Divide and conquer tutorial

Solving past year lab assignment

Problem statement

You have joined as a trainee in an Institution researching on creating devices to help people with defective eyesight. **You have been assigned to a group which deals with the binary eyesight defect, where a person can only see 2 colors, black and white. This means that when 2 objects have a greater contrast than a particular threshold, then they can be perceived as different.** They are at the initial phases of the project and they want to understand and stress on the difficulties faced by these people in recognizing objects and the loss of depth perception … **CONTINUED**

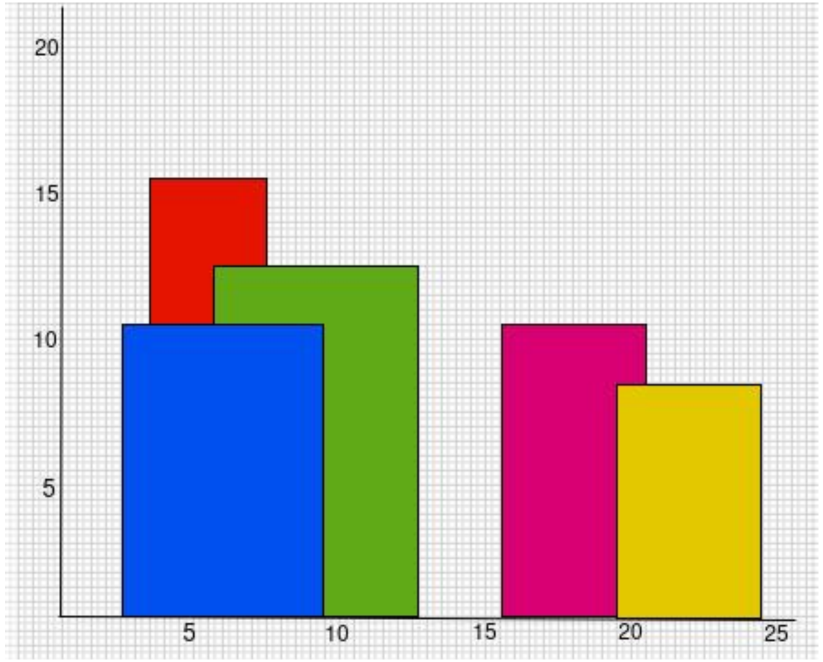
Problem statement

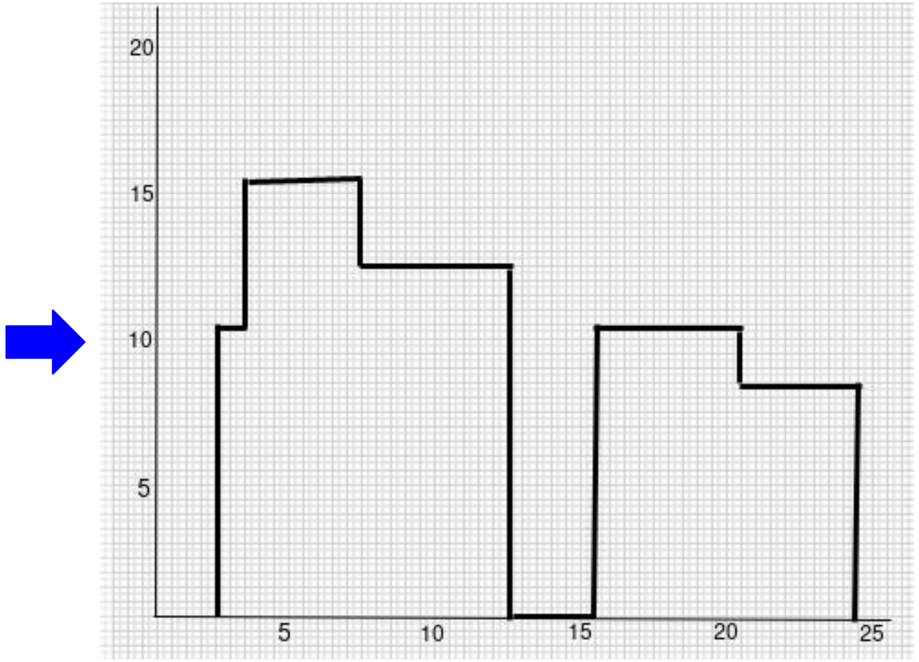
…. Since you are still a trainee you have been tasked to get the perspective of users on a set of rectangular objects kept on a table. **Imagine that you are looking right at it, and you cannot see the top of these objects (this simulates the irrelevance of depth information).** The objects are of various colours and the background is white, and it is known that t**he defective eye will not be able to differentiate between each of the objects but there is enough contrast to differentiate between the objects and the background.**

Problem statement

Devise an algorithm to convert a normal vision to that of a defective vision.

