Container with Most Water

Here is the pseudo code for the algorithm:

Algorithm 1: *maxArea*(heights)

$left \leftarrow 0$;
$right \leftarrow length(heights) - 1$;
$maxArea \leftarrow 0$;
while $left < right$ **do**
  $area = min(heights[left], heights[right]) * (right - left)$;
  if $area > maxArea$ **then**
    $maxArea \leftarrow area$;
  if *heights[left]* $<$ *heights[right]* **then**
    $left \leftarrow left + 1$;
  else
    $right \leftarrow right - 1$;
return *maxArea*

The solution presented in maxArea.cc probably seems questionable at first sight, below is some intuition that clicked for me:
We are tasked to find the max area formed by the any of the two distinct vertical lines in heights. Observe that the area formed by vertical lines at indeces i, j is which is given by

$$min(height[i], height[j]) * (j - i)$$

Equivalently, the algorithm need to find two indeces [i, j] (i $<$ j) such that the above formula is maximized. Note that the smaller of height[i] and height[j] plays a dominant role in deciding the area.
Suppose height[i] $<$ height[j] (for i $<$ j), then the algorithm will perform i++. Note that by incrementing i by 1, we effectly removed [i, k] (i $<$ k $<$ j) as candidates for the final solution. Observe that

$$\begin{aligned} min(height[i], height[j]) * (j - i) &= height[i] * (j - i) && (height[i] < height[j]) \\ &> height[i] * (k - i) && (k < j) \\ &\geq min(height[i], height[k]) * (k - i) \end{aligned}$$

which means the water in the container with vertical lines at indeces [i, k] is less than the water in the container with the vertical lines at indeces [i, j] which means we can garantee that [i, k] (for i $<$ k $<$ j) are not candidates for the optimal solution.
The case of height[i] $\geq$ height[j] is symmetric to the case above. Hence the algorithm above will always reach the optimal solution.