### Container With Most Water.

### Statement ….

Given **N** non-negative integers a1,a2,....an where each represents a point at coordinate (i, ai). **N**vertical lines are drawn such that the two endpoints of line**i**is at (i, ai) and (i,0). Find two lines, which together with x-axis forms a container, such that it contains the most water.

Note : In Case of single verticle line it will not be able to hold water.

**Example 1:**

**Input:**

N = 4

a[] = {1,5,4,3}

**Output:** 6

**Explanation:** 5 and 3 are distance 2 apart.

So the size of the base = 2. Height of

container = min(5, 3) = 3. So total area

= 3 \* 2 = 6.

**Example 2:**

**Input:**

N = 5

a[] = {3,1,2,4,5}

**Output:** 12

**Explanation:** 5 and 3 are distance 4 apart.

So the size of the base = 4. Height of

container = min(5, 3) = 3. So total area

= 4 \* 3 = 12.

### Java Code

//{ Driver Code Starts

//Initial Template for Java

import java.io.\*;

import java.util.\*;

import java.lang.\*;

class CodingMaxima {

public static void main (String[] args) throws IOException{

BufferedReader br = new BufferedReader(new InputStreamReader(System.in));

int t = Integer.parseInt(br.readLine().trim()); //Inputting the testcases

while(t-->0){

int n = Integer.parseInt(br.readLine()); // input size of array

int arr[] = new int[n];

String inputLine[] = br.readLine().trim().split("\\s+");

for(int i=0; i<n; i++){

arr[i] = Integer.parseInt(inputLine[i]); // input elements of array

}

Solve T = new Solve();

long ans = T.maxArea(arr,n);

System.out.println(ans);

}

}

}

// } Driver Code Ends

//User function Template for Java

class Solve{

long maxArea(int A[], int len){

long area=0;

int i=0;

int j=len-1;

while(i<j){

area=Math.max(area, (j-i)\*Math.min(A[i], A[j]));

if(A[i]<A[j])

i++;

else

j--;

}

return area;

}

}