An example

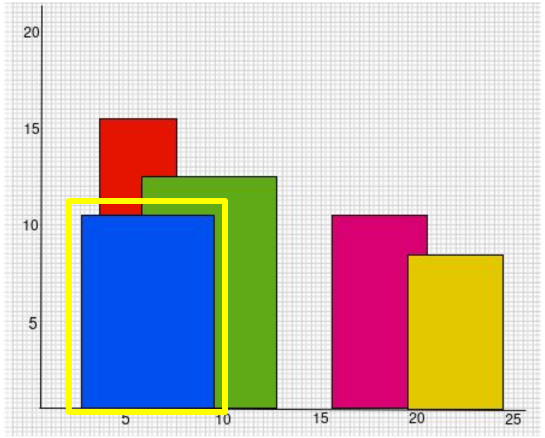




Normal vision

Defective vision

An example

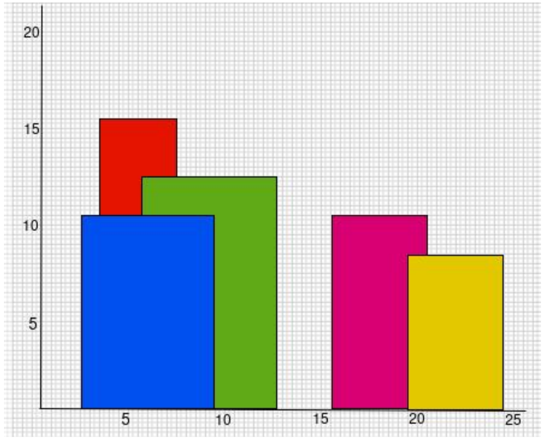


2 - Left boundary
9 - Right boundary
10 - Height

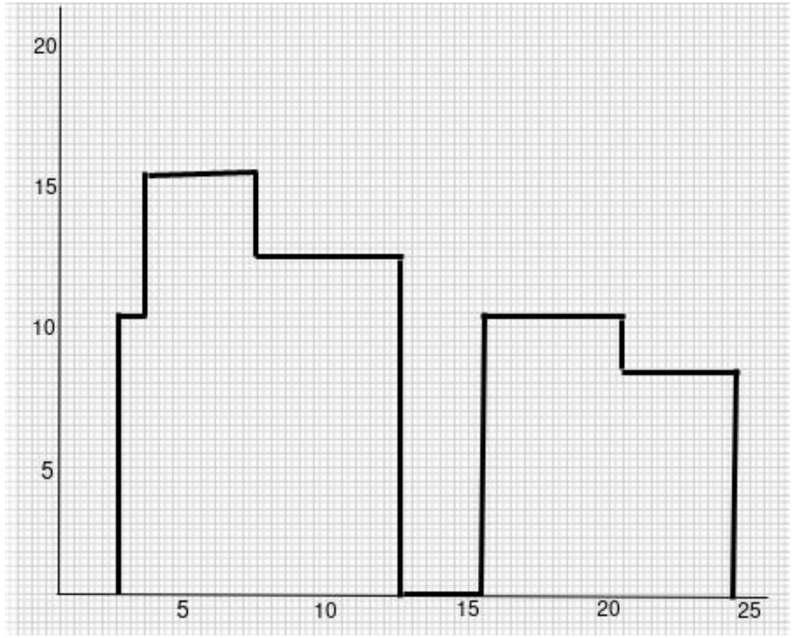
[**[2 9 10]**, [3 7 15], [5 12 12], [15 20 10], [19 24 8]]

Normal vision (input)

An example



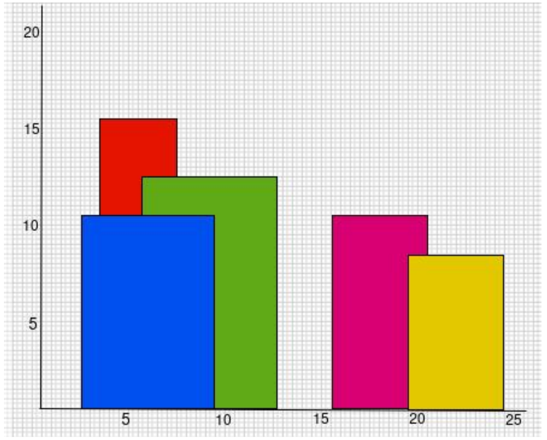
[[2 9 10], [3 7 15], [5 12 12], [15 20 10], [19 24 8]]



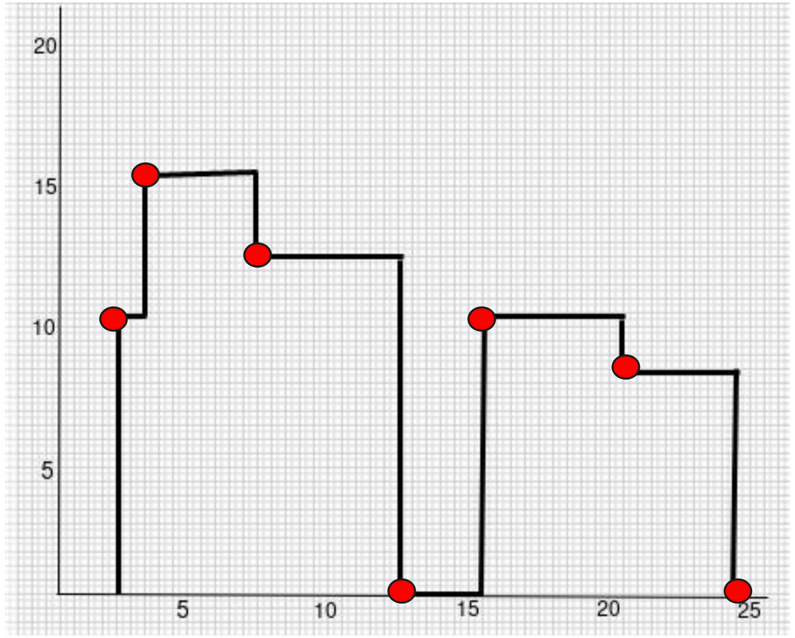
Normal vision (input)

Defective vision (output)

An example



[[2 9 10], [3 7 15], [5 12 12], [15 20 10], [19 24 8]]

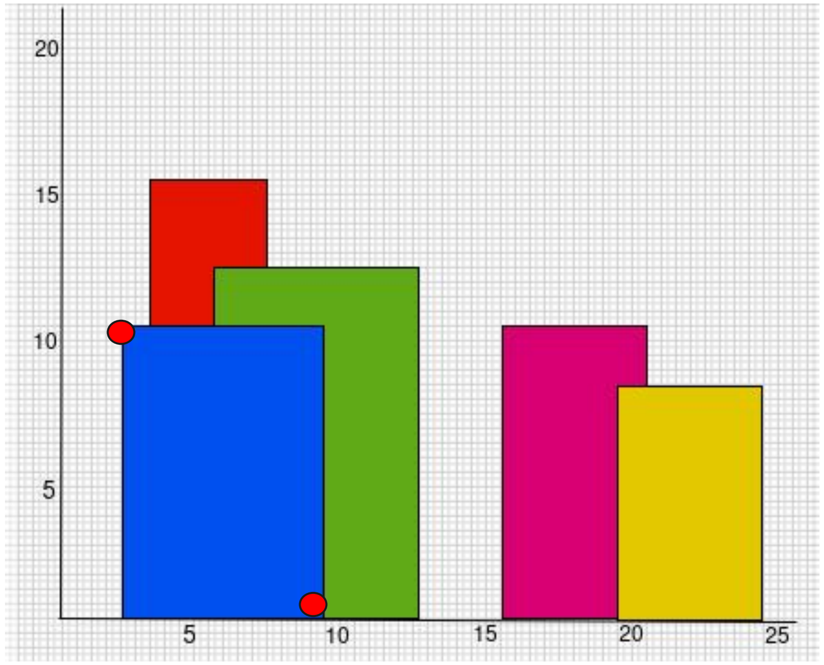


[[2,10],[3,15],[7,12],[12,0],[15,10],[20,8],[24,0]]

Normal vision (input)

