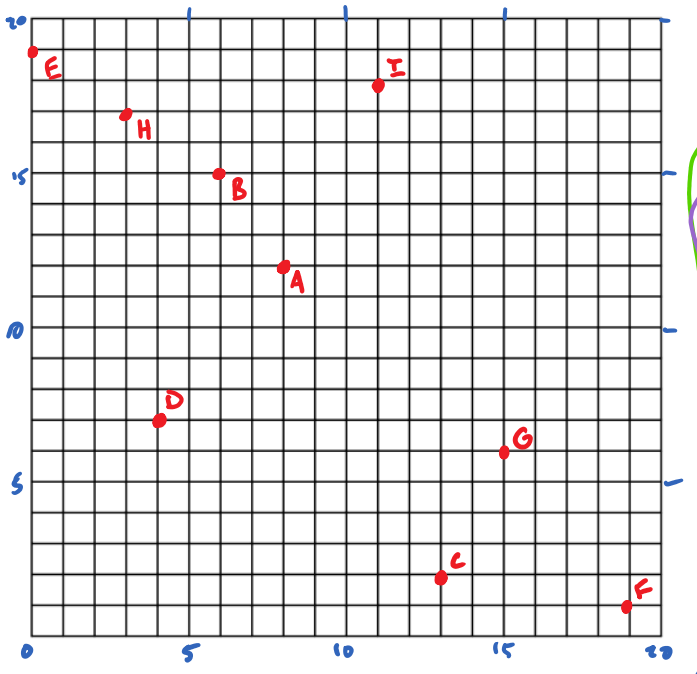
Building a Balanced kd-Tree




```
kdtree_create(points)
    sort_x <- points sorted by x (lon)
    sort_y <- points sorted by y (lat)

    root <- kdtree_create_helper(sort_x, sort_y)

maintain order in other dim
kdtree_create_helper(sort_cut, sort_other)
    if sort_cut has 0 points, return NULL
    node <- new node containing middle point from sort_cut
    cut_left <- 1st half of sort_cut
    cur_right <- 2nd half of sort_cut
    other_left <- pts from sort_other to left of middle
    other_right <- pts from sort_other to right of middle
    node->left <- kdtree_create_helper(other_left, cut_left)
    node->right <- kdtree_create_helper(other_right, cut_right)
    return node
```

$O(n \log n)$

do this all in $\Theta(n)$ time

switching the order does the cut dimension alternation

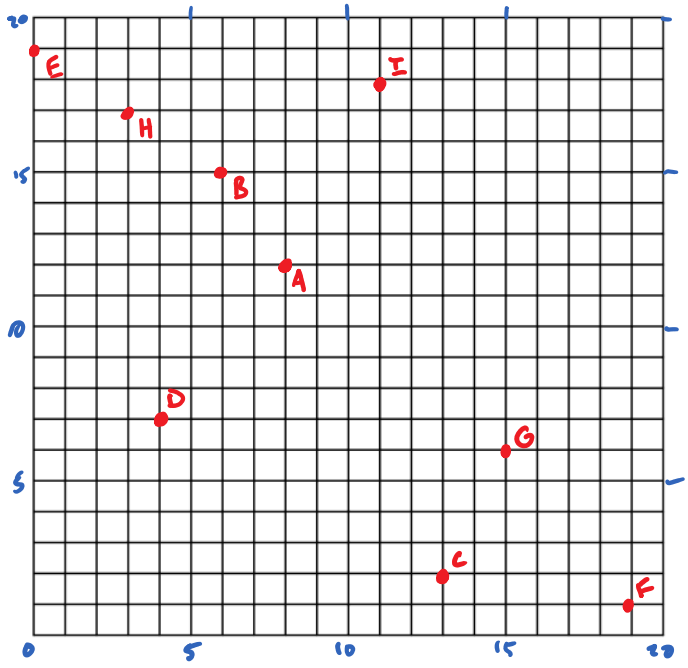
$T(n) = 2T(\frac{n}{2}) + \Theta(n)$

$T(n) = \Theta(n \log n)$

$\Theta(n)$ work aside from the recursive calls

this is the same as for mergesort

2 recursive calls on list of size $\sim \frac{1}{2}$ the previous size



x: E H D B A I C G F

y: F C G D A B H I E