num

```
find_closest_pair(points_arr, n)        # n = length of points_arr
    if n ≤ 3
        find_min_dist(points_arr, n)
    else
        L = find_L(points_arr, n)       # make new left and right, size n/2
        δ1 = find_closest_pair(points_arr_left, n/2)
        δ2 = find_closest_pair(points_arr_right, n/2)
        δ = min(δ1, δ2)
```

Naive

```
        for each point in points_arr
            if |L - point.x| < δ
                Add point to My
```

Slide 9

```
        My = merge_sort(My, My_length, 'y')
        dm = closest_cross_pair(My, δ)
        return dm
```

num

```
find_L(points_arr, n)
    points_arr = merge_sort(points_arr, 'x')
    L = points_arr[n/2]
    return L
```

Enhanced

Presort

```
presorted_by_y = merge_sort[points_arr, 'y']
presorted_by_x = merge_sort[points_arr, 'x']
```

Enhanced

```
for each point in presorted_by_y
    if | L - point.x | < δ
        add point to My
```

num

```
find_min_dist (points_arr, n)        ← will be less than/equal to 3
    for each point in points_arr
        curr_min = √((point.x)² + (point.y)²)
        if curr_min < min
            min = curr_min
    return min
```

arr

```
merge_sort (points_arr, n, char 'y' or 'x')
    if char == 'y'
        sort by y
    else
        sort by x
```

Pseudo code for closest cross pair, merge_sort in slides

void

```
if for every point
        if point.x < L
            put in points_arr_left
        else
            points_arr_right
```