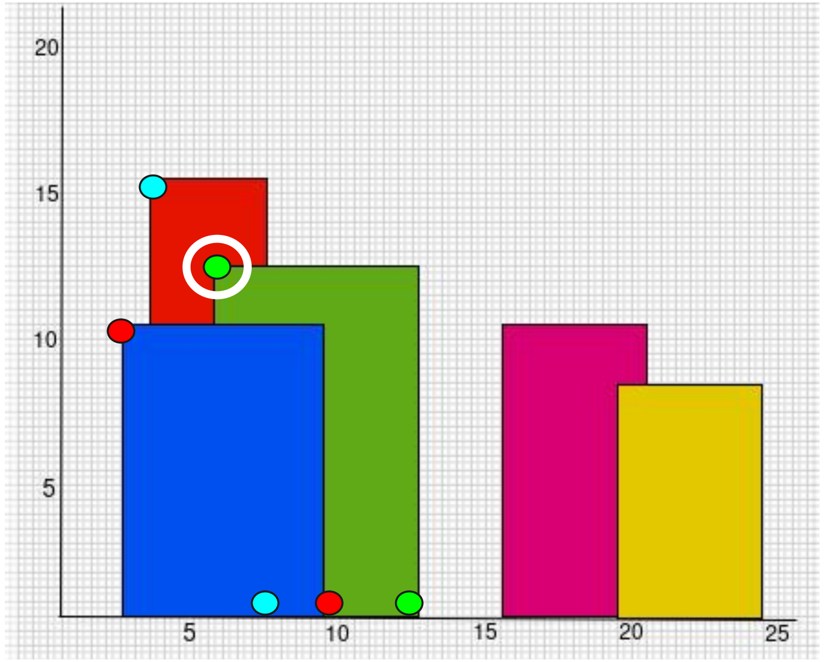
Defective vision (output)

What can be a simple approach ?



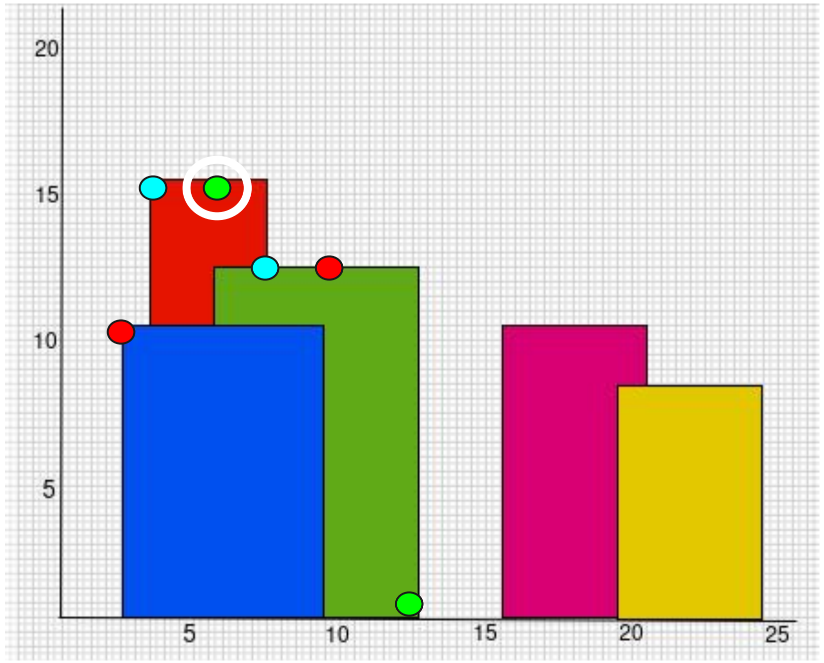
- Mark key points for each block

What can be a simple approach ?



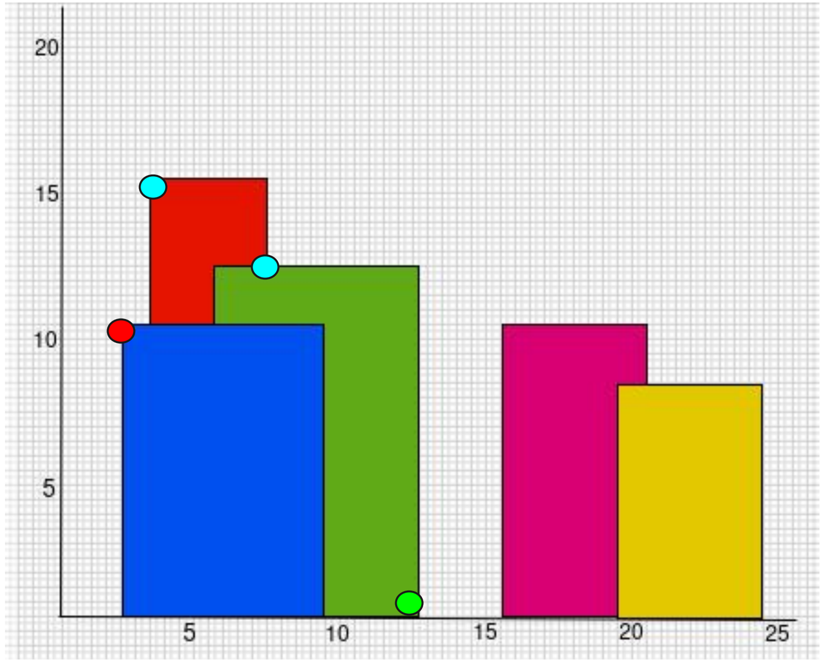
- Mark key points for each block
- For each marked point, if buildings with more height overlap with it, change y to **maximum height of tallest overlapping buildings**

What can be a simple approach ?



- Mark key points for each block
- For each marked point, if buildings with more height overlap with it, change y to **maximum height of tallest overlapping buildings**
- Remove redundant points

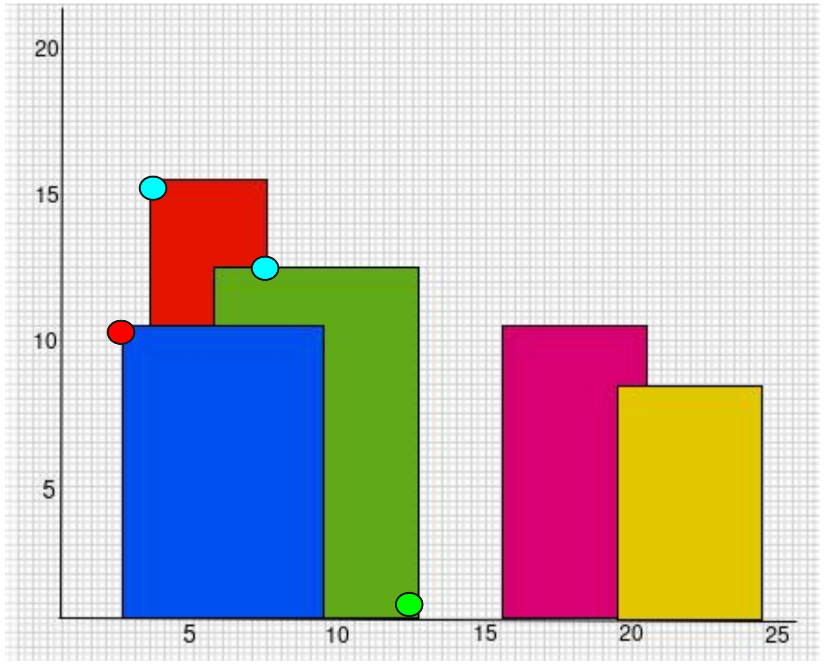
What can be a simple approach ?



