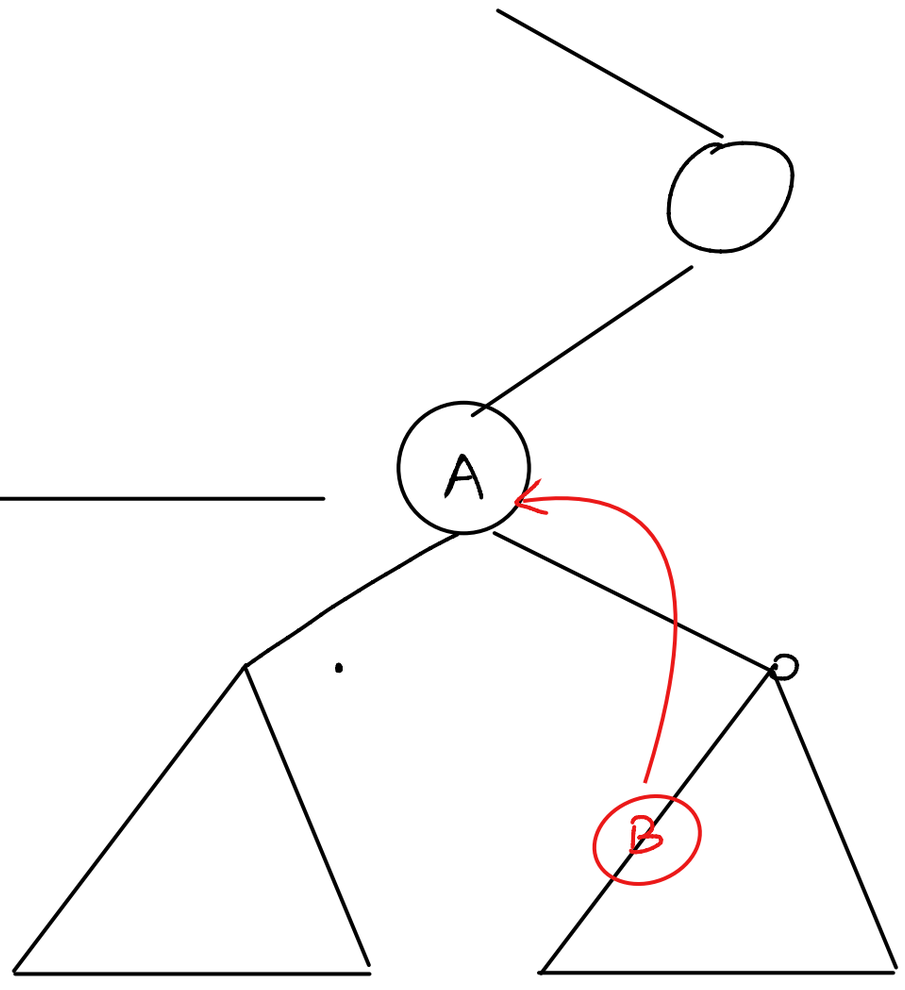
Find min (dim : 2)

min { 10, 1, 30 }

```
Find min ( node t, dim, cut_dim)

    if   cd == dim

         if  t.left == NULL :  return t.data

         else  return FindMin ( t.left, dim, (cut_dim+1) % total_dim)


    else
    return   min ( t.data,
                   Find min ( t.left, dim, (cut_dim +1) % total_dim),
                   Find min ( t.right, dim, (cut_dim +1) % total_dim)
                   )
```

Say we want to delete A, if right subtree is not empty

Find min (t. right, cut. dim at level A)



