6.1 We used looped double sided two linked lists as one object. Each container have next and prev for both x and y.

6.2 Runtime:

1. O(1) because its only initializing the field.
2. At the worst case its O(2n)=O(n) because it will go in x through everything and put it at the end and in y through everything and put it at the end.
3. At worst case scenario we will iterate over the points twice in the help function, thus making the runtime complexity O(n)
4. At worst case scenario we will iterate over the points twice in the help function, thus making the runtime complexity O(n)
5. O(1) cause its using data from functions or fields that doesnt go through more than 1 element.
6. O(|A|) cause its going from the top and bottom to every point out of the zone and changing the nexts and pervs.
7. O(1) cause you take fields of object fields so its all O(1)
8. O(n/2)=O(n) cause you go through half of the elements in the container.

6.3 SPLIT:

Split will return an array of datastructures when in spot 0 there will be the points smaller then value and in place 1 the ones that bigger. Youll be able to use each part of the array as you use datastructure objects and ofcurse you could save them as two seperet objects if needed.

Split part of O(4|c|)=O(|c|)

Pseudo:

```
Container[] split(int value, Boolean axis)
        datastracture[] ans <- new datastracture [2];
        Container low;
        Container high;
        If(axis)
                Low <- xhead;
                High <- xhead.getprevx();
                Ans[0].add(xhead);
                Ans[1].add(xhead.getprevx());
```

```
        While(low.getnextx().getx() < value & high.getprevx().getx() > value)
                Ans[0].add(low.getnextx());
                Ans[1].add(high.getprevx());
                Low <- low.getnextx();
                High <- high.getnextx();
        If(low.getnextx().getx() < value)
                Ans[1] <- narrow(value, xhead.getprevx().getx(),true);
        Else
                Ans[0] <- narrow(xhead.getx(),value,true);
else
        Low <- yhead;
        High <- yhead.getprevy();
        Ans[0].add(yhead);
        Ans[1].add(yhead.getprevy());
        While(low.getnexty().gety() < value & high.getprevy().gety() > value)
                Ans[0].add(low.getnexty());
                Ans[1].add(high.getprevy());
                Low <- low.getnexty();
                High <- high.getnexty();
        If(low.getnexty().gety() < value)
                Ans[1] <- narrow(value, yhead.getprevy().gety(),true);
        Else
                Ans[0] <- narrow(yhead.gety(),value,true);
```

6.4 our method calls nearestPairInStrip but with a “B” (number of points) of size n, making its runtime of n*log(n) cause its using that function 6 times which can be ignored cause its still blogb

And that’s why its O(nLog(n))

6.5 our implementation of split does the idea in question. Its going through the points and finds the place of the value we are looking for and then making two datastructures one with every point to this value and one with every point from that value. Its doing it by going through the points from

both sides until finding the value and the one side is finished and the other gets every other point left from narrowing the datastrucure. And you can make it as the value is of the median point.