- Mark key points for each block
- For each marked point, if buildings with more height overlap with it, change y to maximum height of tallest overlapping buildings
- **Remove redundant points**

TIME COMPLEXITY ?

What can be a simple approach ?



- Mark key points for each block
- For each marked point, if buildings with more height overlap with it, change y to maximum height of tallest overlapping buildings
- Remove redundant points

TIME COMPLEXITY - O(n^2)

Devising a more efficient approach

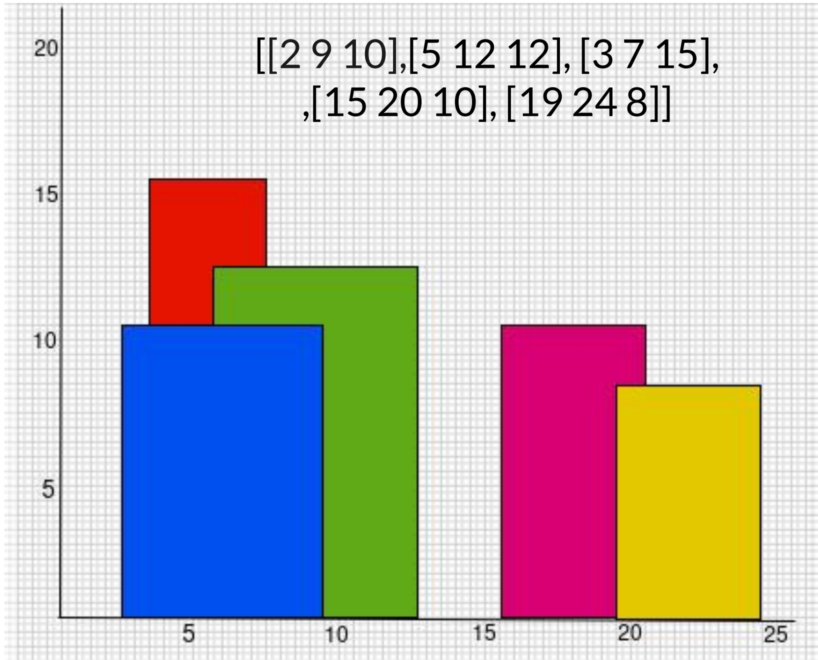
- Divide and conquer ?

Devising a more efficient approach

- Divide and conquer approach ? Merge sort like ?

Devising a more efficient approach

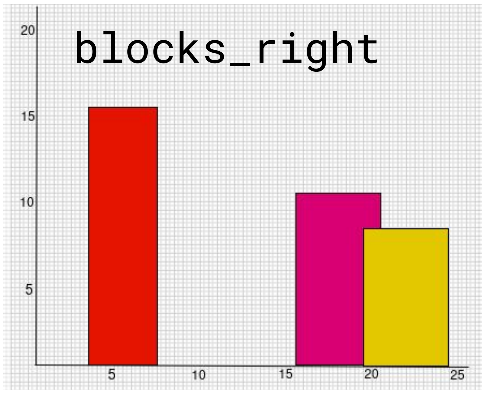
- Divide and conquer approach ? Merge sort like ?

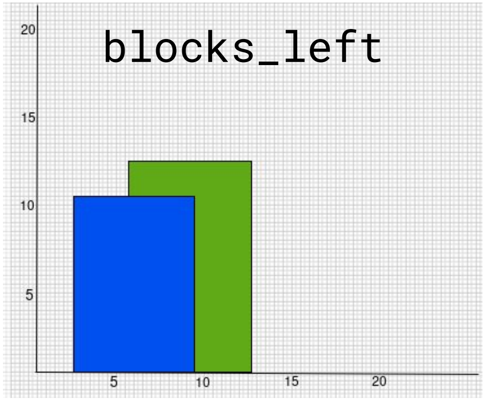


Devising a more efficient approach

- Divide and conquer approach ? Merge sort like ?

```
get_def_vis(set_of_blocks)
 divide into blocks_left,blocks_right
```

