A non-empty array A consisting of N integers is given. The *product* of triplet (P, Q, R) equates to A[P] \* A[Q] \* A[R] (0 ≤ P < Q < R < N).

For example, array A such that:

A[0] = -3 A[1] = 1 A[2] = 2 A[3] = -2 A[4] = 5 A[5] = 6

contains the following example triplets:

* (0, 1, 2), product is −3 \* 1 \* 2 = −6
* (1, 2, 4), product is 1 \* 2 \* 5 = 10
* (2, 4, 5), product is 2 \* 5 \* 6 = 60

Your goal is to find the maximal product of any triplet.

Write a function:

def solution(A)

that, given a non-empty array A, returns the value of the maximal product of any triplet.

For example, given array A such that:

A[0] = -3 A[1] = 1 A[2] = 2 A[3] = -2 A[4] = 5 A[5] = 6

the function should return 60, as the product of triplet (2, 4, 5) is maximal.

Write an **efficient** algorithm for the following assumptions:

* N is an integer within the range [3..100,000];
* each element of array A is an integer within the range [−1,000..1,000].

def solution(A):

A.sort()

tot1 = A[0] \* A[1] \* A[-1]

for i in range(0, len(A)-2 ):

if tot1 < A[i] \* A[i+1] \* A[i+2] :

tot1 = A[i] \* A[i+1] \* A[i+2]

return tot1

A = [-5, 5, -5, 4]

car1 = solution(A)

print(car1)

or

Java

import java.util.\*;

class Solution {

public int solution(int[] A) {

Arrays.sort(A);

int product = 1;

if(A.length == 0) return 0;

for(int i= A.length -1 ; i > A.length - 4 ; i-- ){

product \*= A[i];

}

if(A[0] < 0 && A[1] < 0) {

product = (A[0]\*A[1]\*A[A.length-1]) > product ?(A[0]\*A[1]\*A[A.length-1]):product;

}

return product;

}

}