Efficient Solution:

Approach: Given an array of heights of lines of boundaries of the container, find the maximum water that can be stored in a container. So start with the first and last element and check the amount of water that can be contained and store that container. Now the question arises is there any better boundaries or lines that can contain maximum water. So there is a clever way to find that. Initially, there are two indices the first and last index pointing to the first and the last element (which acts as boundaries of the container), if the value first index is less than the last index increase the first index else decrease the last index. As the increase in width is constant, by following the above process the optimal answer can be reached.

Algorithm:

1. Keep two index, *first = 0* and *last = n-1* and a value max_area that stores the maximum area.
2. Run a loop until first is less than the last.
3. Update the max_area with maximum of max_area and *min(array[first] , array[last])*(last-first)*
4. if the value at array[first] is greater tha array[last] then update last as last – 1 else update first as first + 1
5. Print the maximum area.

Complexity Analysis:

- **Time Compelxity:** O(n).
  As only one traversal of the array is required, so time complexity is O(n).
- **Space Complexity:** O(1).
  No extra space is required, so space complexity is constant.