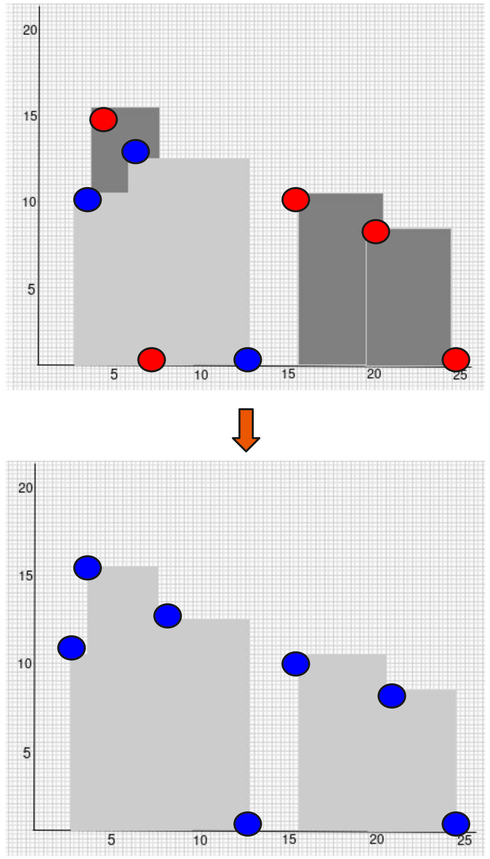




Devising a more efficient approach

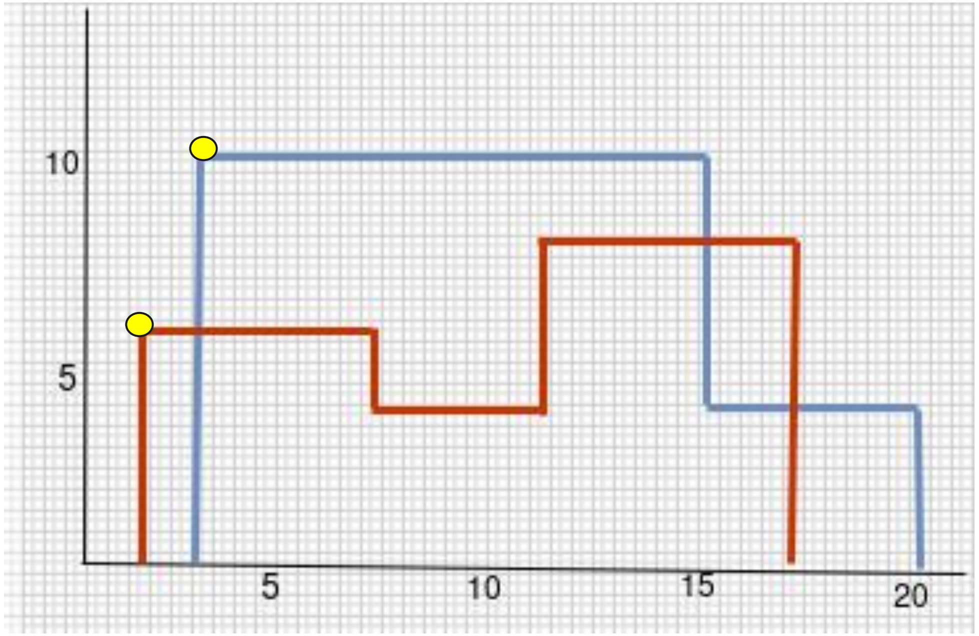
- Divide and conquer approach ? Merge sort like ?

```
get_def_vis(set_of_blocks)
 divide into blocks_left,blocks_right
 left_def_vis=get_def_vis(blocks_left)
 right_def_vis=get_def_vis(blocks_right)
 merged_def_vis=merge(left_def_vis,right_def_vis)
```



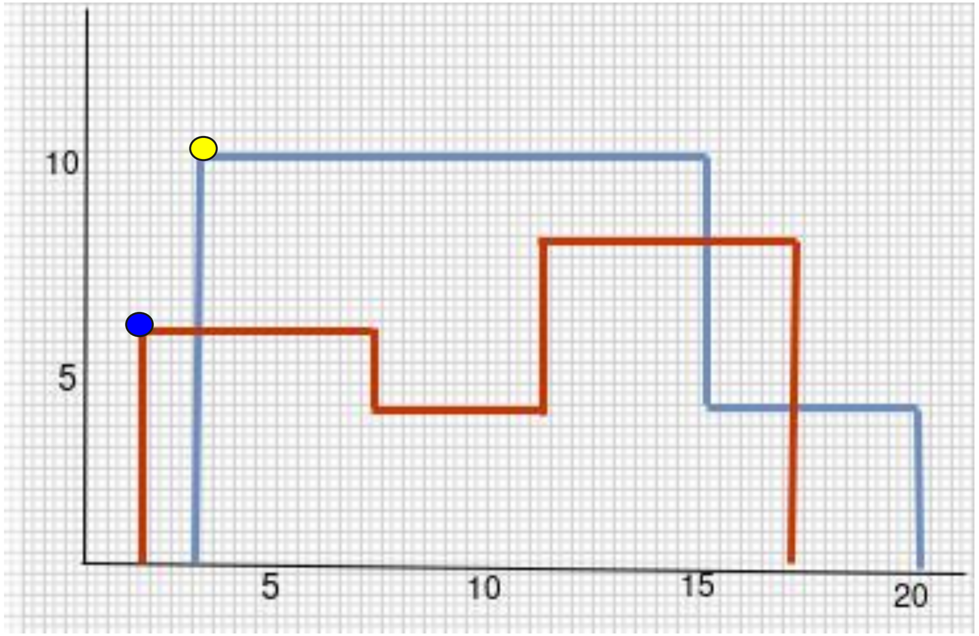
How do we merge the two defective vision output ?

Merge two defective vision output



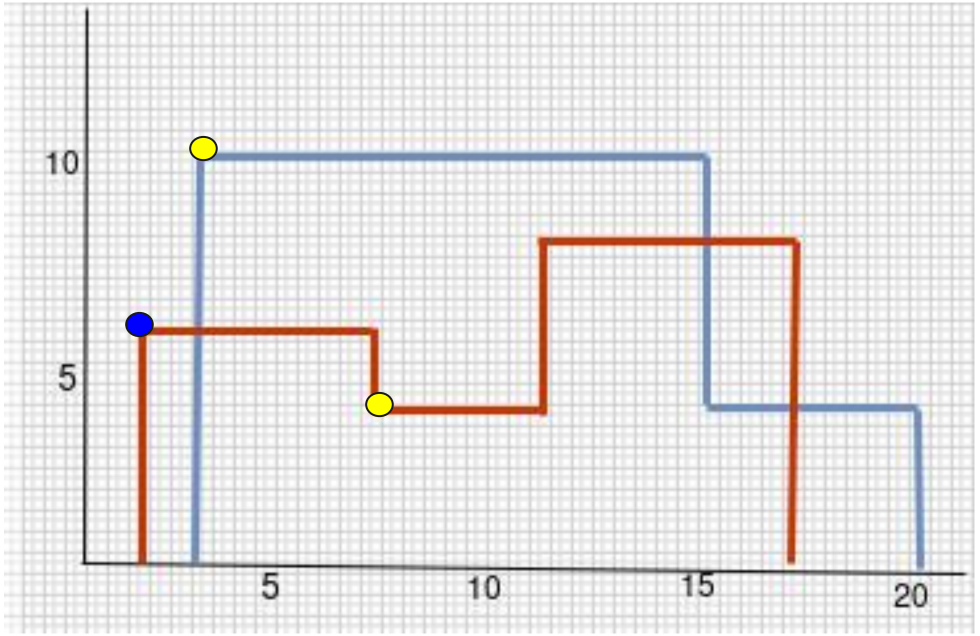
1. Compare both the defective visions outputs, start with leftmost point

