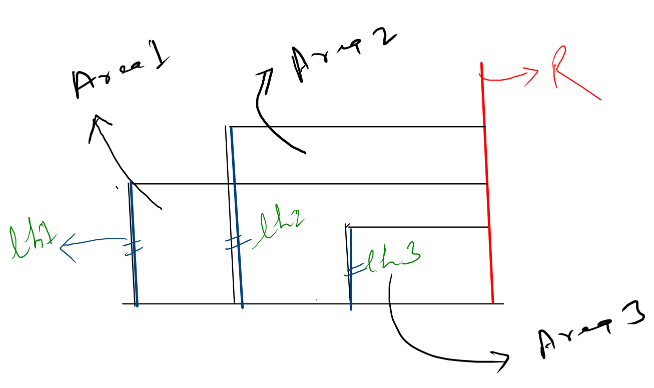
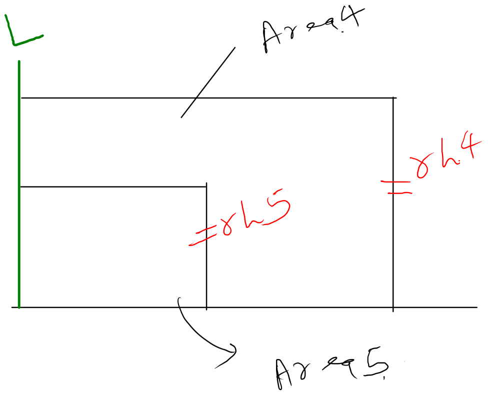
Two Pointer approach to calculate the max/min area between lines

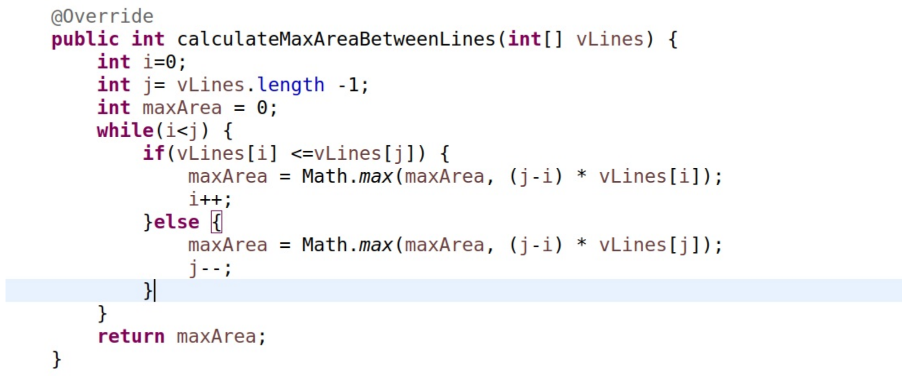




CASE1: When left_pointer line heights(lh1, lh2,lh3) are smaller than right_pointer line height(R) then horizontal line to be drawn for area calcualtion will always be considered with respect to left_pointer heights(lh1,lh2,lh3).
CASE2: When right_pointer line heights(rh4, rh5) are smaller than left_pointer line height(L) then horizontal line to be drawn for area calcualtion will always be considered with respect to right_pointer line heights(rh4, rh5).

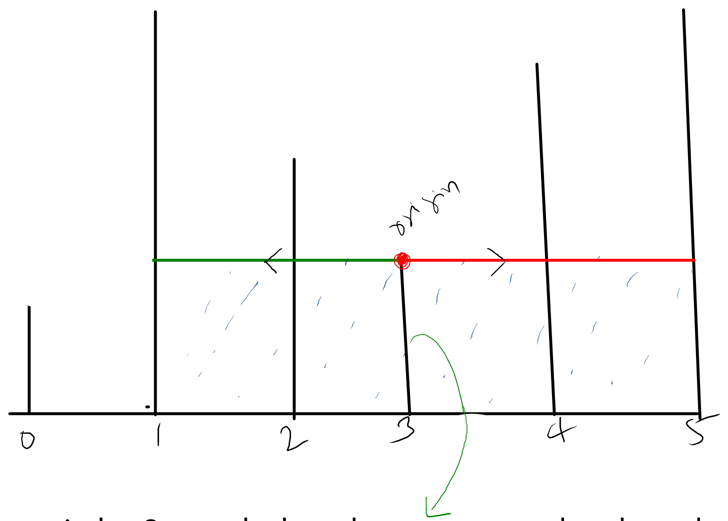
To calculate MaxArea/MinArea we need to compare all the areas(Area1, Area2, Area3, Area4, Area5).