Cut- dim Of level

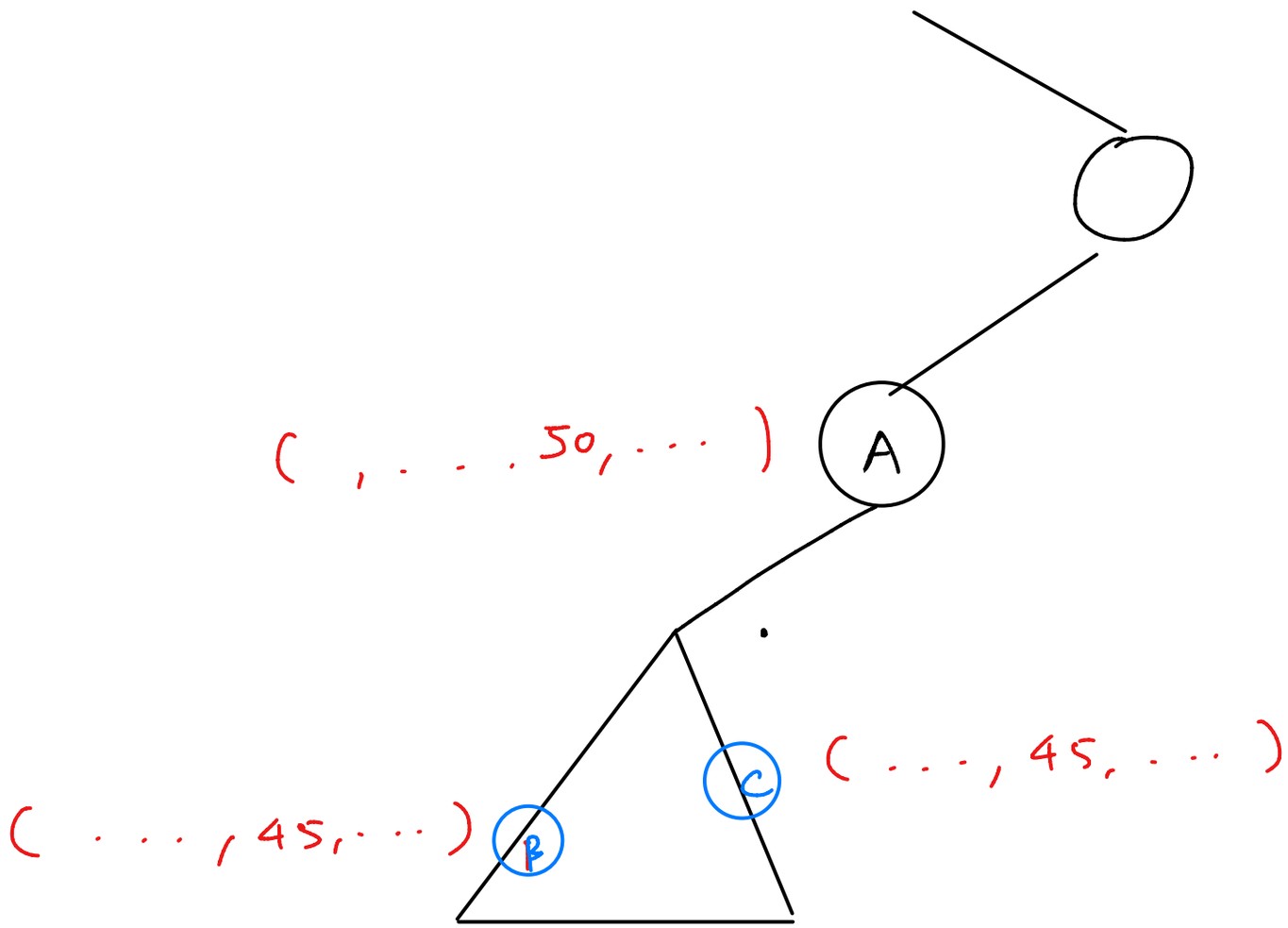
A is removed

B replaces A

Now B is empty

This initiates a series of deletes until a leaf gets deleted

Say the right subtree is empty, search for the maximum of the left subtree



Say break the tie somehow, say B was chosen to replace, we violate the < Property

- make the left sub-tree the right sub-tree
- Find min of the right subtree