Merge two defective vision output



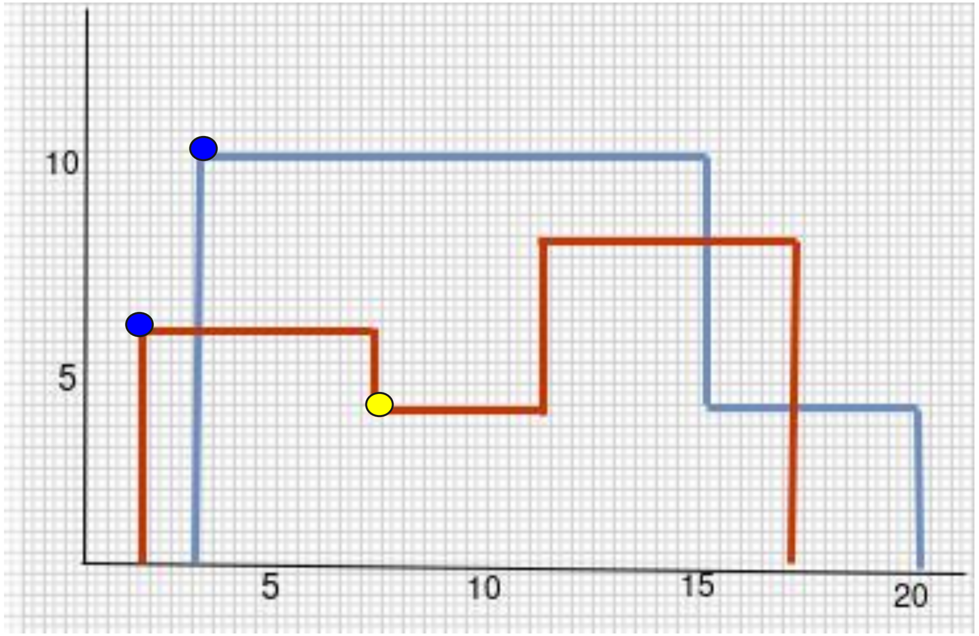
1. Compare both the defective visions outputs, start with leftmost point
2. **Choose the one with less x value**
3. If y value of chosen key point is less than last seen height of other defective vision, update the key point's y to last seen height of other defective vision

Merge two defective vision output



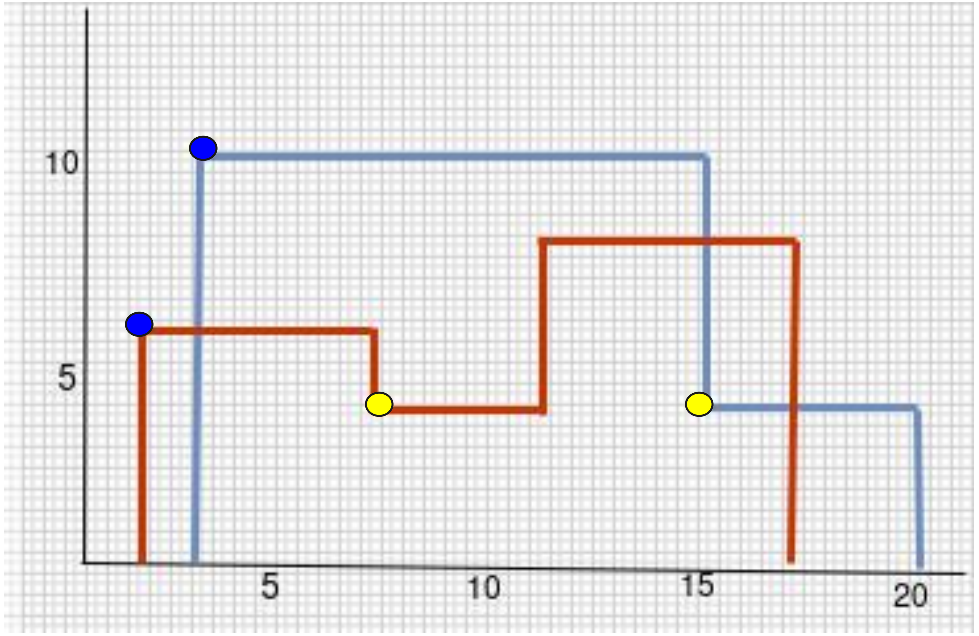
1. Compare both the defective visions outputs, start with leftmost point
2. Choose the one with less x value
3. If y value of chosen key point is less than last seen height of other defective vision, update the key point's y to last seen height of other defective vision
4. **Proceed to next key point of the chosen defective vision output**
5. Repeat above steps till both the defective vision outputs are completed

Merge two defective vision output



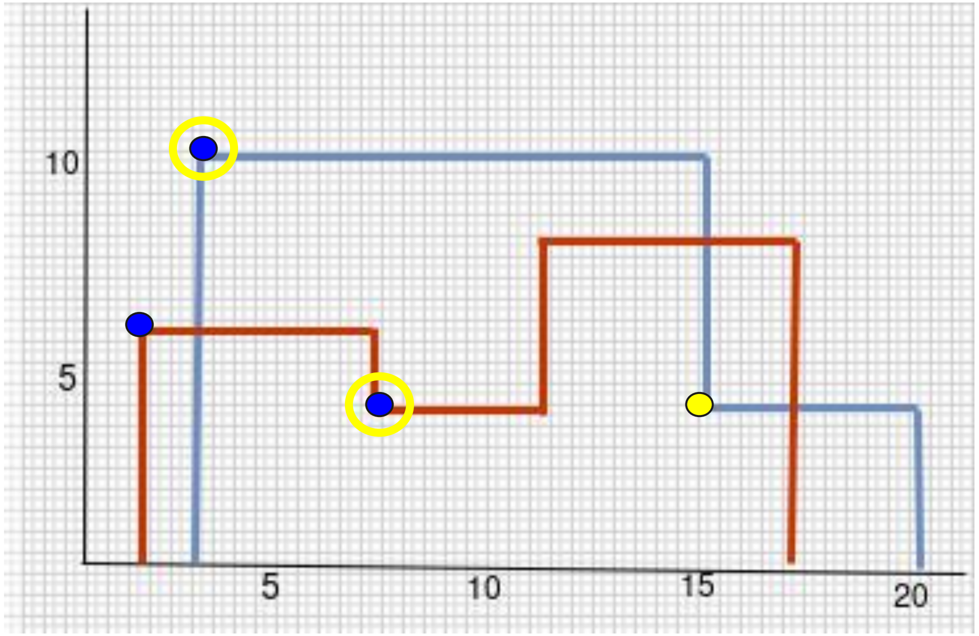
1. Compare both the defective visions outputs, start with leftmost point
2. Choose the one with less x value
3. If y value of chosen key point is less than last seen height of other defective vision, update the key point's y to last seen height of other defective vision