Note : It means for area calculation we always need to pick smaller height as a refrence to draw the horizontal line.



```
@Override
public int calculateMaxAreaBetweenLines(int[] vLines) {
    int i=0;
    int j= vLines.length -1;
    int maxArea = 0;
    while(i<j) {
        if(vLines[i] <=vLines[j]) {
            maxArea = Math.max(maxArea, (j-i) * vLines[i]);
            i++;
        }else {
            maxArea = Math.max(maxArea, (j-i) * vLines[j]);
            j--;
        }
    }
    return maxArea;
}
```

Brute force approach to calculate the max/min area between lines



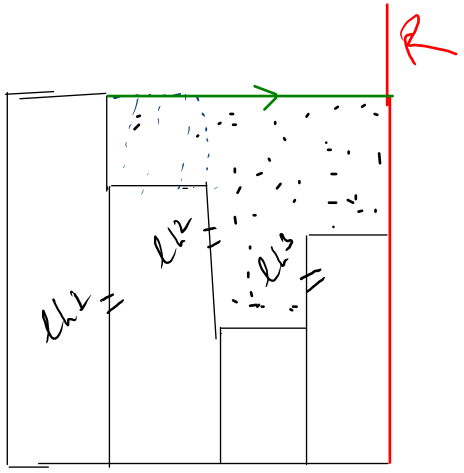
For line present at index 3, to calculate the area we need to draw the horizontal line both left and right side. But in brute force apporach where to find the max-area we compare area between all the possible line pairs C(n,2) = (0-1, 0-2,0-3,..0-5; 1-2,1-3,..,1-5; 2-3,2-4,2-5;3-4,3-5;4-5). The area shown in diagarm remains un-computed.

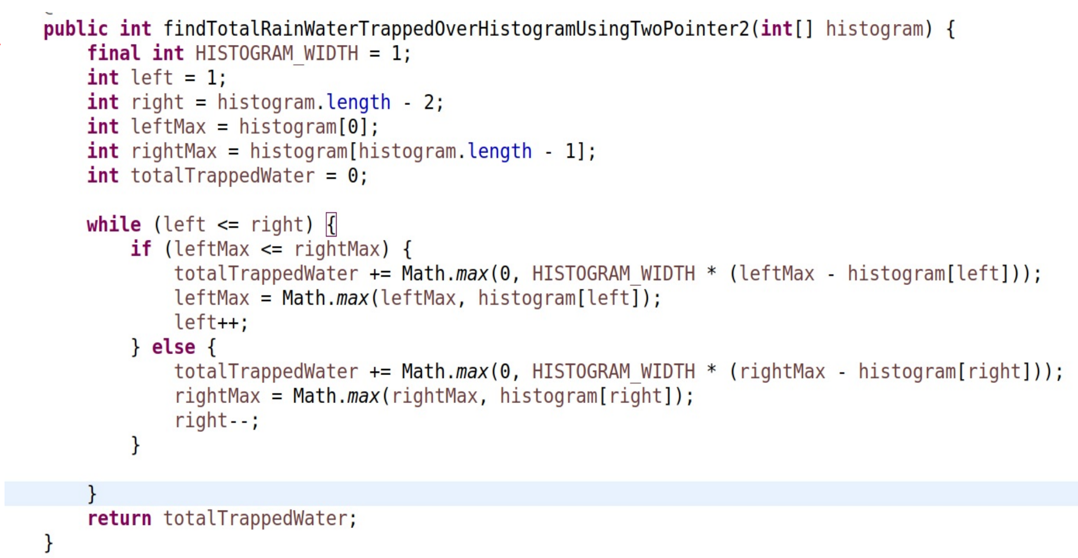
So, then how the brute-force approach gives the correct max-area as answer ?

As we know, to draw the horizontal line we always take smaller height as starting point. So, at 3 we have origin and lines goes in both left(1) and right side(5). Means the other sides of origin point will always be of taller height. This can be used to infer the fact that pair(1,2) in brute force solution will always give area of large size than the area passing through index 3.

This issue of area at index 3 remain uncomputed will be there also in Two Pointer approach. But there will be no impact on max/min area calculation.

Two Pointer approach to calculate the total rain-water trapped over histogram





```java
public int findTotalRainWaterTrappedOverHistogramUsingTwoPointer2(int[] histogram) {
    final int HISTOGRAM_WIDTH = 1;
    int left = 1;
    int right = histogram.length - 2;
    int leftMax = histogram[0];
    int rightMax = histogram[histogram.length - 1];
    int totalTrappedWater = 0;

    while (left <= right) {
        if (leftMax <= rightMax) {
            totalTrappedWater += Math.max(0, HISTOGRAM_WIDTH * (leftMax - histogram[left]));
            leftMax = Math.max(leftMax, histogram[left]);
            left++;
        } else {
            totalTrappedWater += Math.max(0, HISTOGRAM_WIDTH * (rightMax - histogram[right]));
            rightMax = Math.max(rightMax, histogram[right]);
            right--;
        }

    }
    return totalTrappedWater;
}
```

CASE1: When left_pointer histogram heights(lh1, lh2,lh3) are smaller than right_pointer histogram height(R) then horizontal line to be drawn for area calcualtion will always be considered with respect to left_pointer height 'leftMax' = max(lh1,lh2,lh3)). Note leftMax is smaller than right_pointer histogram height(R).

CASE2: Similarly, when right_pointer histogram heights(rh4, rh5) are smaller than left_pointer histogram height(L) then horizontal line to be drawn for area calcualtion will always be considered with respect to right_pointer histogram height 'rightMax'= max(rh4, rh5). Note rightMax is smaller than left_pointer histogram height(L).

To calculate TotalArea area we need to sum up all the areas (area1, area2, area3, area4)

Note : It means for area calculation we always need to pick smaller height(leftMax/rightMax) as a refrence to draw the horizontal line.