Merge two defective vision output



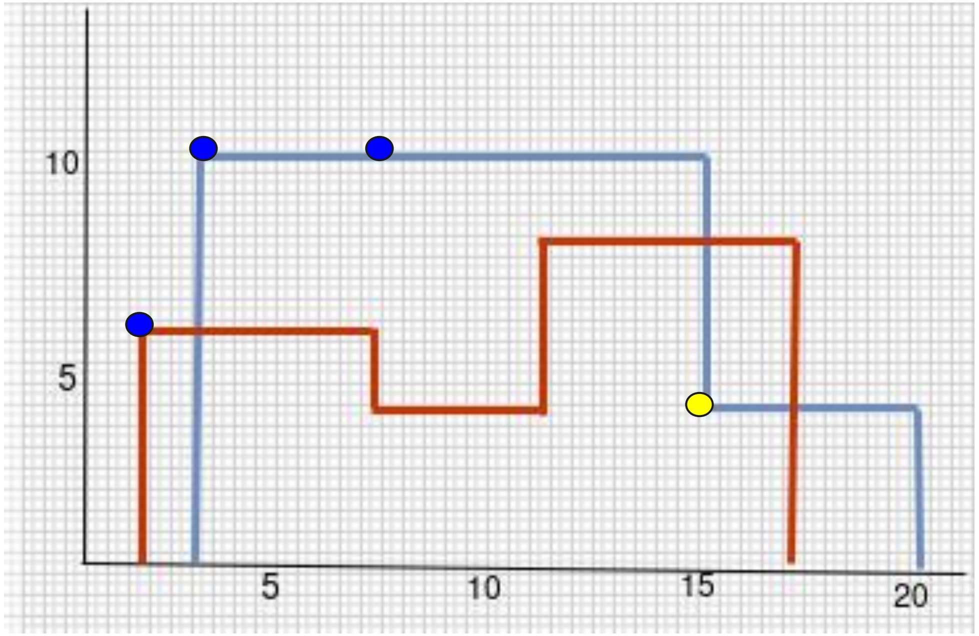
1. Compare both the defective visions outputs, start with leftmost point
2. Choose the one with less x value
3. If y value of chosen key point is less than last seen height of other defective vision, update the key point's y to last seen height of other defective vision
4. **Proceed to next key point of the chosen defective vision output**
5. Repeat above steps till both the defective vision outputs are completed

Merge two defective vision output



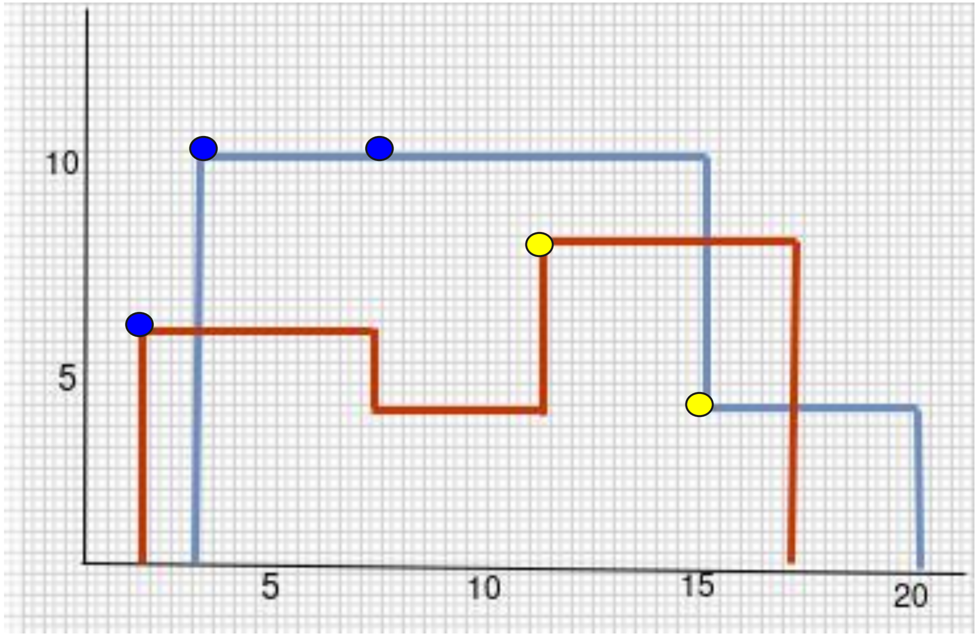
1. Compare both the defective visions outputs, start with leftmost point
2. Choose the one with less x value
3. **If y value of chosen key point is less than last seen height of other defective vision, update the key point's y to last seen height of other defective vision.**

Merge two defective vision output



1. Compare both the defective visions outputs, start with leftmost point
2. Choose the one with less x value
3. **If y value of chosen key point is less than last seen height of other defective vision, update the key point's y to last seen height of other defective vision.**

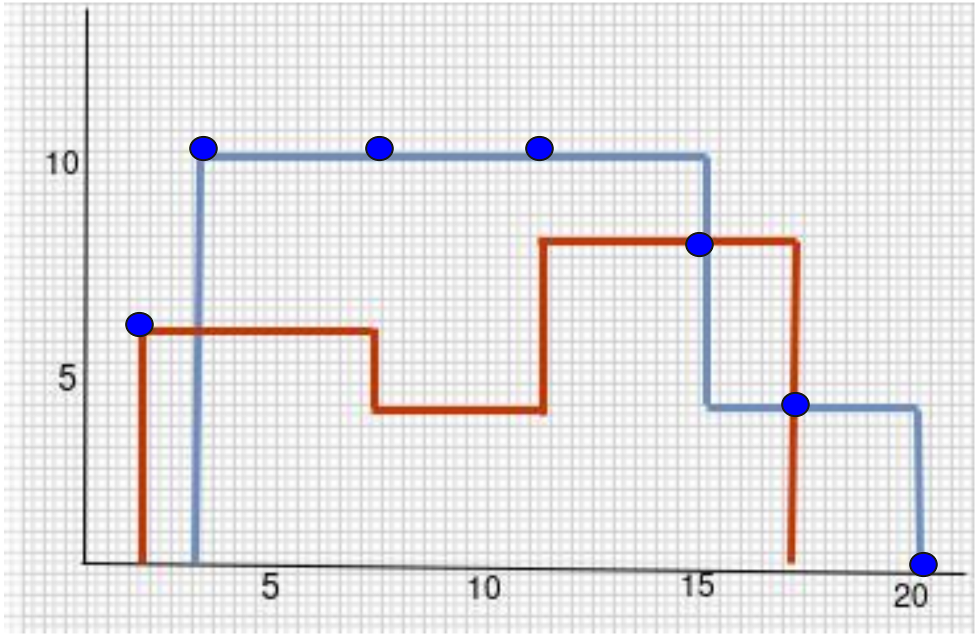
Merge two defective vision output



1. Compare both the defective visions outputs, start with leftmost point
2. Choose the one with less x value
3. If y value of chosen key point is less than last seen height of other defective vision, update the key point's y to last seen height of other defective vision.
4. **Proceed to next key point of the chosen defective vision output**
5. Repeat above steps till both the defective vision outputs are completed

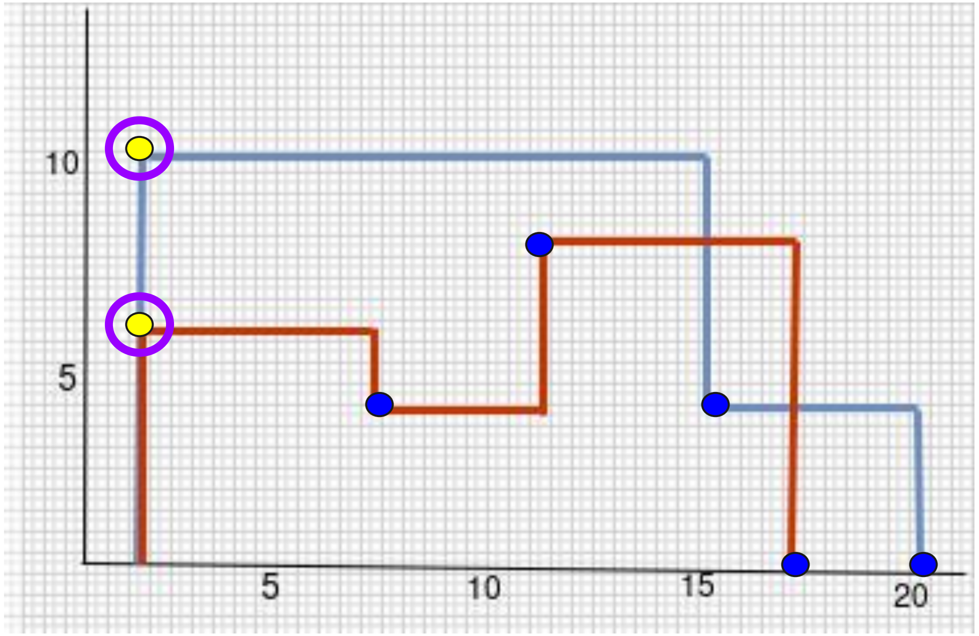
Can you make the final figure when all the points are covered ?

Merge two defective vision output



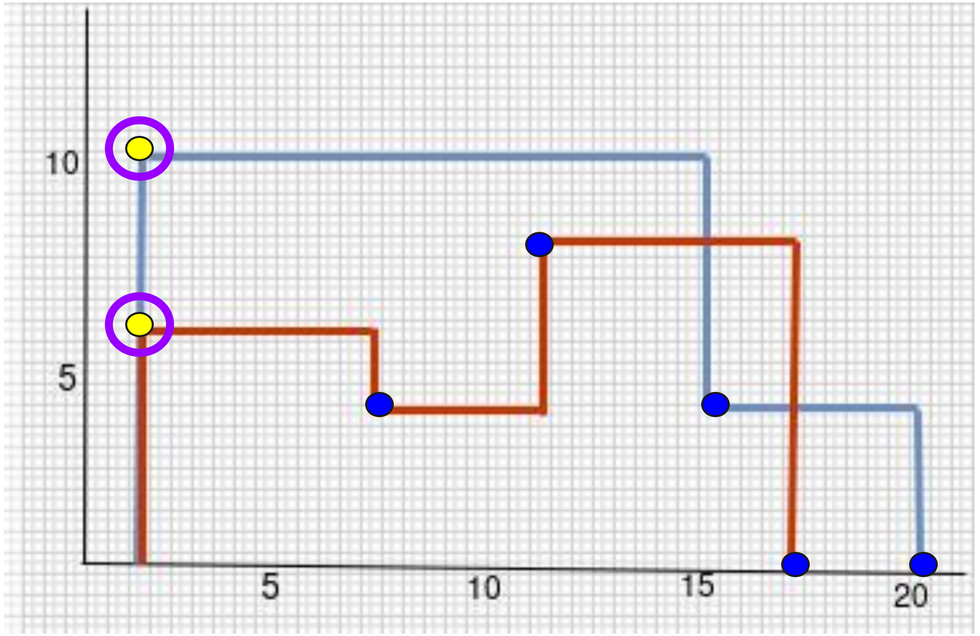
1. Compare both the defective visions outputs, start with leftmost point
2. Choose the one with less x value
3. If y value of chosen key point is less than last seen height of other defective vision, update the key point's y to last seen height of other defective vision.
4. Proceed to next key point of the chosen defective vision output
5. Repeat above steps till both the defective vision outputs are completed
6. **Remove redundant points**

A corner case



What happens if same x value in 2nd step ?

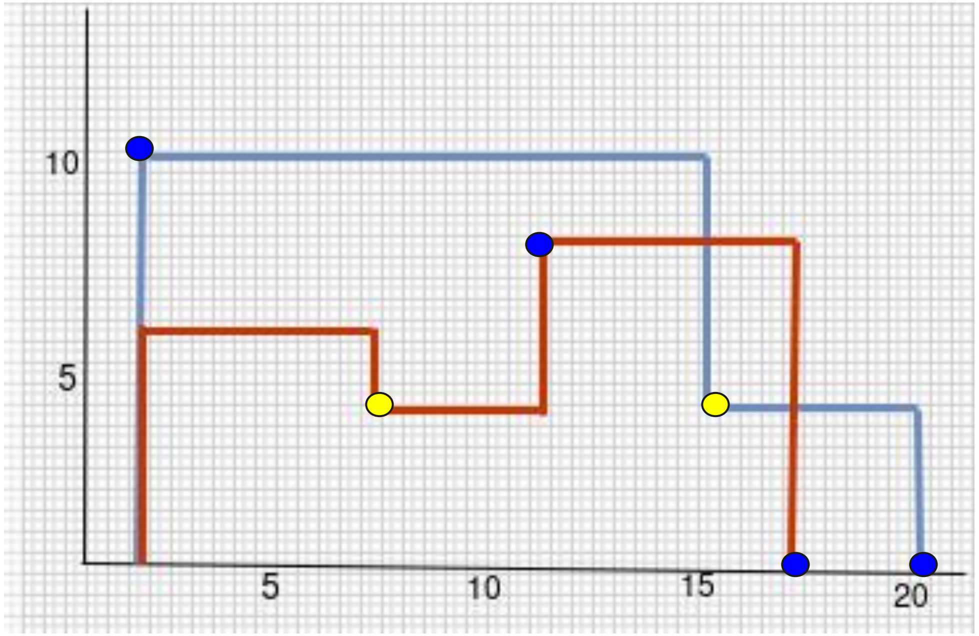
A corner case



What happens if same x value in 2nd step ?

Choose the one with higher y advance both key points and skip 2nd step

A corner case



What happens if same x value in 2nd step ?

Choose the one with higher y advance both key points and skip 2nd step

Takeaways

- Start from a simple example
- Take a more complex example and try to understand what can be the divide step
- Finally think about the algorithm for conquer (here is it simply merging)

Task - try to write the code for this problem

Similar problem:-
https://www.geeksforgeeks.org/the-skyline-problem-using-divide-and-conquer-algorithm/